Simulasi Port I/O pada Atmega8535 Terhubung dengan Proteus dan Interrupt

Diperuntukkan untuk memenuhi salah satu tugas praktikum Mata Kuliah Aplikasi Mikrokontroler



Praktikum : Aplikasi Mikrokontroler

Praktikum ke : 7

Tanggal Praktikum : Kamis, 12 November 2020

Tanggal Pengumpulan Laporan : Rabu, 18 November 2020

Nama dan NIM : 1. Amir Husein (181344003)

Kelas : 3-TNK

Instruktur : 1. Ferry Satria, BSEE., M.T

2. Rahmawati Hasanah, S.ST., M.T

Politeknik Negeri Bandung

Tahun Ajaran 2020/2021

**I. TUJUAN**

* Mahasiswa dapat memahami prinsip penggunaan port pada Atmega8535
* Mahasiswa mampu melakukan koneksi antara simulator AVR Studio dengan simulator Proteus
* Mahasiswa dapat memahami dan mengaplikasikan penggunaan interrupt pada Atmega8535

**II. LANDASAN TEORI**

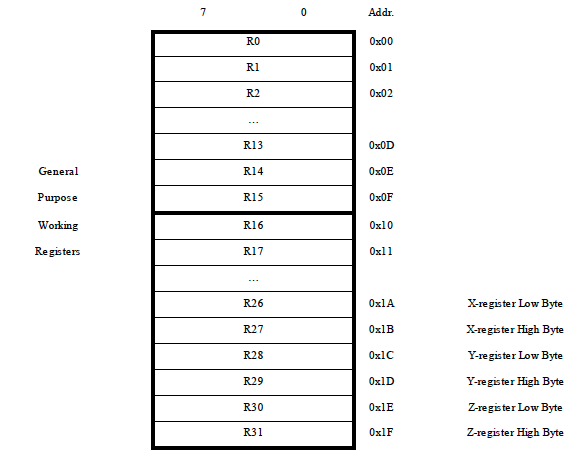
1. Mikrokontroler Atmega8535

Mikrokontroler merupakan suatu chip komputer mini, dimana di dalamnya sudah terdapat sebuah mikprosesor disertai memori, baik itu RAM, ROM, maupun EEPROM. Selain itu, mikrokontroler telah memiliki sistem integrasi Input dan Output (I/O) yang telah dikemas sedemikian rupa pada rangkaian Integrated Circuitnya, sehingga memudahkan dalam melakukan tugas atau operasi tertentu.

Atmega8535 merupakan sebuah mikrokontroler 8-bit yang dikeluarkan oleh perusahaan Atmel pada tahun 2006. Mikrokontroler ini pun memiliki flash memori sebesar 8kb serta EEPROM 512 byte. Selain itu, pada papan mikro ini pun sudah terdapat sebuah ADC dan 3 buah Timer sebagai pencacah waktu pemrosesan data.

1. General Purpose Register

Register ini merupakan register umum yang dapat digunakan sebagai kontrol pengisian dan penyesuaian data. Pada Atmega8535, terdapat 32 general purpose register, yang penamaannya berupa rentang angka dari R0 sampai dengan R31.



Gambar 1. General Purpose Register pada Atmega8535

Terlihat pada gambar diatas, diantara R15 dan R16 terdpat sebuah garis tebal, hal ini menandakan bahwa register R15 kebawah tidak berlaku instruksi *immediate* atau langsung, sedangkan register R16 keatas berlaku. Pada general purpose register ini juga terdapat register khusus sebagai pointer alamat 16-bit yaitu R2 hingga R21 yang terbagi menjadi 3 segmen yaitu pointer X, Y, dan Z.

1. Instruksi LPM dan LD

Instruksi LPM (Load Program Memory) digunakan untuk melakukan *load* dari memori dengan tujuannya adalah register pointer. LPM digunakan untuk mendapatkan nilai yang tepat dari setiap alamat program memori yang dideklarasikan.

Instruksi LD (Load from Data) digunakan untuk melakukan *load* dari memori, apabila LPM dari program area, maka LD dari data area.

1. I/O pada Atmega8535

ATMEGA 8535 memilki empat buah port yang benar-benar memiliki fungsi untuk Read-Modify-Write ketika digunakan sebagai port General digital I/O. Driver pin pada I/O port cukup kuat untuk menggerakkan tampilan LED secara langsung. Semua pin port memiliki resistor pull-up yang dapat dipilih secara individual dengan resistansi invarian tegangan suplai.

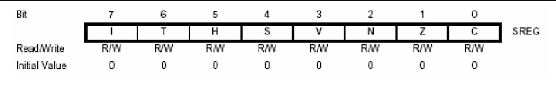
Dalam mengatur sebuah port I/O pada Atmega8535, kita perlu mengatur bit pada DDRX, X disini berdiri sebagai jenis port. Misal jika kita ingin mengatur port A, maka digunakan DDRA, begitupun dengan B hingga port D. Jika DDRX diberikan nilai 1 (high), maka pin digunakan sebagai output. Jika DDRX diberikan nilai 0 (low), maka pin difungsikan sebagai input. Selain DDRX, ada juga register PORTX digunakan untuk mengaktifkan pull-up resistor (pada saat pin difungsikan sebagai input), dan memberikan nilai keluaran pin high/low (pada saat difungsikan sebagai output). Terdapat juga PINX, sebuah register yang berfungsi untuk mengetahui keadaan tiap-tiap pin pada mikrokontroller.

1. Interrupt pada Atmega8535

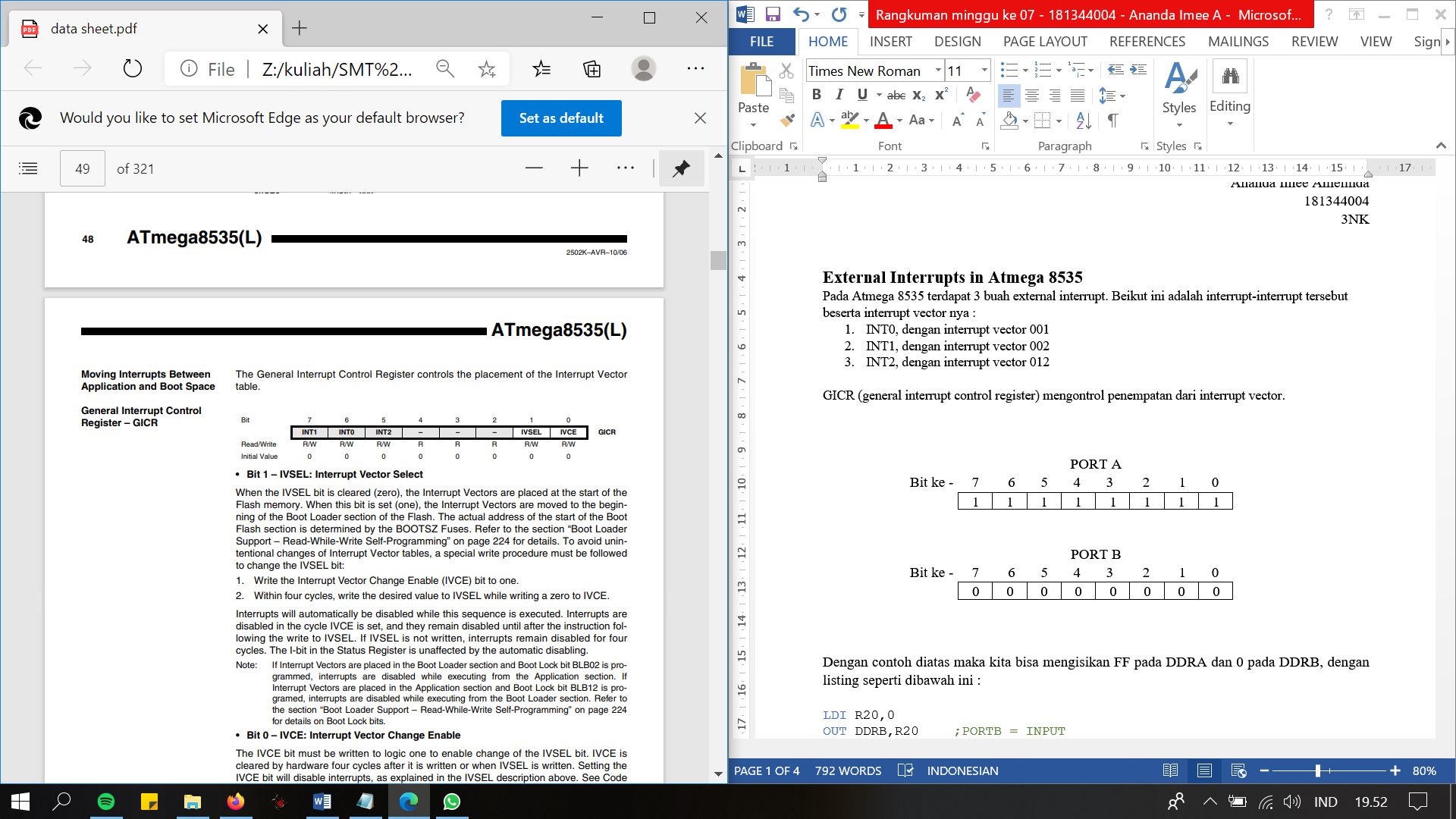
Pada Atmega8535 terdapat 3 buah eksternal interrupt, yaitu sebagai berikut dengan disertai alamat vectornya:

1. INT0 === 001
2. INT1 === 002
3. INT2 === 012

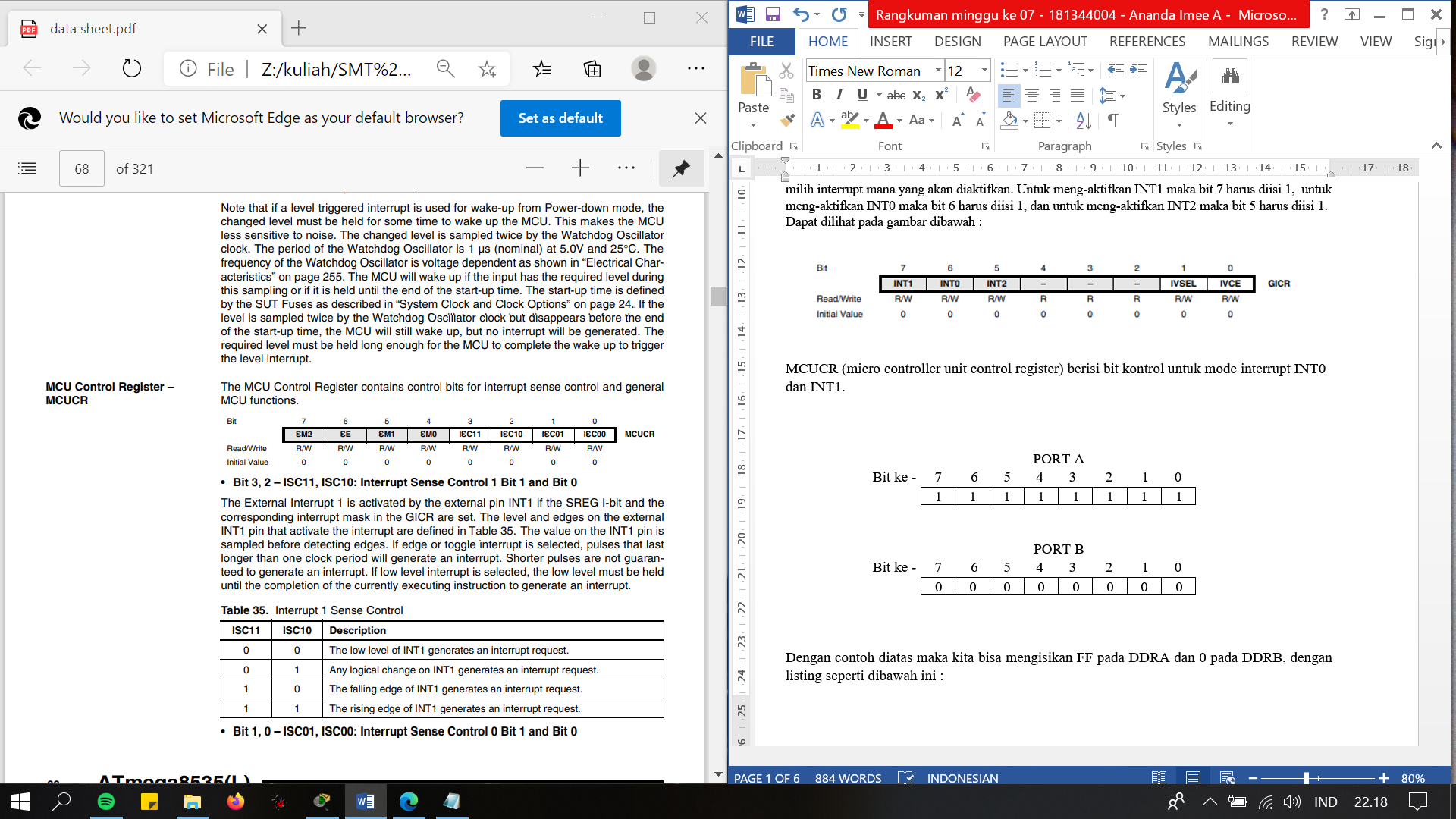
Untuk dapat menggunakan interrupt ini, langkah yang perlu dilakukan pertama kali ialah memastikan bahwa register ‘I’ (Interrupt) pada SREG bernilai 1, agar interrupt enable, sebaliknya bila diisikan 0, maka interrupt akan disable.



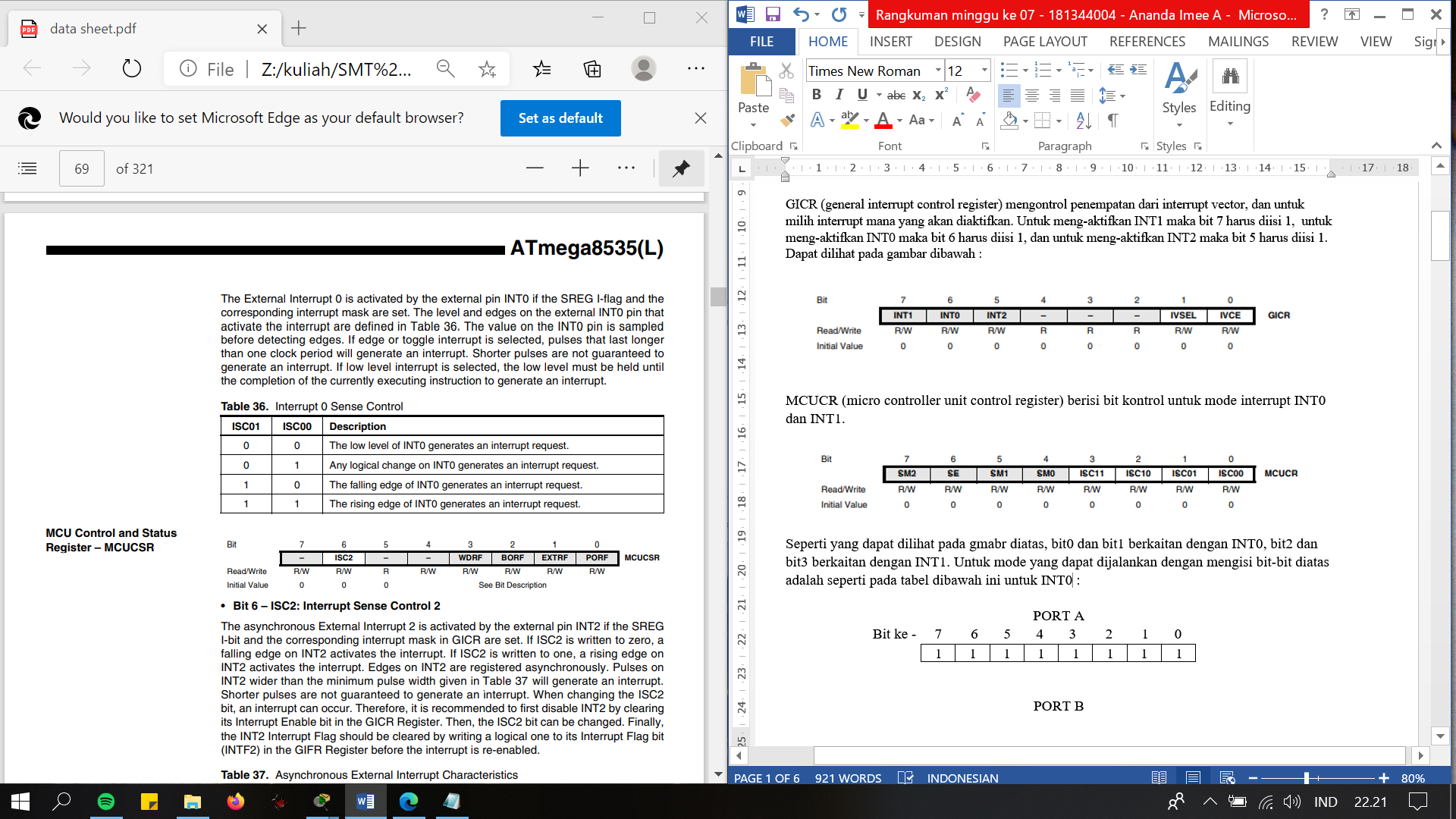
Selain itu, terdapat GICR (General Interrupt Control Register) yang berfungsi untuk mengontrol penempatan dari interrupt vector, dan untuk milih interrupt mana yang akan diaktifkan. Untuk meng-aktifkan INT1 maka bit 7 harus diisi 1, untuk meng-aktifkan INT0 maka bit 6 harus diisi 1, dan untuk meng-aktifkan INT2 maka bit 5 harus diisi 1. Sebaliknya, apabila diisi dengan logika 0, maka interrupt yang bersangkutan akan disable. Untuk struktur GICR dapat dilihat pada gambar dibawah ini:



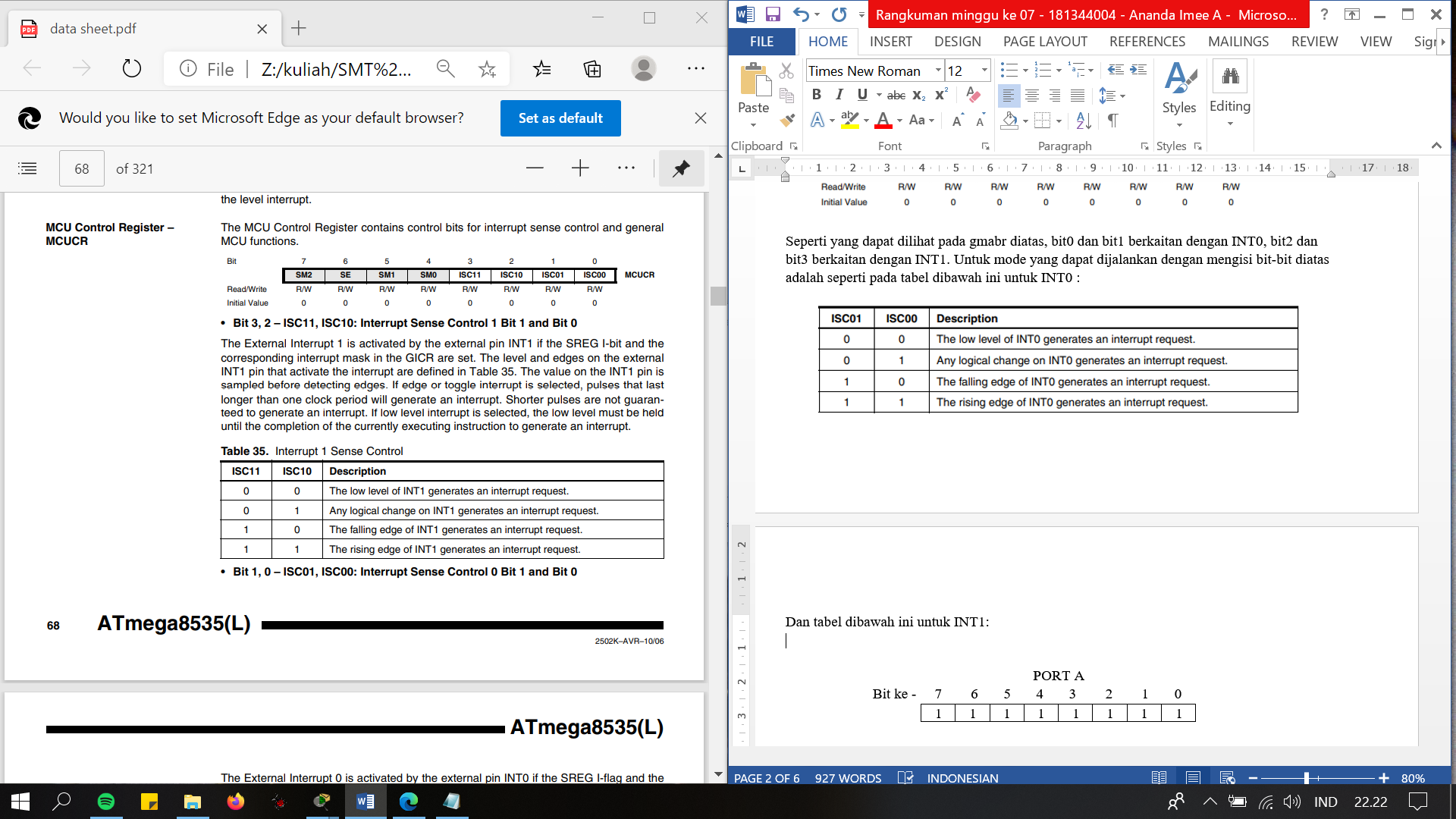
Terdapat juga MCUCR (MicroController Unit Control Register)yang berisi bit kontrol untuk mode interrupt INT0 dan INT1. MCUCR ini berguna untuk menentukan tipikal mode bagi interrupt INT0 dan INT1, seperti bagaimana respon dalma menerima input untuk dapat diproses pada mikrokontroler. Untuk struktur MCUCR dapat dilihat pada gambar dibawah ini:



Bit 0 dan Bit 1 pada MCUCR berkaitan dengan INT0 dan Bit 2 serta 4 berkaitan dengan INT1, mode dari interrupt dapat diatur sedemikian rupa sesuai pada tabel kombinasi ISC. Untuk INT0 ditunjukkan sebagai berikut:

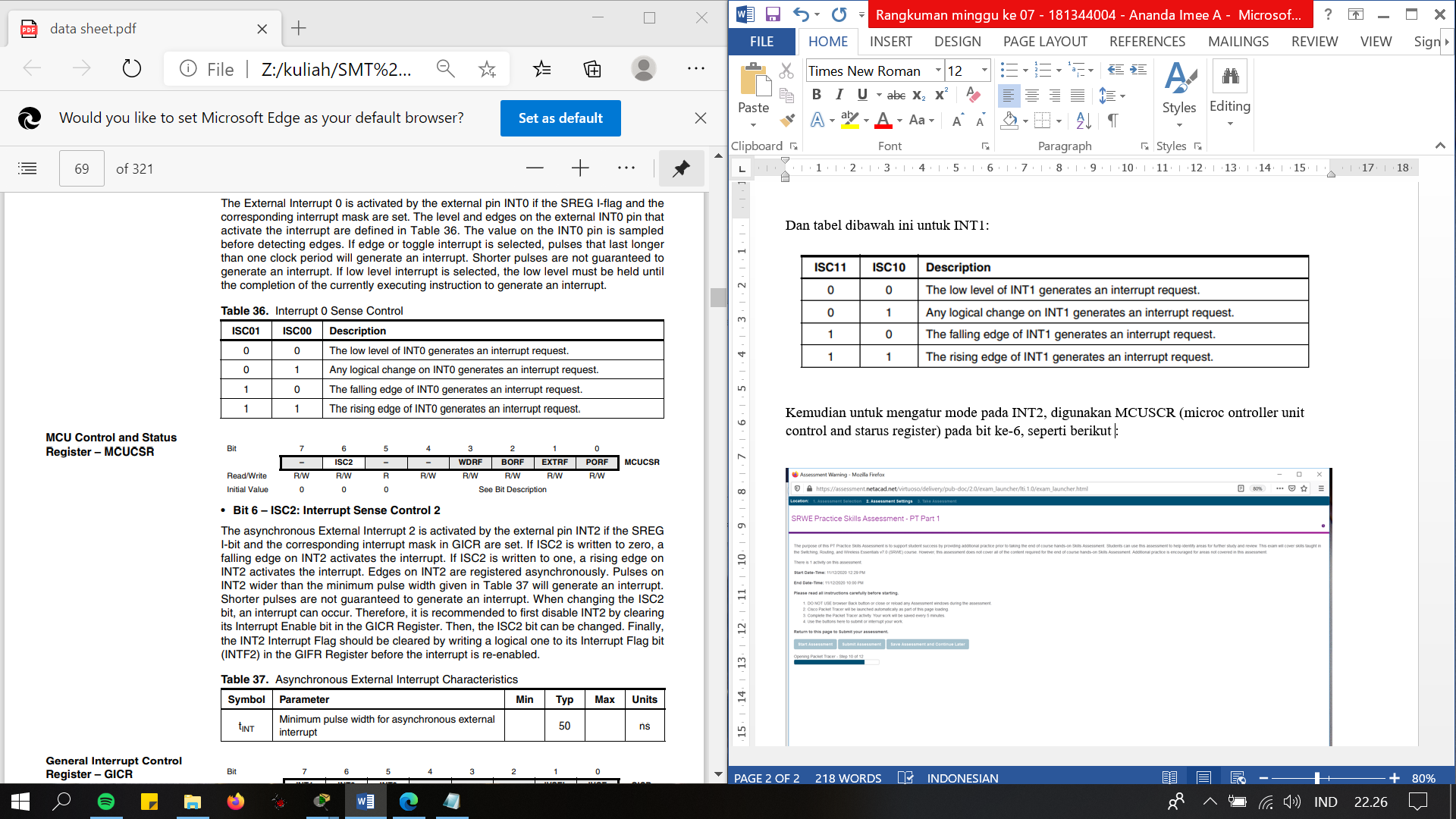


Dan tabel dibawah ini untuk INT1:



Sebagai contoh, untuk INT1, apabila ISC11 diatur pada logika 0 dan ISC10 pada logika 1, maka INT1 akan bekerja ketika menerima setiap perubahan logika, sedangkan bila ISC11 dan ISC10 diatur 1, maka INT1 akan bekerja ketika rising edge atau ada peningkatan nilai logika.

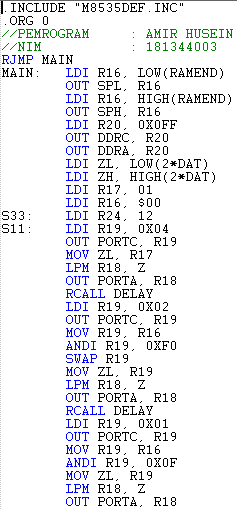
Kemudian untuk mengatur mode pada INT2, digunakan MCUSCR (MicroController Unit Control and Status Register) pada bit ke-6, seperti berikut :

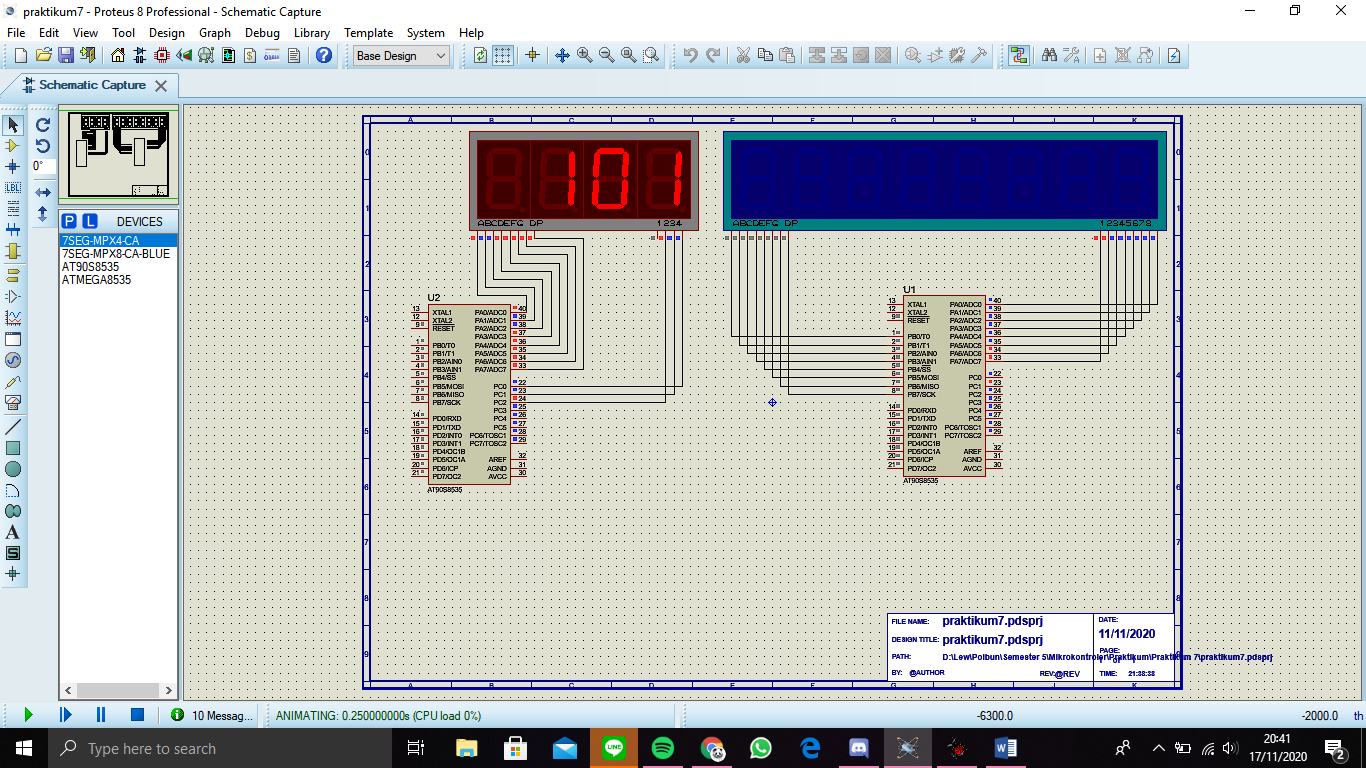


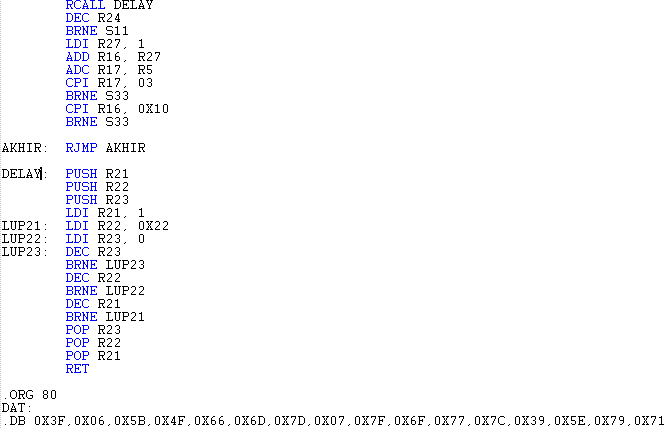
**III. SOAL LAB**

1. Buat program untuk Heksa-Desimal up-counter 3 digit (menghitung dari $100 sampai $30F) yang outputnya berubah 1/5 detik sekali. Tidak berulang.
2. Buat program untuk Heksa-Desimal up-counter 3 digit (menghitung dari $100 sampai $30F) yang outputnya berubah 1/5 detik sekali. Berulang tanpa henti.
3. Buat program untuk Desimal up-counter 3 digit (menghitung dari 100 sampai 359) yang outputnya berubah 1/5 detik sekali. Tidak berulang.
4. Buat program untuk Desimal up-counter 3 digit (menghitung dari 100 sampai 359) yang outputnya berubah 1/5 detik sekali. Berulang tanpa henti.
5. Buat program untuk Octal up-counter 3 digit (menghitung dari 100 sampai 359) yang outputnya berubah 1/5 detik sekali. Tidak berulang.
6. Buat program untuk octal up-counter 3 digit (menghitung dari 100 sampai 359) yang outputnya berubah 1/5 detik sekali. Berulang tanpa henti.
7. Buat program untuk Binary up-counter 5 bit yang outputnya berubah 1 detik sekali, menghitung berulang terusmenerus. Gunakan unit peraga 7 segment sebagai output.
8. Buat program untuk Binary up-counter 8 bit yang outputnya berubah 1/4 detik sekali, menghitung berulang terusmenerus. Gunakan unit peraga 7 segment sebagai output.
9. Buat program untuk menampilkan 2 kata pada unit peraga 7 segmen (8 karakter) secara bergantian berulang, masing-masing selama 4 detik. Kata pertama: “rAJInLAH”. Kata ke dua: “bELAJAr-”.
10. Buat program untuk menampilkan kata: “--rAJIn-”,”-JUJUr--”, dan “—tULUS-” pada unit peraga 7 segmen (8 karakter) secara bergantian berulang, masing-masing selama 4 detik.
11. Buat program untuk menampilkan kata: “--rAJIn-”,”-JUJUr--”, dan “—tULUS-” pada unit peraga 7 segmen (8 karakter) secara bergantian berulang. Pergantian kata dikendalikan melalui tombol interupsi int0.
12. Buat program untuk menampilkan kata: “--rAJIn-”,”-JUJUr--”, dan “—tULUS-” pada unit peraga 7 segmen (8 karakter) secara bergantian berulang, masing-masing selama 4 detik. Kata “-JUJUr--” berkedip 2 kali tiap detik.

