ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Προγραμματισμός εμπεδωμένου συστήματος

Προγραμματισμός εμπεδωμένου συστήματος

ΧΑΝΙΑ
ΦΕΒΡΟΥΑΡΙΟΣ 2006

Προγραμματισμός εμπεδωμένου συστήματος

Εκπόνηση

Μαυρουδής Γεώργιος

Όνομα Πατρός: Ιωάννης
Α.Μ: 2457
Τηλέφωνο : 6942257913
e-mail: gim@ics.forth.gr

Επιβλέπων καθηγητής : Αντωνιδάκης Εμμανουήλ

ΧΑΝΙΑ
ΦΕΒΡΟΥΑΡΙΟΣ 2006

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

Περίληψη:

Στην παρούσα πτυχιακή εργασία περιγράφεται ο τρόπος προγραμματισμού αναπτυξιακού συστήματος μικροεπεξεργαστή της οικογένειας 8051 χρησιμοποιώντας ολοκληρωμένο περιβάλλον σχεδίασης (IDE) με τη γλώσσα προγραμματισμού C. Ακολουθεί ανάπτυξη εφαρμογής για την οδήγηση κινητήρων συνεχούς ρεύματος- με διεξοδική ανάπτυξη της μεθόδου και του αλγόριθμου που χρησιμοποιήθηκαν για την παραγωγή των απαραιτήτων σημάτων οδήγησης γέφυρας ημιαγωγών .Τέλος παρουσιάζονται η γέφυρα καθώς και τα συμπεράσματα από τα πειράματα και τη λειτουργία της κατασκευής.

In this project the 8051 family's microprocessor programming is described, using Integrated designing environment (IDE) through C programming language. The application's development comes next, to lead the direct current engines, (including the algorithm used to lead them through semiconductors bridge). Finally the bridge, the trial run, and the programming results are presented.

Εισαγωγή

Τα εμπεδωμένα συστήματα-Embedded systems

Ένα εμπεδωμένο σύστημα είναι ένα ειδικής χρήσης σύστημα υπολογιστή το οποίο συνήθως περιέχεται και στη συσκευή που ελέγχει. Έχουν συγκεκριμένες απαιτήσεις και εκτελούν προκαθορισμένες εργασίες σε αντίθεση με του υπολογιστές γενικού σκοπού. Τα εμπεδωμένα συστήματα είναι προγραμματισμένα συστήματα. Η "πρώτη ύλη" είναι ένα προγραμματιζόμενο chip προγραμματισμένο με συγκεκριμένη εφαρμογή. Αυτό πρέπει να γίνει κατανοητό σε σύγκριση με τα παλαιότερα συστήματα με πλήρες υλικό σύστημα (hardware) ή συστήματα με υλικό (hardware) γενικού σκοπού και λογισμικό που είναι εξωτερικά αποθηκευμένο . Τα εμπεδωμένα συστήματα είναι συνδυασμός υλικού και λογισμικού που διευκολύνουν τη μαζική παραγωγή και αυξάνουν την ποικιλία των εφαρμογών.

Ιστορικά

Το πρώτο ευδιάκριτο σύγχρονο ενσωματωμένο σύστημα ήταν ο υπολογιστής καθοδήγησης της διαστημικής αποστολής απόλλων, που αναπτύχθηκε από τον Charls Stark Draper στο εργαστήριο οργανολογίας του MIT. Κάθε πτήση στο φεγγάρι είχε δύο. Χειρίζονταν τα συστήματα αδρανειακής καθοδήγησης της ενότητας εντολών και του LEM (Lunar Excursion Module-Μονάδα σεληνιακής περιήγησης).

Στην έναρξη του προγράμματος, ο υπολογιστής καθοδήγησης του απόλλων θεωρήθηκε το πιο επισφαλή σημείο του. Η χρησιμοποίηση των καινούριων μονολιθικών ολοκρηρωμένων για μείωση του βάρους και του όγκου αύξανε αυτόν τον κίνδυνο.

Υπολογιστής καθοδήγησης απόλλων.
πηγή: Από μουσείο ιστορίας υπολογιστών

Παραδείγματα των ενσωματωμένων συστημάτων

Αυτόματες ταμιακές μηχανές (ATMs).

Αεροναυτική ηλεκτρονική, όπως αδρανειακά συστήματα καθοδήγησης,

Συστήματα ελέγχου πτήσης/ λογισμικό και άλλα εμπεδωμένα συστήματα μέσα στα αεροσκάφη και οπλικά συστήματα.

Κυψελοτή τηλεφωνία και τηλεφωνικοί μεταγωγής .

Εξοπλισμός δικτύων υπολογιστών, δρομολογητές, timeservers και συστήματα προστασίας εισόδου (firewall) .
Εκτυπωτές υπολογιστών

Αντιγραφείς

Συσκευές αποθήκευσης (δισκετών και σκληρών δίσκων)

Εγκέφαλοι μηχανών εσωτερικής καύσης και ελεγκτές αντιμπλοκαρίματος φρένων για τα αυτοκίνητα

Προϊόντα οικιακής αυτοματοποίησης, όπως θερμοστάτες, κλιματιστικά μηχανήματα, ψεκαστήρες, και συστήματα ελέγχου ασφάλειας

Φορητοί υπολογιστές

Οικιακές συσκευές, φούρνοι μικροκυμάτων, πλυντήρια ρούχων, συστήματα τηλεόρασης , DVD / όργανα καταγραφής
Ιατρικός εξοπλισμός

Εξοπλισμός μετρήσεων όπως παλμογράφοι μεψηφιακή αποθήκευση , λογικοί αναλυτές, και συσκευές ανάλυσης φάσματος

Ρολόγια χειρός με διάφορες λειτουργίες

Πολυσύνθετοι εκτυπωτές (MFPs)

Προσωπικοί ψηφιακοί βοηθοί (PDAs), δηλαδή μικροί φορητοί υπολογιστές με PIMs και άλλες εφαρμογές.

Κινητά τηλέφωνα με τις πρόσθετες ικανότητες, παραδείγματος χάριν, κινητοί ψηφιακοί βοηθοί με το ενσωματωμένο PDA και Java (MIDP)

Προγραμματιζόμενοι λογικοί ελεγκτές (PLCs) για βιομηχανικούς αυτοματισμούς και έλεγχο.

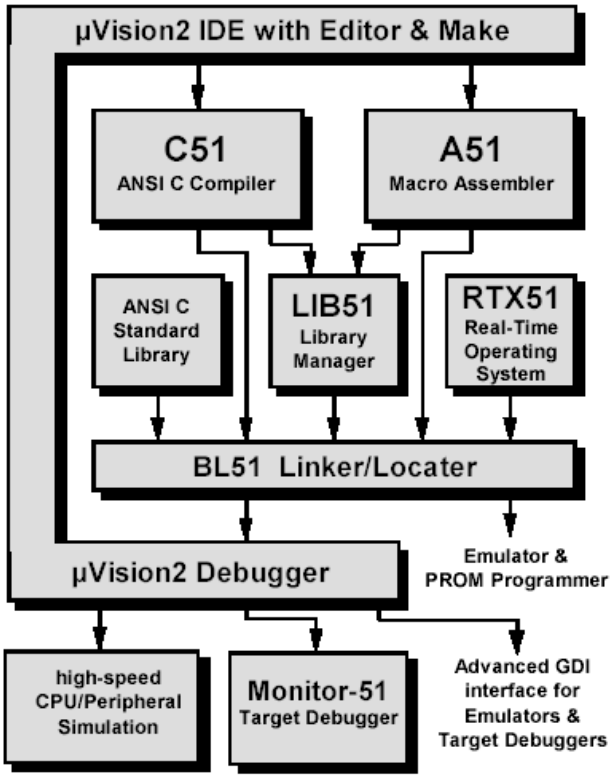
Videogames -κονσόλες και φορητές κονσόλες παιχνιδιών

Το περιβάλλον εργασίας του μVision 2

Το περιβάλλον εργασίας του μVision 2 παρέχει μια ενσωματωμένη πλατφόρμα ανάπτυξης λογισμικού που υποστηρίζει τα εργαλεία ανάπτυξης λογισμικού. Συνδυάζει τη διαχείριση του προγράμματος, την έκδοση κώδικα πηγής, και τη διόρθωση του προγράμματος σε ένα περιβάλλον.

Ξεκινώντας τη δημιουργία μιας εφαρμογής

Η διαδικασία είναι η ίδια σε όλα τα προγράμματα αυτής της κατηγορίας
1.Δημιουργούμε ένα project
2.Δημιουργούμε το πηγαίο αρχείο (source file) σε γλώσσα C ή assembly
3.Κτίζουμε την εφαρμογή μας με τον διαχειριστή σχεδίου (project manager) -στο menu project.
4.Διορθώνουμε τα λάθη στο πηγαίο αρχείο με τον εκσφαλματωτή (debugger).
5.Ελέγχουμε την εφαρμογή μας με τον προσομοίωτη λειτουργίας (simulator).



Εικόνα 1.Επεξηγηματικό μπλοκ διάγραμμα για τη διαδικασία ανάπτυξης εφαρμογών .

µVision IDE (Integrated Development Environment)

Το περιβάλλον του µVision συνδυάζει ένα διαχειριστή σχεδίου ,ένα πρόγραμμα διόρθωσης κειμένου με επιπλέον δυνατότητες –όπως αναγνώριση τον εντολών της C και της assembly με διαφορετικό χρωματισμό-και διαδραστική εκσφαλμάτωση λαθών, ρύθμιση παραμέτρων που έχουν να κάνουν με την αρχιτεκτονική του συστήματος που θα υλοποιηθεί η εφαρμογή, πρόσθετα εργαλεία για την προσομοίωση του συστήματος και επιγραμμική (online) βοήθεια.

```
HELLO.C

Copyright 1995-1999 Keil Software, Inc.
------------------------------------------------------------------------------*/

#include <REG1210.H>              /* special function register declarations   */
                                  /* for the intended 8051 derivative         */

#include <stdio.h>                /* prototype declarations for I/O functions */


#ifdef MONITOR51                          /* Debugging with Monitor-51 needs   */
char code reserve [3] _at_ 0x23;          /* space for serial interrupt if     */
#endif                                    /* Stop Exection with Serial Intr.   */
                                          /* is enabled                        */


/*------------------------------------------------
The main C function.  Program execution starts
here after stack initialization.
------------------------------------------------*/
void main (void) {

/*------------------------------------------------
Setup the serial port for 1200 baud at 16MHz.
------------------------------------------------*/
#ifndef MONITOR51
    SCON0 = 0x50;                 /* SCON: mode 1, 8-bit UART, enable rcvr      */
    TMOD |= 0x20;                 /* TMOD: timer 1, mode 2, 8-bit reload        */
    TH1   = 221;                  /* TH1:  reload value for 1200 baud @ 16MHz   */
    TR1   = 1;                    /* TR1:  timer 1 run                          */
    TI    = 1;                    /* TI:   set TI to send first char of UART    */
#endif
```

Μεταφραστής (C51 Compiler) και συμβολομεταφραστής (A51 Assembler).

Τα πηγαία αρχεία αρχεία που δημιουργούνται από το μVision IDE διοχετεύονται στον μεταφραστή C51 ή στο συμβολομεταφραστή A51 επεξεργάζονται και δημιουργούνται τα object files (*.obj). και τα list *.lst .

