Bölüm 13
I²C Seri Arabirimi

Doç. Dr. Fatih Evran
Elektrik-Elektronik Mühendisliği
Düzce Üniversitesi
Email: fatihevran@duzce.edu.tr

I^2C (Inter-Integrated Circuit) Özellikleri

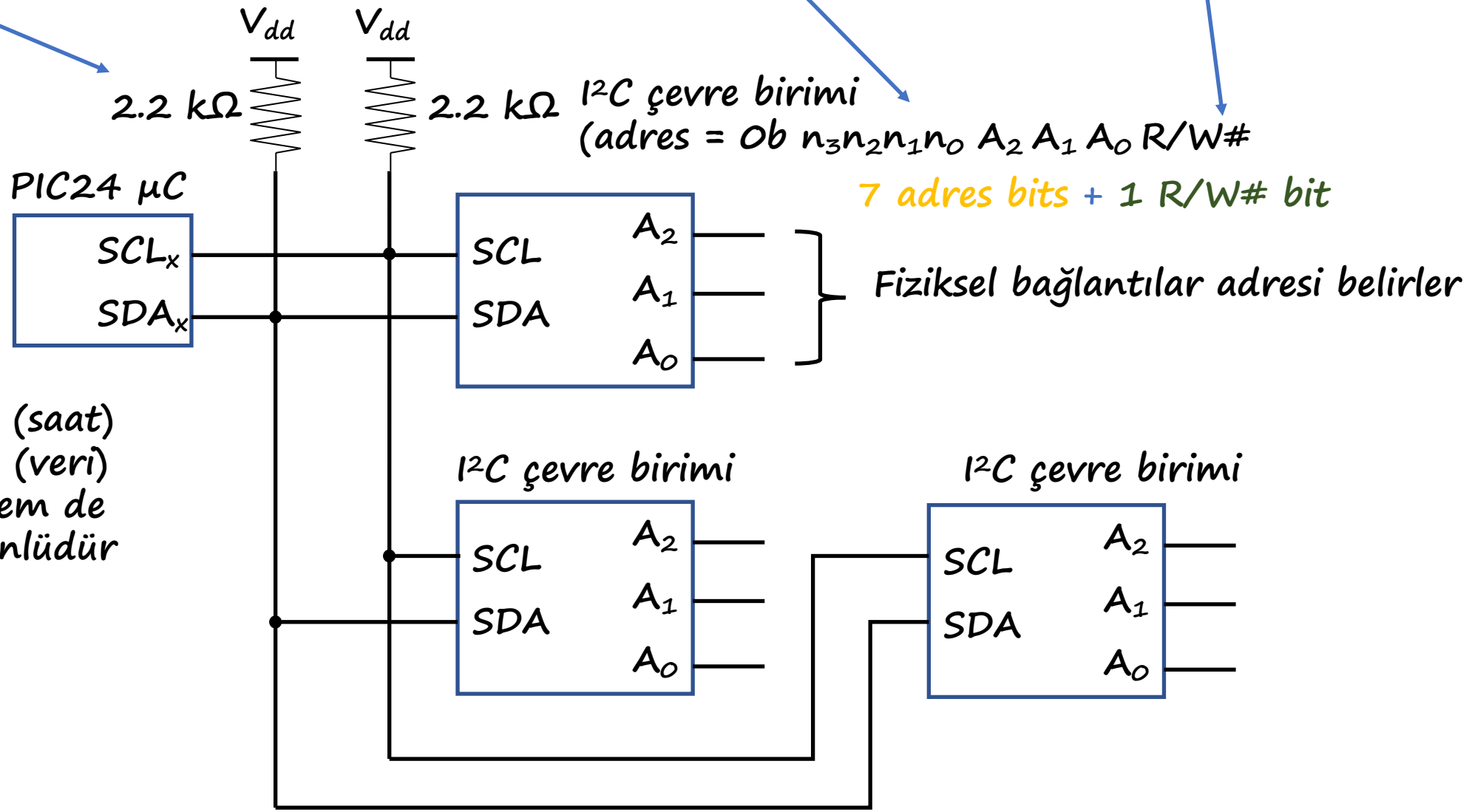
- SPI' da olduğu 'slave' cihazları seçmek için ayrı seçim hatlarına ihtiyaç duyulmaz.
  A. Her I^2C işleminin başlangıcında 7 bitlik bir slave cihazın adresi gönderilir.
  B. Her slave cihaz yolu dinler: Master tarafından gönderilen slave cihaz adresi, cihazın dahili adresi ile eşleşirse slave cihaz yanıt verir
- SDA (veri hattı) çift yönlüdür, iletişim yarı çift (half duplex) yönlüdür
- SDA ve SCLK pinleri 'open-drain' durumundadır, yani 'pull-up' direncine ihtiyaç duyarlar
  - Yol üzerinde birden fazla 'master' olmasına izin verir

