INTERNATIONAL HELLENIC UNIVERSITY

Διεθνές Πανεπιστήμιο της Ελλάδος

Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων

Ενσωματωμένα Συστήματα

(6ο εξάμηνο)

04-Ανάγνωση και απόδοση τιμής σε ακροδέκτη

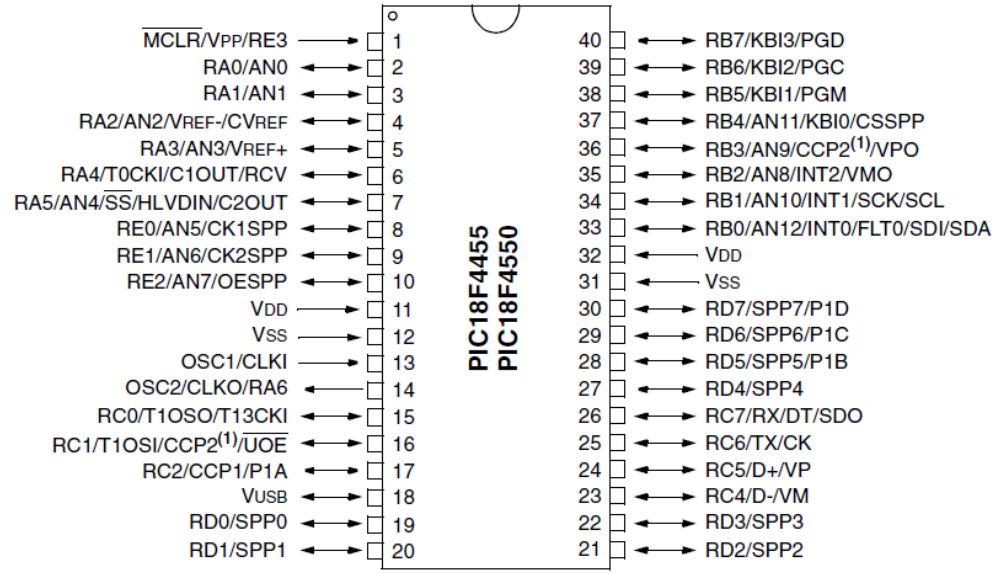
Διδάσκουσα: Παπαδοπούλου Μαρία

Επίκουρη Καθηγήτρια

Θεσσαλονίκη 2025

Παράλληλες πόρτες του μικροελεγκτή PIC 18F4550

40-Pin PDIP



- Οι μοναδικές πόρτες που μπορούν να έχουν 8 ακροδέκτες που μπορούν να χρησιμοποιηθούν ως ψηφιακές είσοδοι/ έξοδοι είναι οι πόρτες B και D
- Οι πόρτες A, C και E έχουν κάποιους ακροδέκτες προορισμένους μόνο για άλλες χρήσεις και όχι για ψηφιακούς ακροδέκτες εισόδου/ εξόδου