Εκσφαλματωτής (μVision2 Debugger)

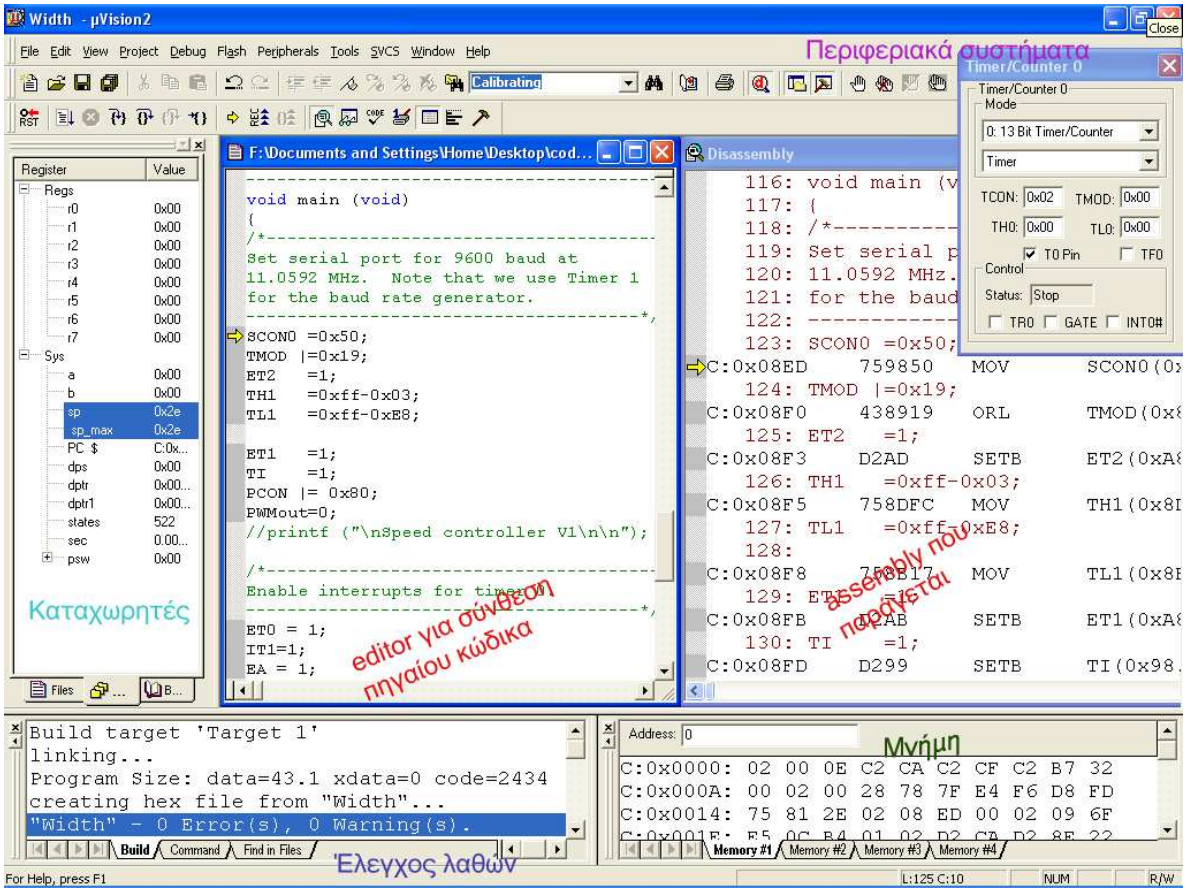
Ο εκσφαλματωτής περιέχει ένα προσομοιωτή που επιτρέπει την προσομοίωση ενός συστήματος 8051 περιλαμβάνοντας τα περιφερειακά του και εξωτερικό υλικό. Τα χαρακτηριστικά του chip που χρησιμοποιείτε καθορίζονται αυτόματα με την επιλογή του εξαρτήματος από τη βάση δεδομένων η καθορίζονται από τον σχεδιαστή (τύπο μνήμης, διεύθυνση κ.α).

Δομή φακέλλων

Το πρόγραμμα εγκαθιστά τα εργαλεία ανάπτυξης στους υποφακέλλους του βασικού φακέλου. Φάκελος βάσεων προεπιλογής είναι: C:\KEIL Ο ακόλουθος πίνακας απαριθμεί δομή μιας πλήρους εγκατάστασης που περιλαμβάνει την ολόκληρη γραμμή των 8051 εργαλεία ανάπτυξης.

Φάκελος	Περιγραφή
C:\KEIL\C51\ASM	Assembler SFR definition files and template source file.
C:\KEIL\C51\BIN	Εκτελέσιμα αρχεία της αλυσίδας εργαλείων 8051.
C:\KEIL\C51\EXAMPLES	Εφαρμογές δειγμάτων.
C:\KEIL\C51\RTX51	RTX51 Full files.
C:\KEIL\C51\RTX_TINY	RTX51 Tiny files.
C:\KEIL\C51\INC	C compiler include files.
C:\KEIL\C51\LIB	C compiler library files, startup code, and source of I/O routines.
C:\KEIL\C51\MONITOR	Target Monitor files and Monitor configuration for user hardware.
C:\KEIL\UV2	Γενικής χρήσης αρχεία του μVision2.

Το περιβάλλον του προγράμματος



Εικόνα 2 Το περιβάλλον μVision IDE

Menu Commands, Toolbars, and Shortcuts

Η μπάρα του μενού παρέχει σε σας τις επιλογές για την επεξεργασία του κώδικα , την ρύθμιση των παραμέτρων για τα αναπτυξιακά εργαλεία ,την εκσφαλμάτωση και τη διόρθωση του προγράμματος, Τα κουμπιά της εργαλειοθήκης σας επιτρέπουν τη γρήγορη εκτέλεση των εντολών του μVision2.

Μενού Debug και εντολές Debug

Debug Menu	Toolbar	Shortcut	Description
Start/Stop Debugging		Ctrl+F5	Start or stop µVision2 Debug Mode
Go		F5	Run (execute) until the next active breakpoint
Step		F11	Execute a single-step into a function
Step over		F10	Execute a single-step over a function
Step out of current function		Ctrl+F11	Execute a step out of the current function
Stop Running		ESC	Stop program execution
Breakpoints...			Open Breakpoint dialog
Insert/Remove Breakpoint			Toggle breakpoint on current line
Enable/Disable Breakpoint			Enable/disable breakpoint on the current line
Disable All Breakpoints			Disable all breakpoints in the program
Kill All Breakpoints			Kill all breakpoints in the program
Show Next Statement			Show next executable statement/instruction
Enable/Disable Trace Recording			Enable trace recording for instruction review
View Trace Records			Review previous executed instructions
Memory Map...			Open memory map dialog
Performance Analyzer...			Open setup dialog for the Performance Analyzer
Inline Assembly...			Stop current build process
Function Editor...			Edit debug functions and debug INI file

Μενού Περιφερειακών

Peripherals Menu	Toolbar	Shortcut	Description
Reset CPU			Set CPU to reset state
Interrupt, I/O-Ports, Serial, Timer, A/D Converter, D/A Converter, I^2C Controller, CAN Controller, Watchdog			Open dialogs for on-chip peripherals, these dialogs depend on the CPU selected from the device database. This list will vary by microcontroller.

Προσθήκη εργαλείων

Με την εγκατάσταση των απαραιτήτων *.dll στο μενού peripherals προστίθενται τα παρακάτω εργαλεία:

- Scope
- LCD simulation(HD44780)

- I2C simulation (7bit address)
- Signal generator
- LED simulation
- 8051 statemachine
- Time Measurement
- TCP/IP

Η διαδικασία είναι η εξής: Εντοπίζουμε και ανοίγουμε το αρχείο tools.ini -που βρίσκεται στο βασικό φάκελο στην περιοχή εγκατάστασης του μVision π.χ C:\KEIL - και ψάχνουμε την παράγραφο (μέσα στο αρχείο) με τίτλο [C51] όπου και γράφουμε τις παρακάτω προτάσεις αυτούσιες :

```
AGSI1=SCOPE.DLL           ("Scope simulation")
AGSI2=LCD.DLL             ("LCD simulation")
AGSI3=I2C.DLL             ("I2C simulation")
AGSI4=LED_CONTROL.DLL     ("LED simulation")
AGSI4=Statemachine.dll    ("8051 statemachine")
AGSI5=Signalgenerator.dll ("Signal generator")
AGSI6=TimeMeasure.dll     ("Time Measurement")
AGSI7=TCPIP.dll           ("TCPIP Anbindung")
```

Στην περιοχή εγκατάστασης \keil\c51\bin μεταφέρουμε τα αρχεία με επέκταση *.dll τα οποία βρίσκονται στο cd της πτυχιακής στο φάκελο updates\mvision dll
Την επόμενη φορά που θα ανοίξουμε το περιβάλλον του μVision θα έχουν προστεθεί και τα παραπάνω περιφερειακά / εργαλεία .

Mode λειτουργίας του μVision

Build Mode:Επιτρέπει τη μετάφραση όλων των αρχείων της εφαρμογής και την παραγωγή εκτελέσιμων προγραμμάτων.

Debug Mode: Παρέχει ένα πολυδύναμο εκσφαλματωτή (debugger) για να δοκιμή των εφαρμογών

Και στα δύο mode λειτουργίας μπορεί να τροποποιηθεί ο πηγαίος κώδικας μέσω του editor.

Η γλώσσα προγραμματισμού C

1. Τύποι Δεδομένων

2. Δομές Ελέγχου

3. Πίνακες (Arrays)

4. Δείκτες (Pointers)

5. Συναρτήσεις

1. Τύποι δεδομένων

Η C έχει πέντε βασικούς τύπους δεδομένων:

char (character),

int (integer),

float (floating point),

double (double floating point),

void (no value).

Όλοι οι άλλοι τύποι της C βασίζονται σ' αυτούς. Όλοι οι βασικοί τύποι εκτός από τον τύπο void μπορεί ν' αλλάξουν γράφοντας πριν από τον τύπο τον κατάλληλο μετασχηματισμό. Οι μετασχηματισμοί αυτοί είναι οι**: signed, unsigned, long, και short.** Το μέγεθος και τα διαστήματα τιμών των τύπων της C εξαρτάται από τον επεξεργαστή. Στον πίνακα δίνουμε τους τύπους δεδομένων όπως ορίζονται από το πρότυπο ANSI.

Τύπος	Μέγεθος σε bits	Διάστημα τιμών
Bit	1	0 έως 1
signed char	8	-128 to +127
unsigned char	8	0 to 255
enum	16	-32768 to +32767
signed short	16	-32768 to +32767
unsigned	16	0 to 65535
signed int	16	-32768 to +32767
unsigned int	16	0 to 65535
signed long	32	-2147483648 to 2147483647
unsigned long	32	0 to 4294967295
float	32	±1.175494E-38 to ±3.402823E+38
sbit *	1	0 to 1
sfr *	8	0 to 255
sfr16 *	16	0 to 65535

* **bi**t, **sbi**t, **sf**r, και **sfr16** specific to είναι συγκεκριμένα για την οικογένεια 8051 και τους compilers C51 και C251.Δεν είναι μέρος του πρότυπου ANSI C και δεν μπορούν να προσπελασθούν μέσω pointers.

Δηλώσεις μεταβλητών

Τα αναγνωριστικά στη C μπορούν να έχουν όσους χαρακτήρες θέλουμε. Αν το αναγνωριστικό είναι εξωτερικό όνομα (όνομα συνάρτησης η καθολική μεταβλητή) τότε μόνο οι έξι πρώτοι χαρακτήρες είναι σημαντικοί διαφορετικά για εσωτερικά ονόματα οι πρώτοι 31 χαρακτήρες είναι σημαντικοί. **Τα κεφαλαία γράμματα στην C είναι διαφορετικά από τα μικρά.**

Η δήλωση μιας μεταβλητής έχει την γενική μορφή:

<τύπος> <λίστα μεταβλητών>

```
int i=0,j;

char q='?';

short int si;

float f,g;
```

Δηλώσεις μεταβλητών κάνουμε μέσα σε συναρτήσεις (local variables, automatic), στις παραμέτρους μιας συνάρτησης (formal parameters) και έξω απ' όλες τις συναρτήσεις (global variables). Μπορούμε επίσης να δηλώσουμε μία τοπική μεταβλητή μέσα σε μία ενότητα π.χ.

```
 if (συνθήκη)

{

char x[30];

...

...

}
```

Στη περίπτωση αυτή η εμβέλεια της μεταβλητής είναι η ενότητα στην οποία είναι δηλωμένη. Έτσι αποφεύγουμε πλάγια αποτελέσματα και έχουμε οικονομία χώρου.

Καθολικές μεταβλητές έχουν εμβέλεια σε ολόκληρο το πρόγραμμα και δηλώνονται στην αρχή του κώδικα έξω απ' όλες τις συναρτήσεις. Όταν μια καθολική και μία τοπική μεταβλητή έχουν το ίδιο όνομα τότε μέσα στην εμβέλεια της τοπικής μεταβλητής αναφερόμαστε πάντα στην τοπική. Μεταβλητές που είναι παράμετροι συναρτήσεων συμπεριφέρονται ως τοπικές μεταβλητές της συνάρτησης.

