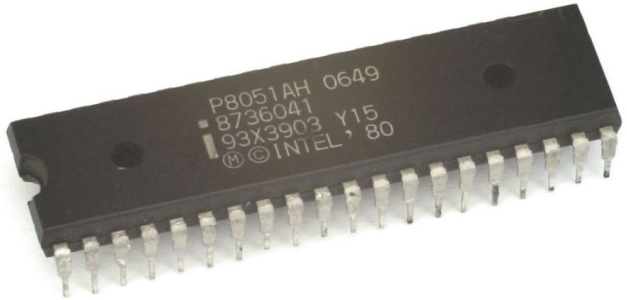


8051-es mikrovezérlő

mikrovezérlő 1980-ból napjainkban

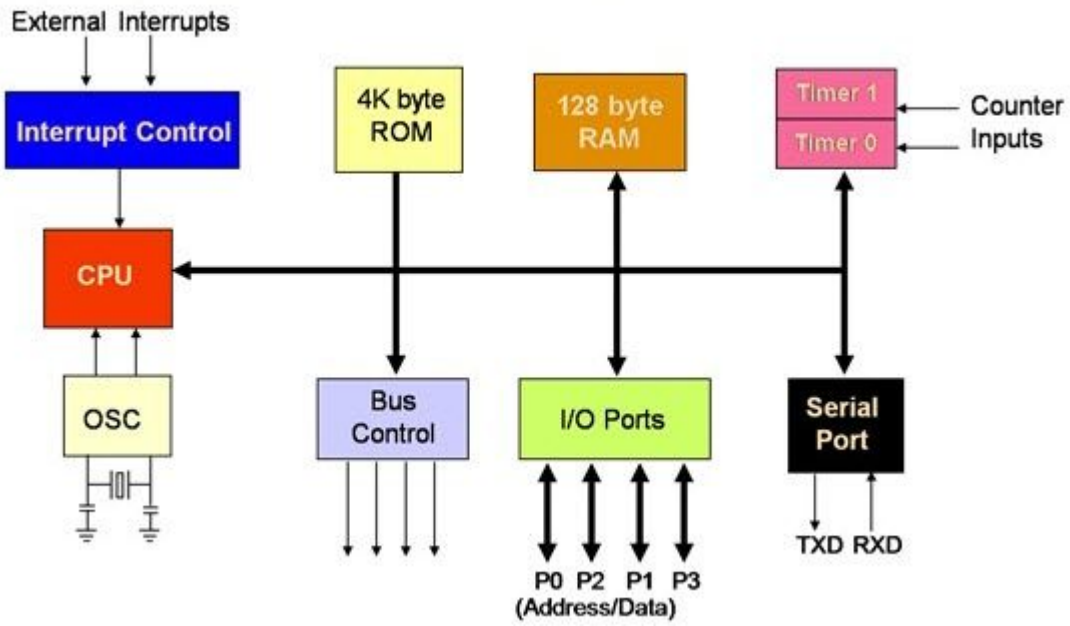
Mikrovezérlők

A mikrokontroller egy mikroprocesszor és további periféria-áramkörök egyetlen közös egységbe integrálva.

Első mikrovezérlő a Texas Instruments TMS 1000 1971-ben, 1974-ben került kereskedelmi forgalomba. Az Intel cég 1975-ben állította elő az első mikrokontrollert. a 8048-ast. Az MCS-48 sorozat tagjai módosított Harvard-architektúrájú csipek, amelyekben vagy belső ROM szolgál a program tárolására, vagy külső ROM-ot használhatnak erre a célra, és 64-256 bájtos belső (lapkára integrált) RAM-ot tartalmaznak. Az MCS-48 családban volt 8049 és 8050 -re elkeresztelt is ami több memóriát tartalmazott.

1980 -as fejlesztés az MCS-51-es sorozat ebből is a 8051. Nem volt sem pin kompatibilis sem binary kompatibilis a 8048-al, de a forrás lefordítható volt. A korai verziók NMOS technológiával készültek, mint az MCS-48, de később a kisebb energiafogyasztás miatt a CMOS terjedt el (1986 után). Mind a mai napig készülnek új fejlesztésű vezérlők 8051 utasításkészlettel, a fordítókra (IAR, Keil) is kiadnak mai napig frissítéseket.

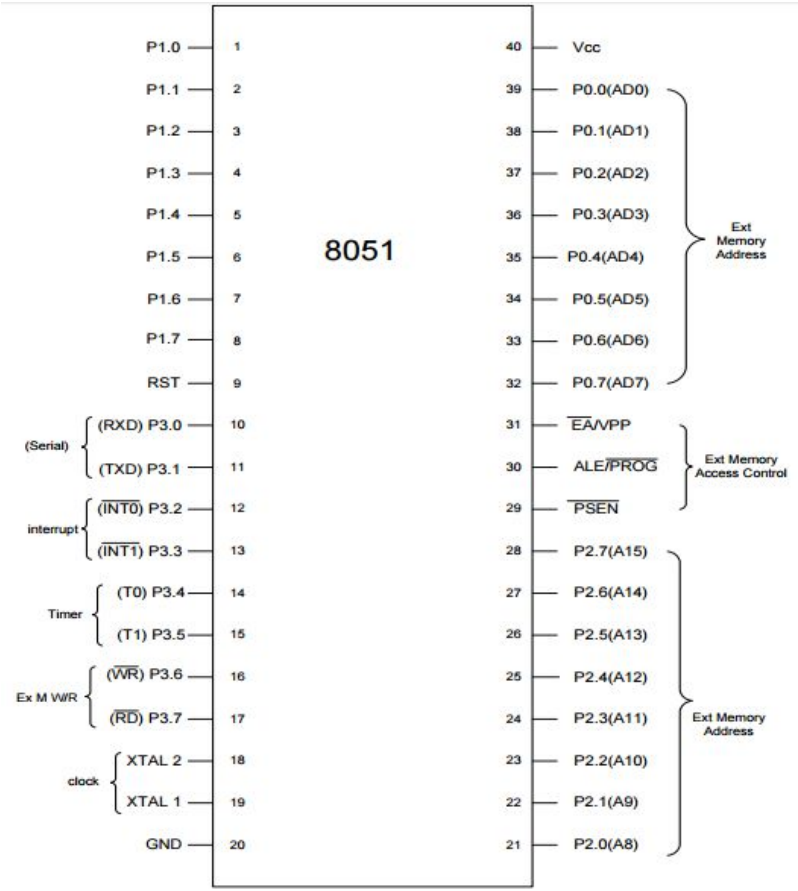
8051 felépítése



- megszakításvezérlő
- 4 Kbájt ROM
- 128 bájt RAM
- két 16 bites időzítő/számláló
- négy 8 bites párhuzamos port, 32 I/O vonalként
- sorosvonal-illesztő
- energiatakarékos mód

