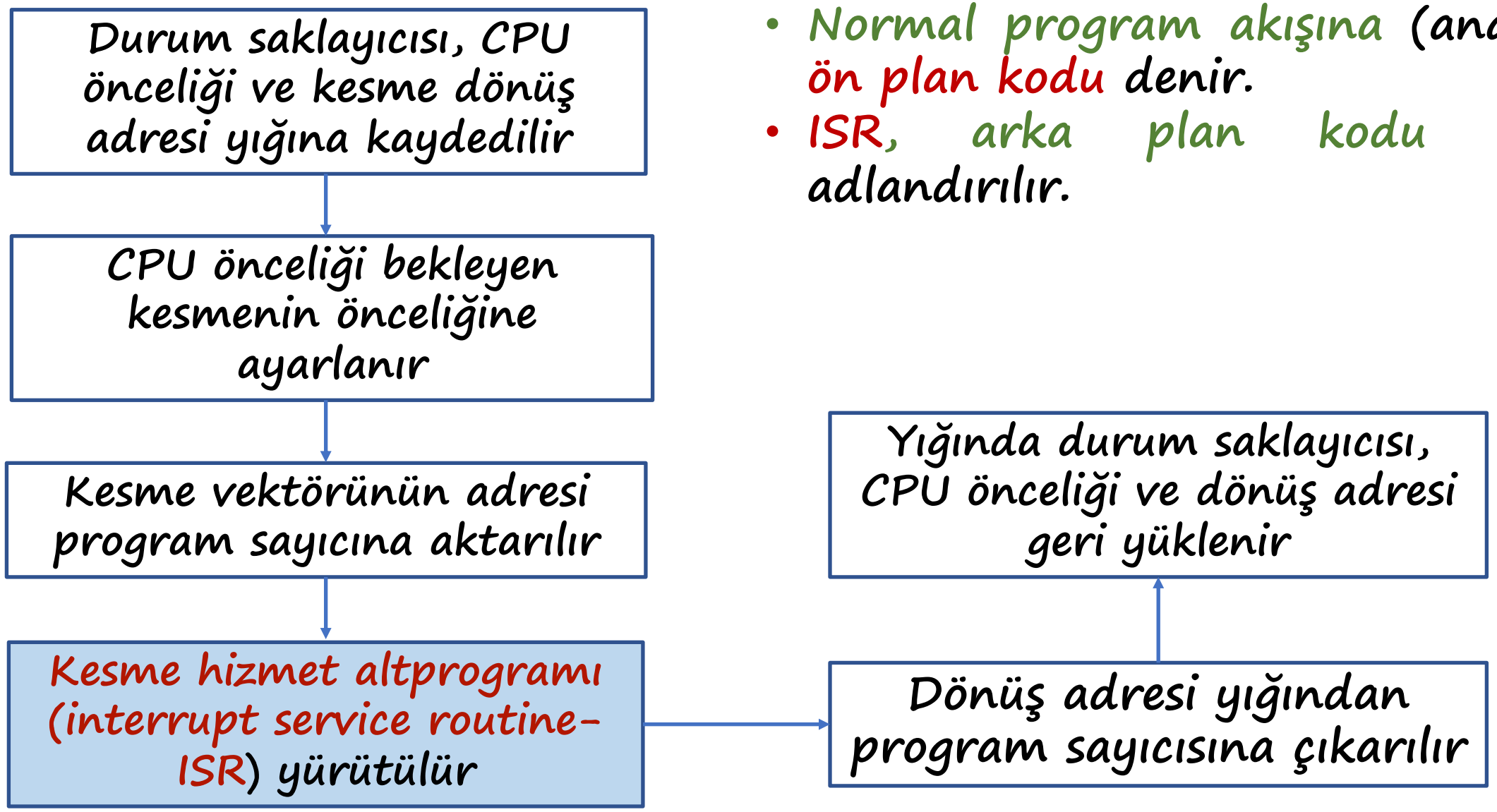
Bölüm 8
Kesme ve Zamanlayıcı

Doç. Dr. Fatih Evran
Elektrik ve Elektronik Mühendisliği
Ofis: 421 Mühendislik Binası
Düzce Üniversitei
Email: fatihevran@duzce.edu.tr

Yoklama Giriş/Çıkış ve Kesmeye Dayalı Giriş/Çıkış

- Yoklama Giriş/Çıkış: İşlemci, veri aktarımının hazır olup olmadığını görmek için Giriş/Çıkış cihazlarını sürekli kontrol eder
  A. Yetersiz. İşlemci Giriş/Çıkış cihazlarını kontrol etmek için zaman harcıyor
  B. Ya çok sık kontrol ediyor ya da özellikle rastgele olaylar için yeterince sık değil
- Kesmeye (Interrupt) dayalı Giriş/Çıkış: Giriş/Çıkış cihazı, veri aktarımı için hazır olduğunda işlemciyi kesme yöntemi ile bildirir.
  A. Çok verimlidir. İşlemci, son veri aktarımının tamamlanmasını beklerken başka görevler yapıyor olabilir
  B. Modern μC' deki tüm Giriş/Çıkış işlemleri kesmeye dayalıdır

PIC Kesme Çalışması



- Normal program akışına (ana akış) ön plan kodu denir.
- ISR, arka plan kodu olarak adlandırılır.

