

# 08 UART

5 Mayıs 2021 Çarşamba

08:03

## 08 UART

### Giriş

- <https://cennttceylmn.medium.com/elektronik-haberleşme-protokolleri-nedir-d11a6d3a5957> link üzerinden haberleşme protokolleri hakkında bilgi alabiliriz.
- <https://www.ercankoclar.com/2018/04/uart-iletisim-protokolu-ve-mikroc-kutuphanesi/> <https://arduinodestek.com/uart-haberleşme-nedir-ve-nasil-gerçekleşir/>
- <https://youtu.be/uktFwZX2TTE>, <https://youtu.be/K0JuUAQYsaQ> ve <https://youtu.be/UCXVFJSrIbE> protokol hakkında videolardan bilgi edinebiliriz.
- <https://youtu.be/GRmYKJgAtQ4>, <https://youtu.be/NDwpWbXJ0sc>, <https://youtu.be/vrSzdoKv558>, [https://youtu.be/vzRuLn\\_Gzx8](https://youtu.be/vzRuLn_Gzx8), <https://youtu.be/ic8NUSytU-g>, <https://youtu.be/y7ZETFlohp0>, <https://youtu.be/1lGm99He7g4> linklerinden STM32 ile yapılmış örnek uygulamaları izleyebiliriz.
- UART (Universal Asynchronous Receiver Transmitter), 1 ve 0'lerden oluşan verileri iki dijital sistem arasında alıp verme işlemlerinde kullanılan bir iletişim protokolüdür.

### Avantajları ve Dezavantajları

#### Avantajlar;

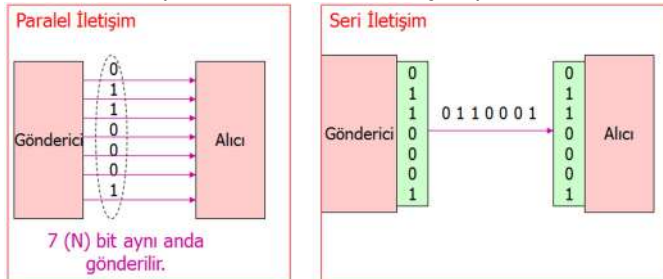
- Sadece iki kablo kullanır.
- Saat sinyali gerekli değildir.
- Hata denetimine izin vermek için bir eşlik biti vardır.
- Veri paketinin yapısı, her iki taraf da buna göre ayarlandığı sürece değiştirilebilir.
- İyi belgelenmiş ve yaygın olarak kullanılan yöntem.

#### Dezavantajlar;

- Veri çerçevesinin boyutu maksimum 9 bit ile sınırlıdır.
- Birden çok bağımlı veya birden çok ana sistemi desteklemez.
- Her UART'ın baud hızı, birbirinin %10'u dahilinde olmalıdır.

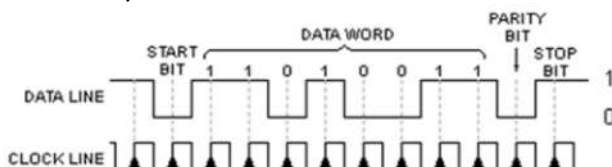
### İletişim Yöntemi

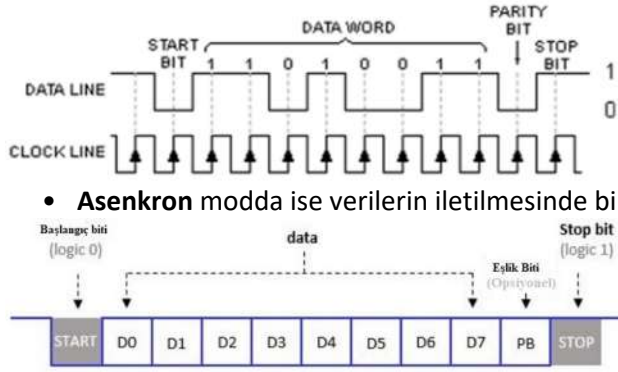
- Dijital sistemlerde iletişim paralel ve seri olmak üzere iki türlü yapılır.
- **Paralel** iletişimde bilgi vericiden alıcıya aynı anda birden fazla bit gidecek şekilde gönderilir. Böylece tek hamlede birden fazla veri karşı tarafa gönderildiği için iletişim hızı yüksektir. Fakat bu iletişim türünde kullanılan hat sayısı fazladır ve uzun mesafeler için uygun değildir.
- **Seri** iletişimde ise vericiden alıcıya gönderilecek bilgi, tek hat üzerinden sırayla gönderilir. Bu şekilde giden bilginin, tek hamlede tek biti gönderebileceği için iletişim hızı yavaşlatır. Fakat seri iletişimde hat sayısı azdır ve uzun mesafeli iletişim için daha uygundur.
- USART protokollü bir seri iletişim protokolüdür.



