برمجة المتحكمات المصغرة

التجارب العملية

الجلسة الثامنة

Programming

Embedded Systems Microcontroller

You Can Practice Microcontroller Programming Easily Now!

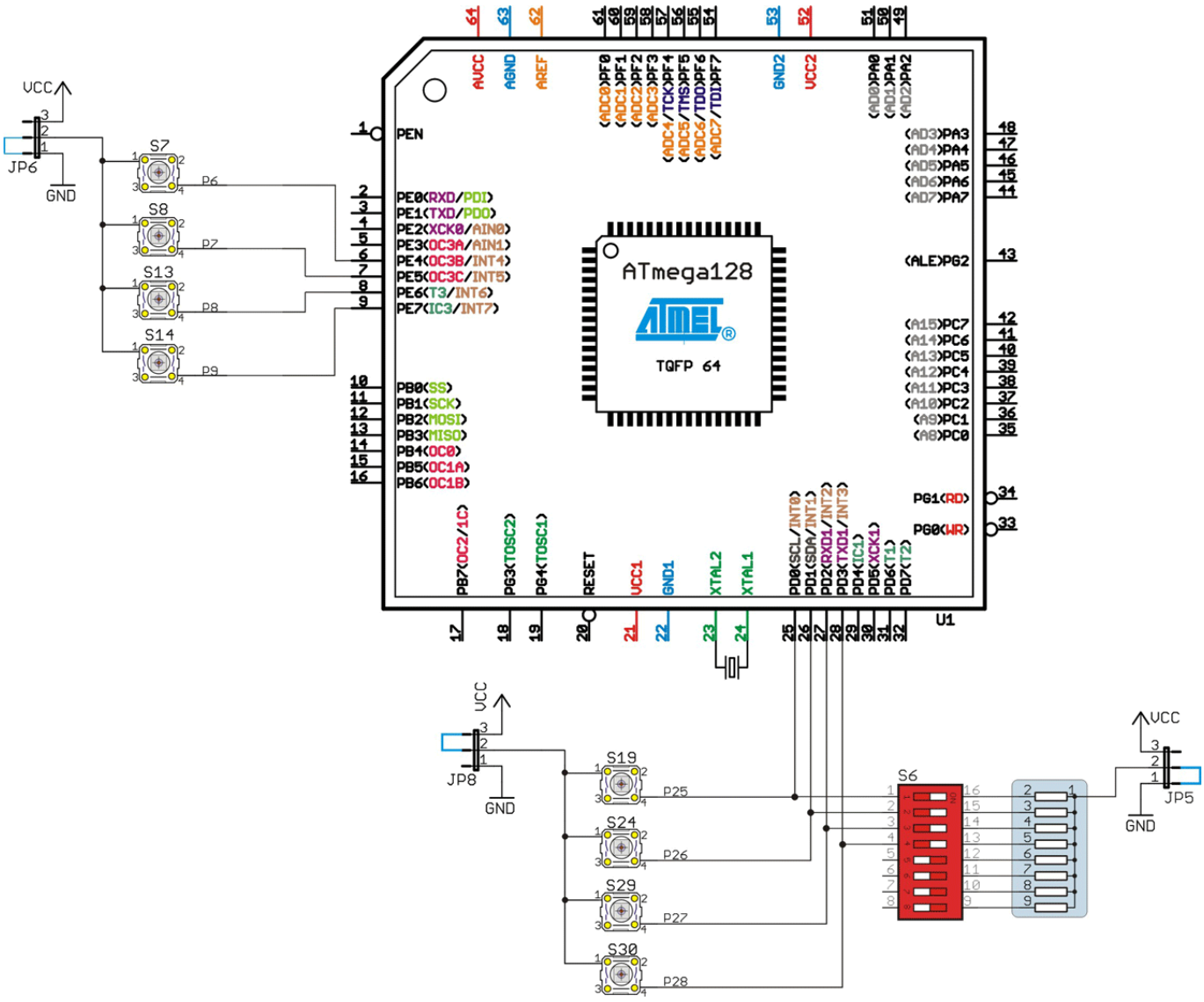
WALID BALID, Tuesday, December 15, 2009

التجربة الرابعة والعشرون: المقاطعات الخارجية
Exp.24:External Interrupts

الغاية من التجربة:

توصيل وبرمجة مجموعة مفاتيح إلى أقطاب المقاطعات الخارجية INT7 ~ INT0.

مخطط التوصيل:



متطلبات التوصيل:

يجب إغلاق النقطة JP5, JP6, JP8 إلى النقطة الأرضية.

شرح عمل الدارة:

سوف نقوم بكتابة برنامج لقراءة مفاتيح موصولة إلى أقطاب المقاطعات الخارجية.

إن الفكرة الأساسية هي أن المعالج لن ينشغل بتفحص المفاتيح من أجل معرفة حالة المفتاح، وإنما عندما تتغير الحالة المنطقية للمفتاح على قطب المقاطعة سوف يتم مقاطعة المعالج ويقفز إلى برنامج خدمة المقاطعة الخارجية المتعينة من أجل تنفيذها.

إن الطريقة التقليدية لقراءة حالة مفتاح موصول إلى قطب متحكم مصغر يمكن أن تتم بأشكال متعددة:

```
Do
      If Key1 = 0 Then
         ...
         ...
      End If

      Debounce Key1 , 0 , Sw1 , Sub

      Bitwait Key1, reset
Loop
```

التحقق من حالة المفتاح باستخدام التعليمات الشرطية

التحقق من حالة المفتاح باستخدام Debounce

التحقق من حالة المفتاح باستخدام Bitwait

إن جميع الطرق أعلاه سوف تشغل المعالج في عملية الفحص الدوري للتحقق من حالة المفتاح، أما في حال استخدمنا لمقاطعات الخارجية، فإن المعالج يقوم بمعالج الحالة فقط عندما تتحقق المقاطعة.

التعليمات الجديدة:

شرح التعليمة	التعليمة البرمجية
إعداد وتعريف المقاطعة الخارجية (x=0,1,~,7). يمكن تحديد أربعة حالات (State) للاستجابة للمقاطعة كما يلي: Falling: يتم توليد المقاطعة عند الجبهة الهابطة للنبضة المطبقة على القطب Rising: يتم توليد المقاطعة عند الجبهة الصاعدة للنبضة المطبقة على القطب Low Level: يتم توليد مقاطعة طالما أن المستوى "0" مطبقة على القطب Change: يتم توليد المقاطعة عند تغير الجبهة للنبضة المطبقة على القطب	`Config Intx = State`
عند تحقق المقاطعة يتم القفز إلى برنامج خدمة المقاطعة الموجود عن اللافتة label وينفذ برنامج خدمة المقاطعة ويعود إلى البرنامج الرئيسي.	`On Intx label`
تفعيل الاستجابة للمقاطعة الخارجية	`Enable Intx`
إلغاء تفعيل الاستجابة للمقاطعة الخارجية	`Disable Intx`
تفعيل شعاع المقاطعات العام	`Enable Interrupts`

ملاحظة: في حال كان المعالج يقوم بتنفيذ برنامج خدمة مقاطعة ما، وفي نفس الوقت حصلت مقاطعة أخرى، فإن المعالج سوف يكمل المقاطعة الجارية ويقوم بتخزين المقاطعة الطارئة حتى إذا انتهى من المقاطعة الجارية عاد إلى البرنامج الرئيسي واستدعى المقاطعة الطارئة وقام بتنفيذها.

ملاحظة: إذا حصلت عدة مقاطعات أثناء عمل المعالج في برنامج خدمة مقاطعة ما فإنه يقوم بمراكمتها حسب أولويتها ويقوم بتنفيذها وفق تسلسل الأولوية بعد انتهائه من برنامج خدمة المقاطعة الجارية.

ملاحظة: في حال أريد رفض أي مقاطعة أخرى (Int0 مثلاً) أثناء تنفيذ برنامج خدمة المقاطعة عينها، فإنه يجب إلغاء تفعيل شعاع المقاطعة (Disable Int0) أثناء معالجة برنامج المقاطعة المذكورة، وبعد الانتهاء يتم إعادة تفعيل شعاع المقاطعة (Enable Int0).

ملاحظة: ينصح بأن يكون برنامج خدمة المقاطعة قصيراً جداً (يمكن تفعيل علم تحقق المقاطعة وتفحص العلم في البرنامج الرئيسي وتنفيذ جملة تعليمات تبعاً لحالة علم المقاطعة) وجميع المعالجات تتم في البرنامج الوظيفي.

ملاحظة: عند استخدام أي مقاطعة (داخلية أو خارجية) فإنه يجب تفعيل شعاع المقاطعات العام والذي يمكن تمثيله كقاطع رئيسي لجميع المقاطعات (**Enable Interrupts**).

الجدول التالي يبين ترتيب أولوية المقاطعات.

Vector No.	Program Address(2)	Source	Interrupt Definition
1	$0000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	$0002	INT0	External Interrupt Request 0
3	$0004	INT1	External Interrupt Request 1
4	$0006	INT2	External Interrupt Request 2
5	$0008	INT3	External Interrupt Request 3
6	$000A	INT4	External Interrupt Request 4
7	$000C	INT5	External Interrupt Request 5
8	$000E	INT6	External Interrupt Request 6
9	$0010	INT7	External Interrupt Request 7
10	$0012	TIMER2 COMP	Timer/Counter2 Compare Match
11	$0014	TIMER2 OVF	Timer/Counter2 Overflow
12	$0016	TIMER1 CAPT	Timer/Counter1 Capture Event
13	$0018	TIMER1 COMPA	Timer/Counter1 Compare Match A
14	$001A	TIMER1 COMPB	Timer/Counter1 Compare Match B
15	$001C	TIMER1 OVF	Timer/Counter1 Overflow
16	$001E	TIMER0 COMP	Timer/Counter0 Compare Match
17	$0020	TIMER0 OVF	Timer/Counter0 Overflow
18	$0022	SPI, STC	SPI Serial Transfer Complete
19	$0024	USART0, RX	USART0, Rx Complete
20	$0026	USART0, UDRE	USART0 Data Register Empty
21	$0028	USART0, TX	USART0, Tx Complete
22	$002A	ADC	ADC Conversion Complete
23	$002C	EE READY	EEPROM Ready
24	$002E	ANALOG COMP	Analog Comparator
25	$0030(3)	TIMER1 COMPC	Timer/Counter1 Compare Match C
26	$0032(3)	TIMER3 CAPT	Timer/Counter3 Capture Event
27	$0034(3)	TIMER3 COMPA	Timer/Counter3 Compare Match A
28	$0036(3)	TIMER3 COMPB	Timer/Counter3 Compare Match B
29	$0038(3)	TIMER3 COMPC	Timer/Counter3 Compare Match C
30	$003A(3)	TIMER3 OVF	Timer/Counter3 Overflow
31	$003C(3)	USART1, RX	USART1, Rx Complete
32	$003E(3)	USART1, UDRE	USART1 Data Register Empty
33	$0040(3)	USART1, TX	USART1, Tx Complete
34	$0042(3)	TWI	Two-wire Serial Interface
35	$0044(3)	SPM READY	Store Program Memory Ready

