O'ZBEKISTON RESPUBLIKASI AXBOROT TEXNOLOGIYALARI
VA KOMMUNIKATSIYALARINI RIVOJLANTIRISH VAZIRLIGI
TOSHKENT AXBOROT TEXNOLOGIYALARI UNIVERSITETI
URGANCH FILIALI

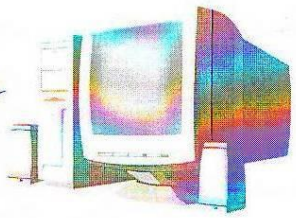
KOMPYUTER INJINIRINGI FAKULTETI
"TELEKOMMUNIKATSIYA INJINIRINGI" KAFEDRASI

"Mikroprosessor" fanidan yozgan

Mavzu: ATmega 8 mikrokontroller bazasida 8x48 matritsali yuguruvchi satr logixasini ishlab chiqish

KURS ISHI

88 bala
30.05.16



Topshirdi: 931-13 guruhi talabasi
Ruzmetov A.Yu.

Qabul qildi: Setmetov N.U.

Urganch – 2016

Bajardi:	Ruzmetov .A.Y	Bet	
Tekshirdi:	Setmetov N.U.		

MUNDARIJA.

Kirish…….

1. Atmel corporation firmasining AVR mirokontrollerlari asosida qurilmalarni ishlab chiqish…….....
2. AVR Atmel Studio 7.0 dasturida Atmega8 mikrokontrolleriga dastur kodini yozish ……...…….
3. ISIS Proteus virtual dasturida 8x48 matritsali yuguruvchi satr loyihasini ishlab chiqish…….……...
4. Dastur kodi…….……...

Xulosa……..……..

Foydalanilgan adabiyotlar ro’yxati……..……..

Bajardi:	Ruzmetov .A.Y	Bet	
Tekshirdi:	Setmetov N.U.		

Kirish

Bu o’quv jarayoniga elektron qurilmalarni professonal loyihalovchisining PROTEUS nomli avtomatlashtirilgan ish o’rni tizimini kiritilganligi munosabati bilan ana shu tizim asosida talabalar zamonaviy mikrokontrollerlar va ularni dasturlash vositalari bilan tanishtirish imkoniyatiga hamda loyihalash jarayonida kompyuter va zamonaviy orgtexnika vositalaridan foydalanishning amaliy ko’nikmalariga ega bo’ladilar.

ISIS dasturiy majmuasida turli mikrokontrollerlar uchun “C/C++” tilida tayyorlangan dasturni tarjima qilib beruvchi translyatorlar mavjud. Ushbu traslyatorlar yordamida dastlabki dastur faylidan mikrokontroller uchun ishchi dasturni tayyorlab olamiz. Buning uchun ISIS ishchi oynasida Sourse menyusi tarkibidagi Add/Remove Sourse files menyusi bilan loyihadagi mikrokontrollerga yuklanishi lozim bo’lgan dastur faylini va kerakli mikrokontrollerni xamda translyatorni tanlash lozim. SHundan so’ng translyatsiya qilinsa va dasturda hech qanday xato bo’lmasa, translyatsiya natijasida hosil bo’lgan (*.hex) fayl loyihadagi yuklanadi. Agar loyiha dasturi S tilida tayyorlangan bo’lsa va bu dastur S translyatori bilan translyatsiya qilinsa va dasturda hech qanday xato bo’lmasa, translyatsiya natijasida hosil bo’lgan (*.hex) fayl loyihadagi mikrokontrollerning dastur xotirasiga aloxida yuklanishi lozim.

Dastlabki dastur matnini tayyorlashda tanlangan Atmega8 mikrokontrollerining komandalar tizimidan hamda ushbu loyihada ishlatilgan DS1307 mikrokontrolleri komandalaridan foydalanamiz. Bu komandalar haqidagi ma’lumotlar mos ravishda [1],[2] hamda [3] da to’liq keltirilgan.

AVR mikrokontrollerlari Atmel korporotsiyasining nisbatan kenja mahsulotlaridan biridir. Ushbu tur mikrokontrollerlar kundan kun rivojlanib bormoqda, yangidan yangi kristallar kashf qilinmoqda, mavjud mikrosxema chiplarini qo’llab quvvatlaydigan dasturiy ta’minotlar takomillashtirilib kengaymoqda. Atmel firmasining AVR mikrokontrollerlari rasmiy katalogi nashri 1997 yil mayida birinchi marta e’lon qilingan edi. Ushbu katalogdan atiga to’rtta dastlabki AVR "classic" oilasiga mansub AT90S mikrokontrollerlari joy olgan edi. Navbatdagi kengaytirilgan ikkinchi katalogi nashri 1999 yil avgustda chiqqan bo’lib, unda AVR mikrokontrollerlarining uchta oilasi, ya’ni "tiny", "classic" va "mega" oilalari joy olgan edi. Ana shundan beri Atmel firmasi mikrokontrollerlarining yangi katalogi jurnal ko’rinishida nashr qilinmagan. Ammo, Atmel firmasi mikrokontrollerlarning elektron ko’rinishdagi (Data Sheet) texnik ma’lumotlarini doimo internetdagi

www.atmel.com/atmel/products/prod23.htm. informatsion saytida e’lon qilib boradi.

Hozirda barcha AVR mikrokontrollerlari dastur yozishga mo’ljallangan Flash-xotirasiga ega. Ushbu xotiraga oddiy programmator qurilmasi yoki mikrokontroller SPI-interfeysi orqali to’g’ridan –to’g’ri murojaat qilish mumkin. Mikrokontroller Flash-xotirasiga 1000 martagacha dasturlarni qayta yozish mumkin. 2001-2002 yillarda ishlab chiqarilgan "AVR-mega" oilasi mikrokontrollerlari o’z-o’zini dasturlash hususiyatiga ega. Bu mikrokontroller kristali mustaqil ravishda hech qanday tashqi programmator qurilmalarisiz Flash-xotirasida yozilgan dasturni o’zgartirish mumkinligini anglatadi. Ya’ni yangi AVR mikrokontrollerlari o’ziga yozilgan dastrurni va ishlash algoritmlarini o’zgartirishi hamda keyinchalik ushbu o’zgargan algoritm yoki dastur bo’yicha ishlashi mumkin. Masalan, bir qancha ishchi dasturlar yozish va saqlash hamda zaruriyat yuzasidan almashtirish mumkin.

Shuningdek barcha AVR mikrokontrollerlari energiyaga bog’liq bo’lmagan EEPROM-xotiraga ega. Ushbu tur xotiraga mikrokontroller o’ziga yozilgan dastur bajarilayotganda murojaat qilishi mumkin. SHuningdek vaqtinchalik saqlanuvchi ma’lumotlarni, turli xil o’zgarmaslarni, qayta kodlash tizimi jadvallarini, kalibrovkalovchi koeffitsientlarni va shunga o’xshash ma’lumotlarni saqlashga qulay. EEPROM-xotira ham SPI interfeysi yoki programmator orqali yuklanishi mumkin. Mikrokontroller EEPROM -xotirasiga 1000 martagacha ma’lumotlarni qayta yozish mumkin. Ikkita FLASH Lock Bits va EEPROM Lock Bits konfiguriyatsiya bitlari yordamida mos holda FLASH-xotiradagi dastur hamda EEPROM –xotiradagi ma’lumotlarni ruhsat berilmagan o’qishdan himoya qilish mumkin. Ichki operativ SRAM xotira barcha AVR mikrokontrollerlari "classic" va "mega" oilasida mavjud. "tiny" oilasidan faqatgina ATtiny26/L mikrokontrollerida ichki operativ SRAM xotira ko’zda tutilgan. Ba’zi mikrokontrollerlarga ma’lumotlarni yozish uchun 64 Kbaytgacha tashqi xotira ulanish imkoniyati mavjud.

1) Atmel corporation firmasining AVR mirokontrollerlari asosida qurilmalarni ishlab chiqish

Atmel kompaniyasining AVR mikrokontrollelari barcha ilg'or texnologiyalarni mujassamlantirganligi, foydalanuvchi tomonidan elektr usulida programmalashtiriluvchi ППЗУ, minimal manba iste'moli, yuqori tezlikli, rivojlangan RISC-arxitekturali, funktsional tugallanganligi va minimal razmerlari bilan ajralib turadi.

Atmel Corporation kompaniyasi nazorat va boshqarish tizimlari uchun elektron komponentlar ishlab ishlab chiqarishga ixtisoslashgan bo'lib, hozirda MCS-51, ARM, AVR, AVR32 mikrokontrollerlarini ishlab chiqarish bo'yicha etakchilik qilib kelmoqda. Asosiy ishlab chiqarilayotgan mahsulotlari asosan quyidagilardan tashkil topgan:

8 – razryadli universal mikrokontrollerlar (MCS-51, ARM, AVR, AVR32);

Ixtisoslashtirilgan xotira mikrosxemalari;

Foydalanishni cheklovchi qurilmalar (KeyLock);

Loyihalashning texnik va dasturiy vosityalari (CodeVisionAVR, AVR Studio, WinAVR, AVR ISP).

Bugunga kelib, AVR mikrokontrollerlar standart oilasini uch guruhga ajratish mumkin:

- ***tinyAVR (ATtinyxxx)***:
  - Flash-xotira - 16 Kbaytgacha; tashqi opreativ xotira SRAM - 512 baytgacha; EEPROM –xotira 512 baytgacha;
  - Kirish-chiqish liniyalar soni 4-18 (mikrosxema chiqimlarining umumiy soni 6-32);
  - periferiya qurilmalarining cheklangan nabori.
- ***megaAVR (ATmegaxxx):***
  - Flash-xotira - 256 Kbaytgacha; tashqi opreativ xotira SRAM - 16 Kbaytgacha; EEPROM –xotira 4 Kbaytgacha;
  - Kirish-chiqish liniyalar soni 23-86 (mikrosxema chiqimlarining umumiy soni 28-100);
  - Apparatli umnojitel;
  - Kengaytirilgan periferiya qurilmalari va komandalar tizimi.
- ***XMEGA AVR (ATxmegaxxx):***
  - Flash-xotira - 384 Kbaytgacha; tashqi opreativ xotira SRAM - 32 Kbaytgacha; EEPROM –xotira 4 Kbaytgacha;
  - To'rt kanalli DMA-kontrolleri;
  - Hodisalarni qayta ishlovchi innovatsion tizim.

AVR mikrokontrollelarini ishlatilish sohalari ko'p qirrali. "Tiny" oilasiga mansub mikrokontrollerlar avtomobillarlarda turli xil intelluktual datchiklar ko'rinishida, o'yinchoqlar, kompyuterning bosh platasida, mobil telefonlarida

tormoqga ulanish kontrollerlari, zaryadlash qurilmalari, tutun va olov detektorlari, maishiy texnikada, turli xil infraqizil nurli masofaviy boshqaruv pultlari sifatida ishlatlib kelinmoqda.