Kesme Vektör Tablosu (Interrupt Vector Table) (1)

- Kesme vektör tablosu (IVT), her kesme kaynağı için ISR' nin başlangıç adresini içerir

Kesme Adresi	Kesme Numarası	PIC24 Derleyici İsmi	Kesme İsmi
0x000006	1	_OscillatorFail	Oscillator Failure
0x000008	2	_AddressError	Address Error
0x00000A	3	_StackError	Stack Error
0x00000C	4	_Math Error	Math Error
0x000014	8	_INT0Interrupt	INT0-External Interrupt
0x000016	9	_IC1Interrupt	IC1- Input Capture 1
0x000018	10	_OC1Interrupt	OC1- Output Compare 1
0x00001A	11	_T1Interrupt	T1-Timer1 Expired
0x00001E	13	_IC2Interrupt	IC2- Input Capture 2
0x000020	14	_OC2Interrupt	OC2- Output Compare 2
0x000022	15	_T2Interrupt	T2-Timer2 Expired
0x000024	16	_T3Interrupt	T3-Timer3 Expired

Kesme Vektör Tablosu (2)

Kesme Adresi	Kesme Num	PIC24 Derleyici İsmi	Kesme İsmi
0x000026	17	_SPIErrInterrupt	SP11E-SPI1 Error
0x000028	18	_SPIInterrupt	Address Error
0x00002A	19	_U1RXInterrupt	U1RX-UART1 Recevier
0x00002C	20	_U1TXInterrupt	U1TX-UART1 Transmitter
0x00002E	21	_ADCInterrupt	ADC1-ADC1 convert done
0x000034	24	_SI2C1Interrupt	SI2C1-I2C1 Slave Events
0x000036	25	_MI2CInterrupt	MI2C1-I2C1 Master Events
0x00003A	27	_CNInterrupt	Change Notification Interrupt
0x00003C	28	_INT1Interrupt	INT1-External Interrupt
0x000040	30	_IC7Interrupt	IC7- Input Capture 7
0x000042	31	_IC8Interrupt	IC8- Input Capture 8
0x00004E	37	_INT2Interrupt	INT2-External Interrupt
0x000096	73	_U1ErrInterrupt	U1E-UART1 Error

Kesme Öncelikleri

- Bir kesmeye 0 ile 7 arasında bir öncelik atanabilir.
- Normal bir komutun (ana akıştaki komut) yürütme önceliği 0' dır.
- Bir kesme, normal bir yürütmeyi kesebilmek için 0' dan daha yüksek bir önceliğe sahip OLMALIDIR.
- Bir kesmeye sıfır (0) önceliğin atanması kesmeyi maskeler yani devre dışı bırakır.
- Daha yüksek önceliğe sahip bir kesme, halihazırda yürütülen daha düşük öncelikli bir kesmeyi kesebilir.
- Aynı önceliğe sahip eşzamanlı kesmeler meydana gelirse, daha düşük vektör numaralı kesme daha yüksek doğal önceliğe sahiptir.
- Örneğin, INT1 ve INT2' nin vektör (kesme) numarası sırasıyla 28 ve 37' dir. INT1 ve INT2 için aynı öncelik verilirse, PIC aynı anda iki kesme meydana geldiğinde önce INT1' e yanıt verir.

Bir Kesmeyi Aktif Hale Getirmek

- Bayrak biti, kesme oluştuğunda otomatik olarak bir (1) olmaktadır.
- Öncelik bitleri (priority bits), kesme önceliğini (interrupt priority) ayarlar (0 ila 7).
- Bir kesmeye yanıt vermek için etkinleştirme yani izin biti (enable bit) bir (1) olarak ayarlanmalıdır.
- PIC, aşağıdaki üç koşul aynı anda karşılandığında kesmeyi yani ISR' yi yürütür :
  - A. Kesme oluşur (kesme bayrağı 1 olur)
  - B. Kesme önceliği > 0
  - C. İzin biti=1
- Reset sonrasında, tüm öncelik bitleri ve izin bitleri 0 olmaktadır, bu nedenle kesme ISR' lerinin yürütülmesi devre dışı bırakılır.

Kesme Bayrağının Temizlenmesi

- PIC, kesme bayrağını otomatik olarak temizleyemez. Bu nedenle, ISR' de bayrak bitini komut yoluyla temizlemeniz gerekmektedir; aksi takdirde PIC, ISR' yi yürüten sonsuz bir döngüde takılıp kalır.

Örnek: ISR fonksiyonunda INT1 kesme bayrağının temizlenmesi

```
void _ISRFAST _INT1Interrupt ( void )
        {
                _INT1IF = 0; // Clear the INT1 interrupt flag
        }
```

Tuzaklar (Traps) ve Kesmeler

- Tuzak, özel bir kesme türüdür, her zaman etkindir, normal kesmelerden daha yüksek önceliğe (10 – 14) sahiptir.
- Hard trap: CPU, tuzağın oluştuğu komuttan sonra durur
- Soft trap: Tuzak örneklenirken ve onaylanırken CPU komutları yürütmeye devam eder

Trap	Category	Öncelik
Oscillator failure	Hard	14
Address error	Hard	13
Stack error	Soft	12
Math error	Soft	11
DMAC error	Soft	10

ISR Ek Yükü (1)

- Ientry: ISR' ye giriş için toplam komut çevrim sayısı

  ↓

  PIC' de komut çevrim sayısı 4' tür

- Ibody: ISR fonksiyonunda gerçekleştirilen komut çevrim sayısı

  ↓

  Koda bağlıdır

- Iexit: ISR' den çıkış için gerekli komut sayısı

  ↓

  PIC' de komut çevrim sayısı 3' tür

- Fisr: ISR' nin tetiklenme frekansı (saniyede kaç kez).
- Tisr: ISR tetikleme periyodu = 1/Fisr. Örneğin, bir ISR 1 KHz' de tetiklenirse, Tisr 1 ms' dir.

