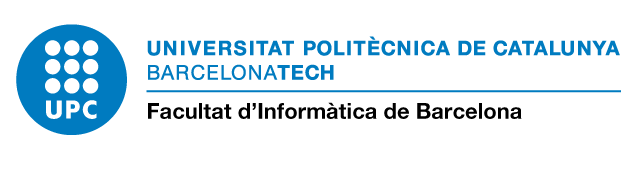
****

Enunciat de la pràctica de laboratori

**Lab 4:**

**Display gràfic de cristall líquid GLCD**

**L4. GLCD**

**L4 (A) Funcions bàsiques per presentar informació**

**L4 (B) Cronómetre amb timer**

**1- Objectius**

Els exercicis sobre la GLCD (Graphic Liquid Crystal Display) estan dividits en dues parts pràctiques. Cadascuna de les parts es farà en una sessió de laboratori diferent.

La primera part anomenada **L4(A),** és una introducció al funcionament del display gràfic i consta d’unes subrutines bàsiques per a programar la pantalla gràfica.

A la segona part de la pràctica anomenada **L4(B)**, farem servir els timers del microcontrolador per programar un rellotge que es presentarà a la pantalla gràfica.

**2- Funcionament de la pantalla GLCD**

A la figura 1 podeu veure l’esquema de Proteus de la pantalla GLCD connectada al microcontrolador.

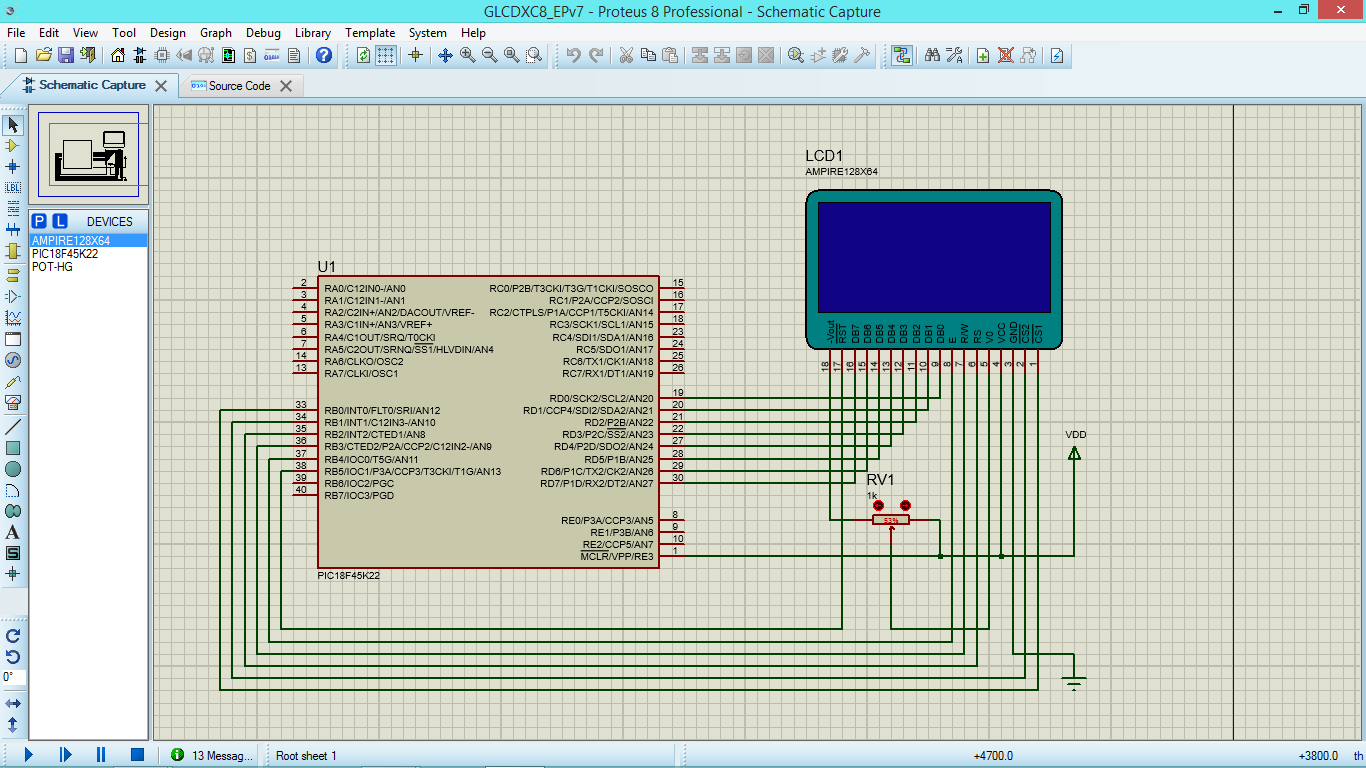


Figura 1. Connexió de la GLCD al PIC

Fixeu-vos que s’ha connectat la pantalla als ports B i D del microcontrolador. El port B té una sèrie de senyals de control de la pantalla (Enable, Reset, Read/Write...) i el port D serà un bus de 8 bits per on s’envien dades/informació a la pantalla (posició a pintar, caràcters a pintar, ordres d’esborrar, moure cursor, etc.).

Com a documents annexos a aquesta pràctica, trobareu els següents:

- Projecte base de Proteus amb la pantalla GLCD (L4\_GLCD\_Intro.pdsprj).

- Datasheet de la GLCD

A la següent figura, es mostra l’aspecte del projecte base L4\_GLCD\_Intro.pdsprj carregat a Proteus:

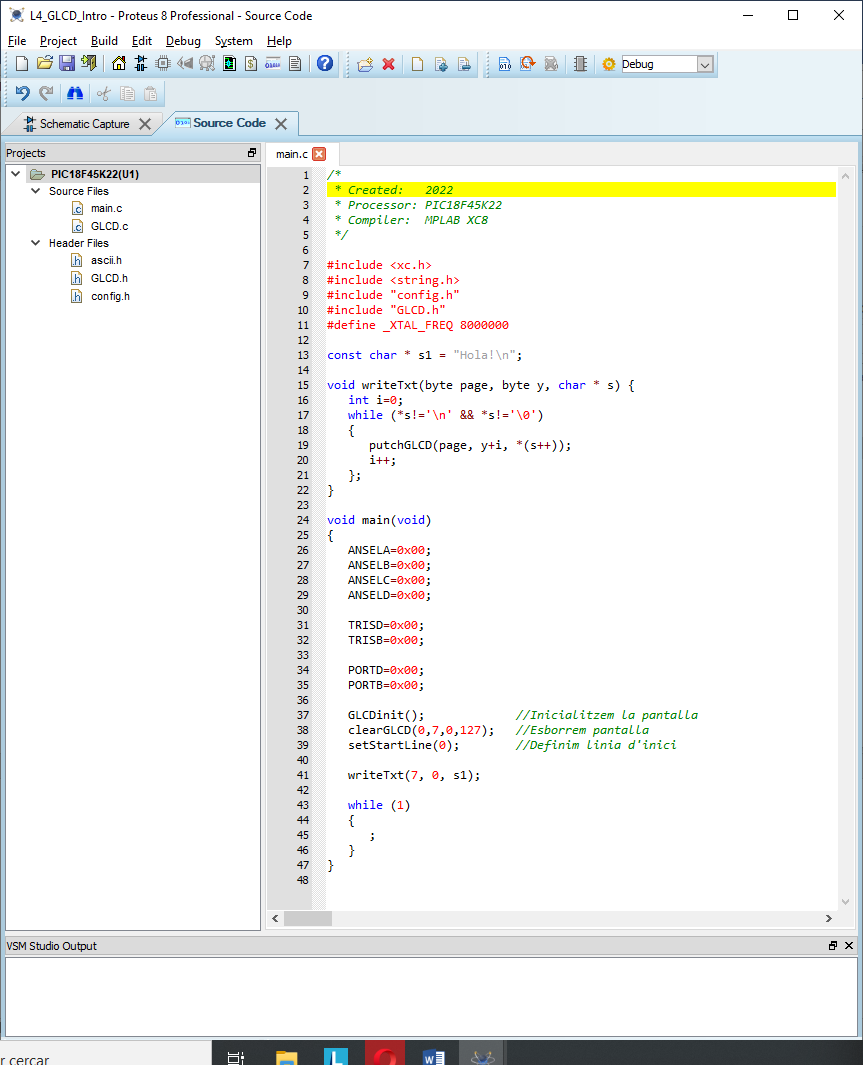


Figura 2. Captura del projecte a Proteus.

Com podeu observar, en el projecte hi han diferents fitxers .c i .h per poder usar llibreries, tenir fitxers de configuració, de dades, etc.

Concretament teniu el fitxer main.c amb el codi principal que configura i utilitza la pantalla, el fitxer GLCD.c amb les funcions de baix nivell de la pantalla, el fitxer GLCD.h amb les capçaleres de les funcions i el fitxer ascii.h on es defineixen els patrons per poder pintar caràcters a la pantalla.

Si executeu el programa, heu de veure la pantalla funcionant amb el següent text:

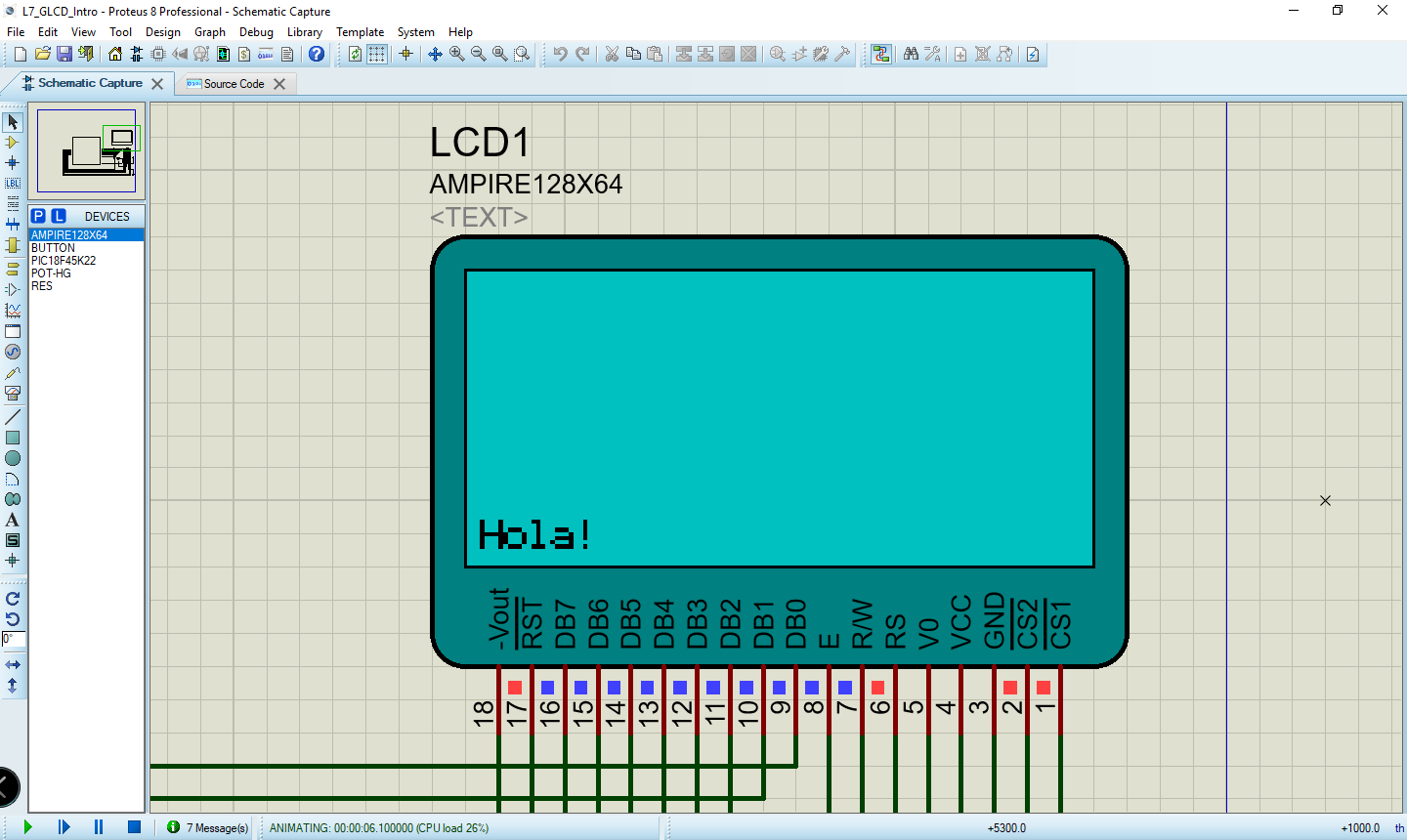


Figura 3. Demo de funcionament de l’exemple.