Σταθερές εισάγονται με την προκαθορισμένη λέξη **const** π.χ.

```
const int i=1;

const char q='?';
```

Πτητικές (volatile) μεταβλητές πληροφορούν τον μεταγλωττιστή ότι η τιμή τους μπορεί ν' αλλάξει χωρίς αυτό να δηλώνεται σαφώς στο πρόγραμμα.

Τρόποι αποθήκευσης μεταβλητών

Υπάρχουν τέσσερις τρόποι να πληροφορήσουμε τον μεταγλωττιστή πως ν' αποθηκευτεί μια μεταβλητή.

Extern

Επειδή η C υποστηρίζει ξεχωριστή μεταγλώττιση των διαφόρων ενοτήτων ενός μεγάλου προγράμματος θα πρέπει να υπάρχει κάποιος τρόπος που να πληροφορεί τον μεταγλωττιστή ότι ορισμένες μεταβλητές είναι ορισμένες κάπου αλλού. Υπενθυμίζουμε ότι μια καθολική μεταβλητή μπορεί να δηλωθεί μία μόνο φορά.

Αρχείο 1 Αρχείο 2

int x,y; extern int x,y;

... ...

static

Η δήλωση static έχει διαφορετικό αποτέλεσμα πάνω σε τοπικές μεταβλητές και διαφορετικό σε καθολικές μεταβλητές.

Μία static τοπική μεταβλητή έχει εμβέλεια μέσα στη συνάρτηση που είναι δηλωμένη και κρατά την τιμή της μεταξύ διαδοχικών καλεσμάτων της συνάρτησης. Έτσι μπορεί να χρησιμοποιηθεί σε μία συνάρτηση παραγωγής μιας σειράς αριθμών π.χ.

```
increment(void)

{

static int count=0;

count=count+5;

return (count);

}
```

Η αρχικοποίηση της count γίνεται μία μόνο φορά, στο πρώτο κάλεσμα της increment.

Μία καθολική μεταβλητή static έχει εμβέλεια μόνο στο αρχείο στο οποίο είναι δηλωμένη. Αυτό σημαίνει ότι αν και είναι καθολική δεν μπορούν να την δουν ρουτίνες από άλλα αρχεία και ν' αλλάξουν το περιεχόμενό της.

Οι μεταβλητές static μας δίνουν την δυνατότητα να αποκρύψουμε ένα μέρος από ένα πρόγραμμα. Αυτό μας βοηθάει πολύ στο γράψιμο μεγάλων προγραμμάτων καθώς επίσης στο γράψιμο συναρτήσεων που θα μπουν σε βιβλιοθήκες.

register

Μία μεταβλητή register αποθηκεύεται σ' ένα καταχωρητή της CPU αντί για την μνήμη όπως με τις απλές μεταβλητές (συνεπώς οι μεταβλητές αυτές δεν έχουν διεύθυνση). Αυτό σημαίνει ότι πράξεις με μεταβλητές τύπου register είναι πολύ πιο γρήγορες εφόσον δεν απαιτείται προσπέλαση στη μνήμη για να δούμε ή ν' αλλάξουμε την τιμή τους. π.χ.

register int temp;

Δηλώσεις σταθερών

Σταθερές χαρακτήρες γράφονται σε απλά εισαγωγικά.

char question_mark = '?';

Σταθερές τύπου string γράφονται μέσα σε διπλά εισαγωγικά, "---". Έτσι συνήθως τυπώνουμε διάφορα μηνύματα. Παραδείγματα σταθερών για τους άλλους τύπους δίνονται στον παρακάτω πίνακα.

int 1234

long int 38754L

short int 123

unsigned int 62222

float 12.345F

float 1.1e-3F

double -0.9876544

char '?'

Υπάρχουν όμως ορισμένοι χαρακτήρες που δεν τυπώνονται με μια σταθερά τύπου string για παράδειγμα τα διπλά εισαγωγικά. Για τους χαρακτήρες αυτούς έχουμε τις λεγόμενες σταθερές backslash που δίνονται στον επόμενο πίνακα.

Κώδικας	**Σημασία**
\b	backspace
\f	form feed
\n	νέα γραμμή
\r	carriage return
\t	οριζόντιο tab
\"	διπλά εισαγωγικά
\'	απλά εισαγωγικά
\0	Null
\\	backslash
\v	κάθετο tab
\a	alert
\N	οκταδική σταθερά
\xN	δεκαεξαδική σταθερά

Τελεστές

Η C έχει τέσσερις τύπους τελεστών: **αριθμητικοί, σύγκρισης (συσχεσιακοί), λογικοί και τελεστές χειρισμού bits (bitwise operators).**Παρακάτω δίνουμε ένα πίνακα με όλους τους τελεστές διατεταγμένους σύμφωνα με την προτεραιότητά τους.

Προτεραιότητα	**Τελεστές**
Υψηλότερη	( ) [ ] ->
	! ~ ++ -- -(type) * & sizeof
	* / %
	- +
	<< >>
	< <= > >=
	== !=
	&
	^
	|
	&&
	||
	?
	= += -= *= /=
Χαμηλότερη	

Συμβατότητα με εκχωρήσεις

Ο τελεστής εκχώρησης έχει την γενική μορφή

<αναγνωριστικό> = <παράσταση>

Κατά την εκχώρηση η τιμή της παράστασης μετατρέπεται στον τύπο της μεταβλητής στο αριστερό μέρος της παράστασης. Για παράδειγμα:

int i;

char ch;

float f;

```
void function(void)

{

ch=i; /* ch= ο χαρακτήρας που αντιστοιχεί στο δεύτερο

byte του i */

i=f; /* i= το ακέραιο μέρος του f */

f=ch; /* f= ο αριθμός που αντιστοιχεί στο ένα byte του ch */

f=i; /* f= ο αριθμός που αντιστοιχεί στα δύο bytes του int */

}
```

Οταν σε μία παράσταση υπάρχουν τελεσταίοι διαφορετικών τύπων τότε μετασχηματίζονται στον τύπο του "ισχυρότερου" τελεσταίου. Στο παρακάτω σχήμα φαίνονται όλοι οι μετασχηματισμοί τύπων που γίνονται κατά τον υπολογισμό της παράστασης

```
r=(ch / i) + (f * d) - (f + i)

------- ------- ------

i d f

-----------

d

----------------

d
```

Μετασχηματισμός του τύπου μιας παράστασης μπορεί να γίνει προσδιορίζοντας τον νέο τύπο μέσα σε παρένθεση πριν από την παράσταση. π.χ.

(τύπος)<παράσταση>

Για παράδειγμα

```
int i;

(float) i/2;
```

Τέλος στη C μπορούμε να έχουμε πολλαπλή εκχώρηση x=y=z=0;

Αριθμητικοί τελεστές

-	αφαίρεση, πρόσημο
+	πρόσθεση
*	πολλαπλασιασμός
/	διαίρεση
%	(mod)
--	ελάττωση μεταβλητής κατά 1
++	αύξηση μεταβλητής κατά 1

Οι τελεστές -- και ++ μπορεί να τοποθετηθούν μπροστά ή μετά ένα τελεσταίο. Η εντολή --x; ισοδυναμεί με την x:=x-1 αλλά η αφαίρεση εκτελείται πριν χρησιμοποιήσουμε την τιμή της x. Όμοια και η εντολή ++x;

Η εντολή x--; ισοδυναμεί με την x:=x-1 αλλά η αφαίρεση εκτελείται αφού χρησιμοποιήσουμε την τιμή της x.

x=3; x=3;

y=++x; y=x++;

αποτέλεσμα: y=4 (x=4) αποτέλεσμα: y=3 (x=4)

Συσχεσιακοί και λογικοί τελεστές

>	Μεγαλύτερο
>=	Μεγαλύτερο ή ίσο
<	Μικρότερο
<=	Μικρότερο ή ίσο
==	Ίσον
!=	διάφορο του ίσον
&&	AND
||	OR
!	NOT

Στη C true είναι μια τιμή διαφορετική του μηδενός και false είναι το μηδέν.

Τελεστές χειρισμού bits (bitwise operators)

Χειρισμός των bits σημαίνει την δυνατότητα επέμβασης στα bits ενός byte ή μιας λέξης που αντιστοιχούν στους τύπους char και int.

&	AND
|	OR
^	XOR
~	συμπλήρωμα ως προς 1
>>	shift right
<<	shift left

Για παράδειγμα η παράσταση (ch & 127) όπου ch είναι τύπου char εκχωρεί την τιμή 0 στο parity bit.

ch: 1 0 0 1 1 1 0 1

127: 0 1 1 1 1 1 1 1

& 0 0 0 1 1 1 0 1

Ο τελεστής << έχει τον τύπο: variable << number και μετακινεί τα όλα τα bits της μεταβλητής προς τ' αριστερά number θέσεις. Οι κενές θέσεις που δημιουργούνται από τα δεξιά αντικαθίστανται με 0. π.χ.

x=5; x: 0 0 0 0 0 1 0 1

x=x << 2; 0 0 0 1 0 1 0 0

(αποτέλεσμα x=20)

Συντμήσεις (Shorthands)

Η C σε ορισμένες περιπτώσεις εκχώρησης μας επιτρέπει να συντομεύσουμε τον κώδικα. Έτσι μία παράσταση της μορφής:

<μεταβλητή> =<μεταβλητή> <τελεστής><παράσταση>

μπορεί να γραφτεί ως:

<μεταβλητή> <τελεστής> = <παράσταση>

Για παράδειγμα οι εντολές:

year+=11; year=year+11;

i-=3; ισοδυναμούν με i=i-3;

x*=10; x=x*10;

y/=5; y=y/5;

2. Δομές Ελέγχου

1. Εντολή if - then - else

if (<expression>) <statement>;

if (<expression>) <block_1>;

if (<expression>) <statement_1>; else <statement_2>;

if (<expression>) <block_1>; else <block_2>;

block: {

<statement 1>

<statement 2>

....

}

Παράδειγμα: if (x>0) absx=x; else absx= -x;

Και στη C μπορούμε να έχουμε εμφωλιασμένα if. Τότε το else αντιστοιχεί στο πλησιέστερο if που είναι στο ίδιο block με το else.

2. Ο τριαδικός τελεστής ?

Ο τελεστής ? μπορεί να χρησιμοποιηθεί αντί για το if-then-else. Ο τύπος του ? είναι:

<expression> ? <expression_1> : <expression_2>;

όπου <expression_1> και <expression_2> είναι απλές παραστάσεις και όχι

άλλες εντολές της C, για παράδειγμα

y = x>0 ? x : -x;

y ? f1()+f2() : printf("a message");

3. Εντολή while

while (<expression>)

<block>

while (i<10)

{

printf("Iteration no= %d\n",i);

++i;

}

Ενα while μπορεί να μην έχει εντολές στο σώμα του:

while ((ch=getchar()) != 'A');

4. Εντολή do - while

do <block>

while (<expression>);

5. Εντολή for

for (<initial expression>;<test expression>; <Increment expression>)

<Block>

Παραδείγματα: for (i=1; i<20; ++i)

printf("Iteration no= %d\n",i);

for (i=1; i<=100; ++i)

sum+=number;

Μέσα στο for μπορούμε να χρησιμοποιήσουμε τον τελεστή , (κόμμα) για να "ενώσουμε" περισσότερες από μία παραστάσεις π.χ. στο παρακάτω for και το x και το y ελέγχουν το for.

a) for (x=0,y=0; x+y<10; ++x)

{

scanf(%c",y);

y=y - '0'

....