PIC18F2455/2550/4455/4550

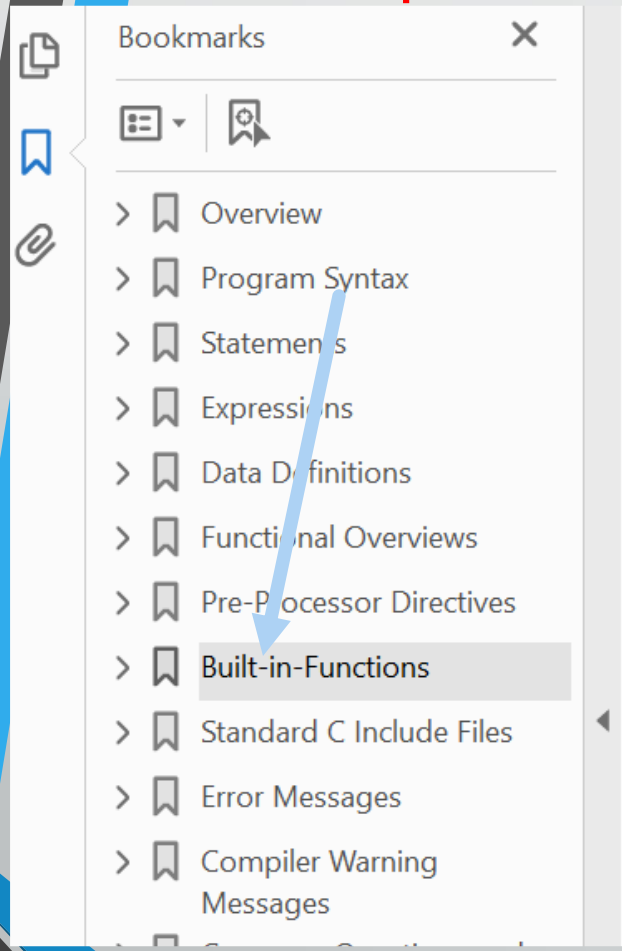
TABLE 10-1: PORTA I/O SUMMARY

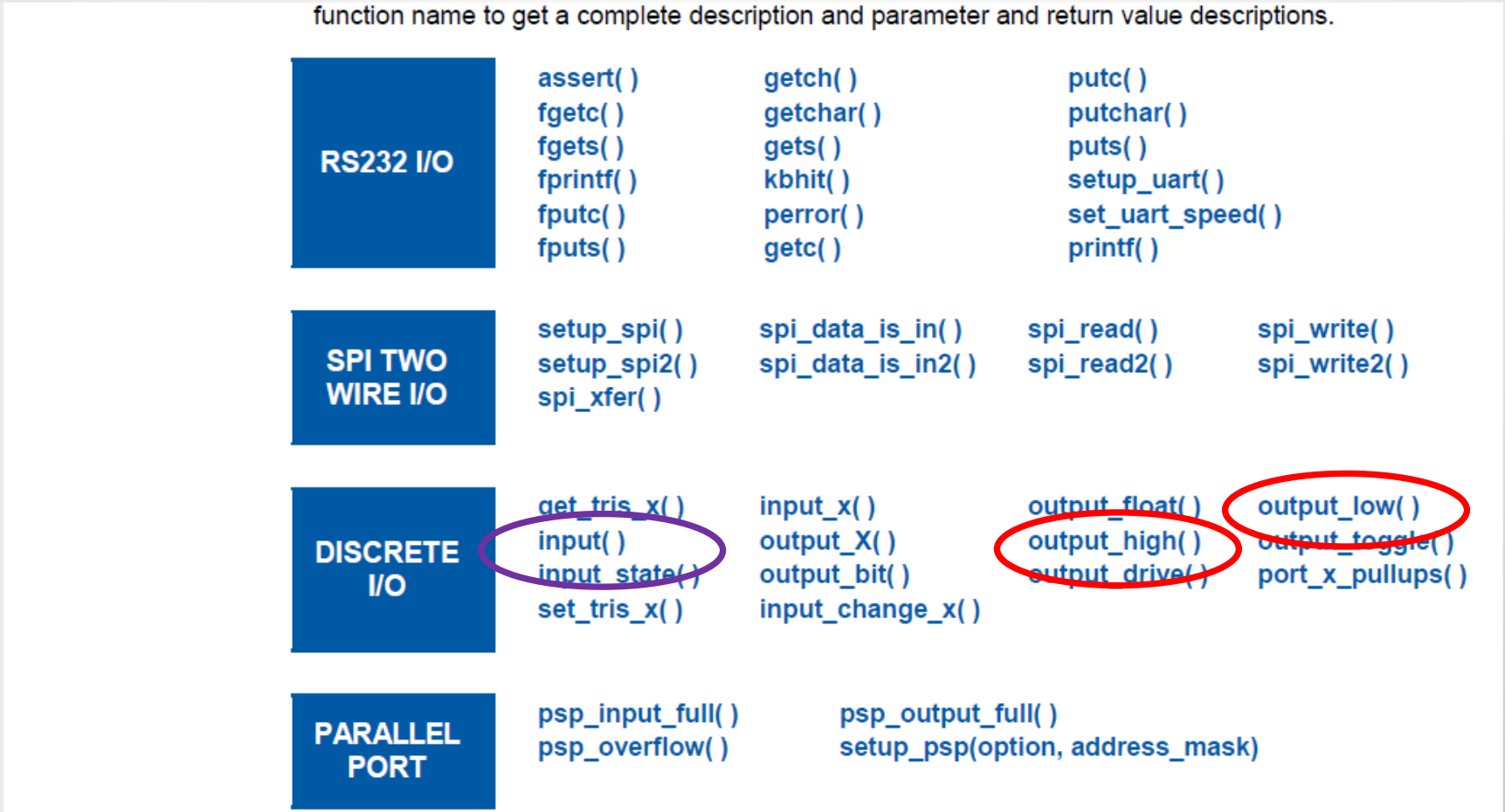
Pin	Function	TRIS Setting	I/O	I/O Type	Description
RA0/AN0	RA0	0	OUT	DIG	LATA<0> data output; not affected by analog input.
		1	IN	TTL	PORTA<0> data input; disabled when analog input enabled.
	AN0	1	IN	ANA	A/D input channel 0 and Comparator C1- input. Default configuration on POR; does not affect digital output.
RA1/AN1	RA1	0	OUT	DIG	LATA<1> data output; not affected by analog input.
		1	IN	TTL	PORTA<1> data input; reads '0' on POR.
	AN1	1	IN	ANA	A/D input channel 1 and Comparator C2- input. Default configuration on POR; does not affect digital output.
RA2/AN2/ VREF-/CVREF	RA2	0	OUT	DIG	LATA<2> data output; not affected by analog input. Disabled when CVREF output enabled.
		1	IN	TTL	PORTA<2> data input. Disabled when analog functions enabled; disabled when CVREF output enabled.
	AN2	1	IN	ANA	A/D input channel 2 and Comparator C2+ input. Default configuration on POR; not affected by analog output.
	VREF-	1	IN	ANA	A/D and comparator voltage reference low input.
	CVREF	x	OUT	ANA	Comparator voltage reference output. Enabling this feature disables digital I/O.
RA3/AN3/ VREF+	RA3	0	OUT	DIG	LATA<3> data output; not affected by analog input.
		1	IN	TTL	PORTA<3> data input; disabled when analog input enabled.
	AN3	1	IN	ANA	A/D input channel 3 and Comparator C1+ input. Default configuration on POR.
	VREF+	1	IN	ANA	A/D and comparator voltage reference high input.
RA4/T0CKI/ C1OUT/RCV	RA4	0	OUT	DIG	LATA<4> data output; not affected by analog input.
		1	IN	ST	PORTA<4> data input; disabled when analog input enabled.
	T0CKI	1	IN	ST	Timer0 clock input.
	C1OUT	0	OUT	DIG	Comparator 1 output; takes priority over port data.
	RCV	x	IN	TTL	External USB transceiver RCV input.
RA5/AN4/SS/ HLVDIN/C2OUT	RA5	0	OUT	DIG	LATA<5> data output; not affected by analog input.
		1	IN	TTL	PORTA<5> data input; disabled when analog input enabled.
	AN4	1	IN	ANA	A/D input channel 4. Default configuration on POR.
	SS	1	IN	TTL	Slave select input for SSP (MSSP module).
	HLVDIN	1	IN	ANA	High/Low-Voltage Detect external trip point input.
	C2OUT	0	OUT	DIG	Comparator 2 output; takes priority over port data.
OSC2/CLKO/ RA6	OSC2	x	OUT	ANA	Main oscillator feedback output connection (all XT and HS modes).
	CLKO	x	OUT	DIG	System cycle clock output (FOSC/4); available in EC, ECPLL and INTCKO modes.
	RA6	0	OUT	DIG	LATA<6> data output. Available only in ECIO, ECPIO and INTIO modes; otherwise, reads as '0'.
		1	IN	TTL	PORTA<6> data input. Available only in ECIO, ECPIO and INTIO modes; otherwise, reads as '0'.