برنامج تشغيل الدارة:

```
$regfile = "m128def.dat"
$crystal = 1000000
$baud = 9600
'-----------------------
Config Int0 = Rising
Config Int1 = Rising
Config Int2 = Change
Config Int3 = Change
Config Int4 = Falling
Config Int5 = Falling
Config Int6 = Change
Config Int7 = Low Level

On Int0 Int0_isr
On Int1 Int1_isr
On Int2 Int2_isr
On Int3 Int3_isr
On Int4 Int4_isr
On Int5 Int5_isr
On Int6 Int6_isr
On Int7 Int7_isr Nosave

Porte = &B11110000 : Portd = &B00000000

Enable Int0 : Enable Int1
Enable Int2 : Enable Int3
Enable Int4 : Enable Int5
Enable Int6 : Enable Int7
Enable Interrupts
'-----------------------
Do
   nop
Loop Until Inkey() = 27
End
'-----------------------
Int0_isr:
   Disable Interrupts
   Print "Int0 Occurred at Falling Edge!"
   Enable Interrupts
Return
'-----------------------
Int1_isr:
   Print "Int1 Occurred at Falling Edge!"
Return
'-----------------------
Int2_isr:
   Print "Int2 Occurred at Low Level!"
Return
'-----------------------
Int3_isr:
   Print "Int3 Occurred at Low Level!"
Return
'-----------------------
Int4_isr:
   Print "Int4 Occurred at Rising Edge!"
Return
'-----------------------
Int5_isr:
   Print "Int5 Occurred at Rising Edge!"
Return
'-----------------------
Int6_isr:
   Disable Int6
   Print "Int6 Occurred at Change of Edge!"
Return
'-----------------------
Int7_isr:
   Disable Int7
   Print "Int7 Occurred at Change of Edge!"
Return
```

التوجيهات.

إعداد وتعريف جميع المقاطعات الخارجية (x=0,1,~,7) للمتحكم ATmega128

عند تحقق المقاطعة يتم القفز إلى برنامج خدمة المقاطعة الموافقة الموجود عن اللافتة label

تفعيل مقاومات الرفع الداخلية للمقاطعات التي ستحدث عند الجبهات الهابطة وإلغائها من أجل المقاطعات عند الجبهات الصاعدة.

تفعيل المقاطعات الخارجية الثمانية.

حلقة البرنامج الرئيسي لا يتم فيها أي عملية

برنامج خدمة المقاطعة الخارجية INT0 الذي يتم تنفيذه عند ورود جبهة صاعدة على قطب هذه المقاطعة

برنامج خدمة المقاطعة الخارجية INT1 الذي يتم تنفيذه عند ورود جبهة صاعدة على قطب هذه المقاطعة

برنامج خدمة المقاطعة الخارجية INT2 الذي يتم تنفيذه عند تغير الجبهة على قطب هذه المقاطعة

برنامج خدمة المقاطعة الخارجية INT3 الذي يتم تنفيذه عند تغير الجبهة على قطب هذه المقاطعة

برنامج خدمة المقاطعة الخارجية INT4 الذي يتم تنفيذه عند ورود جبهة هابطة على قطب هذه المقاطعة

برنامج خدمة المقاطعة الخارجية INT5 الذي يتم تنفيذه عند ورود جبهة هابطة على قطب هذه المقاطعة

برنامج خدمة المقاطعة الخارجية INT6 الذي يتم تنفيذه عند ورود تغير الجبهة على قطب هذه المقاطعة

برنامج خدمة المقاطعة الخارجية INT7 الذي يتم تنفيذه طالما أن المستوى low على قطب هذه المقاطعة

التجربة الخامسة والعشرون: التعامل مع المؤقتات Exp.25: Dealing with Timers

الغاية من التجربة:

التعرف إلى أنماط عمل المؤقت Timer0 والمؤقت Timer1.

المؤقتات/العدادات:

تملك متحكمات العائلة AVR مؤقتات زمنية/عدادات ذات أداء عالي وميزات عديدة، كما أن موضوع المؤقتات/عدادات في متحكمات العائلة AVR هو موضوع واسع جداً لما لهذه المؤقتات من أنماط عمل متعددة بالإضافة لوجود أكثر من مؤقت مختلف الطول على نفس الشريحة.

ما هو المؤقت/عداد؟

بأبسط تعبير يمكننا القول بأن المؤقت/عداد عبارة عن مسجل يختلف طوله بحسب نوع المؤقت، غالباً تكون المؤقتات بطول (دقة) 8bit أو 16bit.

من أجل مؤقت/عداد 8bit فإن المؤقت يستطيع العد من 0 – 255، وأما من أجل مؤقت/عداد بطول 16bit فإن المؤقت يستطيع العد من 0 – 65536.

يملك المؤقت/عداد مدخل قدح (Clock) حيث يقوم بالعد (تصاعدياً أو تنازلياً) عند جبهات الساعة الداخلية للهزاز الكريستالي، وهذه النبضات يمكن تقسيمها عبر مقسم ترددي دخلي في بنية المؤقت (Prescaler)، وبالتالي لا حاجة لإشغال المعالج بزيادة أو إنقاص قيمة المؤقت والمؤقت سوف يعمل بشكل مستقل تماماً.

بما أن المؤقت/عداد يعمل بشكل مستقل عن المعالج، فإن المؤقت/عداد يملك مقاطعات من أجل إعلام المعالج بإتمام الوظيفة التي أعد المؤقت/عداد من أجلها.

على سبيل المثال مقاطعة الطفحان (Overflow)، حالما تصبح القيمة في مسجل المؤقت/عداد مساوية إلى القيمة العظمى (255>8bit | 65535>16bit)، فإن المؤقت/عداد يقوم بتوليد مقاطعة تسمى مقاطعة الطفحان (يجب أن تكون معدة بشكل مسبق) يقوم فيها بإعلام المعالج أنه انتهى من العد إلى القيمة العظمى.

المقسم الترددي (Prescaler):

هو عبارة عن آلية ذات مسجل بطول 10bit تقوم بتقسيم تردد الهزاز الكريستالي للمعالج من أجل تطبيقه كتردد عمل للمؤقت/عداد، بمعنى آخر إن هذا التردد المطبق على قطب الـ Clock للمؤقت/العداد هو للتزامن؛ إن قيم التقسيم الترددي المتاحة موضحة بالجدول التالي:

قيمة التقسيم	تردد عمل المؤقت بعد التقسيم
Prescaler = 1	$\mathbf{TIMER_{Clock} = F_{CPU}}$
Prescaler = 8	$\mathbf{TIMER_{Clock} = F_{CPU} / 8}$
Prescaler = 64	$\mathbf{TIMER_{Clock} = F_{CPU} / 64}$
Prescaler = 256	$\mathbf{TIMER_{Clock} = F_{CPU} / 256}$
Prescaler = 1024	$\mathbf{TIMER_{Clock} = F_{CPU} / 1024}$

إن الإعداد السابق للمؤقت يعتمد بالكلية على تردد الهزاز الكريستالي للمعالج، بالإضافة إلى نمط الإعداد لمصدر التواقت للمؤقت، بالإضافة إلى ذلك يمكن أيضاً إعداد المؤقت ليتم قدحه عن طريق تطبيق مصدر نبضات (Clock) على قطب المؤقت/العداد الخارجي (T0, T1) وبالتالي سيعمل كعداد.

حساب الزمن الأعظمي للمؤقت:

إن العلاقة التي تحسب القيمة الأعظمية لزمن المؤقت حتى الوصول إلى القمة (الطفحان) بناءً على تردد الهزاز الكريستالي وقيمة المقسم الترددي المحدد تعطى بالشكل التالي:

$$T = 2^N \frac{Prescaler}{fosc}$$

حيث أنّ:

T: الزمن الأعظمي لحدوث الطفحان للمؤقت وتعطى بالثانية.

N: دقة (طول) المؤقت؛ $N_{TIMER0}=8$، $N_{TIMER1}=16$.

Prescaler: قيمة المقسم الترددي المحددة [1, 8, 64, 256 or 1024].

f_{OSC}: تردد الهزاز الكريستالي.

حساب دقة المؤقت:

إن الدقة هنا تعني أصغر زمن يستطيع المؤقت قياسه (عده) وهو عبارة عن دور نبضة توقيت واحدة مطبقة على مدخل الـ Clock للمؤقت ويعطى بالعلاقة التالية:

$$Timer_{RESOLUTION} = \frac{1}{Input\ Ferequesncy}$$

مثال: بفرض أن تردد الهزاز الكريستالي للمعالج f_{OSC}=2MHZ وتم إعداد المقسم الترددي Prescaler=64 وبالتالي فإن تردد عمل المؤقت (Clock) يساوي:

$$Timer_{CLOCK} = \frac{fosc}{Prescaler} = \frac{2000000}{64} = 31250HZ$$

وبالتالي فإن دقة المؤقت تساوي:

$$Timer_{RESOLUTION} = \frac{1}{Input\ Ferequesncy} = \frac{1}{31250} = 0.000032\ Sec = 32\mu S$$

حيث أن الفترة 23uS هي أصغر زمن يستطيع المؤقت قياسه وفق الشروط المحددة أعلاه.

حساب القيم الأعظمية والدقة للأزمنة:

الجدول التالي يلخص القيم الأعظمية والدقة لأزمنة كلاً من Timer0, Timer1 من أجل عدة ترددات قياسية.

Timing of Timer0 at f_{OSC} = 1MHZ

Prescaler	1	8	64	256	1024
Timer Resolution in (mSec)	0.001	0.008	0.064	0.256	1.024
Maximum Timer Period (mSec)	0.256	2.048	16.384	65.536	262.144

Timing of Timer0 at f_{OSC} = 2MHZ

Prescaler	1	8	64	256	1024
Timer Resolution in (mSec)	0.0005	0.004	0.032	0.128	0.512
Maximum Timer Period (mSec)	0.128	1.024	8.192	32.768	131.072

Timing of Timer0 at f_{OSC} = 4MHZ

Prescaler	1	8	64	256	1024
Timer Resolution in (mSec)	0.00025	0.002	0.016	0.064	0.256
Maximum Timer Period (mSec)	0.064	0.512	4.096	16.384	65.536

Timing of Timer0 at f_{OSC} = 8MHZ

Prescaler	1	8	64	256	1024
Timer Resolution in (mSec)	0.000125	0.001	0.008	0.032	0.128
Maximum Timer Period (mSec)	0.032	0.256	2.048	8.192	32.768

Timing of Timer1 at f_{OSC} = 1MHZ

Prescaler	1	8	64	256	1024
Timer Resolution in (mSec)	0.001	0.008	0.064	0.256	1.024
Maximum Timer Period (Sec)	0.0655	0.5243	4.1943	16.777	67.1088

Timing of Timer1 at f_{OSC} = 2MHZ

Prescaler	1	8	64	256	1024
Timer Resolution in (mSec)	0.0005	0.004	0.032	0.128	0.512
Maximum Timer Period (Sec)	0.0328	0.262	2.097	8.3886	33.554

