COUNTER TIMER CIRCUIT (CTC) Z80

Yoyo somantri
Dosen Jurusan Pendidikan Teknik Elektro
FPTK Universitas Pendidikan Indonesia

Counter Timer Circuit (CTC) Z80 adalah komponen LSI yang dapat diprogram dan khusus dipakai untuk sistem mikroprosesor yang menggunakan CPU Z80. CTC Z80 mempunyai 4 buah saluran (*saluran*) yang tidak saling bergantungan (berdiri sendiri) yang dapat berfungsi untuk menghitung (*counting*) dan menghitung waktu (*timing*). CPU dapat mengkonfigurasikan saluran-saluran di dalam CTC agar beroperasi dalam berbagai mode dan kondisi yang diinginkan sesuai keperluan hubungan dengan berbagai macam peralatan luar.

CTC Z80 dikemas dalam bentuk sebuah DIP 28 pin dan menggunakan teknologi N-*saluran silicon gate depletion*. CTC Z80 hanya memerlukan sebuah catu daya sebesar +5V dan sebuah clock 1 fasa sebesar +5V.

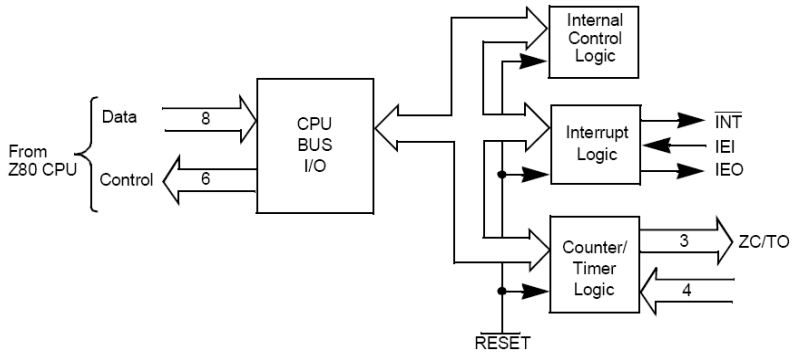
Kemampuan utama dari CTC Z80 adalah sebagai berikut :

a. Semua saluran input dan output sesuai dengan standar TTL.
b. Setiap saluran dapat dipilih untuk beroperasi sebagai mode *counter* atau mode *timer*.
c. Pada mode *counter* maupun mode *timer*, CPU dapat membaca isi dari *down counter* (penghitung mundur) yang sedang dalam proses menghitung hingga mencapai nol.
d. Pada mode *counter* maupun mode *timer*, sebuah *register* konstanta waktu (*time constant register*) secara otomatis akan mengisi kembali *down counter* setelah ia mencapai nol.
e. Dalam mode *timer*, dapat dipilih *trigger input positif* atau *negatif* untuk memulai operasi penghitungan waktu. Input yang sama digunakan juga untuk memonitor penghitungan kejadian (*event count*) di dalam mode *counter*.
f. Ada 3 buah saluran output *Zero Count/ Time Out* yang dapat digunakan untuk men'*drive*' transistor darlington.

g. Pada setiap saluran dapat deprogram agar pada saat kondisi '*Zero Count*' (penghitungan mencapai nol) akan terjadi interupsi.
h. Di dalam CTC sudah terdapat rangkaian *logic* yang berfungsi untuk melayani prioritas *interrupt* (*daisy chain priority interrupt*) yang berfungsi untuk memberikan vector *interrupt* secara otomatis, sehingga tidak diperlukan rangkaian logic tambahan di luar.

1. Arsitektur CTC Z80

Struktur bagian dalam CTC Z80 terdiri dari *interface bus* untuk CPU Z80. rangkaian *control logic* dan 4 buah perangkat saluran *counter/ timer*. CTC mempunyai kemampuan untuk menghasilkan vector *interrupt* bagi masing-masing saluran secara terpisah. Keempat saluran tersebut dapat dihubungkan sehingga membentuk prioritas *interrupt* standar Z80 yang berurutan dimana saluran nomor nol mempunyai prioritas tinggi. Karena adanya rangkaian *interface logic* untuk *bus* CPU memungkinkan peralatan CTC ini dihubungkan secara langsung tanpa memerlukan rangkaian *logic* di luar. Diagram CTC Z80 dapat dilihat pada gambar 1.