ISR Ek Yükü (2)

- Bir ISR tarafından gerçekleştirilen CPU süresinin yüzdesini hesaplayın

$$Percentage\ time = \frac{[(lentry + lbody + lexit) \times F_{isr}]}{F_{osc}} \times 100\%$$

- Örnek: Saat frekansının Fosc = 40 MHz, Ibody = 50 komut çevrimi olduğunu varsayalım. Tisr = 1 ms ise, bu ISR tarafından kullanılan CPU süresinin yüzdesini hesaplayın.

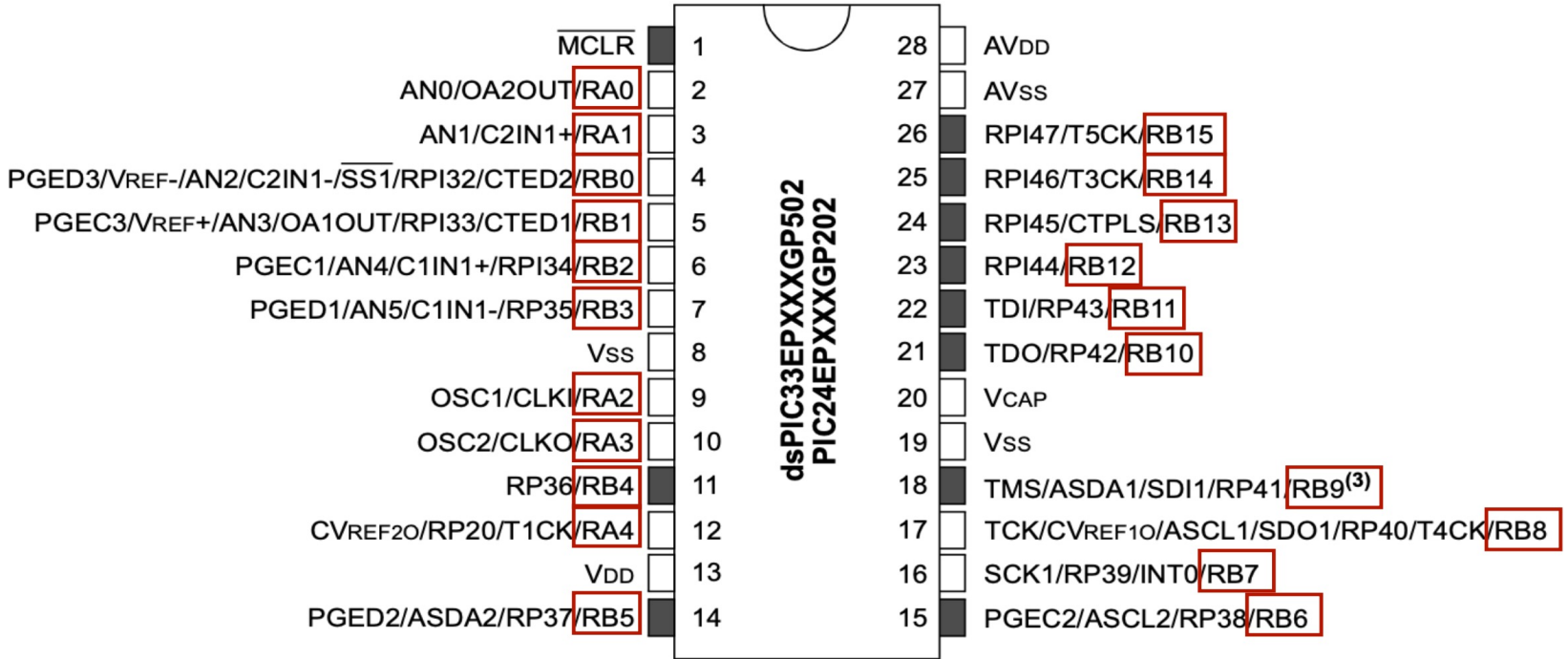
  Ientry = 4 komut çevrimi, Iexit = 3 komut çevrimi, $F_{isr} = 1/T_{isr} = 1/1ms = 1000$ Hz

$$Percentage\ time = \frac{[(lentry + lbody + lexit) \times F_{isr}]}{F_{osc}} \times 100\%$$