### İletişim Modu

- USART protokolünde veriler senkron veya asenkron olarak alınabilirler.
- **Senkron** veri alışverişinde bir data hattı ve bir clock hattı bulunmalıdır. Daha hattından gidecek veriler clock hattından gönderilen sinyalin her düşen veya yükselen kenarında alıcıya iletilir.





- **Asenkron** modda ise verilerin iletilmesinde bir clock hattına ihtiyaç duyulmaz.

## Parametreler

- Verilerin gönderilmeye başlayacağı, alıcıya bir başlangıç için **Start** sinyali ile bildirir ve hemen arkasından veriler akmaya başlar daha sonrasında **Stop** biti ile sonlandırılır.
- Start biti "0" stop biti "1" verilerinden oluşmaktadır. Veri bitleri ise 7 veya 8 bit olabilir.
- Verilerin doğru olarak gönderilip gönderilmediğini anlamak için kullanılan **Parity** biti bulunmaktadır. Parity bitinin gönderilmesi şart değildir.
- Parity biti genellikle üç farklı şekilde kullanılır. Even, Odd ve None olarak seçilebilir. Parite, **Even** olarak kullanıldığında, iletilen verinin toplamda çift sayıda yüksek seviyeye sahip olması gerekmektedir. Eğer iletilen verinin yüksek seviyeye sahip bit sayısı tek ise, parite biti 1 olacak şekilde ayarlanır ve toplamda çift sayıda yüksek seviye olmasını sağlar. Eğer iletilen verinin yüksek seviyeye sahip bit sayısı zaten çift ise, parite biti 0 olarak ayarlanır. **Odd** ise bunun tam tersidir. Parite bitinin kullanılmadığı durumlarda **None** seçeneğini kullanırız ve böylece iletilen verinin doğruluğu kontrol edilmez. Parite biti için boş bir bit alanı bırakılır ve sadece veri bitleri iletimi gerçekleştirilir.
- Parite biti, basit bir hata kontrol mekanizmasıdır ve veri bütünlüğünü sağlamak için kullanılır. Ancak, parite biti tek başına tüm hataları tespit etmek veya düzeltmek için yeterli değildir. Daha güvenli ve sağlam hata kontrol yöntemleri için farklı yöntemler ve algoritmalar kullanılabilir, örneğin **CRC** (Cycle Redundancy Check)
- **Baudrate** saniyede gönderilen bit sayısıdır ve bps (bits per second - saniyede gönderilen bit sayısı) birimi ile ölçülür. Standart veri gönderme hızları 110, 150, 300, 600, 1200, 2400, 4800, 9600, 56000, 115200 gibi hızdadır. Baudrate hızı **artıkça** veri iletim mesafesi **azalır**.
- İki cihaz arasında UART haberleşmesi yapılıyorsa, her iki cihazın da **aynı** baud oranı, veri bitleri, parite biti ve stop bitleri yapılandırmasına sahip olması gerekmektedir. Bu parametrelerin doğru bir şekilde yapılandırılması, güvenilir ve hatasız veri iletimini sağlar.

