

السلام عليكم ورحمة الله وبركاته

حديثنا اليوم اخوتي الكرام عن ربط شاشة *LCD* المحرفية بالمتحكم ..

(دون الدخول في تفاصيل نظرية)

ALCD: alphanumeric liquid crystal display

هذه الشاشة (كالتى نراها في الآلات الحاسبة).. تحوي متحكماً صغيراً يتحكم بالإظهار عليها .. ويمكن التخاطب بين المتحكم الموجود فيها والمتحكم الذي نبرمجه ..

(وهذا ما سنتعلمه اليوم بإذن الله)



تتألف من 16 أو 20 عمود .. وسطر أو سطرين أو أربع .. (سنتعامل اليوم مع 16 * 2)

حيث أن الحجرة الواحدة يمكنها أن تحوي حرفاً أو رمزاً أو رقماً

يمكننا اعتبارها جدولاً كما يلي :

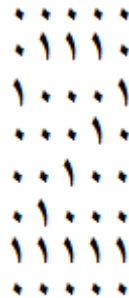
A	I	p	h	a	n	u	m	e	r	i	c	a	I		
L	C	D		1	2	3	4	5	6	7	8	9	/	*	-

كل حجرة من هذه الحجرات عبارة عن مصفوفة من البكسلات ... (٥ أعمدة ٨ أسطر)

بحيث أن كل محرف يتم رسمه رسماً ... وله خريطة محفوظة في ذاكرة الشاشة ..

يتم حفظ كل خريطة في ٨*٨ بت .. بحيث يمثل كل بت حالة البكسل (مضاء أو غير مضاء)

مثلاً الرقم 2 محفوظ كما يلي:



إن دراسة القسم النظري والبنية الداخلية للشاشة .. وأوامر المتحكم الخاص بهاأمر ممتع
ولكننا سنركز هنا على الجزء العملي واستخدام الشاشة في الإظهار ... (بدل استخدام الأضواء
LEDs كل مرة ..)

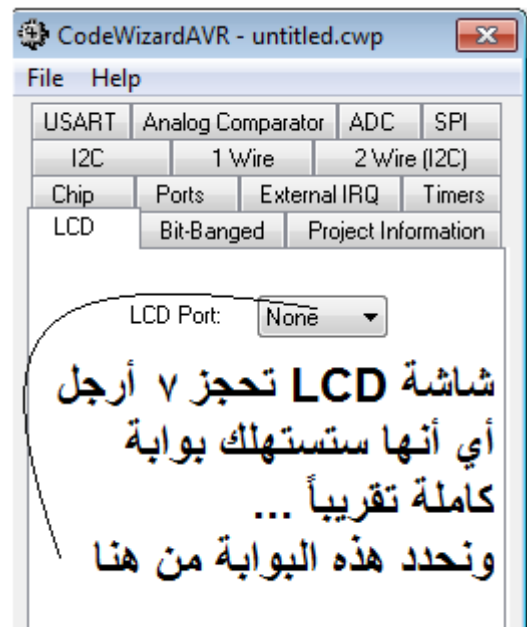
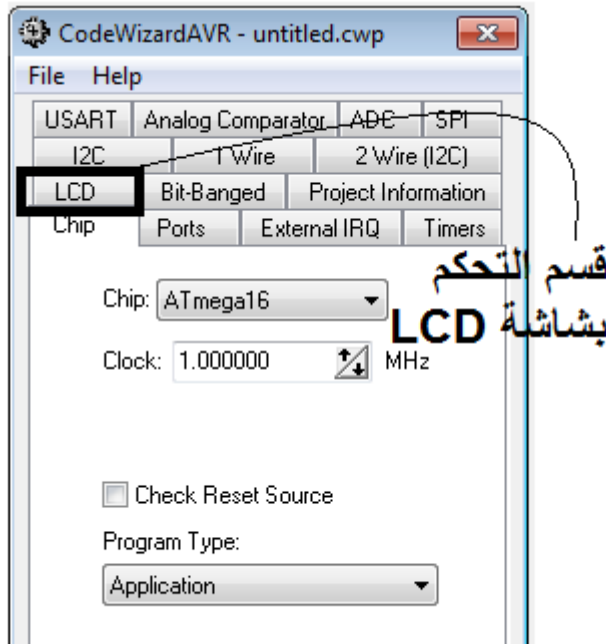
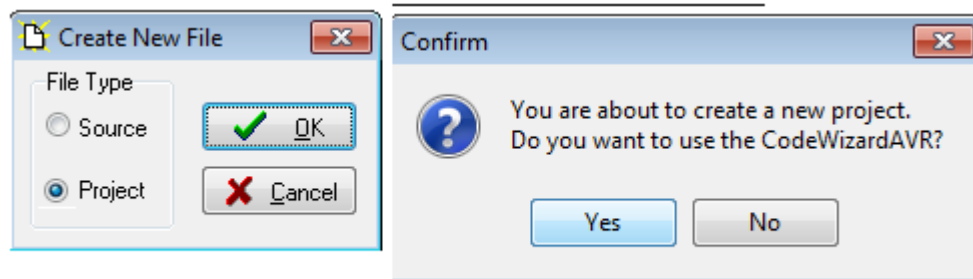
لدراسة القسم النظري ...

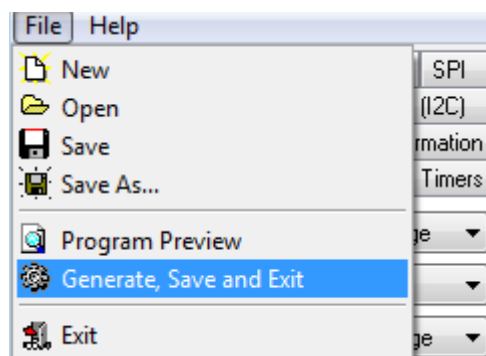
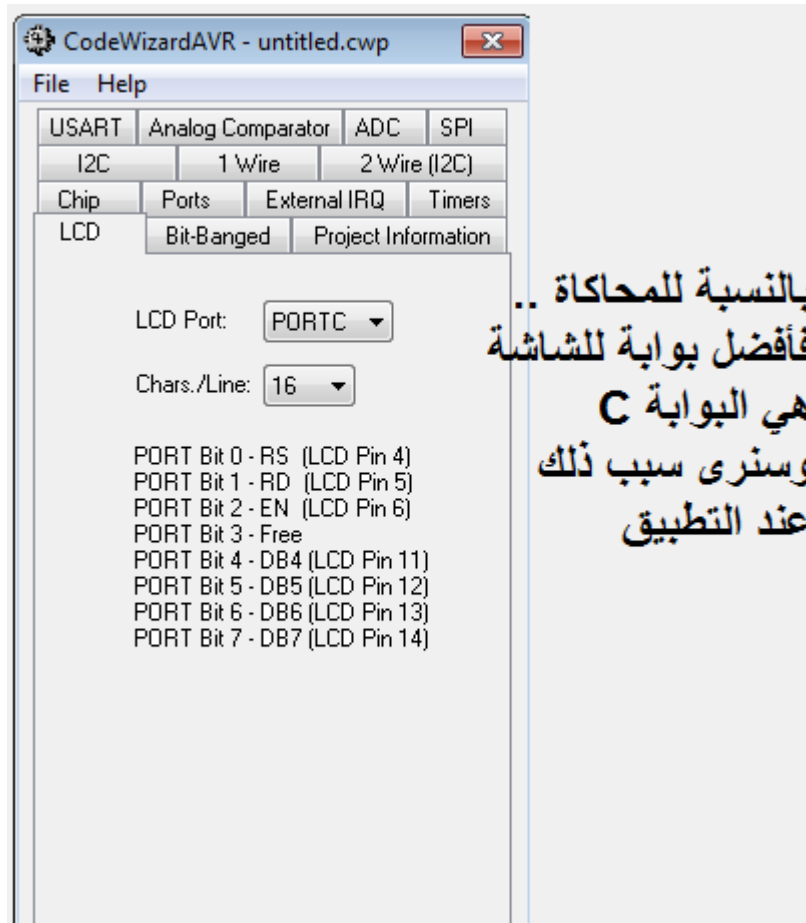
أنصح بقراءته من كتاب الأستاذ اسماعيل الطرودي ..

(برمجة متحكمات AVR بلغة C) (صفحة 67) ..

أما هنا فسنذكر ما يهمنا فقط ...

وسنبداً بكتابة الكود في CodeVision ..





لنتابع داخل الكود .. ما الذي تمت إضافته ؟

تمت إضافة مكتبة *lcd.h* والتي تحوي الدوال الكافية للتعامل مع الشاشة ..