I²C Veri Yolu

PIC24 sistemi için önerilen değer

Cihaz içinde kodlanmış, Cihaza özel

"0" Master -> Slave (yazma)
"1" Slave -> Master (okuma)



SCLx: Clock (saat)
SDAx: Data (veri)
Hem SCL hem de SDA çift yönlüdür

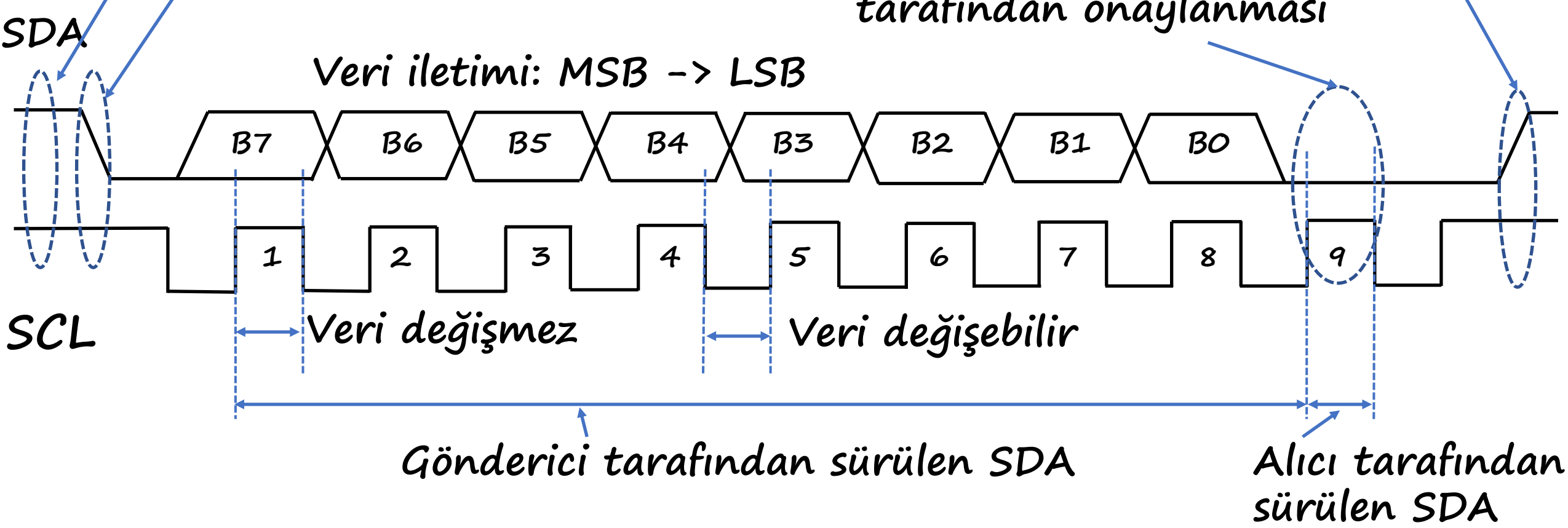
I²C Veri Yolu (devam)

Idle (Boş): SDA high, SCL high

Start: SDA high -> low, SCL high

Stop: SDA low -> high, SCL high

Ack (onay): Verinin geldiğinin alıcı tarafından onaylanması



Gönderilen her verinin uzunluğu onay biti nedeniyle 9 bittir.

I²C Veri Yolu (devam)

- Veri MSb bitinden LSb bitine doğru gönderilmektedir
- Gönderilen veri uzunluğu (9 bit) = 8 bit veri + 1 onay biti (ACK)

  8 bit veri → Gönderici gönderir
  1 onay biti (ACK) → Alıcı gönderir

- ACK (onay biti = 0) gönderilirse, bu verilerin alıcı tarafından başarıyla alındığı anlamına gelir.
- NAK (onay biti = 1) ise, iki durum vardır:

  A. Master verileri gönderdikten sonra, slave bir 'NAK' cevabı gönderebilir

  → Slave verileri başarıyla almadı

  B. Slave verileri gönderdikten sonra, master bir 'NAK' cevabı gönderebilir

  → Master, bu veri iletiminden sonra veri iletiminin durmasını istemektedir

PIC24 µC' de I²C Saklayıcıları

- I2CxCON ve I2CxSTAT sırasıyla kontrol ve durum saklayıcılarıdır
- I2CxRCV veri alma saklayıcısıdır
- I2CxTRN veri gönderme saklayıcısıdır, veri iletimi sırasında veriler bu saklayıcıya yazılır
- I2CxADD saklayıcısı 'slave' adresini tutar
- I2CxBRG, Baud Hızı Üreteci (BRG) olarak çalışır