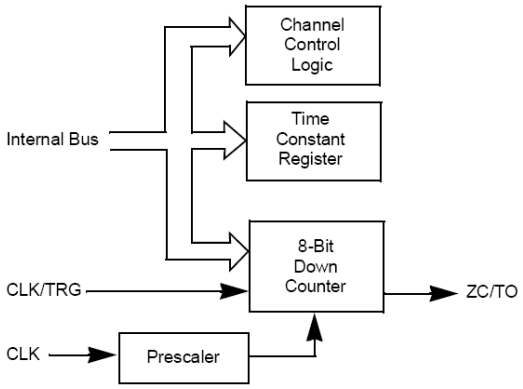


Gambar 1. Diagram blok CTC Z80

(Sumber dari data book Zilog 1986)

Struktur dari salah satu saluran dari keempat perangkat saluran *counter/ timer* dapat dilihat pada gambar 2. Sebuah perangkat saluran *counter/ timer* terdiri dari 2

buah *register*, 2 buah counter dan rangkaian kontrol *logic*. Kedua *register* tersebut berfungsi sebagai 8 bit *saluran control register*, sedangkan kedua *counter* tersebut berfungsi sebagai 8 bit *down counter* yang dapat dibaca oleh CPU dan 8 bit *prescaler* (pembagi awal).



Gambar 2. Diagram blok dari sebuah saluran
(Sumber dari data book Zilog 1986)

Register kontrol saluran dan logik adalah suatu register 8 bit yang dapat diisi oleh CPU untuk menentukan mode dan parameter dari tiap-tiap saluran. Sebuah komponen CTC Z80 mempunyai empat buah *register* jenis ini untuk keempat buah saluran *counter/ timer*-nya. Pengisian ke empat *register* ini secara terpisah dapat dipilih melalui pin input CS0 dan CS1 (umumnya dihubungkan dengan saluran alamat A0 dan A1 dari CPU). Cara memilih ke empat saluran tersebut dapat dilihat pada tabel 1.

Pembagi awal (*prescaler*) hanya digunakan pada mode *timer*. Pembagi awal ini mempunyai 8 bit pembagi yang dapat diprogram oleh CPU melalui saluran kontrol *register* untuk membagi frekuensi sistem *clock* pada inputnya dengan nilai 16 atau 256. Output dari pembagi awal ini dimasukan ke input dari down *counter* dan bila penghitungan mencapai nol, pembagi awal ini secara otomatis diisi kembali dengan nilai yang ada di *time constant register*. Akibatnya akan terjadi pengulangan pebagian sistem *clock* dengan suatu faktor yang ada pada *time constant register*. Setiap kali *down counter* mencapai nol maka output dari *zero count/ time out* (ZC/ TO) akan menghasilkan pulsa.

Tabel 1. Pemilihan saluran

	CS1	CS0
Saluran 0	0	0
Saluran 1	0	1
Saluran 2	1	0
Saluran 3	1	1

Time constant register adalah register 8 bit yang digunakan pada mode *counter* dan mode *timer* yang diprogram oleh CPU setelah pemrograman saluran kontrol. Di dalam *register* dapat diisi dengan suatu nilai (bilangan bulat) sebagai konstanta waktu yaitu dari 1 sampai 256. *register* ini akan mengisi nilai yang telah diprogram padanya ke *down counter* jika CTC pertama kali diinisialisasi dan setelah *down counter* mencapai nol. Jika suatu nilai konstanta waktu diisikan ke *time constant register* sedangkan pada saat itu satu saluran sedang dalam proses menghitung (*counting/timing*), maka proses penghitungan ini akan diselesaikan terlebih dahulu sebelum konstanta waktu yang baru tersebut diisikan ke *down counter*.

Down counter adalah sebuah *register* 8 bit yang digunakan pada mode *counter* dan mode *timer*. Proses pengisiannya terjadi pada saat inisialisasi dan pada saat proses penghitungan mundur register ini mencapai 0. nilai yang diisikan pada saat inisialisasi dan pada saat proses penghitungan mundur ini mencapai nol, nilai yang diisikan pada register ini berasal dari *time constant register*. Nilai dari *down counter* akan berkurang 1 setiap satu perioda *clock* yang dihasilkan oleh pembagi awal (di dalam mode *timer*). Setiap saat CPU dapat membaca isi *register* ini pada masing-masing saluran CTC dapat diprogram untuk menghasilkan sinyal permintaan interupsi setiap penghitungan mundur sampai mencapai nol.

Pada saluran 0, 1, dan 2 jika penghitungan nol tercapai, pada kaki ZC/ TO dari masing-masing saluran akan dikeluarkan sebuah pulsa. Karena keterbatasan jumlah pin dalam kemasannya, maka saluran 3 tidak mempunyai pin output sehingga saluran 3 hanya dapat dipakai pada penggunaan-penggunaan yang tidak memerlukan pulsa tersebut.