كما تمت إضافة تعليمة أسمبلي .. لتعريف البوابة المستخدمة للشاشة ..

```

25
26 // Alphanumeric LCD Module functions
27 #asm
28     .equ __lcd_port=0x15 ;PORTC
29 #endasm
30 #include <lcd.h>

```

كما تمت إضافة هذا التابع :

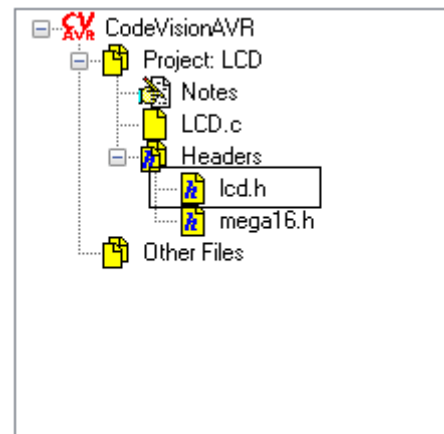
```
120
121 // LCD module initialization
122 lcd_init(16);
...
```

والمسؤول عن تجهيز استخدام الشاشة .. الوسيط الممرر للتابع هو عدد الأعمدة للشاشة
والآن يمكننا استخدام التوابع ...

ما هي التوابع ؟ .. اضغط **ctrl+F9** لعمل **Build All** ..

(طبعاً لن يكون هناك أخطاء في البرنامج)

والآن من الشاشة الجانبية اضغط على اسم المكتبة التي تريد قراءة دوالها :



فيما يلي : التوابع الموجودة في المكتبة ..

```
25 void _lcd_ready(void);
26 void _lcd_write_data(unsigned char data);
27 // write a byte to the LCD character generator or display RAM
28 void lcd_write_byte(unsigned char addr, unsigned char data);
29 // read a byte from the LCD character generator or display RAM
30 unsigned char lcd_read_byte(unsigned char addr);
31 // set the LCD display position x=0..39 y=0..3
32 void lcd_gotoxy(unsigned char x, unsigned char y);
33 // clear the LCD
34 void lcd_clear(void);
35 void lcd_putchar(char c);
36 // write the string str located in SRAM to the LCD
37 void lcd_puts(char *str);
38 // write the string str located in FLASH to the LCD
39 void lcd_putsf(char *flash);
40 // initialize the LCD controller
41 unsigned char lcd_init(unsigned char lcd_columns);
```

تابع للتحكم بمكان مؤشر الكتابة

تابع محو الشاشة

تابع كتابة محرف واحد على الشاشة

تابع كتابة نص مخزن مع البرنامج

تابع تهيئة الشاشة

يجب أن تعلم بوضوح أن التوابع السابقة كتبها المهندسون الذين صمموا الـ **Code Vision**

وهي ليست أفضل ما يمكن للعلم تقديمه .. فلو استعملت لاحقاً بيانات برمجة أخرى قد تجد توابع أكثر وأفضل وأسهل استخداماً ... فلا تنزعج من ذلك ..

علينا معرفة بعض المعلومات النظرية عن شاشة *ALCD* تحوي هذه الشاشة متحكماً وذاكرة داخلية .. تحتفظ الذاكرة بالمحارف التي تقابل كل رقم ..
 مثلما يحدد ترميز *ASCII* في الحاسوب الحرف *A* بالرقم ٦٥ ..
 وكل محرف له خريطة لرسمه .. (مثلا ٨*٨ بكسل)
 ففي شاشة *ALCD* هناك ترميز محدد (ليس *ASCII*) ..
 ويتم حفظ الخريطة بأبعاد ٨*٥ غالبا ..
 مثلاً إذا كنت تريد كتابة الحرف "ب" على الشاشة ..
 فعليك برمجتها لتحتفظ بخريطة الحرف "ب" وأن يكون له رقم معين مثلاً ٩٧ ..
 وعندما يتم إرسال أمر كتابة الرمز رقم ٩٧ .. ستكتب الحرف "ب"
 إذاً : **نحفظ خريطة معينة بالنسبة لرقم معين ..**
 أما عن كيفية رسم الخريطة فهي عبارة عن ٨*٥ بت .. كل بت يعبر عن بكسل إما أسود أو أبيض ..

لنعتبر أن ٠ مضاء و ١ غير مضاء

سيكون الحرف "ب" كما يلي (مجرد مثال) :



وبذلك كل سطر سيستهلك بايت .. ومن أجل 8 أسطر سنحتاج 8 بايت ..
 (قلنا أن السطر 5 بت فقط .. وهي البتات التي على اليمين (الأقل أهمية) وآخر 3 بتات على اليسار لا يتم اعتبارها من الخريطة ولا يتم رسمها)
 يكفي كلاماً نظرياً ...
 لنبدأ بالتطبيق العملي ...

سنكتب برنامجاً لطباعة *Hello World* !!

أو أي عبارة أخرى تفضلها ☺

اكتب جزء الكود التالي (خارج حلقة *while*)

```
120
121 // LCD module initialization
122 lcd_init(16);
123
124 lcd_gotoxy(0,0);
125 lcd_putsf("Hello World!!");
126
127 while (1)
```

الدالة *lcd_gotoxy* تقوم بنقل المؤشر إلى x,y الممررين كوسطاء للدالة..

انتبه لعدم تجاوز أبعاد الشاشة ..

الدالة *lcd_putsf* تقوم بطباعة سلسلة محرفية مخزنة داخل الكود ..

حرف *f* آخر الدالة يدل على أن السلسلة مخزنة في الذاكرة *flash*

وهذا يقودنا لبعض الحديث عن أنواع ذواكر المتحكم :

للمتحكم الصغري ثلاث أنواع للذاكرة موجودة معاً ..

ذاكرة *flash* يُحفظ فيها الكود .. والثوابت ..

وذاكرة *SRAM* يتم فيها إنشاء المتحولات الديناميكية .. (أي أنها تُستعمل وقت التنفيذ)

وذاكرة *EEPROM* .. (نتحدث عنها في وقتها ..يمكن الكتابة عليها والقراءة منها .)

الذاكرة *sram* مثلما في الحاسب الشخصي *ram* لا تحتفظ بالبيانات بعد انقطاع التغذية

أما *flash* فتحتفظ بالبيانات .. و *eprom* يمكن برمجتها لتحتفظ بالبيانات

المهم أن نعرف أنه لو كان لدينا سلسلة ثابتة فسنستعمل *lcd_putsf*

أما لو أنشأنا مصفوفة أثناء وقت التنفيذ *Runtime* فستحفظ في *sram*

وسنستخدم التابع *lcd_puts* لإظهارها على الشاشة ..

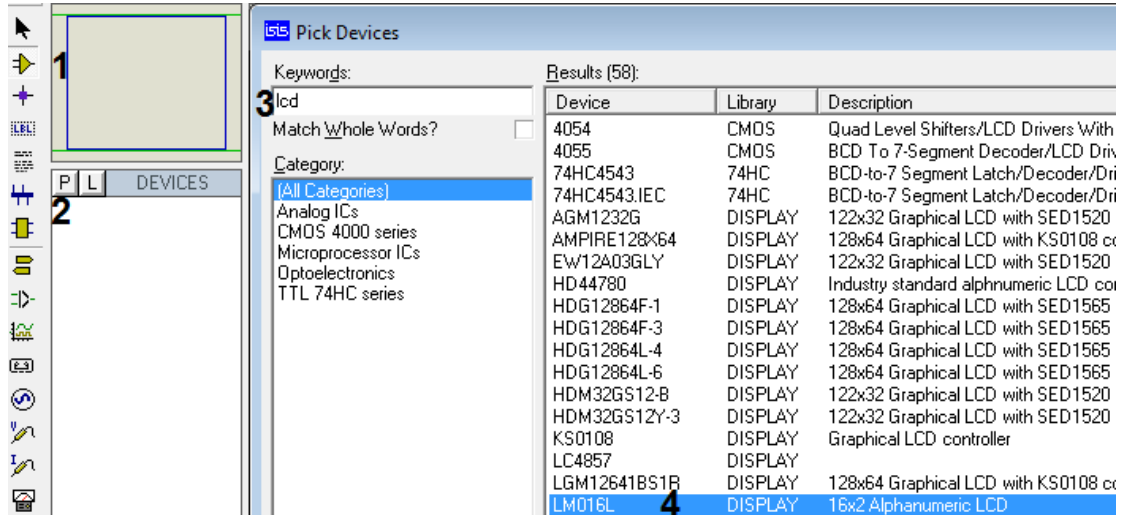
هذا كل ما يهمنا حتى الآن ...

كفى كلاماً ولنبدأ بالتنفيذ ...

