ESP32forth and Arduino C++ - notes about ESP32 exact time with SNTP syncing.

ESP32forth is written in Arduino C and some knowledge of C is very helpful. As amateur programmer, with basic knowledge of Forth only, I resolved to learn also basics of Arduino C to be able to better understand and use ESP32forth. I created some notes about my examples to help others in the same situation.

ESP32 chip has WiFi onboard and ESP32forth support for it. My next experiments are based on free texts on excellent web site *https://techtutorialsx.com/2021/09/01/esp32-system-time-and-sntp/* about synchronization of internal ESP32 timer with help of SNTP connecting to internet time server. ESP32 chip is equiped with internal timer running independently of users program. It is only necessary to adjust it with correct date time information. Functions for it are part of Arduino ESP32 core, so it is only necessary to open it for ESP32forth usage.
At first I tested Arduino C++ code from above mentioned web site, which runs perfectly. Code is also on my github page about this article.
For usage in ESP32forth I created simple userwords.h file and compiled it together with ESP32forth.ino .

```
#define USER_WORDS \
Y(configTime, configTime( n2, n1, c0); DROPn(3); ) \
Y(getLocalTime, n0=getLocalTime((tm *) a0); ) \
Y(asctime, const char* atime=asctime((tm *) a1); strcpy(c0, atime); NIP; )
```

Stack diagrams for new words:
configTime (gmtoffset daylightoffset z-ntp ---) synchronize ESP32 time with ntp server, needs internet connection running
getLocalTime (tmstructure---t/f) fill tm structure with actual time from ESP32 internal timer
asctime (tmstructure timestring---z-timestring) create z-string with date time info from tm structure

Now ESP32forth program to use this new functions. At first necessary definitions.

```
also structures decimal
struct tm ( --|-- size of struct) \ definition of structure tm
    i32 field tm_sec    \ int seconds after the minute 0-61
    i32 field tm_min    \ int minutes after the hour 0-59
    i32 field tm_hour   \ int hours since midnight 0-23
    i32 field tm_mday    \ int day of the month 1-31
    i32 field tm_mon    \ int months since January 0-11
    i32 field tm_year   \ int years since 1900
    i32 field tm_wday   \ int days since Sunday 0-6
    i32 field tm_yday   \ int days since January 1 0-365
    i32 field tm_isdst  \ int Daylight Saving Time flag

create time tm allot    \ creation of structure time with tm content
    time tm erase       \ erase new structure time
create atime 80 allot   \ buffer for time string
    atime 80 erase
: ntpServer ( ---z-addr )
    z" time.google.com"
;
```

Structure time is used by getLocalTime function , which fills values from internal ESP32 timer. I use google time server, but possible to adjust different one.
Of course network connection is mandatory for syncing.

```
: wificonnection    ( -- )          \ connect to my wifi
    z" yourSSID" z" yourPSW" login
;
: syncTime  ( --- )     \ synchronize ESP32 time with SNTP
    3600 3600 ntpServer configTime
;
: createTime ( --- )     \ fill time structure with time data
    time getLocalTime
    if cr ." Time info obtained"
       else cr ." Time info error"
    then
;
```

After internet connection is possible to do synchronization with *syncTime* and fill time structure with actual data from internal timer. And finally it is possible to use obtained values in next programs. There are two possibilities.

```
: printTime ( ---)          \ print current date time info line
    createTime              \ fill time structure with actual time
    time atime asctime      \ fill atime buffer and print
    cr z>s type cr
;
: year? ( --- year_number)          \ return year from time struct
    time tm_year @ 1900 +
;
: month? ( ---month_number)
    time tm_mon @ 1+
;
: day? ( ---day_number)
    time tm_mday @
;
```

First it is to use *asctime* function, which internally elaborates on time structure and creates date time string for simple print on terminal. Or it is possible to use values from *timer* struct directly as is in examples *year?*, *month?* and *day?* Of course to have exact information it is necessary to refresh time structure with values from internal timer before usage.
This time information is much easier to obtain than I did in my previous experiments with WorldTimeAPI, which was slow and complicated. If I understand correctly if we use core configTime function there is automatic synchronization with used time server after default time 1 hour, but I did not do any tests in this.
According my test internal timer runs correctly after soft reset with *bye* command, but is reseted with hw button reset.
Next is screen copy from MyFORTHshell terminal window with obtained results from ESP32 forth program.

```
 ok
--> wificonnection
192.168.1.59
MDNS started
 ok
--> synctime
 ok
--> createtime

Time info obtained ok
--> printtime

Time info obtained
Wed Jan  8 12:45:38 2025

 ok
--> year?
 ok
2025 -->
 ok
```

If it helps somebody I will be pleased, if there are errors in my explanation I will be also pleased to be corrected.
Files from my article are located on github:
https://github.com/Vaclav-Poselt/ESP32_forth_code

Final note: tested on ESP32forth 7.0.7.20, Arduino IDE2.3.2, ESP32 lib 2.0.14, ESP32 Dev Module.