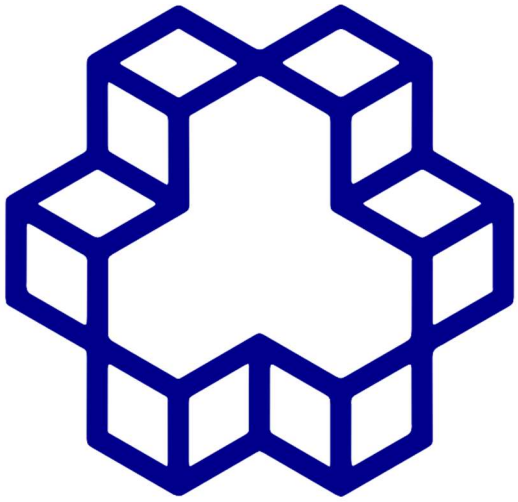


دانشگاه صنعتی خواجه نصیرالدین طوسی

گزارشکار پروژه شماره ۱

درس سیستم‌های دیجیتال ۲

آنائیس گل‌بوداغیانس ۴۰۱۲۲۱۱۳

چکیده

در این پروژه خواسته شده بود تا برنامه یک بازی‌ای به میکروکنترلر ATMEGA64 داده شود. بازی دوتا بازیکن دارد و بازیکن اول عددی را در پورت D ذخیره می‌کند و بازیکن دوم باید آن عدد را با ذخیره‌سازی در پورت A حدس بزند. بازی در دو شکل به پایان می‌رسد.

از Proteus برای شبیه‌سازی مدار، برای اجرا، دیباگ و ترجمه کد از AtmelStudio استفاده شده است.

ویدیو کلیپی کوتاه از اجرای شبیه‌ساز مدار نیز ضمیمه شده است.

توضیحات کد بخش اصلی

```
;AVR Project
; Anaies Golboudaghians 40122113
;Project no.1

.Include "M64DEF.INC"
.ORG 0x0000
JMP Main
.ORG 0x000C
JMP INT5_ISR
.ORG 0x0020
JMP ready_1s
.ORG 0x0050
```

ابتدا آدرس‌دهی‌های لازم انجام می‌شود. Ready_1s همان تایمر 0 هست.

تعریف استک‌پوینتر:

```
Main:
    LDI R20, low(RAMEND)
    OUT SPL, R20
    LDI R20, high(RAMEND)
    OUT SPH, R20
```

اجازه فعالیت وقفه ۵ و حساسیت آن به لبه پایین‌رونده:

```
    LDI R20, 0x20
    OUT EIMSK, R20 ; to active INT5
    LDI R20, 0x08
    OUT EICRB, R20 ; falling edge for INT5
```

تعریف پورت‌های ورودی و خروجی:

```
    LDI R20, 0xFF ; defining port C as an output
    OUT DDRC, R20
    LDI R20, 0x00 ; defining port A and D as an input
    OUT DDRA, R20
    OUT DDRD, R20
```

فرکانس کاری پردازنده 4.096MHz در نظر گرفته شده است. در خطوط بعدی، تنظیمات تایمر را اعمال می‌کنیم.

```
    ;initialising the timer
    LDI R20, 1
    OUT TIMSK, R20
    LDI R20, 7 ;1024, prescaler
    OUT TCCR0, R20
    LDI R20, 56 ; Fcpu = 4.096MHz, Ftimer = 4KHz,
```

```
    ;Ttimer=250micros, 50ms/250micsec = 200, 256-200=56
    OUT TCNT0, R20
```

تنظیم اولیه و پاک کردن رجیسترهای مورد نیاز:

```
    LDI R16, 10 ; counter for player no.2 turns

    CLR R21
    CLR R22

    CLR R24; flag for starting
    CLR R25; flag for end of the game
```

در مجموع از ۶ رجیستر استفاده شده که کاربرد آن در جدول زیر آمده است.