Timing of Timer1 at f_{OSC} = 4MHZ

Prescaler	1	8	64	256	1024
Timer Resolution in (mSec)	0.00025	0.002	0.016	0.064	0.256
Maximum Timer Period (Sec)	0.016	0.131	1.049	4.194	16.777

Timing of Timer1 at f_{OSC} = 8MHZ

Prescaler	1	8	64	256	1024
Timer Resolution in (mSec)	0.000125	0.001	0.008	0.032	0.128
Maximum Timer Period (mSec)	0.0082	0.0655	0.5243	2.097	8.3886

Timing of Timer1 at f_{OSC} = 12MHZ

Prescaler	1	8	64	256	1024
Timer Resolution in (uSec)	0.0833	0.666	5.333	21.333	85.333
Maximum Timer Period (mSec)	0.0055	0.0437	0.3495	1.3981	5.5924

Timing of Timer1 at f_{OSC} = 16MHZ

Prescaler	1	8	64	256	1024
Timer Resolution in (uSec)	0.0625	0.5	4	16	64
Maximum Timer Period (mSec)	0.0041	0.0328	0.262	1.0486	4.1943

التعليمات الجديدة:

التعليمة البرمجية	شرح التعليمة
Config Timer0 = **Timer** , Prescale = 1|8|64|256|1024	إعداد وتعريف المؤقت Timer0 ليعمل كمؤقت بطول 8bit. Prescaler: تعيين قيمة المقسم الترددي.
Config Timer0 = **Counter** , Prescale = 1|8|64|256|1024 , Edge = Rising / Falling , **Clear Timer** = 1|0	إعداد وتعريف المؤقت Timer0 ليعمل كمؤقت بطول 8bit. Edge: تعيين الجبهة (صاعدة أو هابطة) التي سيحصل عندها العد. Clear: تصفير محتوى المؤقت عند حدوث المقاطعة.
Start Timer0 **Stop Timer0**	تفعيل|إلغاء نبضة البدء للمؤقت Timer0 ليبدأ|يتوقف العد.
Enable Ovf0 / **Enable Timer0** **Disable** Ovf0 / **Disable Timer0**	تفعيل|إلغاء مقاطعة الطفحان للمؤقت Timer0.
On Ovf0 T0_isr	في حال تحقق مقاطعة الطفحان يقفز إلى البرنامج T0_isr
Timer0 = Value Value = **Timer0**	إسناد قيمة أولية|قراءة محتوى المؤقت Timer0.
Config Timer1 = Counter|Timer, Edge = Rising|Falling, Prescale = 1|8|64|256|1024, Noise Cancel = 0|1 , Capture Edge = Rising|Falling , Clear Timer = **1|0**, Compare A = Clear|Set|Toggle|Disconnect, Compare B = Clear|Set|Toggle|Disconnect إعداد وتعريف المؤقت Timer1 ليعمل كمؤقت|عداد بطول 16bit. Edge: تعيين الجبهة (صاعدة|هابطة) التي سيحصل عندها العد. Prescaler: تعيين قيمة المقسم الترددي. Noise Cancel: تفعيل|إلغاء نمط رفض الضجيج للمؤقت Timer1. Capture Edge: تعيين الجبهة (صاعدة أو هابطة) التي سيحصل عندها حادثة المسك للمؤقت Timer1. Clear Timer: تصفير|عدم تصفير محتوى المؤقت Timer1 عند حدوث مقاطعة نظير المقارنة. Compare A: تعيين سلوك خرج نظير المقارنة (OC1A) عند تحقق المقارنة بين محتوى مسجل النظير (OCR1A) وبين قيمة مسجل المؤقت Timer1 (TCNT1)، فإما أن يقوم بتطبيق "1" أو "0" أو "Inv" على القطب OC1A أو يقوم بفصل القطب. Compare B: تعيين سلوك خرج نظير المقارنة (OC1B) عند تحقق المقارنة بين محتوى مسجل النظير (OCR1B) وبين قيمة مسجل المؤقت Timer1 (TCNT1)، فإما أن يقوم بتطبيق "1" أو "0" أو "Inv" على القطب OC1B أو يقوم بفصل القطب.	
Stop Timer1 **Start Timer1**	تفعيل|إلغاء نبضة البدء للمؤقت Timer1 ليبدأ|يتوقف العد.
Enable Ovf1 / **Enable Timer0** **Disable** Ovf1 / **Disable Timer0**	تفعيل|إلغاء مقاطعة الطفحان للمؤقت Timer1.
On Ovf1 T1_isr	في حال تحقق مقاطعة الطفحان يقفز إلى البرنامج T1_isr
Enable Capture1 **Disable** Capture1	تفعيل|إلغاء مقاطعة حادثة المسك للمؤقت Timer1.
On Capture1 Cpa1_isr	في حال تحقق مقاطعة حادثة المسك يقفز إلى البرنامج Cap1_isr
Enable Compare1a **Enable Compare1b**	تفعيل|إلغاء مقاطعة نظير المقارنة للمؤقت Timer1.
On Compare1a Oc1a **On Compare1b** Oc1b	في حال تحقق مقاطعة نظير المقارنة يقفز إلى البرنامج OC1a/b
Compare1a = Value **Compare1b** = Value	إسناد قيمة أولية لمسجل نظير المقارنة للمؤقت Timer1.
Value = **Compare1a** Value = **Compare1b**	قراءة محتوى مسجل نظير المقارنة المؤقت Timer1.

أنماط عمل المؤقتات في عائلة المتحكمات AVR

أولاً: نمط العمل العام (Normal Mode):

في هذه الحالة سيعمل المؤقت عند نفس تردد عمل المعالج، وبافتراض أن تردد المعالج هو 1MHZ فإن دقة المؤقت هي 1uS، وهذا يعني أنه من أجل كل نبضة على مدخل الـ Clock للمؤقت سينقضي زمن قدره 1uSec.

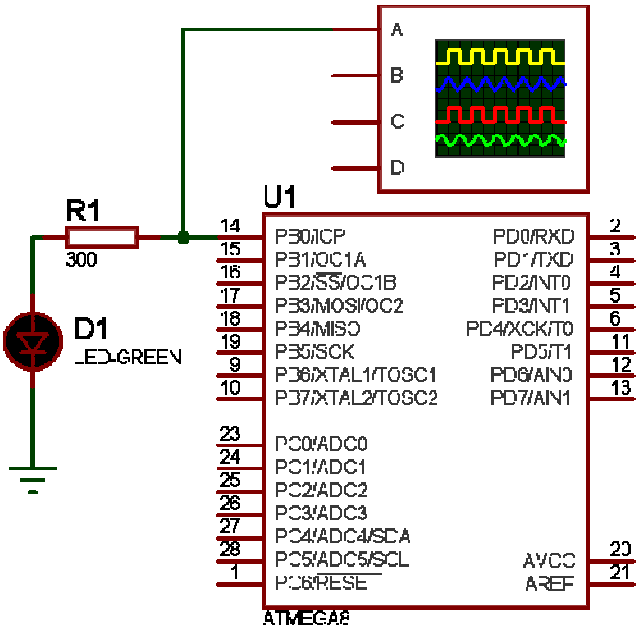
بافتراض أنه يراد انقضاء زمن تأخير قدره 50mSec، فإنه يمكن حساب عدد النبضات على مدخل التوقيت للمؤقت من أجل الزمن المذكور بالعلاقة التالية:

$$TimeCount_{Target} = \frac{\frac{1}{T_{Elapsed}}}{\frac{1}{Timer_{CLOCK}}} = \frac{\frac{1}{50\times10^{-3}}}{\frac{1}{1\times10^{6}}} = \frac{0.05}{10^{-6}} = 50000\ Pulses$$

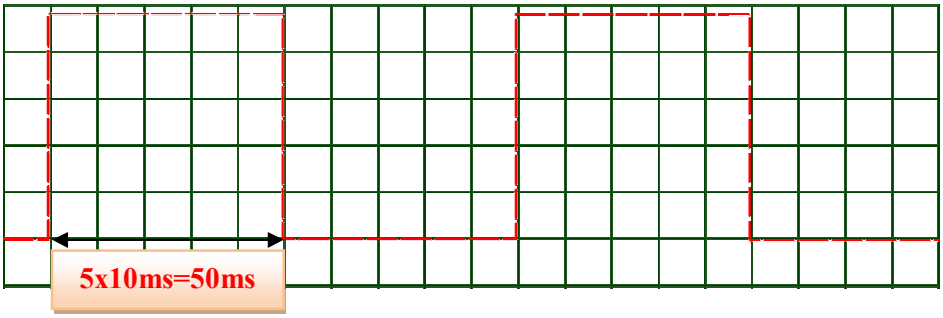
كما هو ملاحظ أن هذا العدد هو أكبر بكثير من القيمة الأعظمية التي يمكن أن يعدها المؤقت Timer0 والتي تساوي 255، لذلك يمكن أن نستخدم المؤقت Timer1.

تطبيق: سوف نستخدم زمن التأخير المسحوب أعلاه من أجل توليد قطار نبضات بفواصل زمنية 50mS.

```
$regfile = "m8def.dat"
$crystal = 1000000
$baud = 4800
'-----------------------
Config Timer1 = Timer , Prescale = 1
Start Timer1

Config Portb.0 = Output
Led Alias Portb.0
'-----------------------
Do
   If Timer1 >= 50000 Then
      Stop Timer1
         Toggle Led
         Timer1 = 0
      Start Timer1
   End If
Loop
End
```



سنحصل على راسم الإشارة على الشكل التالي:



ثانياً: نمط طفحان المؤقت (Overflow):

يمكن تمثيل المؤقت على أنه خزان "مياه" يتسع لكم معين من "المياه" متعلق بأبعاد هذا الخزان وسعته الحجمية (f_{OSC}, Prescaler, N)، وسوف يتم تفريغه بشكل آني كلما تم تعبئته بشكل كامل. وبالتالي فإنه عند تعبئة هذا الخزان بالمياه ووصول المياه إلى قمة الخزان، فإن الخزان سيصل إلى حالة تسمى الطفحان (القيمة الأعظمية للمؤقت)، عندها سيتم تفعيل علم الطفحان "OVF1 or OVF0"[1][2] الموجود في مسجل علم مقاطعة المؤقت "TIFR"، وأثناء البرنامج الرئيسي سيقوم المراقب (CPU) بمراقبة حالة هذا العلم، وعند تحقق الشرط (تصبح حالة العلم "1") أي: **Tifr.x[3] = 1**، يقوم بتنفيذ البرنامج المتعلق.