2. Bagian Logik untuk Kontrol Interupsi (*Interrupt Control Logic*)

Rangkaian ini berfungsi untuk mengatur kerja dari CTC sesuai dengan aturan interrupsi pada sistem Z80, misalnya prioritas percabangannya dan proses kembalinya program setelah interupsi. Dua buah sinyal (IEI dan IEO) yang ada pada perlatan CTC berfungsi untuk membentuk sistem *daisy chain*. Saluran 0 mempunyai prioritas paling tinggi dan saluran 3 mempunyai prioritas paling rendah. Fungsi dari interupsi yang dihasilkan oleh CTC untuk memaksa CPU melaksanakan *routine service* interupsi sesuai dengan aturan interupsi sistem Z80. peralatan atau saluran yang mempunyai prioritas lebih rendah rendah tidak dapat menginterupsi peralatan atau saluran yang mempunyai prioritas lebih tinggi yang sudah mendahului menginterupsi CPU dan belum selesai melaksanakan *routine service interrupt*-nya. Sedangkan peralatan atau saluran yang mempunyai prioritas lebih tinggi dapat menginterupsi CPU yang sedang melaksanakan *routine service interrupt* dari peralatan atau saluran dengan prioritas yang lebih rendah.

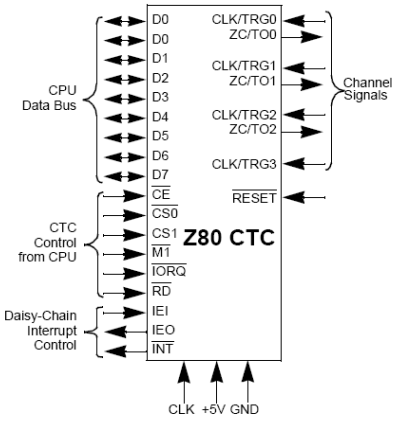
Sebuah saluran CTC harus diprogram dengan menggunakan interrupt mode 2 agar setiap kali menghasilkan sinyal interupsi jika *down counter* mencapai penghitungan nol. Kadang-kadang setelah adanya permintaan interupsi, CPU akan mengirimkan sinyal '*interrupt acknowledge*' dan ICL dari CTC akan menentukan saluran yang meminta interupsi yang mempunyai prioritas paling tinggi di dalam peralatan CTC tersebut. Jika pin input IEI dari CTC sedang aktif yang menandakan CTC mempunyai prioritas (dapat segera menginterupsi) di dalam keseluruhan sistem *daisy chain*, dan CTC dapat mengeluarkan 8 bit interupsi vektor pada bus data sistem. Lima bit orde tinggi dari vektor ini harus terlebih dahulu diisi di dalam CTC (merupakan proses inisialisasi CTC) sedangkan 2 bit berikutnya dihasilkan oleh rangkaian ECL CTC sebagai suatu kode biner sesuai dengan saluran yang meminta interupsi. Bit 0 (LSB) selalu harus sama dengan nol.

Vector interrupt digunakan untuk membentuk suatu petunjuk lokasi di dalam memori dimana *address* dari *routine service interrupt* disimpan. Vektor ini merupakan 8 bit orde rendah (LSB) sedangkan CPU akan membaca isi *register* 1 yang dipakai sebagai penunjuk 16 bit. Memori pada *address* yang ditunjuk oleh penunjuk di atas berisi byte orde tinggi dari suatu *address* yang ditunjuk oleh

penunjuk di atas berisi byte orde rendah dan memori pada *address* berikutnya berisi byte orde tinggi dari suatu address yang berisi op-code (instruksi) pertama dari *routine service interrupt*.

3. Konfigurasi Pin CTC Z80

Konfigurasi pin CTC Z80 dapat dilihat pada gambar 3.