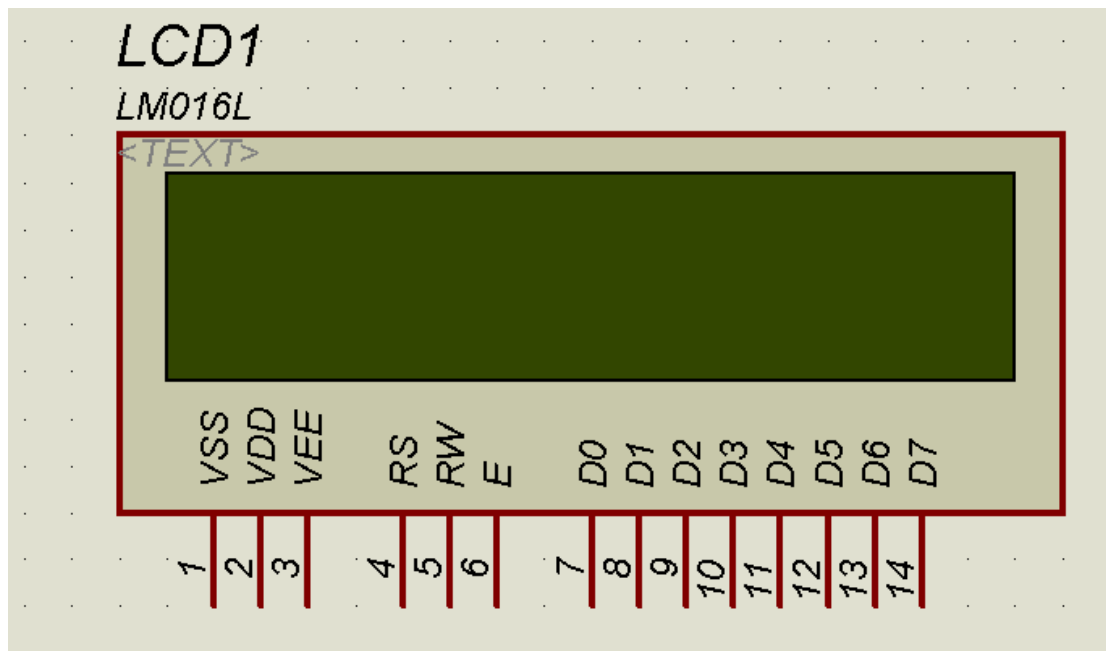
Ctrl+F9 اعمل Build للكود ..وتأكد من عدم وجود أخطاء ..

والآن ننقل إلى المحاكاة ..في Proteus

بعد فتح البرنامج سنختار الشاشة ١٦*٢ :



هذا شكل الشاشة هنا :



بالنسبة للأرجل VSS, VDD.. أرجل التغذية vss الجهد الموجب .. vdd الأرضي

أما vee فهو مسؤول عن تباين الشاشة .. وهي بين أن تكون: مضاءة لدرجة أن تكون الكتابة فاتحة جداً .. إلى .. أن تكن مظلمة بحيث يصعب تمييز الحروف عن خلفية الشاشة ..

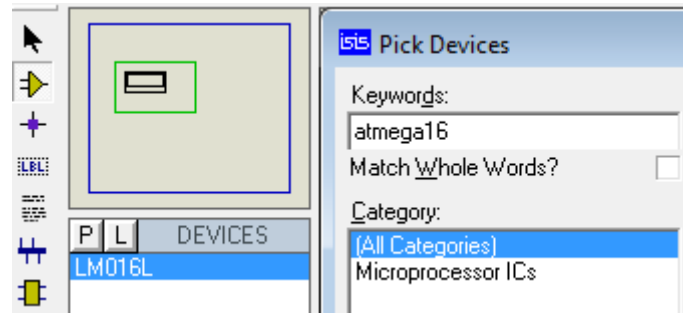
RS:register select قطب اختيار المسجلات داخل الشاشة ..

RW:read or write لاختيار نمط القراءة أو الكتابة ..

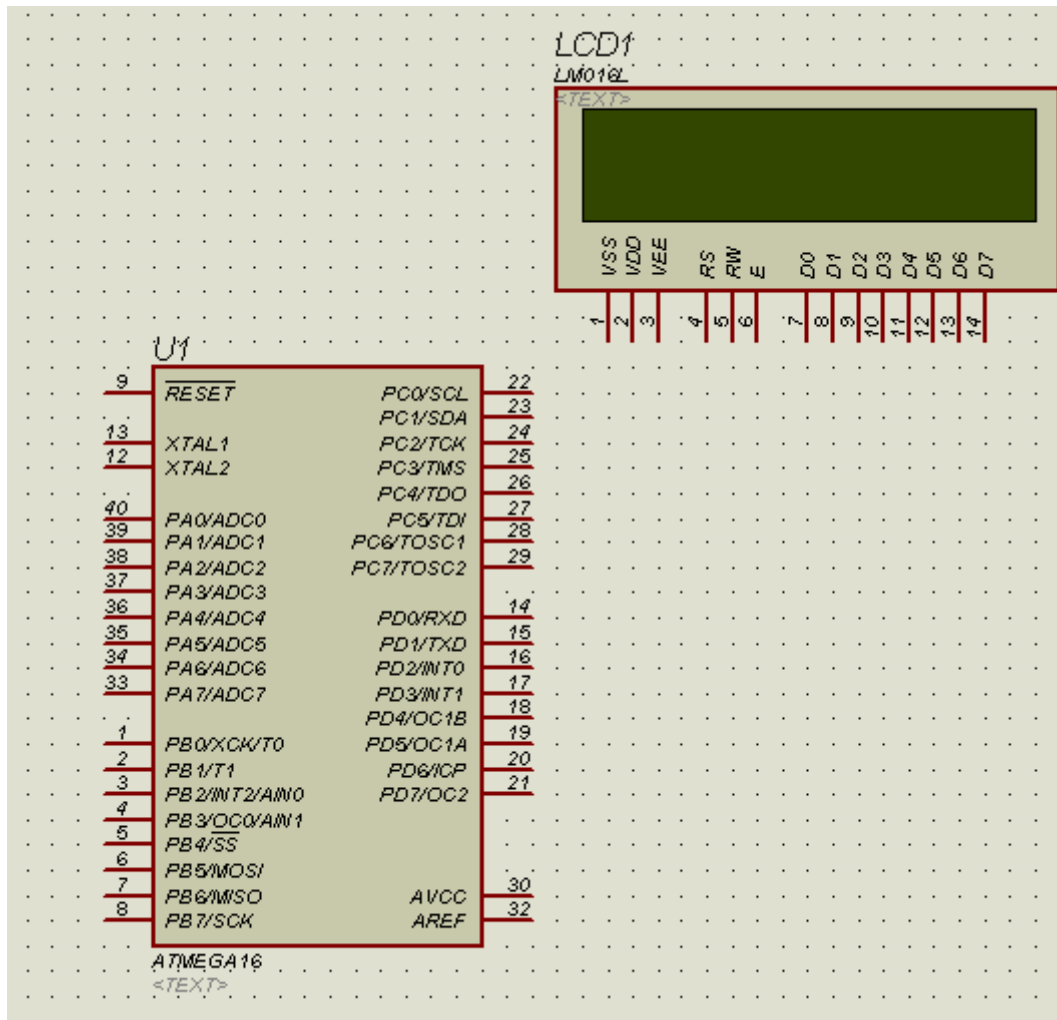
.. *Enable* : E لتأهيل الإدخال ..

والباقى $D0 > D7$ أرجل المعطيات ..