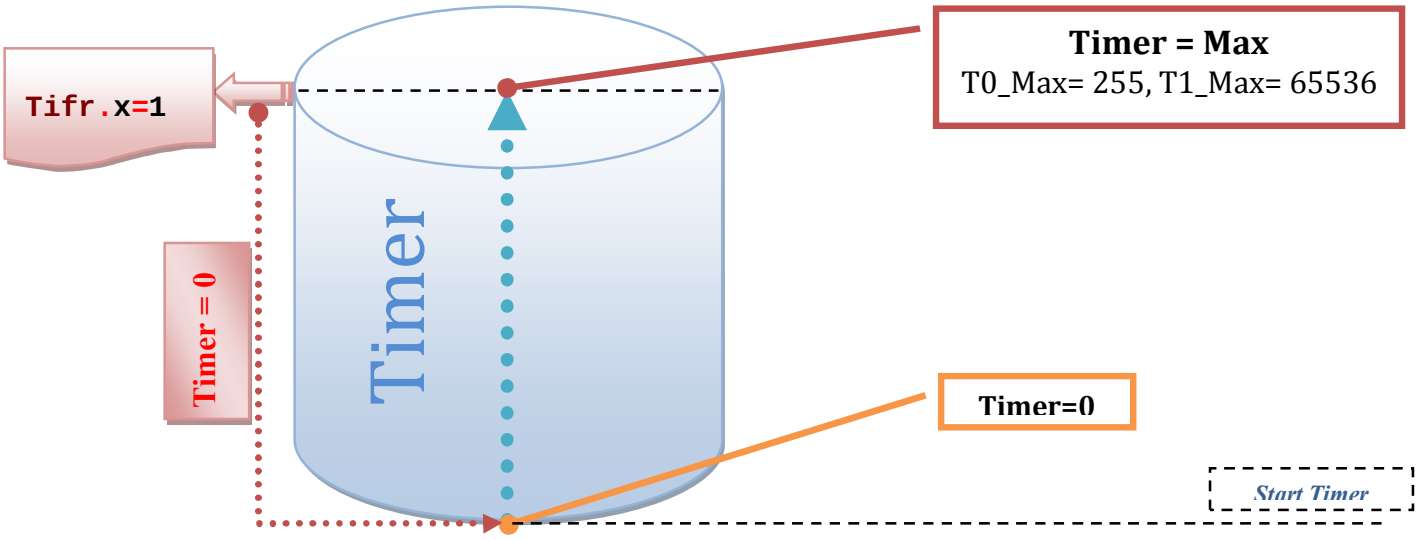
ملاحظة: في نمط الصفحان بدون تفعيل مقاطعة الطفحان، فإنه يجب تصفير علم الطفحان بكتابة القيمة "1" عليه [**Tifr.x = 1**]، أمّا عند تفعيل مقاطعة الطفحان فإن هذا العلم يتم تصفيره بشكل آني.

$$MaxVal = 2^{RES} - 1$$

إن الزمن الذي سوف يستهلكه الخزان لكي يصل إلى حالة الطفحان هو نفسه الزمن الأعظمي للمؤقت (*Maximum Timer Period*)، وبالتالي فإن هذا الزمن متعلق بشكل مباشر بما يلي:

$$T = 2^N \frac{Prescaler}{fosc}$$

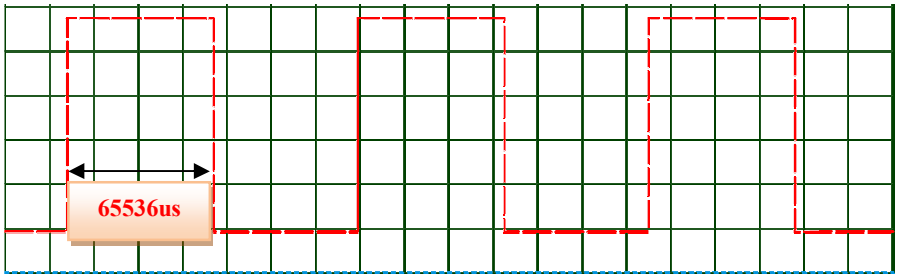
- دقة المؤقت الذي تم اختياره (N=8, 16).
- تردد الهزاز الكريستالي للمعالج (f_{OSC}).
- قيمة المقسم الترددي (Prescaler).



[1] OVF1: Timer/Counter1 Overflow Flag located in TIFR/bit 2, to check or reset this bit: TIFR.2=1
[2] OVF0: Timer/Counter0 Overflow Flag located in TIFR/bit 0, to check or reset this bit: TIFR.0=1
[3] **X = 0** for Timer0, **X = 2** for Timer1

```
$regfile = "m8def.dat"
$crystal = 1000000
$baud = 4800
'-----------------------
Config Timer1 = Timer,
Prescale = 1
Start Timer1

Config Portb.0 = Output
Led Alias Portb.0
'-----------------------
Do
   If Tifr.2 = 1 Then
      Tifr.2 = 1
      Stop Timer1
         Toggle Led
      Start Timer1
   End If
Loop
End
```

تطبيق: البرنامج جانباً يقوم بفحص علم الطفحان الذي سيفعّل عندما تصل قيمة مسجل المؤقت إلى القيمة الأعظمية، وبالتالي سوف يتم توليد قطار نبضات بفواصل زمنية 65536µSec وذلك لأن القيمة الأعظمية للمؤقت Timer1 هي 65536 والمقسم الترددي يساوي الواحد وبالتالي تردد عمل المؤقت يساوي تردد عمل المعالج (1MHZ).



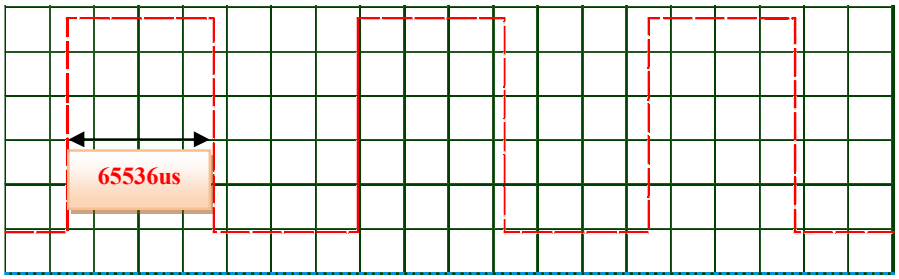
ثالثاً: نمط مقاطعة طفحان المؤقت (Overflow INT):

إن هذا النمط مشابه تماماً لنمط طفحان المؤقت السابق إلا أنه يعتمد على مقاطعة الطفحان (Hardware) لمعرفة إذا ما حصل الطفحان، وبالتالي لن ينشغل المعالج بتفحص خانة الطفحان في مسجل TIFR.

في هذه الحالة سيتم تصفير خانة علم الطفحان بشكل آلي كلما تحققت المقاطعة.

```
$regfile = "m8def.dat"
$crystal = 1000000
'-----------------------
Config Timer1 = Timer ,
Prescale = 1
Enable Ovf1
On Ovf1 T1_isr
Enable Interrupts

Config Portb.0 = Output
Led Alias Portb.0

Dim Flag As Bit
Start Timer1
'-----------------------
Do
   If Flag = 1 Then
     Reset Flag
     Toggle Led
     Start Timer1
   End If
Loop
End
'-----------------------
T1_isr:
   Stop Timer1 : Set Flag
Return
```

تطبيق: البرنامج جانباً يعتمد على مقاطعة الطفحان التي سوف تحدث عندما تصل قيمة مسجل المؤقت إلى القيمة الأعظمية، وبالتالي سوف يتم توليد قطار نبضات بفواصل زمنية 65536µSec وذلك لأن القيمة الأعظمية للمؤقت Timer1 هي 65536 والمقسم الترددي يساوي الواحد وبالتالي تردد عمل المؤقت يساوي تردد عمل المعالج (1MHZ).



رابعاً: نمط مسجل نظير المقارنة (CTC_{OCR} Mode):

يملك المؤقت Timer0 مسجل نظير مقارنة وحيد OCR0، بينما يملك المؤقت Timer1 مسجلي نظير مقارنة OCR1A, OCR1B وكذلك يملك المؤقت Timer2 مسجل نظير مقارنة وحيد OCR2.

في هذا النمط تستخدم مسجلات نظير المقارنة من أجل تخزين القيمة التي سيتم مقارنة محتويات المؤقت (TCNTn) في كل لحظة معها، وعند تحقق نتيجة المقارنة (TCNT=OCRnx) سوف يفعّل علم نظير المقارنة في مسجل الحالة للمؤقت (OCFnx).

من أجل تفعيل هذا النمط يجب كتابة "1" إلى البت الرابع (WGM12=1) في مسجل TCCR1B.

يتم حساب قيمة الشحن (المكافئة للزمن المطلوب) لمسجل نظير المقارنة بالعلاقة التالية:

$$OCRnx_{VALUE} = \frac{fosc \times T_{required}}{Prescaler}$$

حيث أن:

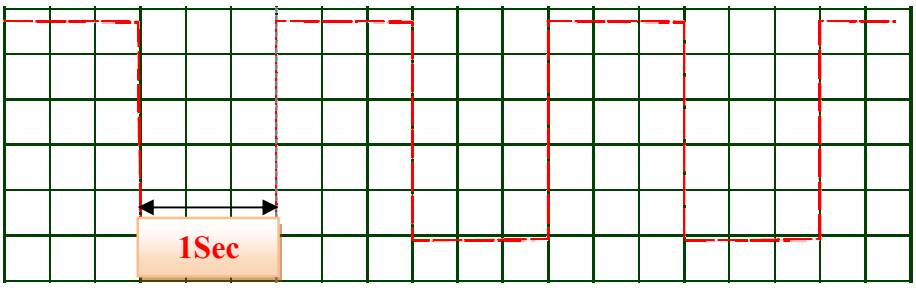
$OCRnx$: هي القيمة التي سيتم شحنها لمسجل النظير (OCR1AL, OCR1AH).

$T_{required}$: الزمن المطلوب والذي يكافئ قيمة الشحن المحسوبة.