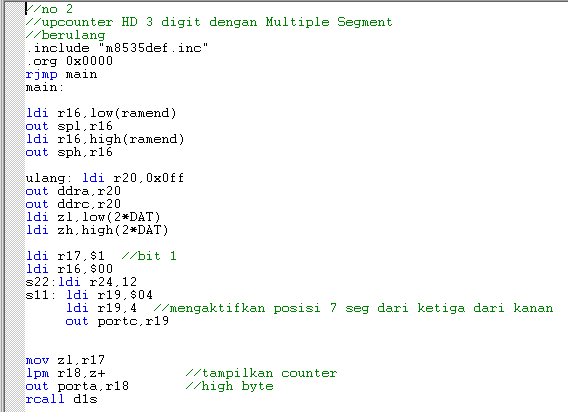
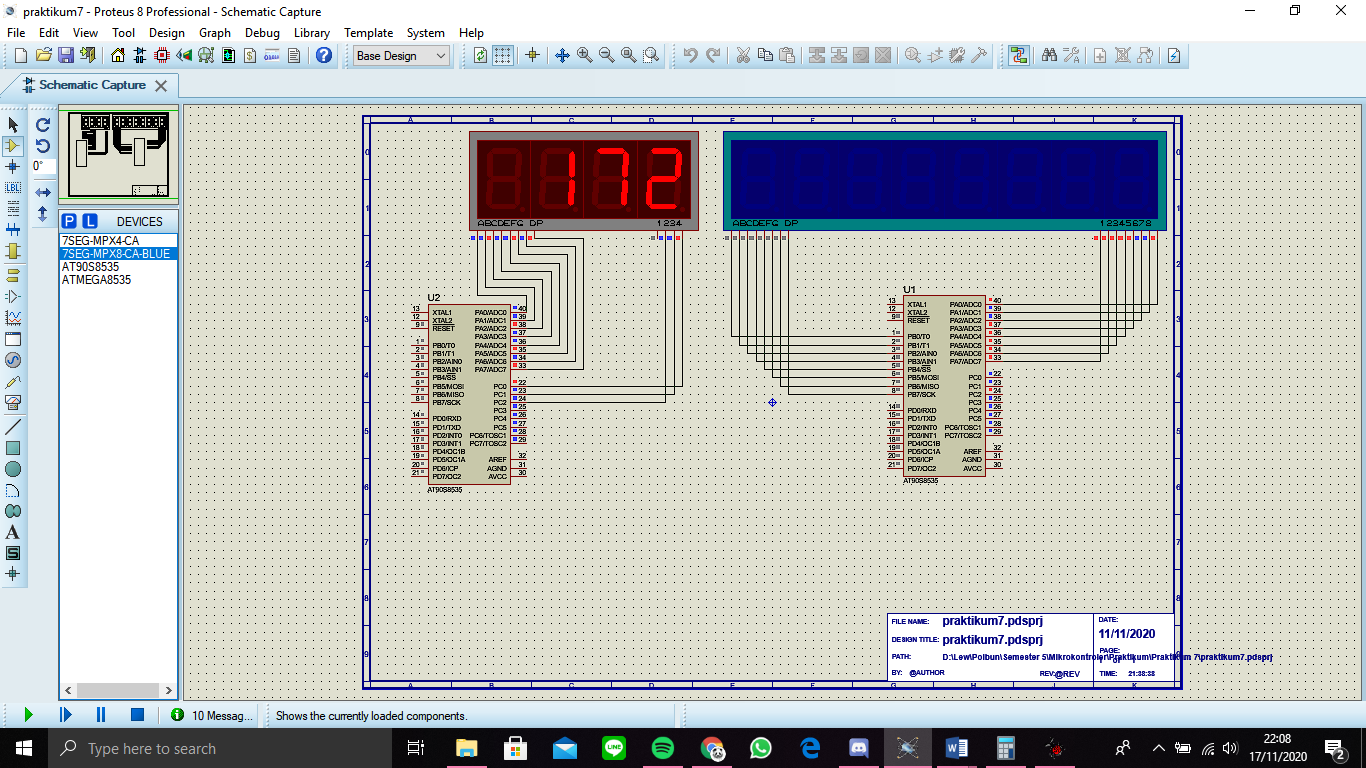
**IV. PROGRAM & HASIL**

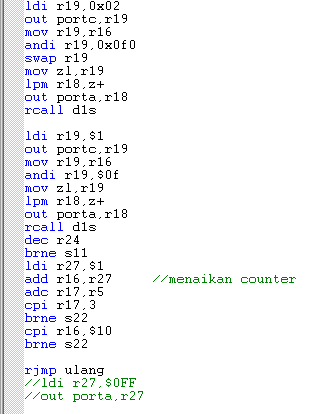
1. Program: Hasil:

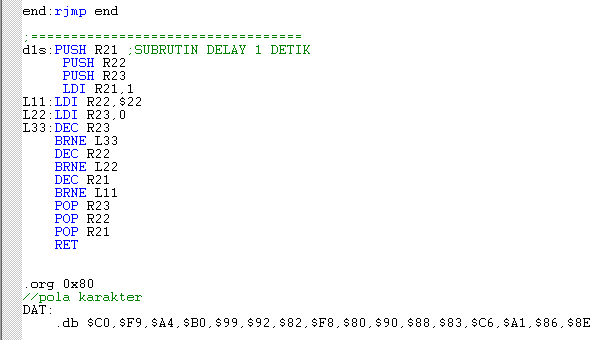




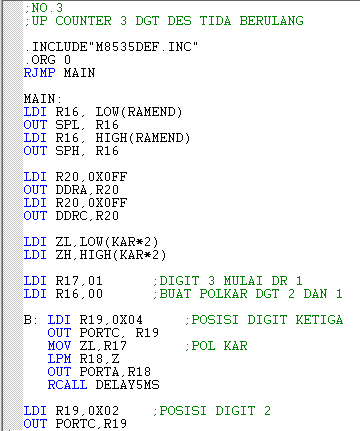
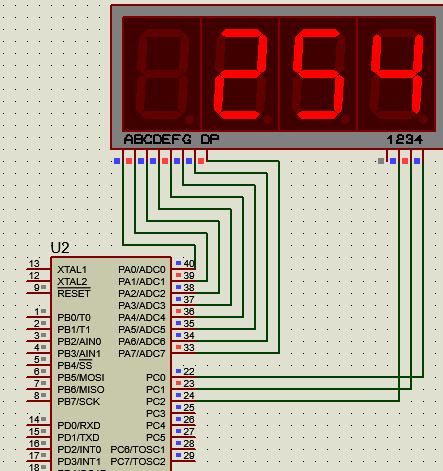
2. Program: Hasil:

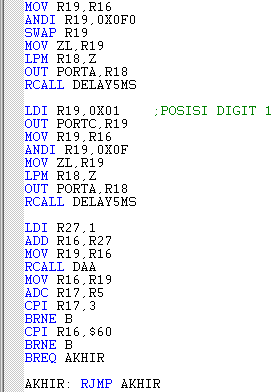
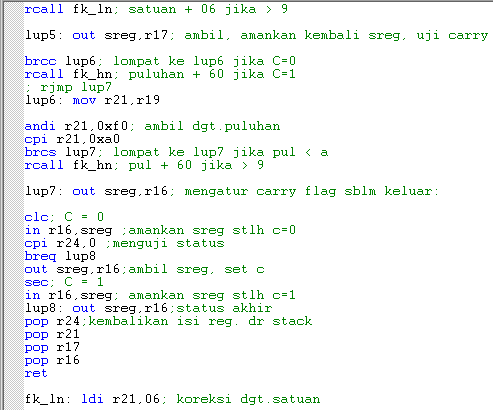
 

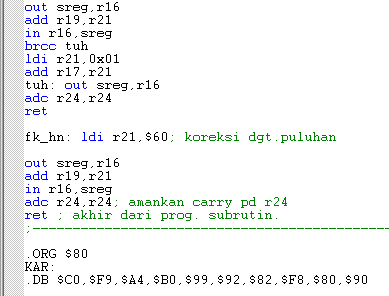


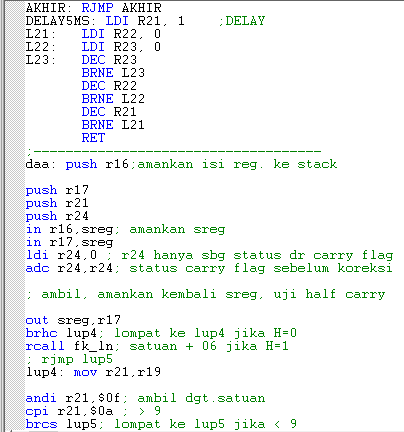


3. Program: Hasil:

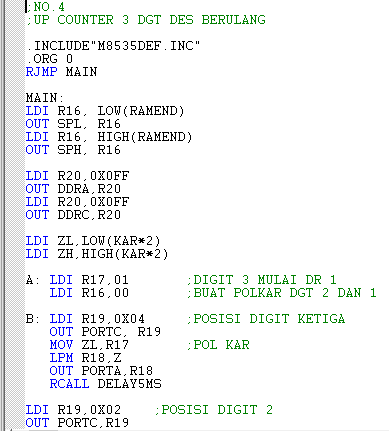
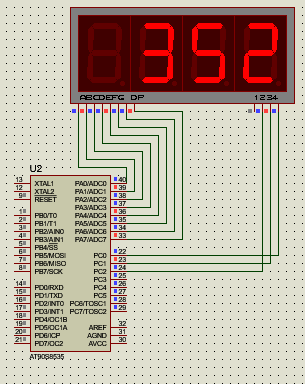
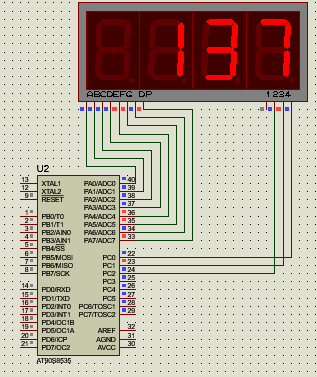