Yaygın Olarak Kullanılan I²C Kontrol ve Durum Bitleri

Bit İsmi	Saklayıcı	Görevi
SEN	I2CxCON<0>	'Start' koşulunu başlatır (SEN=1), donanım tarafından silinir
RSEN	I2CxCON<1>	'Repeated Start' koşulunu başlatır (RSEN=1), donanım tarafından silinir
PEN	I2CxCON<2>	'Stop' koşulunu başlatır (PEN=1), donanım tarafından silinir
RCEN	I2CxCON<3>	Veri alma etkinleştirilir (RCEN=1), donanım tarafından silinir
ACKEN	I2CxCON<4>	Onay biti (ACK) etkinleştirilir (ACKEN=1), donanım tarafından silinir
ACKDT	I2CxCON<5>	Onay biti (ACK) bitidir: NAK için 1, ACK için 0
I2CEN	I2CxCON<15>	I²Cx modülünü etkinleştirir (I2CEN=1)
RBF	I2CxSTAT<1>	Veri Alma Durum biti: 1 = Veri alma tamamlandı, 0 = Veri alma tamamlanmadı
SI2CxIF	Interrupt Flag Status Registers	Slave modunda geçerli bir adresin saptanması, veri alınması ve veri iletme isteği üzerine kesme bayrağı 1 (bir) olmaktadır

I²C C Fonksiyonları

(a) I²C İşlemleri için Kullanılan Alt Fonksiyonlar

I²C Fonksiyon İsmi	Tanım
void configI2C1 (uint16 u16_FkHZ)	u16_FkHZ kHZ saat hızında çalışması için I2C1 modülünü etkinleştirir
void startI2C1 (void)	'Start' şartını gerçekleştirir
void rstartI2C1 (void)	'Repeated start' şartını gerçekleştirir
void stopI2C1 (void)	'Stop' şartını gerçekleştirir
void putI2C1 (uint8 u8_val)	u8_val verisi gönderilir; NAK oluşursa yazılımla işlemci sıfırlanır (reset)
uint8_t putNoAckCheckI2C1 (uint8 u8_val)	u8_val verisi gönderilir ve onay biti döndürülür
uint8_t getI2C1 (uint8 u8_ack2Send)	1 byte veri alınır ve onay biti olarak u8_ack2Send gönderilir

Bu fonksiyonlar daha üst düzey fonksiyonları uygulamak için kullanılacaktır.

I²C C Fonksiyonları (devam)

(b) I²C Veri Alma/Gönderme İşlemleri için Kullanılan Üst Fonksiyonlar

I²C Fonksiyon İsmi (Veri Alma/Gönderme)	Görevi
void write1I2C1 (uint8 u8_addr, uint8 u8_d1)	1 byte (u8_d1) veri gönderilir
void write2I2C1 (uint8 u8_addr, uint8 u8_d1, uint8 u8_d2)	2 byte (u8_d1 ve u8_d1) veri gönderilir
void writeNI2C1 (uint8 u8_addr, uint8* pu8_data, uint16 u16_cnt)	pu8_data içindeki u16_cnt sayıda veri gönderilir
void read1I2C1 (uint8 u8_addr, uint8* pu8_d1)	1 byte veri okunur; *pu8_d1 okunan değeri tutar
void read2I2C1 (uint8 u8_addr, uint8* pu8_d1, uint8* pu8_d2)	2 byte veri okunur; *pu8_d1 ve *pu8_d2 okunan değeri tutar
void readNI2C1 (uint8 u8_addr, uint8* pu8_data, uint16 u16_cnt)	u16_cnt sayıda veri okunur; *pu8_data okunan değeri tutar

Bu fonksiyonlar 1 veya daha fazla veri göndermek/almak için kullanılır

I²C üzerinden veri alırken, alınan değerin bir işaretçi değişkeninde saklanması gerekir

I²C Veri Gönderme (Yazma) İşlemleri (1)

İki byte verinin 'slave' cihaza gönderilmesi: write2I2C1 (uint8 u8_addr, uint8 u8_d1, uint8 u8_d2).

Adım 1: 'Start' şartını başlat. startI2C1 ()

Step 2: Slave adresini gönder. putI2C1 (u8_addr & 0xFE)

0xFE ile '&' yapın, çünkü adresin sadece 7 bitine ihtiyacı vardır. ''Veri gönderme" modunda çalışmak için u8_addr değişkeninin son bitinin "0" olması gerekir (bkz. sunum sayfa 3)

Step 3: İlk veriyi gönder. putI2C1 (u8_d1)

Step 4: İkinci veriyi gönder. putI2C1 (u8_d2)

Step 5: 'Stop' şartını başlat. stopI2C1 ()

Her veriyi (1 byte) başarıyla aldıktan sonra, slave bir onay biti ('ACK') gönderecektir.

