Hazırlayan: 2 1712709999 Adem ADEMOĞLU

EEM465 Ödev 1

Çalışma:

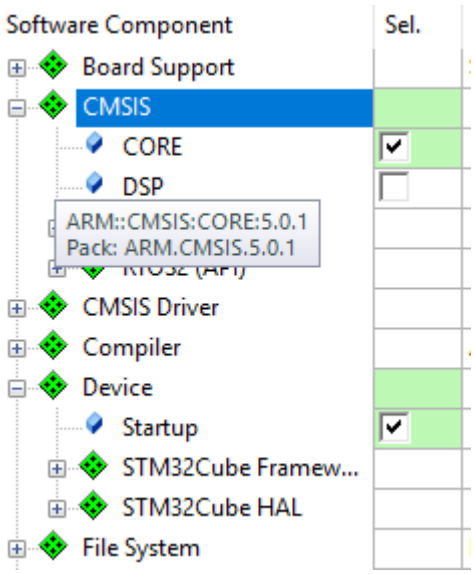
Bu çalışmada RCC yazmaçları kullanılarak mikrokontrolcünün çevrim (clock) kaynağı harici kaynak olarak ayarlanmıştır. D portunun 15 ve 14. pinine bağlı olan LEDlerin sırasıyla yanması için programlama yapılmıştır. Yanma süreleri bir delay fonksiyonu yazılarak sağlanmıştır. Yapılan kodlama, ekran görüntüleri ve simülasyon sonuçları gösterilmiştir. (Buna benzer şekilde yazdığınız programın ne yaptığını yazmanız gerekmektedir.)

Kodlama:

```
#include "stm32f407xx.h"                  // Device header
void delay(void){
          int i;
          for(i=0;i<1000000;i++);
}
int main(void){

          RCC->CR |= RCC_CR_HSEON;
          while( !(RCC->CR & RCC_CR_HSERDY));
          RCC->CFGR |= RCC_CFGR_SW_HSE;

          RCC->AHB1ENR |= RCC_AHB1ENR_GPIODEN;
          GPIOD->MODER |= GPIO_MODER_MODE15_0;
          GPIOD->MODER |= GPIO_MODER_MODE14_0;

          while(1){
                    GPIOD->ODR |= GPIO_ODR_OD14;
                    GPIOD->ODR &= ~GPIO_ODR_OD15;
                    delay();
                    GPIOD->ODR |= GPIO_ODR_OD15;
                    GPIOD->ODR &= ~GPIO_ODR_OD14;
                    delay();
          }

          return 1;
}
```

Açıklama:

Projeye Şekil 1'de gösterilen CMSIS CORE ve Device Startup kütüphaneleri eklenmiştir.



Şekil 1: Proje başlangıcında eklenen kütüphaneler.

Çalışma Keil uVision5 programı kullanılarak C dilinde kodlanmıştır. Şekil 2'de yazmaçların tanımlamalarının olduğu stm32f407xx.h header dosyasının eklendiği gösterilmektedir. Ayrıca 3 ile 6. satır arasında LEDlerin yanma süreleri için delay() fonksiyonu yazılmıştır. delay() fonksiyonu i değişkeninin değerinin 0'dan bir milyona kadar arttırılması ile elde edilmiştir.

```
1  #include "stm32f407xx.h"
2
3  void delay(void){
4     int i;
5     for(i=0;i<1000000;i++);
6  }
```

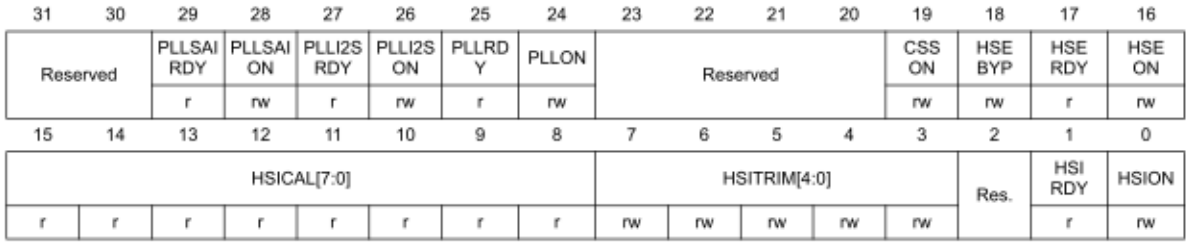
Şekil 2: Eklenilen kütüphane ve delay fonksiyonu.