“Classic” oilasiga mansub mikrokontrollerlar turli xil modemlar, zamonaviy zaryadlash qurilmalari, Smart Cards sinfi mahsulotlari va ularni o’qish qurilmalari, avtomobillarni turgan joyini aniqlash uchun navigatsiya tizimlarida, murakkab maishiy texnikada, masofaviy boshqarish pultlarida, tarmoq kartalarida, kompyuter bosh platalarida, yangi avlod mobil telefonlarida shuningdek turli xil ishlab chiqarishdagi nazorat va boshqarish tizimlarida islatilmoqda.

“Mega” oilasiga mansub mikrokontrollerlar analog (NMT, ETACS, AMPS) va raqamli (GSM, CDMA) mobil telefonlar, printerlar va kontroller kalitlar, faksimil aloqa apparatlari kontrollerlarida va kserokslar, zamonaviy vinchester disklar kontrollerlari, CD-ROM va boshqa qurilmalarda ishlatiladi.

Hozirda barcha AVR mikrokontrollerlari dastur yozishga mo’ljallangan Flash-xotirasiga ega. Ushbu xotiraga oddiy programmator qurilmasi yoki mikrokontroller SPI-interfeysi orqali to’g’ridan –to’g’ri murojaat qilish mumkin. Mikrokontroller Flash-xotirasiga 1000 martagacha dasturlarni qayta yozish mumkin. 2001-2002 yillarda ishlab chiqarilgan “Mega” oilasi mikrokontrollerlari o’z-o’zini dasturlash hususiyatiga ega. Bu mikrokontroller kristali mustaqil ravishda hech qanday tashqi programmator qurilmalarisiz Flash-xotirasida yozilgan dasturni o’zgartirish mumkinligini anglatadi. Ya’ni yangi AVR mikrokontrollerlari o’ziga yozilgan dastrurni va ishlash algoritmlarini o’zgartirishi hamda keyinchalik ushbu o’zgargan algoritm yoki dastur bo’yicha ishlashi mumkin. Masalan, bir qancha ishchi dasturlar yozish va saqlash hamda zaruriyat yuzasidan almashtirish mumkin.

Shuningdek barcha AVR mikrokontrollerlari energiyaga bog’liq bo’lmagan EEPROM-xotiraga ega. Ushbu tur xotiraga mikrokontroller o’ziga yozilgan dastur bajarilayotganda murojaat qilishi mumkin. SHuningdek vaqtinchalik saqlanuvchi ma’lumotlarni, turli xil o’zgarmaslarni, qayta kodlash tizimi jadvallarini, kalibrovkalovchi koeffitsientlarni va shunga o’xshash ma’lumotlarni saqlashga qulay. EEPROM-xotira ham SPI interfeysi yoki programmator orqali yuklanishi mumkin. Mikrokontroller EEPROM -xotirasiga 1000 martagacha ma’lumotlarni qayta yozish mumkin. Ikkita FLASH Lock Bits va EEPROM Lock Bits konfiguriyatsiya bitlari yordamida mos holda FLASH-xotiradagi dastur hamda EEPROM –xotiradagi ma’lumotlarni ruhsat berilmagan o’qishdan himoya qilish mumkin. Ichki operativ SRAM xotira barcha AVR mikrokontrollerlari “classic” va “mega” oilasida mavjud. “tiny” oilasidan faqatgina ATtiny26/L mikrokontrollerida ichki operativ SRAM xotira ko’zda tutilgan. Ba’zi mikrokontrollerlarga ma’lumotlarni yozish uchun 64 Kbaytgacha tashqi xotira ulanish imkoniyati mavjud.

Garvard arxitekturali mikrokontrollerlarda komanda bitta tsiklda o’qiladi (hamma komandalar uzunligi bir hil – 16 razryadli). Bu vaqtda axborot xotirasidan o’qish yoki yozish mumkin, chunki bu shina alohida.

Standart oilalar asosida aniq bir masalaga moslashtirilgan mikrokontrollerlar ishlab chiqariladi:

- USB, CAN LCD interfeyslar o’rnatilgan mikrokontroller;
- radioqabullovchi va radiouzatuvchi qurilmalar o’rnatilgan mikrokontrollerlar —ATAxxxx, ATAMxxx seriyali;
- dvigatellarni boshqarish uchun — AT90PWMxxxx seriyali;
- avtomobillar elektron sxemalari uchun;
- yoritish texnika uchun.

AVR mikrokontrollerlari strukturasi. Mikrokontrollerning har bir qismi quyidagi uch guruhga tegishli bo’lishi mumkin:

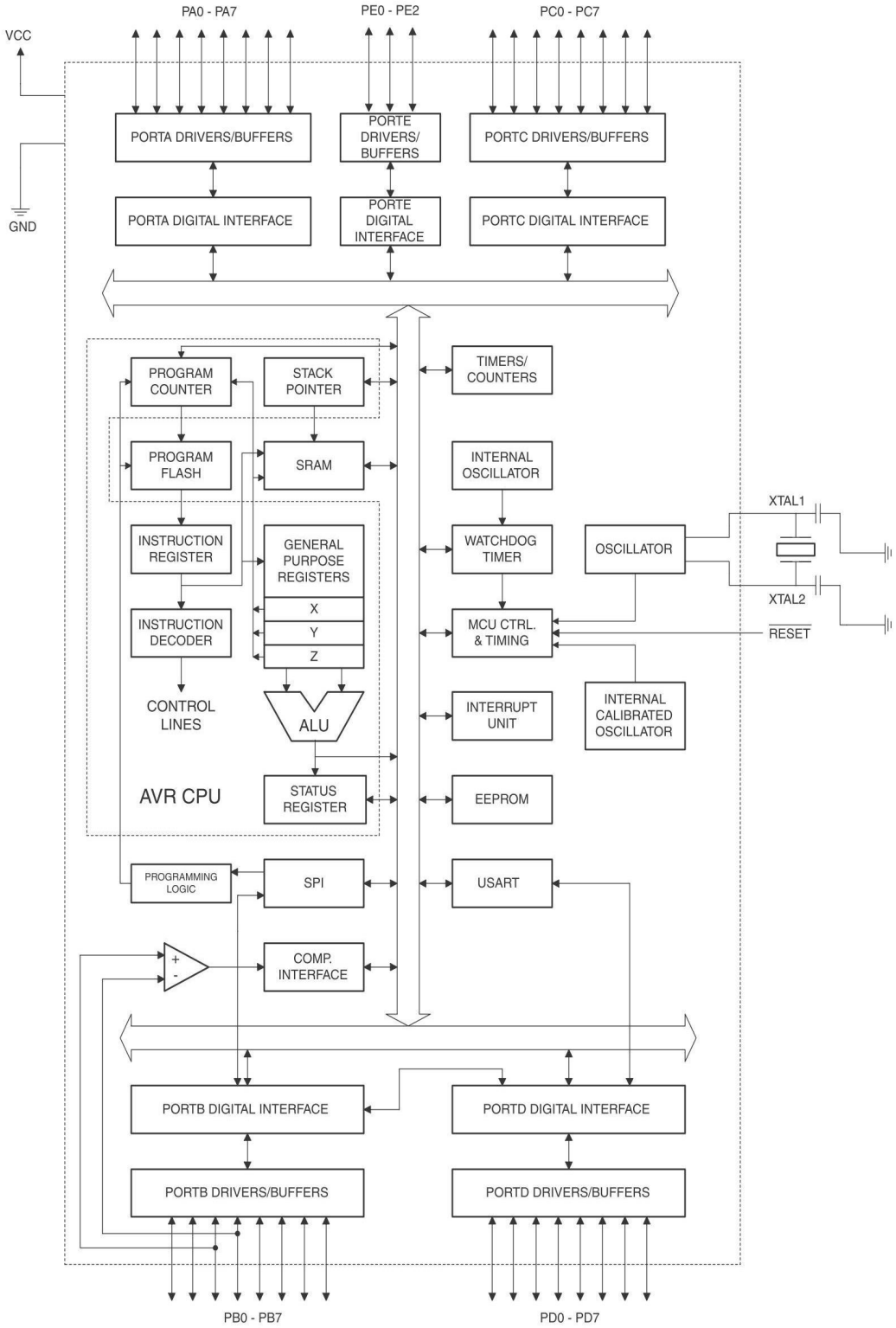
1. Mikrokontroller yadrosi;
2. Periferiya moduli;
3. Mikrokontrollerning maxsus qismi.

Mikrokontroller yadrosi. Yadro asosiy qism bo’lib, u mikrokontrollerni ishlashga majbur qiladi. Bu guruh tarkibiga quyidagilar kiradi (1.14.-rasm):

1. Takt generatori
2. Dastlabki holatga qaytish sxemasi (Логика*сброса)
3. Markaziy protsessor (CPU)
4. Arifmetik-logik qurilma (ALU)
5. Komandalar xotira qurilmasi
6. Axborot xotira qurilmasi
7. Uzilishlar tizimi (Прерывания)
8. Komandalar tizimi.

AVR protsessorlari registrli faylga birlashtirilgan 8-bitli 32 ta registrdan iborat. Ideal RISC yadrosi arxitekturasidan farqli o’laroq ushbu registrlar mutlaq ortogonal emas:

- Ba’zi komandalar faqat r16…r31 registrlarida ishlaydi. ANDI/CBR, ORI/SBR, CPI, LDI, LDS(16-bit), STS(16-bit), SUBI, SBCI, shunikdek SER va MULS operandlari bilan bevosita ishlaydigan komandalar shular jumlasidandir;
- 16-bit qiymatli oshiruvchi va kamayuvchi komandalar (ADIW, SBIW) operandlari bevosita faqat quyidagi r25:r24, r27:r26 (X), r29:r28 (Y), yoki r31:r30 (Z) registrlari juftlaridan birida ishlashga mo’ljallangan;



1.2-rasm AVR mikrokontrollerlarining umumiy struktura sxemasi

- Registrlar juftini nusxalash komandalari faqat toq sondagi (r1:r0, r3:r2, …, r31:r30) qo’shni registr juftlarida ishlaydi;
- Ko’paytirish natijasi (ko’paytirish moduli mavjud modellarda) doimo

Bajardi:	Ruzmetov .A.Y	Bet	
Tekshirdi:	Setmetov N.U.		

r1:r0 registrlar juftligida joylashadi. SHuningdek, faqat ushbu juftliklar o’z-o’zini dasturlash komandalari uchun operandlari sifatida ishlatiladi;

- Ko’paytirish komandalarining ba’zi variantlari faqat r16…r23 (FMUL, FMULS, FMULSU, MULSU) diapazonidagi registrlarni argument sifatida qabul qiladi.

Komandalar tizimi. AVR mikrokontrollerlari komandalar tizimi instruktsiyalari yuksak rivojlangan va turli xil modellarda turlicha bo’lib, 90 dan 133 tagacha etadi.

Ko’pchilik komandalar bitta xotira yacheykasini (16 bit) egallaydi. Ko’pchilik komandalar bitta takt mobaynida bajariladi.