والآن إلى التوصيل .. (لا تنس اختيار المتحكم)

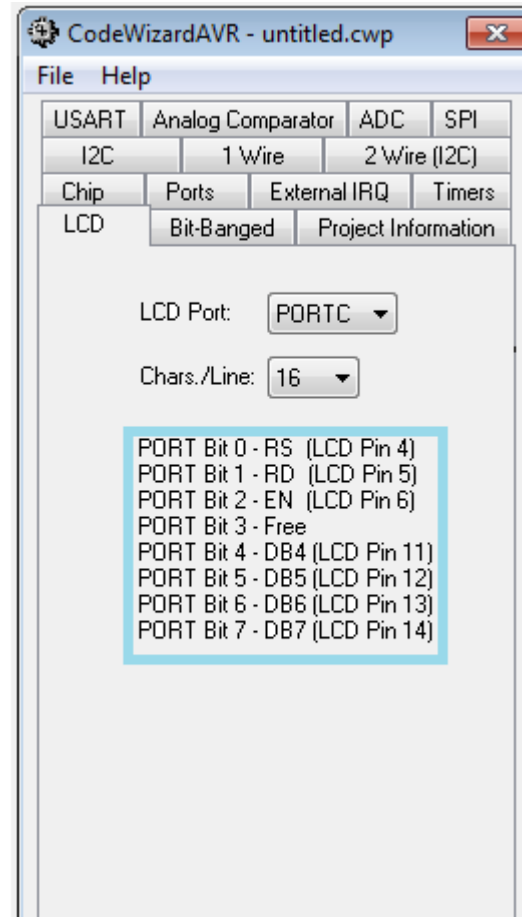


نضع الشاشة والمتحكم كما في الشكل التالي :



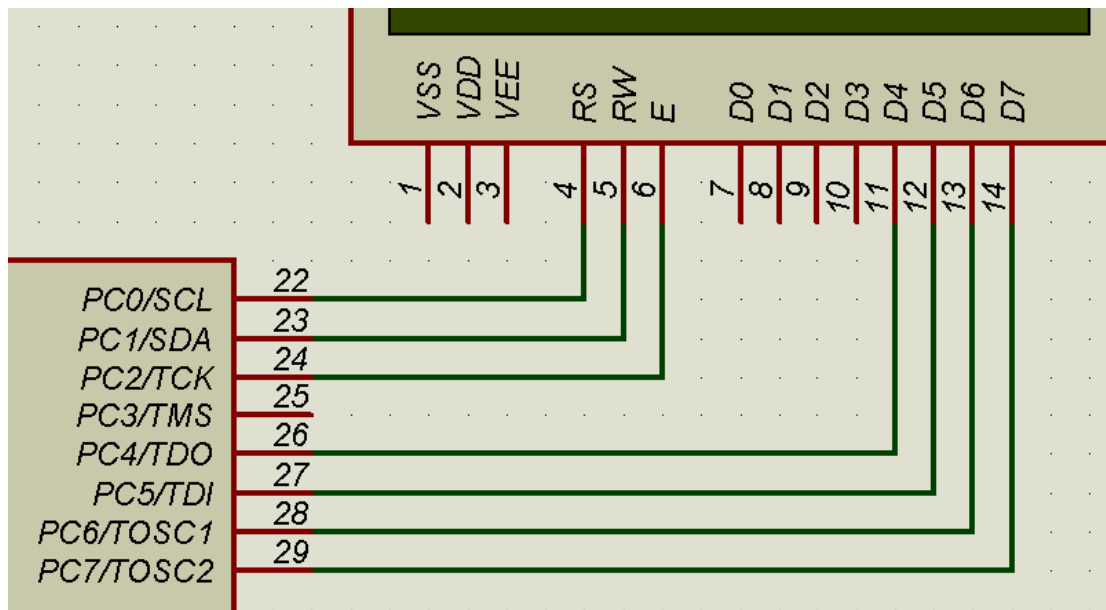
أما بالنسبة لتوصيل الأرجل ..

فلاحظ ماالذي كان مكتوبا عندما حددنا من *codevision* البوابة الخاصة بالشاشة :



على اليسار مكتوب أرجل المتحكم .. وعلى اليمين أرجل الشاشة :

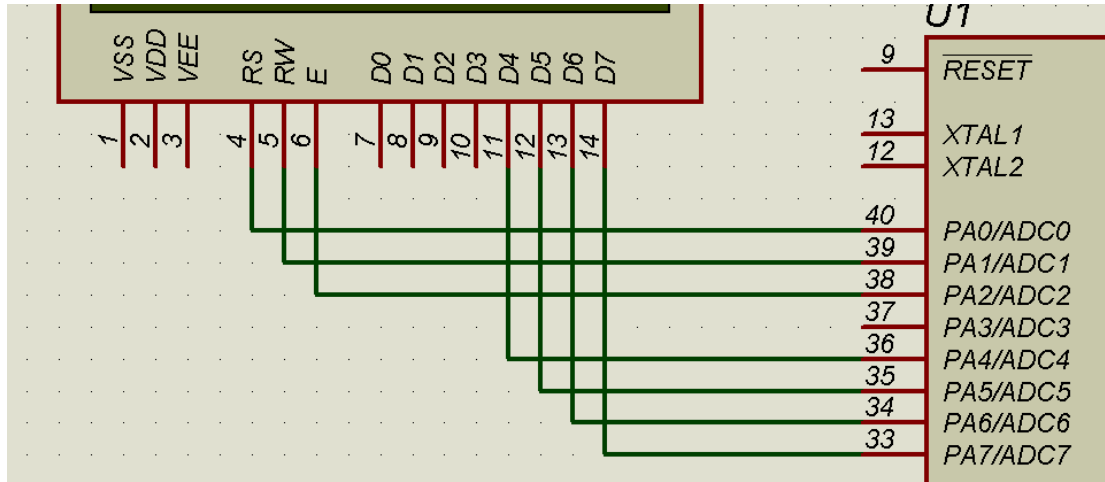
وهكذا سيبدو التوصيل :



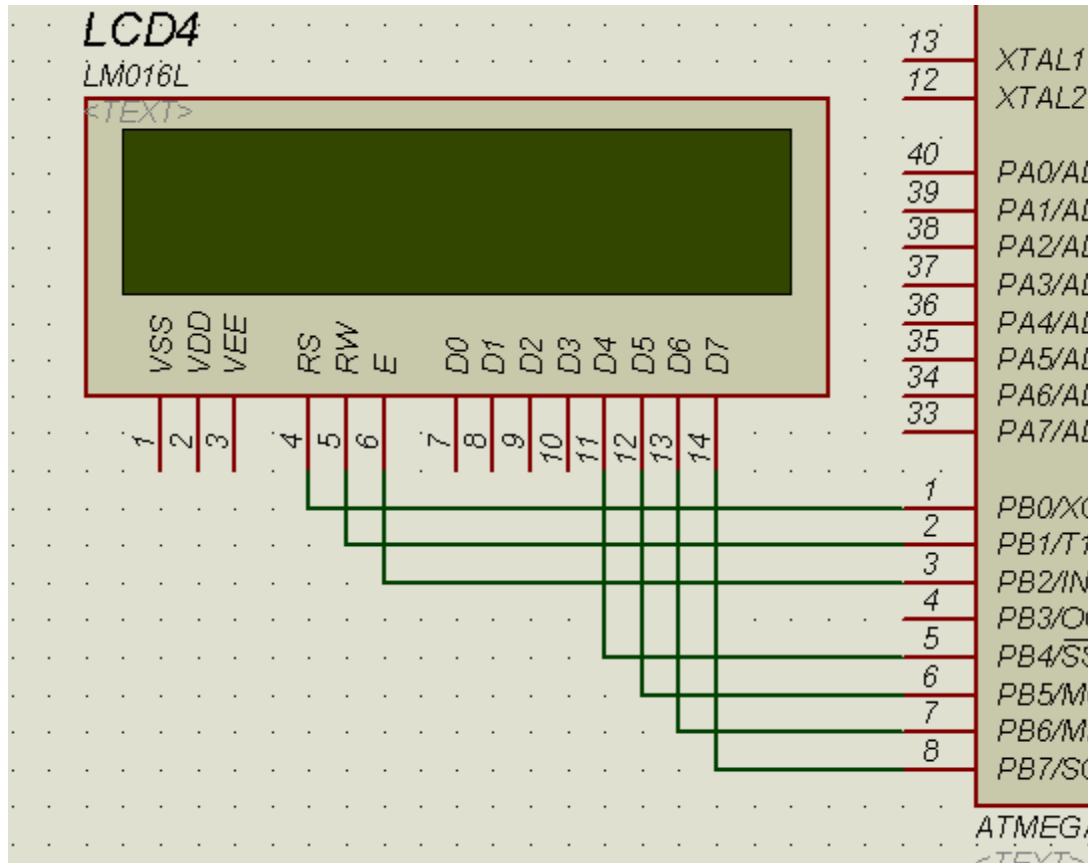
آخر أربعة مع آخر أربعة ... وأول ثلاثة مع أول ثلاثة ☺

لو أننا قمنا بتحديد بوابة أخرى فسيكون التوصيل كما يلي :

بالنسبة للبوابة A :



البوابة B :

