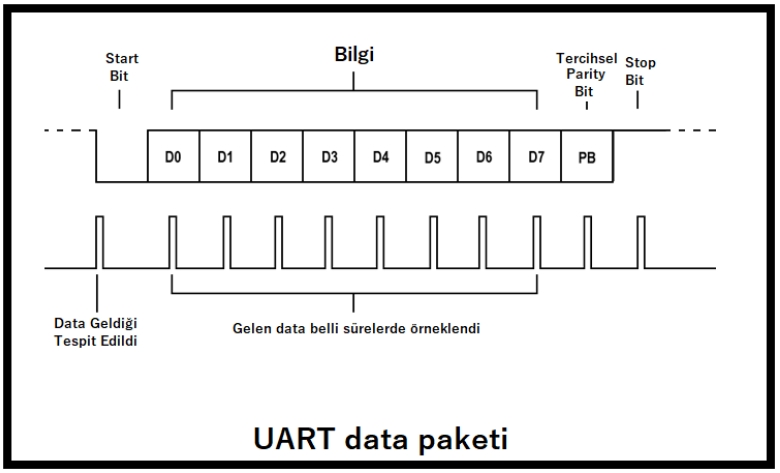
UART (Universal asynchronous receiver-transmitter):



UART data paketi

Baud Rate: Bir saniye içinde kaç adet bit iletileceğidir. Örneğin BaudRate 9600 ise bir bit yaklaşık 104us de iletilir. UART'ın Baud Rate'i arttıkça gönderilen bilginin iletim sırasında bozulmasının ihtimali artmaktadır. Bu sebeple baud rate'i mümkün olduğunca az tutmak gerekir. Tercih edilebilecek baudrate'ler çok çeşitli olsa da bilgi iletim hızının kritik olmadığı durumlarda 9600BPS, bilgi iletim hızının önemli olduğu durumlarda ise 115200BPS sık tercihlerdir.

Sık kullanılan Baud Rate'leri:
300,1200, 2400, 4800, **9600**, 19200, 38400, 57600, 74880, **115200**, 230400

Start bit: Paketin ilk kısmıdır ve "0" dır. Paket iletilmezken UART hattı MANTIKSAL YÜKSEK konumda durur. Paket iletimi başladığında start biti UART hattını MANTIKSAL DÜŞÜK'e çeker. Bu sayede alıcı alet mesaj iletiminin başladığını anlar.

Data Bits: UART ile iletilen paketler farklı kısımlardan oluşur. Bu kısımlardan biri de bilgidir. Bilgiye ayrılan kısmın kaç bit olacağı katı bir şekilde tanımlanmamakla beraber 5 ila 9 adet bitten oluşabilir. Günümüzdeki neredeyse tüm UART haberleşmelerinde bilgiye ait olan kısmın 8 bitten oluştuğunu söylemek yanlış olmaz.

Parity: Hiçbi iletişim kusursuz değildir. Bu sebeple gönderilen bilgi iletim sırasında bozulmaya uğrayabilir. İletilen datanın bozulmaya uğrayıp uğramadığını bilmek iletilen bilginin doğrulunun kritik olduğu sistemlerde çok önemlidir. Bu sebeple UART ile gönderilen mesajlara odd veya even parity biti eklenebilir. Datanın doğrulunun kritik olmadığı sistemlerde ise parity biti mesaja eklenmeyebilir.

•Odd parity: Bilginin içindeki bitlerinin ve odd parity bitinin toplamı tek sayı verir.
•Even parity:Bilginin içindeki bitlerinin ve even parity bitinin toplamı çift sayı verir.
•None Parity;UART paketinin içinde parity biti bulunmaz.

Stop bit: Paketin son kısmıdır ve “1” dır. Stop biti UART hattını MANTIKSAL YÜKSEK konumuna çeker. Genelde 1 stop biti yeterli olmaktadır fakat ,özellikle yavaş donanımlarla uğraşırken onlara zaman tanımak adına, 2 adet stop biti de tercih edilebilir.