Gambar 3.. Konfigurasi pin CTC Z80

(Sumber dari data book Zilog,1986)

Fungsi dari masing-masing pin CTC Z80 adalah sebagai berikut :

a. D0 – D7

   Bus data ini bersifat dua arah (*bidirectional*), *tri state*, digunakan untuk mentransfer semua data dan perintah antara CPU Z80 dan CTC Z80.

b. CS0 – CS1

   CS0 dan CS1 digunakan sebagai pemilih saluran, merupakan sinyal masukan yang aktif pada logika 1. CS0 dan CS1 membentuk kode 2 bit *address biner* yang digunakan untuk memilih salah satu dari empat saluran CTC yang ada untuk menulis atau membaca I/O.

c. $\overline{\text{CE}}$ (Chip Enable)

   Merupakan sinyal masukan, jika CTC berlogika 0 CTC berada dalam keadaan aktif untuk menerima data control, vector interrupt, konstanta waktu data bus Z80

pada saat perioda penulisan (*write*) I/O atau mengeluarkan isi *down counter* ke CPU pada saat perioda pembacaan (*read*) I/O.

d. Clock

Merupakan sinyal masukan, clock 1 fase ini digunakan oleh CTC untuk sinkronisasi di dalam CTC.

e. $\overline{\text{M1}}$(*Machine Cycle One*)

M1 merupakan sinyal masukan yang berasal dari CPU yang akan aktif jika berlogika nol. Jika sinyal M1 dan RD aktif, berarti CPU sedang menerima sebuah permintaan interupsi atau mempersilahkan CTC untuk menempatkan interupsi vektornya di bus data Z80.

f. $\overline{\text{IORQ}}$ (*Input Output Request*)

Merupakan sinyal masukan yang berasal dari CPU, aktif pada logika 0. Sinyal IORQ ini digunakan bersama dengan RD dan CE untuk keperluan mentransfer data dan kontrol saluran antara CPU Z80 dan CTC. Selama perioda pengisian data ke CTC , IORQ dan CE harus berlogika 0 serta RD berlogika 1. CTC tidak memerlukan sinyal *write*. Sinyal *write* akan dihasilkan sendiri di dalam CTC yang diambil dari kebalikan/ *inverse* sinyal RD. pada saat perioda pembacaan dari CTC, IORQ, CE dan RD harus dalam keadaan aktif (berlogika 0) supaya isi *down counter* dikeluarkan oleh CTC ke bus data Z80. Jika IORQ dan M1 aktif, berarti CPU sedang menerima sebuah permintaan interupsi dan saluran yang menginterupsi dengan prioritas tertinggi akan mengeluarkan interupsi vektornya ke bus data Z80.

g. $\overline{\text{RD}}$ (Read)

Merupakan sinyal masukan yang berasal dari CPU, yang akan aktif jika berlogika 0. Sinyal RD bersama dengan sinyal IORQ dan CE digunakan untuk mentransfer data dan control saluran antara CPU Z80 dan CTC. Selama perioda pengisian data ke CTC, IORQ dan CE harus aktif (berlogika 0) dan RD berlogika 1. Pada saat perioda pembacaan dari CTC, IORQ, CE dan RD harus dalam keadaan aktif agar isi *down counter* dikeluarkan oleh CTC ke bus data CTC Z80.

h. IEI (*Interrupt Enable Input*)

Merupakan sinyal masukan, aktif pada logika 1. sinyal ini diperlukan untuk membentuk suatu *uffer interrupt daisy chain* (antrian prioritas) yang besar dengan menerapkan *uffer* prioritas, apabila terdapat lebih dari satu peralatan peripheral di dalam *uffer* yang mempunyai kemampuan menginterupsi. Logika 1 pada pin IEI menandakan bahwa tidak ada peralatan lain yang mempunyai prioritas yang lebih tinggi dari interupsi yang sedang dilayani oleh CPU Z80.

i. IEO (*Interrupt Enable Output*)

Merupakan sinyal keluaran yang aktif pada logika 1. Sinyal IEO dalam hubungannya dengan IEI digunakan untuk membentuk sistem prioritas *interrupt daisy chain* yang besar. IEO akan berlogika 1 hanya pada saat IEI berlogika 1 dan CPU tidak sedang melayani interupsi dari saluran CTC manapun. Jadi sinyal ini mencegah agar perlatan yang mempunyai prioritas yang lebih tinggi sedang dilayani oleh CPU.

j. $\overline{\text{INT}}$ (*Interrupt*)

Merupakan sinyal keluaran, aktif jika berlogika 0. sinyal reset akan menghentikan proses menghitung pada semua saluran dan mereset bit *enable interrupt* dari *register* kontrol pada semua saluran, sehingga mencegah timbulnya interupsi dari CTC. ZC/ TO dan INT akan kembali pada kondisi tidak aktif. Kondisi pin IEO akan sama dengan pin IEI, dan rangkaian pada CTC yang berhubungan dengan bus data kembali ke kondisi impedansi tinggi (*high impedance state*).