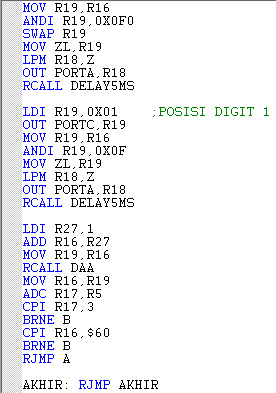


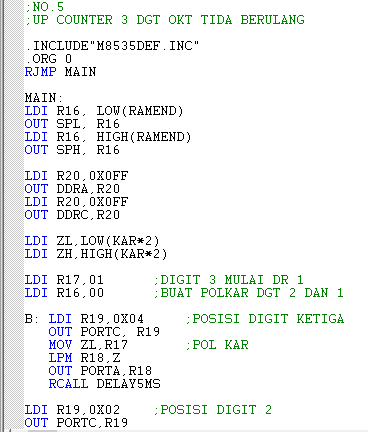


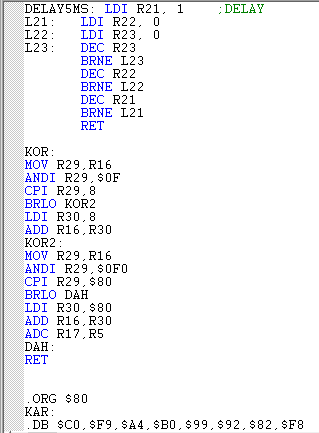
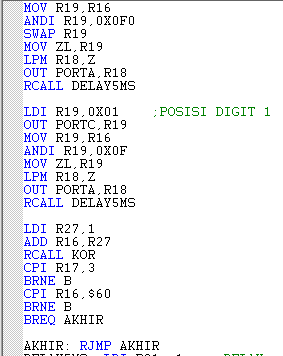


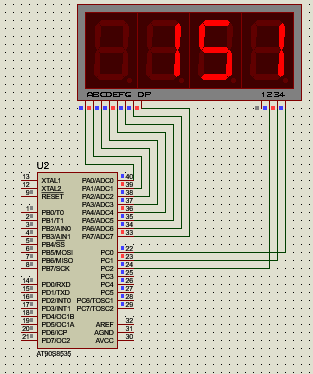
4. Program: Hasil:

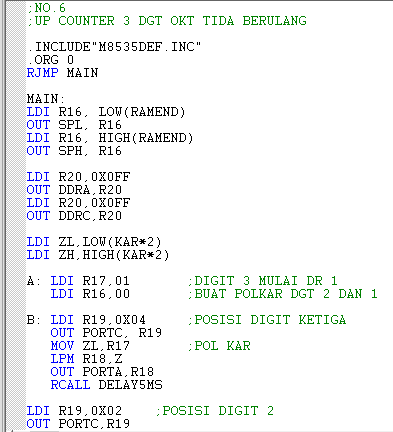
  

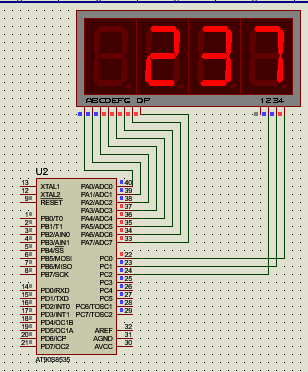


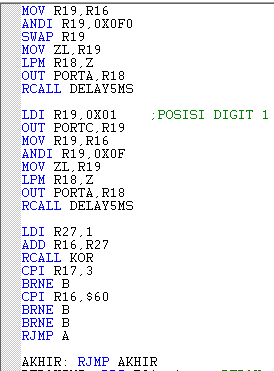
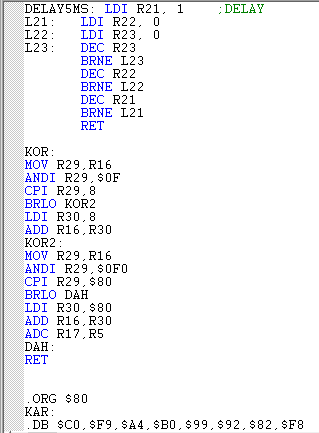
5. Program:



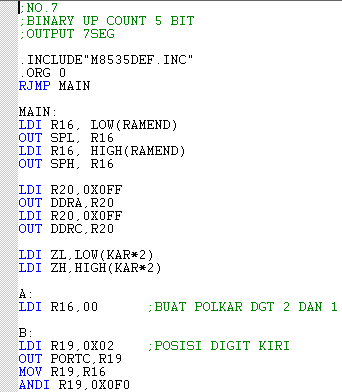
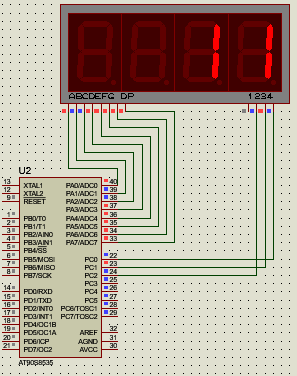
 Hasil:

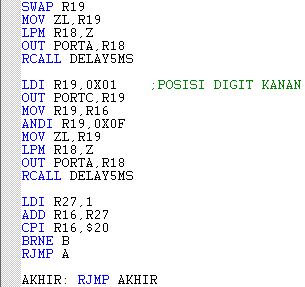
6. Program: Hasil:

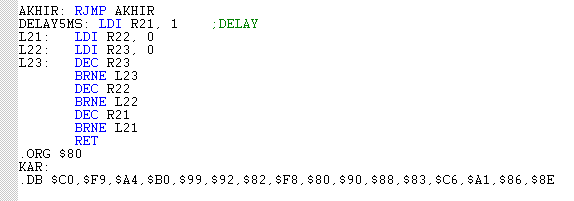


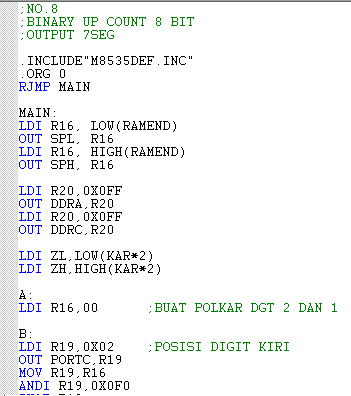


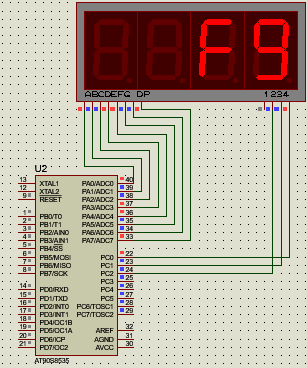
7. Program: Hasil:

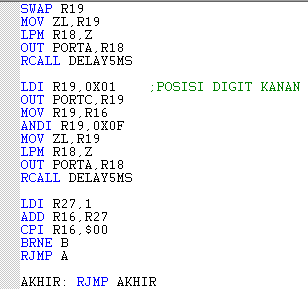


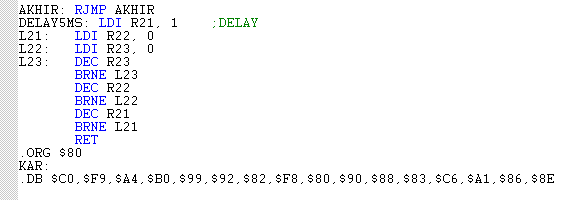




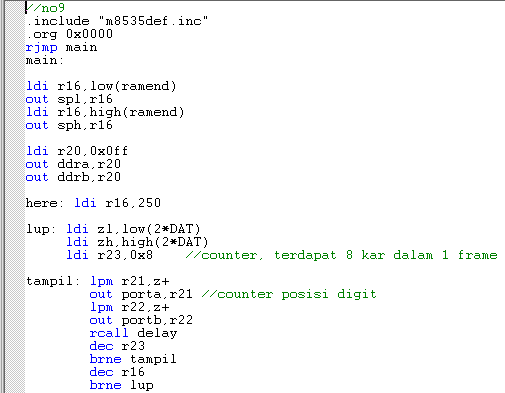
8. Program: Hasil:

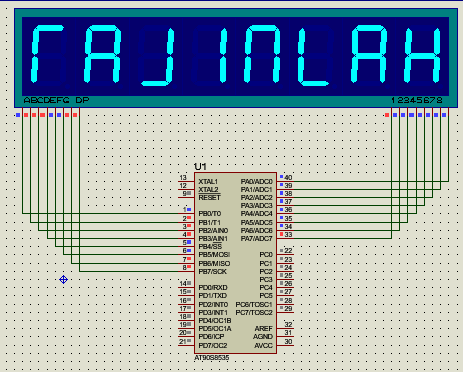


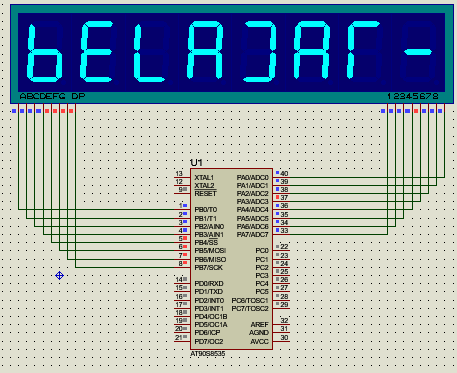


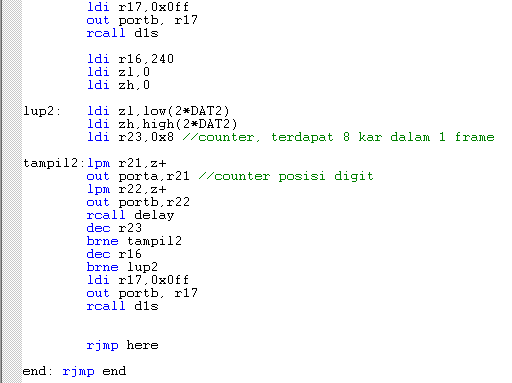


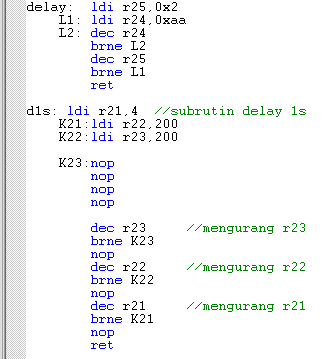
9. Program: Hasil:

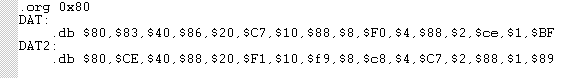




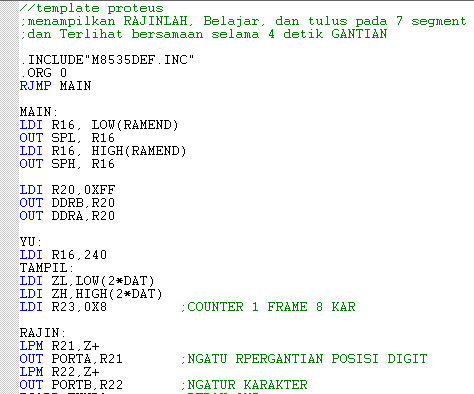
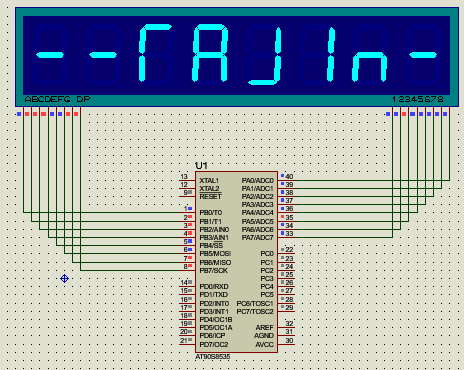