## Veri İletimi Süresi

- Belirli bir baud hızında (baud rate) veri iletiminin ne kadar sürdüğünü hesaplamak için aşağıdaki hesaplama yöntemi kullanılır.
  - **Baud Hızı (Baud Rate):** Bu, saniyede kaç karakter gönderildiğini belirtir. Eğer her karakter 1 bit'e eşitse, bu durumda saniyede kaç bit gönderildiğini de gösterir. Örneğin, 9600 baud, saniyede 9600 bit'in iletildiği anlamına gelir.
  - **Bir Bit'in İletim Süresi:** Bir bit'in iletilmesi için geçen süre, baud hızı ile ters orantılıdır. Yani bir bit'in iletim süresi, Bit Süresi =  $1 / \text{Baud Rate}$  hesap edilir. Saniye cinsinden sonuç verir.
  - **Bir Byte'ın İletim Süresi:** Bir byte 8 bit'ten oluşur, fakat genellikle veri iletiminde başlangıç biti, durdurma biti ve bazen hata kontrol biti gibi ek bitler de olabilir. Standart bir asenkron iletişimde bu, genellikle 10 bit olarak kabul edilir. Byte Süresi = Bit Sayısı  $\times$  Bit Süresi şeklinde hesap edilir.

- Baud hızına göre bir byte'ın iletim süresini hesaplamak için kullanılan formül,

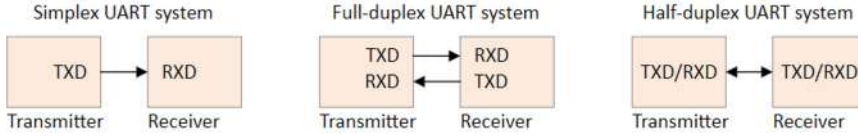
$$\text{Byte Süresi} = 10 / \text{Baud Hızı}$$

- Bu formülü kullanarak farklı baud hızları için byte süreleri;
  - 115200 baud hızı için Byte Süresi  $\approx 86.8 \mu s$
  - 57600 baud hızı için Byte Süresi  $\approx 173.6 \mu s$
  - 38400 baud hızı için Byte Süresi  $\approx 260.4 \mu s$
  - 19200 baud hızı için Byte Süresi  $\approx 520.8 \mu s$
  - 9600 baud hızı için Byte Süresi  $\approx 1.04 ms$
  - 4800 baud hızı için Byte Süresi  $\approx 2.08 ms$

## Çalışma Modları

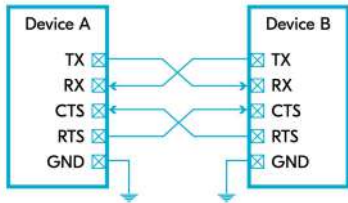
- UART'ın üç çalışma modu vardır. Bunlar Full Duplex, Half duplex ve Simplex'dir.

- **Full Duplex** iletişimde, veri gönderme ve veri alma işlemleri aynı anda ve bağımsız olarak gerçekleştirilir.
- **Half Duplex** iletişimde, veri gönderme ve veri alma işlemleri aynı hattı paylaşan cihazlar arasında sırayla gerçekleştirilir.
- **Simplex** iletişimde, veri iletimi sadece bir yönde gerçekleşir.



## Pin Yapısı

- **TX** (Transmit) ve **RX** (Receive), UART haberleşmesinde kullanılan veri gönderme ve veri alma hatlarını temsil eder.
- **CTS** (Clear To Send) ve **RTS** (Request To Send) ise UART haberleşmesinde kullanılan kontrol sinyalleridir. RTS ve CTS sinyalleri, veri akışını kontrol etmek için kullanılır ve genellikle aşırı yüklenmeyi önlemek veya veri kaybını engellemek için kullanılır.
- **TX** hattı, veri gönderici tarafından kullanılan seri veri gönderme hattını temsil eder. Veri gönderici, veri paketini seri olarak TX hattına gönderir ve bu hattı kullanarak veriyi alıcıya iletir.
- **RX** hattı, veri alıcı tarafından kullanılan seri veri alma hattını temsil eder. Veri alıcı, veriyi seri olarak RX hattından alır ve bu hattı kullanarak veriyi işler.
- Veri gönderici tarafından veri göndermeye hazır olduğunu bildirmek için **RTS** sinyalini HIGH seviyeye çeker. Bu, veri alıcının veriyi almasını ve işlemeye başlamasını sağlar.
- Veri alıcı tarafından veri almayı ve işlemeyi kabul etmek için hazır olduğunu belirtmek için **CTS** sinyalini HIGH seviyede tutar. Bu, veri göndericinin veriyi göndermeye başlamasını sağlar.
- RX giriş pini ile TX çıkış pini ve CTS giriş pini ile RTS çıkış pini birbirlerine ters bağlanır.