k. CLK/ TRG3 – CLK/ TRG0 (*External Clock/ Timer Trigger*)

Merupakan sinyal masukan, aktif pada logika 1 atau 0 tergantung pemakai. Ada empat buah pin CLK/ TRG yang berhubungan dengan ke empat saluran di dalam CTC. Pada mode *counter*, setiap menerima input aktif pada pin ini akan memulai proses penghitungan waktu.

l. ZC/ TO2 – ZC/ TOO (*Zero Count/ Time Out*)

Merupakan sinyal keluaran, aktif pada logika 1. ada 3 pin ZC/ TO yang berhubungan dengan saluran 0 sampai saluran 2 di dalam CTC (karena keterbatasan jumlah pin kemasan, saluran 3 tidak mempunyai pin ZC/ TO). Baik di dalam mode *counter* atau *timer*, jika *down counter* mencapai nilai 0, maka pin ini akan mengeluarkan pulsa 1.

4. Mode Operasi CTC

Pada saat sumber daya di-ON-kan, kondisi CTC Z80 dalam keadaan acak (tidak menentu). Dengan memberikan sinyal reset pada CTC, maka kondisi dari CTC akan berubah menjadi kondisi *counting* atau *timing*, harus dimasukan data saluran kontrol dan data konstanta waktu pada *register-register* yang telah ditentukan pada masing-masing saluran.

Mode *Counter*

Pada mode *counter* CTC menghitung perubahan logik pada *pin* input CLK/ TRG. Pada mode *counter* logik 1 pada bit ke 6 dari *register* kontrol saluran. Input eksternal *clock* (CLK/ TRG) dari tiap-tiap saluran dimonitor perubahan logiknya. Setelah adanya perubahan logik aktif yang disinkronkan dengan perubahan logic 0 ke 1 berikutnya dari sistem *clock*. *Down counter* (yang telah diinialisasi dengan cara konstanta waktu pada saat awal dari setiap proses penghitungan mundur) akan berkurang nilainya dengan 1. eksternal *clock input* dari setiap saluran dapat ditentukan terlebih dahulu apakah proses penghitungan diaktifkan oleh perubahan input logik 0 ke logik 1 atau logik 1 ke logik 0 dengan cara menentukan bit ke 4 dari *register* kontrol saluran.

4.1. Mode Counter

Pada mode *counter* CTC menghitung perubahan logik pada *pin* input CLK/ TRG. Pada mode *counter* sebuah saluran dapat diprogram dengan mengisikan logik 1 pada bit ke 6 dari *register* kontrol saluran. Input eksternal *clock* dari tiap-tiap saluran dimonitor perubahan logiknya. Setelah adanya perubahan logik aktif yang disinkronkan dengan perubahan logik 0 ke 1 berikutnya dari sistem *clock*. *Down counter* (yang telah diinialisasi dengan data konstanta waktu pada saat awal pada setiap proses penghitungan mundur) akan berkurang nilainya dengan 1. eksternal *clock* input dari setiap saluran dapat ditentukan terlebih dahulu apakah proses penghitungan diaktifkan oleh perubahan input logik 1 ke 0 dengan cara menentukan bit ke 4 dari *register* kontrol saluran.

4.2. Mode *Timer*

Pada mode *timer* CTC menghasilkan interval waktu yang merupakan kelipatan bilangan bulat dari perioda sistem *clock*. Pada mode *timer* sebuah saluran di dalam CTC dapat kita program dengan mengisikan logic 0 pada bit 6 dari kontrol saluran (*saluran control word*). Saluran tersebut dapat dipergunakan untuk mengukur interval waktu berdasarkan perioda sistem *clock*. Sistem *clock* dimasukkan melalui dua buah *counter* yaitu *prescaler* dan *down counter*. Prescaler membagi sistem *clock* dengan faktor pembagi pada bit ke 5 dari *saluran control word.* Output dari *prescaler* digunakan sebagai *clock* untuk mengurangi *down counter*, yang dapat diprogram sebelumnya dengan suatu konstanta waktu yang berupa bilangan bulat dari 1 sampai 256. seperti halnya di dalam mode *counter*, nilai konstanta waktu ini secara otomatis akan diisi kembali ke dalam *down counter* jika kondisi *zero count*, *output* dari ZC/TO akan mengeluarkan pulsa yang akan membentuk deretan pulsa yang mempunyai perioda yang tetap. Perioda dapat dihitung dengan rumus :

Perioda = tc * P * TC,

tc = perioda dari sistem *clock*

P = pembagi *prescaler* (16 atau 256)

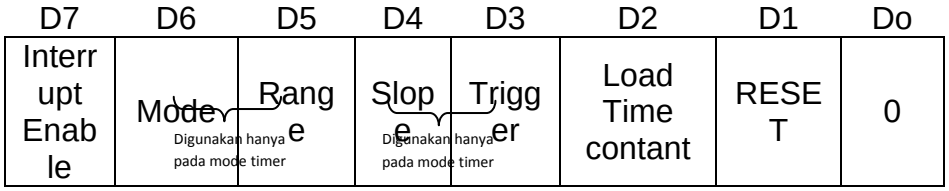
TC = konstanta waktu.