AVR mikrokontrollerlari komandalar to’plamini bir nechta guruhlarga ajratish mumkin:

- mantiqiy amal bajaruvchi komandalari;
- arifmetik amal bajaruvchi va siljitish komandalari;
- bitlar ustida amal bajaruvchi komandalari;
- ma’lumotlarni jo’natish komandalari;
- uzatishlarni boshqarish komandalari;
- tizimni boshqarish komandalari.

Periferiya modullari. Periferiya modullari tashqi sxemalar bilan aloqa interfeysini tashkil qilish imkonini beradi. Masalan, kiritish – chiqarish universal portlari, suyuq kristalli indikatorlar (ЖКИ) drayverlari, ARO’ kirishlari, ШИМ chiqishlari va vaqt intervallarini hisoblash qurilmalari (taymerlar). Periferiya qurilmalarini boshqarish adreslangan ma’lumotlar makoni orqali amalga oshiriladi. Qulaylik uchun qisqartirilgan komandalar ishlatiladi (IN/OUT).

Mikrokontrollerlarning maxsus qurilmalari:

1. Konfiguratsiya bitlari
2. Manba ulanganda ishlovchi integrallangan “Сброс” sxemasi (POR)
3. Manba kuchlanishi pasayishidan ishlovchi “Сброс” sxemasi (BOR)
4. Storojevoy taymer
5. Energiya tejamkor rejimi (SLEEP)
6. Integrallangan (ichki) RC takt generatori
7. O’z – o’zini (ichki sxemali) (внутрисхемное) programmalashtirish.

2) AVR Atmel Studio 7.0 dasturida Atmega8 mikrokontrolleriga dastur kodini yozish.

Atmel Studio dasturi Atmel firmasining maxsuloti bo'lib, bu dastur Atmel firmasi chiqaradigan mikrokontrollerlariga kod yozish imkoniyatlari mavjud.

Quyidagi mikrokontrollerlariga dastur kodini yozish mumkin :

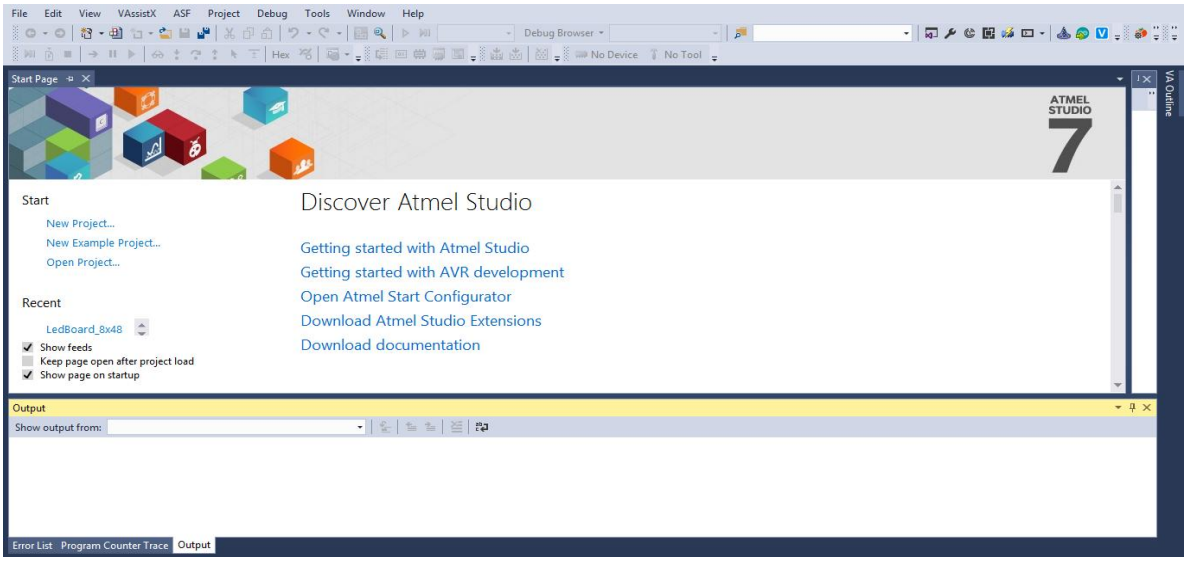
ATMEGA103	AVR
ATMEGA128	AVR2
ATMEGA1280	AVR2
ATMEGA1281	AVR2
ATMEGA1284P	AVR2
ATMEGA16	AVR2
ATMEGA162	AVR2
ATMEGA164P	AVR2
ATMEGA165	AVR2
ATMEGA165P	AVR2
ATMEGA168	AVR2
ATMEGA168_32PIN	AVR2
ATMEGA168P	AVR2
ATMEGA168P_32PIN	AVR2
ATMEGA169	AVR2
ATMEGA169P	AVR2
ATMEGA2560	AVR2
ATMEGA2561	AVR2
ATMEGA32	AVR2
ATMEGA324P	AVR2
ATMEGA325	AVR2
ATMEGA3250	AVR2
ATMEGA3250P	AVR2
ATMEGA325P	AVR2
ATMEGA328P	AVR2
ATMEGA328P_32PIN	AVR2
ATMEGA329	AVR2
ATMEGA3290	AVR2
ATMEGA3290P	AVR2
ATMEGA329P	AVR2
ATMEGA48	AVR2
ATMEGA48_32PIN	AVR2
ATMEGA48P	AVR2
ATMEGA48P_32PIN	AVR2
ATMEGA64	AVR2
ATMEGA640	AVR2
ATMEGA644	AVR2

AT89C1051	MCS8051
AT89C2051	MCS8051
AT89C4051	MCS8051
AT89C51	MCS8051
AT89C51.BUS	MCS8051
AT89C51RB2	MCS8051
AT89C51RB2.BUS	MCS8051
AT89C51RC2	MCS8051
AT89C51RC2.BUS	MCS8051
AT89C51RD2	MCS8051
AT89C51RD2.BUS	MCS8051
AT89C52	MCS8051
AT89C52.BUS	MCS8051
AT89C55	MCS8051
AT89C55.BUS	MCS8051
AT90S1200	AVR
AT90S2313	AVR
AT90S2323	AVR
AT90S2333	AVR
AT90S2343	AVR
AT90S4433	AVR
AT90S4434	AVR
AT90S8515	AVR
AT90S8535	AVR
AT90USB1286	AVR2
AT90USB646	AVR2

ATMEGA644P	AVR2
ATMEGA645	AVR2
ATMEGA6450	AVR2
ATMEGA649	AVR2
ATMEGA6490	AVR2
ATMEGA8	AVR2
ATMEGA8515	AVR2
ATMEGA8535	AVR2
ATMEGA88	AVR2
ATMEGA88_32PIN	AVR2
ATMEGA88P	AVR2
ATMEGA88P_32PIN	AVR2
ATMEGA8_32PIN	AVR2

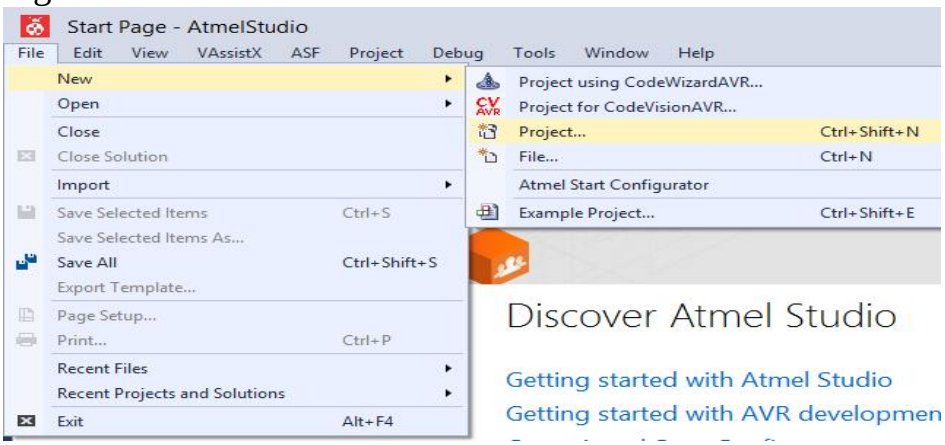
Atmel Studio dasturida ATmega8 mikrokontroller bazasida 8x48 matritsali yuguruvchi satr loyihasini uchun dastur kodini yozamiz. Buning uchun biz avvalo Atmel Studio 7.0 dasturini o'rnatamiz. as-installer-7.0.

Dasturni o'rnatib bo'lganimizdan keyin uni ishga tushiramiz. Ishga tushirilga dastur (2.1-rasm)da ko'rsatilgan.



2.1-rasm. Atmel Studioning umumiy ko'rinishi.

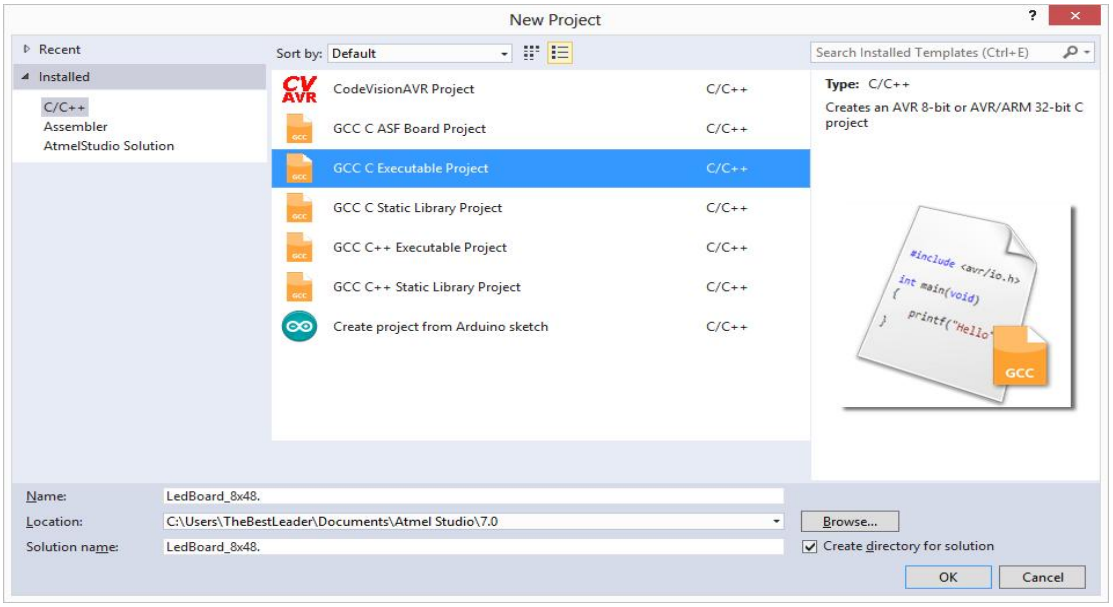
Dasturda ATmega8 mikrokontroller bazasida 8x48 matritsali yuguruvchi satr loyihasini uchun dastur kodini yozish uchun quyidagi ketma-ketlikni bajarish kerak. "**File - >New - > Project...**" shu tarzda amallarni bajarish (2.2-rasm)da ko'rsatilgan.