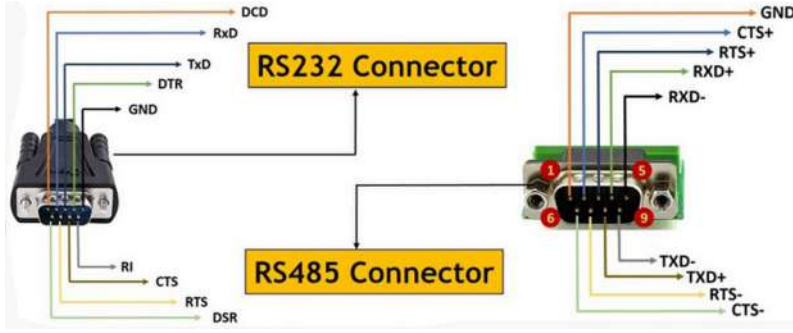


## Fiziksel Standartlar

- USART, seri iletişimde geniş bir kullanım alanına sahiptir ve çeşitli protokollerin uygulanmasına olanak tanır. Bu protokoller arasında RS232, RS422, RS485 fiziksel standartlar yer alır.
- <https://blog.direnc.net/rs232-ve-rs485-nedir-kullanim-alani-avantaj/>, <https://www.elektrikde.com/rs-232-rs-485-ve-rs-422-seri-iletisim/> ve <https://youtu.be/ChCRIU2kEE0> link üzerinden fiziksel standartlar hakkında bilgi edinebiliriz.
- **RS232**, tek bir veri iletim hattı üzerinden iletişim sağlar ve asenkron veri iletimine uygun bir protokol kullanır. Genellikle 15 volt ile -15 volt arasında bir gerilim seviyesi kullanır. Seri iletişim yaklaşık 15 metre kadardır.
- **RS-422** ve **RS-485**, her ikisi de seri haberleşme standardı olan ve diferansiyel sinyal iletimini kullanan protokollerdir. RS485, RS422'nin bir üst kümesidir, bu nedenle tüm RS422 cihazları RS485 tarafından kontrol edilebilir.
  - RS422 hattında yalnızca bir verici bulunabilirken, bu vericiden gelen sinyalleri alabilen birden fazla alıcı olabilir.
  - RS485, birden fazla cihaz arasında noktadan noktaya veya çok noktaya bağlantılar sağlayabilir.
- RS232 ile RS485 standartlarının özellikleri şu şekildedir:

|                                  | RS232                     | RS485  |
|----------------------------------|---------------------------|--|
| Voltaj Sistemi                   | Gerilim seviyesine dayalı | Diferansiyel                                     |
| Tek Hattı Toplam Sürücü ve Alıcı | 1 Sürücü, 1 Alıcı         | 32 Sürücü, 32 Alıcı (Aynı anda bir Sürücü etkin) |
| Hat Yapılandırması               | Noktadan Noktaya          | Çok Aktarmalı                                    |
| Maksimum Operasyonel Mesafe      | 15M/50FT                  | 1200M / 3000FT                                   |
| Maksimum Veri İletim Hızı        | 1 MBit/sn                 | 10 MBit/sn                                       |
| Maksimum Sürücü Çıkış Voltajı    | ±25V                      | -7V ila +12V                                     |
| Alıcı Giriş Direnci              | 3 ila 7 kΩ                | 12 kΩ  |
| Alıcı Giriş Voltaj Aralığı       | ±15V                      | -7V ila +12V                                     |
| Alıcı Duyarlılığı                | ±3V                       | ±200mV   |