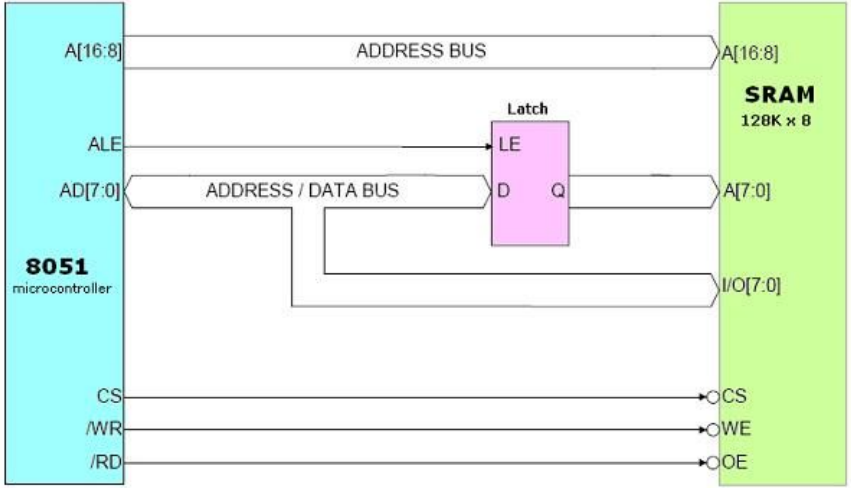
8051 hardver kiépítése

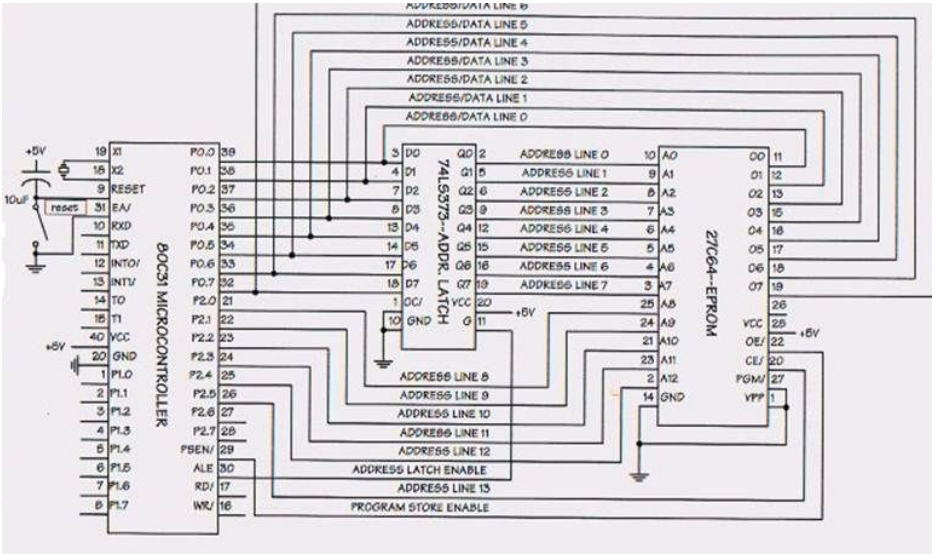
VCC	5v
GND	föld
P0	címbusz alsó 8 bitje, adatbusz
P1	általános IO
P2	címbusz felső 8 bit
P3	speciális IO
XTAL2 / XTAL1	oszcillátor
RST	reset
EA/VPP	extermal access 0 ha külső tároló van
ALE	a cím az ALE jellel írható a külső tárolóba
PSEN	program store enable
WR/RD	Az RD az olvasást, a WR pedig az írást vezérli.



Külső adat/program tároló használata esetén (pl. EPROM)

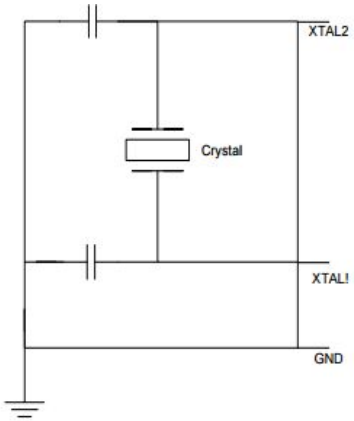
P0	egyfelől a címbusz alsó 8 bitje, másfelől az adatbusz
P2	címbusz felső 8 bitje
WR/RD	Az RD az olvasást, a WR pedig az írást vezérli. Mindkét jel 0-val aktív
ALE	Address Latch Enable, A cím az ALE jellel írható a külső tárolóba
PSEN	Program Store Enable, 0 a programtárolóhoz, 1 pedig az adatmemóriához szól a kontroller





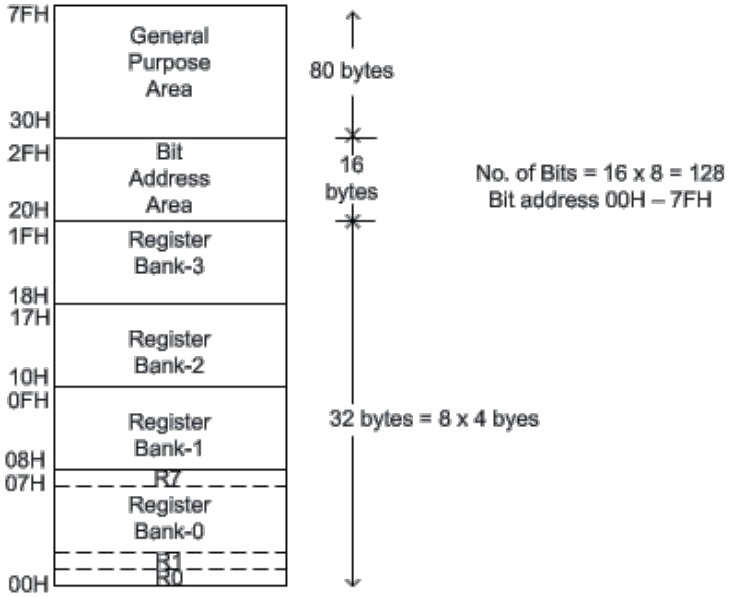
Órajel és oscillátor

- A működéshez szükséges egy 12 MHz -es külső oscillátor
- minden gépi ciklus 2 darab 6 oscillátor periódusú ütemből áll
- az első 6 oscillátor ütem beolvassa a következő kódot, a második 6 oscillátor ütem pedig végrehajtja
- minden művelet 1-3 bájt hosszú és 1 vagy 2 gépi ciklus alatt történik, kivétel a szorzás (MUL) és osztás (DIV), mert ezek 4 ciklus alatt hajtódnak végre



Memória

- Harvard architektúra, a programkód és az adatok külön fizikailag elkülönített úton közlekednek
- 256 bájt belső memória, aminek az alsó 128 bájtot használhatja a program adattárolásra
- az alsó 128 bájt (0x00 - 0x7F) első 32 bájtja 4 regiszter bank-ot tartalmaz
- minden regiszter bank 8 regisztert tartalmaz (R0-R7)
- a PSW regiszter 2 bitje határozza meg az aktuális bank számát, 0 az alapértelmezett
- a második 128 bájt az SFR (Special Function Register) (0x80 - 0xFF), ez a terület tartalmazza az akkumlátort, a B regisztert, a P0-P3- ig az IO regisztereket és több egyéb funkcióval bíró regisztereket



Energiafogyasztás, üzemmódok

- A fogyasztás függ az órajel frekvenciájától, kissebb frekvencia alacsonyabb fogyasztást jelent, a konkrét értékek adatlapból olvashatóak ki
- Power-down üzemmód
  A Power-down üzemmódban a belső oszcillátor leáll, ezáltal az összes belső funkció is megszűnik.
  A belső RAM tartalma azonban nem változik, vagyis az éppen aktuális értékek megmaradnak.
- Idle üzemmód
  Ekkor - a Power-down móddal ellentétben - az órajel tovább működik, a CPU viszont nem működik. Az időzítő, a megszakítás továbbra is működik

Energiafogyasztás vezérlő ***0x97*** **PCON** *(Power Control Register)*

két alapvető energiatakarékos mód áll rendelkezésre, idle és power down

SMOD	-	-	-	GF1	GF0	PD	IDL

SMOD Dupla baud sebesség bit. Ha a Timer 1 állítja elő a soros vonal órajelét és az SMOD=1, akkor a soros vonal sebessége duplázódik.

GF1 Általános célú bit.

GF0 Általános célú bit.

PD Ennek a bitnek az 1-be állítása aktiválja a Power Down üzemmódot

IDL Ennek a bitnek az 1-be állítása aktiválja az Idle Mode üzemmódot

Amennyiben a PD és az IDL bit egyidõben 1-es értékû, a PD-nek van elsőbbsége.

Időzítő és számláló üzemmódok

- Időzítő módban minden gépi ciklusban lépteti a számlálót
- Számláló módban egy külső megszakítás lépteti a számlálót, mintavételezés gépi ciklusonként történik, egy magas szintet egy alacsonynak kell követni, a lefutó él lépteti a számlálót
- Timer0 és Timer1 alábbi módoikban működhet

M1	M0	
0	0	(0-ás üzemmód) 13 bites számláló
0	1	(1-es üzemmód) 16 bites számláló
1	0	(2-es üzemmód) 8 bites, autómatikus újratöltésű
1	1	(3-as üzemmód) a TL0 és TH0 egymástól függetlenül dolgoznak

Időzítő mód beállító ***0x89*** **TMOD** *(Timer/Counter Mode Control Register)*

A T0 és Tl időzítő/számlálók különböző üzemmódjai állíthatók be a TMOD regiszter egyes bitjeivel

T1 GATE	T1 C/T	T1 M1	T1 M0	T0 GATE	T0 C/T	T0 M1	T0 M0

GATE Kapuzás. Ha ez a bit 1-es, akkor a számláló ki- illetve bekapcsolása az bemenetre adott jellel végezhető

(ha az magas, akkor a számláló mûködik). Ha ez a bit 0, akkor a számláló a TCON regiszterben lévő TR bittel kapcsolható be, vagy ki.

C/T Számláló, vagy idõzítõ üzemmód választás: 1 **=** számláló, 0 **=** időzítő.