کاربرد	رجیستر
شمارنده نوبت‌های بازی	R16
مقداردهی‌های اولیه و تنظیمات روشن کردن LEDها	R20
ساختن ۱ ثانیه	R21
ساختن ۵ ثانیه	R22
پرچم شروع بازی و ذخیره کردن مقدار PORTA	R24
پرچم پایان بازی و ذخیره کردن مقدار PORTD	R25

در ادامه حلقه اصلی و دستور پذیرش وقفه را می‌نویسیم. بایستی جایی شروع بازی را تشخیص دهیم. می‌توانستیم این کار را در حلقه اصلی انجام دهیم اما در این صورت با پذیرش وقفه تایمر به مشکل برمی‌خوردیم. پس شروع بازی را بعد از اینکه یک ثانیه را ساختیم تشخیص می‌دهیم.

```
    SEI
Loop:
    JMP Loop
```

در وقفه شماره ۵، پرچم شروع بازی را ست می‌کنیم.

```
INT5_ISR:
    LDI R24, 1 ; to show that the game has started
    RETI
```

حال، لیبل تایمر را تعریف می‌کنیم و ۱ ثانیه را می‌سازیم.

```
ready_1s:
    LDI R20, 56 ; Fcpu = 4.096MHz, Ftimer = 4KHz,
    ;Ttimer=250micros, 50ms/250micsec = 200, 256-200=56
    OUT TCNT0, R20
```

```
    INC R21
    CPI R21, 20 ; to make 1s
    BRNE ev_1s_end
    CALL checkFlag
ev_1s_end:
    RETI
```

همان‌گونه که گفته شد، پرچم شروع بازی این‌جا چک می‌شود؛ با این کار می‌توانیم از پذیرش وقفه حین بازی جلوگیری کنیم.

```
checkFlag:
    CPI R24,1
    BRNE checkFlag_end
    CALL ready_5s
checkFlag_end:
    RET
```

اگر پرچم ست نباشد، برمی‌گردیم از اول و دوباره یک ثانیه را می‌سازیم. اگر پرچم ست شده باشد، می‌رویم که پنج ثانیه بسازیم و شروع بازی را فرابخوانیم. ما در پایان و حین بازی به این پنج ثانیه نیاز داریم پس برنامه را طوری می‌نویسیم که در هر دو حالت قابل استفاده باشد.

پس اول لازم است چک کنیم در پایان بازی هستیم یا خیر.

```
ready_5s:
    CPI R25, 1
    BREQ forEnd ;to check if the game has ended or not
    PUSH R20
    LDI R20, 0b11111110
    OUT PORTC, R20
    POP R20
```

رجیستر بیستم را پوش و پاپ می‌کنیم تا بتوانیم از آن برای روشن کردن LED سفید استفاده کنیم.

```
forEnd:
    CLR R21
    INC R22
    CPI R22, 5 ; to make 5s, we need 5s while playing and ending the game
    BRNE ready_5s_end
    CPI R25, 1 ;to check if the game has ended or not
    BREQ clearAll ;ending the game
    CALL play ;starting the game
ready_5s_end:
    RET
```

اگر در پایان بازی قرار داشتیم، clearAll فراخوانده می‌شود تا شمارنده‌های زمانی را clear و سایر رجیسترها را به تنظیمات اولیه برمی‌گردانیم. در غیر این‌صورت، بازی شروع می‌شود.

```
play:
    PUSH R24 ; putting flags in the stack
    PUSH R25
    IN R24, PINA ;reading the input ports
    IN R25, PIND
    CP R24, R25
    POP R25
    POP R24
    BRLO red
    BREQ green
```

پورت‌های ورودی را می‌خوانیم. اگر در هیچ یک از حالت‌های قرمز یا سبز نباشیم، خط بعدی لیبل زرد تعریف شده که مربوط به بزرگ‌تر بودن عدد حدسی بازیکن از عدد اصلی است.

```
yellow:
    PUSH R20
    LDI R20, 0b11111011
    OUT PORTC, R20
    POP R20
    JMP tryagain
```

چه در حالت زرد و چه قرمز، باید به حالت tryagain برویم تا بازیکن فرصت مجدد داشته باشد. این فرصت‌ها حداکثر ده‌بار می‌باشند.

```
green:
    CLR R21
    CLR R22 ; they should be cleared here too, otherwise it ends so late (after
4-5min)
    LDI R16, 1 ; to prevent entering the tryagain loop when the player has won
    PUSH R20
    LDI R20, 0b11111101
    OUT PORTC, R20
    POP R20
    JMP EndGame
```

در این حالت که بازیکن برنده شده، باید برای جلوگیری از تاخیر رجیسترهای زمانی را پاک کنیم. ۱ را در رجیستر ۱۶ می‌ریزیم تا از تاخیرات ناخواسته جلوگیری کنیم.

```
red:
    PUSH R20
    LDI R20, 0b11110111
    OUT PORTC, R20
```

```
    POP R20
```

بعد از این بخش، قاعدتاً بخش تلاش مجدد می‌آید.

```
tryagain:
    CLR R21
    CLR R22
    DEC R16
    BRNE ready_1s
```