Bit Order: UART paketinin içindeki bilgi kısmının 8 bit olduğunu söylemiştik. Bu 8 bit ile 64 sayısını temsil edebiliriz fakat bu temsil iki farklı şekilde yapılabilir.

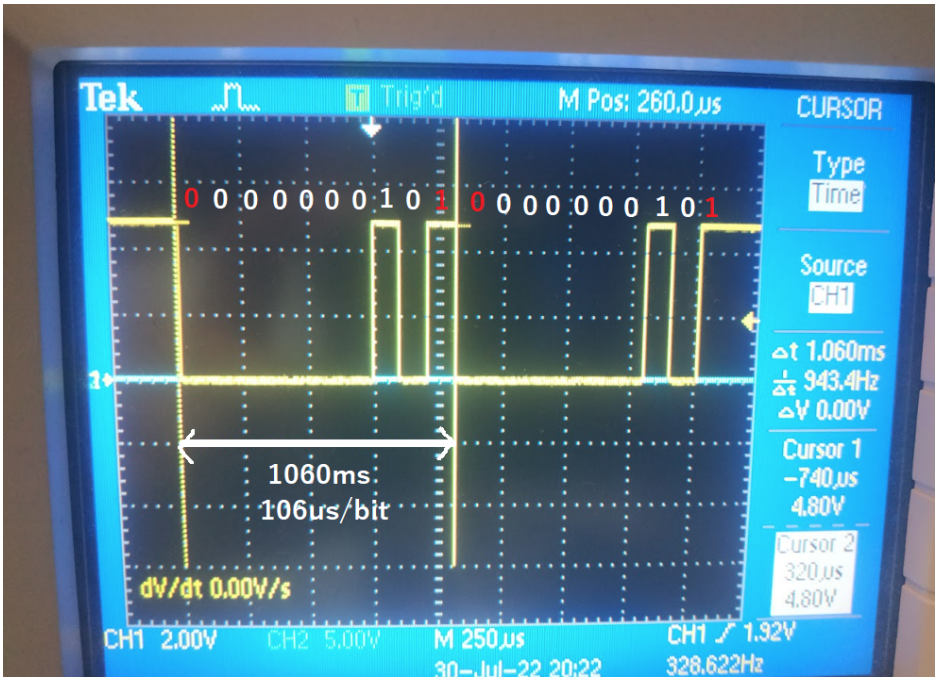
•Little- endian: En büyük basamak değerine sahip bit en sağda, en düşük basamak değerine sahip bit en soldadır..

64 = 0000 0010

•Big-endian: En düşük basamak değerine sahip bit en sağda, en büyük basamak değerine sahip bit en soldadır..

64 =0100 0000

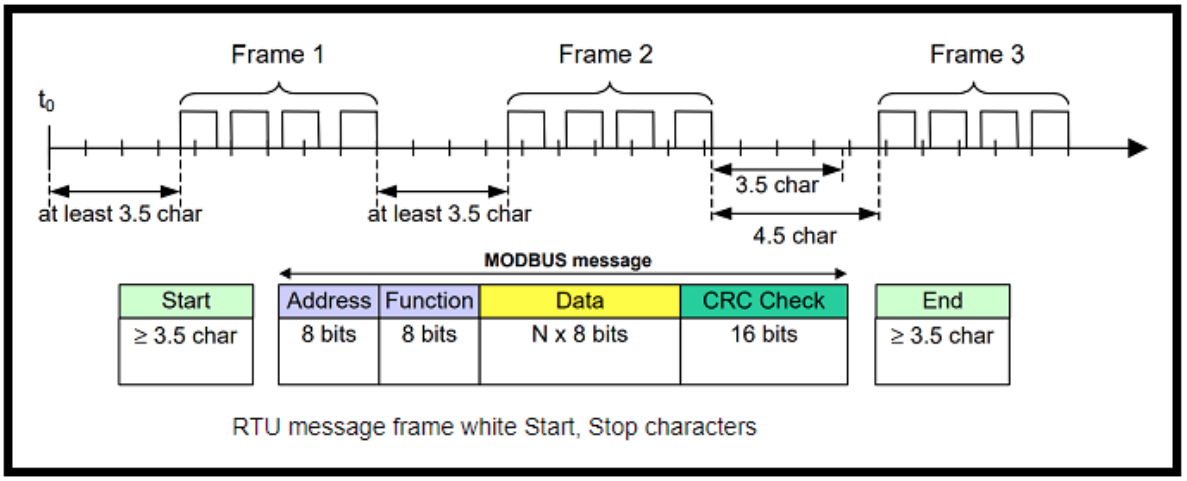
UART paketlerinde bilgi gönderilirken ilk olarak en düşük basamak değerine sahip bit, en son ise en yüksek basamak değerine sahip bit gönderilir. Aşağıdaki resimde 9600BPS, none Parity ve 1 Stop biti (8N1) formatındaki UART haberleşmesi ile art arda gönderilmiş 64 verisinin osiloskop çıktısını görülebilir.