Legend: OUT = Output, IN = Input, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Buffer Input, TTL = TTL Buffer Input, x = Don't care (TRIS bit does not affect port direction or is overridden for this option)

Διάβασμα από έναν ακροδέκτη εισόδου - Αποστολή τιμής σε έναν ακροδέκτη εξόδου

CCS C Compiler





Ανάγνωση τιμής από ακροδέκτη εισόδου

```
a=input(PIN_B4); //μετάφερε την τιμή του
                 //ακροδέκτη B4 στη μεταβλητή a
```

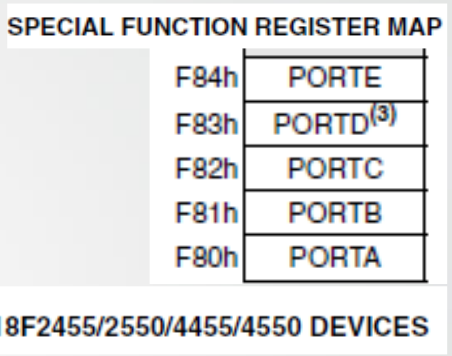
Αποστολή τιμής σε ακροδέκτη εξόδου

```
output_low(PIN_B5);   //κάνε 0 τον ακροδέκτη B5
output_high(PIN_D6); //κάνε 1 τον ακροδέκτη D6
```

Άσκηση 01e. Να γραφεί πρόγραμμα που να ελέγχει την κατάσταση των ακροδεκτών RA0 και RA1.

Αν RA0=1 και RA1=1 να τίθεται RD0=1.