- RS232 ve RS485 konnektörlerinin pin bağlantı bilgileri şu şekildedir:

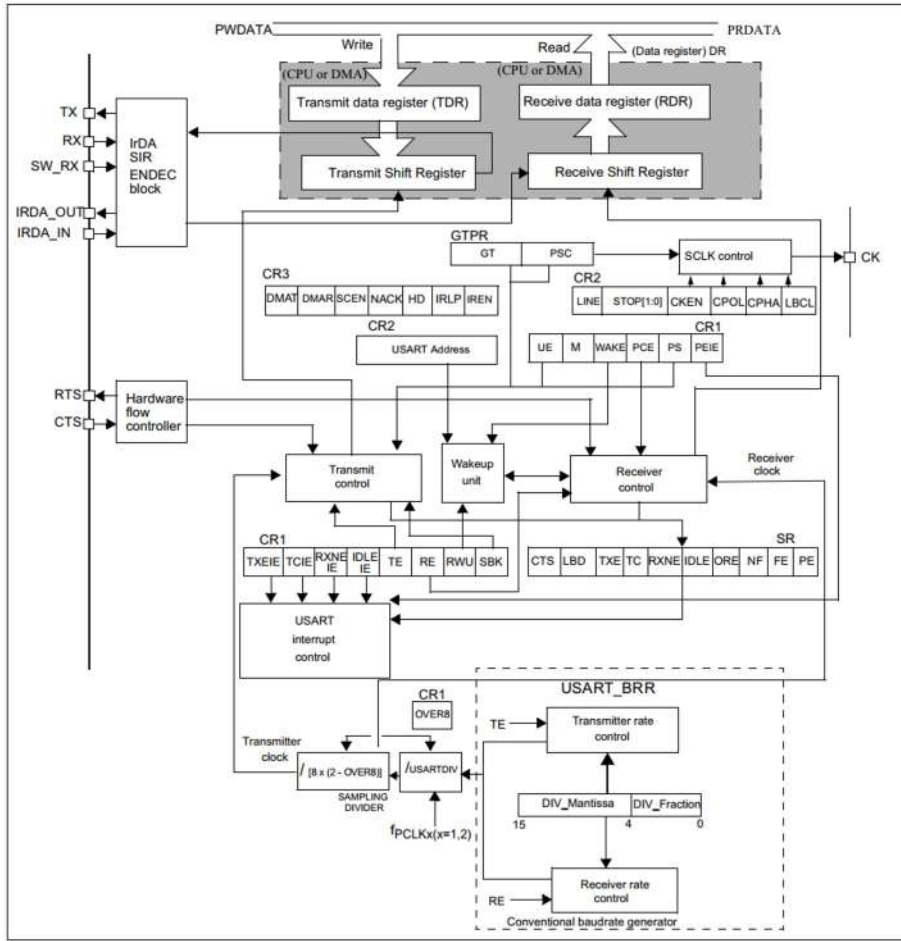


- RS-232, eski ve yaygın olarak kullanılan bir seri haberleşme standardıdır. Ancak daha yüksek veri hızları, uzun mesafe iletimi ve çok noktalı haberleşme gerektiren uygulamalarda RS-485 gibi daha modern haberleşme standartları tercih edilmektedir.
- RS422 ve RS485 protokollerinde **2-wire** ve **4-wire** bağlantı tipleri, veri iletişim şekline göre farklılık gösterir.
  - 2-Wire bağlantı, **Half-duplex** iletişim sağlar. Bu yüzden iletim ve alım aynı anda yapılamaz. Sinyal hattı, **A** ve **B** hatları üzerinden veri iletilir. Bu iki tel, hem verici hem de alıcı tarafında kullanılır, bu nedenle sadece bir verici aktif olduğunda iletişim gerçekleşir.
  - 4-Wire bağlantı, **Full-duplex** iletişim sağlar. Bu sayede veri aynı anda her iki yönde de iletebilir. İki ayrı çift sinyal hattı kullanılır. Bir çift **Tx+**, **Tx-** veriyi gönderirken, diğer çift **Rx+**, **Rx-** veriyi alır. Bu sayede, veri iletimi ve alımı için bağımsız hatlar oluşturulur.