Kısaltma	Start biti sayısı	Bilgiyi taşıyan bit sayısı	Parity tipi	Stop biti sayısı	Paketteki toplam bit sayısı
8N1	1	8	None	1	10
8O1	1	8	Odd	1	11
8E1	1	8	Even	1	11
8N2	1	8	None	2	11

8O2	1	8	Odd	2	12
8E2	1	8	Even	2	12

MODBUS:



MODBUS RTU ve ASCII farkı: 5 basamaklı 11111 sayısını karşı tarafa göndermek isteyelim. Bu göndermeyi karakter karakter de yapabiliriz direkt sayısal olarak da yapabiliriz. ASCII tablosuna bakıldığında '1' e karşılık gelen sayısal değer 49 dur. Biz UART ile 5 adet 49 değerini gönderdiğimizde karşı taraf bu karakterleri birleştirip sayıya çevirebilir. Fakat bu işlem için 5 byte'lık bilgi göndermek zorunda kaldık. Diğer yöntemde bu sayıyı iletmek için gereken byte sayısı daha azdır. 11111 sayısı 2'lik tabanda ifade edilebilir.

$$(11111)_{10} = (00101011\ 01100111)_{2}$$

Gözüktüğü üzre karakter karakter değilde doğrudan sayısal değerininin bilgisini göndermek için 2 byte'lık bilgi yeterli olmaktadır. Alıcı bu bilgileri daha sonradan birleştirebilir. Sayıyı karakter karakter gönderdiğimiz metodu MODBUS ASCII'ye, sayının doğrudan sayısal değer olarak iletildiği metodu ise MODBUS RTU'ya benzetebiliriz.

Slave address: MODBUS protokolü bir ustadan ve kölelerden oluşur. Ustanın isteğinin hangi köleye gittiğini anlamak için köleler'e 0 dan 255 e kadar olan özel adresler atanmıştır.

Register address (Starting address): Okuma ve yazma işlemlerinin yapılacağı, başlayacağı konumdur. Örneğin sıcaklık verisi aletin 5 numaralı register'ında olabilir.

Register value (Data): 16 Bitten oluşur. Sayısal bilgiyi taşır.

Query (request): master'ın isteğini içeren paket

Response: Slave'in geri dönüşünü içeren paket

Byte count: Usta köleden birden fazla register'ını okumak isteyebilir. Örneğin 5. adresten başlayarak toplamda 3 adet register okumak istenirse köle 5. 6. ve 7. registerlarında bulunan

datalar “response” paketinde bulunur. Her data 2 byte’ dan oluştuğundan bilginin “byte count”’u 6 olur.