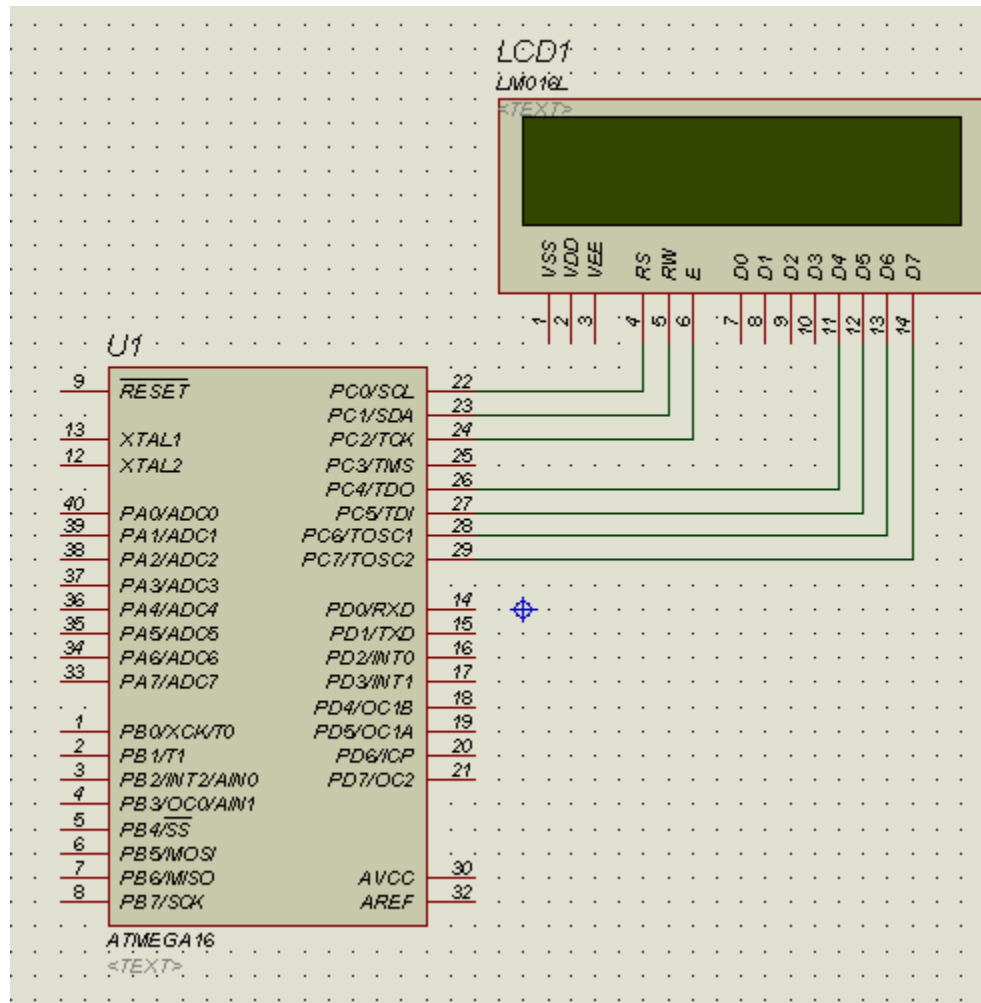


قاعدة التوصيل كما ذكرنا: آخر أربعة مع آخر أربعة .. وأول ثلاثة مع أول ثلاثة ...

ولكن في باقي البوابات نرى أن التوصيلات قد لا تبدو مرتبة ..

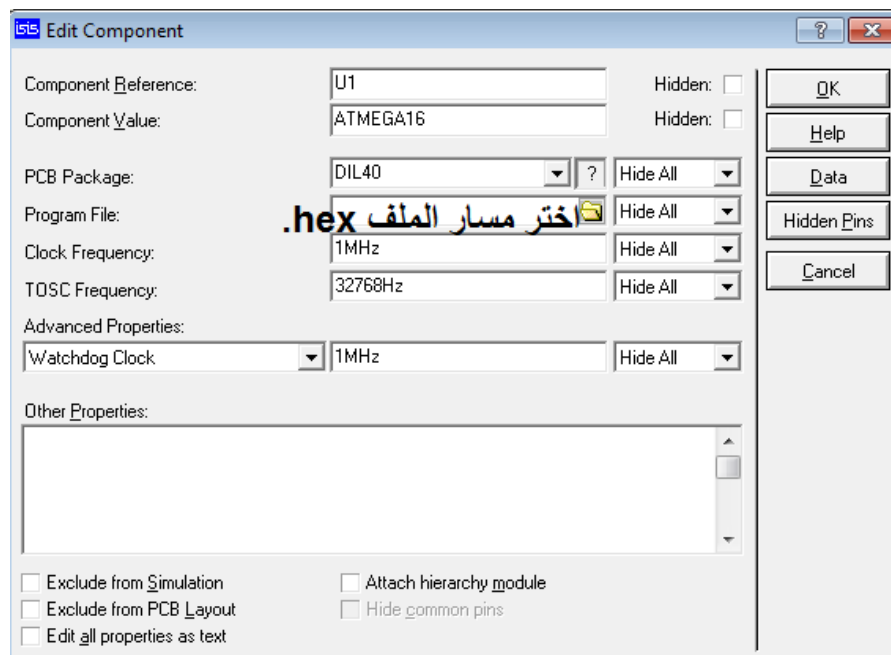
حاول توصيل الشاشة للبوابة D واكتشف سبب تفضيل C عليها ☺

بعد أن وصلنا البوابة C إلى الشاشة :



سنقوم بجلب الكود إلى المتحكم :

(نقرتان على المتحكم وتظهر النافذة التالية)