Σε όλες τις άλλες περιπτώσεις τιμών των RA0 και RA1 να τίθεται RD0=0.


SPECIAL FUNCTION REGISTER MAP

F84h	PORTE
F83h	PORTD(3)
F82h	PORTC
F81h	PORTB
F80h	PORTA

FOR PIC18F2455/2550/4455/4550 DEVICES

1. Η πόρτα Α πρέπει να γίνει είσοδος και η πόρτα D έξοδος:

```
set_tris_a(0b11111111); // Όλα 1 στον καταχωρητή κατεύθυνσης της πόρτας A
set_tris_d(0b00000000); // Όλα 0 στον καταχωρητή κατεύθυνσης της πόρτας D
```

2. Πρέπει να ελέγχεται συνεχώς η κατάσταση των ακροδεκτών RA0 και RA1 και με βάση το αποτέλεσμα του ελέγχου να τίθεται η κατάλληλη τιμή στον ακροδέκτη RD0

```
int1 a;
int1 b;
while(TRUE){
    a=input(PIN_A0);
    b=input(PIN_A1);
    if(a&&b){
        output_high(PIN_D0);
    }
    else{
        output_low(PIN_D0);
    }
}
```

3. Θα μπορούσε να γραφεί και ως εξής:

```
if(input(PIN_A0) && input(PIN_A1)) {
    output_high(PIN_D0);
}
```

4. Θα μπορούσε να γραφεί και ως εξής:

```
if((a==1) && (b==1)) {
    output_high(PIN_D0);
}
```

Άσκηση 01e. Να γραφεί πρόγραμμα που να ελέγχει την κατάσταση των ακροδεκτών RA0 και RA1.

Αν RA0=1 και RA1=1 να τίθεται RD0=1.

Σε όλες τις άλλες περιπτώσεις τιμών των RA0 και RA1 να τίθεται RD0=0.

Διαφορά τελεστών && και &

Με το **&** γίνεται η **λογική πράξη AND μεταξύ δύο μεταβλητών**.

Παράδειγμα 1:

X1 = 11110000

X2 = 11000000

X1 **&** X2 = 11000000 (έγινε η λογική πράξη AND ανάμεσα στα αντίστοιχα bit)

Με το **&&** γίνεται ο **έλεγχος AND μεταξύ δύο συνθηκών**. Αν και οι δύο συνθήκες είναι αληθείς το αποτέλεσμα είναι αληθής συνθήκη.

Παράδειγμα 2:

```
If(X1>5 && X2>=0) {     }
```

- Πρέπει να σημειωθεί ότι η **αληθής συνθήκη** αντιστοιχεί στην τιμή **1**.
  Για παράδειγμα το while(TRUE) και το while(1) είναι ακριβώς το ίδιο
- Οποιαδήποτε τιμή εκτός από 1 αντιστοιχεί στην ψευδή συνθήκη

Άσκηση 01e. Να γραφεί πρόγραμμα που να ελέγχει την κατάσταση των ακροδεκτών RA0 και RA1.

Αν RA0=1 και RA1=1 να τίθεται RD0=1.

Σε όλες τις άλλες περιπτώσεις τιμών των RA0 και RA1 να τίθεται RD0=0.