2.2-rasm.

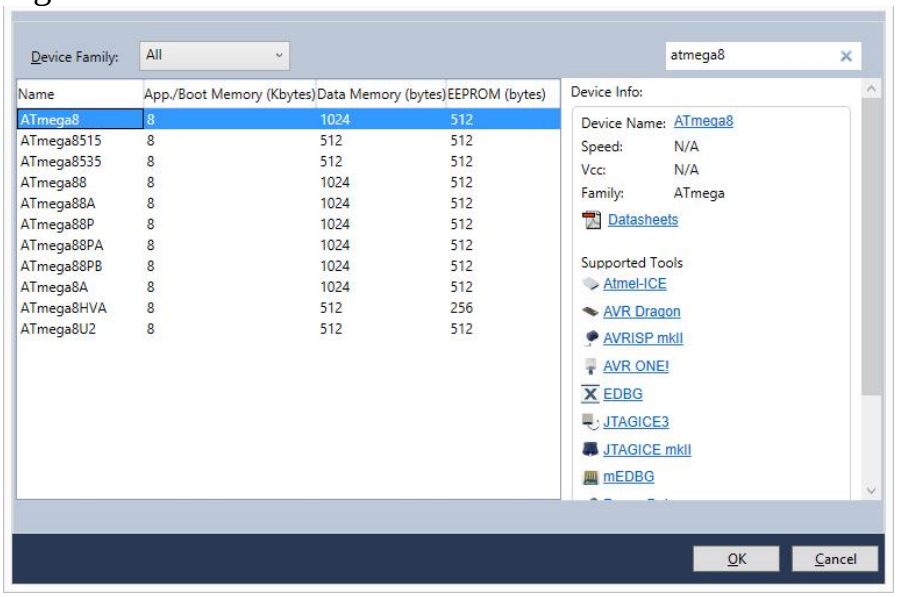
Atmel Studio dasturida mikrokontrollerlarga C/C++ va Assembler tillarida dasturiy kodlar yozish mumkin.

New Project dastur oynasidan **"C/C++"** tilini tanlab undan **GCC C Executable Project** ni tanlab **Name** joyiga project nomini kiritamiz va **"OK"** tugmasini bosamiz ya'ni bu jarayon 2.3-rasmda ko'rsatilgan.



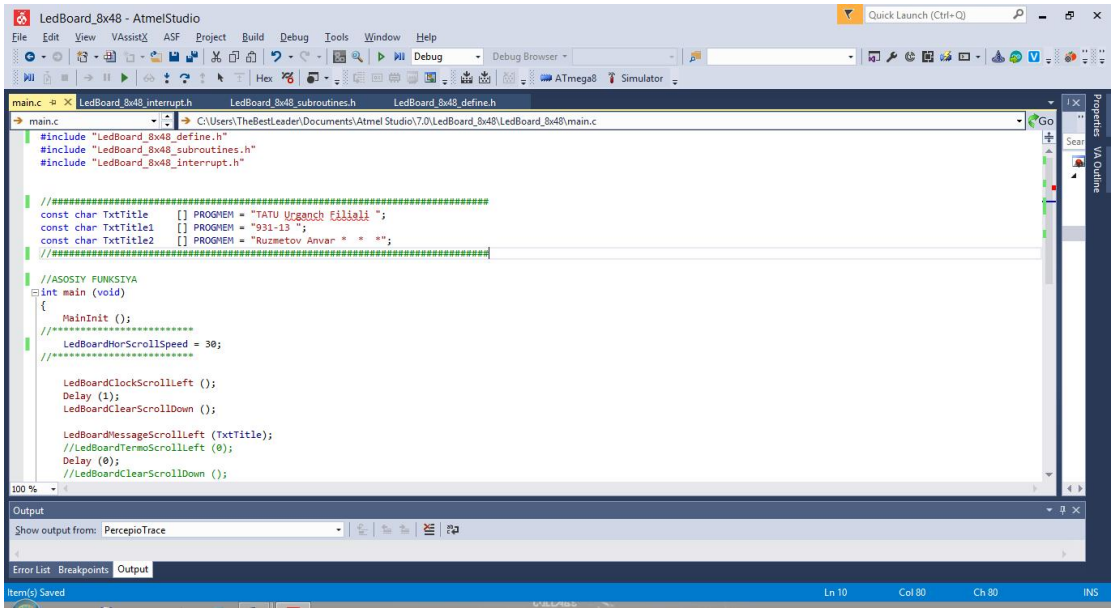
2.3-rasm

OK tugmasini bosganimizdan keyin dastur kodini yozish uchun bizga kerak bo'ladigan mikrokontrollerni turini tanlab olamiz. Bu jarayon 2.4-rasmda ko'rsatilgan.



2.4-rasm

Dastur muhiti ichiga loyiha kodi kiritilishi 2.5-ramda ko'rsatilgan.



2.5-rasm

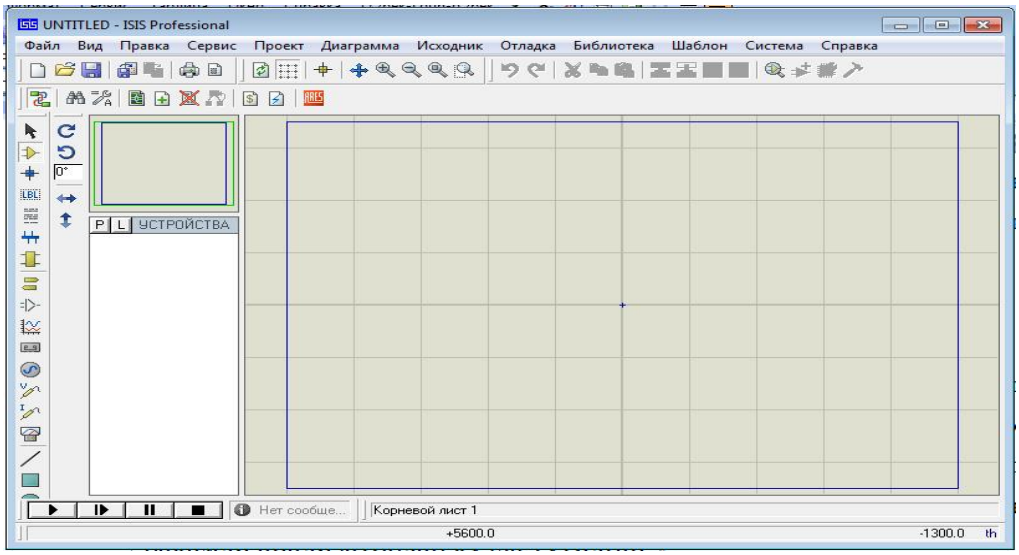
Dastur kodini kiritib so’ngra .HEX kodini olamiz va .hex kodini olib loyihani ishga tushirish uchun mikrokontrollerga hex kodini yozamiz va loyihamiz ishga tushadi.

3) ISIS Proteus vertual dasturida 8x48 matritsali yuguruvchi satr loyihasini ishlab chiqish.

ISIS Proteus dasturiy kompleksi elektron qurilmalar ishini o’rganishda amaliy animatsion va simulyatsion texnologiyadan mustaqil foydalanish imkonini beradi. Bu dasturiy kompleks ISIS va ARES dasturlarini hamda ko’plab zamonaviy mikrokontrollerlarning simulyatsion modellari va ularni dasturlash vositalarini o’z ichiga oladi. ISIS dasturi yordamida interaktiv rejimda har qanday murakkablikdagi analog yoki raqamli elektron qurilmani uning printsipial sxemasi asosida tadqiq etish, sxemaga kerakli o’zgartirishlar kiritish, yoki mikrokontrollerli qurilmalarning dasturini tekshirish, kerkli o’zgartirishlarni kiritish mumkin.

ISIS Proteus dasturiy paketiga **Пуск ► программы ►** Proteus 7 Professional tizim menyusi orqali murojaat qilish mumkin.

ISIS Proteus dasturi ishga tushirilgach, quyidagi oyna hosil bo’ladi (3.1-rasm):

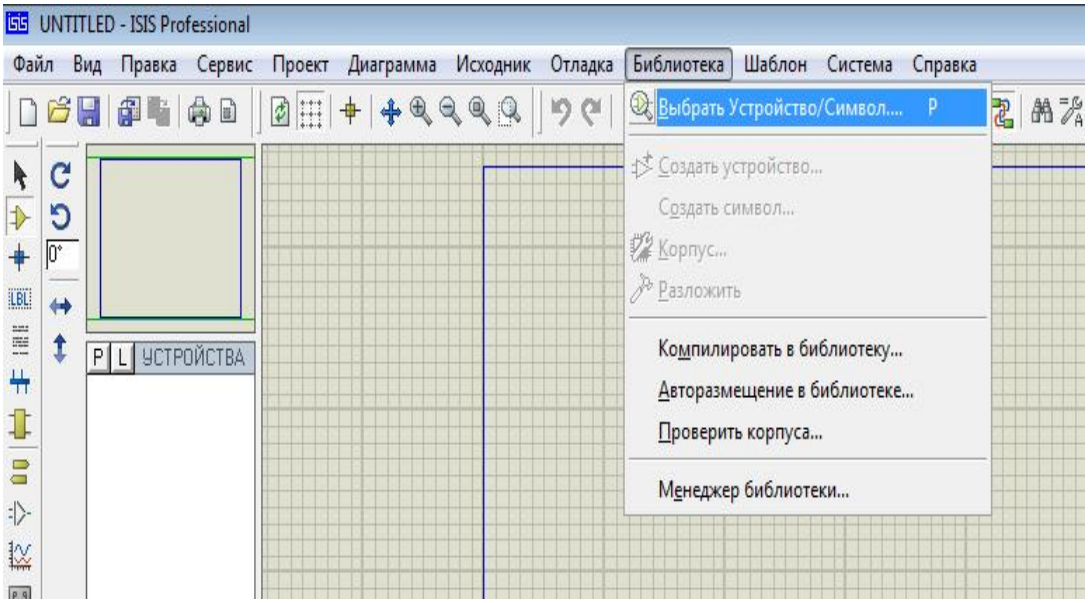


3.1-rasm. ISIS Proteus dasturi ishchi muhiti.

Agarda ushbu dasturda yangi loyiha yig'ish kerak bo'lsa, tanlangan loyihadagi qurilmani ishlashi, printsipial sxemasi va sxemada keltirilgan izohlar asosida qurilma haqida quyidagi ma'lumotnoma tayyorlanadi.