## Haberleşme Metotları

- UART üzerinden Polling, Interrupt ve DMA olmak üzere üç farklı haberleşme yapılabilir.
- <https://deepbluembedded.com/how-to-receive-uart-serial-data-with-stm32-dma-interrupt-polling/>, <https://controllerstech.com/uart-receive-in-stm32/> ve <https://controllerstech.com/uart-transmit-in-stm32/> link ile bu metotlar ile yapılan örnekleri inceleyebiliriz.
- **Polling** yani Yoklama yöntemi, mikrodenetleyici tarafından sürekli olarak UART veri alımını kontrol etmek için kullanılır. Mikrodenetleyici, UART veri alımını düzenli aralıklarla sorgular ve yeni veri varsa onu işler. Veri gelene kadar mikrodenetleyici diğer işlemleri yapamaz ve sürekli olarak UART'ı kontrol etmek zorunda kalır. Bu yöntem, basit uygulamalarda ve düşük hızlı veri iletiminde tercih edilebilir. Yalnızca UART kullanıyorsak ve başka bir şey kullanmıyorsak bu kullandığımız polling yöntemi kullanmak iyidir, aksi takdirde diğer tüm işlemler etkilenecektir.
- **Interrupt** yöntemi, UART'dan veri alındığında veya veri gönderildiğinde mikrodenetleyiciye kesme sinyali göndererek mikrodenetleyicinin normal işlemlerini kesmesini sağlar. Bu yöntem, mikrodenetleyicinin sürekli olarak UART'yı sorgulamaktan kurtulmasını sağlar ve daha etkili bir şekilde diğer görevlerini gerçekleştirmesine olanak tanır. Interrupt yöntemi, yüksek hızlı veri iletimi veya **zamanlama hassasiyeti** gerektiren uygulamalarda tercih edilir.
- **DMA** yöntemi, veri transferini mikrodenetleyicinin müdahalesi olmadan doğrudan bellekten yapılmasını sağlar. DMA denetleyici, UART'dan gelen veya UART'a gönderilecek verileri doğrudan bellekten okur veya belleğe yazar. Bu yöntem, mikrodenetleyicinin UART veri transferiyle uğraşmadan diğer işlemleri gerçekleştirmesine olanak sağlar ve veri transferinde yüksek hızlı ve verimli bir çözümdür. DMA yöntemi, yüksek hızlı veri transferlerinde veya **sürekli veri akışı** gerektiren uygulamalarda kullanılır.
- Özellikle USART ile gönderilecek yüklü miktarda verimiz var ise, bu veriyi döngü içerisinde göndermek, işlemci zamanının önemli bir bölümünü harcayacaktır. Baud hızımız ne kadar az ise, gönderme hızımız o kadar düşecek, dolayısıyla bekleme hızımız da o kadar artacaktır.
- Gönderme işleminin bitmesini beklemeden işimize devam edebilmek için ya DMA ya da EXTI kullanırız.
- Interrupt kullanımında ise CPU tarafından saniyede çok sayıda kesinti yapılması gerekecektir. Bu yüzden çok etkili yöntem değildir. Verileri doğrudan belleğe yönlendirmek için DMA biriminin kullanılması en etkili yöntemdir.