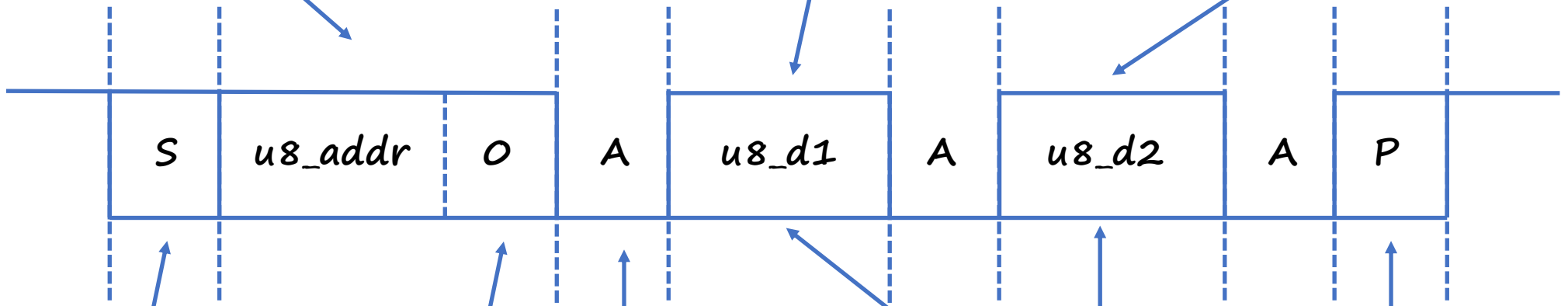
I²C Veri Gönderme (Yazma) İşlemleri (2)

write2I2C1 (uint8 u8_addr, uint8 u8_d1, uint8 u8_d2)

putI2C1 (u8_addr & 0xFE)

putI2C1 (u8_d1)

putI2C1 (u8_d2)



startI2C1 ()

R/W#=0 (write)

slave tarafından 'ACK' gönderilir, putI2C1() tarafından okunur

PIC24 µC veri gönderir

stopI2C1 ()

I²C Okuma İşlemleri (1)

İki byte verinin 'slave' cihazdan okunması: read2I2C1 (uint8 u8_addr, uint8* pu8_d1, uint8* pu8_d2).

Veri gönderme işlemi aşağıdaki adımlardan oluşur

Adım 1: 'Start' şartını başlat. startI2C1 ()

Adım 2: Slave adresini gönder. putI2C1 (u8_addr |0x01)

0x01 ile '|' yapın, çünkü adresin sadece 7 bitine ihtiyacı vardır. ''Veri okuma" modunda çalışmak için u8_addr değişkeninin son bitinin ''1" olması gerekir (bkz. sunum sayfa 3)

Adım 3: İlk veriyi oku ve ve 'ACK' bitini gönder. *pu8_d1 = getI2C1 (I2C_ACK)

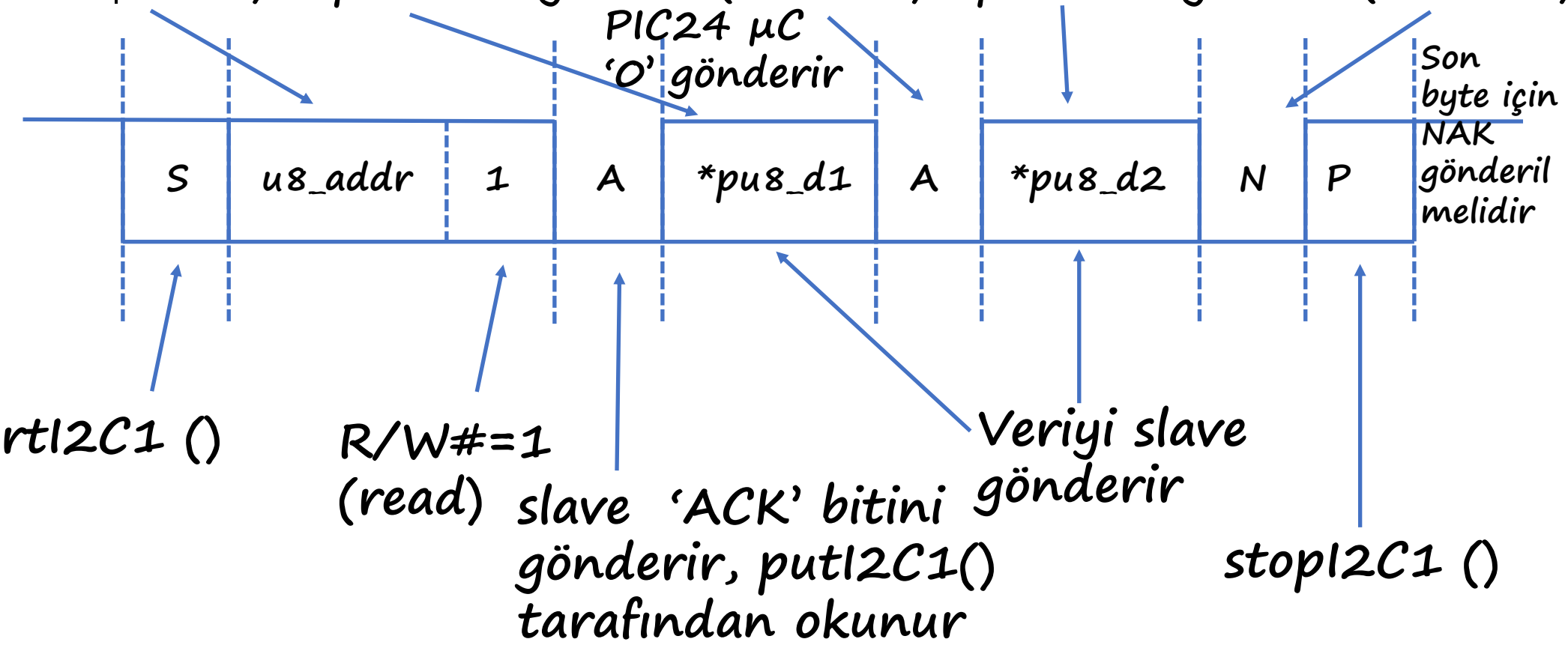
Adım 4: İkinciyi veriyi oku ve 'NAK' bitini gönder. *pu8_d2 = getI2C1 (I2C_NAK)