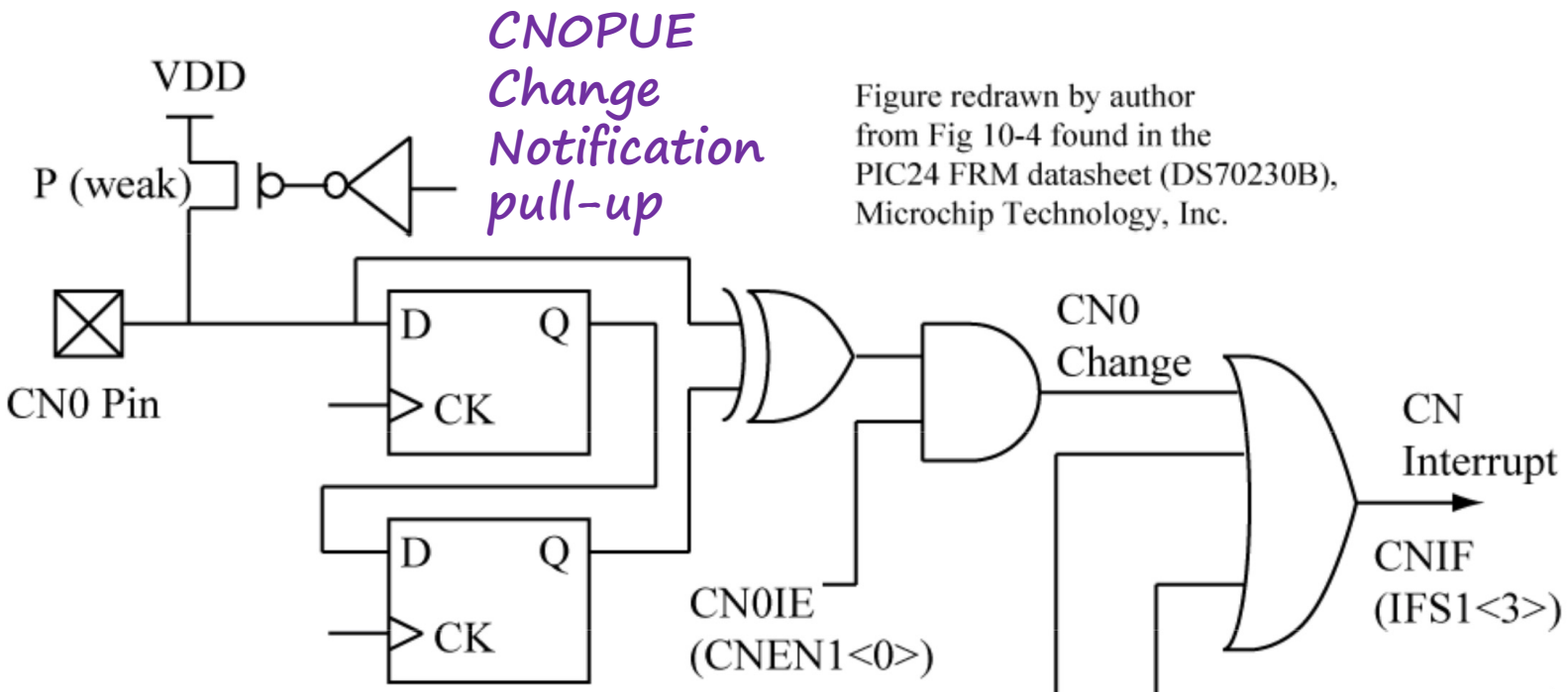
$$= \frac{[(4 + 50 + 3) \times 1000]}{40\ MHz} \times 100\% = 0.14\%$$

Input Change Notification (CN) Kesmeleri

- PIC üzerindeki bütün GPIO pinleri CN kesmesini destekler
  A. PORTA: RA0 ila RA4 arasında olan pinler
  B. PORTB: RB0 ila RB15 arasında olan pinler
- Pin gerilimi düşükten yükseğe veya yüksekten düşüğe değiştiğinde CN kesmesi tetiklenir



Change Notification Kesmeleri (devam)



Etkinleştirildiğinde, bir pin üzerinde bir değişiklik meydana geldiğinde bir kesmeyi tetikler

CN Kesme Örneği

- RB13 pinine bir push-pull anahtar ve RB14 pinine bir LED bağlayın. Anahtara basıldığında veya anahtar bırakıldığında, uykudaki işlemciyi uyandırın.

```
/************CN ISR ***********/
void _ISRFAST _CNInterrupt (void)
{
// Clear the CN interrupt flag in ISR
_CNIF = 0;
}
```

```
/*Configure RB13 to detect the status of push button */
void CONFIG_SW ()
{
CONFIG_RB13_AS_DIG_INPUT ();
ENABLE_RB13_PULLUP ();
ENABLE_RB13_CN_INTERRUPT ();
DELAY_US(1);
}
```

```
#include "pic24_all.h"
#define LED _LATB14
#define CONFIG_LED() CONFIG_RB14_AS_DIG_OUTPUT ()
```

```
void main (void) {
CONFIG_SW (); // Configure RB13
CONFIG_LED (); // Configure RB14
_CNIF = 0;     // Clear interrupt flag
_CNIP = 2;     // Set interrupt priority
_CNIE = 1; // Enable CN interrupt
LED=0;
SLEEP ();
LED=1;
while (1) {}
```

Makroları Yeniden Eşleme (1)

- Dahili modüller için bazı giriş/çıkışlar, kullanılacaksa belirli pinlere eşlenmelidir.