A partir d’aquest programa d’exemple, s’espera que desenvolupeu el vostre treball que s’especifica a les pàgines següents.

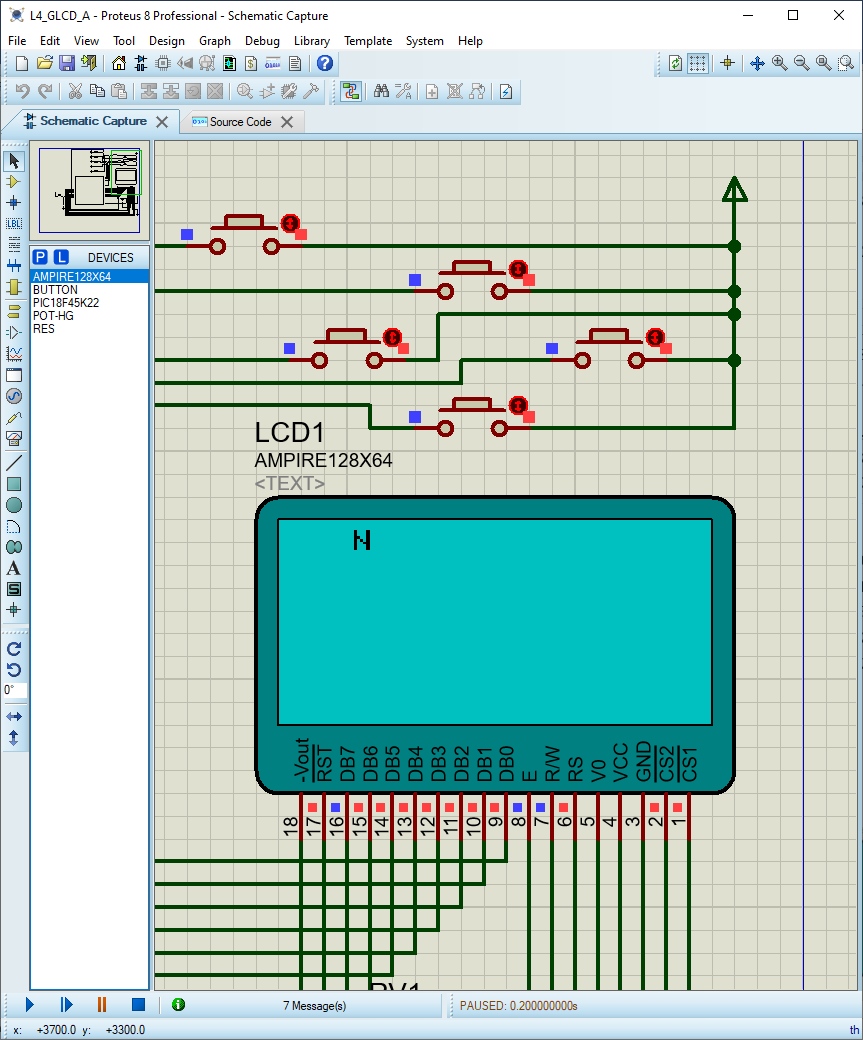
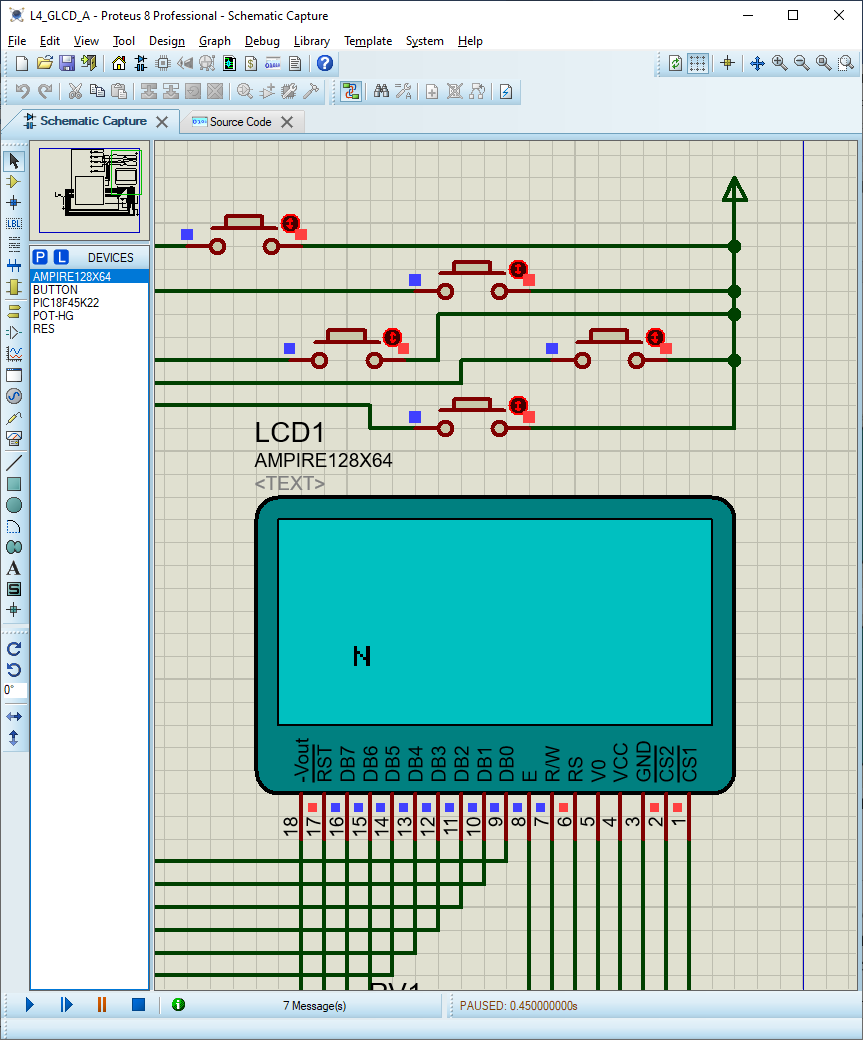
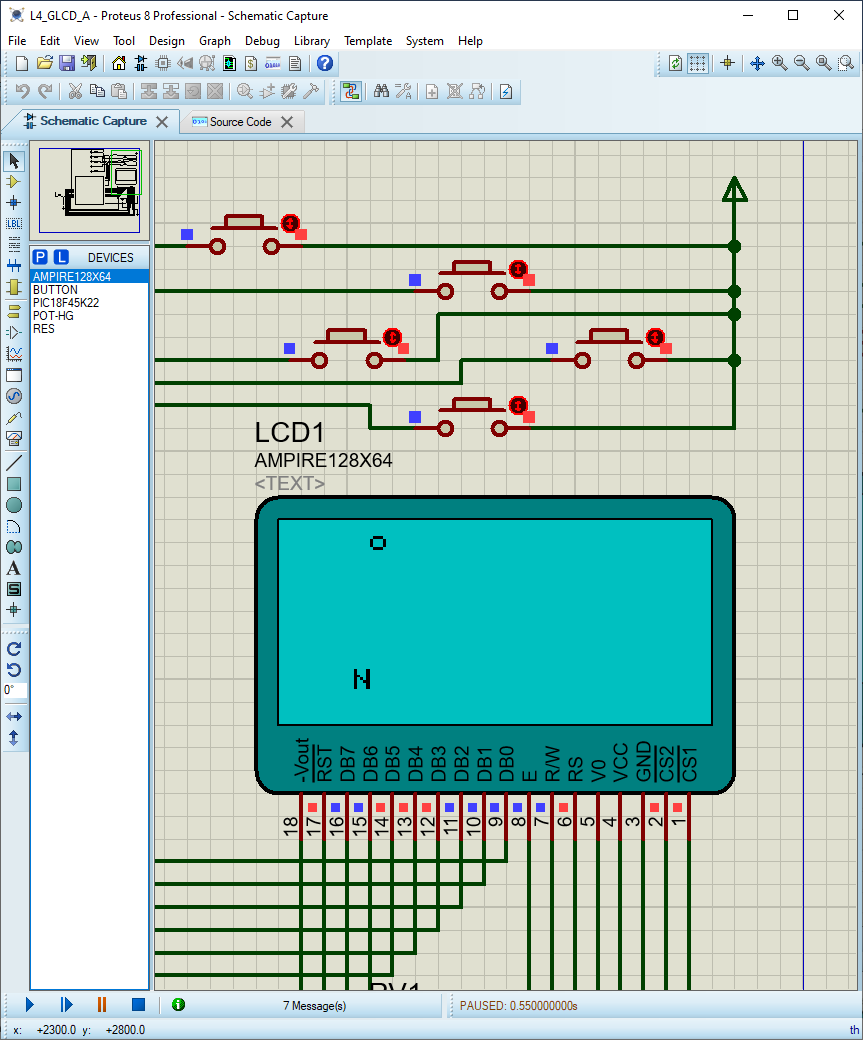
**3- Treball previ L4(A)**

**Tasques a realitzar**

El programa a realitzar consisteix en presentar els noms dels integrants del grup i posteriorment pintar un punt per pantalla que es vagi movent segons es premen uns botons. Les tasques a realitzar són les següents:

1. Estudiar els diferents fitxers entregats amb la pràctica. Manuals i **sobretot el codi de la llibreria per fer servir la pantalla**.
2. Modificar el projecte, programa i esquema base, perquè realitzi les següents funcions:
   1. Fer una pantalla de presentació que mostri el noms i cognoms dels integrants del grup de forma animada. Cada lletra anirà baixant des de la part de dalt de la pantalla fins a la seva posició final, com si fessin un efecte cascada. Podeu veure captures intermitges i el resultat final a la següent pàgina. Un cop mostrat els noms de forma animada, s’esperarà 1 segon i es netejarà la pantalla.
   2. A continuació, s’han d’afegir **4 botons** a l’esquemàtic, en 4 pins que considereu **adequats**, i configurar-los adequadament com a pins d’entrada. Aquest botons permetran moure’s per la pantalla amunt, a baix, a l’esquerra o a la dreta.
   3. Dibuixar un punt al mig de la pantalla. Aquest punt s’anirà movent en funció de l’estat dels 4 botons configurats anteriorment. El punt es mourà a dreta o esquerra sempre i quan hi hagi un botó activat, és a dir no ha d’haver-hi detecció per flancs. En canvi el punt es mourà cap amunt quan es detecti un flanc de pujada. Finalment, el punt es mourà cap a baix quan es detecti un flanc de baixada.
   4. Afegir un text indicant la posició del punt a la part superior dreta de la pantalla.