Adım 5: : 'Stop' şartını başlat. stopI2C1 ()

I²C Okuma İşlemleri (2)

read2I2C1 (uint8 u8_addr, uint8* pu8_d1, uint8* pu8_d2)

putI2C1 (u8_addr | 0x01) *pu8_d1 = getI2C1 (I2C_ACK) *pu8_d2 = getI2C1 (I2C_NAK)



Bir masteri olarak, ilk veriyi aldıktan sonra ACK bitini gönderin; ikinci veriyi aldıktan sonra NAK bitini göndermeniz (bkz. sunum sayfa 3)

Örnek-Alt Fonksiyonlar (1)

```
void configI2C1(uint16_t u16_FkHZ) {
uint32_t u32_temp;
u32_temp = (FCY/1000L)/((uint32_t) u16_FkHZ);
u32_temp = u32_temp - FCY/10000000L - 1;
I2C1BRG = u32_temp;
I2C1CONbits.I2CEN = 1;
}
I2C1BRG = ((Fcy/FSCL)-(Fcy/10,000,000))-1
```

u16_FkHZ kHz'de çalışma için I2C1BRG değeri hesaplanır

I2C modülünü etkinleştirilir

Örnek-Alt Fonksiyonlar (2)

```
void startI2C1(void) {
uint8_t u8_wdtState;
sz_lastTimeoutError = "I2C1 Start";
u8_wdtState = _SWDTEN; //save WDT state
_SWDTEN = 1; //enable WDT
I2C1CONbits.SEN = 1; // initiate start
// wait until start finished
while(I2C1CONbits.SEN);
_SWDTEN = u8_wdtState; //restore WDT
sz_lastTimeoutError = NULL;
}
```

stopI2C1() ve rstartI2C1() fonksiyonları benzerdir, ancak sırasıyla PEN ve RSEN bitlerini kullanır

Start olayını başlat ve bitmesini bekle

Örnek-Veri Göndermede Kullanılan Üst Fonksiyonlar (1)

```
#define I2C_WADDR(x) (x & 0xFE) //clear R/W bit of I2C address
void write1I2C1(uint8_t u8_addr,uint8_t u8_d1) {
startI2C1();
putI2C1(I2C_WADDR(u8_addr));
putI2C1(u8_d1);
stopI2C1();
}
void write2I2C1(uint8_t u8_addr,uint8_t u8_d1, uint8_t u8_d2) {
startI2C1();
putI2C1(I2C_WADDR(u8_addr));
putI2C1(u8_d1);
putI2C1(u8_d2);
stopI2C1();
}
```

Veri gönderme için LSB=0 olmalıdır

1 byte veri gönder (yaz)

Veri gönderme için LSB=0 olmalıdır

2 byte veri gönder (yaz)

Örnek-Veri Göndermede Kullanılan Üst Fonksiyonlar (2)

```
#define I2C_WADDR(x) (x & 0xFE) //clear R/W bit of I2C address
void writeNI2C1(uint8_t u8_addr,uint8_t* pu8_data, uint16_t
u16_cnt) {
        uint16_t u16_i;
        startI2C1();
        putI2C1(I2C_WADDR(u8_addr));
        for (u16_i=0; u16_i < u16_cnt; u16_i++) {
                putI2C1(*pu8_data);
                pu8_data++;
        }
  stopI2C1();
}
```

Veri gönderme için LSB=0 olmalıdır

N byte veri gönder (yaz)