Örnek 1: INT0, pin16' ya atanmıştır, ancak INT1 ve INT2 girişleri herhangi bir pine atanmamıştır. Bu nedenle, INT1 ve INT2 'nin bunları kullanmak için önce bir pine eşlenmesi gerekir. Aşağıdaki ifade, RB6 pininde bulunan pin RP38' i INT1 giriş fonksiyonuna yönlendirir.

```
_INT1R = 38;        // INT1' i RP38/RB6 pinine atayın.
```

Kod, bunu gerçekleştirmek için bir makro kullanır (lib\include\pic24_ports.h yolunda tanımlanmıştır):

```
CONFIG_INT1_TO_RP(RB6_RP);
```

RB6_RP değeri, bu durumda RP38 olan "RB6 pininde bulunan yeniden eşlenebilir pin" anlamına gelir.
Yeniden eşlenebilir bir girişi doğru şekilde kullanmak için, CONFIG_RXy_AS_DIG_INPUT() kullanarak o pini bir dijital giriş olarak yapılandırdığınızdan emin olun.

Makroları Yeniden Eşleme (2)

Örnek 2:

Aşağıdaki ifade, U1TX çıkışını RB11 ile paylaşılan RP43 pinine yönlendirir:

```
_RP43R = 1; //U1TX' i RP43/RB11 pinine atayınız
```

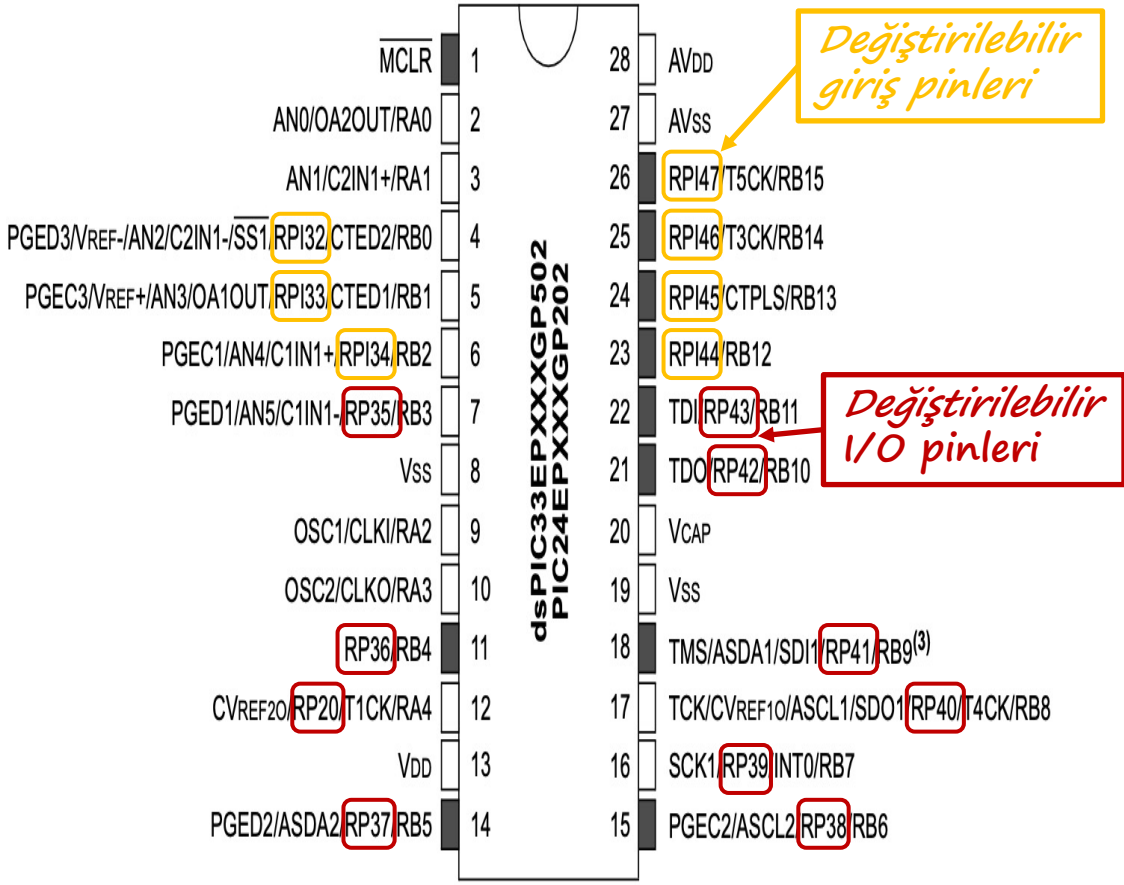
Kod, bunu gerçekleştirmek için bir makro kullanır (lib\include\pic24_ports.h yolunda tanımlanmıştır):

```
CONFIG_U1TX_TO_RP(RB11_RP);
```

Yeniden eşlenebilir bir çıkışı doğru şekilde kullanmak için, CONFIG_Rxy_AS_DIG_OUTPUT() kullanarak o pini bir dijital çıkış olarak yapılandırdığınızdan emin olun.