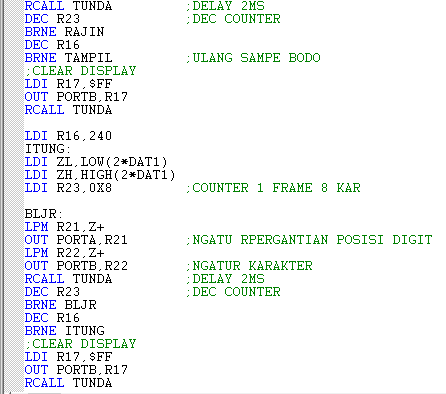


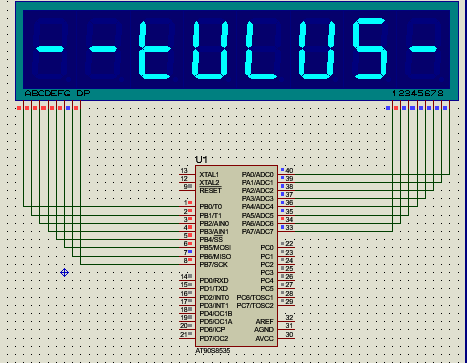


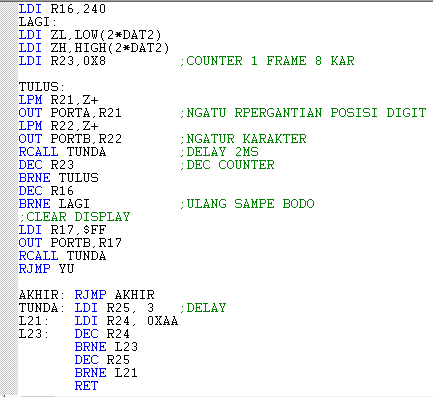


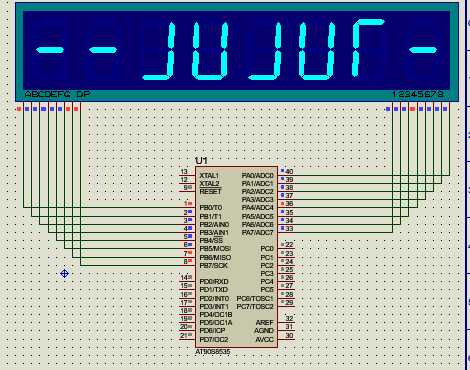
10. Program: Hasil:

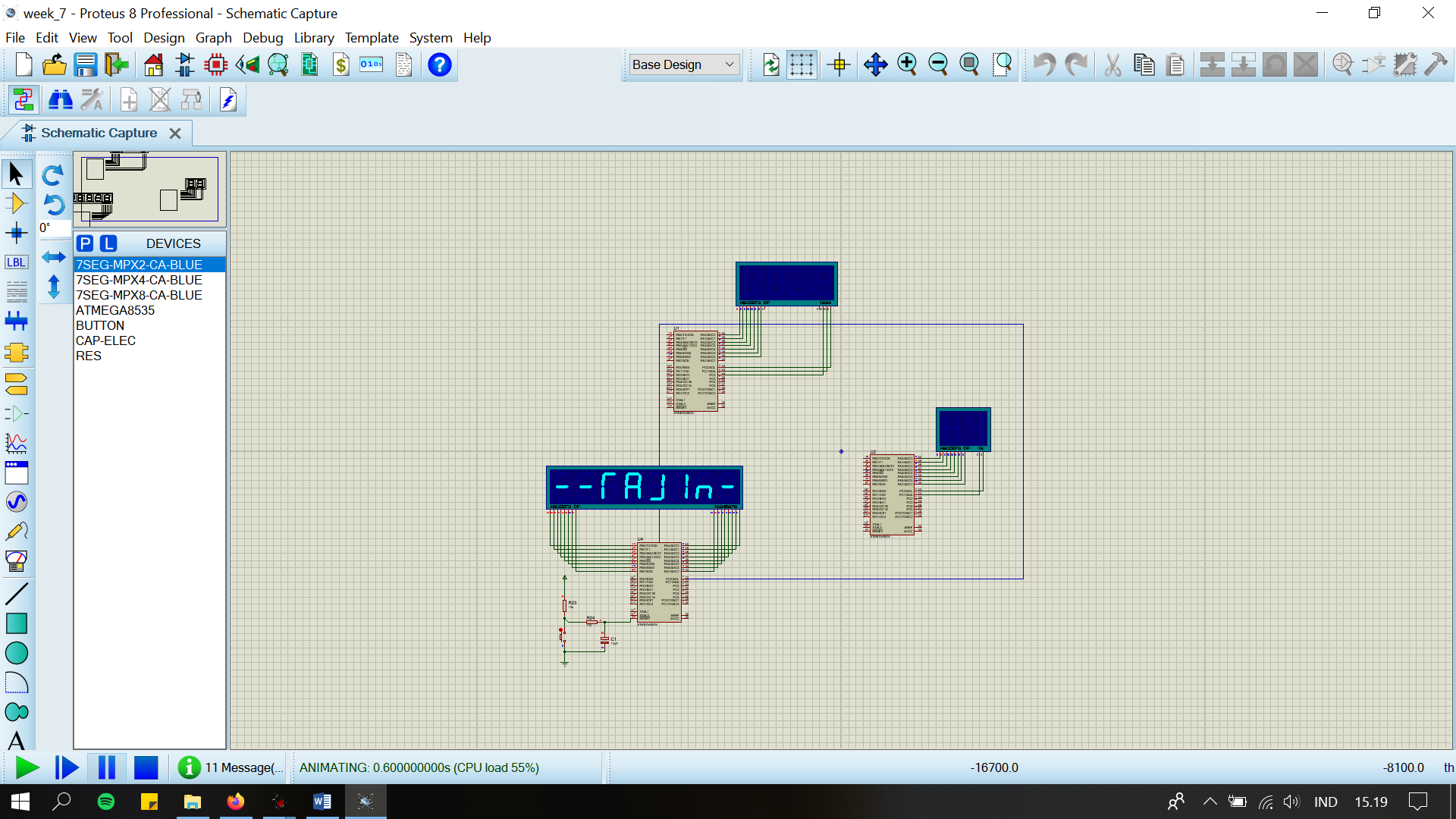


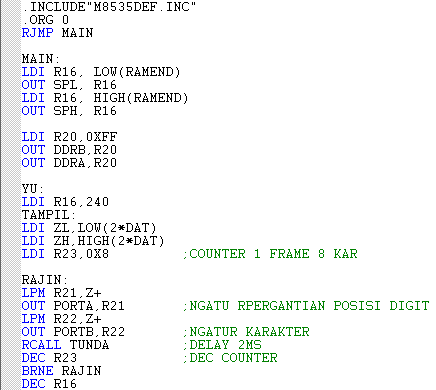


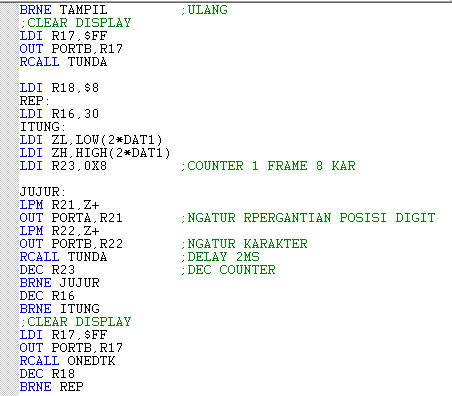
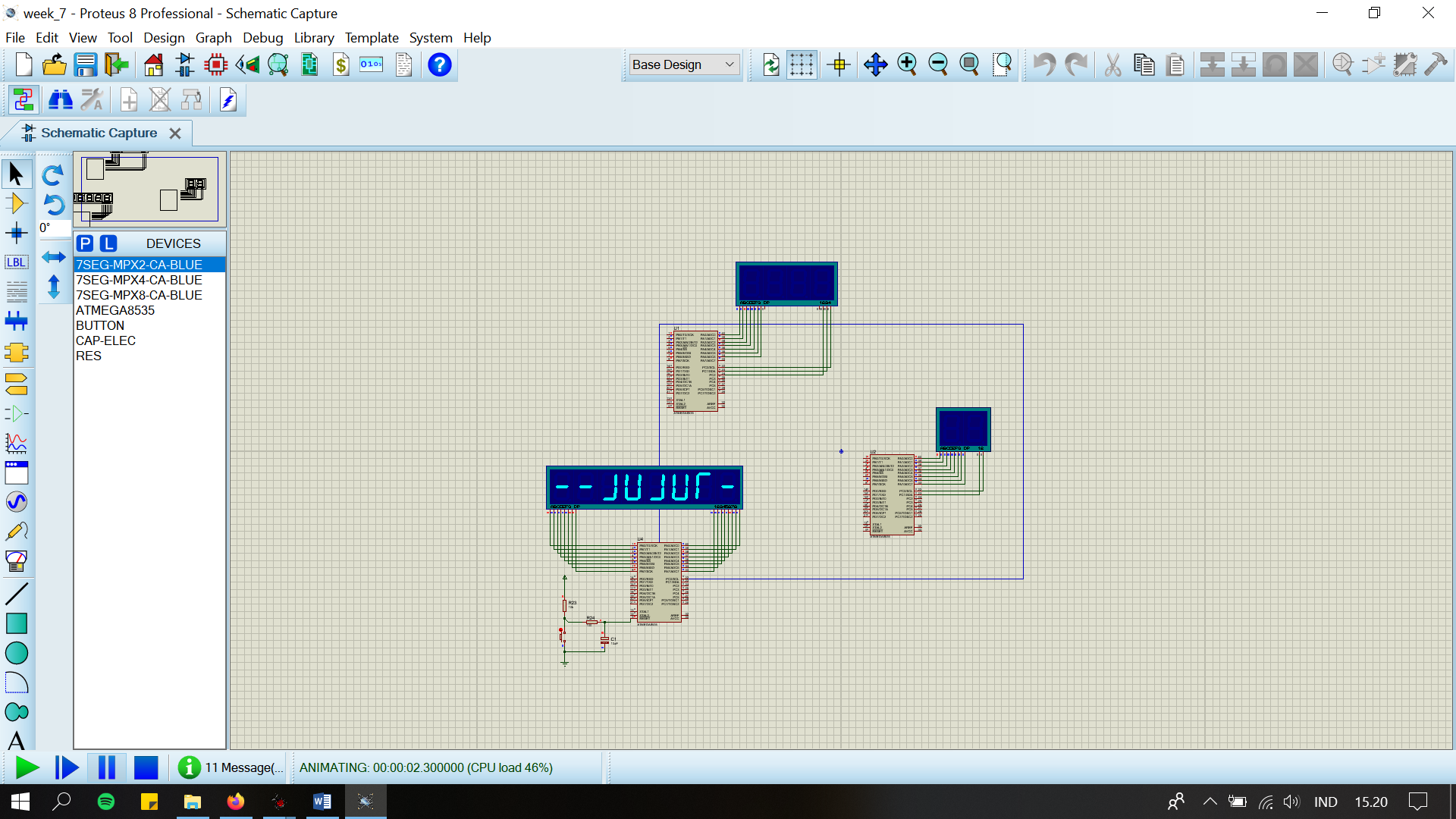