}

b) for (i=1; j=strlen(message); i<j; i++, j--)

Μία άλλη παραλλαγή του for είναι ότι ορισμένα μέρη του μπορεί να μην υπάρχουν (οι παραστάσεις στο for είναι προαιρετικές).

for (x=0; x!=120;) scanf("%d", &x);

Επίσης πολλές φορές η συνθήκη αρχικοποίησης της μεταβλητής ελέγχου του for είναι πολύπλοκη και γι' αυτό υπολογίζεται έξω από το for π.χ.

x= ...;

for (; x<100;){

prinf("%d", x);

```
++x;

}
```

Οταν και οι τρείς παραστάσεις του for δεν υπάρχουν τότε έχουμε μιά άπειρη ανακύκλωση.

```
for (;;) printf("this loop runs forever\n");
```

Τέλος το σώμα του for μπορεί να είναι κενό. Συνήθως χρησιμοποιείται για να δημιουργήσουμε μια καθυστέρηση (time delay loops)

```
for (t=0; t<max; t++);
```

Τί κάνει το παρακάτω loop?

```
for (; *str=' '; str++);
```

6. Εντολή switch

```
switch (<expression>)

{

case <value1>:<statement1>;

case <value2>:<statement2>;

case <value3>:<statement3>;

default: <statement4>

}
```

(Εκτελούνται τα case από την τιμή της παράστασης και κάτω) Στή παρακάτω εντολή switch η εντολή break έχει ως αποτέλεσμα να εκτελεστεί μόνο το case που έχει την τιμή τηςexpression. Περισσότερα για την εντολή break θα δούμε στα επόμενα.

```
switch (<expression>)

{

case <value1>:<statement1>;

break;

case <value2>:<statement2>;

break;

case <value3>:<statement3>;
```

```
break;

default: <statement4>

}


fflush(stdin);

scanf("%g %c %g", &num1,&oper,&num2);

switch (oper)

{

case '+': result= num1 + num2;

break;

case '-': result= num1 - num2;

break;

case '*': result= num1 * num2;

break;

case '/': result= num1 / num2;

default: result= 0.0;

}
```

7. Εντολές μεταφοράς ελέγχου

α. Εντολή return

Χρησιμοποιείται για επιστροφή από συνάρτηση. Η εντολή έχει ως αποτέλεσμα να επιστρέψει η εκτέλεση στο σημείο από το οποίο καλέστηκε η συνάρτηση. Αν το return περιέχει και μία τιμή {return(0)} τότε η τιμή αυτή επιστρέφει από την συνάρτηση (διαφορετικά μπορεί να επιστρέψει οτιδήποτε, εξαρτάται από τον μεταγλωττιστή). Μια συνάρτηση void δεν έχει return που να επιστρέφει τιμή.

β. Εντολή goto

```
/* όπως και στη Pascal*/

loop:
```

```
x++;

if (x<10) goto loop;
```

γ. Εντολή break

Η εντολή break έχει δύο χρήσεις. Πρώτον για να τερματίσει ένα case της εντολής switch και δεύτερον για να τερματίσει ένα loop παρακάμπτοντας την συνθήκη του loop. (θα τερματίσει το πιό εσωτερικό loop) π.χ.

```
for (t=0; t<100; t++) do

{ {/*εξετάζει να βρεί ένα όνομα*/

....

if (kbhit()) break;

if (t==10) break; } while (!found);

....

}
```

δ. Εντολή continue

Η εντολή continue έχει ως αποτέλεσμα να εκτελεστεί η επόμενη επανάληψη ενός loop αγνοώντας τις εντολές μετά το continue.

3.Πίνακες (arrays)

Ορισμός:

<Type> <name of the variable> [<Nmax>], ...;

```
 int vector[10];

double b[10];
```

Σε όλα τα arrays στη C ο δείκτης του πρώτου στοιχείου είναι το 0.

vector[0], vector[1], vector[2], ... , vector[9].

Το μέγεθος ενός array στη μνήμη εξαρτάται από τον τύπο και το πλήθος των στοιχείων του.

bytes = sizeof(type) * number of elements

ΠΡΟΣΟΧΗ, Ο μεταγλωττιστής της C δεν ελέγχει αν οι δείκτες ενός array είναι στα επιτρεπτά όρια. Οπως και στη Pascal έτσι και στη C έχουμε arrays δύο διαστάσεων

float pinakas[10][10];

Όμοια μπορούμε να έχουμε arrays με περισσότερες από δύο διαστάσεις:

<Type> <name of the variable> [<Nmax1>] [<Nmax2>] [<Nmax3>]...;

int matrix[10] [10];

matrix[0][0], matrix[0][1], ... ,matrix[9][9]

Στη C μπορούμε να καθορίσουμε τις τιμές των στοιχείων ενός array κατά την δήλωσή του.

type <array name>[n1]...[nk]={list of values};

π.χ. int x[5]={1, 2, 3, 4, 5} ισοδυναμεί με

(x[0]=1, x[1]=2, x[2]=3, x[3]=4, x[4]=5).

Τα array χαρακτήρων επιτρέπουν να δώσουμε τιμές στα στοιχεία τους υπό μορφή συμβολοσειρών: char s[9]= "I like C" αυτό ισοδυναμεί με

char s[9]={"I ", " ", "l", "i", "k", "e", " ", "C" "\0"}

Με τον ίδιο τρόπο δίνουμε τιμές στα στοιχεία των arrays με 2 ή περισσότερες διαστάσεις. Επειδή ο τρόπος αυτός δεν είναι τόσο καλός αφού πρέπει ο προγραμματιστής να μετράει πρώτα το πλήθος των χαρακτήρων και μετά να καθορίσει το μέγεθος του array η C επιτρέπει να μην δηλώσουμε καθόλου το μέγεθος του array (αυτό γίνεται αυτόματα από τον μεταγλωττιστή) (Unsized array initialization). π.χ. s[] = "I like C". Η εντολή

printf("%s has length %d \n,s,sizeof(s));

θα τυπώσει

I like C has length 9

Για arrays περισσοτέρων διαστάσεων πρέπει να καθορίσουμε το μέγεθος όλων των διαστάσεων εκτός από την πρώτη π.χ. int pinakas[][10];

Συμβολοσειρές (**Strings)**

Στη C η συμβολοσειρά ορίζεται ως ένα array χαρακτήρων που τελειώνει με τον χαρακτήρα null (\0). Ετσι όταν θέλουμε ν' αποθηκεύσουμε μία συμβολοσειρά μεγέθους 10 θα χρησιμοποιήσουμε ένα array μεγέθους 11 π.χ. char s[11];

Μια σταθερά τύπου συμβολοσειράς είναι μία λίστα από χαρακτήρες μέσα στα εισαγωγικά " ".

Προσοχή: η χρησιμοποίηση των συμβολοσειρών γίνεται πάντα με ειδικές συναρτήσεις. Οι πιό συνιθισμένες είναι:

strcpy (s1, s2) αντιγράφει το s2 στο s1.

strcat (s1, s2) συνενώνει το s2 στο τέλος του s1.

strlen (s1) επιστρέφει το μήκος (μέγεθος) του s1.

strcmp (s1, s2) επιστρέφει 0 αν s1= s2, <0 αν s1<s2 και >0 αν s1>s2.

strchr (s1, ch) επιστρέφει ένα δείκτη στη θέση που εμφανιζεται για

πρώτη φορά ο χαρακτήρας ch.

strstr (s1, s2) επιστρέφει ένα δείκτη στη θέση που εμφανίζεται για

πρώτη φορά το s2 μέσα στο s1.

Οι συναρτήσεις αυτές (καθώς και άλλες) είναι στο αρχείο <STRING.H>.

(Arrays συμβολοσειρών: χρησιμοποιούμε ένα array δύο διαστάσεων. Η πρώτη διάσταση δηλώνει το πλήθος των συμβολοσειρών και η δεύτερη το μέγιστο μέγεθός τους.

4.Δείκτες (Pointers)

Οι δείκτες της C είναι ένα πάρα πολύ δυναμικό εργαλείο αλλά και επικίνδυνο διότι μπορεί να δημιουργήσει λάθη πολύ δύσκολα να βρούμε έως να κάνουν το σύστημα crash. Pointer είναι μία μεταβλητή που αποθηκεύει μια διεύθυνση μνήμης.

Ένας δείκτης δηλώνεται ως εξής:

type *<identifier>;

Δείκτης σε μεταβλητή τύπου float	Μεταβλητή τύπου float
Address	Αριθμός
1000	3.14159

1000

Arrays και δείκτες

Τ' όνομα ενός array χωρίς τις παρενθέσεις θεωρείται ως δείκτης στην αρχή του array, π.χ. έστω int s[10]; τότε μπορούμε να δημιουργήσουμε ένα δείκτη που να δείχνει στην αρχή ενός array χρησιμοποιώντας τ' όνομά του (s). Για παράδειγμα

```
int *p;

int s[10];

p= s;
```

Μπορούμε επίσης να καθορίσουμε την διεύθυνση του πρώτου στοιχείου ενός array χρησιμοποιώντας τον τελεστή &. Ο τελεστής & είναι ένας μοναδιαίος τελεστής που επιστρέφει την διεύθυνση στη μνήμη του τελεσταίου του. Ετσι το s είναι ισοδύναμο με &s[0] ή η παράσταση s==&s[0] είναι true. Όπως βλέπουμε στη C οι δείκτες και τα arrays σχετίζονται πολύ. Μια μεταβλητή τύπου δείκτη μπορεί να έχει δείκτες όπως ακριβώς ένα array. Για παράδειγμα

```
int *ptr, x[10];

ptr= x;

ptr[5]= 3;

ptr[p+5]=3; /*χρησιμοποιεί αριθμητική με pointers*/
```

Ομοια d και &d[0][0] είναι ισοδύναμα όπου d είναι ένα δυσδιάστατο array. Γενικά a[j][k] είναι ισοδύναμο με:

```
*(a + (j * rowlength) + k)
```

Η αριθμητική με pointers είναι ταχύτερη από αυτή των arrays γι' αυτό και χρησιμοποιούνται οι pointers πολλές φορές για προσπέλαση στα στοιχεία ενός array.

Τελεστές για δείκτες

1. Ο τελεστής &

Όπως είπαμε ο μοναδιαίος τελεστής & επιστρέφει την διεύθυνση της μνήμης του τελεσταίου του. Δεν έχει καμιά σχέση με την τιμή του.

2. Ο τελεστής * (το αντίθετο του &)

Είναι ένας μοναδιαίος τελεστής που επιστρέφει την τιμή μιάς μεταβλητής που είναι αποθηκευμένη στην διεύθυνση που ακολουθεί π.χ.

```
m= &s;

q= *m; /* εκχωρεί την τιμή του s στο q */
```

Αριθμητική με δείκτες

α. Πρόσθεση Έστω:

```
int *ptr;

/* έστω ptr δείχνει στη θέση 1000 */

ptr++;

/* ptr δείχνει στη θέση 1002 */
```

β. Αφαίρεση Αν χρησιμοποιηθεί στον παραπάνω κώδικα η εντολή ptr--; τότε ο ptr θα δείχνει στην θέση 998. Κάθε φορά που αυξάνεται ένας δείκτης δείχνει στη θέση μνήμης του επόμενου στοιχείου ανάλογα με τον τύπο του δείκτη. Όμοια μπορούμε να προσθέσουμε ή ν' αφαιρέσουμε ένα ακέραιο από ένα δείκτη π.χ. ptr= ptr + 6; /* ptr δείχνει στη θέση 1012 */ Δεν επιτρέπεται να πολλαπλασιάσουμε ή να διαιρέσουμε με δείκτες.

Σύγκριση δεικτών (==, !=)

Συνήθως κάνουμε σύγκριση δεικτών για να ελέγξουμε αν δείχνουν ή όχι στο ίδιο αντικείμενο.

Arrays από δείκτες

int *p[10]; /* δηλώνει ένα array από 10 δείκτες */

p[2]= &x; /* τότε *p[2] έχει την τιμή του χ */

Ακριβέστερα η δήλωση int *p[10]; σημαίνει ότι έχουμε ένα δείκτη σ' ένα array από 10 δείκτες τύπου int. Στη C μπορούμε να έχουμε ένα δείκτη που δείχνει σ' ένα άλλο δείκτη που δείχνει σε κάποια τιμή.