Teniu algunes captures de pantalla a la següent figura.

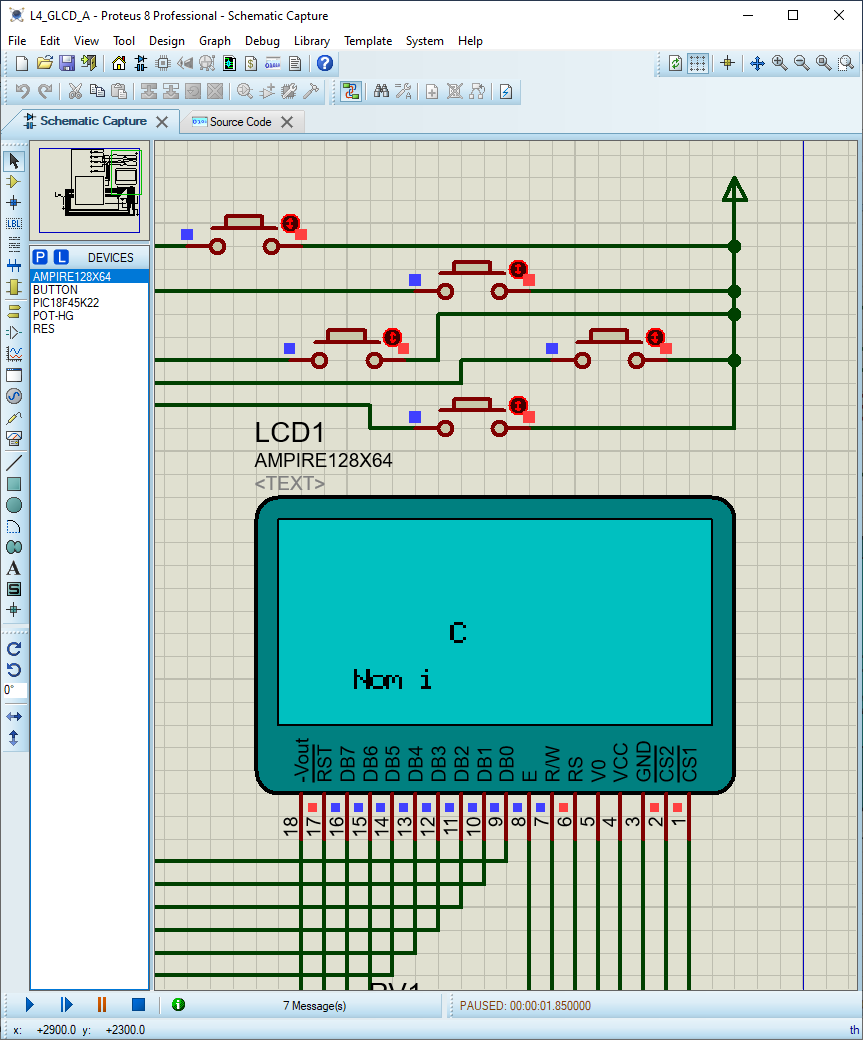
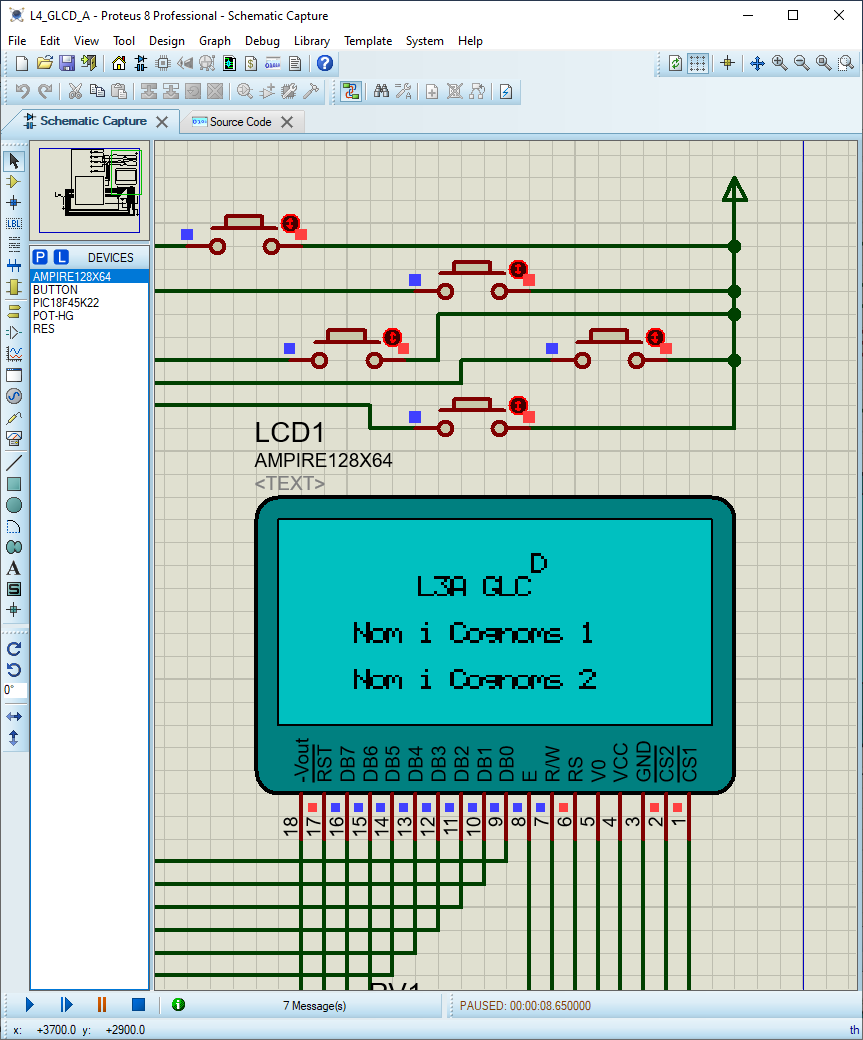