Şekil 3'te programın ana fonksiyonu gösterilmektedir. Ana fonksiyonda öncelikle harici çevrim kaynağının mikrokontrolcünün ana çevrim kaynağı olması için kodlamalar yapılmıştır. Bu kodlamalarda Şekil 4'te gösterilen RCC_CR yazmacı kullanılmıştır. 10. satırda RCC_CR yazmacının 16. biti 1 yapılmaktadır. Daha sonra RCC_CR_HSERDY (yazmacın 17. biti) bitinin 1 olması while döngüsünde beklenmektedir.

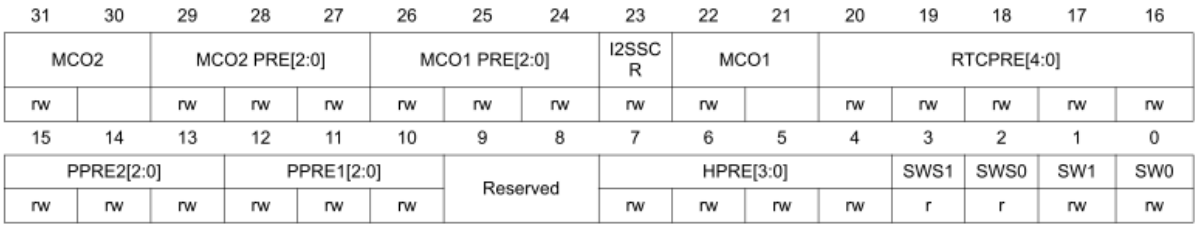
12. satırda ise harici çevrim kaynağı hazır olduktan sonra Şekil 5'te gösterilen RCC_CFGR yazmacının 0. bitine 1 yazılarak harici çevrim kaynağının mikrokontrolcünün ana çevrim kaynağı olması sağlanmaktadır.

```
 8  int main(void){
 9
10    RCC->CR |= RCC_CR_HSEON;
11    while( !(RCC->CR & RCC_CR_HSERDY));
12    RCC->CFGR |= RCC_CFGR_SW_HSE;
13
14    RCC->AHB1ENR |= RCC_AHB1ENR_GPIODEN;
15    GPIOD->MODER |= GPIO_MODER_MODE15_0;
16    GPIOD->MODER |= GPIO_MODER_MODE14_0;
17
18    while(1){
19      GPIOD->ODR |= GPIO_ODR_OD14;
20      GPIOD->ODR &= ~GPIO_ODR_OD15;
21      delay();
22      GPIOD->ODR |= GPIO_ODR_OD15;
23      GPIOD->ODR &= ~GPIO_ODR_OD14;
24      delay();
25    }
26
27    return 1;
28  }
```

Şekil 3: Programın main fonksiyonu.



Şekil 4: RCC_CR yazmacının yapısı.



Şekil 5: RCC_CFGR yazmacının yapısı.

14. satırda D portunun clock sinyali aktif hale getirilmektedir. Bu işlem için Şekil 6'da yapısı gösterilen RCC_AHB1ENR yazmacının 3. biti 1 yapılmıştır.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reser-ved	OTGHS ULPIEN	OTGHSEN	ETHMACPTPEN	ETHMACRXEN	ETHMACTXEN	ETHMACEN	Res.	**DMA2DEN**	DMA2EN	DMA1EN	CCMDATARAMEN	Res.	BKPSRAMEN	Reserved	
	rw	rw	rw	rw	rw	rw		rw	rw	rw			rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CRCEN	Res.	GPIOKEN	GPIOJEN	GPIOIEN	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
			rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Şekil 6: RCC_AHB1ENR yazmacının yapısı.