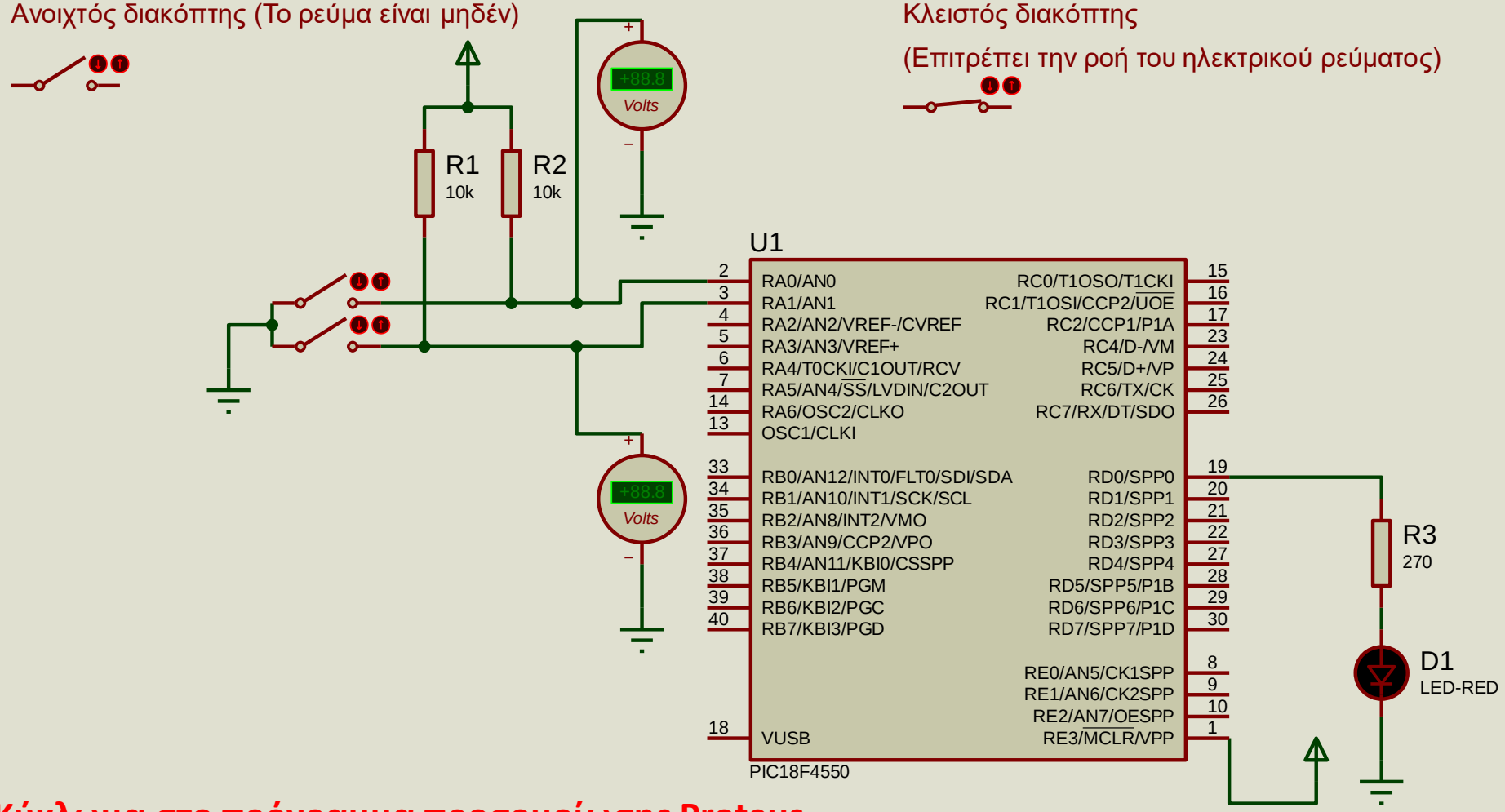
Διακόπτης σε κατάσταση OFF

Ανοιχτός διακόπτης (Το ρεύμα είναι μηδέν)

Διακόπτης σε κατάσταση ON

Κλειστός διακόπτης

(Επιτρέπει την ροή του ηλεκτρικού ρεύματος)



Κύκλωμα στο πρόγραμμα προσομοίωσης Proteus

Εντολές

output_high()

output_high( )

Syntax:	output_high (***pin***)
Parameters:	***Pin*** to write to. Pins are defined in the devices .h file. The actual value is a bit address. For example, port a (byte 5) bit 3 would have a value of 5*8+3 or 43. This is defined as follows: #DEFINE PIN_A3 43 . The PIN could also be a variable. The variable must have a value equal to one of the constants (like PIN_A1) to work properly. The tristate register is updated unless the FAST_I0 mode is set on port A. Note that doing I/0 with a variable instead of a constant will take much longer time.
Returns:	undefined
Function:	Sets a given pin to the high state. The method of I/O used is dependent on the last USE *_IO directive.
Availability:	All devices.
Requires:	Pin constants are defined in the devices .h file

Examples:	`output_high(PIN_A0);`
	
	`Int16 i=PIN_A1;`
	`output_low(PIN_A1);`

Example Files:	ex_sqw.c
Also See:	input(), output_low(), output_float(), output_bit(), output_x(), #USE FIXED_IO, #USE FAST_IO, #USE STANDARD_IO, General Purpose I/O

output_low()

output_low( )

Syntax:	output_low (***pin***)
Parameters:	***Pins*** are defined in the devices .h file. The actual value is a bit address. For example, port a (byte 5) bit 3 would have a value of 5*8+3 or 43 . This is defined as follows: #DEFINE PIN_A3 43 . The PIN could also be a variable. The variable must have a value equal to one of the constants (like PIN_A1) to work properly. The tristate register is updated unless the FAST_I0 mode is set on port A. Note that doing I/0 with a variable instead of a constant will take much longer time.
Returns:	undefined
Function:	Sets a given pin to the ground state. The method of I/O used is dependent on the last USE *_IO directive.
Availability:	All devices.
Requires:	Pin constants are defined in the devices .h file