اگر بازیکن در ۱۰ شانس خود موفق به حدس عدد نشود، بخش loss فراخوانده می‌شود. بخشی که باید تمام چراغ‌ها روشن شود و پس از ۵ ثانیه خاموش. چراغ‌ها باید active-low باشند. یعنی با صفر روشن شوند و به عبارتی در مدار آن‌ها، سر دیگر آن به مقاومت و Vcc وصل باشند.

```
loss:
    LDI R17, 0b11110000
    OUT PORTC, R17

EndGame:
    LDI R25, 1
    JMP ready_1s
```

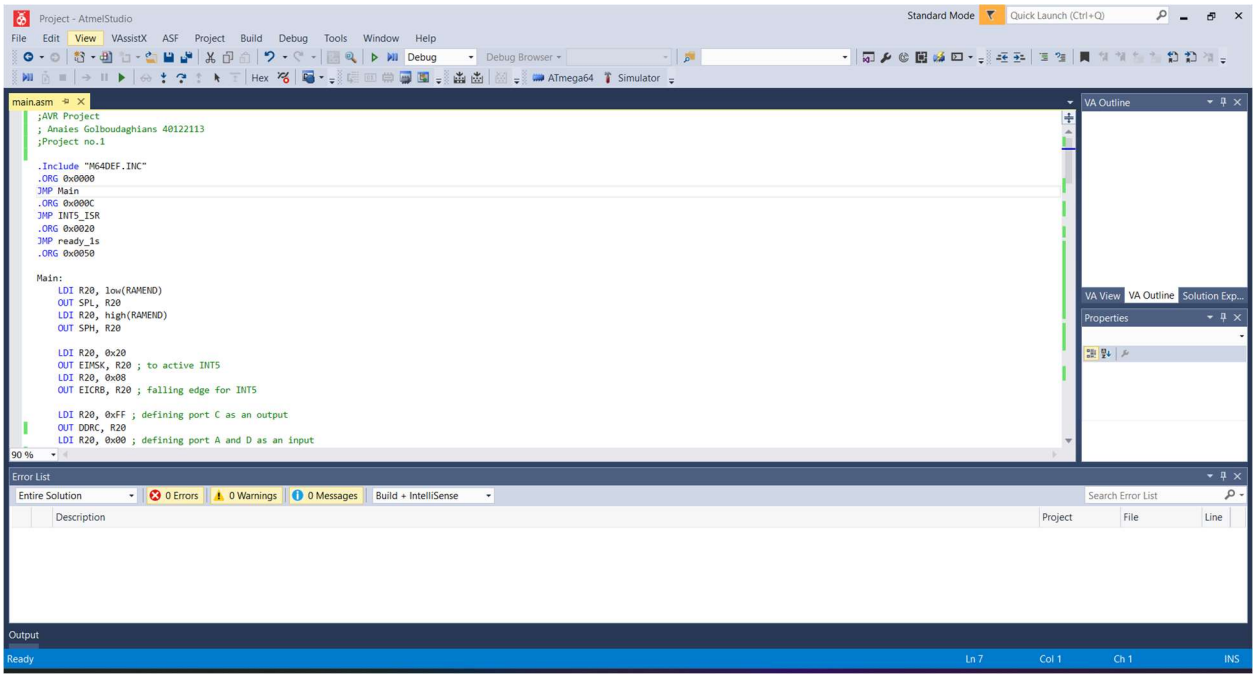
پرچم پایان بازی ست می‌شود و می‌رویم تا یک بار دیگر ۵ ثانیه را بسازیم و پس از آن، LEDها را خاموش می‌کنیم.

```
clearAll:
    LDI R16, 10 ; counter for player no.2 turns
    PUSH R20
    LDI R20, 0x0FF
    OUT PORTC, R20
    POP R20
    CLR R21
    CLR R22
    CLR R24 ; clearing the start flag, to wait for the next game
    CLR R25 ; clearing the end flag
    RET
```