CRC: parity bite benzer ama çok daha gelişmiş ve olasılıksal olarak hatayı tespit etme ihtimali iyileşirilmiştir. Modbus 16Bit’lik CRC kullanmaktadır. Oluşabilecek hatalara göre farklı tipte CRC’ler vardır. CRC’nin nasıl çalıştığını bilmemekteyim ama aşağıdaki kod CRC’yi hesaplayabilmektedir.

```
//MAGICAL CRC_16 MODBUS code.
uint16_t generate_CRC_16_bit(uint8_t number_of_bytes, uint8_t B_0, uint8_t B_1, uint8_t B_2, uint8_t B_3, uint8_t B_4, uint8_t B_5) {
  uint16_t remainder = CRC_16_bit_for_1BYTE(B_0, 65535);
  if (number_of_bytes >= 2 )  remainder = CRC_16_bit_for_1BYTE(B_1, remainder);
  if (number_of_bytes >= 3 )  remainder = CRC_16_bit_for_1BYTE(B_2, remainder);
  if (number_of_bytes >= 4 )  remainder = CRC_16_bit_for_1BYTE(B_3, remainder);
  if (number_of_bytes >= 5 )  remainder = CRC_16_bit_for_1BYTE(B_4, remainder);
  if (number_of_bytes >= 6 )  remainder = CRC_16_bit_for_1BYTE(B_5, remainder);
  return remainder;

}

uint16_t CRC_16_bit_for_1BYTE(uint16_t data, uint16_t last_data) {
  //if this is first data (i.e LAST_DATA==null), LAST_DATA= 65535 = FFFF
  uint16_t key = 40961; //1010 0000 0000 0001
  data = data ^ last_data;//XOR
  for (int i = 0; i < 8; i++) {
    boolean should_XOR = false;
    if (data % 2 == 1)should_XOR = true;
    data = data >> 1;
    if (should_XOR)data = data ^ key;
  }
  return data;
}
```

Function code: MODBUS protokolünde usta kölelerden okumak ve yazmak gibi çeşitli isteklerde bulunabilir ve köle ustasının isteğini yerine getirdiğine yönelik cevap verebilir. Aynı şekilde istekte bulunulan köle isteği yerine getiremeyip hata yollayabilir. İsteğin, başarının ve hatanın ne olduğunun bilgisi function code’da tutulur. MODBUS’da sıkca kullanılan fonksiyon kodlarını aşağıdaki tabloda bulabilirsiniz.
fonksiyon kodu, İsim, ne yapar.

Object type	Access	Size
Coil	Read-write	1 bit
Discrete input	Read-only	1 bit
Input register	Read-only	16 bits
Holding register	Read-write	16 bits

Fonksiyon kodu	Adı	Ne yapar
1	Coils	Okur
2	Discrete Inputs	Okur
3	Multiple holding registers	Okur
4	Input registers	Okur
5	Single Coil	Yazar
6	Single holding register	Yazar
15	Multiple coils	Yazar
16	Multiple holding registers	Yazar

- Fonksiyon kodu 3:

Query

Field Name	RTU (hex)
Header	None
Slave Address	01
Function	03
Starting Address Hi	00
Starting Address Lo	00
Quantity of Registers Hi	00
Quantity of Registers Lo	02
Error Check Lo	C4
Error Check Hi	0B

Response

Field Name	RTU (hex)
Header	None
Slave Address	01
Function	03
Byte Count	04
Data Hi	00
Data Lo	06
Data Hi	00
Data Lo	05
Error Check Lo	DA
Error Check Hi	31

Function Code 3: Read Holding Registers

- Fonksiyon kodu 4:

Query

Field Name	RTU (hex)
Header	None
Slave Address	01
Function	04
Starting Address Hi	00
Starting Address Lo	00
Quantity of Registers Hi	00
Quantity of Registers Lo	02
Error Check Lo	71
Error Check Hi	CB

Response

Field Name	RTU (hex)
Header	None
Slave Address	01
Function	04
Byte Count	04
Data Hi	00
Data Lo	06
Data Hi	00
Data Lo	05
Error Check Lo	DB
Error Check Hi	86

Function Code 4: Read Input Registers

- Fonksiyon kodu 6:

Query

Field Name	RTU (hex)
Header	None
Slave Address	11
Function	06
Register Address Hi	00
Register Address Lo	01
Write Data Hi	00
Write Data Lo	03
Error Check Lo	9A
Error Check Hi	9B

Response

Field Name	RTU (hex)
Header	None
Slave Address	11
Function	06
Coil Address Hi	00
Coil Address Lo	01
Write Data Hi	00
Write Data Lo	03
Error Check Lo	9A
Error Check Hi	9B

Function Code 6: Write holding register

Exception codes: Eğer slave'den istenen şeyi slave yapamıyorsa hata kodu gönderebilir.

