# TOD initialisation - the problem

Each 6526 CIA chip as used in many Commodore machines has the so-called TOD (for Time Of Day) clock, capable of tracking elapsed time in human readable format (hours, minutes, seconds and tenths of seconds). While not offering the granularity of regular, binary timers, It is highly useful for measuring and displaying time in the form we are well conditioned to use every day. The registers' output is BCD encoded to make it even easier to work with base-10 digits and numbers. There is however one caveat. TOD clock circuitry requires pulses of stable frequency of either 50 or 60 Hz being supplied to a dedicated pin. Such pulses can be inexpensively derived from the mains power AC, which was presumably the reason for designing the chip this way. The ability to work with both standard power line frequencies allowed a single version of the 6526 to be used in countries with either 50 or 60 Hz mains power lines. The only thing, which needs to be taken care of is informing 6526 which frequency is actually supplied. This is done by setting/clearing bit seven of CRA ($0e) register of the CIA. KERNAL operating system in C64 initialises both CIA chips as part of the startup/reset sequence, inside IOINIT routine. This routine however does not try to identify what frequency is supplied but unconditionally sets both 6526s to expect 60Hz instead. Obviously, whenever in reality this is not the case (e.g. in C64 machines used in virtually all European countries), TOD clock drifts immediately away. Because of this, every application software, before attempting to utilise TOD as its timekeeping/alarm helper, needs to identify the frequency supplied to the TOD pin and initialise the required CIA's CRA register accordingly. Here comes the question “how” to reliably identify the actually supplied frequency?

## Popular misconception

First let's say how NOT to do it. It is important to note that contrary to a very popular and utterly wrong belief, NTSC or PAL video/colour encoding standards are NOT directly linked to mains power frequency. This means that checking which video standard the machine is of, should never be used to make assumptions about what the TOD frequency is. Not only PAL computers can be used in countries with 60Hz AC (and vice versa - NTSC computers can be run of 50Hz power) but in addition to that some computers do not derive their TOD clock frequency from the mains power source at all. For example both NTSC and PAL variants of Commodore SX-64 use 60 Hz TOD clock supplied by a dedicated crystal.

## Proper solution

While there are various approaches to this problem, we believe the one presented below is not only the most compact but can also serve dual purpose (see below), increasing the byte-size savings even further.

```
; Detecting TOD frequency by Silver Dream ! / Thorgal / W.F.M.H.
tod_freq:
    sei             ; accounting for NMIs is not needed when
    lda #$00        ; used as part of application initialisation
    sta $dd08       ; TO2TEN start TOD - in case it wasn't running
:   cmp $dd08       ; TO2TEN wait until tenths
```

```
    beq :-          ; register changes its value

    lda #$ff        ; count from $ffff (65535) down
    sta $dd04       ; TI2ALO both timer A register
    sta $dd05       ; TI2AHI set to $ff

    lda #%00010001  ; bit seven = 0 - 60Hz TOD mode
    sta $dd0e       ; CI2CRA start the timer

    lda $dd08       ; TO2TEN
:   cmp $dd08       ; poll TO2TEN for change
    beq :-

    lda $dd05       ; TI2AHI expect (approximate) $7f4a $70a6 $3251 $20c0
    cli

    cmp #$51        ; about the middle (average is $50c0)
    bcs ticks_60hz
    ; 50Hz on TOD pin

ticks_60hz:
    ; 60Hz on TOD pin
```

And that's it. Really  At this point we already know what the actual frequency is. If you don't know how this is done then a few explanations follow. We use timer 2 of second CIA to count the number of CPU cycles between two consecutive changes of TO2TEN register (CIA #2 - TOD, tenths of seconds) and – after accounting for the overhead cycles coming from fetching and executing the instructions between changes – expect the result to be approximately one of the four values:

- `$7f4a for 60Hz TOD clock and 985248.444 CPU/CIA clock`
- `$70a6 for 60Hz TOD clock and 1022727.14 CPU/CIA clock`
- `$3251 for 50Hz TOD clock and 985248.444 CPU/CIA clock`
- `$20c0 for 50Hz TOD clock and 1022727.14 CPU/CIA clock`

which correspond to the four possible C64 hardware setup combinations. Comparing to $51, which falls more or less in the middle, gives us the expected answer. If the HI byte of the timer has value higher than $51 we have 60Hz supplied to the TOD pin. If OTOH it has lower value, we have 50Hz supplied.

Advantages:

- Does not break on Super-CPU and similar
- No screen side-effects
- Short and fast

Please also note that we use CIA #2 and not CIA #1. CIA #2 is chosen because changing the timer values there does not affect regular IRQ timings. Moreover, KERNAL re-initialises those timers whenever it wants to use them. This saves as a few bytes, which would otherwise be needed to save and restore timer registers' original values. Last and least we do not disable NMIs as it is assumed that checking/setting the params will be done as part of application initialisation, before setting up IRQ/NMI handlers.

# Dual purpose

I mentioned before that this routine can serve also another purpose and give us potentially even more savings. Yes, with addition of only a few bytes:

```
    cmp #$29        ; about the middle between $20(c0) and $32(51)
    bcs pal50
    ; we run on NTSC machine with 50Hz TOD clock
pal50:
    ; we run on PAL machine with 50Hz TOD clock

ticks_60hz:
    cmp #$78        ; about the middle between $70(a6) and $7f(4a)
    bcs pal60
    ; we run on NTSC machine with 60Hz TOD clock
pal60:
    ; we run on PAL machine with 60Hz TOD clock
```

we can determine not only the TOD frequency but also the video norm. This is possible because – unlike TOD frequency – the CPU clock frequency is directly related to the computer's video standard. PAL computers have their CPU (and CIAs) clocked at 985248.444Hz while NTSC ones run faster and have their CPU/CIAs clocked at 1022727.14Hz. This is a side-effect of deriving all (except TOD) required clock frequencies from a single crystal of either 14318180Hz (for NTSC machines) or 17734472Hz (for PAL ones).