والآن عند تشغيل المحاكاة

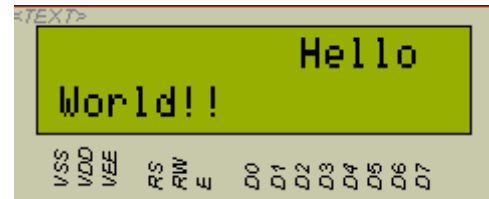
0VSQL	2
1VSDA	2
2VTCM	2
3VTMS	2
4VTD0	2
5VTD1	2
6VTD2	2
7VTD3	2
8VTD4	2
9VTD5	2
10VTD6	2
11VTD7	2
12VTD8	2
13VTD9	2
14VTD10	2
15VTD11	2
16VTD12	2
17VTD13	2
18VTD14	2
19VTD15	2
20VTD16	2
21VTD17	2
22VTD18	2
23VTD19	2
24VTD20	2
25VTD21	2
26VTD22	2
27VTD23	2
28VTD24	2
29VTD25	2
30VTD26	2
31VTD27	2
32VTD28	2
33VTD29	2
34VTD30	2
35VTD31	2
36VTD32	2
37VTD33	2
38VTD34	2
39VTD35	2
40VTD36	2
41VTD37	2
42VTD38	2
43VTD39	2
44VTD40	2
45VTD41	2
46VTD42	2
47VTD43	2
48VTD44	2
49VTD45	2
50VTD46	2
51VTD47	2
52VTD48	2
53VTD49	2
54VTD50	2
55VTD51	2
56VTD52	2
57VTD53	2
58VTD54	2
59VTD55	2
60VTD56	2
61VTD57	2
62VTD58	2
63VTD59	2
64VTD60	2
65VTD61	2
66VTD62	2
67VTD63	2
68VTD64	2
69VTD65	2
70VTD66	2
71VTD67	2
72VTD68	2
73VTD69	2
74VTD70	2
75VTD71	2
76VTD72	2
77VTD73	2
78VTD74	2
79VTD75	2
80VTD76	2
81VTD77	2
82VTD78	2
83VTD79	2
84VTD80	2
85VTD81	2
86VTD82	2
87VTD83	2
88VTD84	2
89VTD85	2
90VTD86	2
91VTD87	2
92VTD88	2
93VTD89	2
94VTD90	2
95VTD91	2
96VTD92	2
97VTD93	2
98VTD94	2
99VTD95	2
100VTD96	2
101VTD97	2
102VTD98	2
103VTD99	2
104VTD100	2
105VTD101	2
106VTD102	2
107VTD103	2
108VTD104	2
109VTD105	2
110VTD106	2
111VTD107	2
112VTD108	2
113VTD109	2
114VTD110	2
115VTD111	2
116VTD112	2
117VTD113	2
118VTD114	2
119VTD115	2
120VTD116	2
121VTD117	2
122VTD118	2
123VTD119	2
124VTD120	2
125VTD121	2
126VTD122	2
127VTD123	2
128VTD124	2
129VTD125	2
130VTD126	2
131VTD127	2
132VTD128	2
133VTD129	2
134VTD130	2
135VTD131	2
136VTD132	2
137VTD133	2
138VTD134	2
139VTD135	2
140VTD136	2
141VTD137	2
142VTD138	2
143VTD139	2
144VTD140	2
145VTD141	2
146VTD142	2
147VTD143	2
148VTD144	2
149VTD145	2
150VTD146	2
151VTD147	2
152VTD148	2
153VTD149	2
154VTD150	2
155VTD151	2
156VTD152	2
157VTD153	2
158VTD154	2
159VTD155	2
160VTD156	2
161VTD157	2
162VTD158	2
163VTD159	2
164VTD160	2
165VTD161	2
166VTD162	2
167VTD163	2
168VTD164	2
169VTD165	2
170VTD166	2
171VTD167	2
172VTD168	2
173VTD169	2
174VTD170	2
175VTD171	2
176VTD172	2
177VTD173	2
178VTD174	2
179VTD175	2
180VTD176	2
181VTD177	2
182VTD178	2
183VTD179	2
184VTD180	2
185VTD181	2
186VTD182	2
187VTD183	2
188VTD184	2
189VTD185	2
190VTD186	2
191VTD187	2
192VTD188	2
193VTD189	2
194VTD190	2
195VTD191	2
196VTD192	2
197VTD193	2
198VTD194	2
199VTD195	2
200VTD196	2
201VTD197	2
202VTD198	2
203VTD199	2
204VTD200	2
205VTD201	2
206VTD202	2
207VTD203	2
208VTD204	2
209VTD205	2
210VTD206	2
211VTD207	2
212VTD208	2
213VTD209	2
214VTD210	2
215VTD211	2
216VTD212	2
217VTD213	2
218VTD214	2
219VTD215	2
220VTD216	2
221VTD217	2
222VTD218	2
223VTD219	2
224VTD220	2
225VTD221	2
226VTD222	2
227VTD223	2
228VTD224	2
229VTD225	2
230VTD226	2
231VTD227	2
232VTD228	2
233VTD229	2
234VTD230	2
235VTD231	2
236VTD232	2
237VTD233	2
238VTD234	2
239VTD235	2
240VTD236	2
241VTD237	2
242VTD238	2
243VTD239	2
244VTD240	2
245VTD241	2
246VTD242	2
247VTD243	2
248VTD244	2
249VTD245	2
250VTD246	2
251VTD247	2
252VTD248	2
253VTD249	2
254VTD250	2
255VTD251	2
256VTD252	2
257VTD253	2
258VTD254	2
259VTD255	2
260VTD256	2
261VTD257	2
262VTD258	2
263VTD259	2
264VTD260	2
265VTD261	2
266VTD262	2
267VTD263	2
268VTD264	2
269VTD265	2
270VTD266	2
271VTD267	2
272VTD268	2
273VTD269	2
274VTD270	2
275VTD271	2
276VTD272	2
277VTD273	2
278VTD274	2
279VTD275	2
280VTD276	2
281VTD277	2
282VTD278	2
283VTD279	2
284VTD280	2
285VTD281	2
286VTD282	2
287VTD283	2
288VTD284	2
289VTD285	2
290VTD286	2
291VTD287	2
292VTD288	2
293VTD289	2
294VTD290	2
295VTD291	2
296VTD292	2
297VTD293	2
298VTD294	2
299VTD295	2
300VTD296	2
301VTD297	2
302VTD298	2
303VTD299	2
304VTD300	2
305VTD301	2
306VTD302	2
307VTD303	2
308VTD304	2
309VTD305	2
310VTD306	2
311VTD307	2
312VTD308	2
313VTD309	2
314VTD310	2
315VTD311	2
316VTD312	2
317VTD313	2
318VTD314	2
319VTD315	2
320VTD316	2
321VTD317	2
322VTD318	2
323VTD319	2
324VTD320	2
325VTD321	2
326VTD322	2
327VTD323	2
328VTD324	2
329VTD325	2
330VTD326	2
331VTD327	2
332VTD328	2
333VTD329	2
334VTD330	2
335VTD331	2
336VTD332	2
337VTD333	2
338VTD334	2
339VTD335	2
340VTD336	2
341VTD337	2
342VTD338	2
343VTD339	2
344VTD340	2
345VTD341	2
346VTD342	2
347VTD343	2
348VTD344	2
349VTD345	2
350VTD346	2
351VTD347	2
352VTD348	2
353VTD349	2
354VTD350	2
355VTD351	2
356VTD352	2
357VTD353	2
358VTD354	2
359VTD355	2
360VTD356	2
361VTD357	2
362VTD358	2
363VTD359	2
364VTD360	2
365VTD361	2
366VTD362	2
367VTD363	2
368VTD364	2
369VTD365	2
370VTD366	2
371VTD367	2
372VTD368	2
373VTD369	2
374VTD370	2
375VTD371	2
376VTD372	2
377VTD373	2
378VTD374	2
379VTD375	2
380VTD376	2
381VTD377	2
382VTD378	2
383VTD379	2
384VTD380	2
385VTD381	2
386VTD382	2
387VTD383	2
388VTD384	2
389VTD385	2
390VTD386	2
391VTD387	2
392VTD388	2
393VTD389	2
394VTD390	2
395VTD391	2
396VTD392	2
397VTD393	2
398VTD394	2
399VTD395	2
400VTD396	2
401VTD397	2
402VTD398	2
403VTD399	2
404VTD400	2
405VTD401	2
406VTD402	2
407VTD403	2
408VTD404	2
409VTD405	2
410VTD406	2
411VTD407	2
412VTD408	2
413VTD409	2
414VTD410	2
415VTD411	2
416VTD412	2
417VTD413	2
418VTD414	2
419VTD415	2
420VTD416	2
421VTD417	2
422VTD418	2
423VTD419	2
424VTD420	2
425VTD421	2
426VTD422	2
427VTD423	2
428VTD424	2
429VTD425	2
430VTD426	2
431VTD427	2
432VTD428	2
433VTD429	2
434VTD430	2
435VTD431	2
436VTD432	2
437VTD433	2
438VTD434	2
439VTD435	2
440VTD436	2
441VTD437	2
442VTD438	2
443VTD439	2
444VTD440	2
445VTD441	2
446VTD442	2
447VTD443	2
448VTD444	2
449VTD445	2
450VTD446	2
451VTD447	2
452VTD448	2
453VTD449	2
454VTD450	2
455VTD451	2
456VTD452	2
457VTD453	2
458VTD454	2
459VTD455	2
460VTD456	2
461VTD457	2
462VTD458	2
463VTD459	2
464VTD460	2
465VTD461	2
466VTD462	2
467VTD463	2
468VTD464	2
469VTD465	2
470VTD466	2
471VTD467	2
472VTD468	2
473VTD469	2
474VTD470	2
475VTD471	2
476VTD472	2
477VTD473	2
478VTD474	2
479VTD475	2
480VTD476	2
481VTD477	2
482VTD478	2
483VTD479	2
484VTD480	2
485VTD481	2
486VTD482	2
487VTD483	2
488VTD484	2
489VTD485	2
490VTD486	2
491VTD487	2
492VTD488	2
493VTD489	2
494VTD490	2
495VTD491	2
496VTD492	2
497VTD493	2
498VTD494	2
499VTD495	2
500VTD496	2
501VTD497	2
502VTD498	2
503VTD499	2
504VTD500	2
505VTD501	2
506VTD502	2
507VTD503	2
508VTD504	2
509VTD505	2
510VTD506	2
511VTD507	2
512VTD508	2
513VTD509	2
514VTD510	2
515VTD511	2
516VTD512	2
517VTD513	2
518VTD514	2
519VTD515	2

سنقوم بتغيير بسيط في الكود الآن :

```
123  
124 lcd_gotoxy(10,0);  
125 lcd_putsf("Hello World!!");  
126
```

المفترض أننا سنتجاوز أعمدة الشاشة فما الذي سيحدث؟

(لا تنس عمل *Build* للكود)

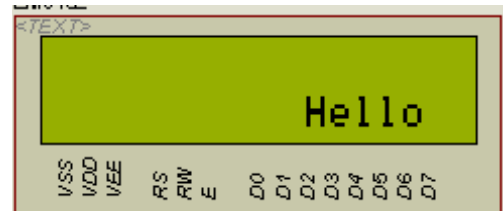


تلقائياً انتقل إلى السطر التالي ..

محاولة أخرى :

```
123  
124 lcd_gotoxy(10,1);  
125 lcd_putsf("Hello World!!");  
...
```

انتقلنا إلى العمود الحادي عشر في السطر الثاني ..



كما ذكرنا في داخل الشاشة متحكم .. وذاكرة ..

والعبارة التي تظهر على الشاشة مخزنة في ذاكرة الشاشة..

لأنها تظهر باستمرار .. فلو أعطى المتحكم أمر الإظهار ثم انتقل إلى أوامر أخرى فستبقى العبارة على الشاشة ..

أيضاً : هناك حجرة تحتفظ فيها بمكان المؤشر ..