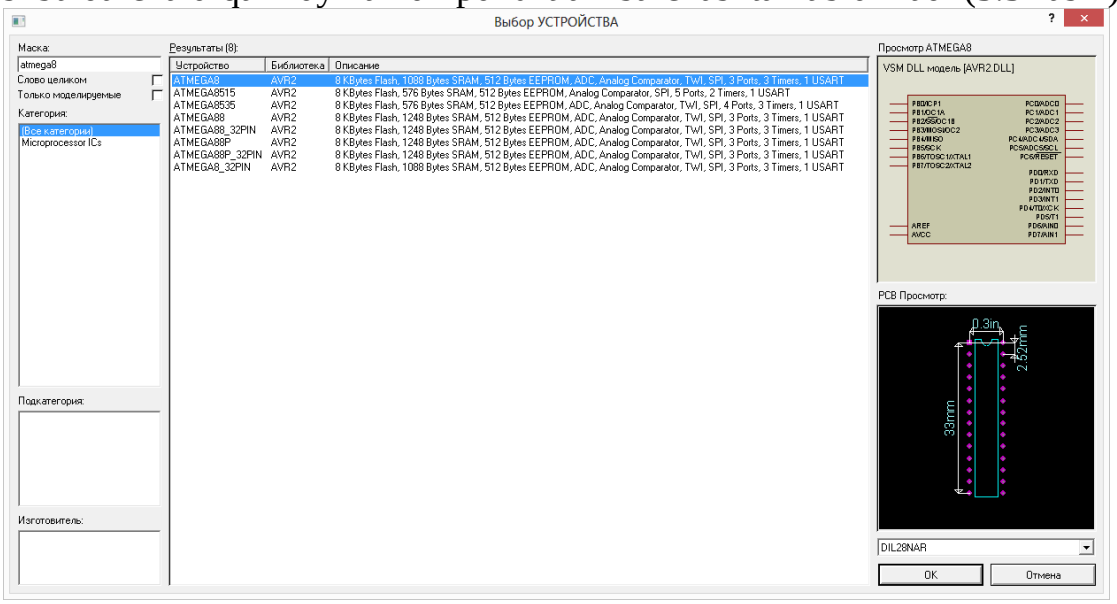
- qurilmaning vazifasi;
- qurilmada ishlatilgan elementlar va ularning vazifalari;
- mikrokontroller dasturi va bu dastur tayyorlangan muhit;
- qurilma ishini nazorat qilish uchun ishlatilishi mumkin bo'lgan o'lchov asboblari;
- qurilma ishini nazorat qilish rejimlari va boshqa ma'lumotlarni o'z ichiga olishi kerak.

Yangi loyihadagi sxema elementlarini bibliotekadan olingan komponentlardan foydalanib tahrirlash darchasiga birlashtirish va joylashtirishga uchun tashlanadi. Buning uchun "Biblioteka menyusidan **"Выбрать Устройство/Символ..P"** komandasi yoki P klavishasini tanlanadi (3.2-rasm).



3.2-rasm.

Ushbu darcha orqali loyiha komponentlari barchasi tanlab olinadi (3.3-rasm).



3.3-rasm. Loyiha komponentlarini Proteus dasturi bibliotekasidan tanlash darchasi

AVR mikrokontrollerlarini dasturlash uchun ularning komandalar tizimini, assembler tilini yoki C tilini hamda dasturlash vositalari yordamida mikrokontroller xotirasiga kiritilishi lozim bo’lgan dastur kodini tayyorlashni o’rganish talab etiladi.

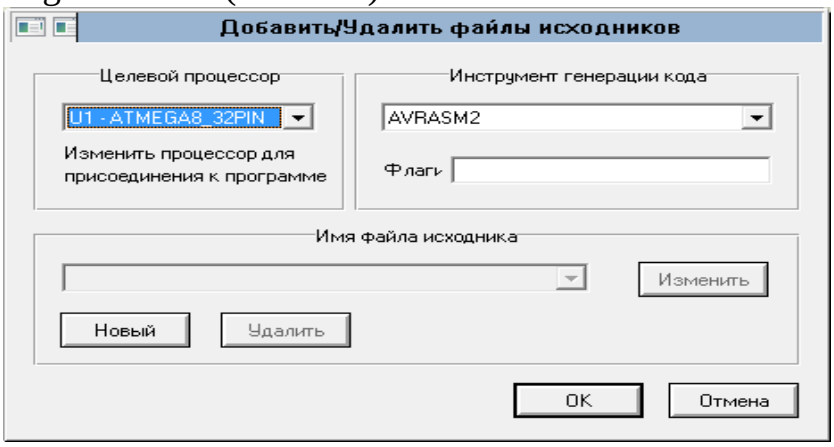
Proteus dasturida assembler tilida yozilgan dasturni HEX faylga aylantiruvchi translyatorlar mavjud. Translyator ishi natijasida hosil bo’lgan faylni mikrokontrollerning xotirasiga bevosita yuklash va ishga tushirish mumkin.

Bajardi:	Ruzmetov .A.Y	Bet	
Tekshirdi:	Setmetov N.U.		

Dastur tayyorlash jarayonida har bir mikrokontroller uchun mo’ljallangan alohida translyator tanlanib, aktivlashtiriladi.

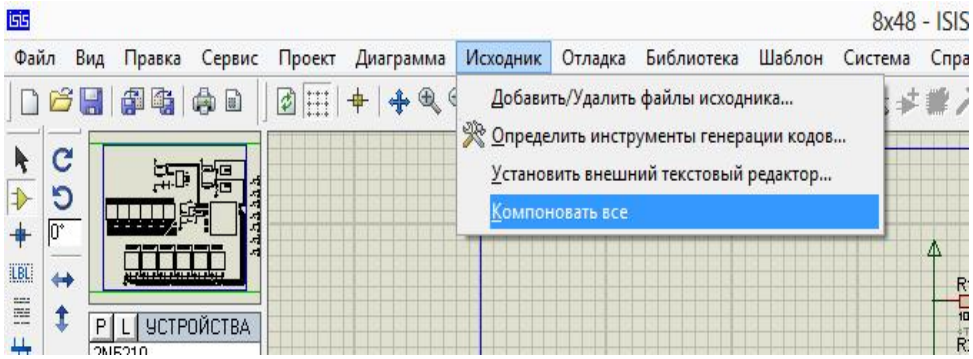
Tanlangan mikrokontroller uchun dastur yozish ISIS Proteus dasturida loyiha tuzishdan boshlangani maqbul. Bu loyiha hech bo’lmasa tanlangan mikrokontrollerning simulyatsion modelidan iborat bo’lishi lozim.

So’ngra shu mikrokontroller uchun yoziladigan dasturning dastlabki matni **“Исходник” → “Добавить/Удалить файлы исходника”** menyusi bilan loyiha tarkibiga kiritiladi (3.4-rasm):



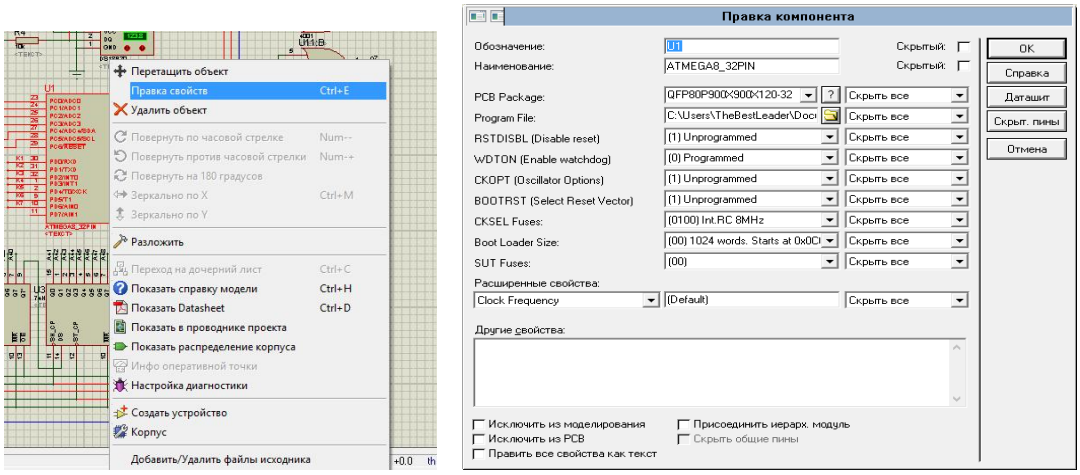
3.4-rasm. Mikrokontrollerga yoziladgan dastur kodini qo’shish.

Unga kerakli o’zgartirishlar kiritib, uni qayta translyatsiya qilish va to’plash quyidagi menyu orqali bajariladi (3.5-rasm):



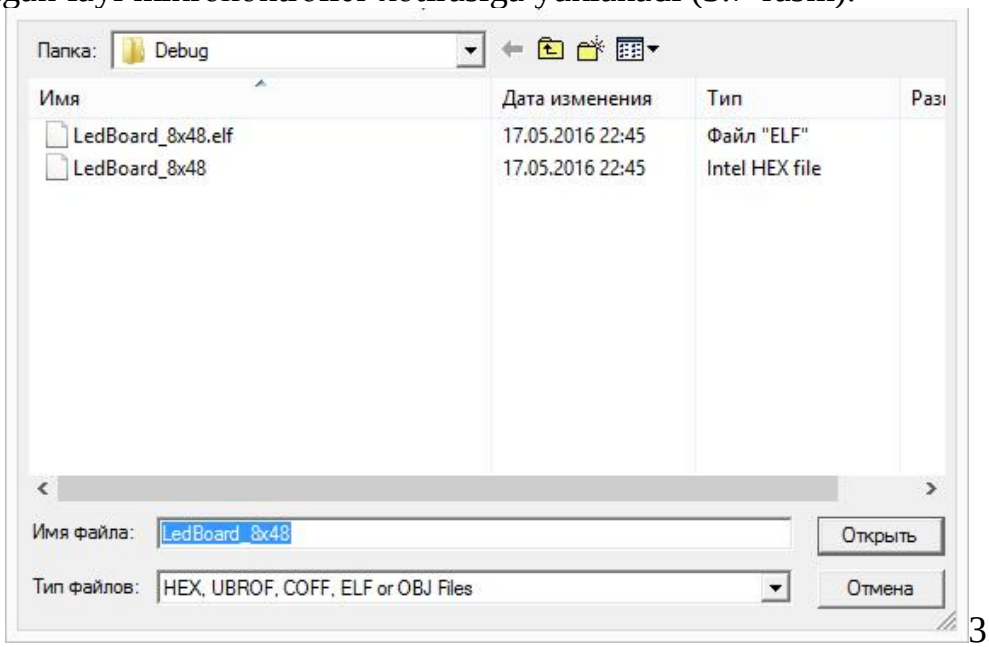
3.5-rasm.

Spundan so’ng tayyor bo’lgan dasturni mikrokontroller xotirasiga yuklash uchun Ctrl+E klavishalar juftligi yoki “**Правка свойств**” menyusi orqali quyidagi “Правка компонента” oynasi ochiladi(3.6-rasm):



3.6-rasm. “Правка компонента” oynasi.