M0	M1	
0	0	A TH 8 bites idõzítõ/számláló, a TL 5 bites elõosztóként mûködik.
0	1	A TH és a TL 16 bites számláló/idõzítõt alkot, elõosztó nincs.
1	0	A TL 8 bites idõzítõ/számlálóként mûködik, túlcsorduláskor automatikusan újratöltõdik a TH-ból.
1	1	(Timer 0) TL0 8 bites idõzítõ/számláló a Timer 0 vezérlõ bitjeivel normál módon vezérelve, a TH0 pedig 8 bites idõzítõ a Timer 1 vezérlõ bitjeivel vezérelve. (Timer 1) A számláló leáll, tartja az értékét.

Időzítő vezérlő ***0x88*** **TCON** *(Timer/Counter Control Register)*

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1	TCON.7	Timer 1 túlcsordulás jelző bit. A hardver 1-esbe állítja amikor a számláló túlcsordul. A hardver törli, amikor a megszakítás elkezdődik.
TR1	TCON.6	Timer 1 ki/bekapcsolás. Beállításával illetve törlésével az időzítő/számláló elindítható/leállítható
TF0	TCON.5	Timer 0 túlcsordulás jelző bit. A hardver 1-esbe állítja amikor a számláló túlcsordul. A hardver törli, amikor a megszakítás elkezdődik.
TR0	TCON.4	Timer 0 ki/bekapcsolás. Beállításával illetve törlésével az időzítő/számláló elindítható/leállítható
IE1	TCON.3	Megszakítás 1 él jelző. A hardver beállítja, amikor a külső megszakítás-kérés bemeneten detektált egy negatív élt. A megszakítás végrehajtásakor törlődik.
IT1	TCON.2	Megszakítás 1 vezérlő bitje. Ha beállítjuk, akkor a bemeneten megjelenő lefutó él fog megszakítást kiváltani. Ha töröljük, akkor az alacsony szint fog megszakítást generálni.
IE0	TCON.1	Megszakítás 0 él jelző. A hardver beállítja, amikor a külső megszakítás-kérés bemeneten detektált egy negatív élt. A megszakítás végrehajtásakor törlődik.
IT0	TCON.0	Megszakítás 0 vezérlő bitje. Ha beállítjuk, akkor a bemeneten megjelenő lefutó él fog megszakítást kiváltani. Ha töröljük, akkor az alacsony szint fog megszakítást generálni.

Program státusz ***0xD0*** **PSW** *(Program Status Word)*

különböző jelzőbitek vannak, amelyek többek között meghatározott programugrások feltételeiben szerepelhetnek

CY	AC	F0	RS1	RS0	OV	-	P

CY	PSW.7		Átvitel Carry flag, az összeadás átvitelekor ill. a kivonás áthozatalakor íródik 1-be
AC	PSW.6		Közbenső átvite jelzőbit, a BCD műveletek használják
F0	PSW.5		Program szabadon használható jelző bit
	RS1	RS0	
	0	0	Bank 0 (0x0000 - 0x0007)
	0	1	Bank 1 (0x0008 - 0x000F)
	1	0	Bank 2 (0x0010 - 0x0017)
	1	1	Bank 3 (0x0018 - 0x001F)
OV	PSW.2		Aritmetikai túlcsordulás jelzőbit, 1-be íródik, ha az összeadás vagy a kivonás kívül esik -128 és +127 tartományon
P	PSW.1		Paritás bit, 1-be íródik, ha az akkumlátorban páros számú 1-es van és törlődik, ha páratlan számú minden gépi cklusban újraszámolódik

Soros vonali illesztő beállítása ***0x98*** **SCON** *(Serial Port Control Register)*

Soros port üzemmód beállítása, az adat regiszter *0x99* SBUF

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0	SM1	
0	0	shift regiszter baud rate fosc/12
0	1	8 bit uart baud rate változtatható
1	0	9 bit UART baud rate fosc/32 vagy fosc/64
1	1	9 bit UART baud rate változtatható
SM2	SCON.5	9 bit UART üzemmódokban engedélyezi a multiprocesszor kommunikáció támogatását.
REN	SCON.4	vétel engedélyező bit
TB8	SCON.3	9 bit UART esetén a küldött adat 9. bitje
RB8	SCON.2	9 bit UART esetén a vett adat 9. bitje
TI	SCON.1	küldés megszakításvezérlő beállítása
RI	SCON.0	fogadás megszakításvezérlő beállítása

Soros vonali illesztő baudrate beállítása

SM0=0 és SM1=0 esetén rögzített: az oszcillátor frekvencia 12-ed része.

SM0=0 és SM1=1 esetén a sebességet a PCON regiszter SMOD bitje határozza meg, ha az SMOD bit 1, akkor a baud rate az oszcillátor frekvencia 32-ed része, ha pedig 0, akkor az oszcillátor frekvencia 64-ed része.

SM0=1 esetén a a sebességet a Timer 1 határozza meg. A Timer 1 túlcsordulásaiból adódó frekvenciát a processzor elosztja 32-vel, így áll elõ a soros vonal baud rate-je. Ekkor a Timer 1-et 8 bites idõzítõ üzemmódba állítjuk, automatikus újra feltöltéssel, a megszakítást pedig letiltjuk (a TMOD regiszter felsõ 4 bitje: 0010). Ekkor a sebességet a következõ formulával számíthatjuk:

$$baud\ rate = \frac{K\ x\ Oscillator\ freq}{32\ x\ 12\ x\ [256\ (TH1)]}$$

Ahol K SMOD =1 esetén 2 ellenben 1

Megszakítások ***0xA8*** **IE** *(Interrupt Enable Register)*

5 megszakítást kezel, 2 külső, 2 időzítő és 1 soros vonali megszakítást

EA	-	-	ES	ET1	EX1	ET0	EX0

EA	IE.7	Minden megszakítást engedélyez, letilt. Ha ez a bit 0, akkor egyetlen megszakítás sem jut érvényre. Ha ez a bit 1, akkor a megszakítások egyenként tilthatók, engedélyezhetõk.
ES	IE.4	A soros port megszakításainak engedélyezése, tiltása.
ET1	IE.3	Timer 1 megszakítás engedélyezõ bit.
EX1	IE.2	Külsõ megszakítás 1 engedélyezõ bit.
ET0	IE.1	Timer 0 engedélyezõ bit.
EX0	IE.0	Külsõ megszakítás 0 engedélyezõ bit.

Megszakítások, prioritásának beállítása ***0xB8*** **IP** *(Interrupt Priority Register)*

Az 5 megszakításnak prioritás szintje állítható be

-	-	-	PS	PT1	PX1	PT0	PX0

PS IP.4 A soros port megszakításainak prioritása.
PT1 IP.3 Timer 1 megszakítás prioritás.
PX1 IP.2 Külsõ megszakítás 1 prioritás.
PT0 IP.1 Timer 0 megszakítás prioritás.
PX0 IP.0 Külsõ megszakítás 0 prioritás.

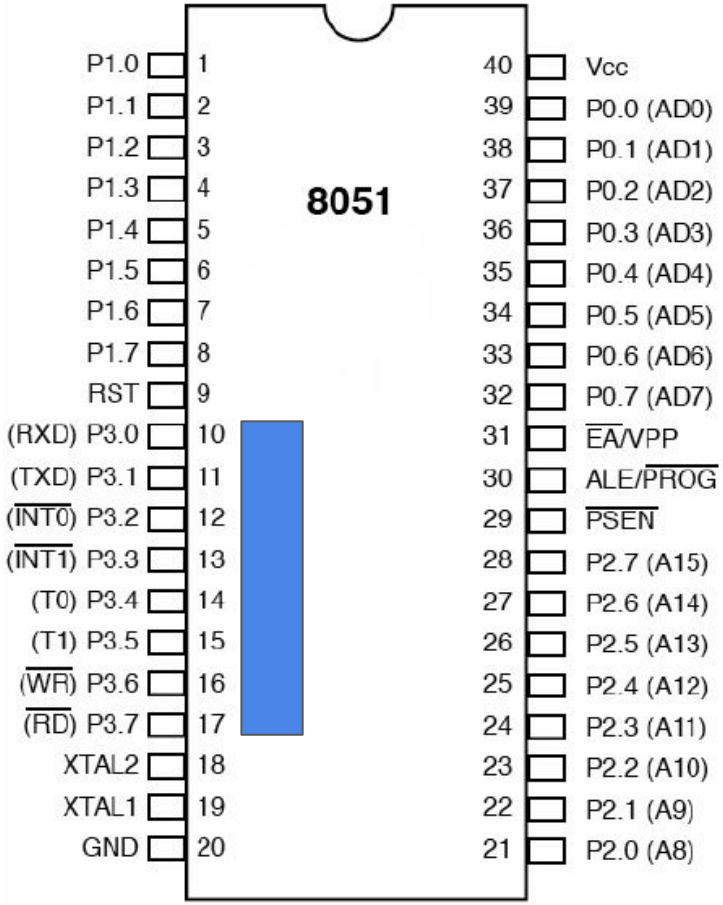
Ha a megfelelő bitet 1-be állítjuk, akkor az adott megszakítás magas szintű lesz, ha 0-ba, akkor, pedig alacsony szintű. Az alacsony szintű megszakításokat a magas szintű megszakítások megszakíthatják, míg a magas szintűeket semmi sem. Ha egyszerre érkezik a CPU-hoz egy magas és egy alacsony szintű megszakítás-kérés, akkor a magas szintűnek van elsőbbsége. Ha két egyforma szintű megszakítás-kérés érkezik, akkor a prioritási sorrend a következő: Külső megszakítás 0, Timer 0, Külső megszakítás 1, Timer 1, Soros illesztő