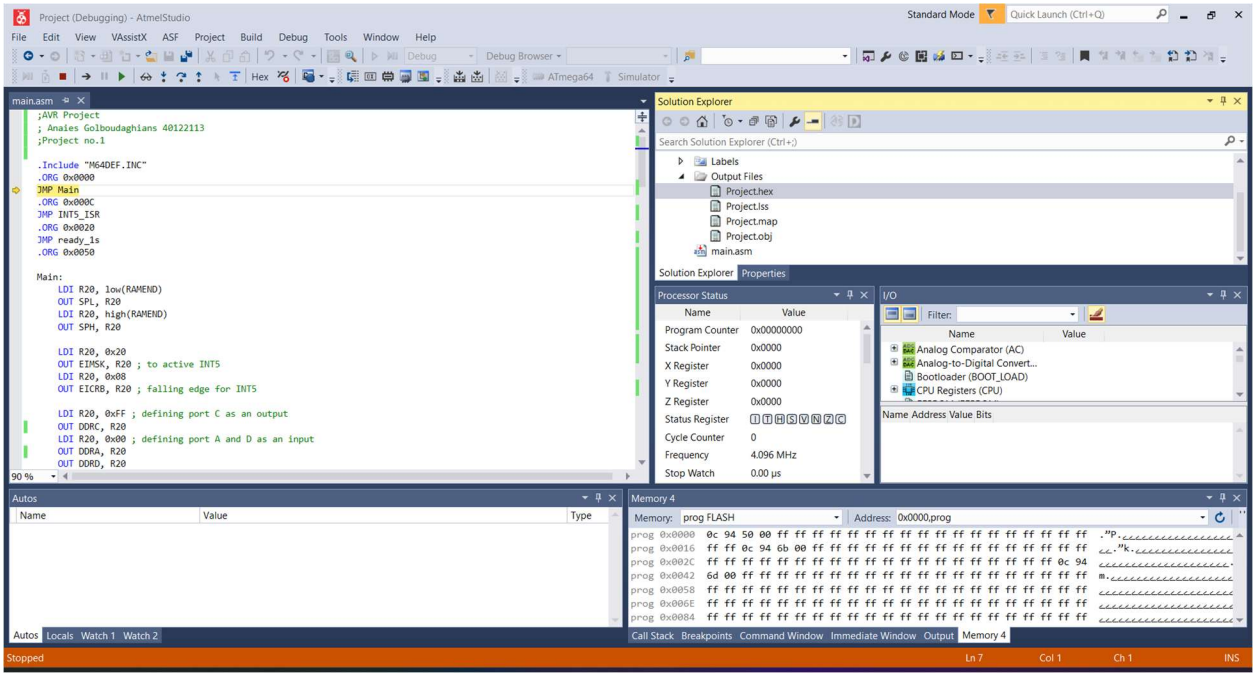
در بخش آخر پرچم و شمارنده‌های زمانی را پاک می‌کنیم و شمارنده نوبت‌ها را دوباره به ۱۰ بار تنظیم می‌کنیم. هم‌چنین، تمام LEDها را خاموش می‌کنیم.