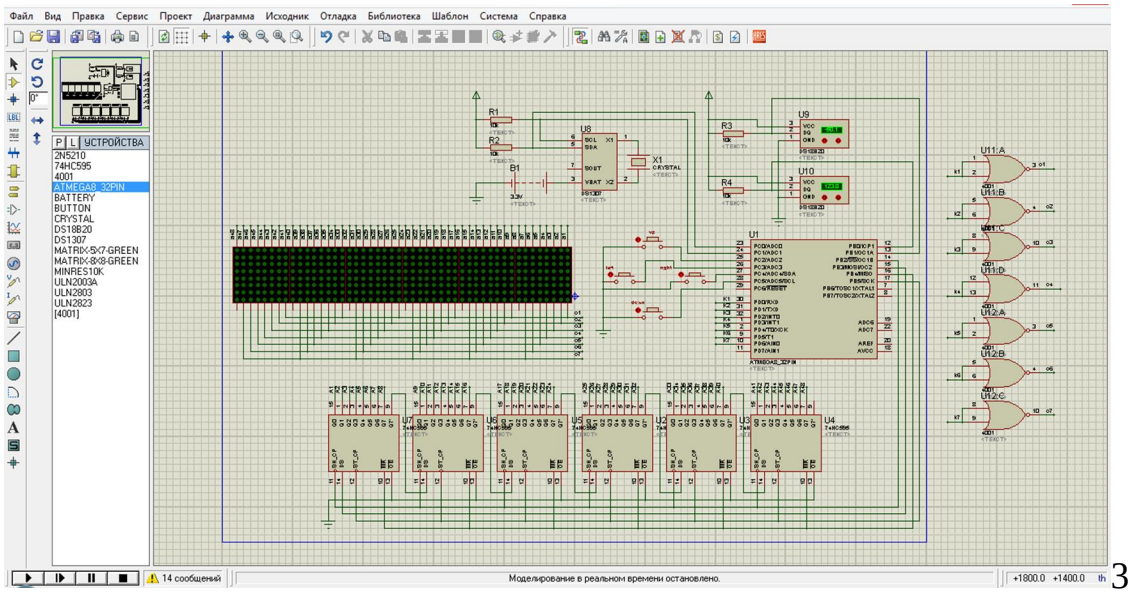
Bu oynadagi  belgi bilan ochiladigan quyidagi dialog oynasidan tanlangan fayl mikrokontroller xotirasiga yuklanadi (3.7-rasm).



3.7-rasm.

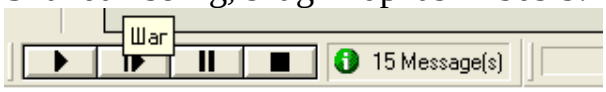
Mikrokontrollerga dastur (Hex-fayl) qo’shish uchun, sichqoncha o’ng tugmachasini mikrokontroller ustida bosib, kontekst menyudan Edit Properties ► Program File komandasi tanlanadi va bizni qiziqtirgan *.hex faylga bo’lgan yo’l qo’rsatiladi. Ushbu darchadan mikrokontrollerning ishchi chastotasini berish mumkin**.**

Proteus dasturida yig’ilgan mikrokontrollerlarning kiritish chiqarish portlarini tadqiq qilish qurilmasi loyihasi 3.8-rasmda keltirilgan.



3.8-rasm.

Shundan so’ng, shag knopkasini bosib:



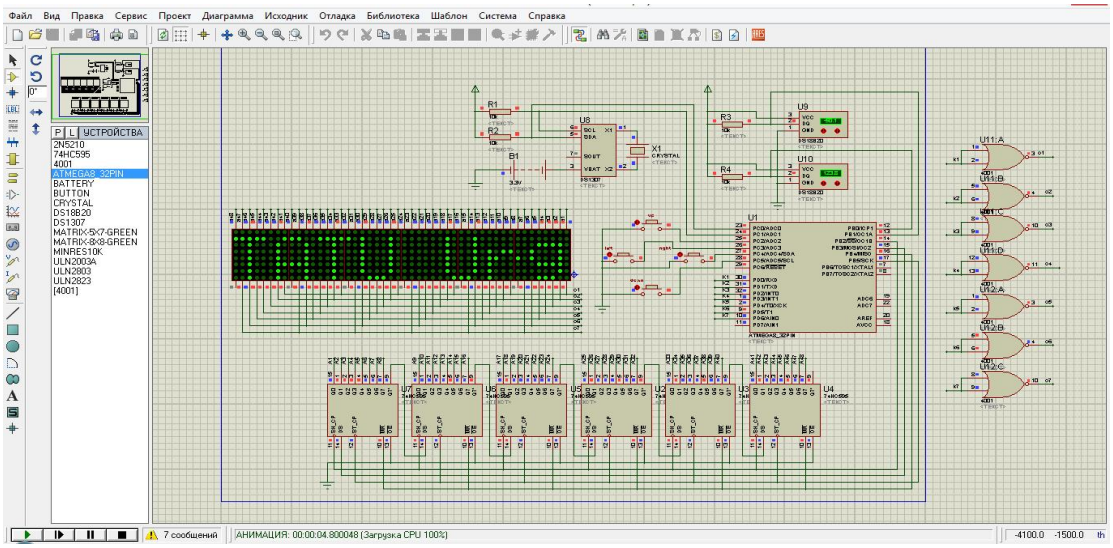
Tanlangan vaqt oralig’iga muvofiqlashtirish uchun imitatsiya rejimi menyusidan System ► Set Animation Options komandasi bajariladi va Single step time parametri beriladi. SHundan keyin imitatsiya qadami ko’rsatilgan vaqt oralig’ida ishlaydi.

Mikrokontroller uchun tayyorlangan dasturingizni qadam – baqadam rejimida ishga tushiringiz mumkin.

Bu oynada dastlabki matn, registrlar hamda vaqtinchalik qiymatlar oynachalari ko’rinib turibdi, agar ular ochilmagan bo’lsa “Отладка” menyusidan kerakli oynalarni ochish mumkin.

Bu oynalardagi axborotlar asosida mikrokontroller bajarayotgan komandalar natijalarini kuzatib borish mumkin va bu axborot asosida dasturni to’g’ri yoki noto’g’ri ishlayotganini bilish mumkin.

Ma’lumki, mikrokontrollerli elektron qurilmalar ishini raqamli analizatorsiz tadqiq etish ancha mushkul. Shning uchun barcha simulyatsion va emulyatsion texnologiyalarni o’z ichiga olgan dasturlash va loyihalash vositalari tarkibida turli virtual o’lchov qurilmalari, jumladan raqamli analizatorlar ko’zda tutilgan.



Proteus dasturiy majmuasida quyidagi virtual qurilmalar yordamida turli murakkablikdagi qurilma va mikroprotsessor tizimlarini tadqiq etish mumkin:Virtual Oscilloscope Model, Virtual Logic Analyser Model, Virtual Signal Generator Model, Virtual Pattern Generator Model, Virtual Counter Timer Model, Virtual Terminal Model, SPI Debugger, I2C Debugger, Voltmetr va ampermetr.

Proteus – virtual modellash tizimida mikrokontrollerli qurilmalarning elektr zanjirlari, sxematik echimlari va dasturiy vositalari to'liq tekshirilib, sozlab bo'lingach, otladka qilingan dastur faylini mikrokontroller xotirasiga doimiy saqlab, ishlatish uchun kiritib qo'yish kerak bo'ladi. Buning uchun mikrokontroller ishlab chiqaruvchilar va boshqalar tomonidan turli texnik va dasturiy vositalar yaratilgan. Bularga Bascom-avr, CodeVisionAVR, PonyProg, Avrdude, Avr Studio 4 kabi programma – texnik vositalarni ko'rsatish mumkin.

4) Dastur kodi.

Main.c

```
#include "LedBoard_8x48_define.h"
#include "LedBoard_8x48_subroutines.h"
#include "LedBoard_8x48_interrupt.h"


//###########################################################################
const char TxtTitle  [] PROGMEM = "TATU Urganch Filiali ";
const char TxtTitle1 [] PROGMEM = "931-13 ";
const char TxtTitle2 [] PROGMEM = "Ruzmetov Anvar *  *  *";
//###########################################################################

//ASOSIY FUNKSIYA
int main (void)
{
          MainInit ();
//*************************
          LedBoardHorScrollSpeed = 30;
//*************************

          LedBoardClockScrollLeft ();
          Delay (1);
          LedBoardClearScrollDown ();

          LedBoardMessageScrollLeft (TxtTitle);
          //LedBoardTermoScrollLeft (0);
          Delay (0);
          //LedBoardClearScrollDown ();
```

```
            LedBoardMessageScrollLeft (TxtTitle1);
            //LedBoardTermoScrollLeft (1);
            Delay (0.8);
            LedBoardClearScrollDown ();

            LedBoardMessageScrollLeft (TxtTitle2);
            LedBoardClearScrollLeft ();
            LedBoardClockScrollLeft ();
            }
```