IO port 3 ***0xB0*** **P3**

a 3-as IO port ki és bemeneteinek funkciói

RD	WR	T1	T0	INT1	INT0	TXD	RXD

P3.7 RD Külső memória olvasó jel
P3.6 WR Külső memória író jel
P3.5 T1 Timer1 bemenet
P3.4 T0 Timer0 bemenet
P3.3 INT1 Külső megszakítás
P3.2 INT0 Külső megszakítás
P3.1 TXD soros vonal kimenet
P3.0 RXD soros vonal bemenet



SDCC, Small Device C Compiler

Az SDCC fordító egy ANSI C fordító, amely jól optimalizálható és több célprocesszor meghatározható számára, mikrovezérlők programozására használható. Nyílt forráskódú.

- Intel MCS51 8031, 8032, 8051, 8052 stb.
- Maxim DS80C390; ezek variánsai
- Freescale (korábban Motorola) HC08 based (hc08, s08)
- Zilog Z80 MCUs (z80, z180, gbz80, Rabbit 2000/3000, Rabbit 3000A, TLCS-90)
- STMicroelectronics STM8

SDCC letöltése

https://sourceforge.net/projects/sdcc/files/

más platformra letölthető
a jelenleg elérhető stabil verzió 3.6
ISO C11 szabványt is támogatja

Home / Browse / Development / Compilers / Small Device C Compiler suite / Files

SDCC **Small Device C Compiler suite**

Brought to you by: benshi, bernhardheld, borutr, dhelton, and 18 others

Summary | Files | Reviews | Support | Wiki | Mailing Lists | Tickets | News | Discussion

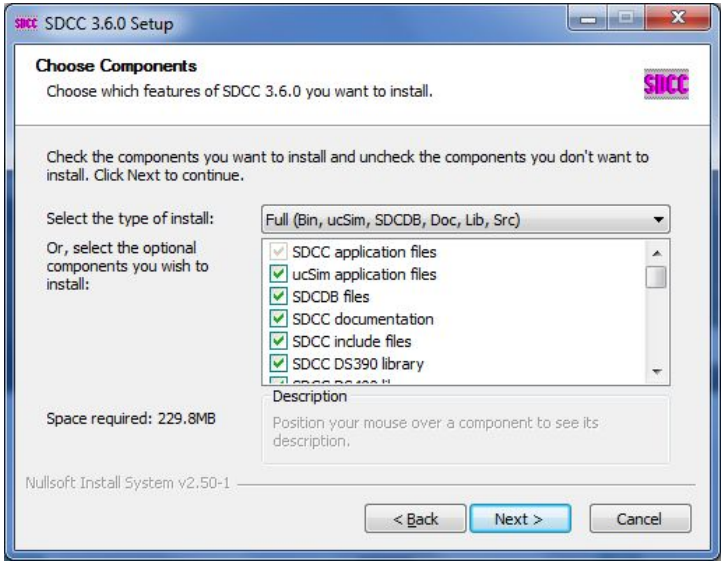
Looking for the latest version? **Download sdcc-3.6.0-x64-setup.exe (6.9 MB)**

Home

Name	Modified	Size	Downloads / Week
sdcc-macosx	2016-05-30		25
sdcc-linux-x86	2016-05-30		215
sdcc-win64	2016-05-30		419
sdcc-win32	2016-05-30		159

SDCC telepítése, fordítás

- pár kattintással feltelepíthető, kicsomagolva ~240Mb, amiből a PIC16 non-free könyvtárak 150Mb
- nem szükségesek környezeti változók beállítása a működéshez
- 8051 forrás lefordítása az alábbi kapcsolókkal történik:
  sdcc -mmcs51 main.c
  ezután még egy átalakítás szükséges
  packihx main.ihx > main.hex



8051 Header állomány SDCC esetén

```
/* BYTE Register */
__sfr __at (0x80) P0   ;
__sfr __at (0x81) SP   ;
__sfr __at (0x82) DPL  ;
__sfr __at (0x83) DPH  ;
__sfr __at (0x87) PCON ;
__sfr __at (0x88) TCON ;
__sfr __at (0x89) TMOD ;
__sfr __at (0x8A) TL0  ;
__sfr __at (0x8B) TL1  ;
__sfr __at (0x8C) TH0  ;
__sfr __at (0x8D) TH1  ;
__sfr __at (0x90) P1   ;
__sfr __at (0x98) SCON ;
__sfr __at (0x99) SBUF ;
__sfr __at (0xA0) P2   ;
__sfr __at (0xA8) IE   ;
__sfr __at (0xB0) P3   ;
__sfr __at (0xB8) IP   ;
__sfr __at (0xD0) PSW  ;
__sfr __at (0xE0) ACC  ;
__sfr __at (0xF0) B    ;
```

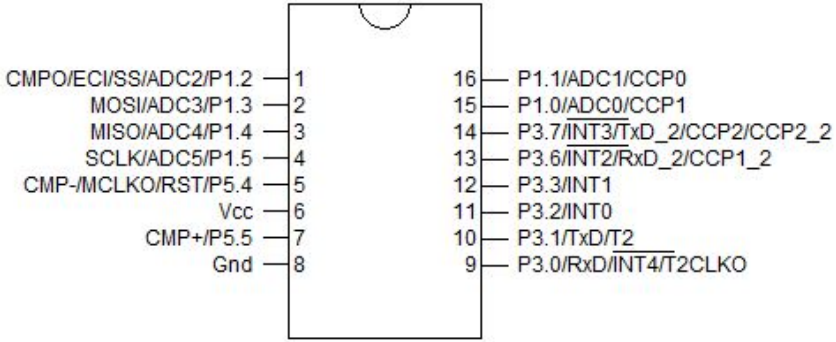
```
/* TCON */
__sbit __at (0x88) IT0  ;
__sbit __at (0x89) IE0  ;
__sbit __at (0x8A) IT1  ;
__sbit __at (0x8B) IE1  ;
__sbit __at (0x8C) TR0  ;
__sbit __at (0x8D) TF0  ;
__sbit __at (0x8E) TR1  ;
__sbit __at (0x8F) TF1  ;
```

```
/* BIT Register */
/* P0 */
__sbit __at (0x80) P0_0 ;
__sbit __at (0x81) P0_1 ;
__sbit __at (0x82) P0_2 ;
__sbit __at (0x83) P0_3 ;
__sbit __at (0x84) P0_4 ;
__sbit __at (0x85) P0_5 ;
__sbit __at (0x86) P0_6 ;
__sbit __at (0x87) P0_7 ;
```

```
/* Interrupt numbers: address = (number * 8) + 3 */
#define IE0_VECTOR    0    /* 0x03 external interrupt 0 */
#define TF0_VECTOR    1    /* 0x0b timer 0 */
#define IE1_VECTOR    2    /* 0x13 external interrupt 1 */
#define TF1_VECTOR    3    /* 0x1b timer 1 */
#define SI0_VECTOR    4    /* 0x23 serial port 0 */
```