دیباگ کد

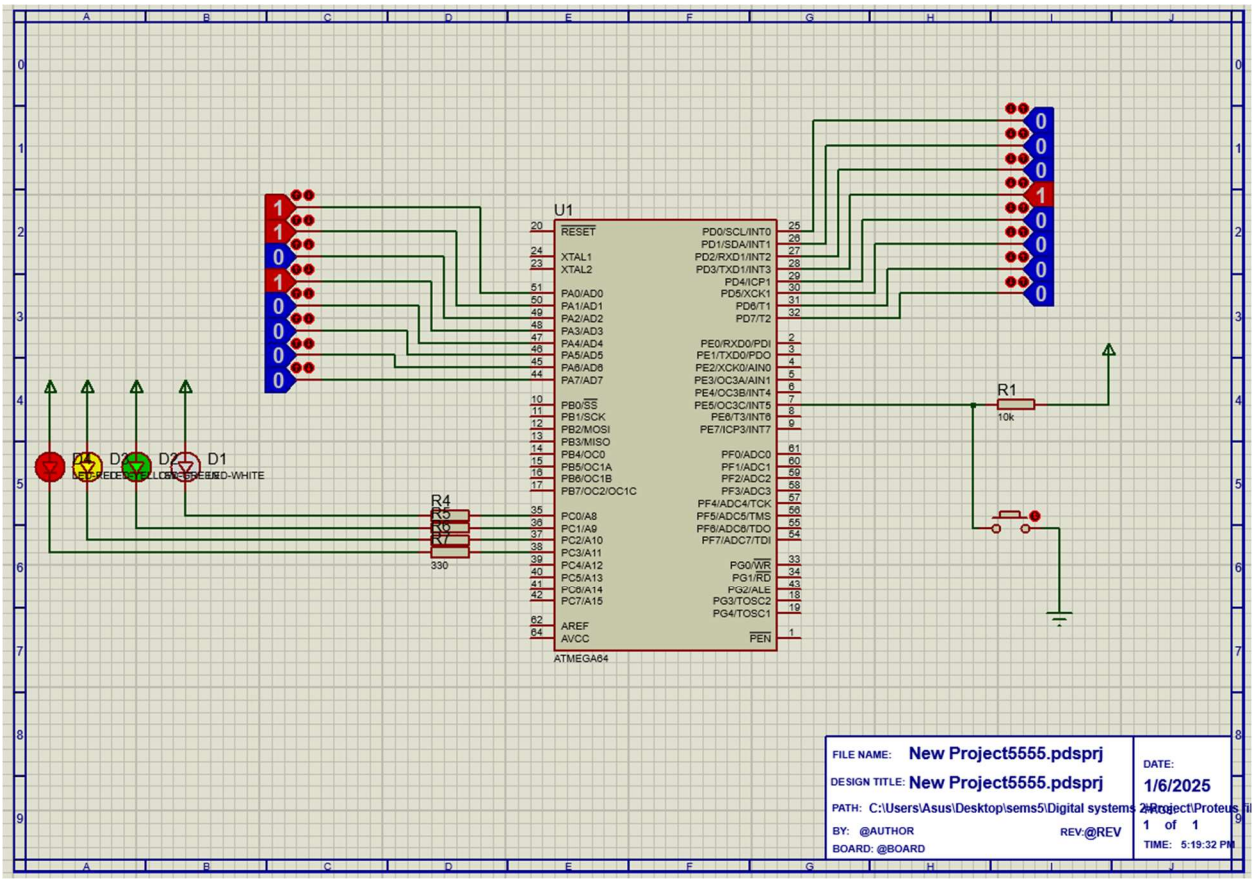
کد را به محیط AtmelStudio می‌بریم. ابتدا باید پروژه جدید را ایجاد کنیم و ATMEGA64 را انتخاب کنیم.



سپس گزینه build و سپس build solution را انتخاب می‌کنیم، حالت simulator را تنظیم می‌کنیم و دیباگ می‌کنیم.