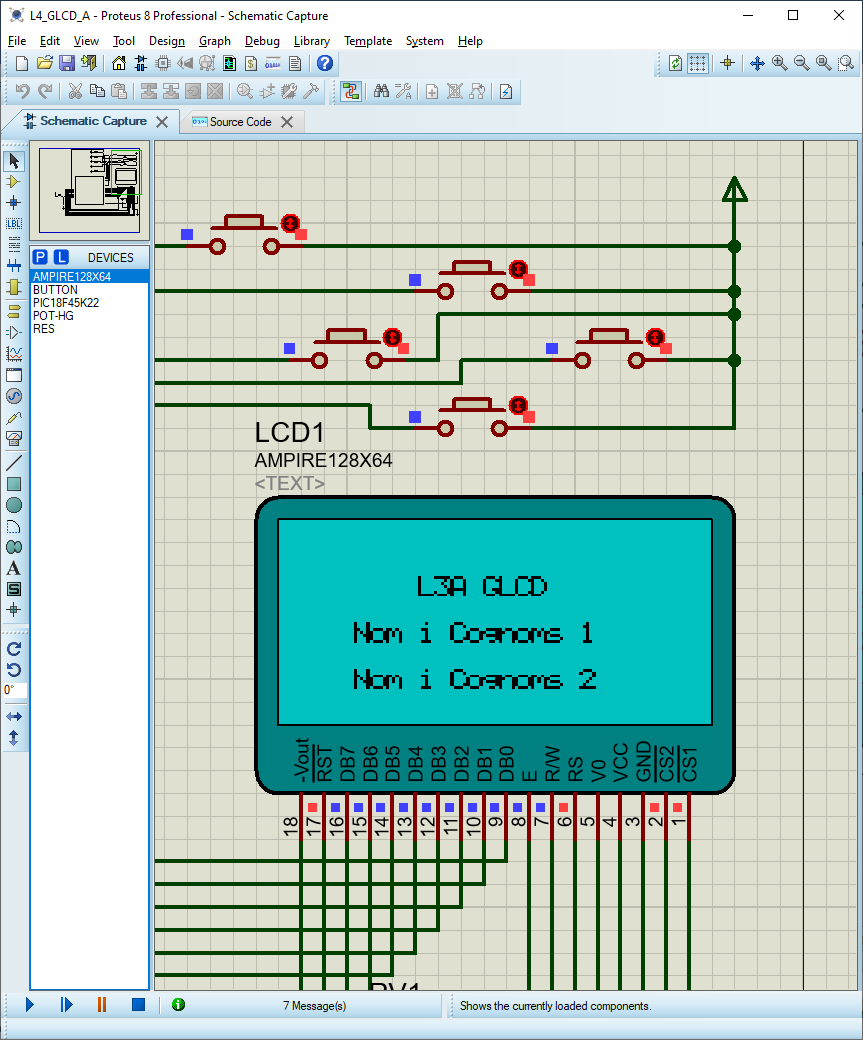
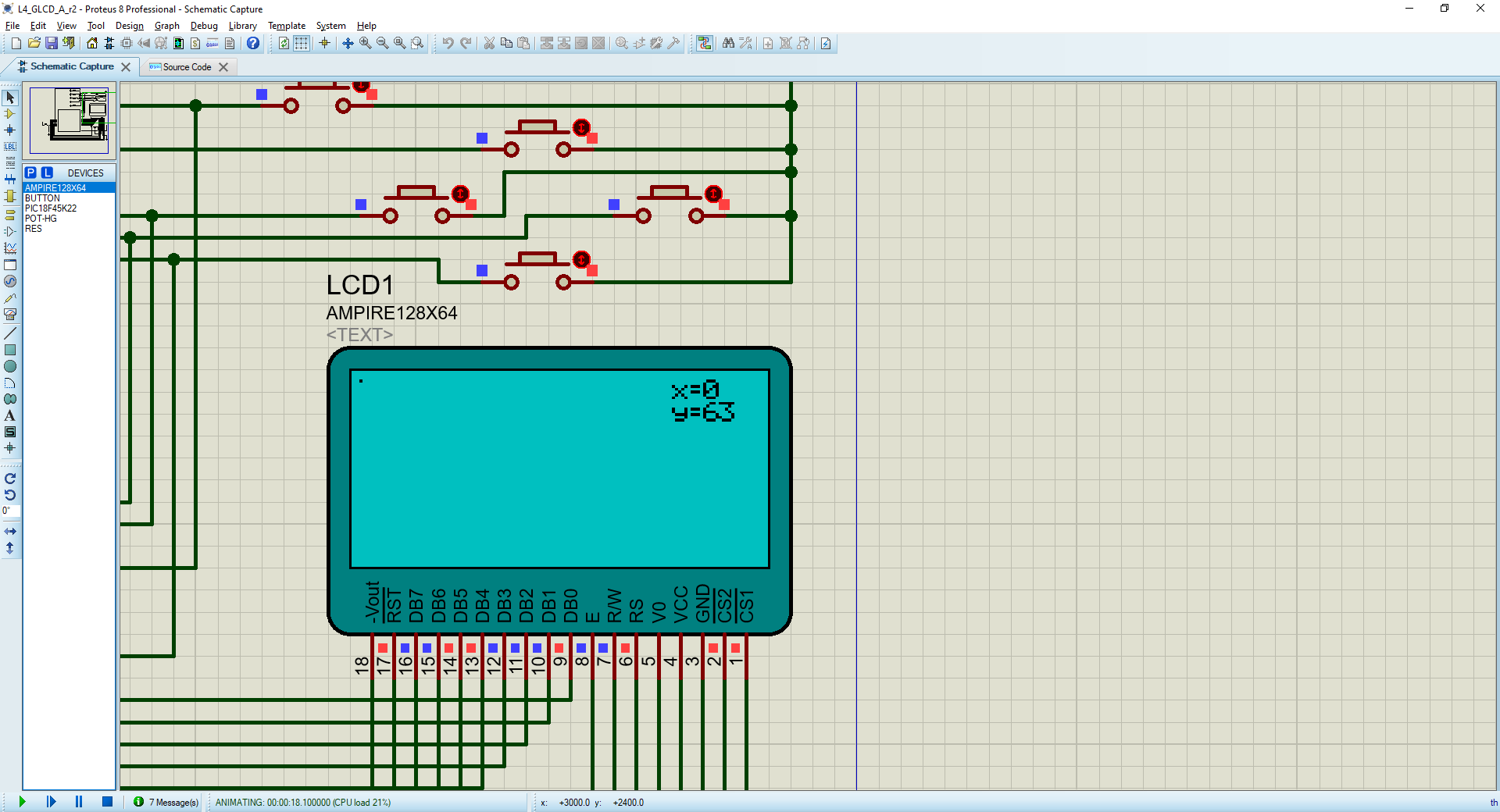
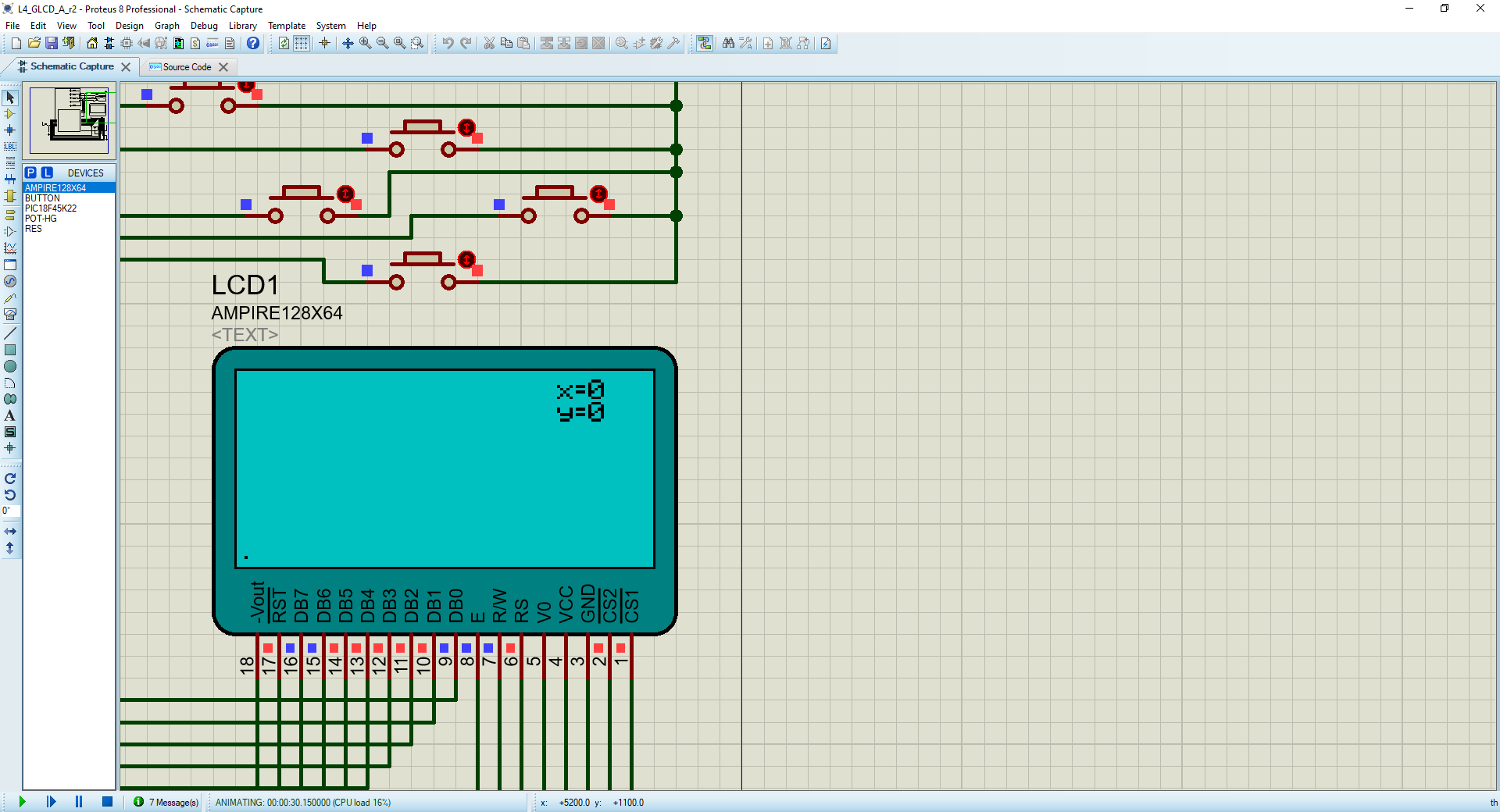
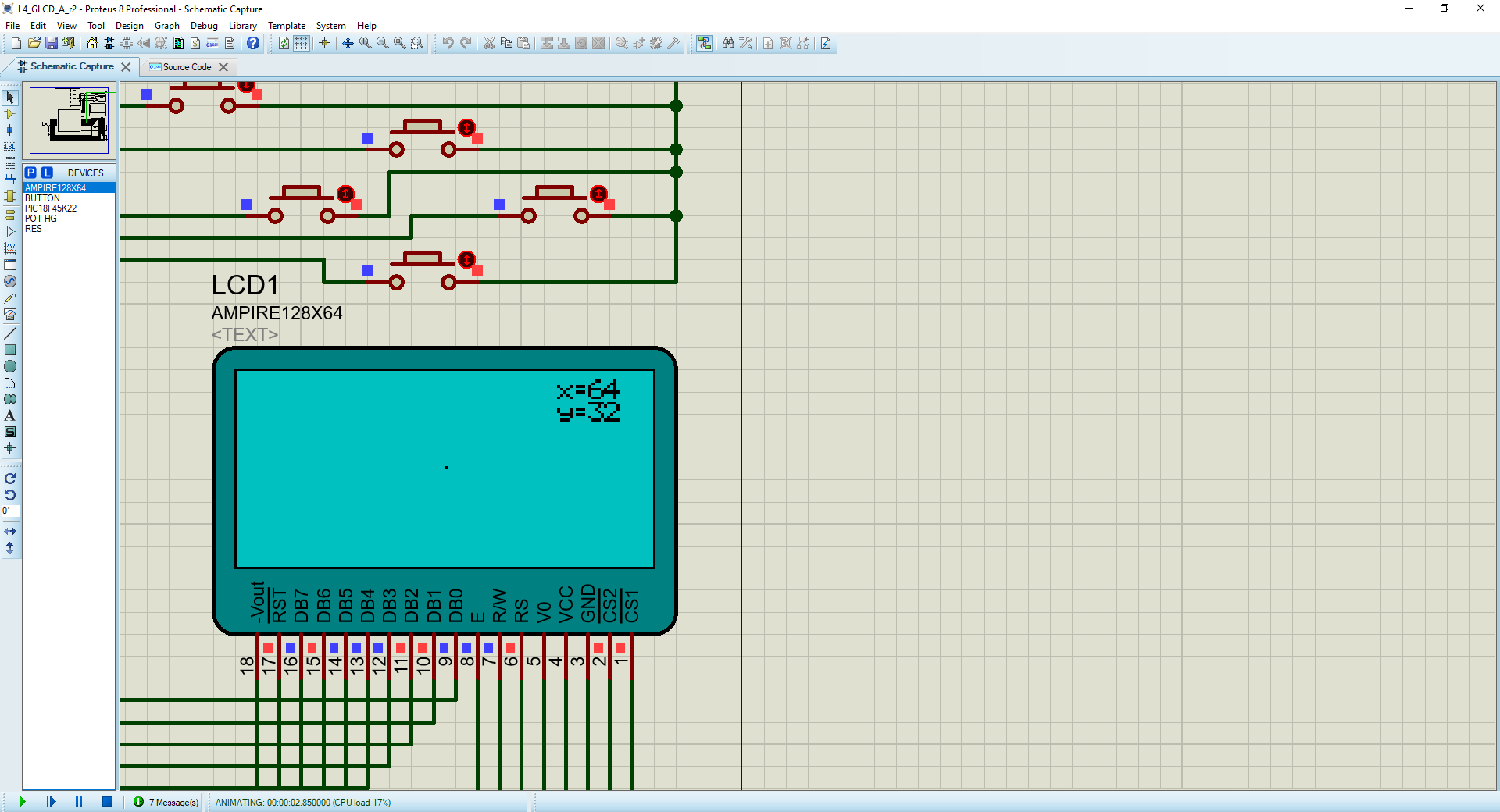
  