dsPIC33EP128GP502 için Seçilmiş Yeniden Eşlenebilir Girişler

Giriş İsmi	Pin İsmi	RPn' ye Örnek Atama
External Interrupt 1	INT1	_INT1R = n;
External Interrupt 2	INT2	_INT2R = n;
Timer2 External Clock	T2CK	_T2CKR = n;
Input Capture 1	IC1	_IC1R = n;
Input Capture 2	IC2	_IC2R = n;
Input Capture 3	IC3	_IC3R = n;
Input Capture 4	IC4	_IC4R = n;
Output Compare Fault A	OCFA	_OCFAR = n;
UART1 Receive	U1RX	_U1RXR = n;
UART2 Receive	U2RX	_U2RXR = n;
SPI2 Data Input	SDI2	_SDI2R = n;
SPI2 Clock Input	SCK2	_SCK2R = n;
SPI1 Slave Select Input	SS1	_SS2R = n;



dsPIC33EP128GP502 için Seçilmiş Yeniden Eşlenebilir Çıkışlar

Çıkış İsmi	Pin İsmi	RPnR<4:0>	RPn' ye Örnek Atama
Default Port Pin	NULL	0	_RPnR = 0;
UART1 Transmit	U1TX	1	_RPnR = 1;
UART2 Transmit	U2TX	3	_RPnR = 3;
SPI2 Data Output	SDO2	8	_RPnR = 8;
SPI2 Clock Output	SCK2	9	_RPnR = 9;
SPI2 Slave Select Output	SS2	10	_RPnR = 10;
Output Capture 1	OC1	16	_RPnR = 16;
Output Capture 2	OC2	17	_RPnR = 17;
Output Capture 3	OC3	18	_RPnR = 18;
Output Capture 4	OC4	19	_RPnR = 19;

Yeniden eşlenebilir çıkışlar yalnızca RPn pinlerini kullanabilir, ancak RPIn pinlerini kullanamaz, çünkü bunlar yalnızca yeniden eşlenebilir girişleri destekler. Her yeniden eşlenebilir pin RPn' ye, RPnR adlı 5 bitlik bir alan atanır; bu bit alanının değeri, kendisine eşlenen çıkış görevini kontrol eder.

INT Kesmesi

- PIC birden çok INT kesme kaynağına (INT0, INT1 ve INT2) sahiptir.
- INT kesmesi, pin gerilimi yüksekten düşüğe (düşen kenar) veya düşükten yükseğe (yükselen kenar) değiştiğinde tetiklenecektir.
  A. INTxEP bit = 1 —> INTx düşen kenarda tetiklenecek
  B. INTxEP bit = 0 —> INTx yükselen kenarda tetiklenecek
- İşlemcide INT0 16 numaralı pine atanmıştır fakat INT1 ve INT2 girişleri herhangi bir pine atanmamıştır. Bu nedenle, INT1 ve INT2'nin bunları kullanmak için önce bir pine eşlenmesi gerekir.

Önemli !!!

CN kesmesi hem yükselen kenarı hem de düşen kenarı algılar.
INT kesmesi, yükselen kenarı veya düşen kenarı algılar

INT Kesme Örneği

- RB13 pinine bir push-pull anahtar ve RB14 pinine bir LED bağlayın. LED' in durumunu anahtar ile değiştiriniz
  - A. Anahtara basınız-> Pin gerilimi yüksekten düşüğe değişir (düşen kenar)
  - B. Aanhtarı bırakınız-> > Pin gerilimi düşükten yükseğe değişir (yükselen kenar kenar)

```
void CONFIG_INT1_INT()
{
  // Configure INT1 interrupt.
  CONFIG_INT1_TO_RP(RB13_RP);
  _INT1IF = 0;     //Clear the interrupt flag
  _INT1IP = 2;     //Choose a priority
  _INT1EP = 1;     //negative edge triggerred
  _INT1IE = 1;     //enable INT1 interrupt
}
```

```
#include "pic24_all.h"
#define LED _LATB14
#define CONFIG_LED() CONFIG_RB14_AS_DIG_OUTPUT ()
```

```
/*Configure RB13 to detect the status of
push button */
ivoid config_pb()
{
CONFIG_RB13_AS_DIG_INPUT ();
ENABLE_RB13_PULLUP ();
DELAY_US(1);
}
```