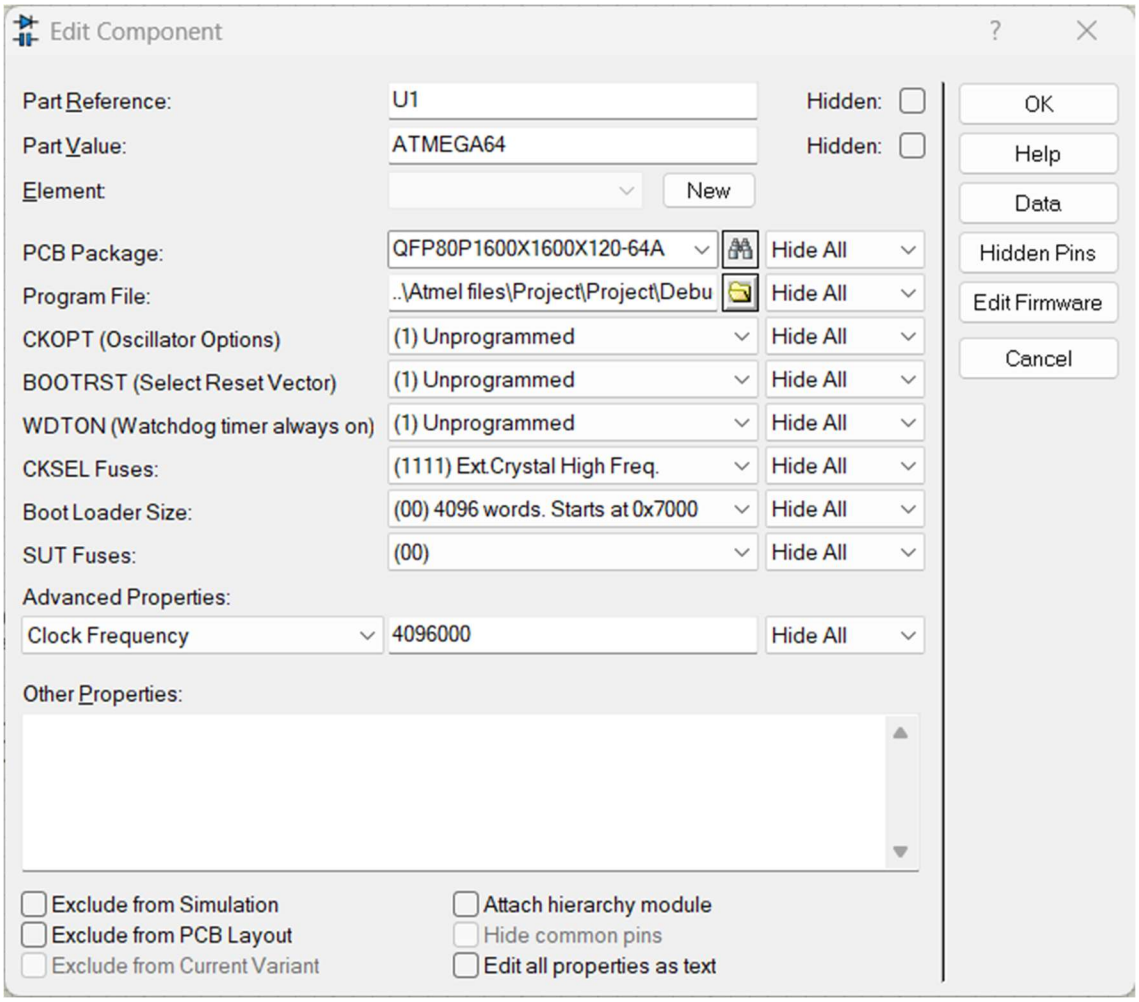


فایل hex. ای ایجاد می‌شود که باید آن را در محیط پروتئوس استفاده کنیم.



بلوک‌های مورد نیاز را در شماتیک می‌آوریم. از مقاومت‌های 330اهمی برای LEDها و از مقاومت 10کیلواهمی برای کلید استفاده می‌کنیم.

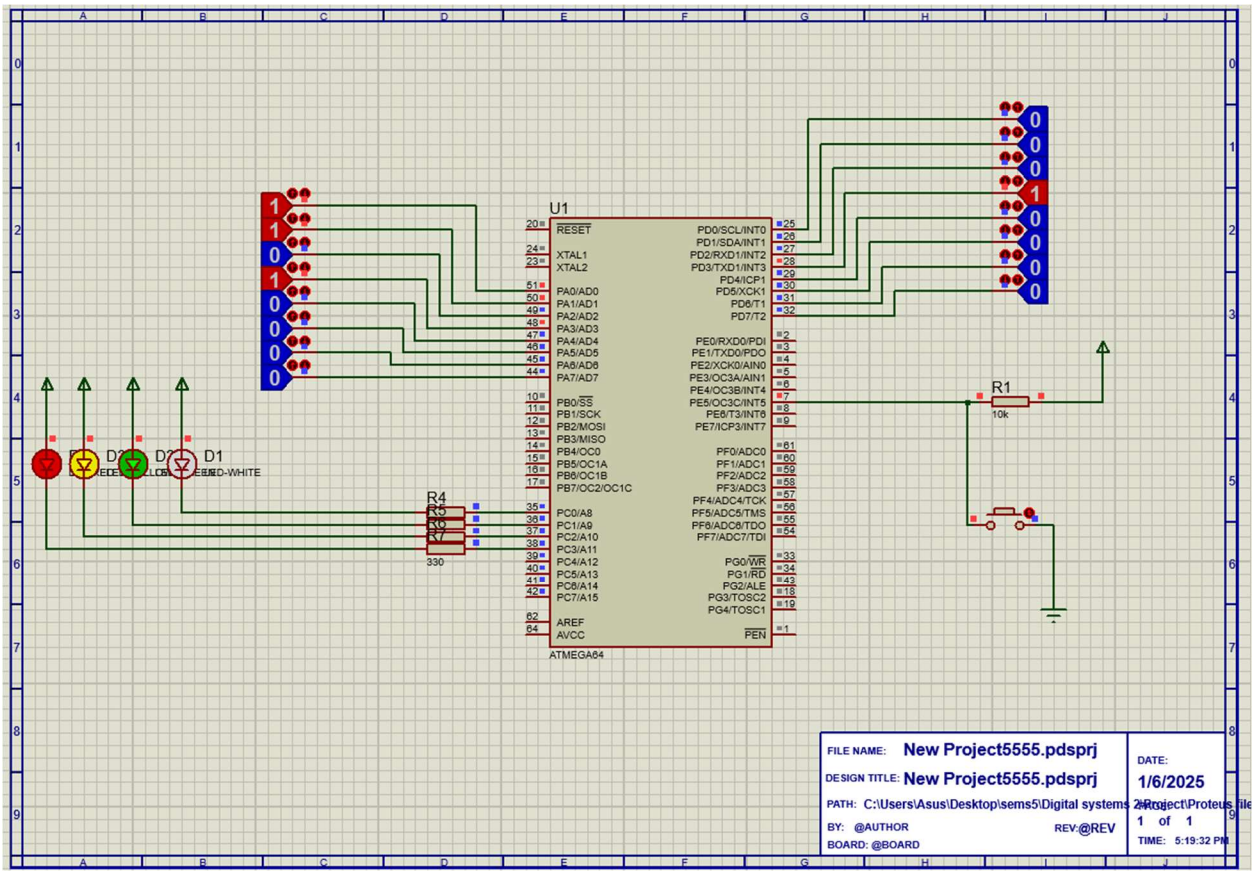
روی بلوک پردازنده کلیک می‌کنیم تا پنجره زیر باز شود.