LedBoard_8x48_interrupt.h

```
ISR (TIMER2_COMP_vect)
{
            static char LedBoardXCoun = 0;
            LedBoardOut (LedBoardXCoun);
            if (LedBoardXCoun == 1)
                        if (Flags1 & b1ClockReadEnable)
                                    IICClockReadData ();
            if ((LedBoardXCoun >= 2) && (LedBoardXCoun <= 5))
                        if (Flags1 & b1OneWireGet)
            OneWireIntTermoGet ();
            if (LedBoardXCoun == 6){
                        BtnGet ();
            VarProcessing ();
            }
            if (++LedBoardXCoun == 8){
LedBoardXCoun = 0;
                        Flags1 |= b1LedBoardPageCompleted;
ClockBufferProcess ();}}
char* pClockBuf;
void ClockBufferProcess (void)
{
            static char ShiftDelay;
static char ShiftMask;
pClockBuf = ClockGraphBuf;
char Flags = ClockChangeMask;
if (! ShiftMask)
                        if (Flags){
                                    ShiftMask = 1;
                                    ShiftDelay = 0;
            }
            if (! ShiftDelay--){
                        ShiftDelay = 6;
                        char ShiftTemp;
                        if (Flags & 0x20) ShiftTemp = ShiftMask;
else ShiftTemp = 0;
                        ClockSimbToBuf ((ClockData[2]>>4)+0x30, ShiftTemp);
                        if (Flags & 0x10) ShiftTemp = ShiftMask;
else ShiftTemp = 0;
                        ClockSimbToBuf ((ClockData[2] & 0x0f)+0x30,ShiftTemp);
                        ClockSimbToBuf (':', 0);
                        if (Flags & 0x08) ShiftTemp = ShiftMask;
                        else ShiftTemp = 0;
                        ClockSimbToBuf ((ClockData[1]>>4)+0x30, ShiftTemp);
                        if (Flags & 0x04) ShiftTemp = ShiftMask;
                        else ShiftTemp = 0;
                        ClockSimbToBuf ((ClockData[1] & 0x0f)+0x30, ShiftTemp);
                        ClockSimbToBuf (':', 0);
                        if (Flags & 0x02) ShiftTemp = ShiftMask;
                        else ShiftTemp = 0;
                        ClockSimbToBuf ((ClockData[0]>>4)+0x30, ShiftTemp);
                        if (Flags & 0x01) ShiftTemp = ShiftMask;
else ShiftTemp = 0;
                        ClockSimbToBuf ((ClockData[0] & 0x0f)+0x30, ShiftTemp);
                        ShiftMask <<= 1;
                        if (! ShiftMask)
                        ClockChangeMask = 0; }
            if (ClockPosition){
                        char* pLedBoardBuffer = LedBoardBuffer + ClockPosition + LedBoardSimbWidth - 1;
                        char* pClockGraphBuf = ClockGraphBuf + sizeof ClockGraphBuf -1;
                        for (uint8_t i =sizeof ClockGraphBuf; i; i--){
                        if ((pLedBoardBuffer < LedBoardBuffer + ((LedBoardSize + 1) * LedBoardSimbWidth)) &&
                        (pLedBoardBuffer >= LedBoardBuffer + LedBoardSimbWidth))
                                    *pLedBoardBuffer = *pClockGraphBuf;
                                    pLedBoardBuffer--;
                                    pClockGraphBuf--;}}}
void ClockSimbToBuf (char byte, char shift){
            char* pSimbMass = SimbMass + ((byte-0x20) * FontSimbWidth);
            for (char i=0; i<FontSimbWidth; i++){
                        char temp = pgm_read_byte (pSimbMass++);
                        if (shift){
                        *pClockBuf >>= 1;
                        if (temp & shift)
                        *pClockBuf |= 0x80;}
                        else
                        *pClockBuf = temp;
                        pClockBuf++;          }
            for (char i=0; i<(LedBoardSimbWidth-FontSimbWidth); i++)
                        *pClockBuf++ = 0;}
void LedBoardOut (char NumLine) {
            char MaskXRow = (1<<(NumLine));
            char* pLedBoardBuffer = LedBoardBuffer + LedBoardSimbWidth;
            for (char i=0; i<(LedBoardSize * LedBoardSimbWidth); i++){
            char temp = *pLedBoardBuffer++;
```

Bajardi:	Ruzmetov .A.Y	Bet	
Tekshirdi:	Setmetov N.U.		

```
//############################
                    temp >>= 1;
//############################
                    if (temp & MaskXRow)
                              LedBoardDataPORT |= LedBoardData;
                    else
                              LedBoardDataPORT &= ~LedBoardData;
                    _delay_us (1);
                    LedBoardDataPORT |= LedBoardClc;
                    _delay_us (1);
                    LedBoardDataPORT &= ~LedBoardClc;}
                    LedBoardRowPORT = 0xFF-LedBoardRowInvert;
                    LedBoardDataPORT |= LedBoardStrobe;
                    _delay_us (1);
                    LedBoardDataPORT &= ~LedBoardStrobe;
                    _delay_us (1);
                    LedBoardRowPORT = (~LedBoardRowInvert)^MaskXRow;
void BtnGet (void){
          #define BtnLockTime 30
          #define BtnLongPressTime 2000
          static unsigned char BtnLockBit;
          static unsigned char BtnLockCoun;
          static unsigned char BtnLongCoun;
          static unsigned char BtnTemp;
          static unsigned char BtnLongSpeedTmp;
          static unsigned char BtnLongTimer;
          char BtnStack = 0;
          char temp = ~BtnPIN;
          if (temp & Button1)             BtnStack = 0x01;
          if (temp & Button2)             BtnStack = 0x02;
          if (temp & Button3)             BtnStack = 0x04;
          if (temp & Button4)             BtnStack = 0x08;
          if (BtnStack){
                    BtnTemp = BtnStack;
                    BtnStack = 0;
                    if (BtnLockCoun < (BtnLockTime/10)){
                              BtnLockCoun++;
                              return;    }
                    BtnLockBit=1;
          if (BtnLongCoun >= (BtnLongPressTime/10)){
          LedBoardFlashCoun = 0;
          BtnLongTimer ++;
          if (! (BtnLongTimer & 0b00001111))
                    BtnLongSpeed = BtnLongSpeedTmp;
                              if (BtnLongTimer == 128){
                                        BtnLongTimer = 0;
                              if (BtnLongSpeedTmp < 64)
                                        BtnLongSpeedTmp <<= 1;}
                              return;}
                    BtnLongCoun ++;
                    if (BtnLongCoun >= (BtnLongPressTime/10)){
                              BtnFlags |= (BtnTemp << 4);
                              BtnLongSpeed = 1;
                              BtnLongSpeedTmp = 1;
                              BtnLongTimer = 0;}}
          else{     if (BtnLockCoun != 0){
                              BtnLockCoun --;return;}
                    if (! (BtnLockBit))
          return;
                    BtnLockBit =0;
                    if (BtnLongCoun < (BtnLongPressTime/10)){
                              BtnFlags |= (BtnTemp);
          LedBoardFlashCoun = 0;}
                    BtnLongCoun = 0;
                    BtnLongSpeed = 0;}}
void VarProcessing (void){
          if (++ LedBoardFlashCoun == 100)
                    LedBoardFlashCoun = 0;
          if (++ ClockMiliSecond == 120)
                    ClockMiliSecond = 0;
          static char OneWireCicle = 0;
          if (++ OneWireCicle == 100){
                    OneWireCicle = 0;
                    Flags1 |= b1OneWireGet;
                    if (BtnTimeOutCoun)
          BtnTimeOutCoun--;}}
void OneWireIntTermoGet (void){
          static unsigned char step = 0;
          char mask;
          if (step < 16)
          mask = OneWireLine1;
          else{
                    if (step < 32)
          mask = OneWireLine2;
                    else
                    mask = OneWireLine1 | OneWireLine2;}
          unsigned char byte;
          byte = pgm_read_byte (&DSReadingMap [step++]);
if (byte < 8)
          OneWireReadBuf [byte] = OneWireData(0xff, mask);
          if (byte == 0x0b)
                    OneWireReadBuf [8] = OneWireCheckSumm;
          if (byte == 0x0c)
                    OneWireData(0xff, mask);
```

Bajardi:	Ruzmetov .A.Y	Bet	
Tekshirdi:	Setmetov N.U.		

```
            if (byte == 0x0d)
                        OneWireReset (mask);
            if (byte == 0x0e){
                        char OneWireCRC       = OneWireReadBuf[8];
                        char* pBufData = OneWireReadBuf;
                        for (unsigned char i=0; i<2; i++){
                                    char DevType = *pBufData++;
                                    int termo = 0;
                                    termo = *pBufData++;
                                    termo |= (*pBufData++) << 8;
char Config = *pBufData++;
            if (OneWireCRC                                                                                                          ||
!((DevType == 0x28)|| (DevType == 0x10)))                                                                                       OneWireErr
|= (1<<i);
else{
                                                OneWireErr &= ~(1<<i);
                                                if (DevType == 0x10)
                                                Config = 0;
                                                Config >>= 5;
                                    Config ^= 0xff;
                                                Config &= 0x03;
                                                while (Config--)
                                                            termo <<= 1;
                        termo *= 10;
            termo /= 16;
                                                OneWireTermoData [i] = termo;}
                                    OneWireCRC = OneWireCheckSumm;}}
            if (byte == 0x0f){
                        Flags1 &= ~b1OneWireGet;        step =0;   }
            if (byte >= 0x10){
                        OneWireData (byte, mask);
if (byte == 0xBE)
                                    OneWireCheckSumm = 0;}}
```

LedBoard_8x48_subroutines.h

```
void LedBoardSimbPrint (unsigned char simbol, unsigned char position){
            if (position > (LedBoardSize+1))
            return;
            char* pPosition = position * LedBoardSimbWidth + LedBoardBuffer;
            if (simbol < 0x20)
            simbol = 0x20;
            if (simbol >= 0xc0)
            simbol -= 0x40;
            char *pSimbol = (simbol - 0x20) * FontSimbWidth + SimbMass;
            for (char i=0; i<FontSimbWidth; i++)
            *pPosition++ = pgm_read_byte (pSimbol++);
            for (char i=0; i<(LedBoardSimbWidth-FontSimbWidth); i++)
            *pPosition++ = 0;}
void LedBoardMessageShow (char *pAdrTxt){
            unsigned char position = 1;
            while (1){
                        if (position > LedBoardSize)
                        return;
                        char byte = pgm_read_byte (pAdrTxt++);
                        if (! byte)                         return;
                        LedBoardSimbPrint (byte, position++);}}
void LedBoardMessageScrollLeft (char *pAdrTxt){
            while (1){
                        char byte = pgm_read_byte (pAdrTxt++);
                        if (! byte)          return;

                        LedBoardSimbPrint (byte, LedBoardSize+1);
                        for (char i=0; i<LedBoardSimbWidth; i++)
                        LedBoard1PixScrollLeft ();}}
void LedBoardClearScrollLeft (void){
            LedBoardSimbPrint (' ', LedBoardSize+1);
            unsigned char Length = LedBoardSize * LedBoardSimbWidth;
            for (char i=0; i<Length; i++)
            LedBoard1PixScrollLeft ();}
void LedBoard1PixScrollLeft (void){
            unsigned char speed = 100/LedBoardHorScrollSpeed;
            for (char z=0; z<speed; z++){
                        while (! (Flags1 & b1LedBoardPageCompleted)){}
                        Flags1 &= ~b1LedBoardPageCompleted;}
            char* pLineBuf1 = LedBoardBuffer+LedBoardSimbWidth;
            char* pLineBuf2 = LedBoardBuffer+LedBoardSimbWidth+1;
            unsigned char Length = (LedBoardSize+1)*LedBoardSimbWidth-1;
            for (char z=0; z<Length; z++)
            *pLineBuf1++ = *pLineBuf2++;
            *pLineBuf1 = 0;
            if (ClockPosition)
            ClockPosition--;}
void LedBoardClearScrollDown (void){
            ClockPosition = 0;
            unsigned char speed = 100/LedBoardVertScrollSpeed;
            for (char i=0; i<8; i++){
            for (char z=0; z<speed; z++){
                        while (! (Flags1 & b1LedBoardPageCompleted)){}
                                    Flags1 &= ~b1LedBoardPageCompleted;}
                        char* pBufPtr = LedBoardBuffer+LedBoardSimbWidth;
                        unsigned char Length = LedBoardSize*LedBoardSimbWidth;
                        for (char z=0; z<Length; z++){
                                    char temp = *pBufPtr;
```