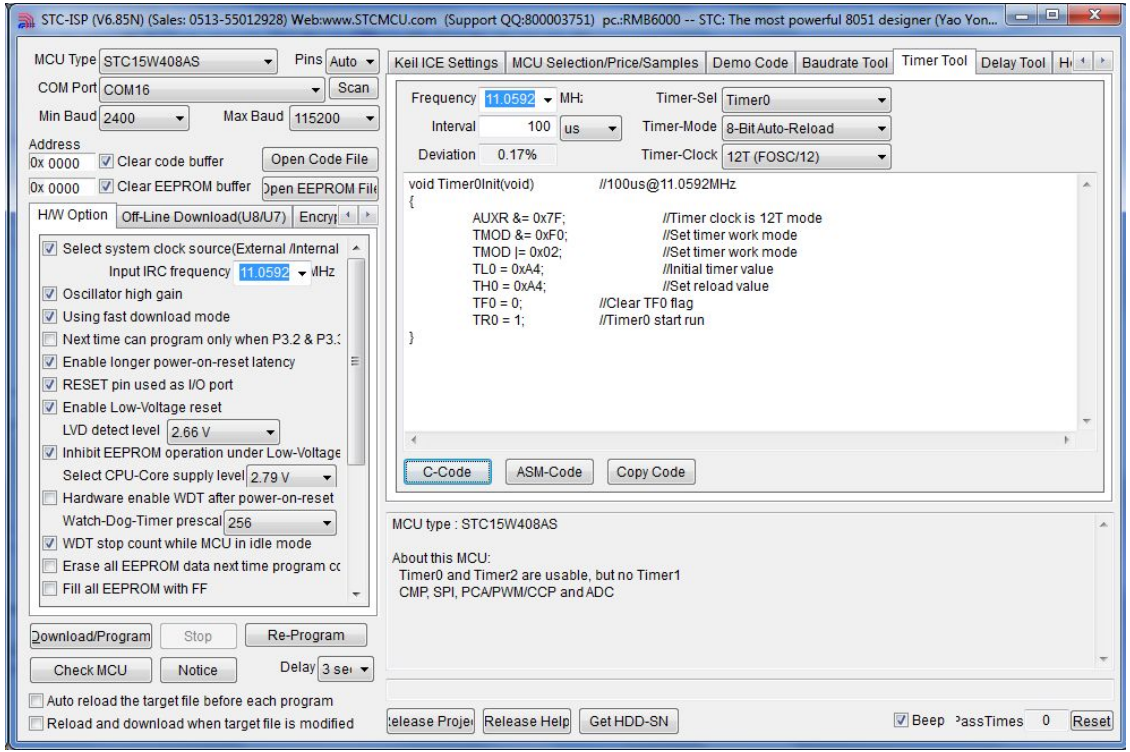
STC15W408AS

- több tokozásban elérhető DIP16, SOP20, SOP28
- 1 órajel 1 gépi ütem, így 12x gyorsabb mint a 8051
- A 15W sorozat 2.5 - 5.5 V között üzemel
- 5K EPROM, 8K FLASH, 512byte RAM
- 10bit ADC
- 1 UART, 1 SPI
- Timer0 és Timer2, kiterjesztett, de nincs Timer1



STC-ISP, az STC in-system programmer

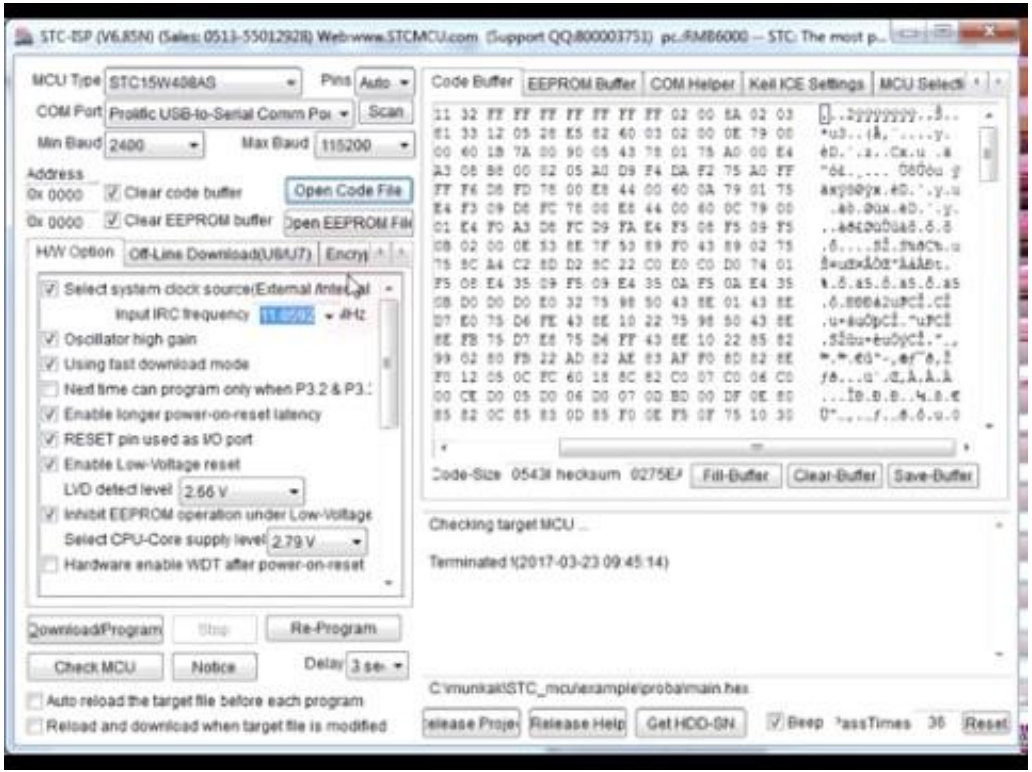
Az STC saját programozója több segítséget is ad, pl a timerek SFR regisztereit segít beállítani, van C és ASM kimenete is.
Az MCU beállításait is itt lehet megtenni, pl belső órajel generátor beállítása
A header fájlok letölthetőek, a lábkiosztás megnézhető

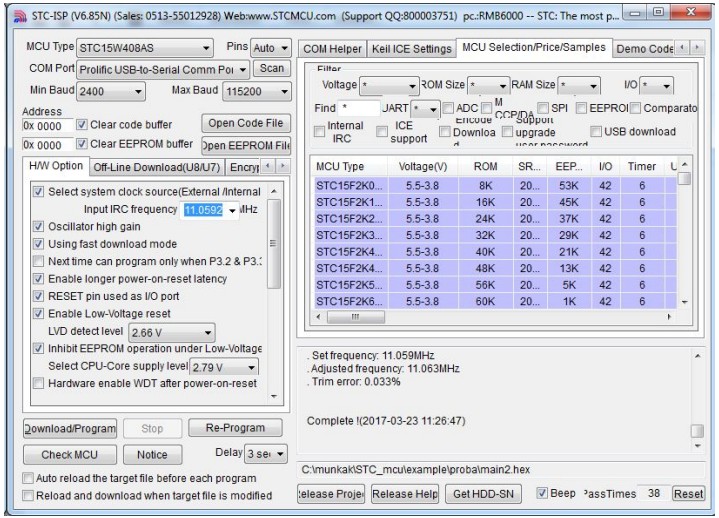


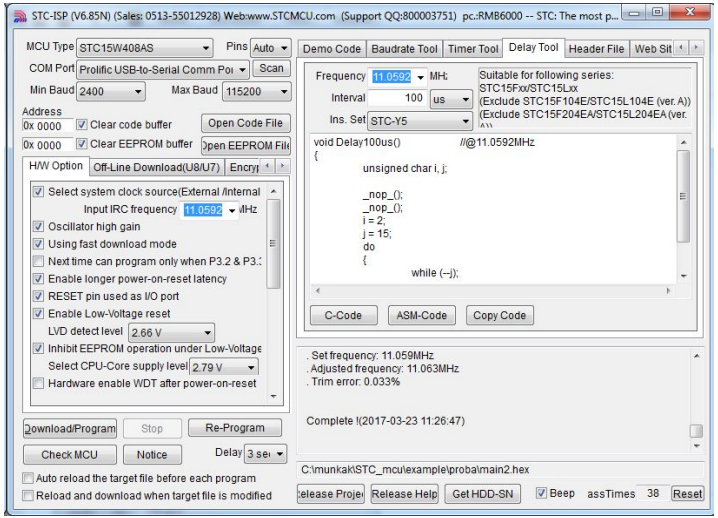
STC ISP használata

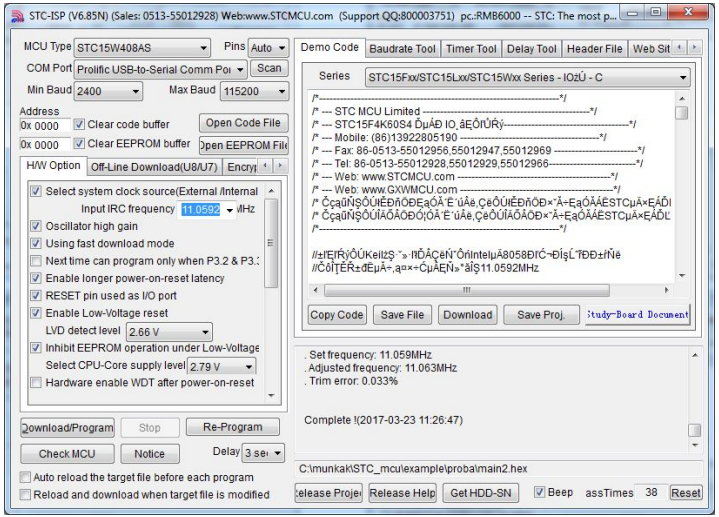
A 15W sorozatnál első használatkor be kell állítani a belső oscillátor frekvenciát. Ezt az első kódfeltöltés az STC-ISP programból megteszi. A fetöltéshez