Figura 4. Captures de pantalla de l’animació de noms.



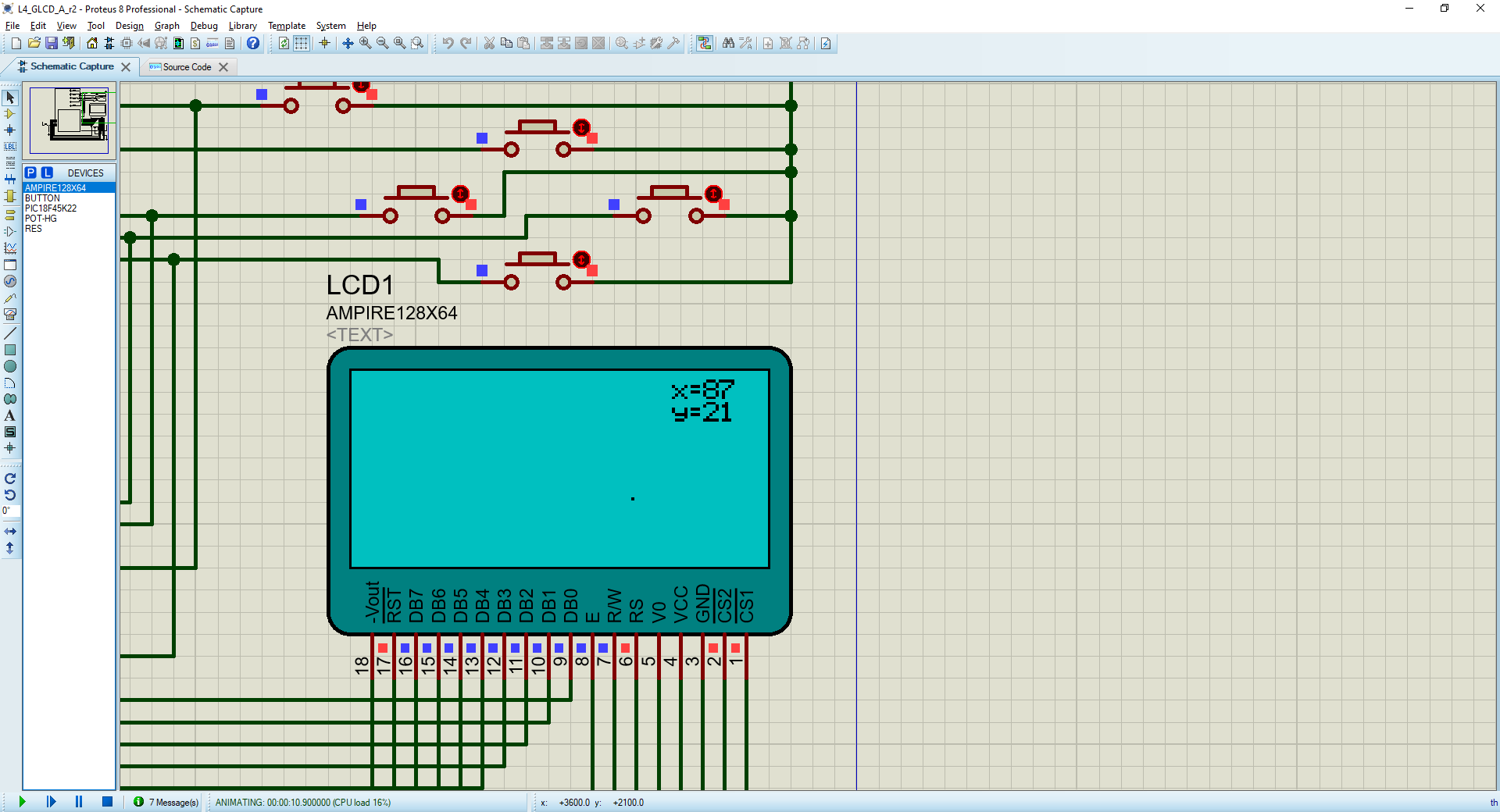
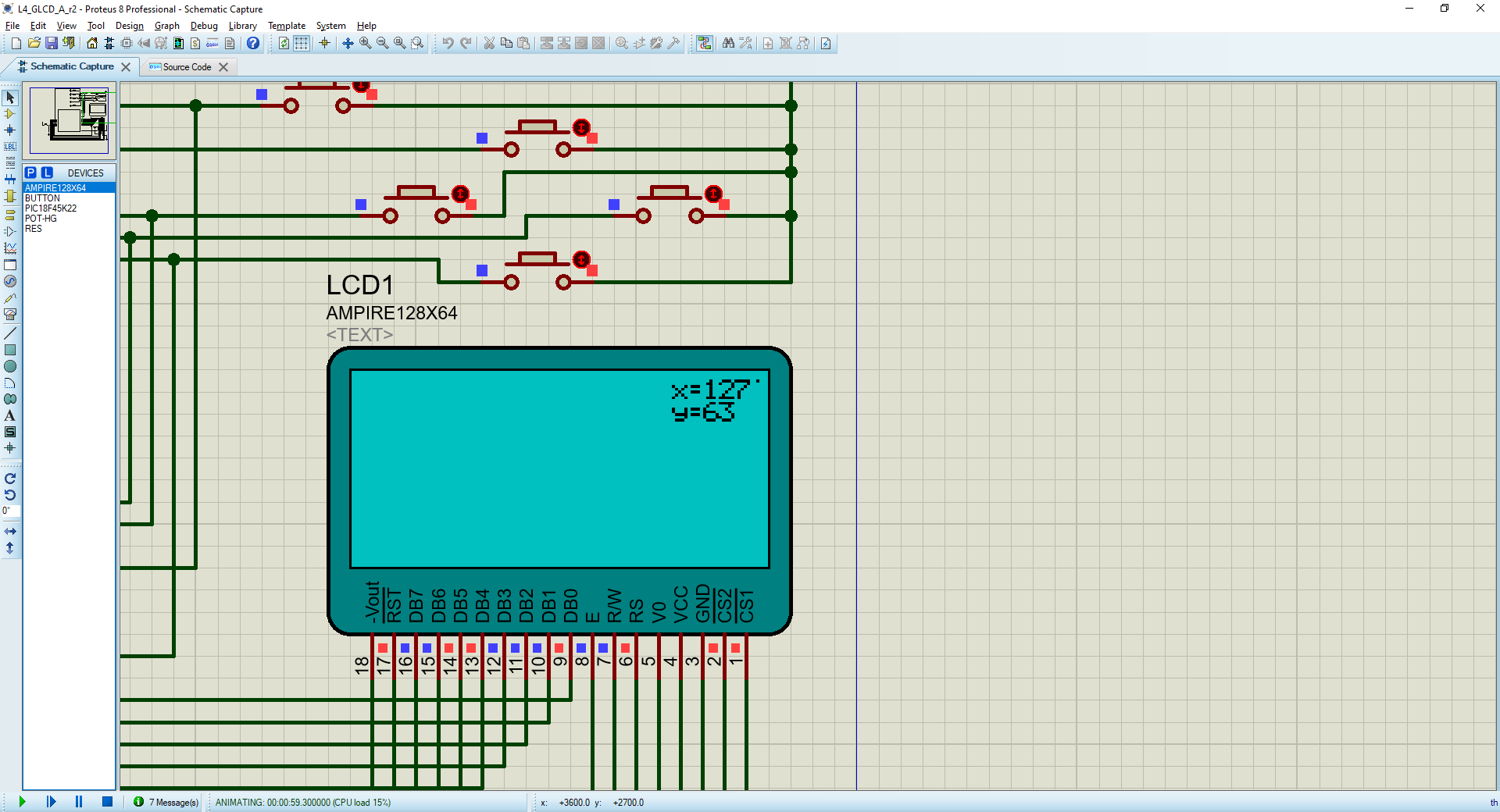
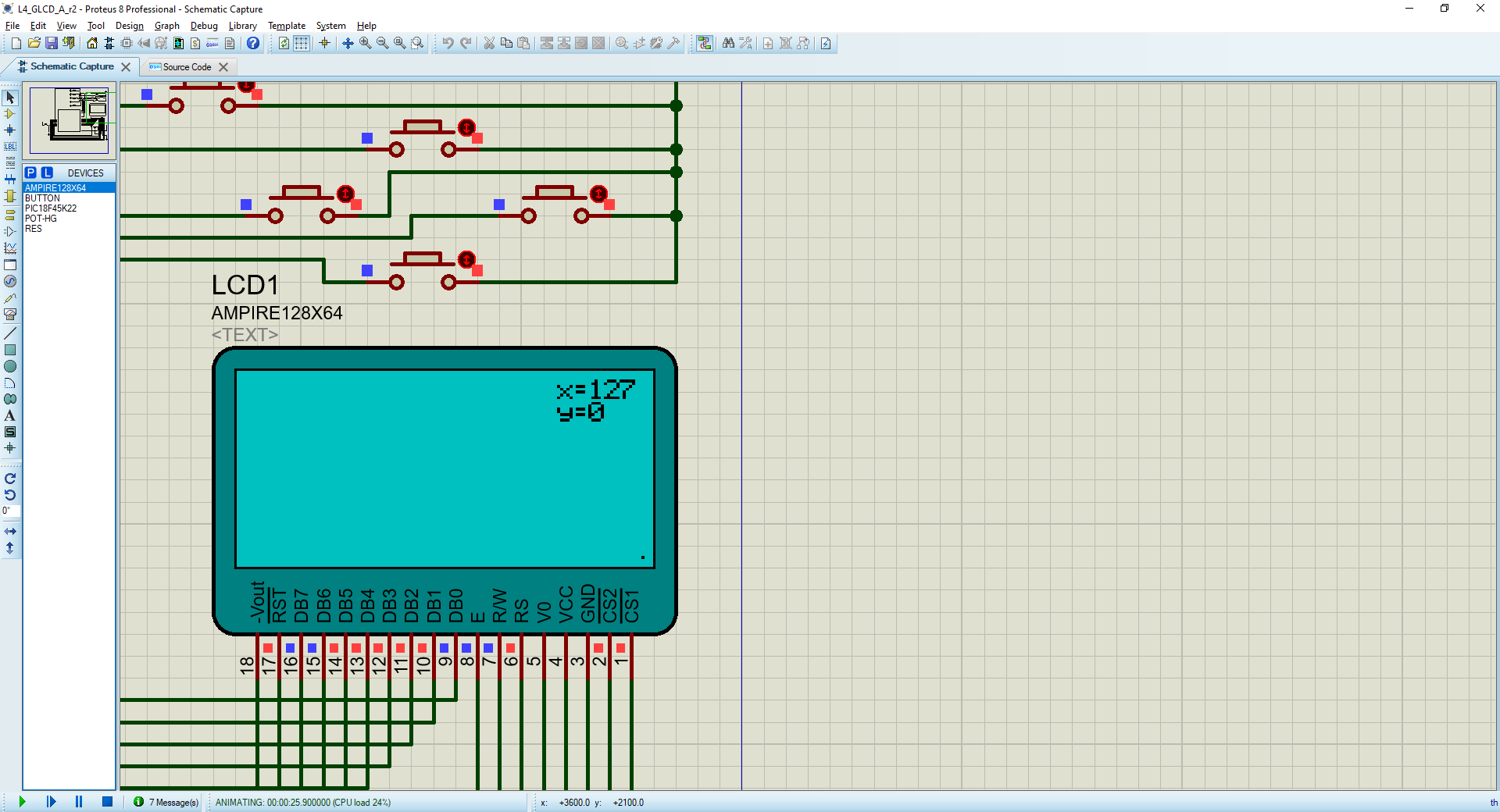


Figura 5. Captures de pantalla del moviment d’un punt per pantalla mitjançant botons.

L’entrega del projecte modificat amb els fitxers de codi adequats la fareu pel Racó i al principi de la sessió mostrareu el resultat al professor.

Es valorarà l’estructura del codi, l’aprofitament dels recursos del microcontrolador, el bon ús de variables i la utilització de funcions auxiliars reutilitzables.

**4- Rúbrica treball previ L4 (A)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Iniciat  (0-2.5 punts) | En desenvolupament  (2.5-5.0 punts) | Aconseguit  (5.0-7.5 punts) | Exemplar  (7.5-10 punts) |
| Animació noms (5 punts): | Funciona malament, no hi ha animació | Hi ha animació però el codi no és gens reutilitzable | El codi té només alguna part reutilitzable | Funciona perfectament i la funció d’animació té en compte espais, posicions inicials, posicions finals, etc. |
| Botons  (5 punts): | Hw mal dissenyat i funcionament erroni | Hw mal dissenyat o funcionament erroni (apretats, flancs) | Funcionament correcte però mal ús de recursos | Funciona perfectament, no hi ha punt fora dels límits de la pantalla, només es refresca la pantalla quan es necessita i s’aprofiten els recursos |

**4- Treball previ L4 (B)**

L’objectiu d’aquesta segona part de la pràctica és familiaritzar-se amb l’ús dels *Interval Timers*, i comprendre com podem gestionar els diferents esdeveniments generats al microcontrolador amb la utilització del tractament d’interrupcions.