İkiden fazla veri aktarımları için kullanılır

Örnek-Veri Okumada Kullanılan Üst Fonksiyonlar (1)

```
#define I2C_RADDR(x) (x | 0x01) // set R/W bit of I2C address
void read1I2C1(uint8_t u8_addr,uint8_t* pu8_d1) {
        startI2C1();
        putI2C1(I2C_RADDR(u8_addr));
        *pu8_d1 = getI2C1(I2C_NAK); // last ack bit from master to slave
                                    // during read must be a NAK
stopI2C1();
}
void read2I2C1(uint8_t u8_addr,uint8_t* pu8_d1, uint8_t* pu8_d2) {
        startI2C1();
        putI2C1(I2C_RADDR(u8_addr));
        *pu8_d1 = getI2C1(I2C_ACK);
        *pu8_d2 = getI2C1(I2C_NAK);
        stopI2C1();
}
```

1 byte veri oku

Veri okuma için LSB=1 olmalıdır

Veri okuma için LSB=1 olmalıdır

2 byte veri oku

Son veriyi aldıktan sonra master, slave cihaza iletimi durdurması için NAK bitini göndermelidir

Örnek-Veri Okumada Kullanılan Üst Fonksiyonlar (2)

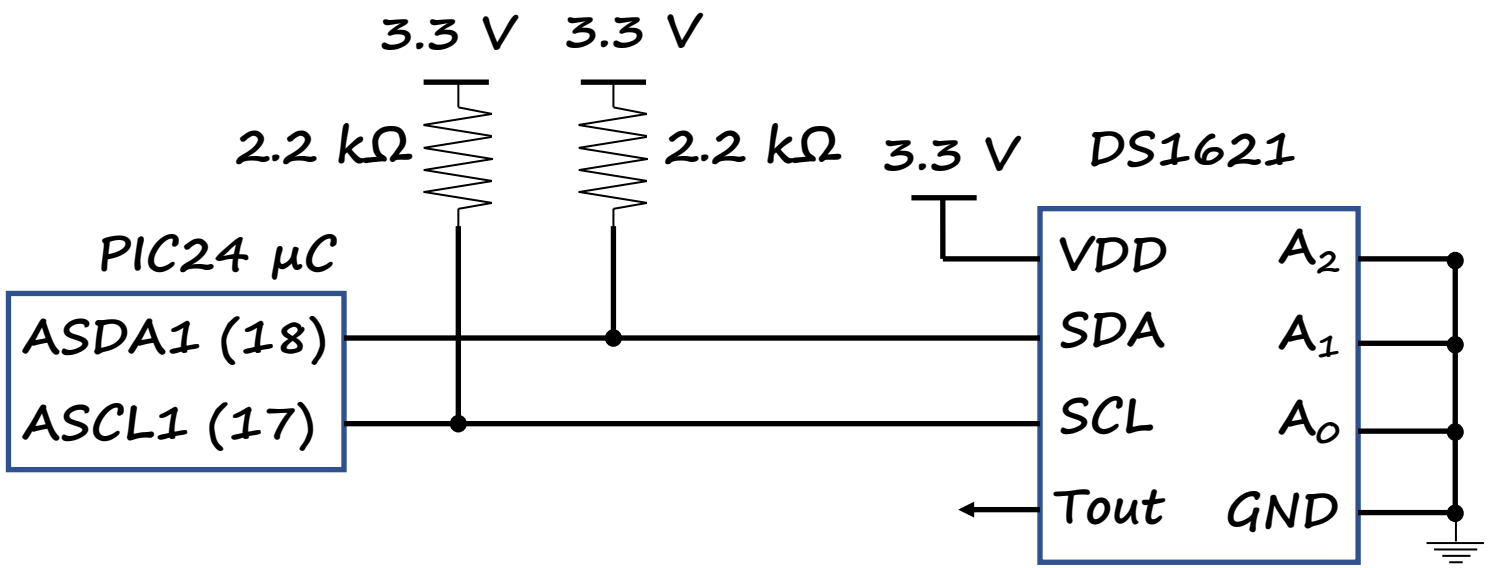
```
#define I2C_RADDR(x) (x | 0x01) // set R/W bit of I2C address
void readNI2C1(uint8_t u8_addr,uint8_t* pu8_data, uint16_t u16_cnt)
{
      uint16_t u16_i;
      startI2C1();
      putI2C1(I2C_RADDR(u8_addr));
      for (u16_i=0; u16_i < u16_cnt; u16_i++) {
      if (u16_i != u16_cnt-1) *pu8_data = getI2C1(I2C_ACK);
      else *pu8_data = getI2C1(I2C_NAK);
            pu8_data++;
      }
      stopI2C1();
}
```

Veri okuma için LSB=1 olmalıdır

N byte veri oku

Son veriyi aldıktan sonra master, slave cihaza iletimi durdurması için NAK bitini göndermelidir

PIC µC – DS1621 Termomotre Bağlantısı



- Maxim DS1621 sayısal bir termometredir (sıcaklık ölçer)
  - A. I^2C portu üzerinden DS1621 cihazını slave olarak yapılandırabiliriz
  - B. Ölçülen sıcaklık Santigrat (Celsius) cinsindedir
  - C. DS1621' in sıcaklık aralığı, 0,5°C artışlarla -55 °C ila +125 °C arasındadır.