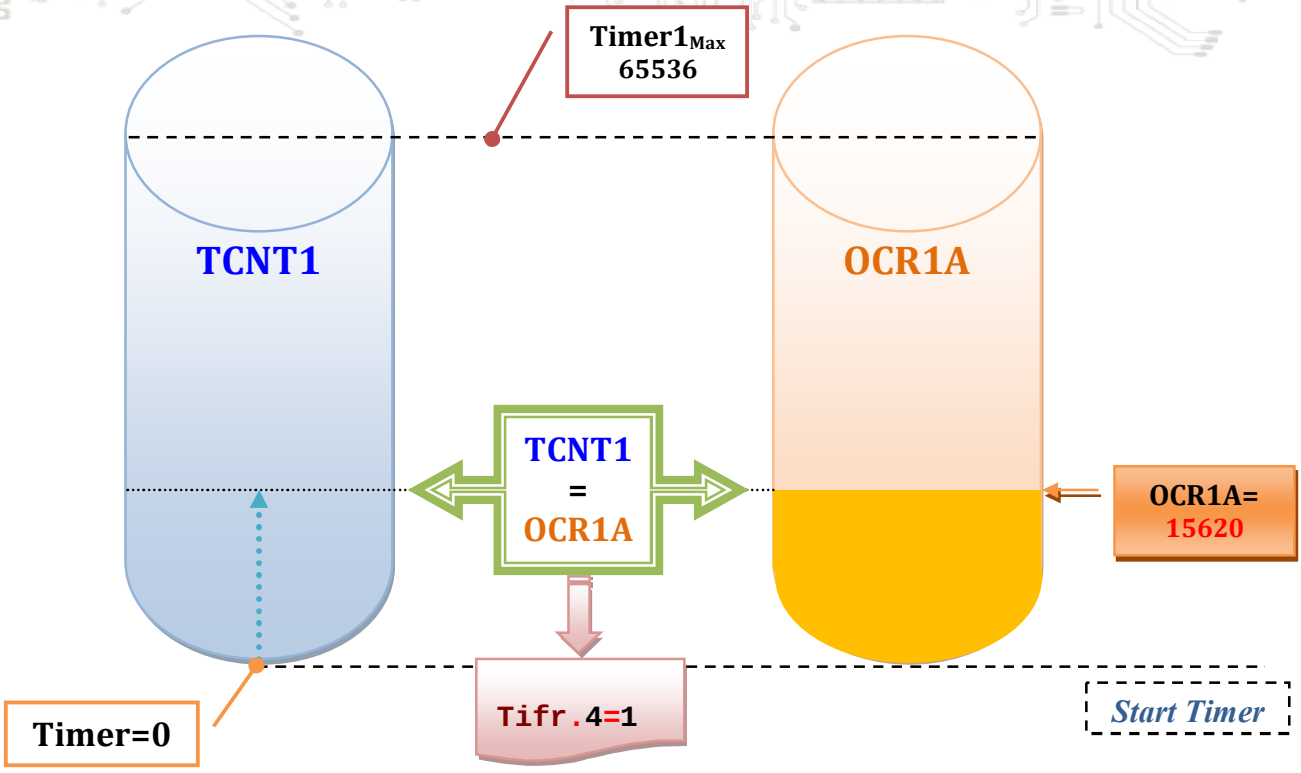
تطبيق: المطلوب حساب قيمة الشحن للمؤقت Timer1 من أجل توليد زمن قدره 1Sec علماً أن تردد عمل المعالج هو 1MHZ والمقسم الترددي 64.

$$OCR1A_{VALUE} = \frac{1MHZ \times 1Sec}{64} = 15620$$

البرنامج جانباً يعتمد على مسجل نظير المقارنة الذي سوف يتم شحنه بالقيمة 15620 من أجل توليد قطار نبضات بفواصل زمنية 1Sec




```
$regfile = "m8def.dat"
$crystal = 1000000
'-----------------------
Config Timer1 = Timer ,
Prescale = 64
Compare1a = 15625
Set Tccr1b.3

Config Portb.0 = Output
Led Alias Portb.0

Start Timer1
'-----------------------
Do
   If Tifr.4 = 1 Then
      Stop Timer1
      Timer1 = 0
      Toggle Led
      Tifr.4 = 1
      Start Timer1
   End If
Loop
End
'-----------------------
```



خامساً: نمط مقاطعة نظير المقارنة (CTC Mode using Interrupt):

إن هذا النمط مشابه تماماً لنمط نمط مسجل نظير المقارنة السابق إلا أنه يعتمد على مقاطعة مسجل النظير عند حصول المساواة، وبالتالي لن ينشغل المعالج بتفحص خانة نظير المساواة (**Tifr.4**) بشكل دائم.

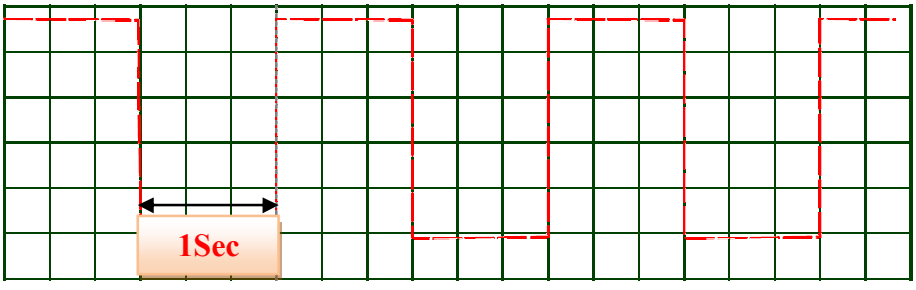
كما أنه عند تفعيل علم مقاطعة نظير المقارنة (OCIEnx) سيتم عندها تصفير خانة نظير المقارنة (**Tifr.4**) بشكل آلي كلما تحققت المقارنة (OCF1A).

تطبيق: المطلوب حساب قيمة الشحن للمؤقت Timer1 من أجل توليد زمن قدره 1Sec علماً أن تردد عمل المعالج هو 1MHZ والمقسم الترددي 64.

$$OCR1A_{VALUE} = \frac{1MHZ \times 1Sec}{64} = 15620$$

البرنامج جانباً يعتمد على مقاطعة مسجل نظير المقارنة الذي سوف يتم شحنه بالقيمة 15620 وسيحدث مقاطعة نظير المقارنة كل 1Sec من أجل توليد قطار نبضات بفواصل زمنية 1Sec.

```
$regfile = "m8def.dat"
$crystal = 1000000
'-----------------------
Config Timer1 = Timer ,
Prescale=64,Clear Timer=1
On Compare1a Timer1_isr
Compare1a = 15625
Enable Compare1a

Config Portb.0 = Output
Led Alias Portb.0

Dim Flag As Bit
Start Timer1
Enable Interrupts
'-----------------------
Do
   If Flag = 1 Then
      Reset Flag
      Toggle Led
      Start Timer1
   End If
Loop
'-----------------------
Timer1_isr:
   Stop Timer1 : Set Flag
Return
```



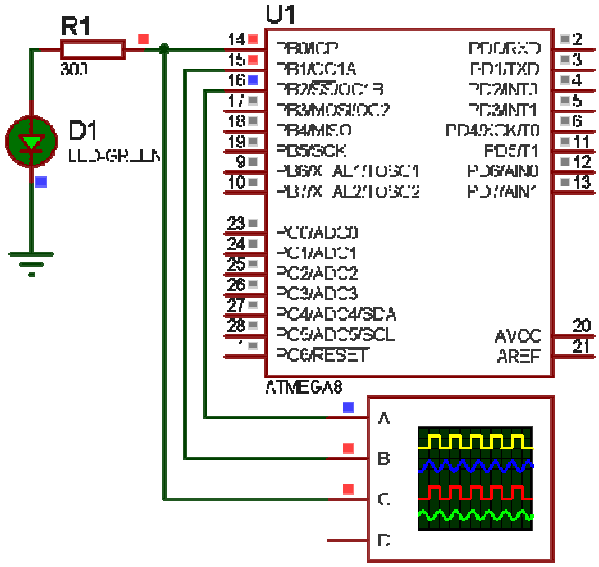
إن البرنامج أعلاه يتضمن التعامل مع المسجل OCR1A فقط؛ يمكن التعامل من المسجل OCR1B أيضاً بنفس الطريقة، البرنامج التالي يعطي مثالاً عن استخدام كلا مسجلي المقارنة من أجل توليد مقاطعات بأزمنة مختلفة.

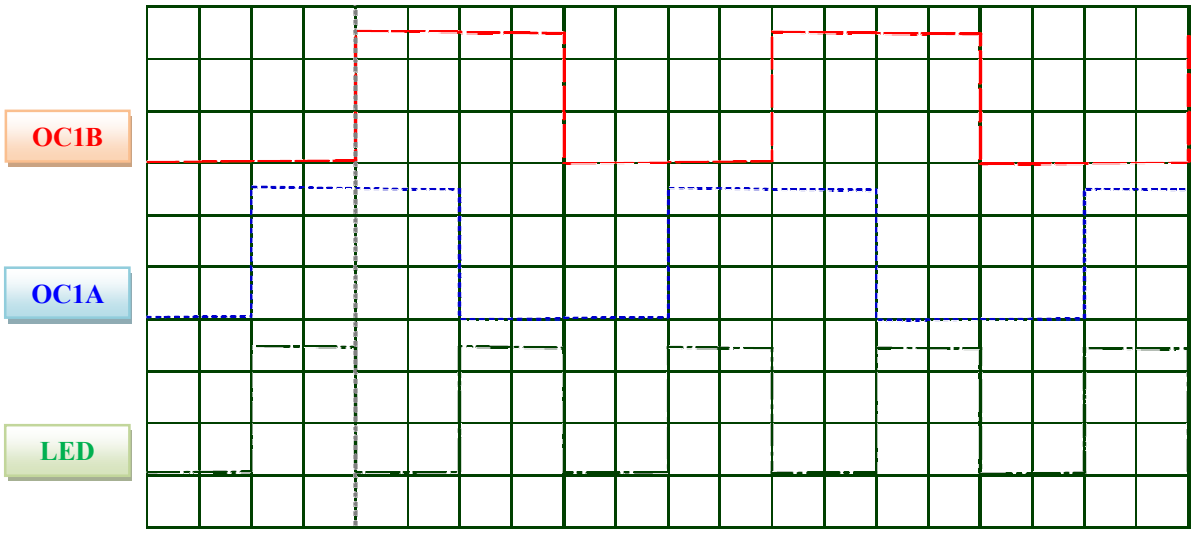
تطبيق: المطلوب حساب قيمة الشحن للمؤقت Timer1 من أجل توليد مقاطعة كل 1Sec على الخرج OC1A ومقاطعة أخرى كل 2Sec على الخرج OC1B وكذلك الخرج على القطب Pinb.0 سوف يتغير عند كل مقاطعة علماً أن تردد عمل المعالج هو 1MHZ والمقسم الترددي 64.

$$OCR1A_{VALUE} = \frac{1MHZ \times 1Sec}{64} = 15620$$

$$OCR1B_{VALUE} = \frac{1MHZ \times 2Sec}{64} = 31250$$

```
$regfile = "m8def.dat"
$crystal = 1000000
'-----------------------
Config Timer1 = Timer ,
Compare A= Toggle, Compare B= Toggle
, Prescale = 64
Compare1a = 15625
Compare1b = 31250
On Compare1a Oc1a
On Compare1b Oc1b
Enable Compare1a
Enable Compare1b

Config Portb.0 = Output
Led Alias Portb.0

Dim Flag1 As Bit , Flag2 As Bit
Start Timer1
Enable Interrupts
'-----------------------
Do
   If Flag1 = 1 Then
      Reset Flag1 : Toggle Led
   End If
   If Flag2 = 1 Then
      Reset Flag2 : Toggle Led
      Timer1 = 0 : Start Timer1
   End If
Loop
End
'-----------------------
Oc1a:
   Set Flag1
Return
'-----------------------
Oc1b:
   Set Flag2 : Stop Timer1
Return
```





سادساً: نمط مقاطعة الطفحان من أجل قيمة شحن أولية (CTC Overflow Mode):

إن هذا النمط يعمل وفق مبدأ طفحان المؤقت وبشكل مشابه لمبدأ مقارن النظير، حيث أنه يتم حساب قيمة الشحن اللازمة لتحقيق زمن معين (مطلوب تحقيقه) ومن ثم يتم طرح هذه القيمة الناتجة من القيمة الأعظمية للمؤقت ($Timerx_{MAX}$).

إن المؤقت سوف يبدأ الزيادة من ناتج الطرح بين القيمة الأعظمية والقيمة المسحوبة للزمن المطلوب، بمعنى آخر إن المؤقت بدلاً من أن يبدأ العد من الصفر إلى قيمة المقارنة كما في نمط "نظير المقارنة"، فإنه سوف يبدأ من قيمة معينة إلى أن يصل إلى القيمة العظمى ويحصل الطفحان وتتولد مقاطعة الطفحان.