Examples:	`output_low(PIN_A0);`
	
	`Int16i=PIN_A1;`
	`output_low(PIN_A1);`

Example Files:	ex_sqw.c
Also See:	input(), output_high(), output_float(), output_bit(), output_x(), #USE FIXED_IO, #USE FAST_IO, #USE STANDARD_IO, General Purpose I/O

Εντολή: input(pin)

input( )

Syntax:	value = input (***pin***)
Parameters:	***Pin*** to read. Pins are defined in the devices .h file. The actual value is a bit address. For example, port a (byte 5) bit 3 would have a value of 5*8+3 or 43 . This is defined as follows: #define PIN_A3 43 . The PIN could also be a variable. The variable must have a value equal to one of the constants (like PIN_A1) to work properly. The tristate register is updated unless the FAST_I0 mode is set on port A. note that doing I/0 with a variable instead of a constant will take much longer time.
Returns:	0 (or FALSE) if the pin is low, 1 (or TRUE) if the pin is high
Function:	This function returns the state of the indicated pin. The method of I/O is dependent on the last USE *_IO directive. By default with standard I/O before the input is done the data direction is set to input.
Availability:	All devices.
Requires:	Pin constants are defined in the devices .h file

Examples:	`while ( !input(PIN_B1) );`
	`// waits for B1 to go high`
	
	`if( input(PIN_A0) )`
	`  printf("A0 is now high\r\n");`
	
	`int16 i=PIN_B1;`
	`while(!i);`
	`//waits for B1 to go high`

Example Files:	ex_pulse.c
Also See:	input_x(), output_low(), output_high(), #USE FIXED_IO, #USE FAST_IO, #USE STANDARD_IO, General Purpose I/O

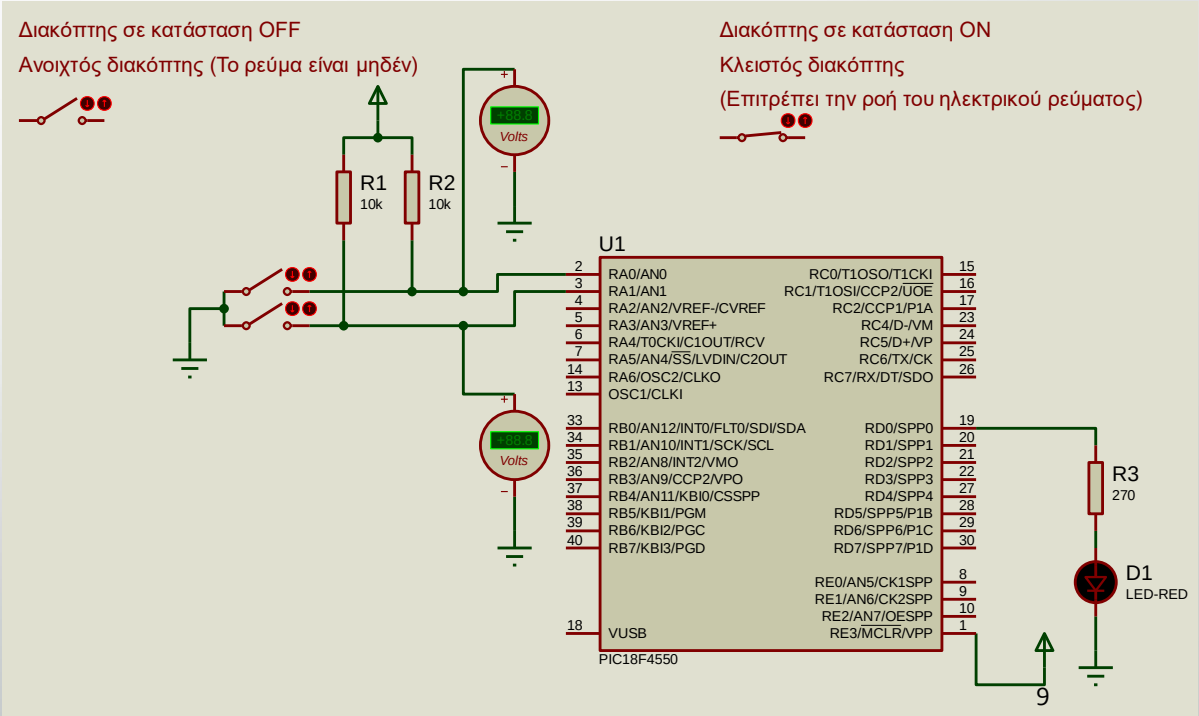
Άσκηση 01e. Να γραφεί πρόγραμμα που να ελέγχει την κατάσταση των ακροδεκτών RA0 και RA1.