DS1621 Adres Biçimi (1)

Adres biçimi:

1	0	0	1	A2	A1	A0	R/W#

MSByte	LSByte	
T[8:1] of temp. value	T[0]	

Sıcaklı veri biçimi:

1 1 1 0 0 1 1 1	0	0 0 0 0 0 0 0 0

OxE700 = -6400 = -6400/256= - 25 °C

A. Sonuç ikiye tümleyen biçimindedir
B. 9 bit çözünürlük (8.1): 8 bit tamsayı + 1 bit ondalık kısım

DS1621 termostat fonksiyonuna sahiptir: Bunun için TH, TL isimli iki adet dahili saklayıcı kullanılmaktadır. Sıcaklık > TH olduğunda, TOUT çıkışı lojik '1' olur. Sıcaklık < TL olduğunda, TOUT çıkışı tekrar lojik '0' olur. TH, TL kalıcı bellekte saklanır

DS1621 Yapılandırma Saklayıcısı

CONFIG Register (0xAC)

7	6	5	4	3	2	1	0
DONE	THF	TLF	NVB	X	X	POL	1SHOT

DONE	ADC işlemi tamamlandı bitidir. "1" = ADC tamamlandı, "0" = ADC devam ediyor.
THF	Sıcaklık ≥ TH olduğunda, THF='1' olur
TLF	Sıcaklık < TH olduğunda, TLF='1' olur
NVB	Kalıcı Bellek Meşgul bayrağıdır. NVB="1" ise E^2 hafızasına yazma işlemi devam etmektedir, NVB="0" E^2 hafıza meşgul değil. E^2 hafızaya kopyalama 10 ms kadar sürebilir.
POL	Çıkış Polaritesi Biti. "1" = aktif yüksek, "0" = aktif düşük. Bu bit uçucu değildir.
1SHOT	1SHOT= "1" ise DS1621, 'Start Convert T' komutunu aldıktan sonra bir sıcaklık dönüşümü gerçekleştirecektir. 1SHOT ="0" ise, DS1621 sürekli olarak sıcaklık dönüşümleri gerçekleştirecektir. Bu bit uçucu değildir. VNM' de depolanır.
X	Kullanılmaz

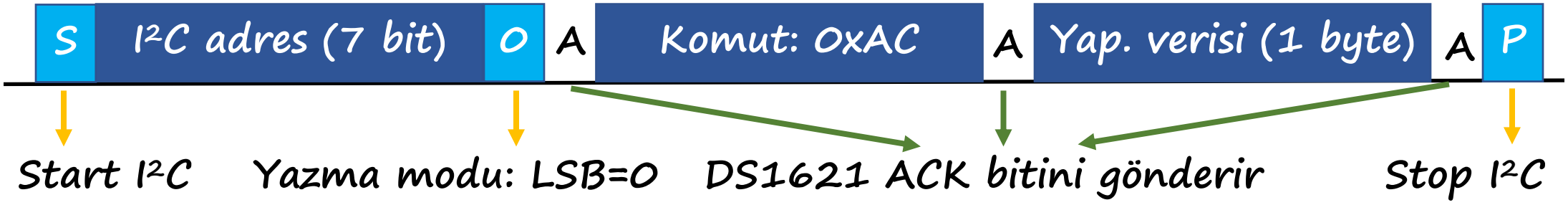
DS1621-Veri Gönderme (Yazma) İşlemi

DS1621' in yapılandırılması:

Adım 1: Yazma modunda DS1621' in adresini gönderiniz

Adım 2: OxAC (Access Config) komutunu gönderiniz

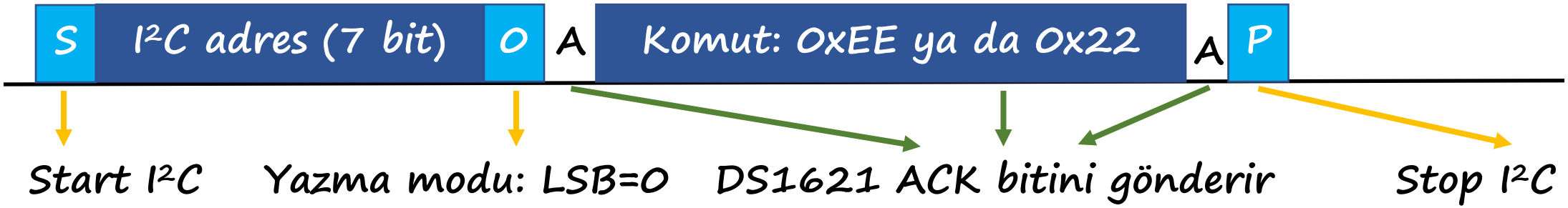
Adım 3: Önceki sayfada tanıtılan yapılandırma verisini (1 byte) gönderiniz



