**UART DMA İçin FIFO Fonksiyonları**

Text

Description automatically generatedTüm fonksiyonları *“fifo.c”* ve *“fifo.h”* dosyaları içermektedir.

Figure 1 - fifo.h Dosyasının Kesiti

Figür 1’de tüm fonksiyonlar gösterilmektedir.

1. **FIFO Struct Değişkenleri**

*“BUF\_SIZE”* tanımlaması FIFO buffer’ının kaç adet *“DATA\_SIZE”* boyutunda veri saklayacağını belirtmektedir. *“DATA\_SIZE”* tanımlaması ise saklanacak verilerin boyutunu belirtmektedir.

Diagram

Description automatically generated with medium confidenceFIFO değişkenleri bir struct yapısı altında toplanmıştır. *“buf”* değişkeni direct olarak FIFO buffer’ının kendisidir. Uint8\_t tipinde tanımlanmasının sebebi gönderilecek verilerin bu tipte olacağının belirlenmesi üzerinedir.

Figure 2 - FIFO Diyagramı

Figür 2’de FIFO yapısı gösterilmiştir. *“head”* değişkeni FIFO’ya veri yazılırken buffer’ın hangi hücresinde olunduğunu belirtmektedir. *“tail”* değişkeni ise FIFO’dan veri okunurken hangi hücre üzerinde olduğunu belirtmektedir. *“size”* değişkeni FIFO’nun kaç byte olduğunu belirtmektedir. *“FIFOindex”* değişkeni FIFO’ya veri yazarken sıranın kaçıncı 32 byte’lık datada olduğunu gösterirken, *“UARTindex”* değişkeni FIFO’dan UART’a veri yazarken sıranın yazılacak olan kaçıncı 32 byte’lık datada olduğunu göstermektedir. 10 adet data figür 2’de gösterildiği biçimde arka arkaya sıralanacak şekilde FIFO’ya doldurulmaktadır.

1. **FIFO Fonksiyonları**
   1. **fifo\_init**

*“fifo\_init”* fonksiyonu fonksiyonların içerisinde eğer kod ilk defa çalıştırılıyorsa FIFO’nun değişkenlerine atama yaparak FIFO’yu init etmektedir. Main kod içerisinde çağrılmasına gerek yoktur.

* 1. **fifo\_init\_control**

*“fifo\_init\_control”* fonksiyonu diğer fonksiyonlar içerisinde FIFO’nun daha önce init edilip edilmediğini kontrol eden fonksiyondur. Main kod içerisinde çağırılmasına gerek yoktur.

* 1. **fifo\_read**

*“fifo\_read”* fonksiyonu FIFO’daki veriyi okuyan fonksiyondur. Her çağrıldığında FIFO’dan *“DATA\_SIZE”* kadar veri okur. UART üzerinden sadece yazma işlemi yapılacaksa main kod içerisinde çağrılmasına gerek yoktur.

* 1. **fifo\_write**

*“fifo\_write”* fonksiyonu FIFO’ya veri yazan fonksiyondur. Her çağrıldığında, verilen veriyi *“DATA\_SIZE”* kadar FIFO’ya yazar. Eğer verilen veri *“DATA\_SIZE”*dan daha küçük bir veri ise kalan byte’ları 0 ile doldurur. Eğer daha büyük bir veri ise sadece *“DATA\_SIZE”* kadar yazma yapar. Main kod içerisinde FIFO’ya veri yazılması gereken yerlerde çağırılabilir.

* 1. **UART\_fifo\_write**

*“UART\_fifo\_write”* fonksiyonu FIFO’daki verileri DMA ile UART üzerinden yazan fonksiyonudur. Main kod içerisinde FIFO’dan okunacak sıradaki verinin DMA ile UART üzerinden yazılması gereken yerlerde çağrılabilir.

Örnek proje içerisinde bu fonksiyon timer ile birlikte kullanılmıştır. Bu yüzden main kod içerisinde çağrılmasına gerek kalmamıştır.

1. **FIFO Fonksiyonları – Detaylı Anlatım**
   1. **Değişkenler**

*“DMA\_WRITE\_BUF”* buffer’ı FIFO’daki veriyi DMA ile UART’a basarken, FIFO’dan çekilen verinin DMA ile UART’a basılmadan önce aktarıldığı ara bir bufferdır. Bu buffer DMA fonksiyonuna verilerek aktarım sağlanır.