Pointer	Pointer	variable
address	address	value

Μια μεταβλητή που είναι δείκτης ενός άλλου δείκτη δηλώνεται με δύο αστερίσκους. π.χ.

int x, *p, **q;

x= 10;

p= &x;

q= &p;

p	q	**q
&x	&p	10

Αρχικοποίηση δεικτών

Αν προσπαθήσουμε να χρησιμοποιήσουμε ένα δείκτη πριν του δώσουμε μια τιμή τότε όχι μόνο το πρόγραμμα γίνεται crash αλλά και το λειτουργικό σύστημα. Συνήθως δίνουμε την τιμή null σ' ένα δείκτη. Αν όμως χρησιμοποιήσουμε ένα null δείκτη αριστερά μιας εκχώρησης τότε το σύστημα γίνεται crash. π.χ. Οι παρακάτω εντολές ψάχνουν ένα array από δείκτες σε συμβολοσειρές για τον εντοπισμό ενός ονόματος. Η αναζήτηση σταματά όταν βρεθεί τ' όνομα ή όταν φτάσουμε σ' ένα null δείκτη.

```
for (t=0; p[t]; t++)

if (!strcmp(p[t],name)) return t;
```

Πολλές φορές στη C χρησιμοποιούμε την εξής αρχικοποίηση μιας συμβολοσειράς.

```
char *ptr= "hello there\n";
```

Ο μεταγλωττιστής της C χρησιμοποιεί ένα χώρο της μνήμης (string table) για την αποθήκευση των σταθερών συμβολοσειρών ενός προγράμματος. Με την παραπάνω δήλωση η διεύθυνση του "hello there" που αποθηκεύεται στο string table περνάει στον δείκτη ptr.

Δυναμική καταχώρηση μνήμης

Η malloc() επιστρέφει ένα δείκτη στο πρώτο byte της μνήμης που καταχωρείται στο σωρό (heap) της μνήμης. Όταν δεν υπάρχει χώρος η malloc επιστρέφει null. π.χ.

```
ptr= malloc(50*sizeof(int)); /*καταχωρείται χώρος για 50 ακεραίους*/
```

Όπως είπαμε ένας null δείκτης μπορεί να δημιουργήσει πρόβλημα. Έτσι μετά την κλήση της malloc πρέπει να εξετάσουμε κατά πόσο η καταχώρηση της μνήμης ήταν επιτυχής. Η συνάρτηση free() κάνει ακριβώς το αντίθετο από την malloc.

Δυναμική καταχώρηση πινάκων

Πολλές φορές θέλουμε να καταχωρήσουμε κάποια μνήμη με την malloc και να χειριστούμε τον χώρο αυτόν όπως τα arrays. Οι παρακάτω εντολές δείχνουν πως μπορούμε να χρησιμοποιήσουμε ένα δυναμικά καταχωρημένο array.

```
char *s;
```

```
register int t;

s= (int *)malloc(80);

if (!s) {

printf("memory failed\n");

exit(1);

}

gets(s);

for (t=strlen(s)-1; t>=0; t--) putchar(s[t]);

free(s);
```

5.Συναρτήσεις

Ο γενικός τύπος μιας συνάρτησης είναι:

<τύπος> <όνομα_συνάρτησης>(<λίστα παραμέτρων>)

{

/* σώμα της συνάρτησης */

}

<λίστα παραμέτρων>::= <τύπος> var1, <τύπος> var2,... <τύπος> varN

Αν δεν έχει καθοριστεί ο τύπος της συνάρτησης τότε ο μεταγλωττιστής υποθέτει ότι η συνάρτηση επιστρέφει μία integer τιμή.

Οταν μια τοπική μεταβλητή έχει δηλωθεί static τότε κρατά την τιμή της μεταξύ διαδοχικών καλεσμάτων της συνάρτησης. Μια τέτοια μεταβλητή ο μεταγλωττιστής την χειρίζεται ως καθολική μεταβλητή αλλά η εμβέλειά της είναι μόνο μέσα στη συνάρτηση. Στη C δεν μπορούμε να έχουμε μια συνάρτηση μέσα σε μια άλλη συνάρτηση (δεν είναι δομημένη κατά ενότητες γλώσσα). Οταν δεν υπάρχει συμβατότητα μεταξύ των πραγματικών και τυπικών παραμέτρων τότε ο μεταγλωττιστής δεν δίνει κάποιο μήνυμα αλλά πέρνουμε άσχετα αποτελέσματα.

Γενικά στα υποπρογράμματα οι τιμές στις τυπικές παραμέτρους περνούν με δύο τρόπους: με τιμή (call by value) και με αναφορά (call by reference). Στη C έχουμε μόνο call by value. Παρακάτω θα δούμε πως μπορούμε να κάνουμε call by reference στη C.

Call by reference

Το κάλεσμα μιας παραμέτρου με αναφορά γίνεται με την βοήθεια ενός δείκτη στη παράμετρο. Αυτό έχει ως αποτέλεσμα να περάσει στη συνάρτηση η διεύθυνση της παραμέτρου και μπορούμε ν' αλλάξουμε την τιμή της παραμέτρου έξω από την συνάρτηση. π.χ.

```
void swap(int *x. int *y)

{

int temp;

temp= *x;

*x= *y;

*y= temp;

}
```

Τότε το κάλεσμα της συνάρτησης swap γίνεται ως εξής:

```
x= 10;

y= 20;

swap(&x, &y);
```

Τα arrays αποτελούν μια εξαίρεση στο κανόνα "call by value" της C. Οταν ένα array είναι παράμετρος μιας συνάρτησης τότε μόνο η διεύθυνση του array περνά στη συνάρτηση και όχι ολοκληρο το array, δηλαδή ένας δείκτης στο πρώτο στοιχείο του πίνακα περνά στη συνάρτηση. (τα' όνομα ενός array όπως είπαμε αποτελεί ένα δείκτη στο πρώτο στοιχείο του array).

```
I. #include <stdio.h>

void display(int num[10]);

void main(void)

{

int t[10], i;

for (i=0; i<10; ++i) t[i]= i;

display(t);

}

void display(int num[10])
```

```
{

int i;

for (i=0; i<10; i++) printf("%d", num[i]);

}
```

Ο μεταγλωττιστής αυτόματα μετατρέπει το array num σε ακέραιο δείκτη.

```
II. /*δήλωσε το array χωρίς μέγεθος */

void display(int num[])

{

int i;

for (i=0; i<10; i++) printf("%d", num[i]);

}
```

Αφού η C δεν κάνει έλεγχο των ορίων ενός array το μέγεθός του δεν έχει καμιά σχέση με την παράμετρο. Και στη περίπτωση αυτή το num θεωρείται ως ακέραιος δείκτης.

```
III. /* δήλωσε το num ως δείκτη */

void display(int *num)

{

int i;

for (i=0; i<10; i++) printf("%d", num[i]);

}
```

Σ' όλες τις περιπτώσεις όταν έχουμε μια παράμετρο πίνακα τότε η διεύθυνσή του περνά στη συνάρτηση.

Arrays και συναρτήσεις

Στη C δεν μπορεί ένα ολόκληρο array να περάσει ως πραγματική παράμετρος μιας συνάρτησης. Μπορούμε να περάσουμε όμως ένα δείκτη (με τ' όνομα του array). π.χ.

```
int s[10];

.....

function (s);
```

Όταν ένα array είναι τυπική παράμετρος τότε μπορεί να δηλωθεί είτε ως δείκτης, είτε ως array κάποιου μεγέθους είτε ως array χωρίς μέγεθος. π.χ.

function (int *s);

ή function (int s[10]);

ή function (int s[]);

Όταν έχουμε ένα array δύο διαστάσεων ως πραγματική παράμετρο σε μιά συνάρτηση τότε χρησιμοποιούμε δείκτη. Ως τυπική παράμετρος το array πρέπει να δηλώσουμε το μέγεθος όλων των διαστάσεών του εκτός από την πρώτη.

Παράμετροι στο main

Πολλές φορές θέλουμε να περάσουμε πληροφορίες στο κύριο υποπρόγραμμα. Αυτό γίνεται σε μια εντολή του συστήματος (command line arguments). Η C έχει δύο ειδικές παραμέτρους : argc και argv. Η argc είναι τύπου integer και δηλώνει το πλήθος των παραμέτρων. Η argv είναι ένας δείκτης σε array από δείκτες χαρακτήρων. Κάθε στοιχείο αυτού του array δείχνει σε μία παράμετρο. Οι παράμετροι αυτές χρησιμοποιούνται για να περάσουμε ονόματα αρχείων ή άλλες επιλογές. (Θα πρέπει να ενσωματώσουμε την βιβλιοθήκη <stdlib.h>) Το παρακάτω πρόγραμμα παίρνει ως παράμετρο ένα αριθμό {count=atoi(argc[1])} και μετρά προς τα κάτω μέχρι το μηδέν οπότε χτυπά το κουδούνι. Αν η δεύτερη παράμετρος είναι η display τότε τα αποτελέσματα εμφανίζονται στην οθόνη. Η πρώτη παράμετρος (argv[0]) είναι πάντα τ’ όνομα του προγράμματος.

```
#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#include <string.h>

void main(int argc, char *argv[])

{

int disp, count;

if argc < 2 {

printf("enter the length of the count\n");

printf("on the command line\n");

exit(1);

}

if (argc==3 && !strcmp(argv[2], "display")) disp=1;
```

```
else disp=0;

for (count=atoi(argv[1]; 0; --count)

if (disp) printf("%d\n", count);

putchar(7);

}
```

Εντολή return

Η return χρησιμοποιείται είτε για να σταματήσει την εκτέλεση μιας συνάρτησης και να επιστρέψουμε στο σημείο από το οποίο καλέστηκε, είτε για να επιστρέωει μιά τιμή. Μια συνάρτηση τελειώνει όταν εκτελεστεί και η τελευταία της εντολή. Για παράδειγμα έχουμε μια συνάρτηση που δεν επιστρέφει τίποτε αλλά απλως τυπώνει ένα μήνυμα. Συνήθως όμως οι συναρτήσεις επιστρέφουν μια τιμή και αυτό γίνεται με την εντολή return. Μια συνάρτηση μπορεί να περιέχει περισσότερα από ένα return.

Η παρακάτω συνάρτηση ψάχνει να βρεί αν υπάρχει μία υπο συμβολοσειρά (s2) μέσα σε μια άλλη συμβολοσειρά (s1). Οταν την βρεί επιστρέφει την θέση του πρώτου χαρακτήρα της s2 διαφορετικά επιστρέφει -1.

```
int find_substr(char *s1, char *s2)

{

register int t;

char *p, *p2;

for (t=0; s1[t]; t++) {

p= &s1[t];

p2= s2;

while (*p2 && *p2==*p) {

p++;

p2++;

}

if (!*p2) return t;

}
```

```
return -1;

}
```

Ορισμένες συναρτήσεις δεν επιστρέφουν καμία τιμή όπως για παράδειγμα η exit() που έχει ως αποτέλεσμα να τερματίσει ένα πρόγραμμα. Ολες οι συναρτήσεις που δεν επιστρέφουν τιμη δηλώνονται τύπου void. Μια void συνάρτηση δεν μπορούμε να την χρησιμοποιήσουμε σε μια παράσταση. Οταν ο τύπος μιας συνάρτησης δεν είναι δηλωμένος τότε αυτόματα θεωρείται ότι είναι τύπου integer. Υπάρχουν δύο τρόποι για να δηλώσουμε τον τύπο μιας συνάρτησης.

Ι. Με τον καθιερωμένο τρόπο ο τύπος μιας συνάρτησης καθορίζεται στην αρχή του προγράμματος π.χ.

```
#include <stdio.h>

float sum();
```

Η δήλωση μιας συνάρτησης με τον καθιερομένο τρόπο έχει την μορφή:

<type> <function_name>();

(ακόμα και όταν η συνάρτηση έχει παραμέτρους).

Μια συνάρτηση που επιστρέφει char μπορεί να μην δηλωθεί ως char. Η C χειρίζεται την μετατροπή ακεραίων σε χαρακτήρες και αντίστροφα.

ΙΙ. Πρωτότυπο συνάρτησης (function prototype)

Στη περίπτωση αυτή παραθέτουμε και τις παραμέτρους στην εντολή δήλωσης του τύπου της συνάρτησης. Με τον τρόπο αυτό έχουμε αυστηρότερο έλεγχο στις παραμέτρους. Η δήλωση μιας συνάρτησης με τον τρόπο αυτό έχει την μορφή

<type> <function name>(list of paremeters);

Όταν μια συνάρτηση δεν έχει παραμέτρους τότε στη δήλωση του πρωτοτύπου της δηλώνουμε void. π.χ.

```
float f(void);
```

Το πρωτότυπο συνάρτησης απαγορεύει τον αυτόματο μετασχηματισμό τύπων π.χ. από char σε int ή από float σε double.