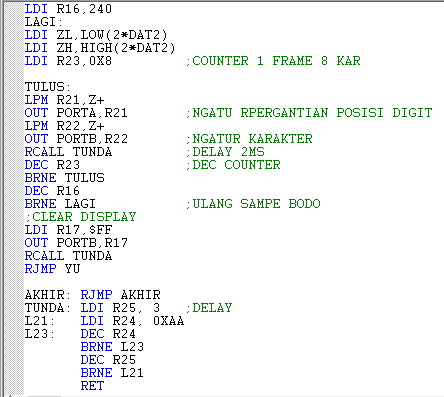
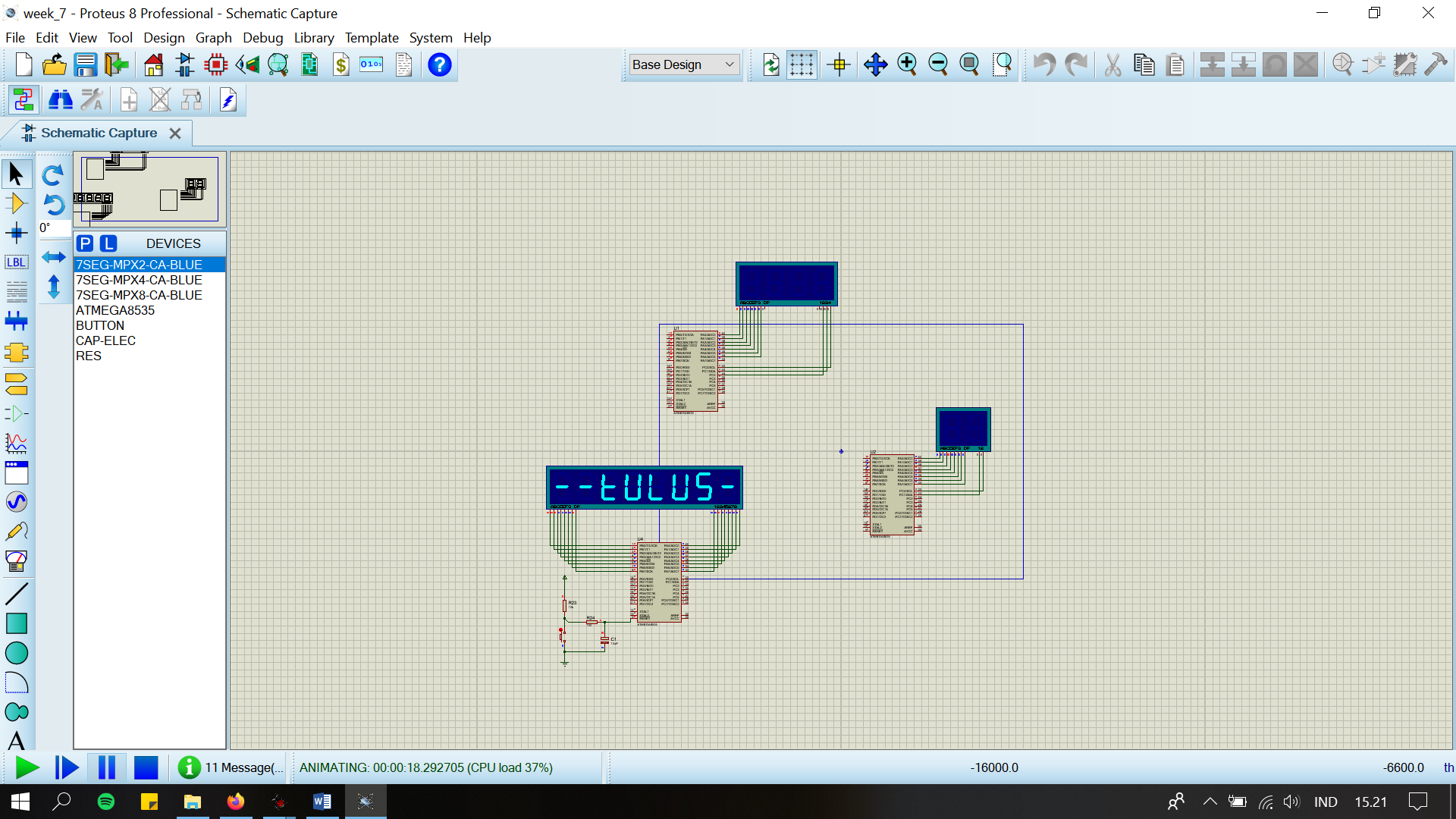


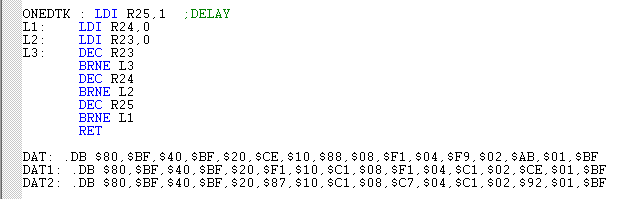


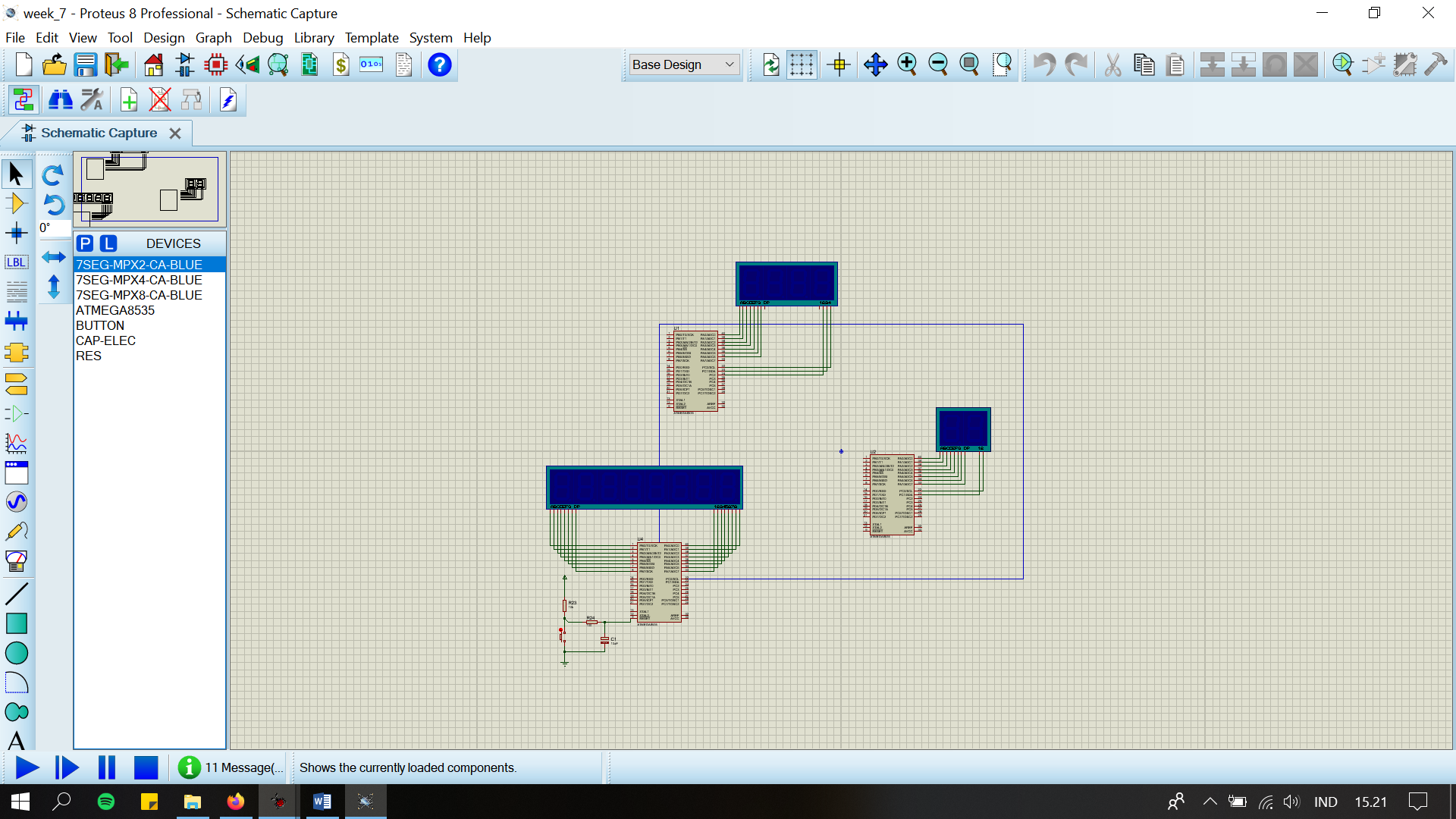
12. Program: Hasil:











**V. ANALISIS**

1. Pada Program ini, akan dibuat sebuah counter 3 digit heksadesimal dari 100 – 30F. Digunakan multiple 7 segment berjenis common anode. Bit paling rendah disimpan paling kanan dengan kode posisi digit yaitu $1 dan bit paling besar disimpan di paling kiri dengan kode posisi digit $4. Pada Rangkaian, hubungkan port C ke pin 4,3,2 dan port A dihubungkan dengan pin A-G,Dp. Kedua port ini bertindak sebagai output maka itu diberi nilai FF agar dapat bertindak sebagai output. Digunakan program area DAT yang berisi kode posisi dan kode karakter heksadesimal (0-F).

Selain itu, Port C akan bertindak sebagai output kode posisi digit dan Port A sebagai output kode karakter. Dari soal, digit paling kiri langsung di outputkan “1”, digit kiri ini akan digunakan counter R17. Sementara itu, untuk digit tengah dan paling kanan digunakan register R16 sebagai counternya. Dimulai dengan memprogram digit paling kiri, maka dimasukan kode posisinya yaitu $4 dioutputkan ke port C. Untuk melakukan *load* data dari program area, digunakan register R18 untuk mengambil kode karakter “1” yang lalu di outputkan ke port A sehingga tercetak 1 pada 7 segment paling kiri.