يتم حساب قيمة الشحن اللازمة لتوليد زمن مقاطعة طفحان محدد (التي سيبدأ المؤقت منها إلى أن يصل إلى القيمة الأعظمية) بالعلاقة التالية:

$$TCNT_{OC\_Val} = 2^N - \frac{fosc \times T_{required}}{Prescaler}$$

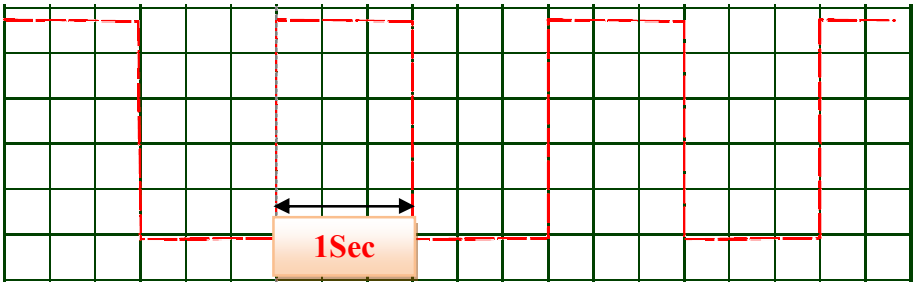
حيث أن: $T_{required}$ قيمة الزمن المطلوب تحقيق المقاطعة من أجله.

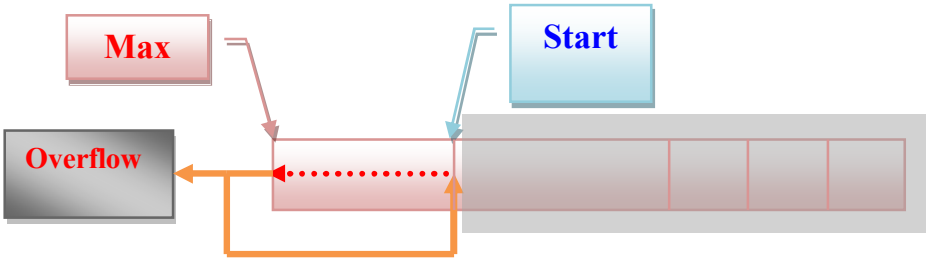
أو يمكن إيجاد الزمن انطلاقاً من القيمة الموجودة في مسجل المؤقت:

$$T_{required} = \frac{Prescaler}{fosc}\left(2^N - TCNT_{OC\_Val}\right)$$

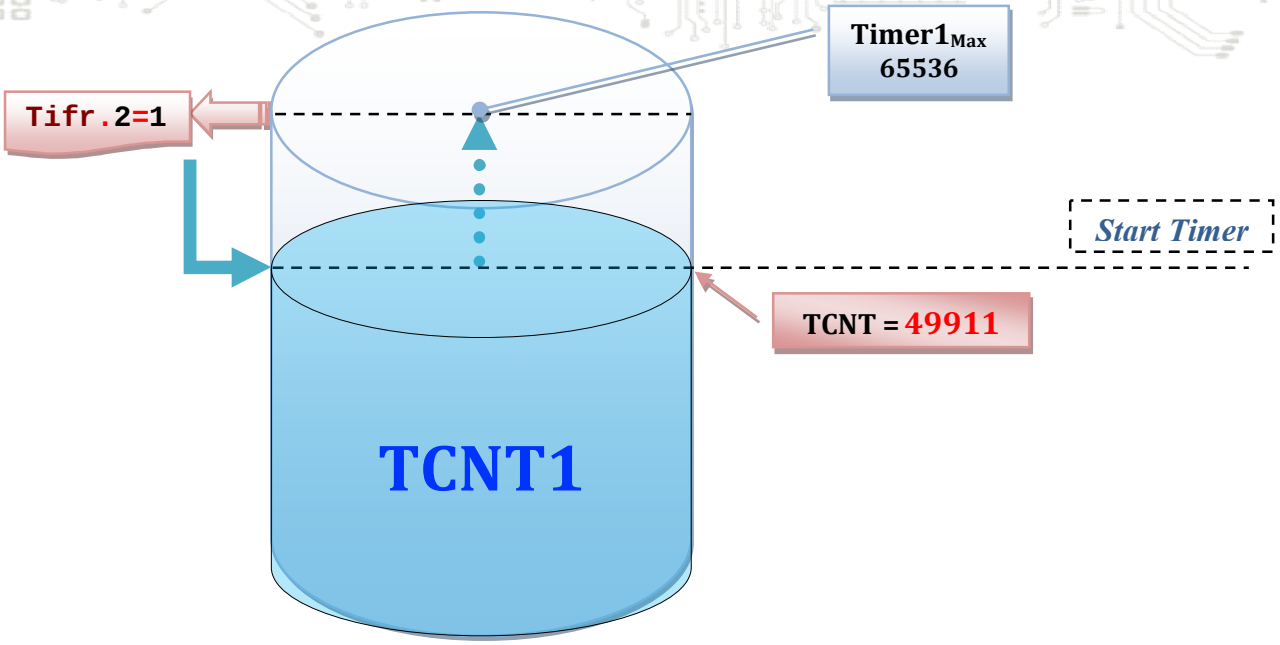
تطبيق: المطلوب حساب قيمة الشحن للمؤقت Timer1 من أجل توليد مقاطعة كل 1Sec علماً أن تردد عمل المعالج هو 1MHZ والمقسم الترددي 64.

$$TCNT_{OC_{Val}} = 65536 - \frac{1MHZ \times 1Sec}{64} = 49911$$






```
$regfile = "m8def.dat"
$crystal = 1000000
'-----------------------
Config Timer1 = Timer ,
Prescale=64
On Timer1 Timer1_isr
Timer1 = 49911
Enable Timer1

Config Portb.0 = Output
Led Alias Portb.0

Dim Flag As Bit
Start Timer1
Enable Interrupts
'-----------------------
Do
   If Flag = 1 Then
      Reset Flag
      Toggle Led
      Start Timer1
   End If
Loop
End
'-----------------------
Timer1_isr:
   Stop Timer1
   Set Flag
   Timer1 = 49911
Return
```

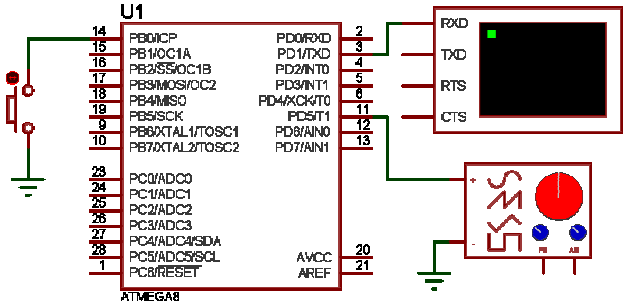


سابعاً: نمط حادثة المسك (Capture Mode):

يملك المؤقت Timer1 ميزة تتعلق بشك فيزيائي بأحد أقطاب المتحكم الخارجية والذي يسمى ICP، حيث إنه عند تطبيق نبضة على هذا القطب (جبهة صاعدة أو هابطة) يتم أخذ صورة (Capture) من محتوى مسجل المؤقت Timer1 (TCCR1) ووضعها في مسجل حادثة المسك Capture1 (ICR1).

تطبيق: يعمل المؤقت1 كعداد من 0 إلى القيمة Max وفي أي لحظة يتم الضغط على المفتاح سوف يتم تخزين القيمة الحالية في مسجل حادثة المسك.

```
$regfile = "m8def.dat"
$crystal = 1000000
$baud = 4800
'----------------------
Config Timer1 = Timer, Capture Edge = Rising
, Noise Cancel = 1 , Prescale = 1024
Start Timer1

Config Pinb.0 = Input
Portb.0 = 1
'----------------------
Do
   Print "Timer: " ; Timer1
   Print "Icr1l: " ; Icr1l
   Print "Icr1H: " ; Icr1h
   Print "Capture: " ; Capture1
   Print "--------------"
Loop
End
```

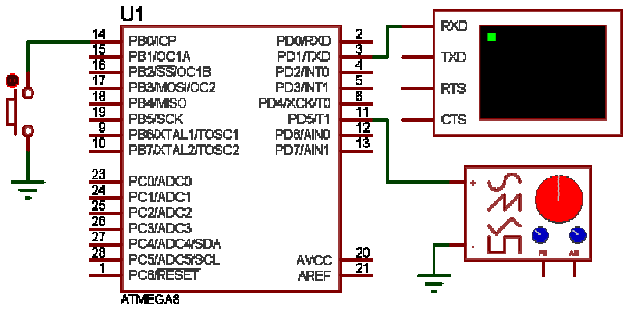


ثامناً: نمط مقاطعة حادثة المسك (Capture Mode Interrupt):

في هذه الحالة وعند تطبيق نبضة على القطب ICP يتم أخذ صورة (نسخة) من محتوى مسجل المؤقت Timer1 (TCCR1) ووضعها في مسجل حادثة المسك Capture1 (ICR1) وتوليد مقاطعة حادثة المسك وسوف يقفز إلى برنامج المقاطعة.

```
$regfile = "m8def.dat"
$crystal = 1000000
$baud = 4800
'----------------------
Config Timer1 = Timer , Capture Edge =
Rising , Noise Cancel = 1 , Prescale = 1024
Enable Interrupts
Enable Capture1
Start Timer1
On Icp1 Timer1_cpa_isr

Config Pinb.0 = Input
Portb.0 = 1
'----------------------
Do
   nop
Loop
End
'----------------------
Timer1_cpa_isr:
   Stop Timer1
   Disable Capture1

   Print "Timer: " ; Timer1
   Print "Captr: " ; Capture1

   Capture1 = 0 : Timer1 = 0
   Enable Capture1
   Start Timer1
Return
```

تطبيق: يعمل المؤقت1 كعداد من 0 إلى القيمة Max وفي أي لحظة يتم الضغط على المفتاح سوف يتم تخزين القيمة الحالية في مسجل حادثة المسك، ويقفز مؤشر البرنامج إلى برنامج خدمة مقاطعة حادثة المسك، وخلال هذا يجب إلغاء تفعيل مقاطعة حادثة المسك وإيقاف المؤقت حتى الانتهاء من معالجة برنامج خدمة المقاطعة.



العدادات (Counters):

في جميع الأنماط أعلاه تم استخدام المؤقت كوحدة منطقية موجودة في داخل المتحكم ولا اتصال لها بالعالم الخارجي (فيزيائياً)، رغم كون وجود أقطاب للمتحكم تدعى T0, T1 (أقطاب دخل للمؤقتات)، إلا أنها تستخدم كمداخل نبضات لدارة العداد.