Bit ke 3 dari *saluran control word* diprogram untuk memilih apakah proses penghitungan waktu secara otomatis berlangsung ataukah proses tersebut di atas baru akan dimulai jika diberikan input pada CLK/ TRG. Jika bit ke 3 = 0, *timer* akan dimulai secara otomatis setelah CPU menuliskan data konstanta waktu ke saluran CTC. Jika bit ke 3 = 1, *timer* akan memulai operasinya pada saat perubahan logik 0 ke 1 yang ke dua dari sistem *clock* setelah adanya perubahan logik aktif *time trigger input* (setelah data konstanta waktu diisikan terlebih dahulu). Jika data konstanta waktu belum diisikan, maka *timer* akan memulai operasinya pada saat perubahan logic 0 ke 1 yang kedua dari sistem *clock* setelah adanya perubahan logik aktif *time trigger input*, setelah proses pengisian CTC *control word*. Bit ke 4 dari *saluran control word* dapat diprogram untuk memilih apakah *time trigger input* sensitif terhadap perubahan logic 0 ke 1. Jika bit ke 7 dari *saluran control word* berlogik 1,

maka kondisi *zero count* pada *down counter* selain menghasilkan pulsa pada pin saluran *time out* juga digunakan untuk menginisialisasi proses permintaan interupsi.

4.3. Pemrograman CTC

Sebelum saluran di dalam CTC Z80 dapat memulai operasi *counting* atau *timing* terlebih dahulu harus dimasukkan data kontrol saluran (*saluran control word*) dan data konstanta waktu (*time constant data*) oleh CPU. *Saluran control word* diisi oleh CPU dengan cara melakukan proses penulisan (*write*) I/O pada *address port* yang sesuai dengan saluran CTC yang dikehendaki. Dua buah input pin input CTC yaitu CS0 dan CS1, digunakan untuk memilih saluran dari empat saluran yang berada di dalam CTC dengan memberikan *address* biner 2 bit padanya. Sebuah data yang diisi ke dalam sebuah saluran CTC akan diartikan sebagai "*saluran word*" akan diisikan ke dalam *saluran control word register* dimana bit ke 0 nya berlogik 1. Arti dari tiap – tiap bit akan dijelaskan di bawah ini :



Gambar 4. Definisi bit-bit dari saluran control register

Bit 0, selalu = 0.

Bit 1 = 1. saluran akan menghentikan proses *counting* maupun *timing*, tetapi isi dari *saluran control word register* tidak berubah. Jika bit 1 dan bit 2 sama dengan 1, maka saluran akan mengulangi operasinya dengan mengisi konstanta waktu.

Bit 1 = 0 . saluran meneruskan operasinya.

Bit 2 = 1. Data *word* yang akan dimasukan berikutnya pada saluran tersebut dianggap sebagai data konstanta di *time constant register*. Jika data *control word* dan konstanta waktu baru dimasukkan di dalam sebuah saluran yang

sedang operasi, *down counter* tetap akan meneruskan perhitungannya sampai mencapai nol, setelah itu konstanta waktu yang baru diisikan ke *down counter* tersebut.

Bit 2 = 0. Tidak diperlukan konstanta waktu baru. Bit 2 = 0 bertujuan untuk mengubah status dari saluran yang sedang beroperasi.

Bit 3 = 1. Hanya digunakan pada mode *timer*. Trigger dari luar hanya berpengaruh untuk memulai operasi *timer* setelah perubahan logik 0 ke 1 dari perioda mesin T yang terjadi setelah pemasukan data konstanta waktu.

Bit 3 = 0. Hanya digunakan pada mode *timer*. Trigger dari luar hanya berpengaruh untuk memulai operasi *timer* setelah perubahan logik 0 ke 1 dari perioda mesin T dari *machine cycle* berikutnya setelah pengisian konstanta waktu.