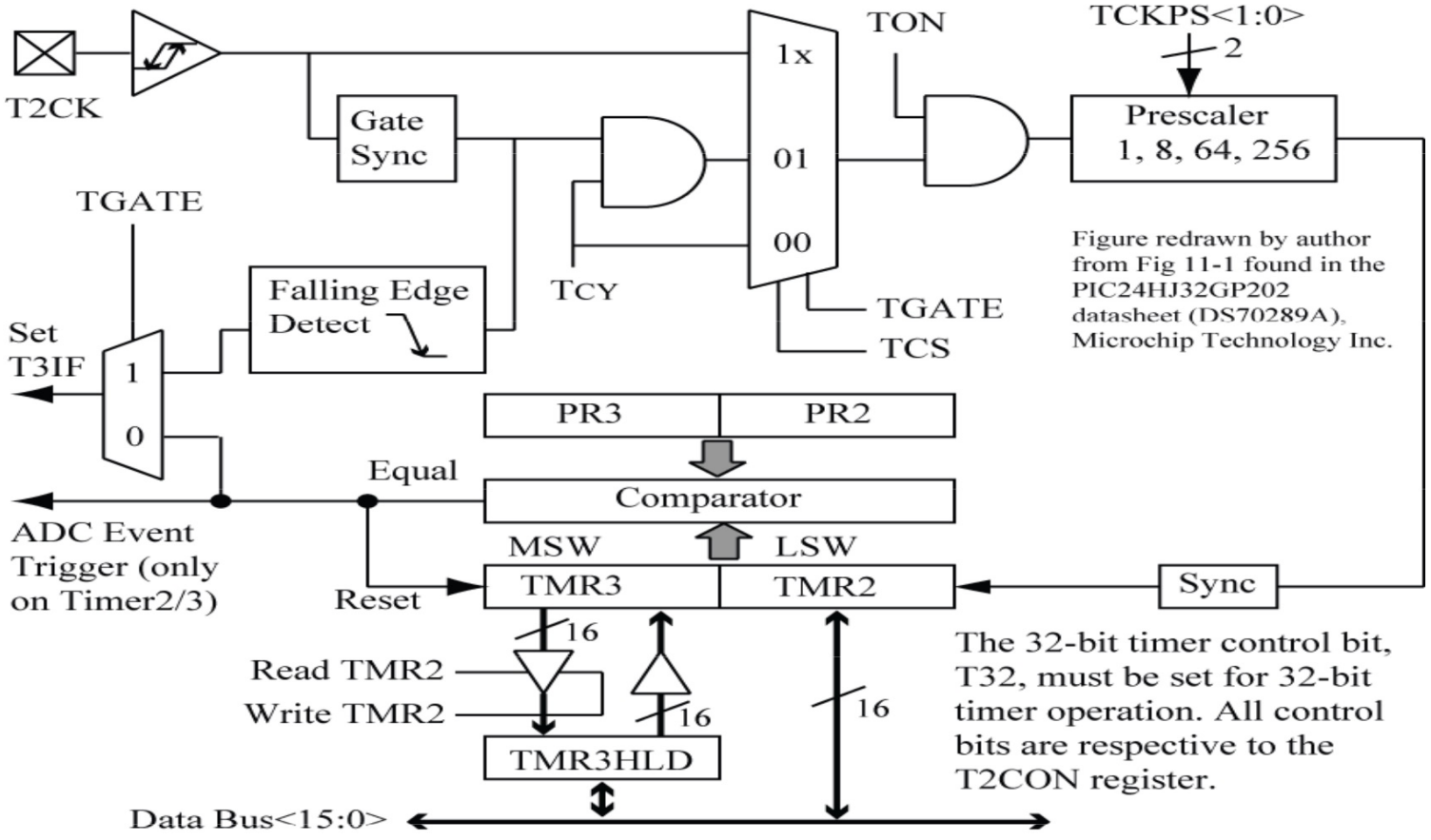
```
/*************INT1 ISR ***********/
void _ISR _INT1Interrupt (void)
{
// Clear the CN interrupt flag in ISR
_INT1IF = 0;
LED=!LED;}
```

```
void main (void) {
CONFIG_LED (); // Configure RB14
config_pb(); // Configure RB13
CONFIG_INT1_INT();
LED=0;
while (1){}
```

Donanım Zamanlayıcıları (1)

- Yazılım zamanlayıcısı: Programda biraz gecikme uygulamak için for döngüsü ve while döngüsü gibi "gecikme prosedürünü" kullanır
  A. Doğru değil: Geciktirme prosedürünün tam olarak ne kadar süredir devam ettiğini bilmiyoruz
  B. Verimsiz: İşlemci, gecikme prosedürünü yürütmek için zaman harcıyor
- Donanım zamanlayıcısı: Yapılandırılabilir bir sayıcı
  A. Doğru: Sistem saatine dayalı olacak çok kesin bir zaman gecikmesi oluşturabilir
  B. Verimli: İşlemci, donanım zamanlayıcısından zaman aşımı kesmesini beklerken başka görevler yapıyor olabilir

32-bit Timer (Timer2/3)



32 bitlik zamanlayıcıda Timer2 ve Timer3 birleştirilmiştir.

Timer2 sayıcının LSW kısmını tutarken, Timer3 sayıcının MSW kısmını tutar.

Zamanlayıcı kontrol bitleri T2CON saklayıcısında bulunmaktadır.

T3IF bayrağı ise Zaman aşımı için kullanılır.

Periyod saklayıcısı PR3:PR2 olmaktadır

Timer2

- PIC33, birkaç Zamanlayıcı modülü sunar. Derste örnek olarak Timer2' yi kullanacağız
- 16 bitlik saklayıcı TMR2' deki değerden 16 bitlik saklayıcı olan PR2' deki değere kadar sayar

Genellikle TMR2 = 0

- Timer2 zaman aşımı (timeout) olayı gerçekleştiğinde T2IF kesme bayrağı bir (1) olmaktadır
- Timer2, ön bölücü (prescaler) değerini yapılandırmak için 2 bit (TCKPS<1:0>) kullanır

TCKPS<1:0> = 00, 01, 11, 10 —> Prescaler = 1, 8, 64, 256

- İstenilen gecikme = Saat periyodu × Prescaler × (PR2 + 1)

"+ 1" çünkü zamanlayıcı 0' dan sayar

Timer2 Periyodu

İstenilen gecikme = Saat periyodu × Prescaler × (PR2 + 1)

$$PR2 = \frac{\text{İstenilen gecikme}}{\text{Saat periyodu} \times \text{Prescaler}} - 1$$

Örnek: Saat frekansı (F_{CY}) = 40 MHz olsun, PR2' yi 15 ms gecikme oluşturacak şekilde ayarlayın.

