1

2 Desired Mechanism

Imagine an embedded computer, based on the famous **Z80** CPU, which has no display means but a single LED. Immediately after power up the board has to give a life sign by flashing the LED every t seconds as a kind of a heartbeat.
The main **program** running on the board shall not be affected by the heartbeat function, except it's interruption every t seconds of course.

3 CTC Device Structure and external wiring

Figure 1 shows the block diagram of the **CTC** device with its 4 timer/counter channels and the two channels we need marked red. Figure 2 shows the structure of a single channel.
Note: The output of channel 3 is not connected to any pin.

3.1 Wiring

Data and control: These are the **Z80** bus signals D[7:0], A[1:0] or CS[1:0], /RD, /IOREQ, /CE, /RESET and CLK.

Interrupt Control Lines: /M1, /INT connected to CPU, IEI and IEO daisy chained to other periphery

Outputs: zero count signals TO[2:0], TO2 is wired to TRG3

Inputs: counter inputs TRG[3:0]

2

Figure 1: **CTC** Block Diagram

Figure 2: Channel Structure

3

4 Programming

Tree problems have to be solved: initializing the **CTC**, implementing the interrupt mechanism and writing an interrupt service routine that handles the flashing of the LED.

4.1 Header

The header shown below defines the hardware addresses of the control port of your four **CTC** channels 0 to 3. In my case here the addresses equal the channel number.

```
CH0 equ 0h
CH1 equ 1h
CH2 equ 2h
CH3 equ 3h
```

Text 1: header

4.2 Interrupt table

Every time **CTC** channel 3 count equals zero an interrupt is triggered causing the CPU to jump to the memory address specified by the term CT3_ZERO.

```
org 16h
DEFW CT3_ZERO
```

Text 2: **CTC** interrupt vector table

4.3 Initializing the **CTC**

First we have to configure the **CTC** channels as shown in Text 3. We don't need channel 0 and 1. They are on hold. We operate the **CTC** channel 2 as frequency divider which scales the CPU clock of 5 MHz down by factor 256*256. The output TO2 of channel 2 drives input TRG3 of channel 3 which further divides by factor AFh. This causes channel 3 to zero count at a frequency of approximately 0.44Hz. So the interrupt occurs every 2.3 seconds. For detailed information on the purpose of certain registers and control bits please read the **CTC** datasheet.

```
INI_CTC:
;init CH 0 and 1
ld  A,00000011b ; int off, timer on, prescaler=16, don't care ext. TRG edge,
; start timer on loading constant, no time constant follows
; swrst active, this is a ctrl cmd
out  (CH0),A ; CH0 is on hold now
out  (CH1),A ; CH1 is on hold now
```

```
;init CH2
;CH2 divides CPU CLK by (256*256) providing a clock signal at TO2. TO2 is connected to TRG3.

ld  A,00100111b ; int off, timer on, prescaler=256, no ext. start,
; start upon loading time constant, time constant follows
; sw reset, this is a ctrl cmd
out  (CH2),A
ld A,0FFh ; time constant 255d defined
out  (CH2),A ; and loaded into channel 2
; T02 outputs f= CPU_CLK/(256*256)



;init CH3
;input TRG of CH3 is supplied by clock signal from TO2
;CH3 divides TO2 clock by AFh
;CH3 interupts CPU appr. every 2sec to service int routine CT3_ZERO (flashes LED)
ld  A,11000111b ; int on, counter on, prescaler don't care, edge don't care,
; time trigger don't care, time constant follows
; sw reset, this is a ctrl cmd
out  (CH3),A
ld A,0AFh ; time constant AFh defined
out  (CH3),A ; and loaded into channel 3

ld A,10h ; it vector defined in bit 73,bit 21 don't care, bit 0 = 0
out  (CH0),A ; and loaded into channel 0
```

Text 3: configure the **CTC**

5

4.4 Initializing the CPU

The CPU is to run in interrupt mode 2. See Text 4 below. This setting has to be done after initializing the **CTC**.

```
INT_INI:
ld A,0
ld I,A ;load I reg with zero
im 2 ;set int mode 2
ei ;enable interupt
```

Text 4: set up the CPU interrupt mode 2

4.5 Interrupt routine

Upon zero count of channel 3 the routine CT3_ZERO as shown in Text 5 is executed. Here you put the code that switches the LED on and off.
Note: In this example we backup only register AF. Depending on your application you might be required to backup more registers like HL, DE, CD, …

```
CT3_ZERO:
;flashes LED
push AF ;backup registers A and F

; now address your periphery that turns the LED on/off e.g. a DFlipFlop

pop AF ;restore registers A and F
EI ;reenable interrups
reti
```

Text 5: CT3 zero count routine

5 **Z80** IC equivalents table

An overview of ICs of the famous **Z80** family gives Table 1.

device	equivalent type
Z80CPU	BU18400APS (ROHM)
	D780C1(NEC)
	KP1858BM1/2/3 / KR1858BM1/2/3 (USSR)
	LH0080 (Sharp)
	MK3880x (Mostek)
	T34VM1 / T34BM1 (USSR)
	TMPZ84C00AP8 (Toshiba)
	UA880 / UB880 / VB880D (MME)
	Z0840004 (ZiLOG)
	Z0840006 (ZiLOG)
	Z80ACPUD1 (SGSAtes)
	Z84C00AB6 (SGSThomson)
	Z84C00 (ZiLOG)
	Z8400A (Goldstar)
	U84C00 (MME)
Z80SIO	UA8560 , UB8560 (MME)
	Z0844004 (ZiLOG)
	Z8440AB1 (ST)

Z0844006 (ZiLOG)
Z84C40 (ZiLOG)
U84C40 (MME)
Z80PIO Z0842004/6 (ZiLOG)
UA855 / UB855 (MME)
Z84C20 (ZiLOG)
U84C20 (MME)
Z80CTC Z84C30 (ZiLOG)
U84C30 (MME)
UA857 / UB857 (MME)

Table 1: **Z80** equivalents

6 Useful Links

A complete embedded **Z80** system can be found at
http://www.trainz.de/trainz

The Free and Open Productivity Suite OpenOffice at http://www.openoffice.org
Z80 assembler for Linux and UNIX at http://www.unix4fun.org/z80pack/
The powerful communication tool Kermit at http://www.columbia.edu/kermit/
The **Z80** history at http://en.wikipedia.org/wiki/Zilog_Z80

EAGLE an affordable and very efficient
schematics and layout tool at
http://www.cadsoftusa.com/

7 Disclaimer

This tutorial is believed to be accurate and reliable. I do not assume responsibility for any errors which may appear in this document. I reserve the right to change it at any time without notice, and do not make any commitment to update the information contained herein.

My Boss is a Jewish Carpenter !