Bu işlem Şekil 3'te gösterilen programın 15 ve 16. satırlarında D portunun 14 ve 15. pinlerinin output modunda çalışması için programlama yapılmaktadır. Şekil 7'de GPIOx_MODER yazmacının yapısı gösterilmektedir. Yapılan programlama ile bu yazmacın 30 ve 28. bitleri 1 yapılmıştır.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

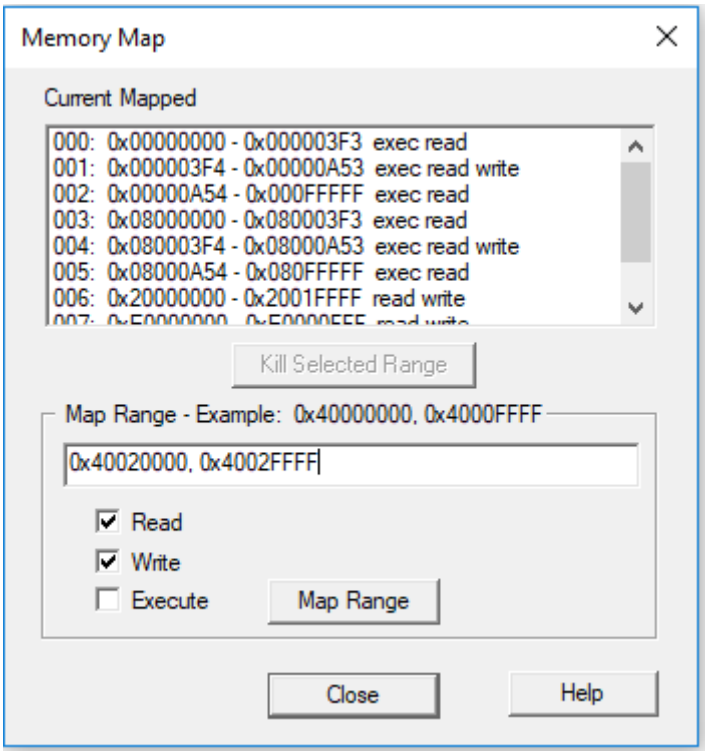
Şekil 7: GPIOx_MODER yazmacının yapısı.

Şekil 3'te gösterilen programın 18. satırından itibaren sonsuz bir döngünün içinde Şekil 8'de yapısı gösterilen D portu GPIOx_ODR yazmacının öncelikle 14. bit 1 yapılarak D portunun 14. pinine lojik 1 seviyesinde voltaj verilmiş oluyor. Daha sonra 20 satırda 15. biti 0 yapılıyor. delay süresi kadar beklendikten sonra 22. satırda 15. biti 1 yapılıyor ve 23. satırda 14. bit 0 yapılarak yine delay süresi kadar bekleniyor. Sonsuz while döngüsünün içerisinde 14. ve 15. pinlere bağlı olan LEDlerin sıra ile yanması ve sönmesi sağlanıyor.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Şekil 8: GPIOx_ODR yazmacının yapısı.

Programın debug edildiğinde bazı adres bölgelerinin okuma izni olmadığı için Şekil 9'da gösterildiği gibi 0x40020000, 0x4002FFFF aralığındaki adres bölgesi için okuma ve yazma izni Debug, Memory Map menüsünden eklenmiştir.