Αν RA0=1 και RA1=1 να τίθεται RD0=1.

Σε όλες τις άλλες περιπτώσεις τιμών των RA0 και RA1 να τίθεται RD0=0.

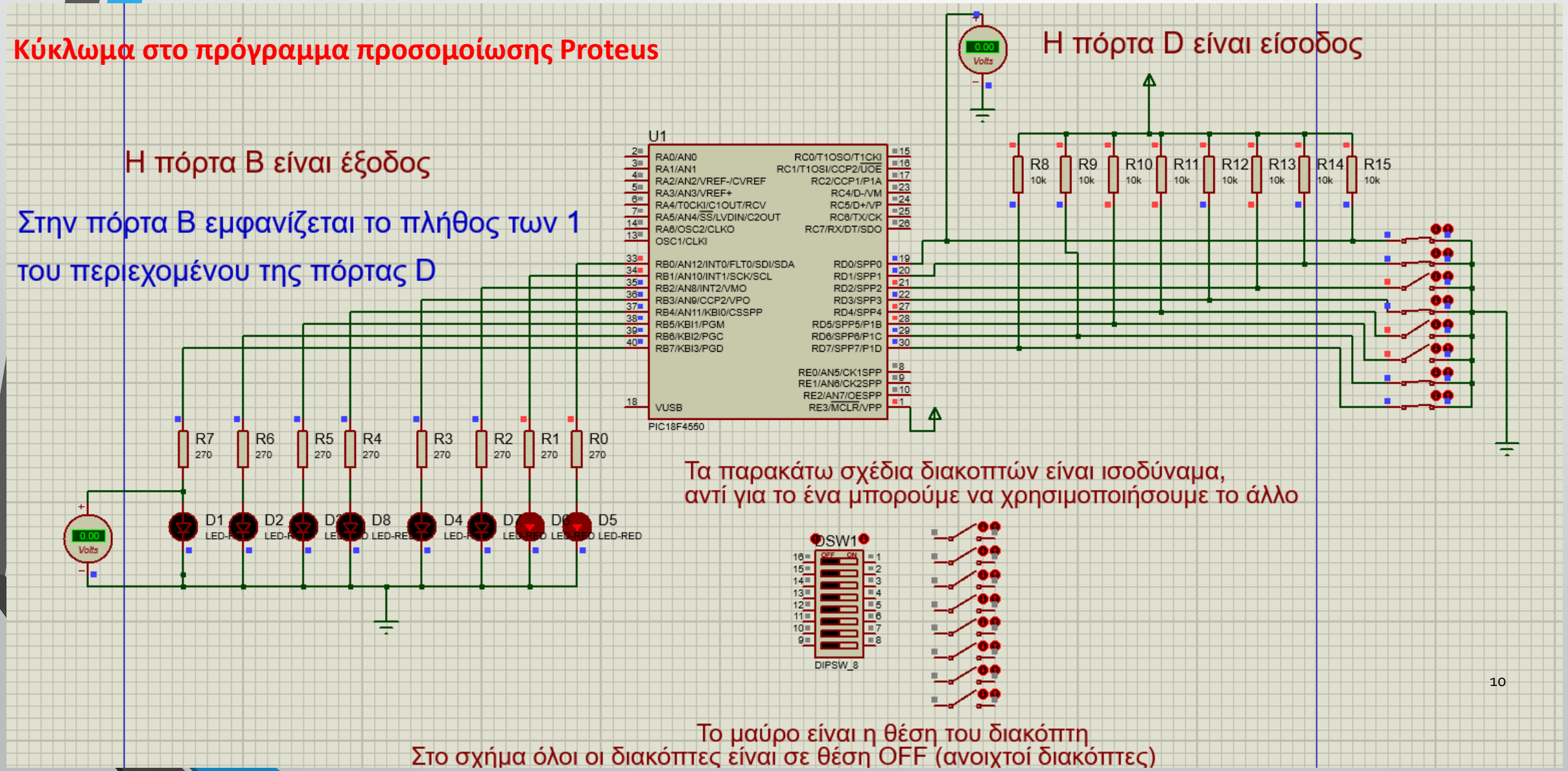
exercise_01e.c

```
#include<main.h>  //Το αρχείο <main.h> περιέχει αρχικές ρυθμίσεις
                  //Πρέπει να τοποθετηθεί οπωσδήποτε στον ίδιο φάκελο στον οποίο θα
                  //αναπτύξετε το project σας.
#byte PORTA=0xF80 //F80 είναι η θέση τη καταχωρητή δεδομένων της πόρτας A
                  // στην μνήμη του μικροελεγκτή
#byte PORTD=0xF83  //F83 είναι η θέση τη καταχωρητή δεδομένων της πόρτας D
                  // στην μνήμη του μικροελεγκτή

// ********Από εδώ αρχίζει το κύριο πρόγραμμα**********************

void main()
{                 //άνοιγμα αγκύλης της συνάρτησης main

set_tris_a(0xff);   //Η θύρα A γίνεται είσοδος(καταχωρητής κατεύθυνσης=1111 1111)
set_tris_d(0x00);   //Η θύρα D γίνεται έξοδος (καταχωρητής κατεύθυνσης=0000 0000)


int1 a;             // Ορισμός ακέραιης μεταβλητής a του 1 bit για αποθήκευση
                    // του περιεχομένου του ακροδέκτη A0
int1 b;             // Ορισμός ακέραιης μεταβλητής a του 1 bit για αποθήκευση
                    // του περιεχομένου του ακροδέκτη A1

while(TRUE) {      //Βρόχος που δεν τελειώνει ποτέ(συνθήκη πάντα αληθής)
            a=input(PIN_A0);
            b=input(PIN_A1);
            if(a&&b)      {
                           output_high(PIN_D0);
                          }
            else          {
                           output_low(PIN_D0);
                          }

            }   //κλείσιμο της αγκύλης του while

}      // κλείσιμο της αγκύλης του main
```



