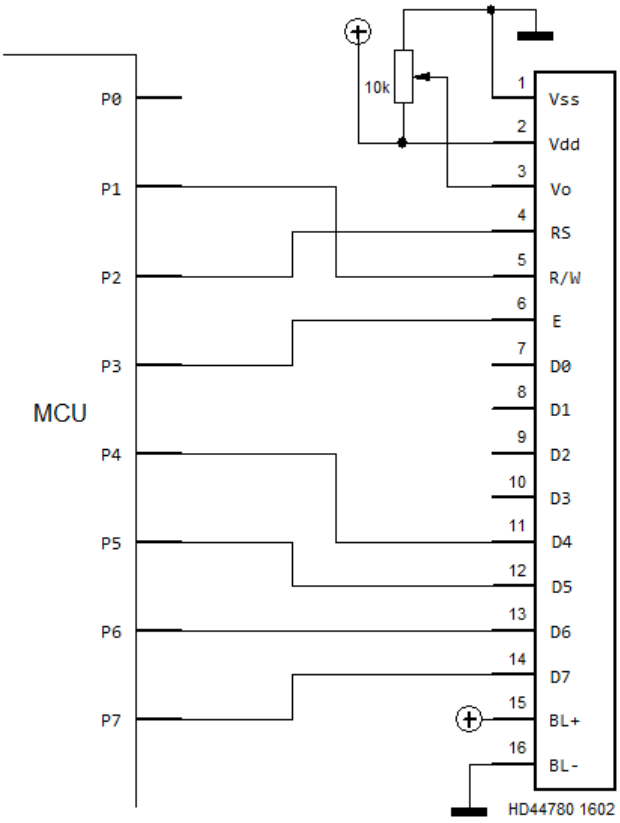
11.Οθόνη LCD 2x16 (4x20)



Οι οθόνες LCD 2x16 ή 4x20 είναι πολύ εύκολες στη σύνδεση και στη χρήση τους σε μικροελεγκτές. Χρειάζονται ελάχιστες γραμμές κώδικα για την προετοιμασία τους (initialize) και για τις εντολές λειτουργίας τους. Στο θεωρητικό κύκλωμα του σχήματος παρουσιάζεται η σύνδεση μιας οθόνης στο port ενός μικροελεγκτή. Διαθέτουν 8 bit data bus αλλά είναι δυνατό να προγραμματιστούν για λειτουργία με 4 bit, ώστε μαζί με τα 3 σήματα ελέγχου να μπορούν να συνδεθούν σε ένα port. Όταν είναι προγραμματισμένες για λειτουργία 4 bit στέλνεται πρώτα το ανώτερο μέρος του byte (upper) και στη συνέχεια το κατώτερο (lower). Έχουν έναν καταχωρητή 80 byte (DD RAM) για αποθήκευση των δεδομένων που εμφανίζονται, ο οποίος είναι εγγραφής – ανάγνωσης. Στις οθόνες με 2 σειρές, τα 40 πρώτα byte αφορούν την πρώτη σειρά με διευθύνσεις στην DD RAM **0x00** έως **0x27** και τα επόμενα 40 τη δεύτερη με διευθύνσεις **0x40** έως **0x67**. Στις οθόνες με 4 σειρές, τα 20 πρώτα αφορούν την πρώτη σειρά με διευθύνσεις στην DD RAM **0x00** έως **0x13**, τα επόμενα 20 (21 – 40) την τρίτη σειρά με διευθύνσεις **0x14** έως **0x27**, τα επόμενα (41 – 60) τη δεύτερη σειρά με διευθύνσεις **0x40** έως **0x53** και τα τελευταία 20 (61 – 80) την τέταρτη με διευθύνσεις **0x54** έως **0x67**. Αν κάθε σειρά

έχει 16 θέσεις για εμφάνιση χαρακτήρων, εμφανίζονται τα δεδομένα που είναι αποθηκευμένα στις 16 πρώτες θέσεις του καταχωρητή, ενώ τα υπόλοιπα μπορούν να εμφανιστούν με ολίσθηση χρησιμοποιώντας την αντίστοιχη εντολή της οθόνης. Υπάρχει δυνατότητα να εμφανίζεται ή όχι ο κέρσορας ή να αναβοσβήνει, επίσης υπάρχει δυνατότητα να ολισθαίνει αυτόματα ο κέρσορας ή η οθόνη καθώς στέλνονται χαρακτήρες. Οι οθόνες διαθέτουν, ακόμα, οπίσθιο φωτισμό (back light) στα pin BL+, BL- και ρύθμιση αντίθεσης στο pin Vo.

Τα δεδομένα στέλνονται στην οθόνη σε κωδικοποίηση χαρακτήρων ASCII και εμφανίζονται όλοι οι λατινικοί χαρακτήρες και τα σύμβολα που υπάρχουν στις 128 πρώτες θέσεις. Για τις επόμενες 128 θέσεις του ASCII η ROM της οθόνης περιέχει πληροφορίες για εμφάνιση ειδικών χαρακτήρων και ανάλογα με την έκδοση της ROM μπορούν να εμφανιστούν Ελληνικοί, Κυριλλικοί, Κινέζικοι ή Γιαπωνέζικοι χαρακτήρες και κάποια μαθηματικά σύμβολα[1]. Επίσης υπάρχει η δυνατότητα να δημιουργηθούν pixel – pixel 8 χαρακτήρες, οι οποίοι αποθηκεύονται σε έναν καταχωρητή (CG RAM) και μπορούν να εμφανιστούν, όμως διαγράφονται με απώλεια τροφοδοσίας.