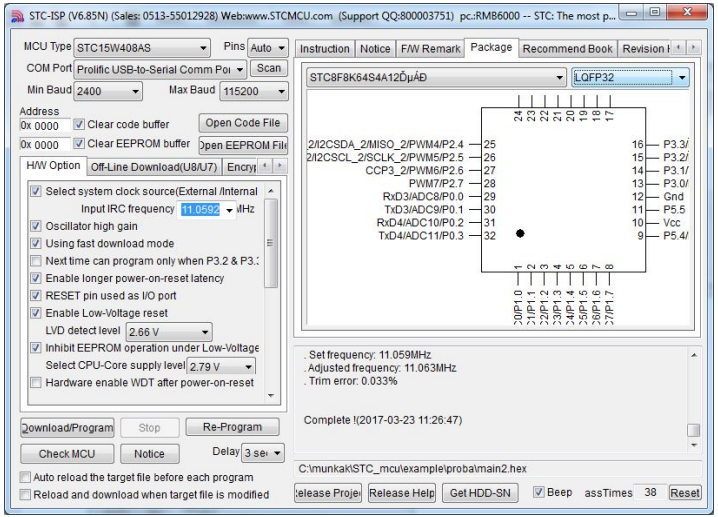
- ki kell választani a típust, MCU Type
- a portot, COM Port
- a feltöltendő kódot, Download/Program
- ezután az MCU-t resetelni kell, a jobb alsó panelen megjelenik a Checking Target MCU... felirat Sikeres feltöltés esetén a Complete! felirat jelenik meg




STC-ISP (V6.85N) (Sales: 0513-55012928) Web:www.STCMCU.com (Support QQ:800003751) pc.:RMB6000 -- STC: The most p...
MCU Type STC15W408AS
Pins Auto
COM Port Prolific USB-to-Serial Comm Por
Scan
Min Baud 2400
Max Baud 115200
Address
0x 0000
Clear code buffer
Open Code File
0x 0000
Clear EEPROM buffer
Open EEPROM File
H/W Option
Off-Line Download(U8/U7)
Select system clock source(External /Internal
Input IRC frequency 11.0592 MHz
Oscillator high gain
Using fast download mode
Next time can program only when P3.2 & P3.3
Enable longer power-on-reset latency
RESET pin used as I/O port
Enable Low-Voltage reset
LVD detect level 2.66 V
Inhibit EEPROM operation under Low-Voltage
Select CPU-Core supply level 2.79 V
Hardware enable WDT after power-on-reset
Download/Program
Stop
Re-Program
Check MCU
Notice
Delay 3 sec
Auto reload the target file before each program
Reload and download when target file is modified
COM Helper
Keil ICE Settings
MCU Selection/Price/Samples
Demo Code
Voltage
ROM Size
RAM Size
I/O
Find
Internal IRC
ICE support
ADC
SPI
EEPROM
USB download
MCU Type
Voltage(V)
ROM
EEP...
I/O
Timer
STC15F2K0... 5.5-3.8 8K 20... 53K 42 6
STC15F2K1... 5.5-3.8 16K 20... 45K 42 6
STC15F2K2... 5.5-3.8 24K 20... 37K 42 6
STC15F2K3... 5.5-3.8 32K 20... 29K 42 6
STC15F2K4... 5.5-3.8 40K 20... 21K 42 6
STC15F2K4... 5.5-3.8 48K 20... 13K 42 6
STC15F2K5... 5.5-3.8 56K 20... 5K 42 6
STC15F2K6... 5.5-3.8 60K 20... 1K 42 6
. Set frequency: 11.059MHz
. Adjusted frequency: 11.063MHz
. Trim error: 0.033%
Complete !(2017-03-23 11:26:47)
C:\munkak\STC_mcu\example\proba\main2.hex
Release Help
Get HDD-SN
Beep
assTimes 38
Reset


STC-ISP (V6.85N) (Sales: 0513-55012928) Web:www.STCMCU.com (Support QQ:800003751) pc.:RMB6000 -- STC: The most p...
MCU Type STC15W408AS
Pins Auto
COM Port Prolific USB-to-Serial Comm Por
Scan
Min Baud 2400
Max Baud 115200
Address
0x 0000
Clear code buffer
Open Code File
0x 0000
Clear EEPROM buffer
Open EEPROM File
H/W Option
Off-Line Download(U8/U7)
Demo Code
Baudrate Tool
Timer Tool
Delay Tool
Header File
Frequency 11.0592 MHz
Suitable for following series:
STC15Fxx/STC15Lxx
(Exclude STC15F104E/STC15L104E (ver. A))
(Exclude STC15F204EA/STC15L204EA (ver. A))
Interval 100 us
Ins. Set STC-Y5
void Delay100us() //@11.0592MHz
{
unsigned char i, j;
nop();
nop();
i = 2;
j = 15;
do
{
while (--j);
C-Code
ASM-Code
Copy Code
. Set frequency: 11.059MHz
. Adjusted frequency: 11.063MHz
. Trim error: 0.033%
Complete !(2017-03-23 11:26:47)
C:\munkak\STC_mcu\example\proba\main2.hex
Release Help
Get HDD-SN
Beep
assTimes 38
Reset


STC-ISP (V6.85N) (Sales: 0513-55012928) Web:www.STCMCU.com (Support QQ:800003751) pc.:RMB6000 -- STC: The most p...
MCU Type STC15W408AS
Pins Auto
COM Port Prolific USB-to-Serial Comm Por
Scan
Min Baud 2400
Max Baud 115200
Address
0x 0000
Clear code buffer
Open Code File
0x 0000
Clear EEPROM buffer
Open EEPROM File
H/W Option
Off-Line Download(U8/U7)
Demo Code
Baudrate Tool
Timer Tool
Delay Tool
Header File
Series STC15Fxx/STC15Lxx/STC15Wxx Series - IOzÚ - C
/*----------*/
/* --- STC MCU Limited ---*/
/* --- STC15F4K60S4 ÕµÁÐ IO,âÉÓÍÚÑÿ---*/
/* --- Mobile: (86)13922805190 ---*/
/* --- Fax: 86-0513-55012956,55012947,55012969 ---*/
/* --- Tel: 86-0513-55012928,55012929,55012966---*/
/* --- Web: www.STCMCU.com ---*/
/* --- Web: www.GXWMCU.com ---*/
/* ÇçaûÑŞÓÚíÉÐñÓÐEąÓÁ´È´úÂe,ÇèÓÚíÉÐñÓÐ×´Ã÷ĘąÓÁÂESTCµÁ×ĘÁÐI
/* ÇçaûÑŞÓÚíÍÁÓÃÓÐÐ0¡ÓÁ´È´úÂe,ÇèÓÚíÍÁÓÃÓÐ×´Ã÷ĘąÓÁÂESTCµÁ×ĘÁÐL
/*----------*/
//±!ĘÍŘÿÓÚKeilzŞ ˜»·IIÐÂÇèÁÑ´ÔñíIntelµÄ8058ÐíÇ¬ÐíşĹ´ÍÐÐ±íÑë
//ĆóÍŢÉŘ±dĘµÃ÷=,a¤×=ĆµÂĘÑ»*ãÍŞ11.0592MHz
Copy Code
Save File
Download
Save Proj.
Study-Board Document
. Set frequency: 11.059MHz
. Adjusted frequency: 11.063MHz
. Trim error: 0.033%
Complete !(2017-03-23 11:26:47)
C:\munkak\STC_mcu\example\proba\main2.hex
Release Help
Get HDD-SN
Beep
assTimes 38
Reset