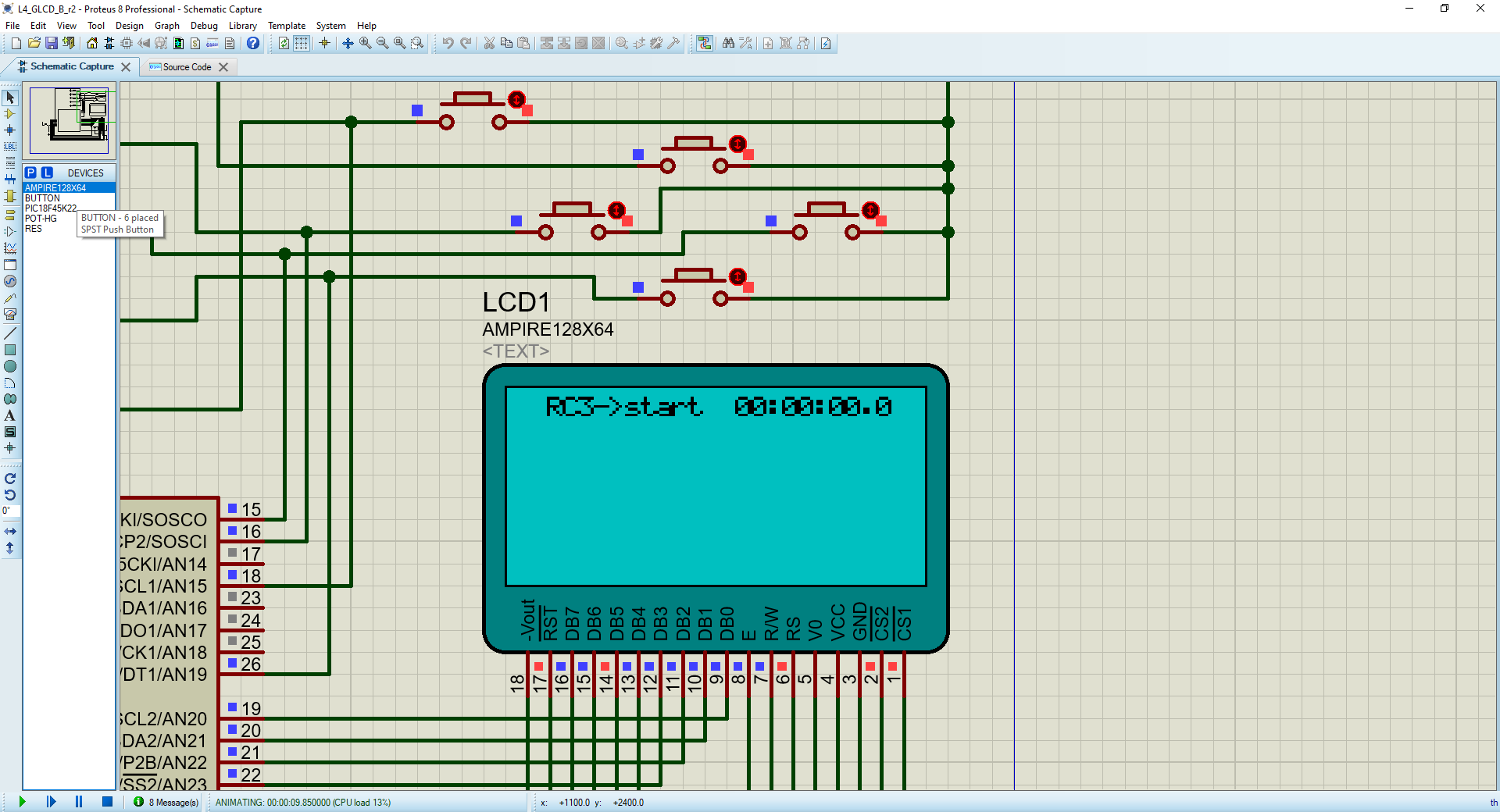
Habitualment, en la programació de microcontroladors, necessitem comptar el temps transcorregut per tal d’executar accions periòdiques als nostres programes. Els microcontroladors ens ofereixen uns perifèrics que s’encarreguen de comptar temps de manera autònoma: els ***Timers***. El seu registre intern s’incrementa a raó d’un clock d’entrada. El PIC ens permet obtenir aquest clock d’entrada efectuant diverses manipulacions sobre l’*Instruction Cycle Clock* del microcontrolador (FOSC/4).

Utilitzareu el Timer0 per tal de comptar el temps transcorregut, i quan hagi passat 1 dècima de segon, actualitzareu el valor del temporitzador que mostreu per la GLCD.

Per a saber quan ha transcorregut 1 dècima de segon, podríem consultar contínuament el valor del comptador de Timer0, però seria una estratègia bastant ineficient que consumiria massa % de CPU del PIC. Per evitar aquest problema, farem servir ***interrupcions***.

**Tasques a realitzar**

El programa a realitzar consisteix en una aplicació de tipus cronòmetre que mostri a la GLCD un temporitzador digital amb el format: hores:minuts:segons.dècimes (veure diagrama a sota). Utilitzarem un polsador per a posar en marxa, aturar i resetejar el temporitzador.





Pulsador RC3

Figura 6. Captura de pantalla a Proteus amb cronòmetre i botó RC3.

Tingueu present que la simulació temporal que fa Proteus té certes limitacions. És probable que el temps comptat pel temporitzador a la simulació presenti petites desviacions respecte a la realitat.

Les tasques a realitzar són les següents:

1. Tornar a presentar la pantalla inicial que mostri el noms i cognoms dels integrants del grup de forma animada. Igual que a la pràctica L4A, cada lletra anirà baixant des de la part de dalt de la pantalla fins a la seva posició final, com si fessin un efecte cascada.

2. Respondre el qüestionari del final de la pràctica per configurar el Timer0 perquè generi una interrupció cada 0.1 segons.

3. Programeu una subrutina tic() que s’executi cada dècima de segon mitjançant interrupcions periòdiques d’alta prioritat provinents del Timer0. **Comproveu que la configuració del timer és correcte mitjançant la inserció de breakpoints a la interrupció (aquesta serà la prova principal per avaluar la configuració).** Per veure el temps entre breakpoints podeu mirar a la part inferior de proteus on apareix la informació xxxx elapsed i podeu fer servir el botó F12 per avançar en la simulació fins al proper breakpoint.

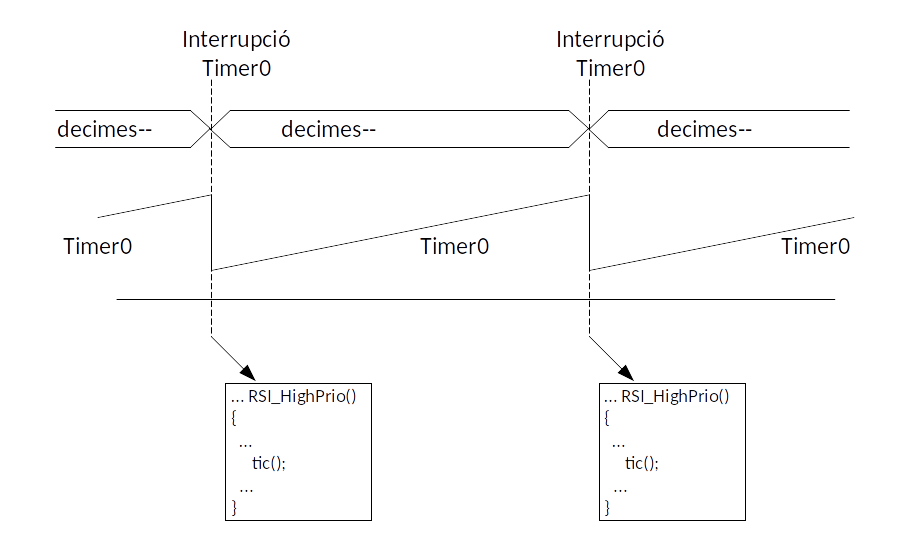
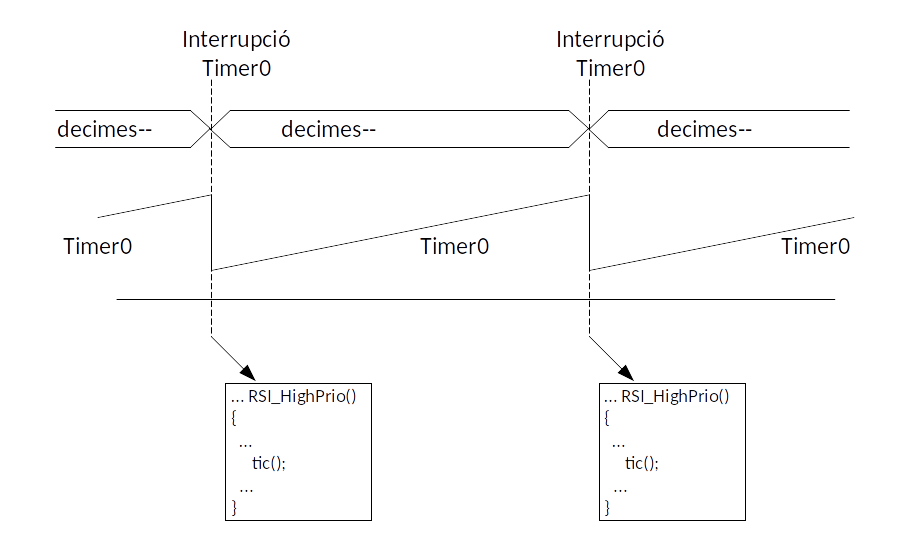


Figura 7. Esquema de funcionament del comptador del timer, la generació d’interrupcions i la crida de la funció tic() a la rutina d’atenció de la interrupció.

4. Programeu un procediment updateGLCD(·) que executant-se **dins del programa principal (main)** actualitzi per pantalla GLCD el valor del temporitzador digital tot consultant una o diverses variables globals que s’actualitzen en la subrutina tic(). El format del temporitzador serà (HH:MM:SS.D).

5. El temporitzador no sempre estarà en marxa. Podrà estar en tres estats: *Init*, *Running* o *Stopped*. El canvi entre estats el fareu amb el **flanc ascendent de la pulsació del botó RC3**. Gestionareu les activacions del polsador RC3 mitjançant la tècnica d’enquesta. Com que treballeu llegint flancs, haureu d’implementar la corresponent estratègia anti-rebots.

La imatge següent mostra els canvis d’estat per polsació del botó d’entrada RC3.

INIT

RUNNING

STOPPED

Botó

Botó

Botó

RC3->start

00:00:00.0

RC3->stop

xx:xx:xx.x

RC3->reset

xx:xx:xx.x

Figura 8. Diagrama de blocs del cronòmetre.