Selanjutnya, perlu dilakukan konfigurasi untuk memprogram digit tengah dengan kode posisinya $2. Digunakan counter R16 yang berinisiasi 00, lalu di load program memory nya untuk mengambil kode karakter “0” di outputkan ke port C sehingga muncul pada digit tengah. Counter R16 ini digunakan juga untuk digit paling kanan dengan status output yang berbeda. Maka dari itu nilai di R16 nya ini harus diambil perenambelasannya saja. Caranya dengan memindahkannya ke R19 yang lalu di beri logic AND dengan 0xF0 sehingga terambil perenambelasannya, tetapi nilai ini masih berada di posisi perenambelasan, maka dari itu digunakan SWAP sehingga tertukar dengan low nibble nya, sehingga posisi nya menjadi di satuan kembali. Data yang ada di satuan ini akan menjadi alamat pointer z, lalu di load datanya (kode karakter) di R18 lalu dioutputkan ke digit tengah. Dilanjutkan untuk memprogram digit 3 atau paling kanan, Digunakan kode posisi $1 selaku posisi digit paling kanan. Digunakan counter R16 sebelumnya yang lalu di proses pada R19 untuk diambil satuannya dengan AND $0F. Tidak perlu di swap karena sudah pada posisi satuan. Nilai tersebut dijadikan counter yang lalu dilakukan load data ke R18 yang di outputkan ke Port C sehingga tercetak di 7 segment paling kanan.

Dengan menaruh pola karakter pada alamat yang sama dengan karakter kita bisa dengan mudah menggunakan 1 register sebagai counter dan juga alamat sekaligus. Misalnya pertama kita ingin mengeluarkan 00, maka kita bisa isikan 00 ke suatu register kemudian satuan dijadikan alamat untuk mengambil data dan begitu juga dengan puluhan karena data tersebut akan ada pada alamat yang sama dengan datanya. Kemudian jika kita ingin mengeluarkan data 12 maka data tersebut bisa dijadikan alamat yang sudah mengandung pola karakter dari data tersbeut. Begitu lah yang dilakukan dengan R16 sebagai counter dan alamat sekaligus untuk digit satuan dan puluhan dengan cara yang sudah di jelaskan di paragraf sebelumnya. Agar nilai counter R16 dan R17 terus meningkat, nilai R16 ini ditambahkan 1 setiap selesai proses dan carry nya di tambahkan ke R17 selaku counter ratusan. Untuk menghentikan counter, dilakukan counter digit paling kiri dengan $3 atau R17 dengan $3. Apabila belum mencapai $3, maka digit tengah dan digit paling kanan terus melakukan counter dan prosesnya (diloncatkan kembali ke s22). Apabila sudah mencapai $3, maka pembanding selanjutnya adalah $10 untuk digit tengah dan akhirnya, yang dimana data terakhir nya menjadi 30F. Apabila persyaratan ini sudah tercapai, masukan nilai FF ke r27 dan dioutputkan ke port A sehingga semua 7 segment menjadi mati, tandanya program sudah selesai.

2. Program ini tidak jauh berbeda dari program nomor 1, hanya saja perlu ditambahkan pengulangan yaitu dengan cara menambahkan label “ulang” diawal program (mulai dari mengatur port A dan C sebagai output) lalu pada akhir program ditambahkan RJMP ULANG agar program terus berulang. Bagian pengisian data R27 dengan FF untuk mematikan 7 segment dihilangkan agar counter terus berulang tidak berhenti.

3. Pada program ini, sebenarnya tidak jauh berbeda dengan program pada nomor sebelumnya yaitu nomor 1 dan 2. Letak perbedaannya ada pada data yang di gunakan adalah desimal 100-359 sehingga dibutuhkan faktor koreksi pada counter R16 nya yaitu digunakan DAA agar digitnya tidak melebihi 9. Counter digit tengah dan digit kanan pun dibatasi sampai $60 sehingga data akhir nya yaitu 359. Pada program data pun, kode karakter yang dimasukan adalah pola karakter 0-9.

4. Pada program nomor 4 ini tidak jauh berbeda dengan program nomor 3 sebelumnya. Perbedaan adalah program counter ini dilakukan berulang-ulang dengan cara membuat label A pada awal program, lalu saat program telah selesai mencapai 359, program akan loncat lagi otomatis ke awal dengan rjmp A sehingga akan terjadi infinite loop atau berulang terus menerus.

5. Pada program ini, akan dibuat sebuah program up-counter bilangan OKTAL dari 100 hingga 357. Proses dan konsepnya tidak jauh berbeda dengan program pada soal nomor sebelumnya. Perbedaan terletak pada digunakannya subrutin KOR yang akan mengoreksi bilangan sehingga tidak melebihi angka 7. Compaare hasil akhir pun dibatasi sampai $60 sehingga nilai akhirnya adalah 357. Program area pun hanya dimasukan kode karakter 0-7.

6. Soal dan program pada nomor 6 ini sebenarnya mirip persis dengan noor sebelumnya. Cukup ditambahkan sebuah label baru ketika pengecekan data telah mencapai akhirnya, dimana perlu dibuat perbandingan juga untuk dapat melakukan lompatan ke label di awal program.

7 dan 8. Pada dua nomor ini, program yang hendak dibuat meupakan program up counter 5 bit dari data 00 – 1F dan up counter 8 bit dari 00 – FF. Operasi yang digunakan cukup sama dan mengacu pada nomor-nomor sebelumnya dengan beberapa perbedaan. Counter yang digunakan hanya 1 karena tidak adanya digit ketiga yang digunakan. Dilakukan program sama seperti sebelumnya yaitu pertama digit MSB dahulu ($2) lalu digit LSB paling kanan ($1), caranya cukup sama dengan mengatur digit tengah dan digit paling kanan seperti soal sebelumnya. Digunakan juga kembali kode karakter heksadesimal, sehingga kodenya dari 0 – F.

Perbedaan kedua nomor ini adalah batas counternya (R16). Pada nomor 7 karena data maksimalnya 1F, maka dilakukan compare hingga $20. Untuk nomor 8, karena datanya maksumumnya FF maka dicompare dengan $00.

9. Program nomor 9 ini sangat berbeda dengan program sebelumnya karena bukan sebuah program counter. Program ini menggunakan output Multiple 7 Segment 8 Digit Common Anode. Digunakan Port A dan Port B sebagai output, dimana Port A sebagai output kode posisi, dan port B sebagai output kode karakter.

Pada program area diatur posisinya yaitu kode posisi dilanjut ke kode karakter lalu dilanjut kode posisi selanjutnya lalu kode karakter selanjutnya dan seterusnya hingga semua kode posisi sudah terpakai, Pada program, dimasukan terlebih dahulu kode posisi $80 sehingga urutan pengisiannya adalah dari kiri ke kanan. Pada program area, digunakan 2 label DAT dan DAT2 dimana DAT sebagai penyimpan kode posisi dan kode karakter “belajar”, dan DAT2 digunakan sebagai kode posisi dan kode karakter “rajinlah”. Program akan mencetak kata “belajar” dahulu. Diberikan inisiasi counter frame R16 sebanyak 240. Ini dimaksudkan agar kata “belajar” akan terus mencetak selama 4 detik sebelum ke proses selanjutnya. Untuk mengambil karakter “belajar” terlebih dahulu, maka lakukan load data dari program area DAT dahulu ke pointer z. Digunakan counter sebanyak 8 kali karena terdapat 8 posisi dan 8 karakter. Dilakukan pengoutputan kode posisi dahulu lewat load data program memory ke r21 lalu dioutputkan ke port A. Dilanjutkan dengan pengoutputan kode karakter lewat load data program memory ke r22 lalu dioutputkan ke port B. Proses ini akan berlangsung 8 kali sampai semua kode karakter dari “belajar” sudah tercetak. Ketika sudah mencapai angka 8 ini, program akan terus mengulang hingga counter R16 (240) terus berkurang hingga 0. Setelah proses tersebut, program akan mereset seluruh 7 segment dengan memberi output port b dengan FF. Setelah clear, lakukan proses yang sama seperti sebelumnya hanya saja dengan kata “rajinlah”, maka dari itu digunakan label DAT2 untuk memanggil kode karakter “rajinlah”, selebihnya prosesnya sama. Pada semua kode karakter “rajinlah” sudah muncul dan counter frame nya sudah habis, loncatkan kembali ke awal program dengan **RJMP HERE** agar data muncul berulang-ulang.

10. Program no 10 ini tidak jauh berbeda dengan program pada soal nomor 9. Terdapat 3 kata yaitu “—rajin-“, “—tulus-“, dan “—jujur-“ sehingga diperlukan 3 label pada program memory yairu DAT,DAT1, dan DAT2. Proses nya cukup sama yaitu dengan menggunakan counter frame r16 240 kali tiap katanya lalu dipanggila kode posisi dan kode karakternya.Karena terdapat 3 kata, maka proses ini menjadi 3 kali. Tidak lupa juga untuk selalu melakukan clean display dengan memberi nilai FF yang dioutputkan ke port b. Karena katanya berbeda dengan nomor sebelumnya, dibuat terlebih dahulu pola karakternya yang baru.

12. Program nomor 12 ini tidak berbeda jauh dengan program no 10 sebelumnya. Perbedaan adalah pada kata “—jujur-“ harus berkedip sekitar 2 kali tiap detik. Pada saat proses kata “—jujur-“ dimulai, dimasukan counter r18 sebanyak 8 kali untuk berkedip 2 kali tiap detik. Counter R16 diisi sebanyak 30, ini berfungsi sebagai counter saat 7 segmentnya mati karena berkedip, agar tidak terlalu lama maka hanya diberi counter sebanyak 30. Diambil kode posisi dan kode karakter kembali sama seperti nomor 10. Pada saat clear display, diberikan delay agar clear display ini tidak terlalu cepat, dilanjutkan dengan pengurangan counter r18 dan memunculkan kata jujur kembali juga dengan clear display nya. Ini membuat kata jujur berkedip. Untuk kata rajin dan tulus bisa disamakan dengan nomor 10.

**VI. KESIMPULAN**

Dari praktikum ini, dapat diambil kesimpulan bahwa untuk dapat mengatur fungsi dari port baik untuk input maupun output, perlu diperhatikan setiap penggunaan bit yang dibutuhkan untuk mengatur fungsi port itu sendiri baik sebagai input maupun output. Selain itu, untuk dapat menampilkan banyak digit 7-segment, dapat digunakan Multiple 7-segment 8 digit, dimana pada 7-segment ini, terdapat dua bagian pin input, satu untuk common (pengatur digit mana yang aktif) dan juga satu lagi adalah bagian pin digit 7-segment itu sendiri. Maka dari itu, perlu diperhatikan pola karakter yang akan diinputkan pada multiple 7-segment ini dalam heksadesimal, dan jangan sampai tertukar dengan kombinasi untuk common.

**VII. DAFTAR PUSTAKA**

Yusrizal. 2016. *Mikrokontroler Atmega8535.* Yusrizal Weblog. \_\_\_\_\_([https://yusrizalandeslubs.wordpress.com/dasar-elektronika](https://yusrizalandeslubs.wordpress.com/dasar-elektronika/)). Diakses 5 Oktober 2020