Text

Description automatically generated

Figure 3 - Değişkenler

*“fifoInit”* değişkeni FIFO’nun init edilip edilmediğini belirten değişkendir. FIFO init edilmediği durumda *“fifo\_init\_control”* fonksiyonu *“fifo\_init”* fonksiyonunu çağırır ve *“fifoInit”* değişkenini *“1”* yapar.

*“STATUS\_DMA”* değişkeni veri DMA ile UART’a yazıldıktan sonra DMA fonksiyonun return ettiği değeri saklamaktadır. Eğer return edilen değer *“0x02U”* ise DMA meşgul anlamına gelmektedir ve veri atlaması olmaması için FIFO’dan yeni veri okuması yapılmaz.

*“FIFO fifo”* tanımlanan struct yapısından bir obje oluşturmak için kullanılmıştır. Bütün FIFO değişkeni işlemleri bu obje üzerinden yapılmaktadır.

* 1. Text

     Description automatically generated**fifo\_init**

Figure 4 - fifo\_init Fonksiyonu

FIFO’ya ait değişkenlere initial atama yapılan fonksiyondur.

* 1. Graphical user interface, application, Word

     Description automatically generated**fifo\_init\_control**

Figure 5 - fifo\_init\_control Fonksiyonu

*“fifo\_init\_control”* fonksiyonu, diğer fonksiyonlar içinde FIFO’nun init edilip edilmediğini kontrol eden fonksiyondur.

* 1. Graphical user interface, text, application

     Description automatically generated**fifo\_read**

Figure 6 - fifo\_read Fonksiyonu

*“i”* değişkeni for içerisinde kullanılmak için oluşturulmuş dummy bir değişkendir. *“bufIndex”* değişkeni, *“i”* değişkeninden bağımsız olarak buffer’ın mevcut hücresini tanımlamak için kullanılmıştır. FIFO’dan okuma veya yazma yaparken 32 byte’lık veriler halinde yapılabilmesi adına *“i”* değişkeni *“UARTindex”* ve *“DATA\_SIZE”*’ın çarpımına eşitlenmiştir. Bir sonraki veri paketine kadar işlem yapılması adına *“UARTindex+1”* ile *“DATA\_SIZE”*’ın çarpımına kadar döngü devam etmektedir.

*“tail != head”* ifadesi FIFO’ya veri yazıldığı anlamına gelmektedir. Eğer eşitlik var ise FIFO’da okunacak veri kalmamıştır demektir. Bu yüzden *“tail != head”* olduğu durumda FIFO’dan okuma işlemi yapılmıştır. FIFO’nun boyutu *“BUF\_SIZE”* değişkeni ile sınırlandırılmıştır. Buffer dolduktan sonra yeni veriler tekrar başa yazılacaktır. Bu yüzden UARTindex son veri’ye eşit olduğunda sıfırlanmıştır.

* 1. **fifo\_write**

*“i”* ve *“bufIndex”* değişkenleri için *“fifo\_read”* fonksiyonunda yazılanların aynısı geçerlidir. Aynı şekilde for döngüsü parametreleri için de *“fifo\_read”* fonksiyonunda yazılanlar geçerlidir. *“fifo\_read”* fonksiyonundan farklı olarak yapılan ilerlemeler *“head”* değişkeni üzerinden yapılmaktadır. Bu sayede okuma yapılacağı zaman *“head”* ve *“tail”* değişkenlerinin eşit olup olmama durumu üzerinden ilerlenebilir.