A l’iniciar el programa, el sistema estarà en estat INIT i apareixerà la següent pantalla amb el text “RC3->start” i el temporitzador inicialitzat a cero:

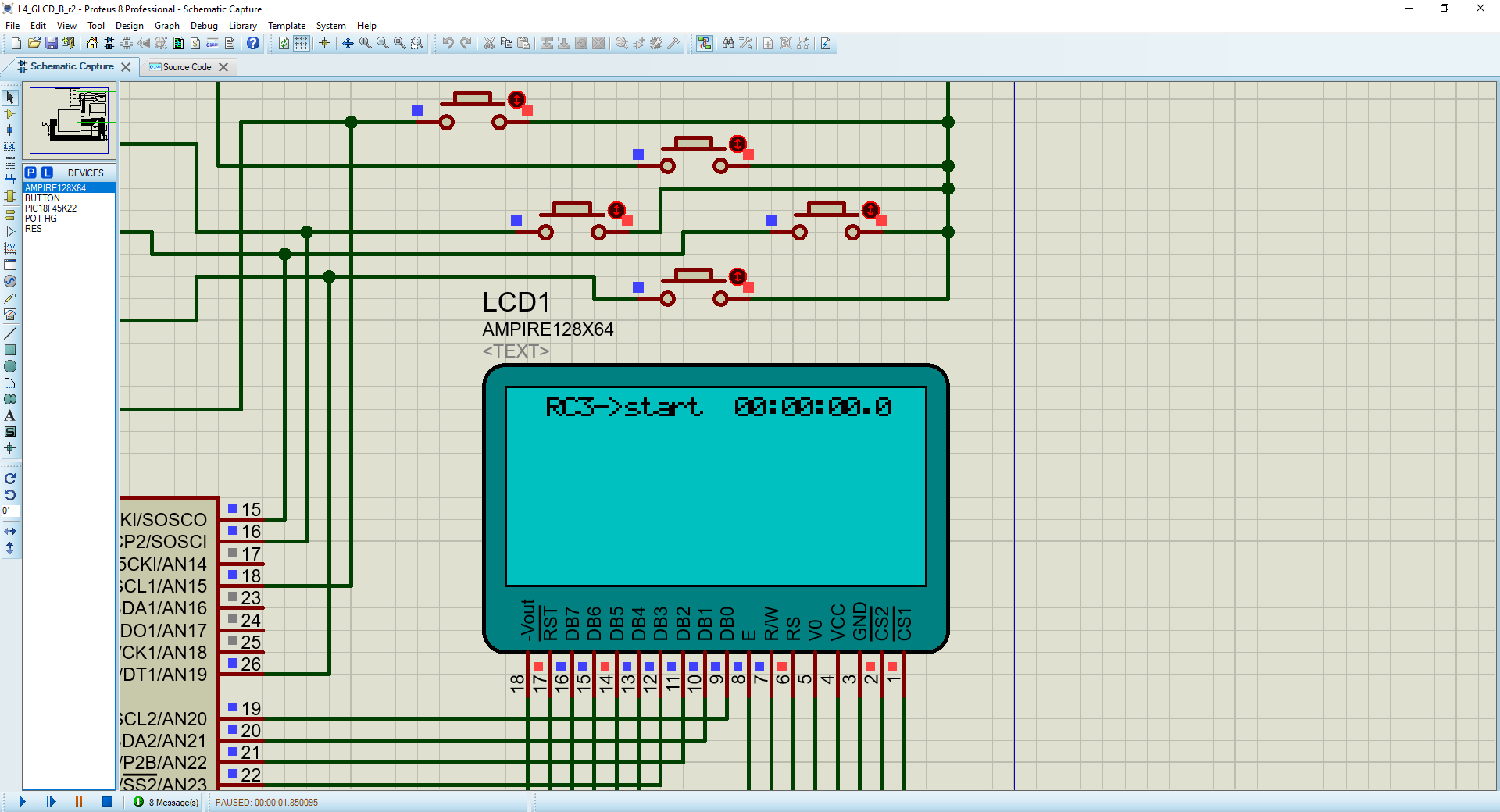


Figura 9. Estat inicial

Una pulsació del botó RC3 farà que passi a RUNNING i comenci a comptar. A la pantalla GLCD mostrareu el text “RC3->stop” indicant que la propera vegada que s’apreti el botó RC3 el cronòmetre passarà a estat stopped. També s’haurà de mostrar el valor actualitzat del temporitzador:

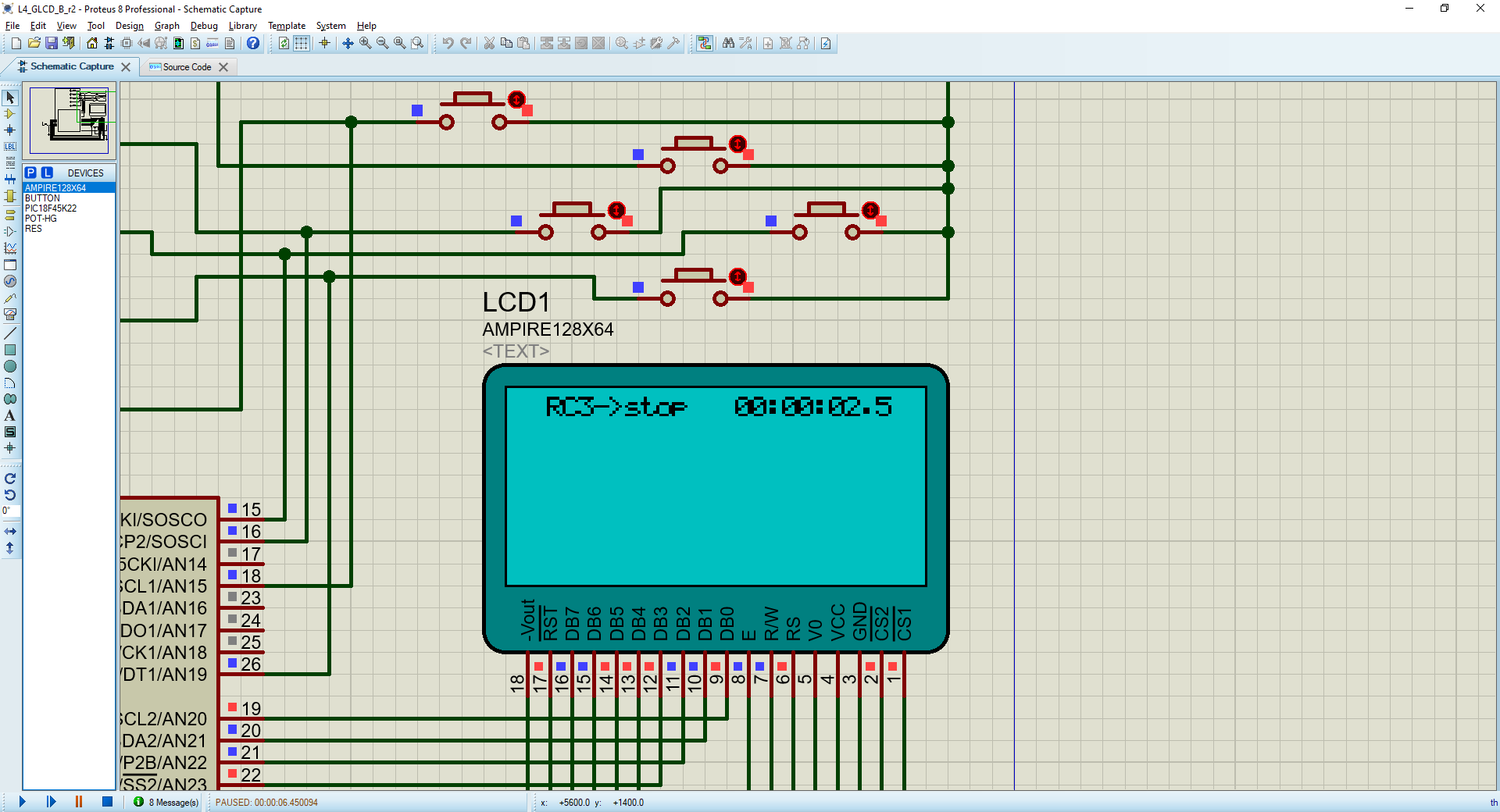
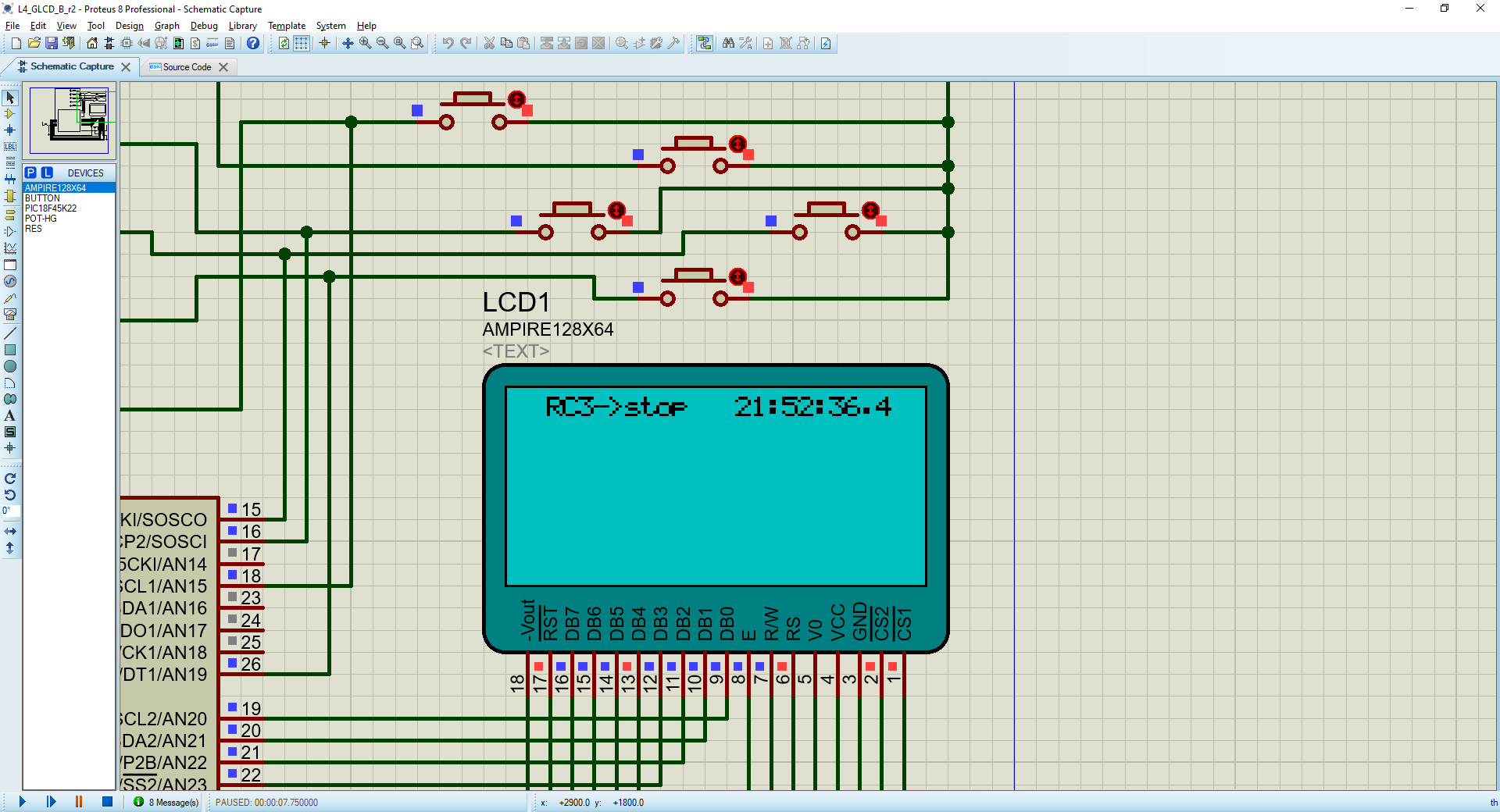
 

Figura 10. Estat running

Una nova pulsació del botó RC3 farà que el programa passi automàticament a STOPPED. S’ha d’indicar amb el text “RC3->reset” per indicar que la propera vegada que s’apreti el botó RC3 es farà un reset del comptador. També s’ha de mostrar el valor del temporitzador.