Request

This command is requesting the ON/OFF status of discrete coil #1186

from the slave device with address 10.

0A 01 04A1 0001 AC63

0A: The Slave Address (0A hex = address10)

01: The Function Code 1 (read Coil Status)

04A1: The Data Address of the first coil to read

(04A1 hex = 1185 , + 1 offset = coil #1186)

0001: The total number of coils requested.

AC63: The CRC (cyclic redundancy check) for error checking.

Response

0A 81 02 B053

0A: The Slave Address (0A hex = address10)

81: The Function Code 1 (read Coil Status - with the highest bit set)

02: The Exception Code

B053: The CRC (cyclic redundancy check).

Function Code in Request	Function Code in Exception Response
01 (01 hex) 0000 0001	129 (81 hex) 1000 0001
02 (02 hex) 0000 0010	130 (82 hex) 1000 0010
03 (03 hex) 0000 0011	131 (83 hex) 1000 0011
04 (04 hex) 0000 0100	132 (84 hex) 1000 0100
05 (05 hex) 0000 0101	133 (85 hex) 1000 0101
06 (06 hex) 0000 0110	134 (86 hex) 1000 0110
15 (0F hex) 0000 1111	143 (8F hex) 1000 1111
16 (10 hex) 0001 0000	144 (90 hex) 1001 0000

Exception Code	Name	Meaning
01 (01 hex)	Illegal Function	The function code received in the query is not an allowable action for the slave. This may be because the function code is only applicable to newer devices, and was not implemented in the unit selected. It could also indicate that the slave is in the wrong state to process a request of this type, for example because it is unconfigured and is being asked to return register values. If a Poll Program Complete command was issued, this code indicates that no program function preceded it.
02 (02 hex)	Illegal Data Address	The data address received in the query is not an allowable address for the slave. More specifically, the combination of reference number and transfer length is invalid. For a controller with 100 registers, a request with offset 96 and length 4 would succeed, a request with offset 96 and length 5 will generate exception 02.
03 (03 hex)	Illegal Data Value	A value contained in the query data field is not an allowable value for the slave. This indicates a fault in the structure of remainder of a complex request, such as that the implied length is incorrect. It specifically does NOT mean that a data item submitted for storage in a register has a value outside the expectation of the application program, since the MODBUS protocol is unaware of the significance of any particular value of any particular register
04 (04 hex)	Slave Device Failure	An unrecoverable error occurred while the slave was attempting to perform the requested action.
05 (05 hex)	Acknowledge	Specialized use in conjunction with programming commands. The slave has accepted the request and is processing it, but a long duration of time will be required to do so. This response is returned to prevent a timeout error from occurring in the master. The master can next issue a Poll Program Complete message to determine if processing is completed.
06 (06 hex)	Slave Device Busy	Specialized use in conjunction with programming commands. The slave is engaged in processing a long-duration program command. The master should retransmit the message later when the slave is free..
07 (07 hex)	Negative Acknowledge	The slave cannot perform the program function received in the query. This code is returned for an unsuccessful programming request using function code 13 or 14 decimal. The master should request diagnostic or error information from the slave.
08 (08 hex)	Memory Parity Error	Specialized use in conjunction with function codes 20 and 21 and reference type 6, to indicate that the extended file area failed to pass a consistency check. The slave attempted to read extended memory or record file, but detected a parity error in memory. The master can retry the request, but service may be required on the slave device.

10 (0A hex)	Gateway Path Unavailable	Specialized use in conjunction with gateways, indicates that the gateway was unable to allocate an internal communication path from the input port to the output port for processing the request. Usually means the gateway is misconfigured or overloaded.
11 (0B hex)	Gateway Target Device Failed to Respond	Specialized use in conjunction with gateways, indicates that no response was obtained from the target device. Usually means that the device is not present on the network.