STC-ISP (V6.85N) (Sales: 0513-55012928) Web:www.STCMCU.com (Support QQ:800003751) pc.:RMB6000 -- STC: The most p...
MCU Type STC15W408AS
Pins Auto
COM Port Prolific USB-to-Serial Comm Por
Scan
Min Baud 2400
Max Baud 115200
Address
0x 0000
Clear code buffer
Open Code File
0x 0000
Clear EEPROM buffer
Open EEPROM File
H/W Option
Off-Line Download(U8/U7)
Instruction
Notice
F/W Remark
Package
Recommend Book
STC8F8K64S4A12ÕµÁÐ
LQFP32
2/I2CSDA_2/MISO_2/PWM4/P2.4 — 25
2/I2CSCL_2/SCLK_2/PWM5/P2.5 — 26
CCP3_2/PWM6/P2.6 — 27
PWM7/P2.7 — 28
RxD3/ADC8/P0.0 — 29
TxD3/ADC9/P0.1 — 30
RxD4/ADC10/P0.2 — 31
TxD4/ADC11/P0.3 — 32
16 — P3.3/
15 — P3.2/
14 — P3.1/
13 — P3.0/
12 — Gnd
11 — P5.5
10 — Vcc
9 — P5.4/
. Set frequency: 11.059MHz
. Adjusted frequency: 11.063MHz
. Trim error: 0.033%
Complete !(2017-03-23 11:26:47)
C:\munkak\STC_mcu\example\proba\main2.hex
Release Help
Get HDD-SN
Beep
assTimes 38
Reset

blink

```
/*
P12        1 +--*--+  16  P11
P13        2 |  *  |  15  P10
P14        3 |     |  14  P37
P15 LED    4 |     |  13  P36
P54        5 |     |  12  P33
VCC        6 |     |  11  P32
P55        7 |     |  10  P31 TX
GND        8 +-----+   9  P30 RX
*/
```

```c
/*
#include "8051.h"
__sfr __at (0x8E)    AUXR;

volatile unsigned long timercounter=0;

void Timer0Init(void)          //100us@11.0592MHz
{
    AUXR &= 0x7F;    //Timer clock is 12T mode
    TMOD &= 0xF0;    //Set timer work mode
    TMOD |= 0x02;    //Set timer work mode
    TL0 = 0xA4;      //Initial timer value
    TH0 = 0xA4;      //Set reload value
    TF0 = 0;         //Clear TF0 flag
    TR0 = 1;         //Timer0 start run
}

void tm0_isr() __interrupt (TF0_VECTOR)
{
    timercounter++;  // a megszakítás
}                    // lépteti a számlálót

void main(void)
{
    unsigned long T_BLINK = 0;
    Timer0Init();
    ET0 = 1;         // Timer0 megszakításkérés
    EA = 1;          // Megszakítások engedélyezése
    while(1) {
        if ((timercounter - T_BLINK) > 5000)
        {
            T_BLINK = timercounter;
            P1_5=!P1_5;
        }
    }
}
```

az AUXR az STC kiegészítő regisztere

AUXR 7.bit, a Timer0 beállítása kompatibilis 8051 módra

TMOD Timer0 -ra vonatkozó rész kinullázása

TMOD 2.bit beállítja a 8bit-es újratöltődő módot

TCON TF0 jelzőbit törlése

TCON TR0 1-be állításával elindul az időzítő

3. AUXR : Auxiliary register (Address:8EH, Non bit-addressable)

SFR name	Address	bit	B7	B6	B5	B4	B3	B2	B1	B0
AUXR	8EH	name	T0x12	T1x12	UART_M0x6	T2R	T2_C/$\overline{T}$	T2x12	EXTRAM	S1ST2

B7 - T0x12 : Timer 0 clock source bit.

0 : The clock source of Timer 0 is SYSclk/12. It will compatible to the traditional 8051 MCU

1 : The clock source of Timer 0 is SYSclk/1. It will drive the T0 faster than a traditional 8051 MCU

B6 - T1x12 : Timer 1 clock source bit.

0 : The clock source of Timer 1 is SYSclk/12. It will compatible to the traditional 8051 MCU

1 : The clock source of Timer 1 is SYSclk/1. It will drive the T0 faster than a traditional 8051 MCU

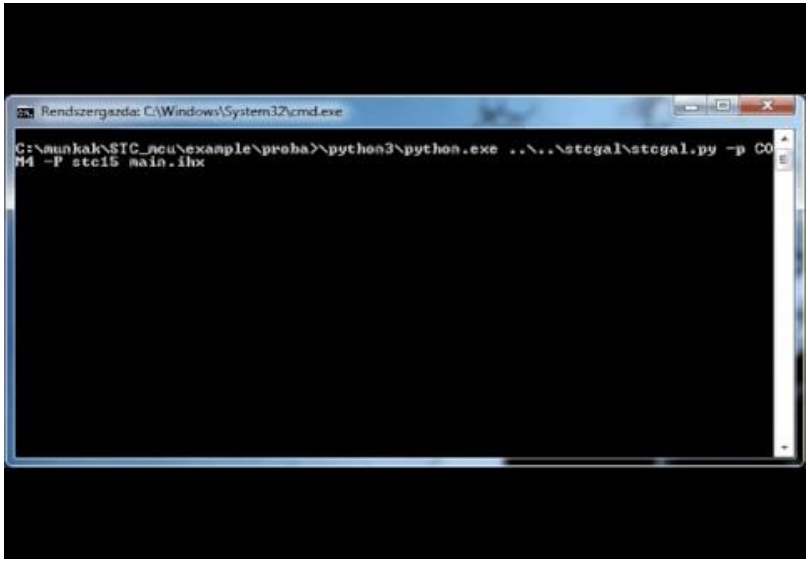
If T1 is used as the baud-rate generator of UART1, T1x12 will decide whether UART1 is 1T or 12T.

blink, fordítása, feltöltés stcgal.py segítségével

```
C:\Windows\system32\cmd.exe

c:\munkak\STC_mcu\example\proba>sdcc -mmcs51 main.c

c:\munkak\STC_mcu\example\proba>packihx main.ihx > main.hex
packihx: read 16 lines, wrote 22: OK.

c:\munkak\STC_mcu\example\proba>findstr "ROM Name" main.mem
   Name             Start    End      Size     Max
   ROM/EPROM/FLASH  0x0000   0x00f4     245    65536

c:\munkak\STC_mcu\example\proba>\python3\python.exe stcgal.py -p COM4 -P stc15 m
ain.ihx
```

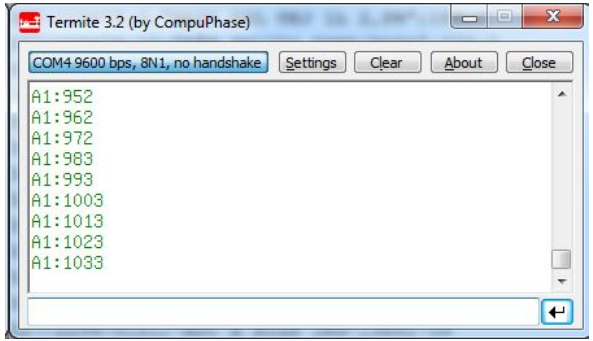
https://github.com/grigorig/stcgal

python3 kell hozzá, de működik portable verzióval is



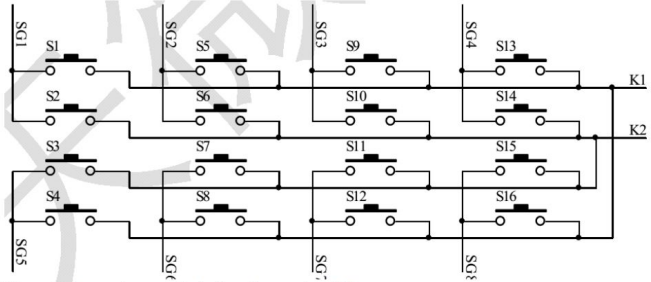
```c
#include "8051.h"

#define NOP() asm { 0x00 }
__sfr __at (0x8E)   AUXR;

__sfr __at (0xD6)   T2H;
  sfr   at (0xD7)   T2L;
void UartInit(void)          //9600bps@11.0592MHz
{
    SCON = 0x50;        //8bit and variable baudrate 0101 0000 SM1  REN
    AUXR |= 0x01;       //Use Timer2 as baudrate generator
    AUXR &= 0xFB;       //Timer2's clock is Fosc/12 (12T)
    T2L = 0xE8;         //Initial timer value
    T2H = 0xFF;         //Initial timer value
    AUXR |= 0x10;       //Timer2 running
}


void UART1_Send_Char(unsigned char CHAR)
{
    SBUF = CHAR;
    while (!TI);
    TI = 0;
}

void UART1_Send_String(unsigned char *STR)
{
    while (*STR != '\0') {
        UART1_Send_Char(*STR);
        STR++;
    }
}

void UART1_Send_Int(unsigned long dt)
{
    const unsigned char DEC_CODE[] = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9'};
    unsigned long i;
    for (i=10000;i>=1;i/=10)
    {
        if (dt>=i) UART1_Send_Char(DEC_CODE[(dt/i) % 10]);
    }
    if (dt==0) UART1_Send_Char(DEC_CODE[0]);
}
```

```c
void main(void)
{
    unsigned long T_BLINK = 0;
    unsigned long T_SERIAL = 0;
    unsigned long T_INC = 0;
    int a1=0;
    Timer0Init();
    UartInit();
    UART1_Send_String("Debug enabled\r\n");
    ET0 = 1;
    EA = 1;
    while(1) {
        if ((timercounter - T_BLINK) > 5000)
        {
            T_BLINK = timercounter;
            P1_5=!P1_5;
        }
        if ((timercounter - T_INC) > 1000)
        {
            T_INC = timercounter;
            a1++;
        }
        if ((timercounter - T_SERIAL) > 10000)
        {
            T_SERIAL = timercounter;
            UART1_Send_String("A1:");UART1_Send_Int(a1);
            UART1_Send_String("\r\n");
        }

    }
}
```