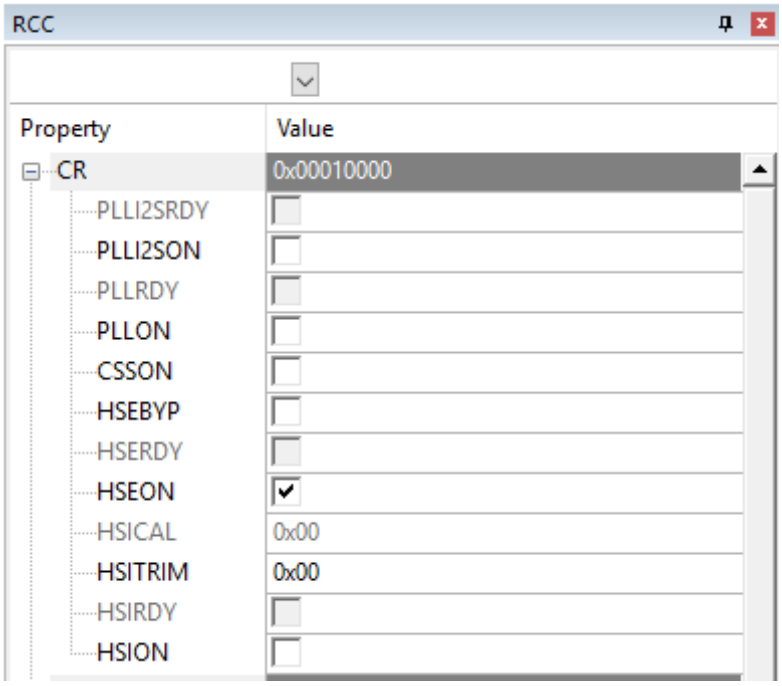


Şekil 9: Bellek Bölgelerinin okuma ve yazma izni için eklenmesi.

Debug işlemi simülasyon modunda programın 11. satırındaki while döngüsünde takılı kaldığı için bu satır yoruma alınarak etkisiz hale getirilmiştir.

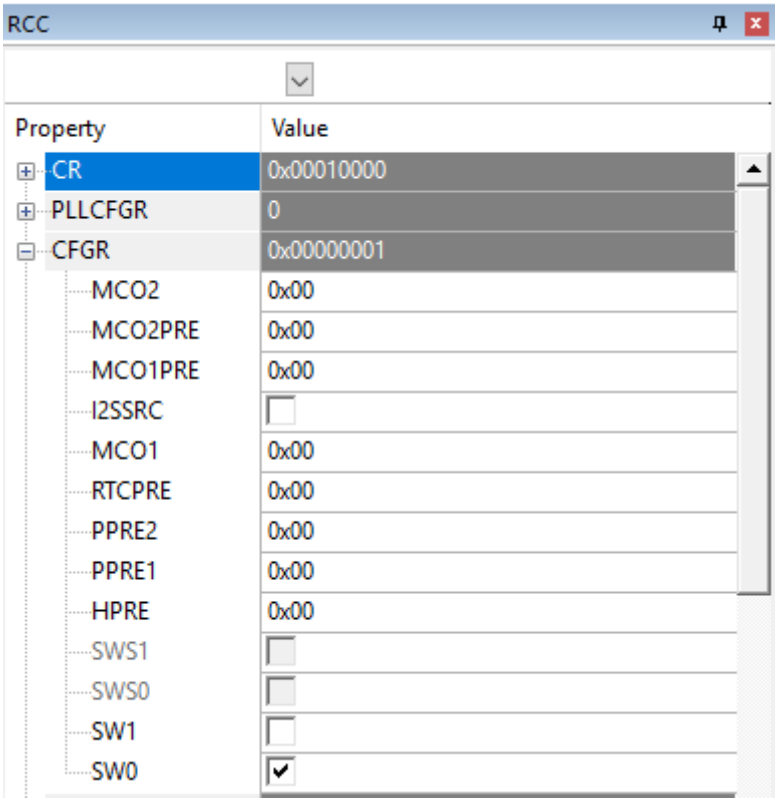
Simülasyon sonuçlarını görmek için Peripheral menüsünden RCC ve GPIOD pencereleri aktif edilmiştir.

10. satırdaki kod yürütüldüğünde Şekil 10'da RCC'nin Control Register (CR) HSEON bitinin 1 yapıldığı görülmektedir. Böylece program cihazın harici çevrim kaynağını aktif hale getirmiştir.