Bit 4 = 1. Pada mode *timer*, perubahan logik 0 ke 1 merupakan perubahan logik aktif yang akan memulai operasi *timer*. Pada mode *counter*, perubahan logik 0 ke 1 merupakan perubahan logik aktif yang akan mengurangi isi *down counter*.

Bit 4 = 0. pada mode *timer*, perubahan logik 1 ke 0 merupakan logik aktif yang akan memulai operasi timer. Pada mode *counter*, perubahan logik 1 ke 0 merupakan perubahan logik aktif yang akan mengurangi isi *down counter*.

Bit 5 =1. Hanya digunakan pada mode *timer*. Faktor pembagi *prescaler* = 256.

Bit 5 = 0. Hanya digunakan pada mode *timer*. Faktor pembagi *prescaler* = 16.

Bit 6 = 1. CTC bekerja pada mode *counter*. Bila terjadi perubahan logik aktif pada input eksternal *clock* (CLK/ TRG) *down counter* akan berkurang 1. *prescaler* tidak digunakan.

Bit 6 = 0. CTC bekerja pada mode *timer*. *Prescaler* mendapat input *clock* dari *uffer clock* dan output dari *prescaler* memberi *clock* pada *down counter*. Output dari *down counter* (output dari saluran ZC/ TO) akan mengeluarkan deretan pulsa yang tetap dengan perioda yang dapat dihitung menggunakan rumus perioda = tc * P * TC.

Bit 7 = 1. Untuk menset bit ini menjadi 1 di dalam setiap *register* control diperlukan pengisian vector *interrupt* pada CTC sebelum operasi dimulai. Saluran ini

dapat menghasilkan sinyal permintaan interupsi setiap *down counter* mencapai kondisi *zero count*. Saluran interupsi dapat diprogram baik dalam mode *counter* maupun *timer*. Jika sebuah saluran *control word* yang baru (dengan bit 7 dalam keadaan set) diisikan ke dalam sebuah saluran yang sedang beroperasi, kondisi *zero count* yang terjadi sebelumnya tidak akan menghasilkan sinyal permintaan interupsi.

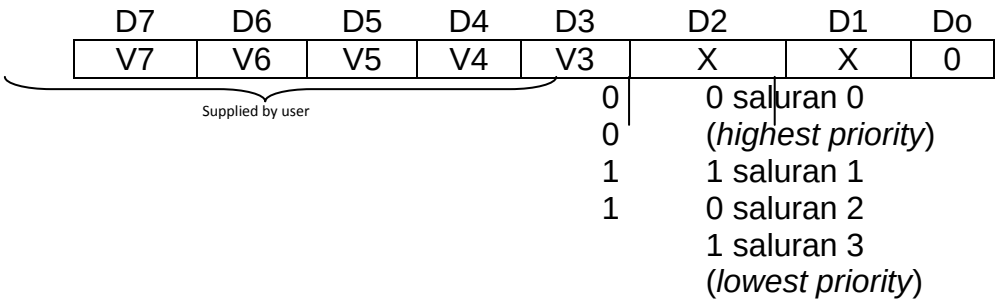
Bit 7 = 0. *Interrupt disable*, saluran tidak akan menghasilkan sinyal permintaan interupsi.

Mengisi *register* konstanta waktu (*time register constant*).

Jika data kontanta waktu belum diisi ke dalam *time constant register* oleh CPU maka sebuah saluran tidak dapat memulai operasinya baik dalam mode *timer* maupun dalam mode *counter*. Data ini diperlukan pada penulisan I/O untuk mengisi saluran *control word* dengan bit 2 dari saluran control word = 1. data konstanta waktu harus merupakan bilangan bulat antara 1 sampai 256. jika data konstanta waktu diisi ke saluran yang sedang beroperasi, *down counter* akan meneruskan proses pengurangannya sampai mencapai nol sebelum konstanta waktu yang baru tersebut diisi ke *down counter* dari *time constant register*.

Mengisi *register vector interrupt*.

CTC Z80 dirancang untuk beroperasi bersama dengan CPU yang diprogram dengan interupsi mode 2 sesuai dengan kebutuhan mode ini, jika saluran CTC meminta pelayanan interupsi dan diterima oleh CPU, sebuah petunjuk (*pointer*) 16 bit harus dibentuk untuk menentukan *address* awal *routine service interrupt* yang sesuai. Delapan bit orde tinggi dari pointer tersebut diambil dari *register* didalam CPU sedangkan 8 bit orde rendahnya dihasilkan oleh CTC dalam bentuk sebuah interupsi vektor yang unik untuk tiap-tiap saluran.