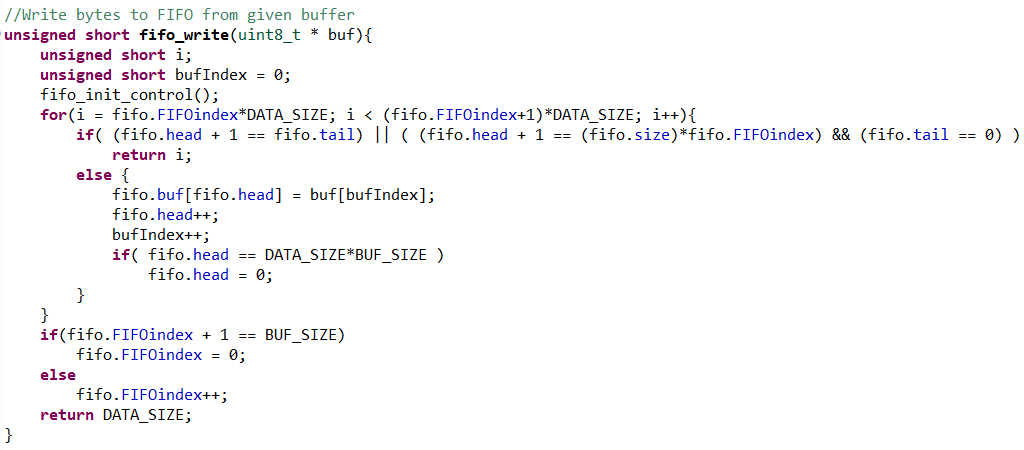


Figure 7 - fifo\_write Fonksiyonu

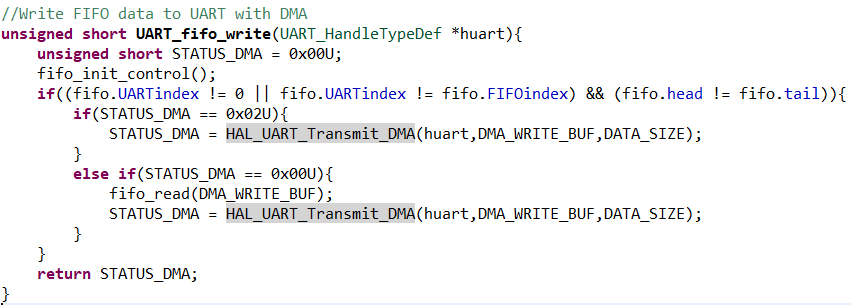
* 1. **UART\_fifo\_write**

Figure 8 – UART\_fifo\_write Fonksiyonu

*“STATUS\_DMA”* değişkeni *“HAL\_UART\_Transmit\_DMA”* fonksiyonun geri döndürdüğü değeri kaydetmektedir. Eğer DMA veri göndermek için müsaitse *“HAL\_OK”* veya değilse *“HAL\_BUSY”* gibi değerleri geri döndürmektedir. *“HAL\_OK”* için 0x00U ve *“HAL\_BUSY”* için 0x02U değerleri karşılık gelmektedir.

İlk if bloğunun içinde çift taraflı controller yapılmaktadır. Eğer *“head”* ve *“tail”* eşit değilse sırada UART’a basılacak veri vardır, veya *“UARTindex”* ile *“FIFOindex”* eşit değilse ve *“UARTindex”* 0 değilse gönderilmemiş veri paketi vardır gibi. Bu bloğun içerisinde *“STATUS\_DMA”* kontrolü yapılmaktadır. Eğer ilk defa veri gönderilecekse DMA meşgul değildir. Bu yüzden initial değer olarak *“STATUS\_DMA”* değişkenine 0x00U değeri atanmıştır. Yani ilk kez veri UART’a basılacağı zaman kod else if bloğuna girer ve FIFO’dan veri okuyarak ara buffer olan *“DMA\_WRITE\_BUF”* bufferına verileri alır. Sonrasında *“HAL\_UART\_Transmit\_DMA”* fonksiyonunu kullanarak DMA ile verileri UART’a basar. Return olarak DMA’in durumunu da alarak *“STATUS\_DMA”* değişkenine atar. Eğer DMA meşgulse *“HAL\_UART\_Transmit\_DMA”* fonksiyonu geri 0x02U değerini döndürecektir. Bu durumda kod ilk if bloğuna girer. Veri kaybı olmaması adına FIFO’dan yeni veri çekilmez ve DMA’in mevcut durumunu öğrenmek adına *“HAL\_UART\_Transmit\_DMA”* fonksiyonu tekrar çağrılır. Eğer bu süre zarfında *“HAL\_UART\_Transmit\_DMA”* fonksiyonu müsait duruma geçmişse veri DMA ile basılır. Değilse DMA müsait olana kadar bu döngü tekrarlanır.

* 1. **Timer**

Kod timer ile birlikte çalışmaktadır. Örnek kod içerisinde TIM3 kullanılmıştır. Timer fonksiyonu içerisinde *“UART\_fifo\_write”* fonksiyonu çağırılmaktadır. Örnek kod içerisinde UART2 kullanılmıştır. Bu yüzden *“UART\_fifo\_write”* fonksiyonuna parameter olarak *“&huart2”* verilmiştir.

Text

Description automatically generated

Figure 9 - Timer Fonksiyonu

Kodun timer’a girdiğinden görsel olarak da emin olunması adına fonksiyon içerisinde LED3 yakılmıştır.