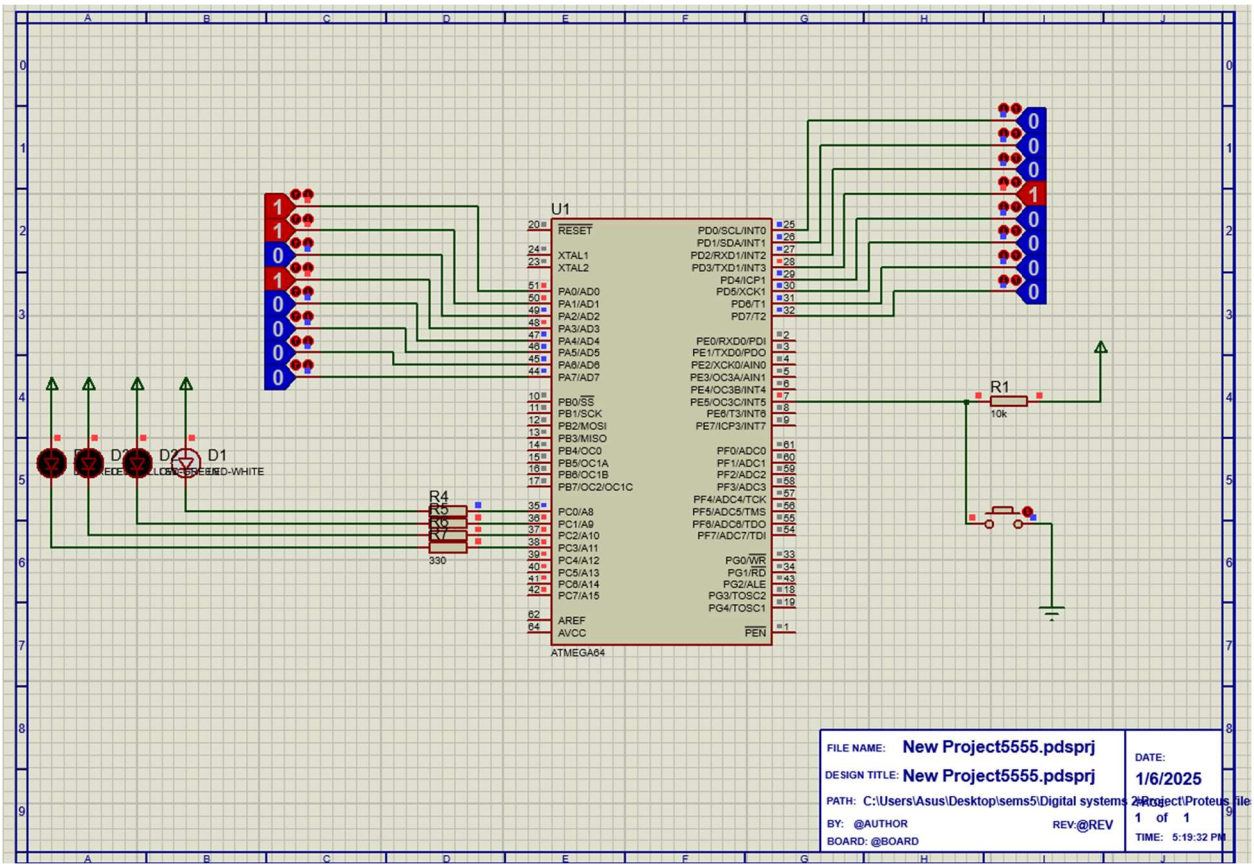
از بخش program files آن فایل hex. را انتخاب می‌کنیم و حتماً فرکانس پردازنده را تغییر می‌دهیم و MHz4.096 قرار می‌دهیم.