Çözüm: İstenilen gecikme=15 ms

Saat frekansı = 40 MHz —> Saat periyodu = $\frac{1}{40\ MHz} = 0.025\ \mu s$

A. prescaler = 256 -> PR2 = $\frac{15\ ms}{0.025\ \mu s \times 256} - 1 = 2343$

B. prescaler = 64 -> PR2 = $\frac{15\ ms}{0.025\ \mu s \times 64} - 1 = 9375$

Timer2 Kontrol Saklayıcısı (T2CON)

R/W-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
TON	UI	TSIDL	UI	UI	UI	UI	UI
15	14	13	12	11	10	9	8
U-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	U-0
UI	TGATE	TCKPS<1:0>		T32	UI	TCS	UI
7	6	5	4	3	2	1	0

Bit 15: TON: Timer2 On Bit

When T32 = 1:
1 = Starts 32-bit Timer2/3
0 = Stops 32-bit Timer2/3

When T32 = 0:
1 = Starts 16-bit Timer2
0 = Stops 16-bit Timer2

Bit 13: TSIDL: Stop in Idle Mode Bit
1 = Discontinue module operation device enters Idle mode
0 = Continue module operation in Idle mode

Bit 6: TGATE: Timer2 Gated Time Accumulation Enable

When TCS = 1:
This bit is ignored.

When TCS = 0:
1 = Gated time accumulation enabled
0 = Gated time accumulation disabled

Bit 5-4: TCKPS<1:0>: Timer2 Input Clock Prescale Select Bits
11 = 1:256 , 10 = 1:64, 01 = 1:8, 00 = 1:1

Bit 3: T32: 32-bit Timer Mode Select bit[1]
1 = Timer2 and Timer3 form a single 32-bit timer
0 = Timer2 and Timer3 act as two 16-bit timers

Bit 1: TCS: Timer2 Clock Source Select bit
1 = External clock from pin T2CK (on the rising edge)
0 = Internal clock (FCY)

Legend:
R = Readable bit
-n = Value at POR
U = Unimplemented bi
read as '0'
W = Writeable bit
'1' = bit is set
'0' = bit is cleared
'x' = bit is unknown

Figure redrawn by author from Reg 11-1 found in the PIC24H32GP202 datasheet (DS70289B), Microchip Technology, Inc.

Note 1: In 32-bit mode, T3CON bits do not affect 32-bit operation

Timer2 Yapılandırma

- Timer2' yi yapılandırmak için birkaç yol vardır.

Yöntem 1: Timer2' yi yapılandırmak için T2CON saklayıcısına 16 bitlik bir değer atamanız yeterlidir.

```
//Timer off, Pre=64, Internal clock
T2CON = 0b0000 0000 0010 0000
```

Yöntem 2: Makrolar kullanarak daha okunaklı yaklaşım

```
T2CON = T2_OFF | T2_IDLE_CON | T2_GATE_OFF |
        T2_32BIT_MODE_OFF | T2_SOURCE_INT | T2_PS_1_64;
```

Method 3: Bitlerin ayrı olarak ayarlanması

```
T2CONbits.TON = 1; //Set TON bit = 1, turn timer on
```

Kare Dalga Üretimi (1)

30 ms periyotlu bir kare dalga üretmek için RB14 pinini kullanın.
Çözüm: Timer2' yi her 15 ms' de bir kesme oluşturacak şekilde yapılandırın. Timer2 zaman aşımı kesmesi oluştuğunda RB14' nin çıkışı değiştirilir.

```
#define LED _LATB14
#define ISR_PERIOD 15
```

```
/****************Configure RB14 to output wave ***************/
inline void CONFIG_LED ()
{
        CONFIGURE_RB14_AS_DIG_OUTPUT (); //Use RB14 as output
}
```

```
/***************ISR for Timer 2 ***************/
void _ISRFAST _T2Interrupt (void)
{
        _T2IF = 0;
        LED =! LED;  //Toggle RB2 output
}
```

Kare Dalga Üretimi (2)

```
/***************Configure Timer 2 ***************/
void configureTimer2 (void){
//16 bit, internal clock, prescaler 64, idle continual timing, timer turn off, gate mode close
T2CON = T2_OFF | T2_IDLE_CON | T2_GATE_OFF
          | T2_32BIT_MODE_OFF | T2_SOURCE_INT | T2_PS_1_64;
PR2 = msToU16Ticks (ISR_PERIOD, getTimerPrescale(T2CON)) - 1;
TMR2 = 0; // Let Timer 2 count from 0
_T2IF = 0; // Clear interrupt flag
_T2IP = 1; // Set the priority level of Timer2 interrupt
_T2IE = 1; // Enable Timer2 interrupt
T2CONbits.TON = 1; // Turn on Timer2
}
```

```
void _main (void) {
CONFIG_LED(); // Configure RB14 to output wave
configTimer2();
while(1) {
}
```