Σύμφωνα με το πρότυπο ANSI η συνάρτηση main() επιστρέφει ένα ακέραιο στο καλούν υποπρόγραμμα δηλαδή στο λειτουργικό σύστημα. Αυτό γίνεται με την exit(). Οταν το main() δεν επιστρέφει τιμή τότε οι περισσότεροι μεταγλωττιστές επιστρέφουν αυτόματα 0.

Τέλος μια συνάρτηση μπορεί να έχει μεταβλητό πλήθος παραμέτρων. Αυτό δηλώνεται με τρείς τελείες ως εξής:

```
fun(int a, int b, ...)
```

Οπωσδήποτε θα πρέπει να υπάρχει μία παράμετρος. π.χ. η fun(...) δεν επιτρέπεται.

Αναδρομή

Στη C μια συνάρτηση μπορεί να καλέσει τον εαυτό της.

```
int factorial(int n)

{

int answer;

if (n==1) return 1;

answer= factorial(n-1)*n;

return answer;

}
```

Πλατφόρμα υλοποίησης

Υπάρχουν πολλές διαφορετικές αρχιτεκτονικές CPU που χρησιμοποιούνται στα εμπεδωμένα συστήματα όπως ο ARM, MIPS, Coldfire/68k, PowerPC, X86, PIC, 8051, Atmel AVR, H8, SH, V850, FR-V, M32R

Εφαρμογή

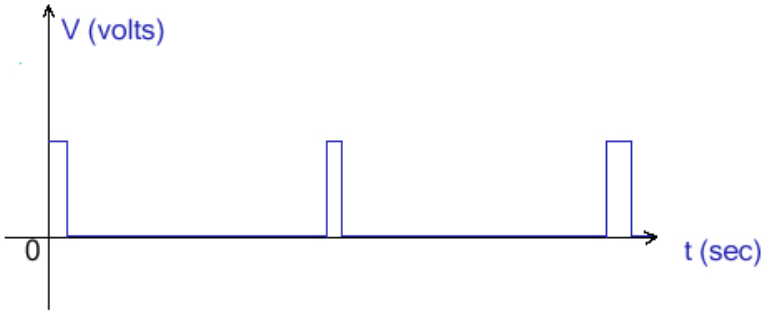
Περιγραφή συστήματος

Σκοπός της παρούσας εφαρμογής είναι να σχεδιαστεί ένα σύστημα που δυναμικά να μπορεί να δέχεται σαν είσοδο ένα παλμό μεταβλητού πλάτους από 1-2 ms και να παράγει στη έξοδο του τα σήματα ελέγχου μίας Η-γέφυρας που θα ελέγχει την φορά

και την ταχύτητα ενός κινητήρα συνεχούς ρεύματος, δηλαδή ένα περιοδικό παλμό συχνότητας f και με κύκλο καθήκοντος (duty cycle) 0 έως 100% ανάλογο του ρεύματος που διαρρέει τον κινητήρα και επομένως * τον στροφών που αναπτύσσει .Επίσης στην έξοδο είναι απαραίτητο ένα σήμα για τον καθορισμό της φόρας του ρεύματος που εισέρχεται στην περιέλιξη του κινητήρα.

Σήμα εισόδου : Σταθερής συχνότητας f = 40 Hz (έως 50 Hz)

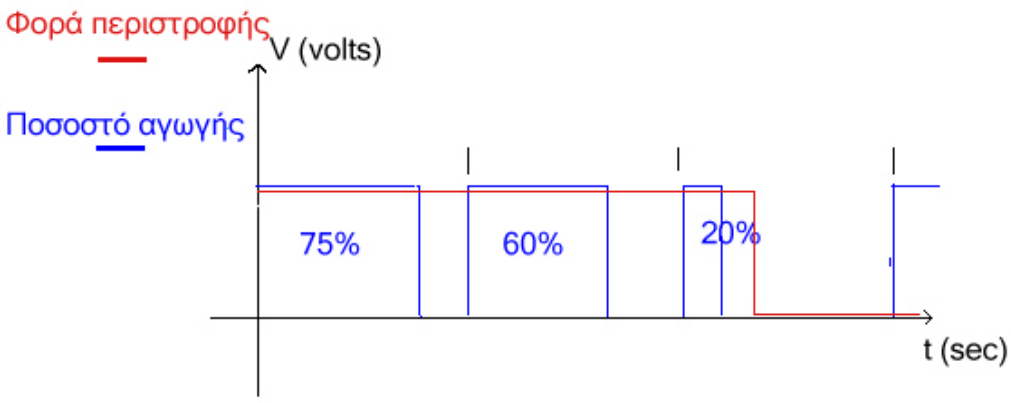
Με χρόνο t_{high}=1-2 msec



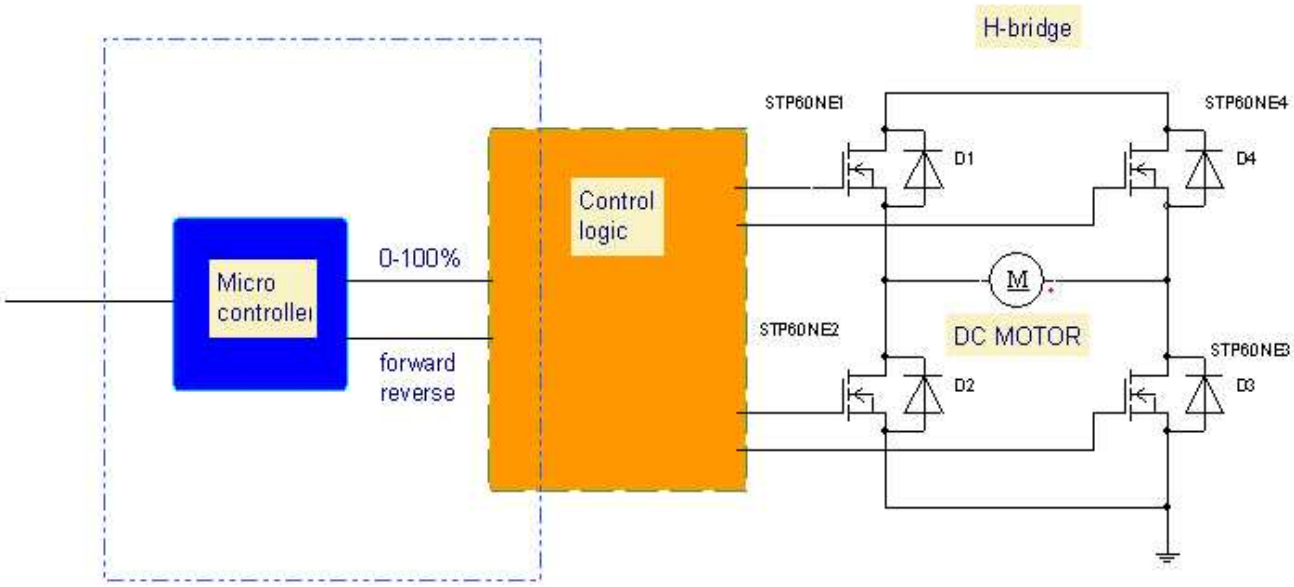
Επεξήγηση:

Το σήμα αυτό εισάγεται στο μικροελεγκτή από μια πόρτα. Το πρόγραμμα του μικροελεγκτή αναλαμβάνει να μετρά το t_{high} του παλμού κάθε φορά που αυτό εμφανίζεται και να αλλάζει την κατάσταση στην έξοδο που αφορά την φορά αγωγής της γέφυρας Η και στην έξοδο που ελέγχει το πόσο της 100% θα άγει η γέφυρα.Για σήμα 1,5 msec ή 1500μsec (δηλ για XTAL12 MHz 1500 κύκλους) Θεώρούμε ότι η γέφυρα δεν άγει καθόλου ενώ για σήμα από 1,5-1 msec αυξάνει αναλογικά την αγωγή της μέχρι να φτάσει στο 100% στο 1 msec . Και για σήμα από 1,5-2 msec άγει κατά την άλλη φορά αναλογικά από 0 έως 100%.

Σήματα εξόδου



Μπλοκ Διάγραμμα Συστήματος



Ο μικροελεκτής και η εργασία που επιτελεί

Πηγαίος κώδικας

```
|       Mavroudis Georgios.                                                    |
|                                                                              |
+------------------------------------------------------------------------------+
|       Project: Speed controller                                              |
|       File: STDIO.H                                                          |
|                                                                              |
+--------------+-------+-----------------------------------------------+--------+
|     DATE     |VERSION| HISTORY/CHANGES                               |   BY   |
|  (DD/MM/YY)  | VX.Y  |                                               |        |
+--------------+-------+-----------------------------------------------+--------+
|   27/07/04   |  1.0  | Initial implementation                        |        |
|   ??/??/??   |  1.1  | Reimplementation                              |        |
+--------------+-------+-----------------------------------------------+--------+
|Purpose    :                                                                  |
|   Standard I/O functions                                                     |
+------------------------------------------------------------------------------*/
```