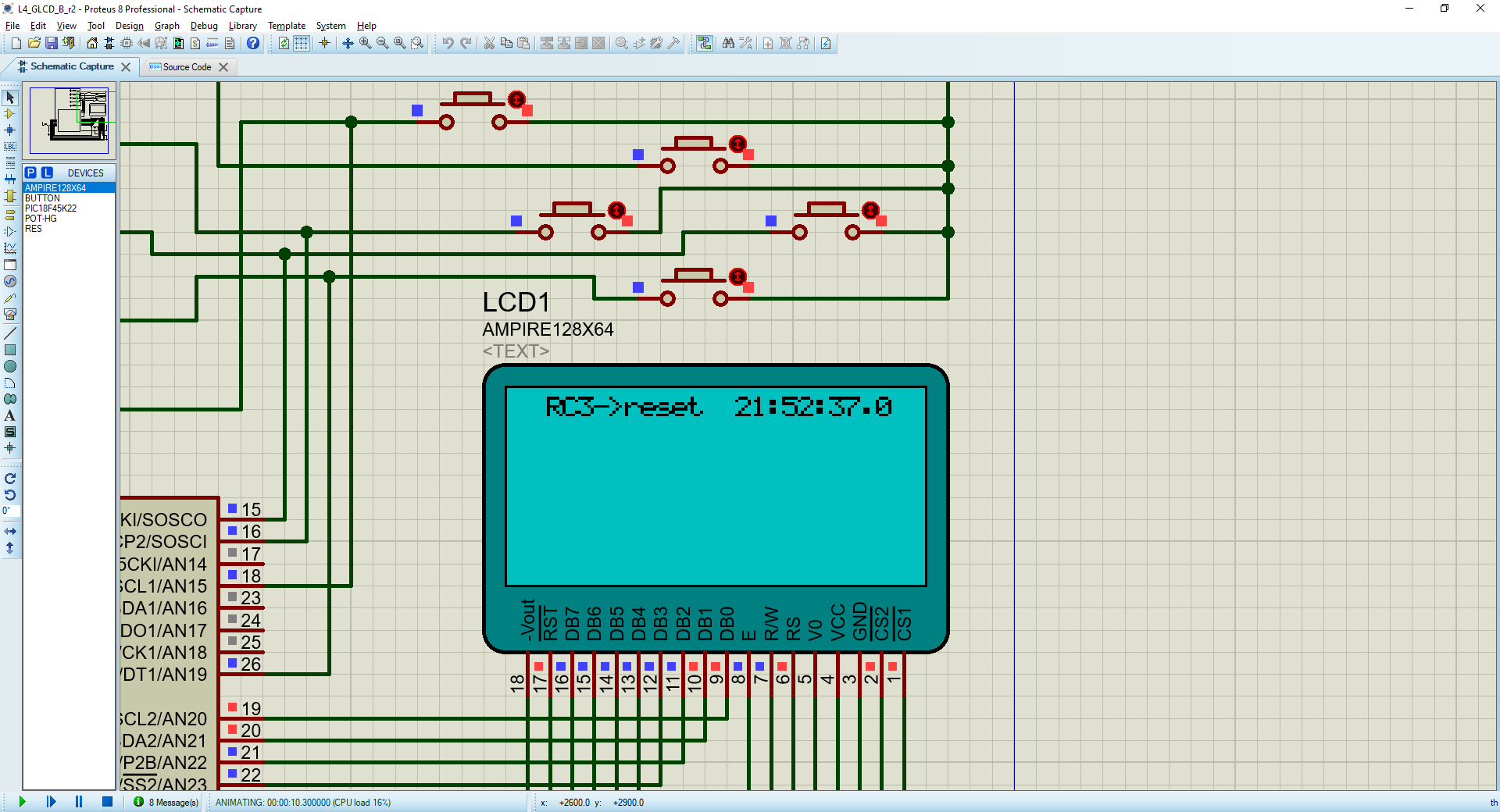


Figura 11. Estat stopped

La següent pulsació del botó RC3 netejarà els comptadors al valor inicial, i passarem de nou a l’estat INIT. S’ha de mostrar per pantalla el text “RC3->start”:

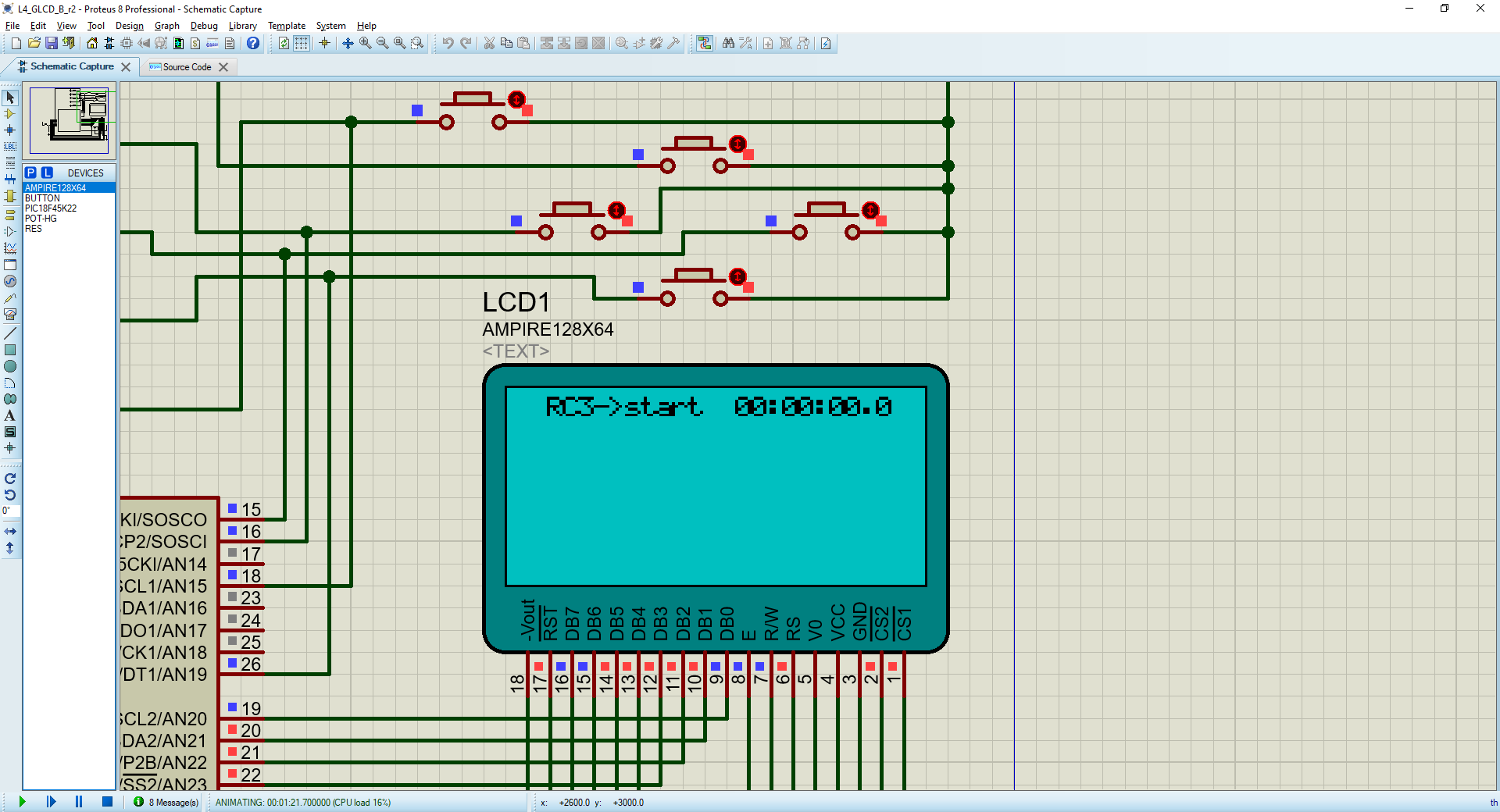


Figura 12. Tornada a l’estat inicial

|  |
| --- |
| // A proposal for main.c ..... just to inspire you  #include <xc.h>  #include "config.h"  #include "GLCD.h"  ....  #define \_XTAL\_FREQ 8000000 // Needed for \_\_delay\_ms function  typedef enum  {  Ready=0,  Running,  Stopped  }state\_t;  // Global Variables (decimes, estat del crono, etc.)  unsigned int decimes=0;  ....  // RSI High Priority for handling Timer0  ....  // Detection of Edge on Button press  char lecturaFlancRC3() {  ....  }  // Initialize PORTs and basic PIC resources  void InitPIC() {  ....  }  void main(void) {  ....  InitPIC();  ....  // MAIN LOOP  while (1) {  if (lecturaFlancRC3()) { // edge of button press detected  switch(estatCrono) {  // depending on the current state, handle the transition  ....  }  }  updateGLCD(·);// show things on the GLCD  ....  }  } |

**4- Rúbrica treball previ L4 (B)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Iniciat  (0-2.5 punts) | En desenvolupament  (2.5-5.0 punts) | Aconseguit  (5.0-7.5 punts) | Exemplar  (7.5-10 punts) |
| Animació noms (0.5 punts): | Funciona malament, no hi ha animació | Hi ha animació però el codi no és gens reutilitzable | El codi té només alguna part reutilitzable | Funciona perfectament i la funció d’animació té en compte espais, posicions inicials, posicions finals, etc. |
| Transició botó RC3 (flanc de pujada)  (2 punts) | Hw mal dissenyat i funcionament erroni | Hw ben dissenyat però funcionament erroni | Funcionament correcte però no hi ha estratègia anti-rebots | Funcionament correcte amb ha estratègia anti-rebots |
| Programació Timer0  (2.5 punts) | Hi han més de 3 errors en la programació dels registres del timer i interrupcions | Hi han entre 2 i 3 errors en la programació dels registres del timer i interrupcions | Hi han entre 1 i 2 errors en la programació dels registres del timer i interrupcions | No hi han errors en la programació dels registres del timer i interrupcions |
| Pantalla GLCD  (2.5 punts) | La pantalla no presenta la informació correctament, apareixen punts on no toca, s’actualitza la pantalla en llocs del codi no adients | La pantalla no presenta la informació correctament, o apareixen punts on no toca, o s’actualitza la pantalla en llocs del codi no adients | La pantalla funciona bé però es fan més actualitzacions de les necessàries | Funciona perfectament i s’aprofiten els recursos |

Cognoms i Nom GRUP

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Quina és la Freqüència de Clock a la que treballa el micro de la EasyPIC?
2. Quant temps dura un Cicle d’Instrucció (*Instruction Cycle*)?
3. Indica els càlculs realitzats per configurar el timer0 amb una periodicitat de 0.1 segons.
4. Indica el valor amb què has configurat els registres següents i una breu descripció de la seva funcionalitat.

T0CONbits.T08BIT =

T0CONbits.T0CS =

T0CONbits.T0SE =

T0CONbits.PSA =

T0CONbits.T0PS2 =

T0CONbits.T0PS1 =

T0CONbits.T0PS0 =

TMR0H =

TMR0L =

INTCONbits.TMR0IF =

INTCONbits.T0IE =

T0CONbits.TMR0ON =

1. Quina és la situació que fa que es generi una Interrupció de Timer0?
2. Repeteix el diagrama de blocs de la Figura 8 perquè l’aplicació funcioni com una alarma que activi una sortida digital passat un cert temps, en lloc d’un cronòmetre.