Άσκηση 01f. Μέτρηση των 1 στους ακροδέκτες εισόδου της PORTD και εμφάνιση του αποτελέσματος στην PORTB

Κύκλωμα στο πρόγραμμα προσομοίωσης Proteus



Άσκηση 01f. Μέτρηση των 1 στους ακροδέκτες εισόδου της PORTD και εμφάνιση του αποτελέσματος στην PORTB

Πρόγραμμα σε γλώσσα C

```
void main()
{                    //άνοιγμα αγκύλης της συνάρτησης main

set_tris_b(0x00);    //Η θύρα Β γίνεται έξοδος(καταχωρητής κατεύθυνσης=0000 0000)
set_tris_d(0xff);    //Η θύρα D γίνεται είσοδος(καταχωρητής κατεύθυνσης=1111 1111)

PORTB=0b00000000;    // Στην πόρτα Β δίνεται η αρχική τιμή 0000 0000


int i=0;               // ακέραιη μεταβλητή που χρησιμοποιούμε μέσα στην for

int a;
while(TRUE) {              //Βρόχος που δεν τελειώνει ποτέ(συνθήκη πάντα αληθής)
            a=0;
            for (i=0; i<=7; i++){
                              a =a + bit_test(PORTD,i); //Η συνάρτηση
                                //bit_test(PORTD,i) έλέγχει το bit i
                                // του καταχωρητή δεδομένων της πόρτας D.
                                // Αν το bit είναι 1 τότε η συνάρτηση παίρνει την
                                // τιμή 1. Αν το bit είναι 0, τότε η συνάρτηση
                                // παίρνει την τιμή 0.
                             } // κλείσιμο της αγκύλης της for
            PORTB=a; // Απόδοση στην πόρτα Β του αποτελέσματος της μέτρησης των 1
                     // που περιέχονται στον καταχωρητή δεδομένων της πόρτας D
            }   //κλείσιμο της αγκύλης του while

}      // κλείσιμο της αγκύλης του main
```

Εντολή: bit_test()

bit_test( )

Syntax:	value = bit_test (***var***, ***bit***)
Parameters:	***var*** may be a 8,16 or 32 bit variable (any lvalue) ***bit*** is a number 0- 31 representing a bit number, 0 is the least significant bit.
Returns:	0 or 1
Function:	Tests the specified bit (0-7,0-15 or 0-31) in the given variable. The least significant bit is 0. This function is much more efficient than, but otherwise similar to: ((var & (1<<bit)) != 0)
Availability:	All devices
Requires:	Nothing

Examples:	`if( bit_test(x,3) || !bit_test (x,1) ){`
	`  //either bit 3 is 1 or bit 1 is 0`
	`}`
	
	`if(data!=0)`
	`  for(i=31;!bit_test(data, i);i--) ;`
	`// i now has the most significant bit in data`
	`// that is set to a 1`

Example Files:	ex_patg.c
Also See:	bit_clear(), bit_set()