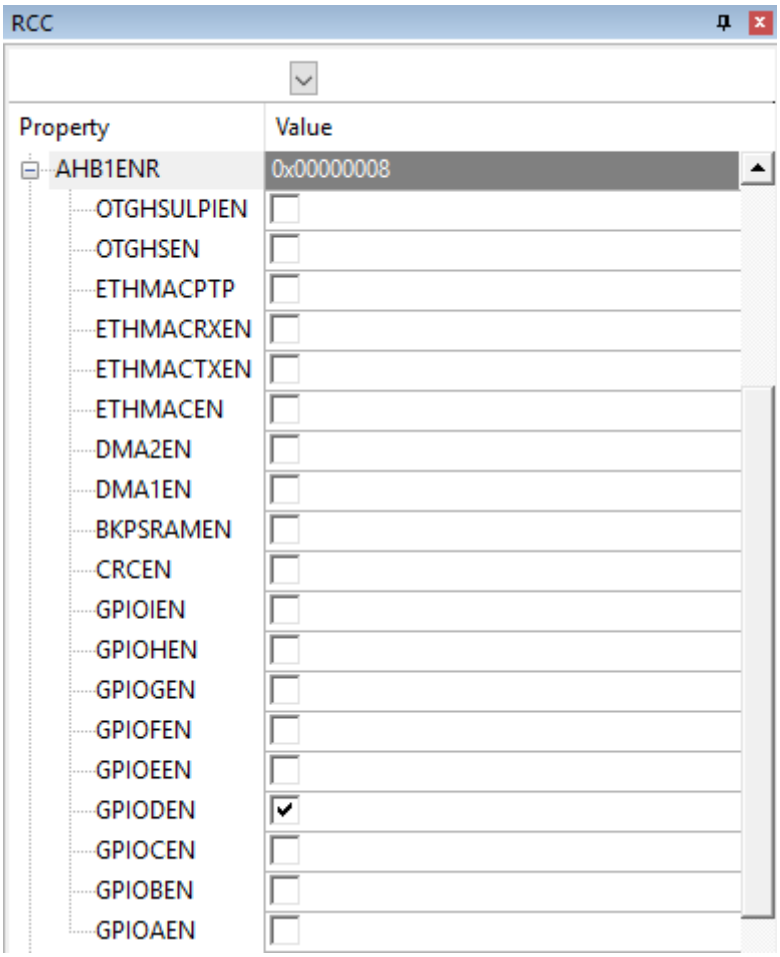
Şekil 10: RCC diyalog penceresi, CR yazmacı.

12. satırdaki kod yürütüldüğünde Şekil 11'de gösterildiği gibi RCC_CFGR yazmacının 1. biti 1 yapılmaktadır. Böylece harici çevrim kaynağı ana çevrim kaynağı olarak seçilmektedir.



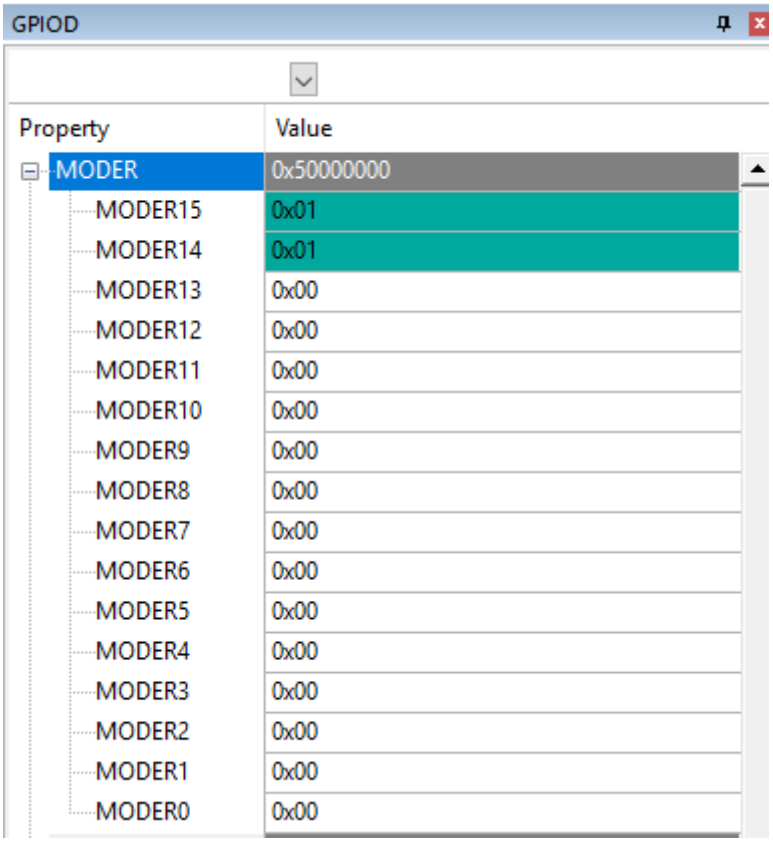
Şekil 11: RCC diyalog penceresi, CFGR yazmacı.