## Birim Yapısı



## Register

| Offset | Register    | 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17                 | 16         | 15   | 14   | 13   | 12   | 11       | 10       | 9                  | 8    | 7        | 6        | 5    | 4    | 3   | 2   | 1  | 0  |
|--------|-------------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|--------------------|------------|------|------|------|------|----------|----------|--------------------|------|----------|----------|------|------|-----|-----|----|----|
| 0x00   | USART_SR    | Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |                    |            |      |      |      |      |          |          | CTS                | LBD  | TXE      | TC       | RXNE | IDLE | ORE | NF  | FE | PE |
|        | Reset value |          |    |    |    |    |    |    |    |    |    |    |    |    |    |                    |            |      |      |      |      |          |          | 0                  | 0    | 1        | 1        | 0    | 0    | 0   | 0   | 0  | 0  |
| 0x04   | USART_DR    | Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |                    |            |      |      |      |      |          |          | DR[8:0]            |      |          |          |      |      |     |     |    |    |
|        | Reset value |          |    |    |    |    |    |    |    |    |    |    |    |    |    |                    |            |      |      |      |      |          |          | 0                  | 0    | 0        | 0        | 0    | 0    | 0   | 0   | 0  | 0  |
| 0x08   | USART_BRR   | Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    | DIV_Mantissa[15:4] |            |      |      |      |      |          |          | DIV_Fraction [3:0] |      |          |          |      |      |     |     |    |    |
|        | Reset value |          |    |    |    |    |    |    |    |    |    |    |    |    |    | 0                  | 0          | 0    | 0    | 0    | 0    | 0        | 0        | 0                  | 0    | 0        | 0        | 0    | 0    | 0   | 0   | 0  |    |
| 0x0C   | USART_CR1   | Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    | OVER8              | Reserved   | UE   | M    | WAKE | PCE  | PS       | PEIE     | TXEIE              | TCIE | RXNEIE   | IDLEIE   | TE   | RE   | RWU | SBK |    |    |
|        | Reset value |          |    |    |    |    |    |    |    |    |    |    |    |    |    | 0                  | Reserved   | 0    | 0    | 0    | 0    | 0        | 0        | 0                  | 0    | 0        | 0        | 0    | 0    | 0   | 0   | 0  |    |
| 0x10   | USART_CR2   | Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    | LINEN              | STOP [1:0] |      | CKEN | CPOL | CPHA | LBCL     | Reserved | LBDE               | LBDL | Reserved | ADD[3:0] |      |      |     |     |    |    |
|        | Reset value |          |    |    |    |    |    |    |    |    |    |    |    |    |    | 0                  | 0          | 0    | 0    | 0    | 0    | 0        | 0        | Reserved           | 0    | 0        | 0        | 0    | 0    | 0   |     |    |    |
| 0x14   | USART_CR3   | Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    | ONEBIT             | CTSIE      | RTSE | DMAT | SCEN | NACK | HDSEL    | IRLP     | IREN               | EIE  |          |          |      |      |     |     |    |    |
|        | Reset value |          |    |    |    |    |    |    |    |    |    |    |    |    |    | 0                  | 0          | 0    | 0    | 0    | 0    | 0        | 0        | 0                  | 0    | 0        | 0        | 0    |      |     |     |    |    |
| 0x18   | USART_GTPR  | Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    | GT[7:0]            |            |      |      |      |      | PSC[7:0] |          |                    |      |          |          |      |      |     |     |    |    |
|        | Reset value |          |    |    |    |    |    |    |    |    |    |    |    |    |    | 0                  | 0          | 0    | 0    | 0    | 0    | 0        | 0        | 0                  | 0    | 0        | 0        | 0    | 0    | 0   |     |    |    |

- **USART\_SR** (Status Register), Bu kayıt, iletişim durumu hakkında bilgi sağlar. Örneğin, veri alımı veya iletimi tamamlandığında veya hata durumlarında flag içerir.
- **USART\_DR** (Data Register), iletişimde iletilen veya alınan veriyi tutar. Veriyi bu kayıta yazarak iletimi başlatabilir veya bu kayıttan okuyarak alınan veriyi elde edebilirsiniz.
- **USART\_BRR** (Baud Rate Register), iletişim hızını ayarlamak için kullanılır.
- **USART\_CR1** ve **USART\_CR2** (Control Register 1 ve 2), UART modunu, iletim ve alım parametrelerini ve diğer iletişim ayarlarını kontrol eder.
- **USART\_CR3** (Control Register 3), üçüncü kontrol kayıtdır ve DMA ayarları gibi daha gelişmiş ayarları kontrol eder.
- **USART\_GTPR** (Guard Time and Prescaler Register), zamanlama ayarları için kullanılır.