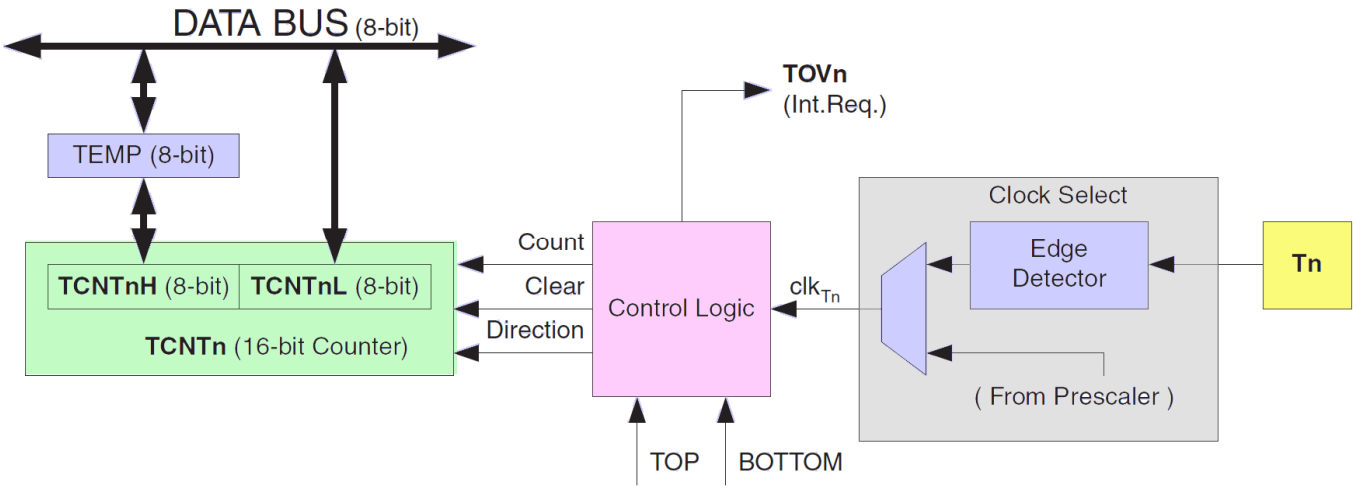
يستخدم نمط العداد من أجل عد نبضات أو أحداث خارجية مطبقة على الأقطاب T0, T1، هذه الأحداث سوف تقوم بقدح دارة المؤقت الداخلي (بدلاً من أن يتم قدحه من دارة الهزاز) ويزداد المؤقت عند كل جبهة واردة (صاعدة أو هابطة أو عند كلاهما)، وعدد النبضات الواردة سوف يكون مخزن في مسجل المؤقت (TCNTx).

إن القطب T0 هو مدخل عداد لدارة المؤقت Timer0 وبالتالي فإن العدد الأعظمي للنبضات على المطبقة على القطب قبل أن تحصل مقاطعة الطفحان هو 255 نبضة أو حدث خارجي.

كما أن القطب T1 هو مدخل عداد لدارة المؤقت Timer1 وبالتالي فإن العدد الأعظمي للنبضات على المطبقة على القطب قبل أن تحصل مقاطعة الطفحان هو 65535 نبضة أو حدث خارجي.

أحد الاستخدامات الهامة للعدادات هو قياس ترددات إشارات خارجية.

ملاحظة هامة: يجب أن لا يكون تردد الإشارة المطبقة على مدخل العداد أكبر من نصف تردد العمل للمعالج وإلا فإن المؤقت لن يستطيع عدد النبضات لأنه عند كل نبضة يقوم المتحكم بتحديد مستوى العينات!

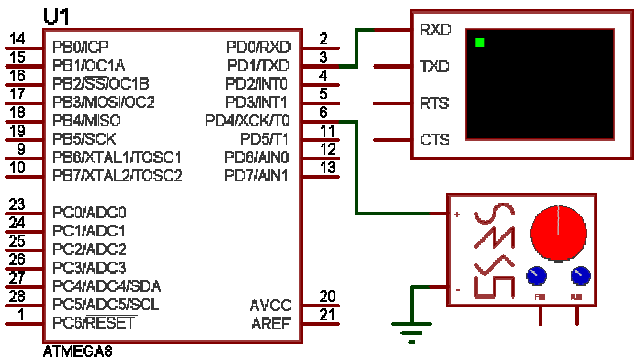


إن أنماط عمل العدادات مشابهة قليلاً إلى أنماط عمال المؤقتات، الأمثلة التالية توضح هذه الأنماط.

العداد Counter1 في النمط العام (Conter0 Normal Mode):

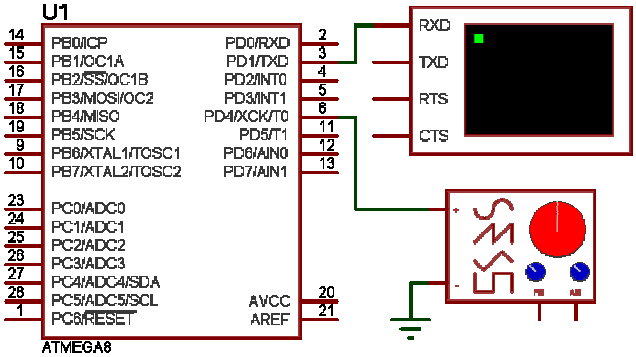
تطبيق(1): البرنامج التالي سيقوم بقراءة النبضات المطبقة على القطب T0 وطباعتها على النافذة التسلسلية، وحالما تصبح قيمة النبضات في مسجل العداد مساوية للقيمة الأعظمية (255) سوف يتوقف البرنامج.

```
$regfile = "m8def.dat"
$crystal = 1000000
$baud = 4800
'----------------------
Config Timer0 = Counter , Edge = Falling
Start Timer0
Tcnt0 = 0
'----------------------
Do
   nop
   Print Counter0
Loop Until Tcnt0 >= 255
End
```



تطبيق(2): البرنامج التالي سيقوم بقراءة النبضات المطبقة على القطب T0 خلال دور 1Sec وطباعتها على النافذة التسلسلية وبالتالي فإن القيمة الناتجة هي التردد الحقيقي المطبق، مع العلم أن التردد الأعظمي الذي يمكن قياسه في هذا البرنامج هو 255HZ.

```
$regfile = "m8def.dat"
$crystal = 1000000
$baud = 4800
'----------------------
Config Timer0 = Counter , Edge = Falling
, Prescale = 1
Stop Timer0
Config Pind.4 = Input
'----------------------
Do
   Counter0 = 0
      Start Counter0
         Waitms 1000
      Stop Counter0
  Print "Counter0: " ; Counter0 ; "HZ"
Loop
```



العداد Counter1 في نمط مقاطعة الطفحان (Overflow Counter0 Interrupt):

تطبيق(3): البرنامج التالي سيقوم بقراءة النبضات المطبقة على القطب T0 خلال دور 1Sec، ولكن لن يكون هناك محدودية لتردد الإشارة المقاسة حيث أنه عندما يصل عدد النبضات الواردة على مدخل العداد إلى 255> سيتولد مقاطعة طفحان العداد وسيتم عد المقاطعات الكلية وحساب التردد الموافق.

```
$regfile = "m8def.dat"
$crystal = 1000000
$baud = 4800
'----------------------
Config Timer0 = Counter , Edge = Falling ,
On Timer0 Counter0_isr
Enable Timer0
Enable Interrupts
'----------------------
Dim Count As Word , Fereq As Long

Do
   Start Timer0
   Waitms 1000
   Stop Timer0
   Fereq = Count * 255
   Fereq = Fereq + Timer0
   Print "Fereq= " ; Fereq
   Timer0 = 0 : Count = 0
Loop
End
'----------------------
Counter0_isr:
   Incr Count
Return
```

العداد Counter1 في نمط مقاطعة الطفحان من أجل قيمة شحن أولية (Overflow Counter0 Interrupt):

تطبيق(4): بنفس المبدأ الذي ناقشناه في المؤقتات فإنه عند شحن العداد بقيمة أولية، فإنه سوف يبدأ العد بدءً من القيمة المشحونة، فإذا كانت القيمة المشحونة على سبيل المثال "100" فإن العداد بعد "155" نبضة سوف يولد مقاطعة الطفحان.

البرنامج جانباً سوف يعطي العداد Counter0 مقاطعة طفحان كلما وردت 10 نبضات.

```
$regfile = "m8def.dat"
$crystal = 1000000
$baud = 4800
'----------------------
Config Timer0 = Counter , Edge = Falling ,
Prescaler = 1
Timer0 = 255 - 10
Enable Timer0
Enable Interrupts
On Timer0 Counter0_isr
'----------------------
Dim Count As Byte
Do
   nop
Loop
End
'----------------------
Counter0_isr:
   Load Timer0 , 10
   Incr Count
   Print "Count=" ; Count
Return
```

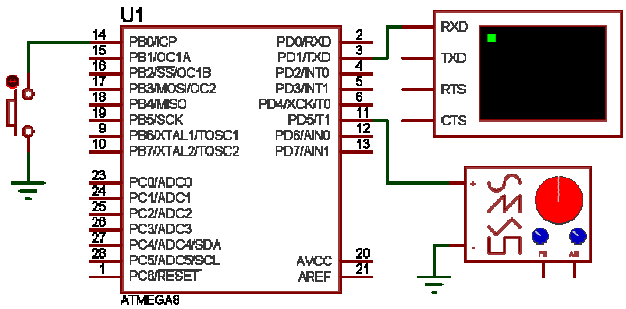
العداد Counter1 في نمط حادثة المسك (Capture Mode):

يمتلك العداد Counter1 ميزات عديدة تشابه ميزات العداد Counter0 وأخرى تفوقها بكثير ، من هذه الميزات:

- إمكانية رفض الضجيج.
- نمط حادثة المسك مع إمكانية تحديد الجبهة التي يحدث عندها المسك.
- نمط نظير مقارنة قيمة العداد.

تطبيق(5): يعمل المؤقت1 كعداد من 0 إلى القيمة Max وفي أي لحظة يتم الضغط على المفتاح سوف يتم تخزين القيمة الحالية في مسجل حادثة المسك.

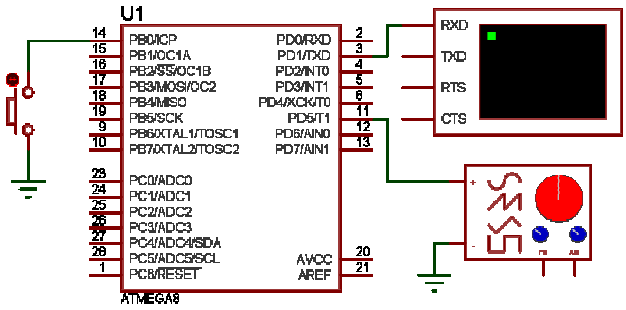
```
$regfile = "m8def.dat"
$crystal = 1000000
$baud = 4800
'----------------------
Config Timer1 = Counter , Prescale = 1024 ,
Edge = Rising , Capture Edge = Falling ,
Noise Cancel = 1
Start Timer1
Config Pinb.0 = Input
Portb.0 = 1
'----------------------
Do
   Print "Timer: " ; Counter1
   Print "Icr1l: " ; Icr1l
   Print "Icr1H: " ; Icr1h
   Print "Capture: " ; Capture1
Loop
End
```



العداد Counter1 في نمط مقاطعة حادثة المسك (Capture Mode Interrupt):

تطبيق(6): يقوم Counter1 بعد النبضات الواردة على القطب T1 وفي أي لحظة يتم فيها الضغط على المفتاح سوف يتم تخزين القيمة الحالية في مسجل حادثة المسك ويقفز مؤشر البرنامج إلى برنامج خدمة مقاطعة حادثة المسك، وخلال هذا يجب إلغاء تفعيل مقاطعة حادثة المسك وإيقاف المؤقت حتى الانتهاء من معالجة برنامج خدمة المقاطعة.