DS1621' de Sıcaklık Ölçümünü Başlat/Durdur:

Adım 1: Yazma modunda DS1621' in adresini gönderiniz

Adım 2: Sürekli ölçümü başlatmak için OxEE veya durdurmak için Ox22 komutunu gönderin



DS1621-Veri Alma İşlemi

DS1621' den sıcaklık okunması:

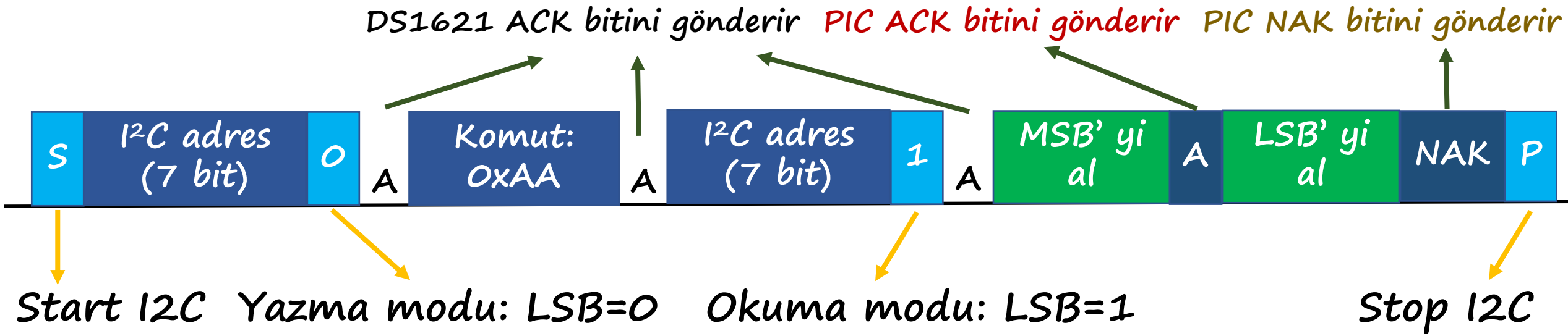
Adım 1: Yazma modunda DS1621' in adresini gönderiniz

Adım 2: Sıcaklık okuma kodunu (OxAA) gönderiniz

Adım 3: Okuma modunda DS1621' in adresini gönderiniz

Adım 4: Sıcaklık saklayıcısının MSB verisini okuyunuz

Adım 5: Sıcaklık saklayıcısının LSB verisini okuyunuz



DS1621-C Kodu (1)

```
#define DS1621ADDR 0x90   //DS1621 address with all pins tied low
#define ACCESS_CONFIG 0xAC
#define START_CONVERT 0xEE
#define READ_TEMP 0xAA
void writeConfigDS1621(uint8_t u8_i)
   {
      write2I2C1(DS1621ADDR, ACCESS_CONFIG, u8_i);
   }
void startConversionDS1621()
   {
      write1I2C1(DS1621ADDR, START_CONVERT);
   }
int16_t readTempDS161()
  {
      uint8_t u8_lo, u8_hi;
      int16_t i16_temp;
      write1I2C1(DS1621ADDR, READ_TEMP);
      read2I2C1 (DS1621ADDR, &u8_hi, &u8_lo);
      i16_temp = u8_hi;
      return ((i16_temp<<8)|u8_lo);
  }
```

DS1621 address = 0b 1001 A2 A1 A0 R/W#
0x90= 0b 1001 0 0 0 ?

8 bit yazma (veri gönderme) komutunu uygular

Bu komut bir sıcaklık dönüşümünü başlatır

16 bitlik okuma/yazma komutunu uygular

DS1621-C Kodu (2)

```
int main (void)
{
   int16_t i16_temp;
   float  f_tempC,f_tempF;
   configBasic(HELLO_MSG);
   configI2C1(400);                    //configure I2C for 400 KHz
   writeConfigDS1621(0x00);            //continuous conversion
   startConversionDS1621();            //start conversions
   while (1)
   {
      DELAY_MS(1500);
      i16_temp = readTempDS161();
      f_tempC = i16_temp;  //convert to floating point
      f_tempC = f_tempC/256;  //divide by precision
      f_tempF = f_tempC*9/5 + 32;
      printf("Temp is: 0x%0X, %4.4f (°C), %4.4f (°F)\n", i16_temp, (double) f_tempC, (double) f_tempF);
}}
```

I2C veri yolunu 400 kHz için yapılandırır

Sürekli modda sıcaklık okumayı yapılandırır

Sıcaklığı okuyun ve sonucu hex, Celsius ve Fahrenheit olarak yazdırın