Gambar 5. Definisi bit-bit dari *interrupt vector register*

Lima bit orde tinggi dari iterupsi vektor ini harus diisi ke dalam CTC pada saat pengisian program inisialisasi CTC. Untuk melakukan hal ini, CPU harus menuliskan pada port I/O dengan alamat yang sesuai dengan saluran 0 CTC sama dengan mengisi mengisi saluran *control word* tetapi bit 0 = 0. jika bit 0 dari daya yang dimasukkan = 1, maka data tersebut dianggap sebagai saluran control word, sedangkan bila bit 0 = 0 data tersebut dianggap sebagai iterupsi vektor dan akan dimasukan ke dalam register interupsi vektor. Bila bit 1 dan bit 2 dari vektor ini tidak perlu diperhatikan. Pada saat saluran yang bersangkutan harus menempatkan iterupsi vektor pada bus data Z80, secara otomatis interupsi kontrol logik yang ada di dalam CTC akan menentukan ke dua bit tersebut yang akan membentuk sebuah kode biner untuk mengidentifikasi salah satu dari ke empat saluran CTC yang sedang dilayani.

Contoh pemrograman CTC :

```
LD A, 18H
LD I, A
LD A, 10110101B
OUT (CTCO), A
LD A, 020H
LD (CTCO), A
LD A, 0A8H
OUT (CTCO), A
IM 2
EI
```

Pada contoh di atas, karena pada *register* interupsi vektor bit 1 dan bit 2 nya masing-masing bernilai 0 maka pemograman dilakukan pada saluran 0. saluran register kontrol berisi 10110101B berarti saluran 0 terprogram dengan mode *timer*. Karena bit 5 di dalam saluran *word* = 1, maka *prescaler* membagi sistem *clock* dengan faktor 256. isi dari *time constant register* adalah 020H, sehingga saluran 0 dari CTC akan menghasilkan sinyal permintaan interupsi setiap *down counter*-nya mencapai kondisi *zero count*. CTC akan menghasilkan sinyal interrupt setiap 256 * 32 = 8192 kali sistem *clock*. Sinyal reset membuat *program counter* menjadi 0 dan menginisialisasi CPU. Inisialisasi CPU menyangkut men-*disable interrupt flip flop* akan men-set *register* I = 00H. IM2 berfungsi untuk memprogram CTC dengan interupsi mode 2.

7.5. Soal Latihan

1. Jelaskan secara singkat cara menginisialisasi PPI 8255 !
2. Bagaimana status dari *port* A, B dan C jika *control word* berisi nilai 8AH, 93H dan 99H?
3. Buatlah program inisialisasi untuk komponen PPI 8255, sehingga *port* A berfungsi sebagai input, *port* B sebagai output dan *port* C sebagai output ?.
4. Apakah CTC itu ?
5. Apakah fungsi prescaler pada CTC Z80 ?
6. Bagaimana cara memprogram CTC agar berfungsi sebagai *counter* ?
7. Jelaskan fungsi dari SIO Z 80 !

8.Referensi :

1. Laventhal, (1986). *Z80 Assembly Language Programming,* Mc Graw Hill, Singapore.
2. Hall, (1985), *Microprocessor and Digital Systems*, Mc Graw Hill.
3. Rodnay Zaks and Austin Lesea. (1979). *Microprocessor Interfacing Techniques.* Sybex Inc.
4. Hartono Partoharsodjo. (1990).*Bahasa Assembly***.** Jakarta : PT. Elek Media Komputindo.
5. James W. Coffron. (1983). *Practical Hardware Details For 8080,Z80,and 6800.*
6. Inelco, (1986). *Guru Mikro Saya.*
7. Brey, Barry B. (2003). *The intel microprocessors : 8086/8088/80186/80286/80386/80486, Pentium, Pentium Pro processor, Pentium II, Pentium III, and Pentium 4: architecture, programming, and Interfacing*- 6 th ed. New Jersey : Pearson Education.

8. Greenfield, Joseph D.(1992). *The 68HC11 Microcontroller*.Orlando, FL:
9. Puadi. (1995). *Upaya Pengembangan Kegunaan TrainerBGC 8088 V3 Sebagai Alat Bantu Belajar Mengajar*. Bandung: IKIP.
10. Endra Pitawarno, (2005). *MIkroprosesor dan Interfacing*. Penerbit ANDI Yogyakarta.
11. Data book Zilog dan Intel 1986.