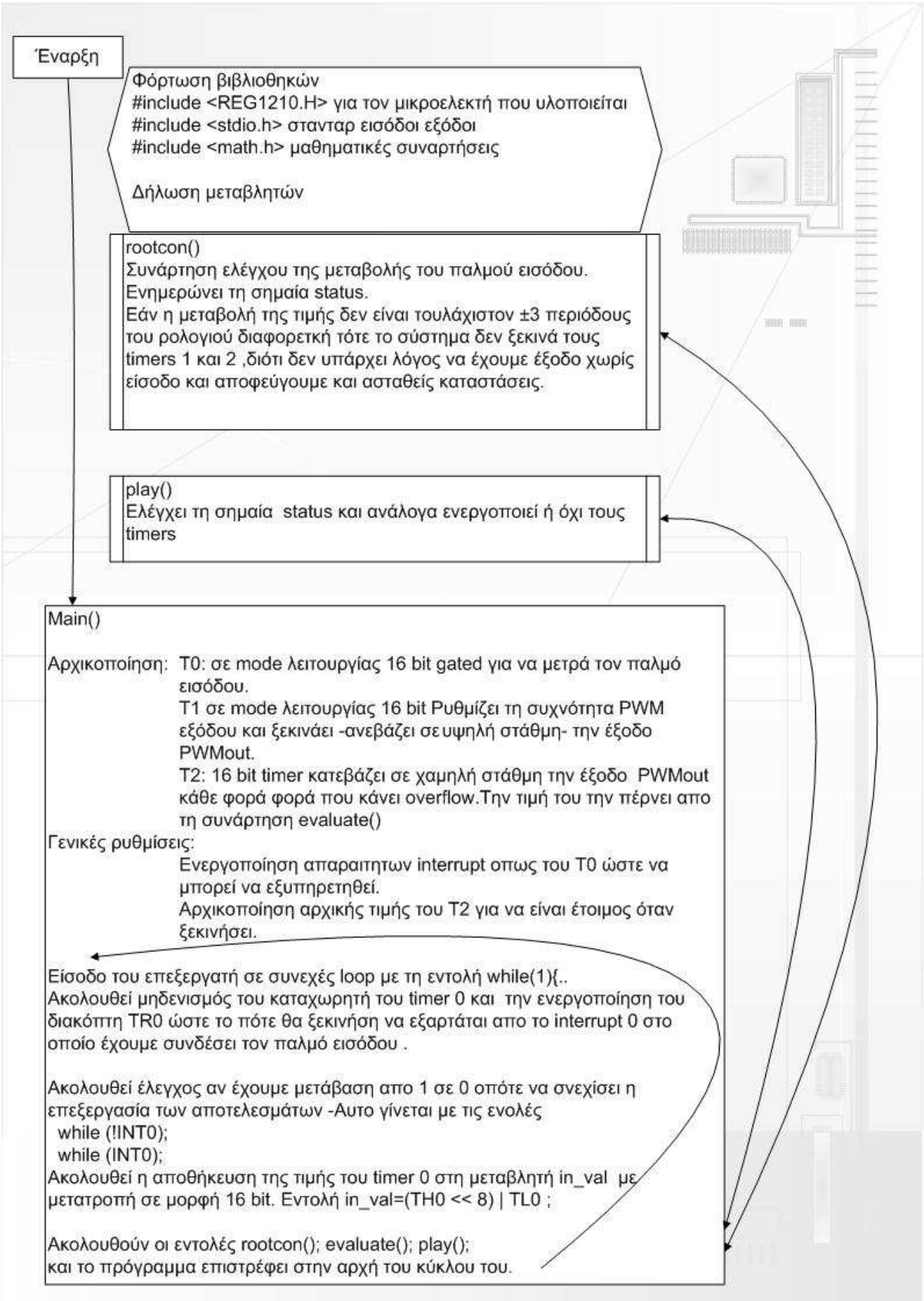
```
#include <REG1210.H>
#include <stdio.h>
#include <math.h>

//Global Variables
int tmp,in_val,in_valy,in_val_old,z;
char high,low,p[3],i,k,high2,low2,status;
sbit PWMout=P3^7;
sbit test=P3^6;

//
//Functions:
//-Root control

void rootcon(void)
{
  if (in_val>(in_val_old-3) && in_val<(in_val_old+3))
     status=0;
  else
     status=1;
}
//
//-Evaluate
void evaluate (void)
{in_val_old=in_val; //Krataei tin proigoumrni timi gia sigrisi
//in_val=(TH0 << 8) | TL0 ;
   k=1;
   if (in_val<=1500)
      k=-1;
   in_valy=k*(2*in_val-0xBB8);
   //Decimal to Hex convertion
   i=2;
    for (i=2;i>=0;i=i-1)
       {if (i==2)
        tmp=in_valy;
         else
          tmp=in_valy-(p[i+1]*pow(16,i+1));

    if (tmp>=pow(16,i))
      p[i]=tmp/pow(16,i);
     else
      p[i]=0;
    high2=0xFF-p[2];
    low2=0xFF-(p[0]+16*p[1]);
       }
}
```

```
//

//-Play
void play (void)
{
if (status==1)
TR2=1;
TR1=1;
}
//
/*-------------------------------------------------------------
Timer 0 Overflow Interrupt
-------------------------------------------------------------*/
void T0_ISR (void) interrupt 1
{
TF0 = 0;// Reset the interrupt request
}
//
/*-------------------------------------------------------------
Timer 1 Overflow Interrupt
-------------------------------------------------------------*/
void T1_ISR (void) interrupt 3
{
TF1=0;
TH2=high2;
TL2=low2;
PWMout=1; //Controls overflow for PWM_high time

TH1 =0xff-0x03;
TL1 =0xff-0xE8;
TR1=1;
TR2=1;
}
```

```
/*---------------------------------------------------------
Timer 2 Overflow Interrupt
---------------------------------------------------------*/
void T2_ISR (void) interrupt 5
{
TR2=0;
TF2=0;
PWMout=0;
}
//

/*---------------------------------------------------------
MAIN C function
---------------------------------------------------------*/
void main (void)
{

SCON0 =0x50;
TMOD |=0x19;
ET2   =1;
TH1   =0xff-0x03;
TL1   =0xff-0xE8;

ET1   =1;
TI    =1;
PCON |= 0x80;
PWMout=0;
//printf ("\nSpeed controller V1\n\n");

/*--------------------------------------
```

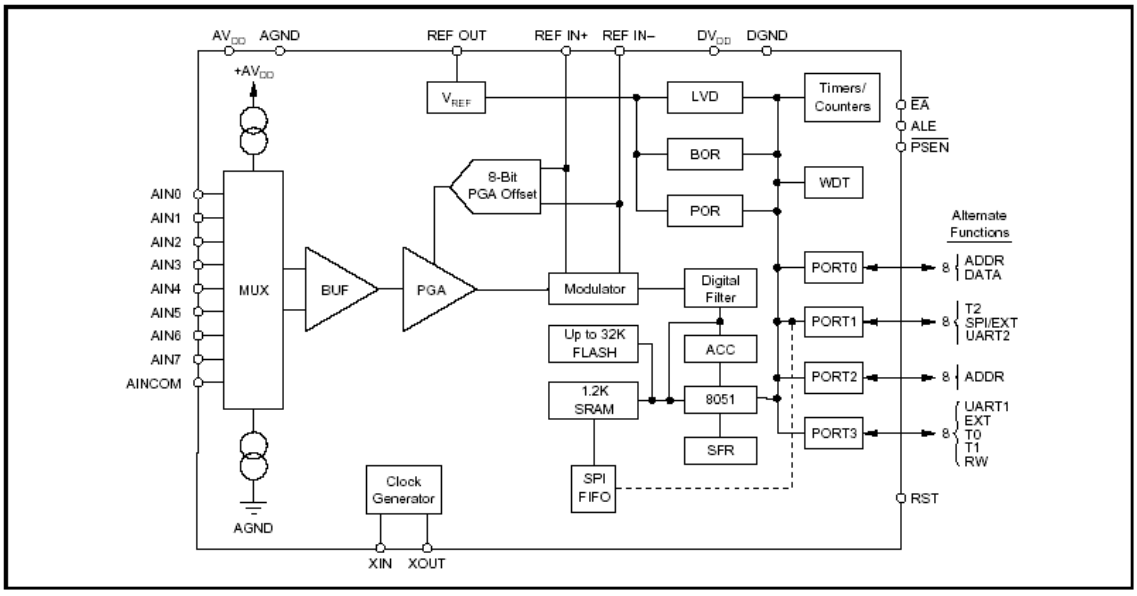
```
/*---------------------------------------
Enable interrupts for timer 0.
---------------------------------------*/
ET0 = 1;
IT1=1;
EA = 1;
//
TH2=0xFC;
TL2=0x17;


while (1)
 {
  TH0 = 0;
  TL0 = 0;
  EX1 = 1;
  TR0 = 1;

/*---------------------------------------
Wait for the pulse to start.
Then, wait for the pulse to end.
---------------------------------------*/
  while (!INT0);
  while (INT0);
test=!test;
in_val=(TH0 << 8) | TL0 ;

rootcon();
evaluate();
play();

  }
}
```

Διάγραμμα ροής του κώδικα

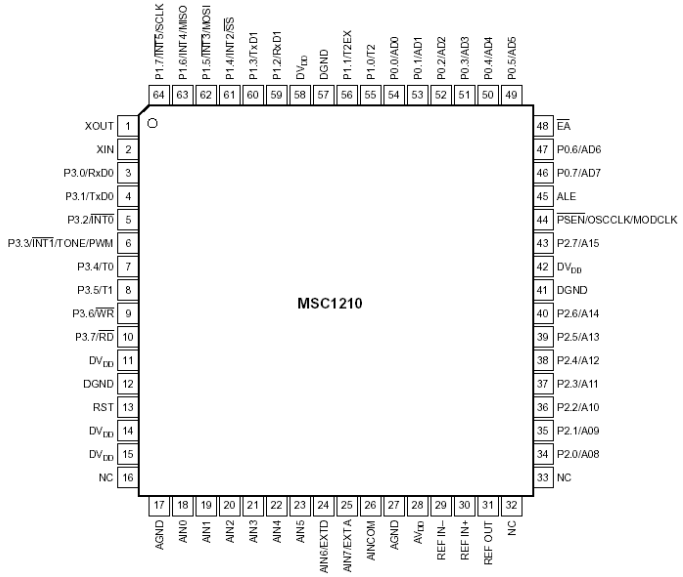


Το αναπτυξιακό σύστημα MSC1210

Μπλοκ διάγραμμα



Ο μικροελεκτής



Χαρακτηριστικά τουMSC1210

Precision Analog-to-Digital Converter (ADC) with 8051 Microcontroller and 16k Flash Memory

- **ANALOG FEATURES**
- 24-BITS NO MISSING CODES
- 22-BITS EFFECTIVE RESOLUTION AT 10Hz
  Low Noise: 75nV
- PGA FROM 1 TO 128
- PRECISION ON-CHIP VOLTAGE REFERENCE:Accuracy: 0.2%
  Drift: 5ppm/°C
- 8 DIFFERENTIAL/SINGLE-ENDED CHANNELS
- ON-CHIP OFFSET/GAIN CALIBRATION
- OFFSET DRIFT: 0.02PPM/°C
- GAIN DRIFT: 0.5PPM/°C
- ON-CHIP TEMPERATURE SENSOR
- BURN-OUT SENSOR DETECTION
- SINGLE-CYCLE CONVERSION
- SELECTABLE BUFFER INPUT
- **DIGITAL FEATURES**
- Microcontroller Core
- 8051 COMPATIBLE
- HIGH SPEED CORE: 4 Clocks per Instruction Cycle
- DC TO 33MHz
- SINGLE INSTRUCTION 121ns
- DUAL DATA POINTER
- 0Memory
- UP TO 32kB FLASH MEMORY
- FLASH MEMORY PARTITIONING
- ENDURANCE 1M ERASE/WRITE CYCLES, 100 YEAR
  DATA RETENTION
- IN-SYSTEM SERIALLY PROGRAMMABLE
- EXTERNAL PROGRAM/DATA MEMORY (64kB)
- 1,280 BYTES DATA SRAM
- FLASH MEMORY SECURITY
- 2kB BOOT ROM
- PROGRAMMABLE WAIT STATE CONTROL
- **Peripheral Features**
- 34 I/O PINS
- ADDITIONAL 32-BIT ACCUMULATOR
- THREE 16-BIT TIMER/COUNTERS
- SYSTEM TIMERS
- PROGRAMMABLE WATCHDOG TIMER
- FULL DUPLEX DUAL UART
- MASTER/SLAVE SPI™
- 16-BIT PWM
- POWER MANAGEMENT CONTROL
- IDLE MODE CURRENT <1mA
- STOP MODE CURRENT <1 µA
- PROGRAMMABLE BROWNOUT RESET
- PROGRAMMABLE LOW VOLTAGE DETECT
- 21 INTERRUPT SOURCES

- TWO HARDWARE BREAKPOINTS
- **GENERAL FEATURES**
- PACKAGE: TQFP-64
- LOW POWER: 4mW
- INDUSTRIAL TEMPERATURE RANGE: –40°C to +85°C
- POWER SUPPLY: 2.7V to 5.25V

APPLICATIONS

- INDUSTRIAL PROCESS CONTROL
- INSTRUMENTATION
- LIQUID/GAS CHROMATOGRAPHY
- BLOOD ANALYSIS
- SMART TRANSMITTERS
- PORTABLE INSTRUMENTS
- WEIGH SCALES
- PRESSURE TRANSDUCERS
- INTELLIGENT SENSORS
- PORTABLE APPLICATIONS
- DAS SYSTEMS

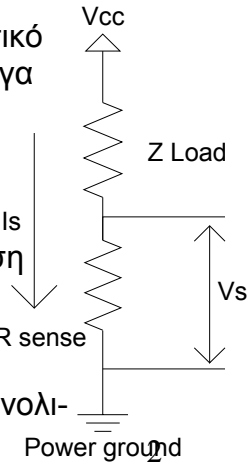
Τεχνικές μέτρησης-ανίχνευσης ρεύματος

Αν και υπάρχουν πολλές τεχνικές ανίχνευσης ρεύματος ωστόσο μόνο τρεις λαμβάνονται σαν δεδομένες στις περισσότερες χαμηλού κόστους εφαρμογές. Οι υπόλοιπες είναι ακριβά εργαστηριακά συστήματα από ανερχόμενες τεχνολογικές γνώσεις (όπως τα magnetoresistives) που σπάνια χρησιμοποιούνται. Στις συχνότερα χρησιμοποιούμενες τεχνικές συμπεριλαμβάνονται οι παρακάτω τρεις τρόποι:

1. Με ωμικής αντίστασης.
2. Γραμμικά αισθητήρια φαινομένου Hall.
3. Μετασχηματιστή ρεύματος.

1.Ανίχνευση ρεύματος με ωμική αντίσταση.