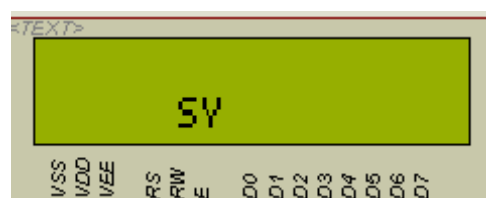
بحيث تنتقل تلقائياً إلى المربع التالي بعد طباعة محرف ما ..

مثال :

```
124 | lcd_gotoxy(5,1);  
125 | lcd_putchar('S');  
126 | lcd_putchar('Y');
```

التابع *lcd_putchar* (واضح من اسمه) يقوم بطباعة محرف على الشاشة

والنتيجة :



لو لم ينتقل المؤشر إلى الحجرة التالية .. لظهر لنا *Y* فقط .. (يطبع *S* ثم يطبع *Y* في نفس المكان)

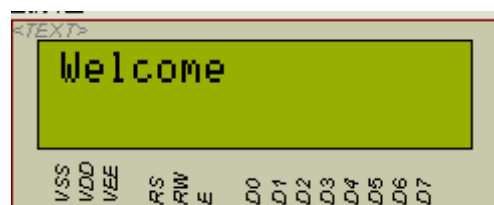
ولنتعرف على تابع آخر .. تابع *lcd_clear*

```
124 | lcd_gotoxy(5,1);  
125 | lcd_putchar('S');  
126 | lcd_putchar('Y');  
127 | lcd_clear();  
128 | lcd_putsf("Welcome");
```

المفترض أن يقوم بطباعة *SY* ثم محو الكتابة من الشاشة ..

ثم طباعة *Welcome*

والنتيجة :



لا يمكننا ملاحظة *SY* لأن الزمن قصير .. نضيف بعض التأخير ..

```
124 | lcd_gotoxy(5,1);  
125 | lcd_putchar('S');  
126 | lcd_putchar('Y');  
127 | delay_ms(1000);  
128 | lcd_clear();  
129 | lcd_putsf("Welcome");
```

(لا تنس تضمين مكتبة *delay* في بداية البرنامج)

والآن يمكننا رؤية الكتابة الأولى .. ثم الكتابة الثانية)

ونلاحظ أنه عند محو الشاشة فإن المؤشر يعود إلى (0,0)

سنكتب الآن تابعاً مفيداً ..

كثيراً ما نحتاج أثناء عمل البرنامج أن نعرف قيمة متحول ما ..

لذلك سنكتب تابعاً يطبع قيمة المتحول الرقمي على الشاشة !..

فكرة التابع :

إذا كان لدينا عدد مؤلف من منزلة واحدة (٠, ١, ٢, ٣, ٤, ٥, ٦, ٧, ٨, ٩) فيمكننا طباعته كما يلي:

لطباعة الرقم صفر :: `lcd_putchar('0');` لانتس أنه محرف وليس رقماً ..

وكما في جدول *ASCII* فإن الأرقام متتالية .. أي الصفر بعده واحد ..

فلو كتبنا `lcd_putchar('0'+1);` سيطلع لنا الرقم ١ ..

وهكذا : `lcd_putchar('0'+5);` سيطلع لنا الرقم ٥ ..

وبالتالي يمكننا طباعة الرقم X كما يلي : `lcd_putchar('0'+X);`

ولكن ماذا لو كان لدينا عدد من أكثر من منزلة ؟

يمكننا طباعة الآحاد كما يلي : `lcd_putchar('0'+X%10);`

أليس كذلك ؟

ويمكننا طباعة العشرات كما يلي : `lcd_putchar( (X/10)%10 )`

مثلاً $X=735$ فإن $X/10=73$

أي أن $(X/10)\%10=3$..

لطباعة المئات : `lcd_putchar( (X/100)%10 )` .. وهكذا ..

ولكن لدينا مشكلتان : الأولى : متى سنتوقف ..

والثانية : يجب طباعة الرقم بحيث نبدأ من أكبر منزلة .. مثلاً ٥٧٨ إذا بدأنا بالآحاد ثم العشرات

ثم المئات .. سنطبعه ٨٧٥ ..

حل المشكلة الأولى: سنتوقف عندما يكون $X/(10^n)$ يساوي الصفر !

مثلاً .. ١٤٥٦ .. $X/10=145$ ثم $X/100=14$ ثم $X/1000=1$ ثم $X/10000=0$ نتوقف

حل المشكلة الثانية : إما أن نتلاعب بالمؤشر بحيث يطبع من اليمين لليسار .. أو أن نخزن الرقم

في مصفوفة ثم نطبعه بشكل نظامي ...

الكلام ممل .. انظر إلى الكود .. لتفهم كل شيء ..

هذا الكود يحل المشكلة الثانية بطريقة تحريك المؤشر:

```
33 void printNumber(int num )
34 {
35     int x=10; تبدأ طباعة الرقم بحيث يكون المؤشر على العمود الحادي عشر
36     char N;
37     while (num!=0)
38     {
39         lcd_gotoxy(x,0); تذهب إلى العمود المناسب للمنزلة
40         N=num%10; تأخذ أحاد الرقم المطلوب طباعته
41         lcd_putchar('0'+N); تطبع الأحاد
42         num =num /10; نقسم الرقم على ١٠ لننتقل للمنزلة التالية
43         x--; تنقل المؤشر إلى الخلف
44     } نكرر حتى تنتهي منازل الرقم
45     delay_ms(50); بعض التأخير بما يكفي لرؤية الرقم
46 }
```

وهنا نحل المشكلة بطريقة التخزين في مصفوفة :

```
void printNumber2(int num )
{
char NUM[11]={0}; مصفوفة سنخزن فيها الرقم
int x=9; رقم يحدد مكاننا في المصفوفة
while (num!=0)
{
    NUM[x]='0'+num%10; نخزن أحاد الرقم
    num =num /10; نقسم الرقم على ١٠
    x--; نرجع للخلف في المصفوفة
}

lcd_gotoxy(0,0); تذهب لبداية الشاشة
lcd_puts(&NUM[x+1]); وتطبع الرقم بدءاً من أكبر منزلة
delay_ms(50);
}
```

لاحظ أننا استخدمنا الدالة `lcd_puts` .. لأن المصفوفة تم انشاؤها وقت التنفيذ ..

ولو وضعنا الدالة `lcd_putsf` فلن تعمل .. فهي خاصة بالمصفوفات الثابتة `const`

والآن لنكتب تطبيقاً نستفيد فيه من الدالة السابقة ..

تطبيق بسيط :

نقوم بتعريف متحول صحيح a في بداية الدالة $main$:

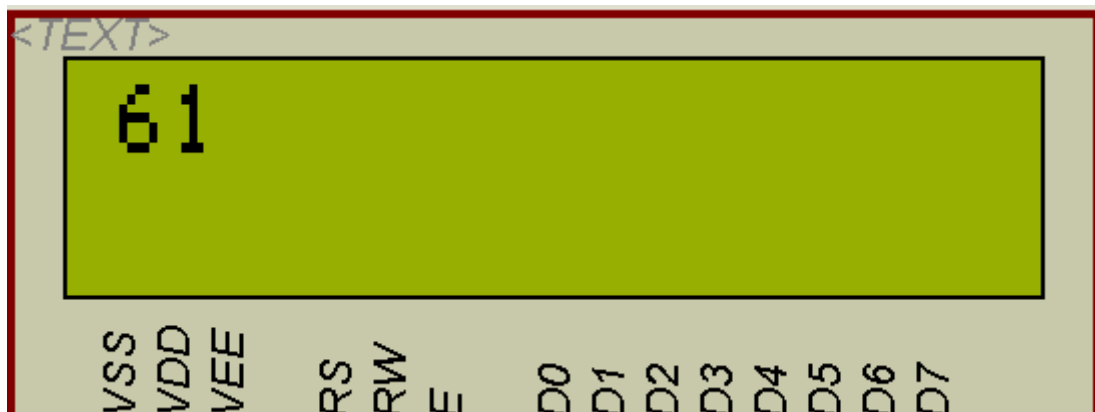
```
void main(void)
{
    // Declare your local variables here
    int a=0;
```

ونكتب ما يلي داخل حلقة $while$ في آخر الكود :

```
while (1)
{
    // Place your code here
    a++;
    printNumber2(a);
};
```

نقوم بزيادة الرقم a ثم طباعته ...

والنتيجة :



يتم طباعة الأرقام بشكل ممتاز .. والسرعة نتحكم بها من التأخير الذي وضعناه في التابع ..

..ما رأيك لو قمنا بكتابة ما يلي :

```
printNumber2 (-1) ;
```

نحن من كتبنا التابع وعلينا توقع النتيجة .. (ويجب أن يكون توقعنا صحيحاً)

طبعاً لم ندعم كتابة الأرقام السالبة .. فما الذي ستم طباعته ؟

يتم تخزين الأرقام السالبة بحيث يكون الرقم السالب هو متمم الرقم الموجب ثنائياً ..