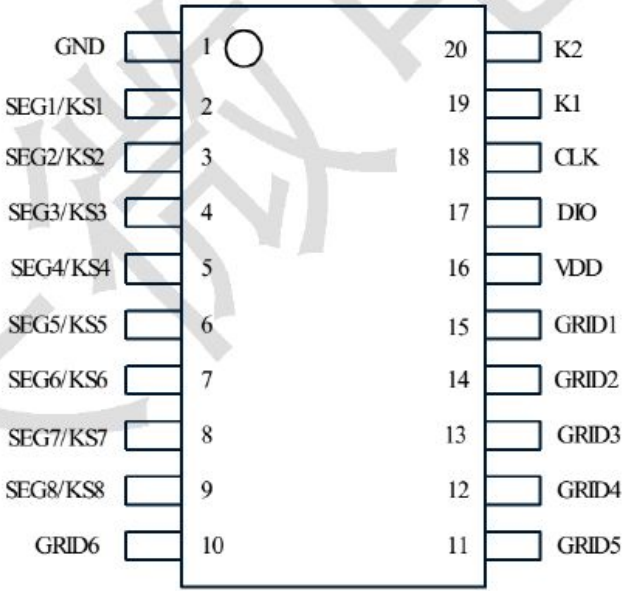
TM1637 a majdnem i2c -s 7 segment display driver

- 6 digites 7 (8) szegmenses kijelző meghajtó
- 8 x 2bites nyomógomb
- 8 fokozatban állítható fényerő
- villogó mód
- two wire protokoll



When the button is pressed, the key data read as follows:

	SG1	SG2	SG3	SG4	SG5	SG6	SG7	SG8
K1	1110_1111	0110_1111	1010_1111	0010_1111	1100_1111	0100_1111	1000_1111	0000_1111
K2	1111_0111	0111_0111	1011_0111	0011_0111	1101_0111	0101_0111	1001_0111	0001_0111



TM1637 adatlapja tartalmazza a program kódot is

```
Reference Program
/ *
* Copyright: Shenzhen Tian Microelectronics
* File name: TM1637
* Current Version: 1.0
* SCM Model: AT89S52
* Development Environment: Keil uVision3
* Crystal earthquake frequency: 11.0592M
* Program features: the TM1637 all filled with all the display data register
address 0xff, and open the display, and then read the key value.
* /
      # include <reg52.h>
      # include <intrins.h>
      / / Define port
      sbit clk = P1 ^ 2;
      sbit dio = P1 ^ 1;
      / / / ========================================
      void Delay_us (unsigned int i) // n us delay
      {
           for (; i> 0; i--)
           _nop_ ();
      }
      / / / ========================================
      void I2CStart (void) // 1637 start
      {
           clk = 1;
           dio = 1;
           Delay_us (2);
           dio = 0;
      }
      / / / ==============================================
      void I2Cask (void) // 1637 Answer
      {
           clk = 0;
           Delay_us (5); // After the falling edge of the eighth clock delay 5us,
ACK signals the beginning of judgment
           while (dio);
           clk = 1;
           Delay_us (2);
           clk = 0;
      }
      / / / ========================================
      void I2CStop (void) // 1637 Stop
      {
           clk = 0;
           Delay_us (2);
           dio = 0;
           Delay_us (2);
           clk = 1;
           Delay_us (2);
           dio = 1;
      }
      / / / ========================================
```

```
 void I2CWrByte (unsigned char oneByte) // write a byte
 {
      unsigned char i;
      for (i = 0; i <8; i + +)
      {
           Clk = 0;
           if (oneByte & 0x01) // low front
           {dio = 1;}
           else {dio = 0;}
           Delay_us (3);
           oneByte = oneByte >> 1;
           clk = 1;
           Delay_us (3);
      }
 }
 / / / ---------------------------------------------- -
unsigned char ScanKey (void) // read buttons \
{
      unsigned char rekey, rkey, i;
      I2CStart ();
      I2CWrByte (0x42); // read command buttons
      I2Cask ();
      dio = 1; // read keys before data lines pulled
      for (i = 0; i <8; i + +) // start reading from the low
      {
           Clk = 0;
           rekey = rekey >> 1;

           Delay_us(30);
           clk=1;
           if(dio)
           {
                rekey=rekey|0x80;
           }
           else
           {
                rekey=rekey|0x00;
           }
           Delay_us(30);
      }
      I2Cask();
      I2CStop();
      return (rekey);
}
```

```
#include "8051.h"
/*
P12       1 +--*--+  16  P11 SCL
P13       2 |  *  |  15  P10 SDA
P14       3 |     |  14  P37 R
P15  LED+ 4 |     |  13  P36 G
P54  JMP  5 |     |  12  P33 B
VCC       6 |     |  11  P32 -
P55       7 |     |  10  P31 TX
GND       8 +-----+   9  P30 RX
*/
#define clk        P1_1
#define dio        P1_0
#define r          P3_7
#define g          P3_6
#define b          P3_3
#define gnd        P3_2

#define _nop_() __asm nop __endasm
    
__sfr __at (0x8E)   AUXR; __sfr __at (0xD6)   T2H; __sfr __at (0xD7)   T2L;
__sbit __at(0xC8 + 4)   P5_4;

void Delay_us(unsigned int us) {while (us--) { _nop_();_nop_();_nop_();_nop_();_nop_(); }}

void dispint(unsigned int num,unsigned char bright,unsigned char point)
{
static const unsigned char pins[]={
    0x3f, 0x21, 0x5d, 0x75, 0x63, 0x76, 0x7e, 0x25, 0x7f, 0x77,
};

I2CStart();
I2CWrByte(0x40); //40H address is automatically incremented by 1 mode, 44H fixed address mode
I2Cask();
I2CStop();

I2CStart();
I2CWrByte(0xc0 | 0); // Set the first address
I2Cask();

if (num<1000) I2CWrByte(0); else I2CWrByte( pins[ (num/1000)%10 ] ); I2Cask();
if (num<100 ) I2CWrByte(0); else I2CWrByte( pins[ (num/100 )%10 ] ); I2Cask();
if (num<10  ) I2CWrByte(0); else I2CWrByte( pins[ (num/10  )%10 ] ); I2Cask();
I2CWrByte(pins[ num%10 ] | ((point==1) ? (1 << 7) : 0)); I2Cask();
I2CStop();

I2CStart();
I2CWrByte(0x88 + bright); // Open display, set brightness
I2Cask();
I2CStop();
}
```

```
void main(void)
{
    unsigned long T_DISP = 1000;
    unsigned long T_BLINK = 0;
    unsigned long T_SERIAL = 0;
    unsigned long T_INC = 0;
    unsigned long T_RGB = 0;
    int a1=0; int a2=0; P5_4 = 1; P1_5 = 0; gnd=0;
    Timer0Init();
    UartInit();
    UART1_Send_String("Debug enabled\r\n");
    ET0 = 1; EA = 1;
    while(1) {
        if ((timercounter - T_DISP) > 1000)
        {
            T_DISP = timercounter;
            if (P5_4) dispint(a1,5,0);
        }
        if ((timercounter - T_BLINK) > 10000)
        {
            T_BLINK = timercounter;
            P1_5=!P1_5;
        }
        if ((timercounter - T_INC) > 7000)
        {
            T_INC = timercounter;
            a1++;
        }
        if ((timercounter - T_RGB) > 7000)
        {
            T_RGB = timercounter;
            switch (a2++%3)
            {
                case 0: r=1; g=0; b=0; break;
                case 1: r=0; g=1; b=0; break;
                case 2: r=0; g=0; b=1; break;
            }
        }
        if ((timercounter - T_SERIAL) > 20000)
        {
            T_SERIAL = timercounter;
            UART1_Send_String("A1:");UART1_Send_Int(a1%8);
            UART1_Send_String("\r\n");
        }
    }
}
```