* 1. **DMA RX Complete Callback**

Text

Description automatically generatedDMA ile UART üzerinden veri aktarımı tamamlandığı zaman kod *“HAL\_UART\_TxCpltCallback”* fonksiyonuna girer. Veri aktarımının tamamlandığından emin olunduktan sonra *“UARTindex”* değişkeni artırılmaktadır. Bu yüzden bu işlem *“HAL\_UART\_TxCpltCallback”* fonksiyonu içinde yapılmıştır.

Figure 10 - HAL\_UART\_TxCpltCallback Fonksiyonu

*“HAL\_UART\_TxCpltCallback”* fonksiyonuna kodun girdiğinden görsel olarak emin olunması adına LED1 yakılmıştır.

* 1. **Main Kod**

Örnek projede 9 adet dummy veri hazırlanmıştır. Her bir veri 32byte’dır. Ayrıca *“fifo.c”* dosyasında oluşturulan *“FIFO”* tipindeki *“fifo”* objesinin de main kod içerisinde extern edilmesi gerekmektedir (*“HAL\_UART\_TxCpltCallback”* fonksiyonunda kullanıldığı için).

Text

Description automatically generated

Figure 11 - Main Koddaki Veriler ve Extern Değişkenler

Text, table

Description automatically generated with medium confidenceFigür 10’daki görselde verilerin tamamı gösterilmemiştir. Örneğin *“data1”* bufferında 32 adet 0x1 verisi bulunmaktadır fakat görselin fazla yer kaplamaması adına sadece birkaç tanesi gösterilmiştir.

Figure 12 - Main Kod İçerisindeki While Döngüsü

Main kod içerisindeki while içinde veriler 2 milisaniye aralıklarla FIFO’ya yazılmıştır. Kodu test etmek amaçlı böyle bir şey yapılmıştır. Kodun while içerisinde döndüğünün görsel olarak da anlaşılabilmesi adına LED2 yakılmıştır.

Kod main içindeki while’da 2 milisaniye aralıklarla verileri sürekli olarak FIFO’ya yazarken, yaklaşık 1 milisaniye aralıklara timer çalışmaktadır. Her 1 milisaniyede bir timer fonksiyonuna girerek sırada DMA üzerinden UART’a yazılacak verinin olup olmadığı kontrol edilmektedir. Timer süresi 10 milisaniyede de denenmiştir ve kod yine düzgün bir şekilde çalışmıştır.

1. **İnceleme**

Kodun düzgün çalışıp çalışmadığı hem RealTerm ile hem de logic analyzer ile test edilmiştir.

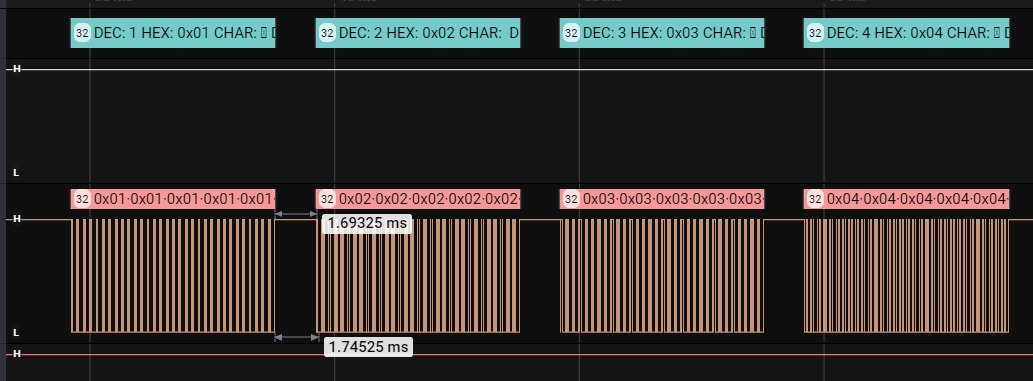


Figure 13 - Logic Analyzer Sonucu

Table

Description automatically generatedFigür 13’de görüldüğü gibi veriler kodda gönderildiği sıra ile UART’a basılmıştır. Timer süresi yaklaşık 1 ms ayarlanmıştır. Figür 13’de de veri aktarımı arasındaki mesafenin 1ms’ye yakın olduğu görülebilir.

Figure 14 - RealTerm Sonucu

Figür 14’de de görüleceği üzere veriler gönderildiği şekilde UART’a basılmıştır.