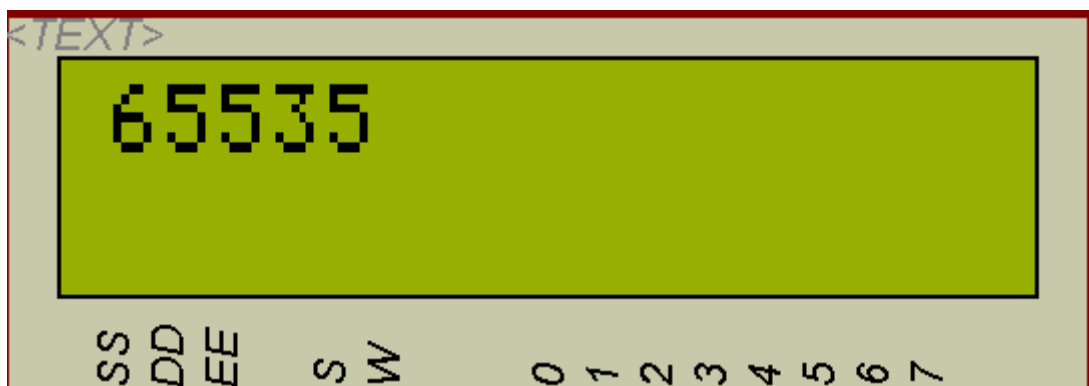
بالتمثيل الثنائي .. ماهو الرقم الذي إذا أضفناه إلى ٠٠٠١ .. سيكون الناتج ٠٠٠٠

الجواب : ١١١١ .. لأن $٠٠٠٠ = ١١١١ + ٠٠٠١$ ويكون هناك منقول إلى الخارج لا نحفظ به ولذلك إذا قمنا بطباعة الرقم ١١١١ على أنه موجب .. سيطلع ١٥ .. أي أكبر رقم يمكن تمثيله في ٤ بت ...

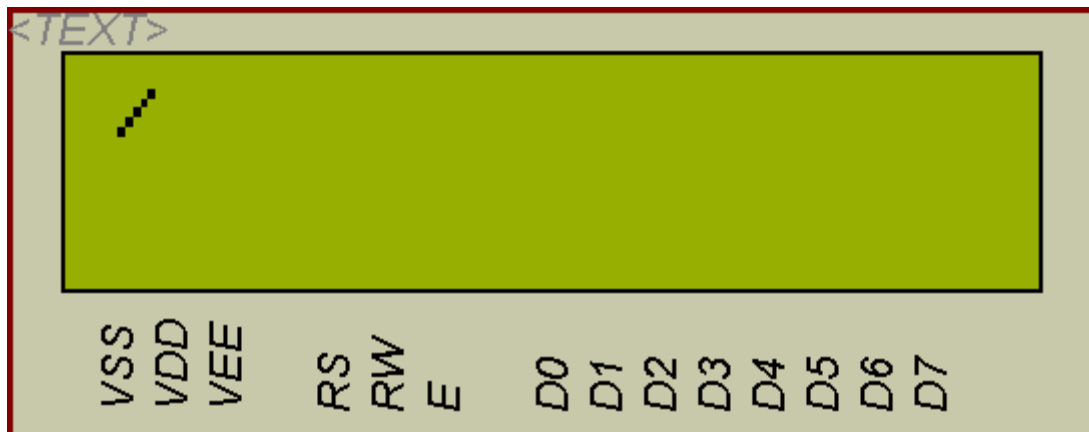
وإذا علمت أن *int* في متحكمنا تحجز ١٦ بت ..

فما الذي سيتم طباعته عوضاً عن -١ ؟

حسب توقعاتنا .. يجب طباعة :



أما في الحقيقة ... فقد تمت طباعة :



```
void printNumber2(int num )  
{  
    char NUM[11]={0};  
    int x=9;  
    while (num!=0)  
    {  
        NUM[x]='0'+num%10;    NUM[x]='0'-1;  
        num =num /10;          num=(-1)/10=0  
        x--;  
    }  
    lcd_gotoxy(0,0);  
    lcd_puts (&NUM[x+1]);  
    delay_ms(50);  
}
```

-1
-1!=0
خروج
طباعة المحرف الذي قبل 0
!!

ولتفادي هذا الخطأ ،، سنعتبر الوسيط الممرر *unsigned int*

```
void printNumber2(unsigned int num )
```

يمكنك الاطلاع على جدول الترميز الخاص بالشاشة :

Upper 4-Bit Lower 4-Bit		0000	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	CG RAM (1)		0	1	2	3	4	5	6	7	8	9	A	B	C	D
0001	(2)	!	1	4	Q	3	7				.	A	7	4	8	9
0010	(3)	"	2	B	R	6	r				「	イ	ウ	*	B	B
0011	(4)	#	3	C	S	c	z				」	ウ	テ	モ	z	*
0100	(5)	\$	4	D	T	d	t				、	エ	ト	ホ	μ	Ω
0101	(6)	%	5	E	U	e	u				・	オ	ナ	コ	ロ	ü
0110	(7)	&	6	F	V	f	v				ヲ	カ	ニ	ヨ	μ	Σ
0111	(8)	*	7	G	W	g	w				ア	キ	ヌ	ラ	9	π
1000	(1)	(	8	H	X	h	x				イ	ク	ネ	リ	」	Σ
1001	(2)	)	9	I	Y	i	y				ウ	ケ	ル	ル	”	μ
1010	(3)	*	:	J	Z	j	z				エ	コ	ロ	レ	」	キ
1011	(4)	+	:	K	C	k	c				オ	サ	ヒ	ロ	*	π
1100	(5)	×	<	L	¥	l	l				ホ	ミ	フ	ワ	キ	μ
1101	(6)	—	=	M	J	m	j				ユ	ヌ	ヘ	リ	ト	÷
1110	(7)	.	>	N	^	n	+				ヨ	セ	ホ	”	π	
1111	(8)	/	?	O	_	o	+				ウ	リ	ヲ	”	ö	■

المحرف / يسبق المحرف 0 مباشرة ..

إلى هنا ينتهي درس ربط شاشة *ALCD* بالمتحكم الصغري ..

ويبقى بضع ملاحظات :

الشاشات نوعان : *Graphical LCD* و *Alphanumeric LCD*

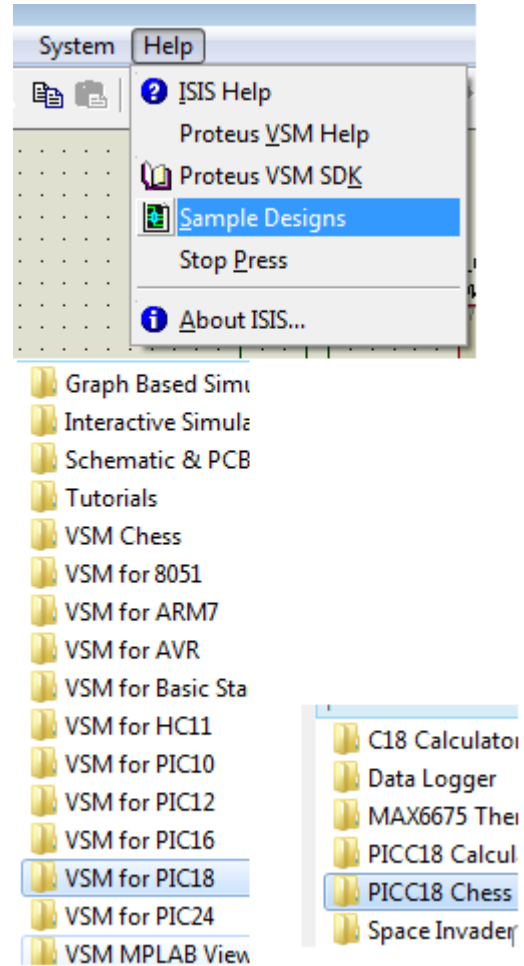
النوع الأول يمكن فيه التحكم بكل بكسل في الشاشة على حدة ..

والنوع الثاني يمكن رسم المحرف المكافئ لكل رقم ..

فمثلاً .. رأينا أن المحرف الذي قبل الصفر كان ' / ' يمكننا تغييره ورسم محرف آخر كما نريد ..

ويمكن اعتماد المحارف الأصلية دون تغيير ..

يمكنك رؤية أحد المشاريع الكبيرة (لعبة شطرنج) تستخدم شاشة الرسومات :



أستودعكم الله ..

والسلام عليكم ورحمة الله وبركاته