Η τεχνική με ωμική αντίσταση χρησιμοποιεί σταθερή αντίσταση με πολύ μικρή τιμή, ώστε να μην προκαλεί ουσιαστική απώλεια. Η πτώση τάσης στα άκρα της αντίστασης είναι ανάλογη προς το ρεύμα και μπορεί να μετρηθεί με βολτόμετρο, παλμογράφο ή άλλη συσκευή για μέτρηση, παρατήρηση. Η ενδεικτική τιμή μιας τέτοιας αντίστασης
–shunt resistor- είναι το πολύ 100mΩ, ενώ άλλο βασική χαρακτηριστικό της είναι η μέγιστη ισχύς που μπορεί να αντέξει και επιλέγεται ανάλογα με τις απαιτήσεις της εφαρμογής.



Το ρεύμα I που διαρρέει τόσο την αντίσταση φορτίου όσο και την αντίσταση Rsense είναι: Is=Vcc/(Zload + Rsense)
και επειδή η Rsense είναι σχεδόν μηδενική μπορεί να γίνει η σύμβαση ότι το ρεύμα δεν επηρεάζεται ιδιαίτερα από την ύπαρξη της Rsense.

Η τάση στα άκρα της αντίστασης είναι σύμφωνα με το νόμο του Ωμ Vs=Is*Rsense με την Rsense να έχει ελάχιστη επίπτωση στη συνολι-

κή εμπέδηση. Ενώ από την προηγούμενη σχέση φαίνεται ότι η τιμή της τάσης είναι ανάλογη του ρεύματος που την διαρρέει.

Βασικό πλεονέκτημα αυτής της μεθόδου είναι η απλότητα της το μικρό μέγεθος που καταλαμβάνει και η γραμμικότητα της. Ενώ μείζων μειονέκτημα της είναι η απώλεια ισχύος
που εμφανίζεται πάνω στην αντίσταση ανίχνευσης Rsense σαν θερμότητα.

2.Ανίχνευση ρεύματος με το φαινόμενο Hall.

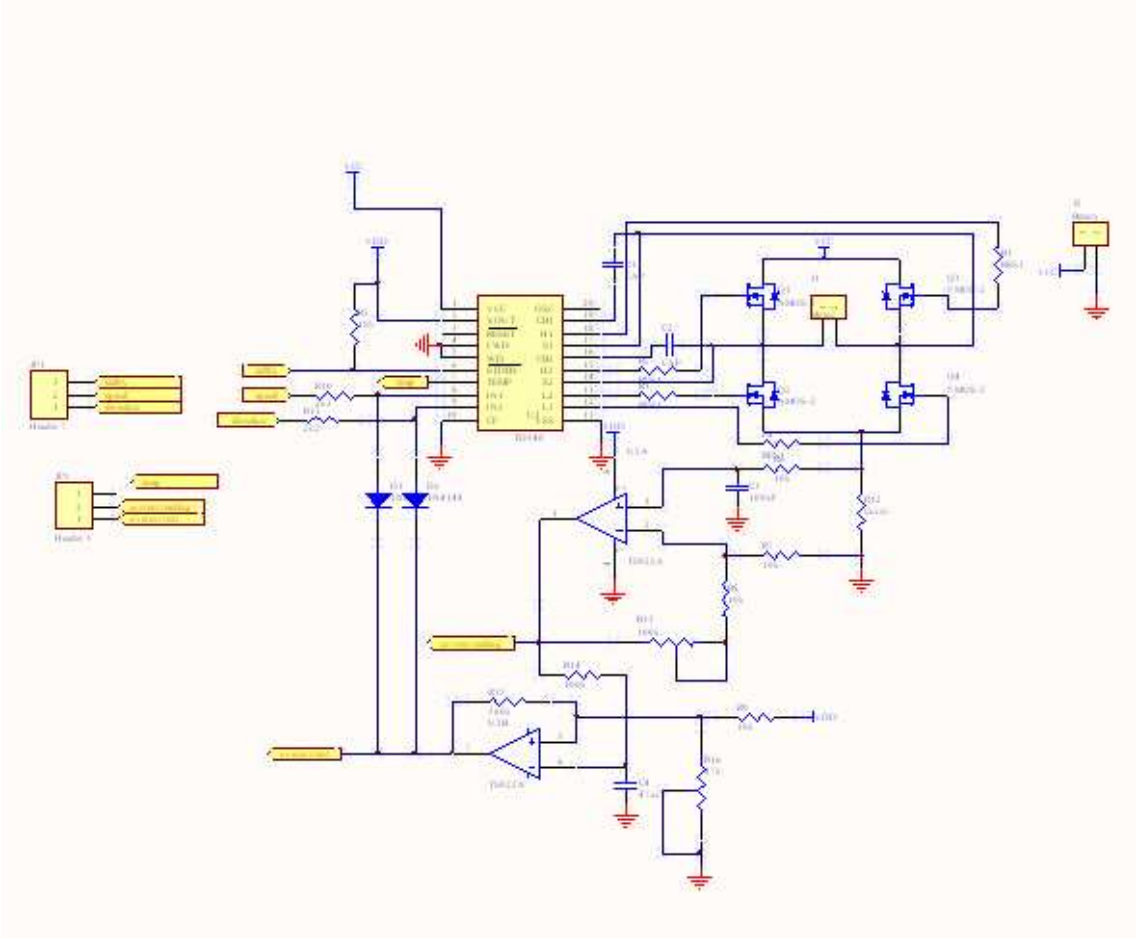
Το φαινόμενο Hall

Το φαινόμενο Hall που για πρώτη φορά παρατηρήθηκε το 1879, συναντάται όταν η επιφάνεια ενός λεπτού φύλλου χρυσού ή άλλου μετάλλου τοποθετηθεί κάθετα σε μαγνητικό πεδίο, και αφήσουμε να διαρρέετε από ρεύμα από τη μια μεριά ως την άλλη, τότε ανάμεσα στις δύο άλλες πλευρές του θα αναπτυχθεί ηλεκτρική τάση. Τα φορτισμένα σωματίδια δέχονται μια δύναμη από το μαγνητικό πεδίο. Η διαφορά δυναμικού ανάμεσα στις δύο πλευρές εξαρτάται από την ένταση του μαγνητικού πεδίου.

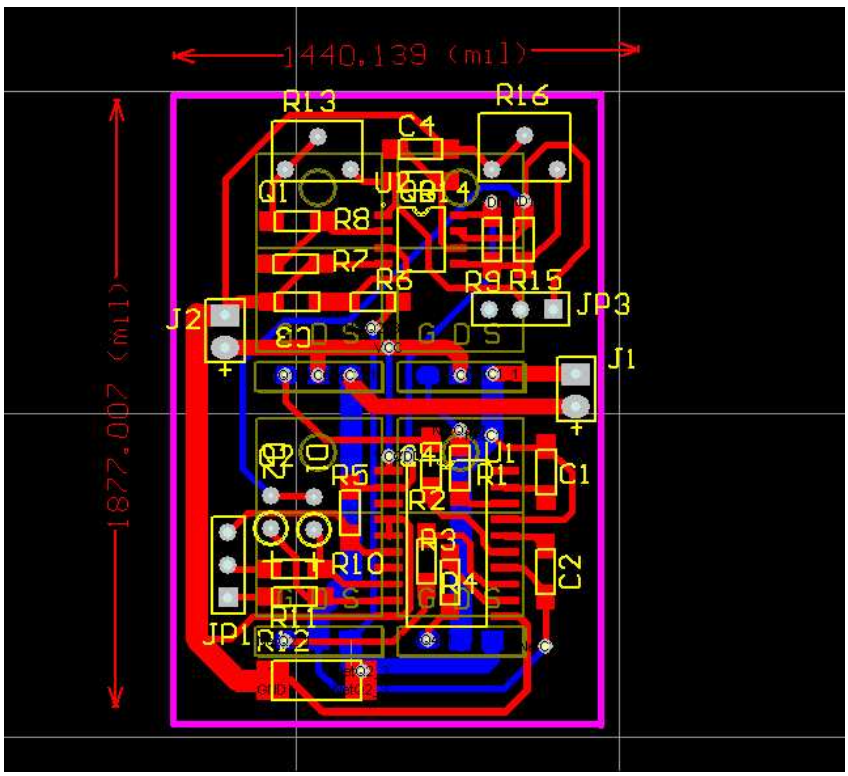
3.Ανίχνευση ρεύματος με μετασχηματιστή ρεύματος.

Η ανίχνευση ρεύματος με μετασχηματιστή ρεύματος δεν επιτρέπει την οικονομική σχεδίαση και φυσικά είναι χρήσιμη μόνο στο εναλλασσόμενο ρεύμα. Οι περισσότεροι μετασχηματιστές ρεύματος είναι σχεδιασμένοι για μικρό εύρος ζώνης, είναι ακριβότεροι από τις δυο παραπάνω μεθόδους και δεν μπορούν να χρησιμοποιηθούν για συνεχές ρεύμα. Ωστόσο με τους μετασχηματιστές ρεύματος αποφεύγουμε τις απώλειες παρεμβολής , έχουμε ηλεκτρική απομόνωση ,δεν χρειαζόμαστε εξωτερική τροφοδοσία και έχουμε μηδενική τάση μετατόπισης για μηδενικά επίπεδα ρευμάτων.

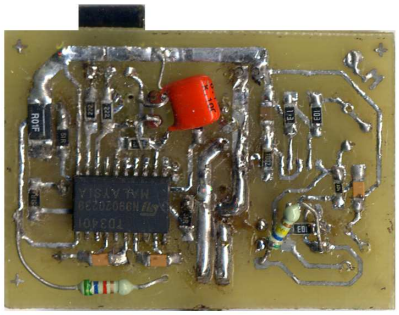
Το ηλεκτρονικό σχέδιο της λογικής έλεγχου της γέφυρας Η.



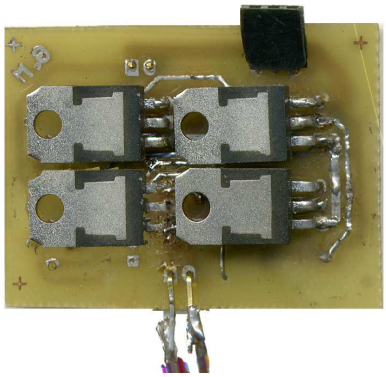
Το τυπωμένο κύκλωμα



Η κατασκευή της πλακέτας



Top Layer



Bottom Layer

Λειτουργία της κατασκευής

Η κατασκευή λειτούργησε κανονικά και κατά την ορθή φορά αγωγής και κατά της ανάστροφη. Ωστόσο με τον καιρό άρχισε να παρουσιάζει πρόβλημα με την θερμοκρασία που εκλυόταν από τα N mosfet της πάνω πλευράς με αποτέλεσμα να καταστρέφονται κατά διαστήματα. Το πρόβλημα πιθανόν συνδέεται με τη λειτουργία synchronous rectification που επιτελεί το TD340 και χρειάζεται λεπτομερέστερη εξέταση στο χρονισμό των παλμό πυροδότησης του ημιαγωγών ώστε να αποκλειστεί η περίπτωση ταυτόχρονης αγωγής των mosfet 1 και 2 ή 2 και 3 που οδηγεί σε βραχυκύκλωμα .

ΙΣΤΟΣΕΛΙΔΕΣ

Wikipedia, ελεύθερη εγκυκλοπαίδεια.

http://en.wikipedia.org

Υλικό σχετικό με τον προγραμματισμό εμπεδωμένων συστημάτων
www.lightplanet.com

Η ιστοσελίδα της keil
www.keil.com

Η ιστοσελίδα του Protel
www.altium.com

ΒΙΒΛΙΟΓΡΑΦΙΑ

C Programming for Engineering & Computer Science H.H Tan – T.B. D'Orazio
(C για μηχανικούς, εκδόσεις Τζιόλα)

C Programming for EMBEDDED SYSTEMS KIRK ZURELL
(Apply C to 8-bit Microprocessors for efficient Development)

Embedded Systems Building Blocks Second Edition JEAN J.LABROSSE
(Complete and Ready-to-Use Modules in C)

The Development of the C Language* Dennis M. Ritchie
Bell Labs/Lucent Technologies
Murray Hill, NJ 07974 USA