نتایج شبیه‌سازی

قبل از آمدن وقفه:



وقفه را می‌زنیم.



عدد بزرگ‌تر است پس چراغ زرد باید روشن شود.

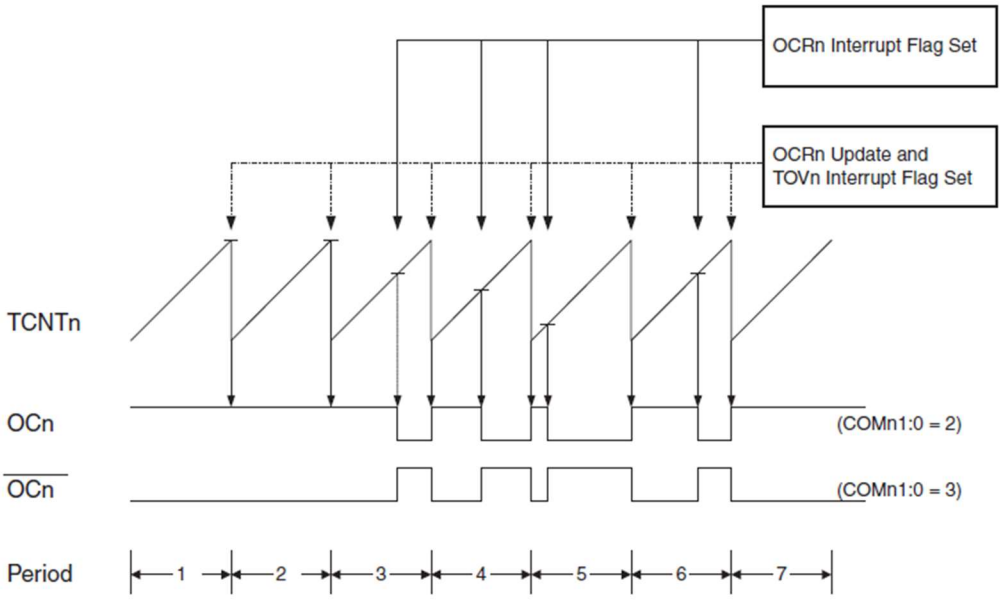
ویدیو کامل عملکرد آن ضمیمه شده است.

بخش امتیازی

مُدی به‌نام Fast pwm برای این کار وجود دارد. fast Pulse Width Modulation یا مدلاسیون عرض پالس سریع، فرکانس بالا برای مولد موج pwm را فراهم می‌سازد.

اساس کار این است که موج پالسی تولید کنیم و duty cycle آن را کاهش دهیم.

Figure 39. Fast PWM Mode, Timing Diagram



برای فعال کردن این مُد باید تغییراتی در TCCR0 به‌وجود بیاوریم.

Bit	7	6	5	4	3	2	1	0	
0x33 (0x53)	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Table 52. Waveform Generation Mode Bit Description[(1)]

Mode	WGM01 (CTC0)	WGM00 (PWM0)	Timer/Counter Mode of Operation	TOP	Update of OCR0 at	TOV0 Flag Set on
0	0	0	Normal	0xFF	Immediate	MAX
1	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	1	0	CTC	OCR0	Immediate	MAX
3	1	1	Fast PWM	0xFF	BOTTOM	MAX

پس باید بیت سوم و ششم این رجیستر را تغییر دهیم.

اگر بخواهیم برای مثال از OCR0 استفاده کنیم، باید LED سفید را به آن وصل کنیم. یعنی به‌جای بیت صفر پورت C، آن را باید به بیت ۴ پورت B وصل کنیم. پس لازم داریم آن را خروجی تعریف کنیم.

```
SBI   DDRB,4
```

مقداری که باید در رجیستر بریزیم تا مد fast pwm فعال شود، چنین چیزی باید باشد.

```
LDI   R20,0x4F
OUT   TCCR0,R20
```

برای آن‌که duty cycle یا شدت روشنایی در ابتدا صددرصد باشد، باید عدد 255 را در OCR0 بریزیم. برای کم‌سو کردن نور، کافی‌ست برای ده بار عدد 25 را کم کنیم و در رجیستر OCR0 بریزیم.

```
SUBI R20, 25
OUT   OCR0,R20
```

منابع

ATMEGA64 Datasheet, Atmel, 2008