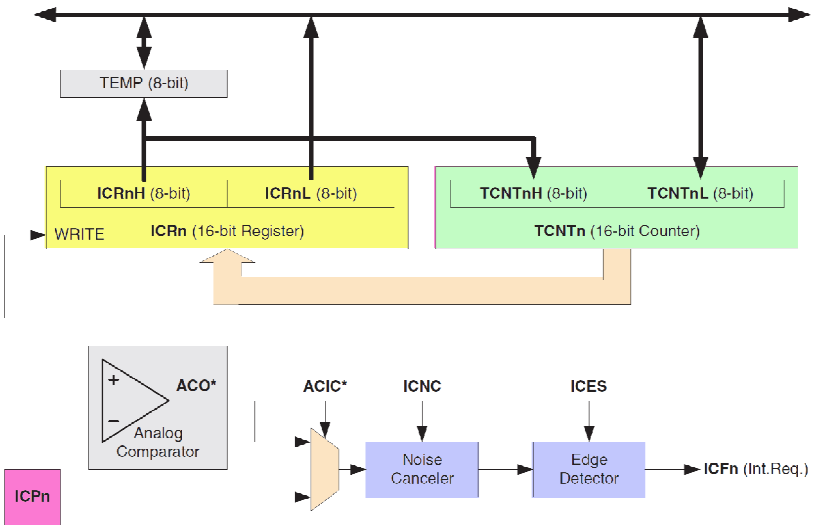
```
$regfile = "m8def.dat"
$crystal = 1000000
$baud = 4800
'----------------------
Config Timer1 = Counter , Edge = Falling ,
Capture Edge = Falling , Noise Cancel = 1 ,
Prescale = 1
Enable Timer1
Enable Capture1
On Capture1 Ctr1_cpa_isr
Enable Interrupts

Config Pinb.0 = Input
Portb.0 = 1
'----------------------
Do
   nop
Loop
End
'----------------------
Ctr1_cpa_isr:
   Stop Counter1
   Disable Capture1

   Print "Counter1: " ; Counter1
   Print "Capture1: " ; Capture1

   Enable Capture1
   Start Counter1
Return
```



العداد Counter1 في نمط نظير المقارنة (Counter1 OC Mode):

تطبيق(7): يقوم Counter1 في هذا النمط بعد نبضات الهزاز الكريستالي بعد تقسيمها عبر Prescaler وعندما تصبح قيمة العداد مساوية إلى القيمة في مسجل نظير المقارنة (Compare1a) يتم تغير حالة القطب OC1A (كل 13uS~)، والتردد الناتج هو 38Khz.

```
$regfile = "m8def.dat"
$crystal = 8000000
'----------------------
Config Timer1 = Counter , Edge = Rising ,
Prescale = 8 , Compare A = Toggle
Compare1a = 102

Config Portb.1 = Output
'----------------------
Do
   nop
Loop
```



مشروع مقياس تردد باستخدام Timer1 & Timer0:

سوف نقوم بإعداد المؤقت Timer0 في نمط مقاطعة الطفحان بحيث يتم توليد مقاطعة طفحان كل 20uS

$$T_{Overflow} = 2^N \frac{Prescaler}{fosc} = 256 \frac{1}{12.8Mhz} = 20\mu Sec$$

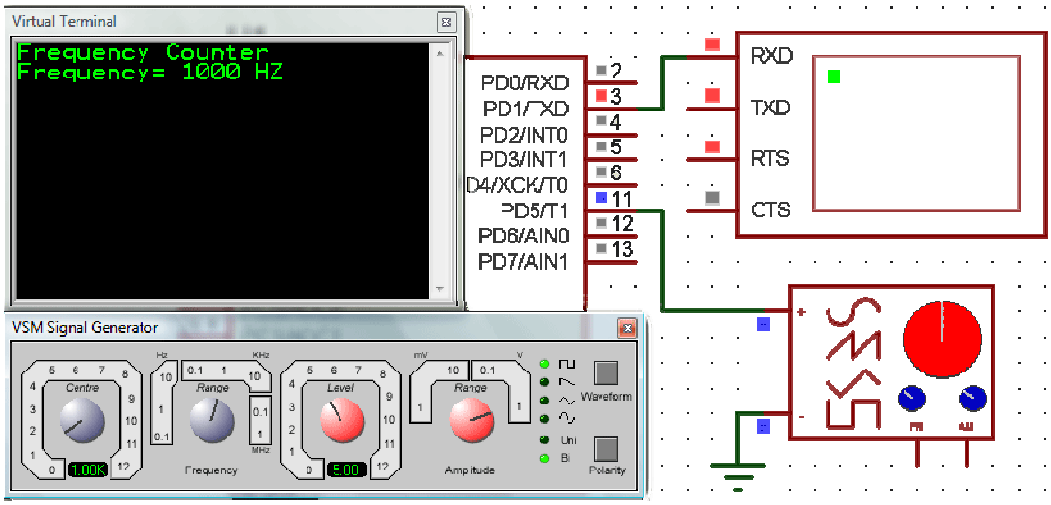
وبالتالي من أجل الحصول على زمن قياس 1Sec نحتاج إلى:

$$INT_{Number} = \frac{Total\ Time}{INT\ Time} = \frac{1Sec}{20\mu Sec} = 500000$$

أي أننا سنحتاج إلى 50000 مقاطعة Timer0 لذلك سنقوم بإعداد متحول يتم إنقاصه من 49999 = Cnt إلى Cnt=0.

سنقوم أيضاً بإعداد المؤقت Timer1 في نمط مقاطعة الطفحان ليعمل كعداد أحداث خارجية على القطب T1 (مدخل الإشارة المراد قياسها) وسنقوم بعد مقاطعات الطفحان لهذا المؤقت خلال زمن القياس (1Sec).

الناتج هو جداء مقاطعات الطفحان للمؤقت Timer1 بدقة المؤقت (65536) مضافاً إليها القيمة الأخيرة المتبقية في مسجل المؤقت ولم يصل إلى طفحان جديد بعد.




```
$regfile = "m8def.dat"
$crystal = 12800000                          ' External 12.8MHZ
$baud = 4800
'----------------------
Config Timer1 = Counter , Edge = Rising , Noise Cancel = 1
Config Timer0 = Timer , Prescale = 1                        '20uS
On Timer0 Timer0_isr
On Timer1 Timer1_isr
Stop Timer0
Stop Counter1

Config Pind.5 = Input
'----------------------
Dim Frequency As Long , Flag As Bit , Overflow As Byte , Cnt As Word
'----------------------
Enable Timer0
Enable Timer1
Enable Interrupts
```

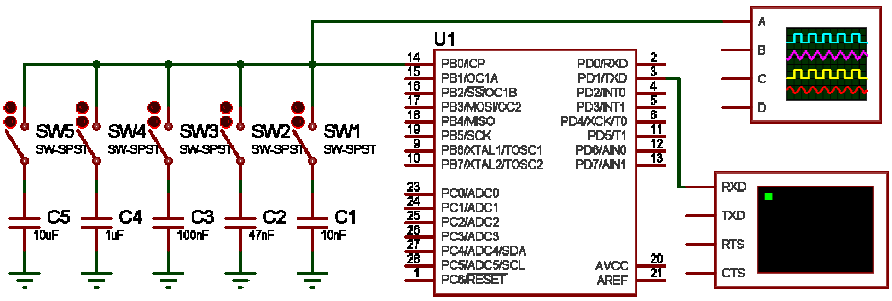
```
Print "Frequency Counter"
Main:
Counter1 = 0 : Overflow = 0 : Cnt = 49999
'----------------------
Start Timer0
Start Counter1
'----------------------
Do
   If Flag = 1 Then
      Reset Flag

      Frequency = Overflow * 65536
      Frequency = Frequency + Counter1

      Print "Frequency= " ; Frequency ; " HZ"
      Goto Main
   End If
Loop
'----------------------

Timer1_isr:
   Incr Overflow
Return
'----------------------
Timer0_isr:
   If Cnt <> 0 Then
      Decr Cnt
   Else
      Stop Counter1
      Stop Timer0
      Set Flag
   End If
Return
'----------------------
```

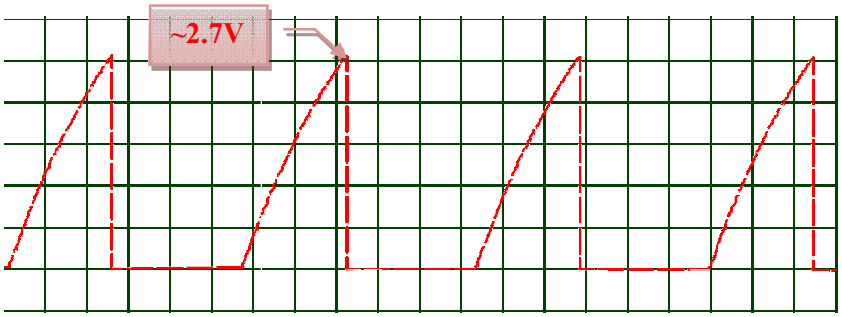
مشروع مقياس سعات (1nF~100uF):

إن المبدأ المستخدم في البرنامج التالي يعتمد على تعريف القطب الموصول معه المكثف كقطب خرج ووضع القيمة "0" على القطب من أجل تفريغ المكثف، وبعد ذلك يتم تحويل القطب إلى قطب دخل ورفع مقاومة الرفع الداخلية للقطب (50KΩ) من أجل شحن المكثف عن طريقها.

يتم قياس الزمن الذي سيستغرقه المكثف ليصل الجهد على طرفيه جهد العمل الأصغري للقطب (2.7V)، ويحسب ثابت الشحن بشكل تقريبي.

```
$regfile = "m8def.dat"
$crystal = 8000000
$baud = 4800
'----------------------
Dim T As Long , C As Single
'----------------------
Do
   T = 0
   Config Pinb.0 = Output
   Portb.0 = 0  'Low Z, 0V

   Waitms 1000

   Config Pinb.0 = Input
   Portb.0 = 1

   Do
      T = T + 1
   Loop Until Pinb.0 = 1

   C = T * 0.0866
   C = Round(c)
   Print C ; " nF"
Loop
End
```





إن الطريقة السابقة لا تعطي الدقة الكافية في القياس لأن الزمن المقاس يعتمد على حلقة شرطية، كما أن القياس سيتوقف عند جهد شحن 2.7V.

من أجل التغلب على المشاكل السابقة، يتم استخدام المؤقت كعداد لنبضات الهزاز الكريستالي ويتم استخدام المبدل التشابهي الرقمي لقياس جهد الشحن.