Programın 14. satırında D portunun aktif hale gelmesi için clock sinyalinin verildiği Şekil 12'de gösterilmektedir.



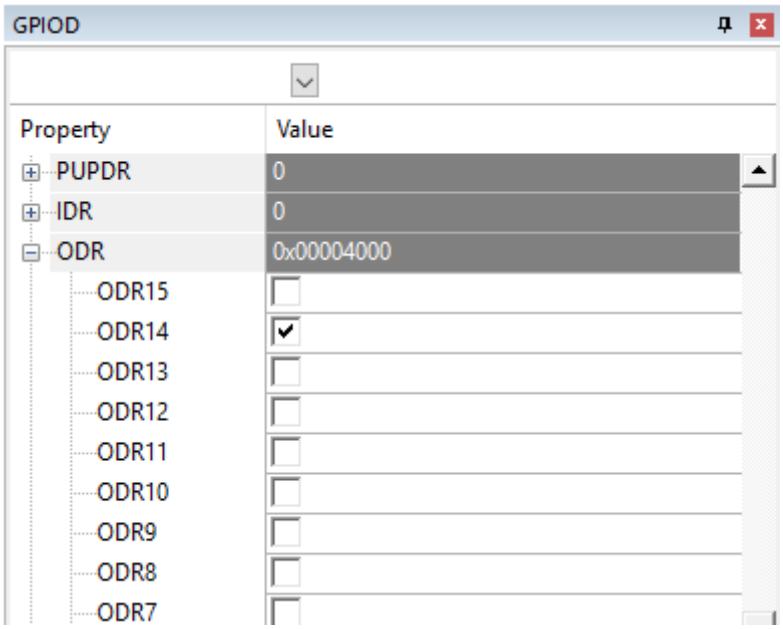
Şekil 12: RCC diyalog penceresi, AHB1ENR yazmacı.

Şekil 13'de GPIOD MODER yazmacına yapılan ayarlamalar gösterilmiştir. Bu yazmacın 15 ve 14. pinleri için ayrılmış bölgelere 01 yazıldığı gösterilmektedir. Böylece bu pinler output modunda çalışmaktadır.

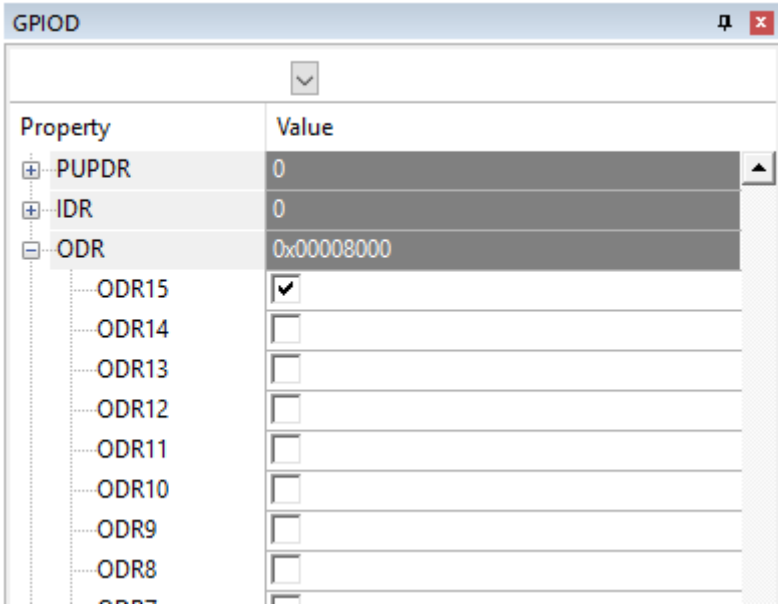


Şekil 13: GPIOD diyalog penceresi, MODER yazmacı.

Şekil 14 ve Şekil 15'te programın 19, 20, 22, 23. satırlarında gösterilen programlama sonucunda pinlerin 14 ve 15. pinlerin 1 ve 0 yapılması durumları gösterilmektedir.



Şekil 14: GPIOD diyalog penceresi, ODR yazmacı.



Şekil 15: GPIOD diyalog penceresi, ODR yazmacı.