Η οθόνη δέχεται εντολές με **0** στο pin RS. Οι εντολές είναι για τη θέση του κέρσορα, τις ιδιότητες του κέρσορα, εντολές ολίσθησης, καθαρισμού της οθόνης, δημιουργίας χαρακτήρων και ανάγνωσης και έχει διαφορετικό χρόνο εκτέλεσης η κάθε μια. Η ανάγνωση χαρακτήρων γίνεται με **1** στο pin R/W. Οι πληροφορίες λαμβάνονται από την οθόνη με **1** στο pin E είτε αυτές είναι δεδομένα, είτε εντολές. Για την initialization της οθόνης πρέπει να εκτελεστεί αποστολή 3 φορές η πληροφορία **0b0011** στο **D7:D4** της οθόνης με διαφορετικό ενδιάμεσο χρόνο και στη συνέχεια να σταλεί η εντολή για λειτουργία 4 bit (μετά την εκκίνησή της η οθόνη είναι σε λειτουργία 8 bit, επειδή όμως δεν έχουμε συνδεδεμένα τα 4 LSB τα pin βλέπουν **0**). Στη συνέχεια παρουσιάζεται ένα παράδειγμα κώδικα σε γλώσσα C για initialization οθόνης:

```
#define lcdport PORTA                   // Define LCD PORT
#define lcdddr DDRA                     // Define LCD DDR
#define En 0b00001000                   // E signal

void InitLCD (void)                     // Start up, initialization rutine
{
    lcdddr |= ~(1<<0);                  // PORTx1:7 outputs (LCD)
    lcdport &= (1<<0);
    _delay_ms (20);                     // Power on delay 20ms
    out = 0b00110000;
    lcdport = (lcdport | out);
    _delay_us (100);
    lcdport = (lcdport | En);           // Init
```

[1] *Η πιο συνηθισμένες ROM είναι οι PN, PS, PM με English – Japanese Font.*

```
    _delay_us (100);
    lcdport = (lcdport & ~En);
    _delay_ms (5);

    lcdport = (lcdport | En);            // Init
    _delay_us (100);
    lcdport = (lcdport & ~En);
    _delay_us (200);

    lcdport = (lcdport | En);            // Init
    _delay_us (100);
    lcdport = (lcdport & ~En);
    _delay_ms (5);

    lcdport &= (1<<0);
    out = 0b00100000;                    // 4 bit interface
    lcdport = (lcdport | out);
    _delay_us (100);
    lcdport = (lcdport | En);
    _delay_us (100);
    lcdport = (lcdport & ~En);
    _delay_us (50);
}
```

Για την αποστολή δεδομένων ή εντολών στην οθόνη μπορούμε να γράψουμε ρουτίνες ώστε να τις καλούμε στο κυρίως πρόγραμμα. Στη συνέχεια παρουσιάζεται ένα παράδειγμα ρουτίνας αποστολής εντολών και η κλήση της ρουτίνας για αποστολή εντολής:

```
void WrIn (uint8_t tmp)              // Write Instruction routine
{
    lcdport &= (1<<0);
    out = tmp & 0b11110000;          // Upper
    lcdport = (lcdport | out);
    _delay_us (100);
    lcdport = (lcdport | En);
    _delay_us (100);
    lcdport = (lcdport & ~En);
    _delay_us (100);
    lcdport &= (1<<0);
    out = tmp << 4;                  // Lower
    lcdport = (lcdport | out);
    _delay_us (100);
    lcdport = (lcdport | En);
    _delay_us (100);
    lcdport = (lcdport & ~En);
    _delay_us (50);
}

#define CDRC 0b00000001              // Clear Display and Reset Cursor (2ms delay)

WrIn (0b00101100);                   // 001, 4 bit, 2 lines 2x16, 5x11 dots, xx
WrIn (0b00001111);                   // 00001, Display on, cursor on, blinking on
WrIn (0b00000110);                   // 000001, Cursor increase, display not shift
WrIn (CDRC);                         // Clear display, reset cursor (2ms delay)
_delay_ms (2);
```

Στη συνέχεια παρουσιάζεται ένα παράδειγμα ρουτίνας αποστολής δεδομένων και η κλήση της ρουτίνας για αποστολή του γράμματος C:

```
#define Da 0b00000100              // RS, data signal (0 for instruction)

void WrDa (uint8_t tmp)            // Write Data routine
{
    lcdport &= (1<<0);
    out = tmp & 0b11110000;                // Upper
    lcdport = (lcdport | out | Da);
    _delay_us (100);
    lcdport = (lcdport | En);
    _delay_us (100);
    lcdport = (lcdport & ~En);
    _delay_us (100);
    lcdport &= (1<<0);
    out = tmp << 4;                        // Lower
    lcdport = (lcdport | out | Da);
    _delay_us (100);
    lcdport = (lcdport | En);
    _delay_us (100);
    lcdport = (lcdport & ~En);
    _delay_us (50);
}

WrDa ('C');
```

Με παρόμοιο τρόπο μπορούμε να γράψουμε ρουτίνα εντολής μετακίνησης του κέρσορα ώστε να αποθηκευτούν τα δεδομένα σε συγκεκριμένη θέση. Στη συνέχεια παρουσιάζεται ένα παράδειγμα ρουτίνας και η κλήση για μετακίνηση στη δεύτερη σειρά πρώτη θέση:

```
void SeCu (uint8_t addr)                   // Send cursor to a specific address
{
    WrIn (0b10000000 | addr);
}

SeCu (0x40);
```