Bajardi:	Ruzmetov .A.Y	Bet	
Tekshirdi:	Setmetov N.U.		

```
                                temp >>= 1;
                                *pBufPtr++ = temp;}}}
const char TxtError   [] PROGMEM = "   ERROR";
void LedBoardTermoShow (char Position, uint8_t DevNum){
          if (OneWireErr & (1<<DevNum))
          LedBoardMessageShow (TxtError);
else{
                    LedBoardSimbPrint (' ', Position);
                    LedBoardIntPrint(OneWireTermoData[DevNum],Position+1, 5, 1);
                    if ((Position+6) <= (LedBoardSize+1))
                    LedBoardSimbPrint ('`', Position+6);
                    if ((Position+7) <= (LedBoardSize+1))
                    LedBoardSimbPrint ('C', Position+7);}}
void LedBoardTermoScrollLeft (uint8_t DevNum){
          if (OneWireErr & (1<<DevNum))
          LedBoardMessageScrollLeft (TxtError);
          else{for (char i=9; i>1; i--){
          LedBoardTermoShow (i, DevNum);
          for (char x=0; x<LedBoardSimbWidth; x++)
          LedBoard1PixScrollLeft ();}     }}
void LedBoardClockShow (uint8_t Position){
          if ((Position > 0) && (Position <= LedBoardSize+1))
          ClockPosition = (Position +7) * LedBoardSimbWidth;}
void LedBoardClockScrollLeft (void){
          LedBoardClockShow (9);
          LedBoardClearScrollLeft ();}
NOINLINE void Delay (unsigned char second){
          for (unsigned char i=0; i<second; i++)
          _delay_ms (1000);}
void LedBoardIntPrint (signed int integer, uint8_t IntPos, uint8_t IntLength, uint8_t DotPosition){
          char IntBuf [IntLength];
          char* pIntBuf = IntBuf+IntLength-1;
          uint8_t DigitCoun = 0;
          uint8_t negative = 0;
          if (integer < 0){
                    integer ^= 0xffff;
                    integer++;
                    negative = 1;}
          for (uint8_t i=0; i<IntLength; i++){
                    *pIntBuf-- = integer %10+0x30;
integer /= 10;
DigitCoun ++;
                    if (! integer)
                    break;}
          if (DotPosition){
while (DotPosition > DigitCoun){
                                *pIntBuf-- = 0x30;
                                DigitCoun++;}
                    if (DotPosition == DigitCoun){
*pIntBuf-- = '.';
                                *pIntBuf-- = 0x30;
                                DigitCoun += 2;
                                goto m1;}
                    char *pMoveCntr = pIntBuf--;
                    char MoveVal = DigitCoun-DotPosition;
                    for (char i=0; i<MoveVal; i++){
                                char temp = *(pMoveCntr+1);
                                *pMoveCntr++ = temp; }
                    *pMoveCntr = '.';
                    DigitCoun++;}
          m1:       if (negative){
                    *pIntBuf-- = '-';
                    DigitCoun++;}
          while (DigitCoun++ < IntLength)
          *pIntBuf-- = 0x20;
          for (uint8_t i=0; i<IntLength; i++){
                    if ((IntPos+i) <= (LedBoardSize+1))
                    LedBoardSimbPrint (IntBuf [i], IntPos+i);}}
void OneWireReset (char mask){
          OneWireDDR |= (mask);_delay_us (480);
          OneWireDDR &= ~(mask);_delay_us (480);}
unsigned char OneWireData (unsigned char OutByte, char mask){
          char InByte =0;
          for (unsigned char i = 0; i<8; i++){
                    InByte >>= 1;
                    cli ();
                    OneWireDDR |= mask;
if (OutByte & 0x01){
                                _delay_us (5);
                                OneWireDDR &= ~mask;
                                _delay_us (5);
                                if (OneWirePIN & mask)
                                InByte |= 0x80;
          sei ();   _delay_us (70);}
                    else{               _delay_us (75);
                                OneWireDDR &= ~mask; sei ();
                                _delay_us (5);}
                    OutByte >>= 1;}
          CheckSumm (InByte);
return InByte;}
void CheckSumm(char data){
          for (char i=0; i<8; i++){
                    char temp = data;
                    temp ^= OneWireCheckSumm;
```

Bajardi:	Ruzmetov .A.Y	Bet	
Tekshirdi:	Setmetov N.U.		

```
                    if (temp & 0x01){
                              OneWireCheckSumm ^= 0x18;
                              temp = 0x80;}
                    else  temp = 0;
                    OneWireCheckSumm >>= 1;
                    OneWireCheckSumm |= temp;
                    data >>= 1;}}
void IICClockReadData (void){
          IICStart();
          IICByteWrite (0xD0);
          IICByteWrite (0x00);
          IICStart();
          IICByteWrite (0xD1);
          char temp;
          char Accumulate = 0;
          char NumBit = (1<<0);
char* pClockData = ClockData;
          for (char i = 0; i<6; i++){
          temp = IICByteRead (0);
                    if (i == 0){
          if (temp != *pClockData){
                                        Flags1 |= b1ClockChange;
          ClockMiliSecond = 0; }}
                    if (i < 3){
                    if ((temp & 0x0F) != (*pClockData & 0x0F))
                              Accumulate |= NumBit;
                              NumBit <<= 1;
                              if ((temp & 0xF0) != (*pClockData & 0xF0))
Accumulate |= NumBit;
                              NumBit <<= 1;}
                    *pClockData++ = temp;}
          *pClockData = IICByteRead (1);
          IICStop();
          ClockChangeMask |= Accumulate;}
void IICClockWriteData (void){
          IICStart();
          IICByteWrite (0xD0);
          IICByteWrite (0x00);
          for (unsigned char i = 0; i<7; i++)
          IICByteWrite (ClockData [i]);
          IICStop();}
char ConvBinDecToHex (char byte){
          char result = 10 * (byte >> 4);
          result += (byte & 0x0f);
          return result;}


char ConvHexToBinDec (char byte){
          char result = byte %10;
          result |= ((byte/10) << 4);
          return result;}
NOINLINE void Delay2Mks (void){
          _delay_us (2);}
void IICStart (void){
          IICDDR &= ~(IICData | IICClc);
          Delay2Mks();
          IICDDR |= IICData;              Delay2Mks();
          IICDDR |= IICClc;               Delay2Mks();}
void IICStop (void){
          IICDDR |= (IICData | IICClc);
          Delay2Mks();
          IICDDR &= ~IICClc;              Delay2Mks();
          IICDDR &= ~IICData;             Delay2Mks();}
char IICByteWrite (char byte){
          for (char i = 0; i<8; i++)
          byte = IICBit (byte);
          byte = 0x80;
          return IICBit (byte);}
char IICByteRead (char bit){
          char byte = 0xFF;
          for (char i = 0; i<8; i++)
          byte = IICBit (byte);
          bit <<= 7;
          IICBit (bit);
return byte;}
char IICBit (char byte){
          if (byte & 0x80)
          IICDDR &= ~IICData;
          else
          IICDDR |= IICData;
          Delay2Mks();
          IICDDR &= ~IICClc;
          while (! (IICPIN & IICClc)){}
          Delay2Mks();
          char temp;
          if (IICPIN & IICData)
          temp = 1;
          else
          temp = 0;
          IICDDR |= IICClc;
          byte <<= 1;
          byte |= temp;
          return byte;}
```

Bajardi:	Ruzmetov .A.Y	Bet	
Tekshirdi:	Setmetov N.U.		

Xulosa

Ushbu kurs ishidan maqsad “Mikroprosessor” fanidan mavjud o’quv dasturiga kiritilgan yangiliklarni, zamonaviy raqobatbardosh texnika va texnologiyalarni talabalar tomonidan mukammal o’zlashtirilishini ta’minlash, zamonaviy raqobatbardosh elektron qurilmalar yaratish texnologiyalarini mukammal bilishdir.

AVR mikrokontrollerlarini tadqiq qilishda Chip Blaster AVR programmatoridan foydalangan holda, eng ko’p tarqalgan AT90S2313, AT90S8515 va ATmega(32-16-8) mikrokontrollerlari - CodeVisionAVR paketidan foydalangan holda **C** tilida dasturlanildi.

Mikrokontrollerlarini tadqiq qilishda kompyuterga Atmel Studio 7.0 build 730 integrallashgan dasturlash muhiti, ISIS Proteus 7.64 imitator dasturiy paketi, CodeVision AVR 2.05.0. dasturlash muhiti, WinAVR dasturi, Chip Blaster AVR programmatori o’rnatilgan va sozlangan bo’lishi lozim.

Shuningdek, kurs ishni ishlash davomida quyidagi ishlar amalga oshirildi:

- Atmel AVR mikrokontrollerlari strukturasi va asosiy qurilmalari ishlash printsipi o’rganildi;
- ISIS Proteus dasturida AVR mikrokontrollerlarining simulyatsion modelini tuzish algoritmi ishlab chiqildi;
- CodeVisionAVR paketida AVR mikrokontrollerlarini dasturlash va tekshirish amalga oshirildi.

Respublikamizda yangi qurilayotgan bino inshoatlarni avtomatlashtirilgan ventiyatsiya tizimini loyihalashda mikroprotsessor texnikasini qo’llash hozirgi kunning dolzarb masalalaridan biridir. Bugungi kunga kelib mamlakatimizda ushbu masalalar yuzasidan bir qancha ishlar amalaga oshirilmoqda.

Foydalanilgan adabiyotlar ro’yxati

1. М.С. Голубцов Микроконтроллеры AVR - от простого к сложному/ М.С. Голубцов - М.:СОЛОН-Пресс, 2003. 288с. – (Серия “Библиотека инженера”) ISBN 5-98003-034-4
2. М.Б. Лебедев CodeVisionAVR: Пособие для начинающих. – М.: Додека -XXI, 2008. -592 с.: ил. ISBN 978-5-94120-192-1
3. Программирования на языке С для AVR и PIC микроконтроллеров./ Сост. Ю.А. Шпак – К. : “МК-Пресс”, 2006 – 400 с., ил. ISBN 966-8806-16-6.
4. Новиков Ю. В., Скоробогатов П. К. Основы микропроцессорной техники. Курс лекций. — М.: Интернет-университет информационных технологий, 2003. — ISBN 5-7163-0089-8
5. Зельдин Е.А. Цифровые интегральные микросхемы в информационно-измерительной аппаратуре/ Е.А. Зельдин. Л.: Энергоатомиздат, 1986. 280 с. ил.
6. Калабеков, Б.А. Цифровые устройства и микропроцессорные системы: учеб. для техникумов связи / Б.А. Калабеков. – М.: Горячая линия – Телеком, 2008. – 336 с.
7. Proteus_Help elektron ma’lumotnoma
8. Mikroprotsessorlar va yarim o’tkazgichli komponentlarning elektron ma’lumotnomasi.
9. Карлашук В.И. Электронная лаборатория на IBM PC. 2002.
10. Агуров П. В. Интерфейс USB. Практика использования и программиррования. – СПб.: БХВ-Петербург, 2005.
11. http://www.ic-prog.com
12. http://www.labcenter.co.uk
13. http://www.microchip.ru
14. http://www.datasheet.com
15. http://www.sxema.ru
16. http://www.radiolocman.ru
17. www.pcports.ru
18. www.eldigi.ru
19. www.atmel.datasheet.com
20. www.proavr.narod.ru