ESP32forth and Arduino C++ - notes about ESP32 Deep Sleep.

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. In this text there are some notes about power mode Deep Sleep.

ESP32 chip has more power modes to reduce power consumption, which can help in constructions with battery power supply. My experiments are based on free text on excellent web site https://randomnerdtutorials.com/esp32-deep-sleep-arduino-ide-wake-up-sources/ , so I don't put C code here, possible to find there.
In ESP32forth is support for Deep Sleep mode in vocabulary ESP. With word deepsleep (time in microsec---) it is possible to lower consumption to fraction of normal run consumption for desired time in microseconds. This is Timer Wake Up method. In Deep Sleep mode remain active only RTC SRAM memory and ULP coprocessor, the rest of chip is off, so no CPU, no WiFi/BT, no SRAM program memory. ESP32forth user program is stored in SRAM, so during Deep Sleep it is lost. After wake-up processor is reseted, ESP32forth restarts. Best solution for forth program using Deep Sleep is use of autoexec.fs file in SPIFFS memory (Flash memory not affected by Deep Sleep). So usage of Deep Sleep is suitable for programs doing repetitive tasks with bigger time delay as meteorologic stations. Whole cycle could be Start - do measurements - store data to some non volatile memory or send out by WiFi/LoRa/BT wireless - go Deep Sleep and reStart again whole cycle. No possibility to only freeze status and after wake up continue in program.
For demo I created simple program DPdemo.fs activated from autoexec.fs after restart with command *include /SPIFFS/DPdemo.fs* in the end of autoexec.fs. For creation/modification of files in /SPIFFS/ I use Bob Edwards program *RECORDFILE ("filename" "filecontents" "<EOF>" --),* I have *RECORDFILE* command in my autoexec.fs together with better *dump* command to show memory content.

```
\ DPdemo.fs with subs words of some real program
cr ." OK, here I am after restart of ESP32forth." 1000 ms cr
: first ." measuring no 1" cr 1000 ms ;
: second ." measuring no 2" cr 1000 ms ;
: third ." meauring no 3" cr 1000 ms ;
: save-send ." now sending/saving measurements" cr 1000 ms ;
esp
: mainprg \ ( --- ) program to run
    first second third
    save-send
    ." going to Deep Sleep for 5 secs"
    5000000  deepsleep
    ;
mainprg
```

The output in terminal window looks like this, also notice source of reset - DEEPSLEEP RESET, which is important for next section about RST memory:

```
OK, here I am after restart of ESP32forth.
measuring no 1
measuring no 2
meauring no 3
now sending/saving measurements
going to Deep Sleep for 5 secsets Jul 29 2019 12:21:46

rst:0x5 (DEEPSLEEP_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:1344
load:0x40078000,len:13964
load:0x40080400,len:3600
entry 0x400805f0

OK, here I am after restart of ESP32forth.
measuring no 1
measuring no 2
meauring no 3
now sending/saving measurements
going to Deep Sleep for 5 secsets Jul 29 2019 12:21:46
```

For end of this repeating cycle activated by autoexec.fs press some button on terminal keyboard during reset to skip autoexec.fs and go back to command line.
Of course I have measured consumption. In normal mode my ESP32 dev board used 75 mA, in Deep Sleep 6 mA, with activated radio of WiFi it was 155mA. Than Deep Sleep consumption is more than according ESP32 specification, but I measure whole board with USB using USB go through meter.

I was also interested in RTC SRAM memory, which remains active during Deep Sleep. From data sheet there is address region from 0x3FF8:0000 to 0x5000:1FFF twice 8kB called slow and fast RTC memory. In Arduino C it is possible to declare usage by
RTC_DATA_ATTR int RTCvar1 =0 ; // variable in RTC memory
In ESP32forth it is possible to dump content or access this memory with *L! and UL@*, but I was afraid to not damage something used by ESP32 itself. So I created simple *userwords.h* file to do some experiments with RTC memory. After compilation there are new words in ESP32forth:
RTC1! (n---) \ save integer into RTC1 variable
RTC1@ (---n) \ read integer from RTC1 var
RTC1addr (---addr) read address of RTC1 variable in RTC memory
RTCstr! (str-z---) \ store c-type string max 31 chars
and so on.
I was interested in addresses of this variables. My idea was if I declare it the space they occupy will be reserved so next direct access from ESP32forth will be correct and safe.

Userwords.h:

```
#include <cstring>

RTC_DATA_ATTR int RTCvar1 =0 ;    // variable in RTC memory
RTC_DATA_ATTR int RTCvar2 =0 ;    // variable in RTC memory
RTC_DATA_ATTR char RTCstr[32];    // char variable in RTC memory
void savetoRTC1(int value1)          // save value to  RTC
    { RTCvar1= value1; }
void savetoRTC2(int value1)          // save value to  RTC
    { RTCvar2= value1; }
void savetoRTCstr(char value3[32])
    { strcpy(RTCstr,value3);}

#define USER_WORDS \
X("RTC1!", savetoRTC1, savetoRTC1(n0); DROP; ) \
X("RTC1@", fromRTC1, PUSH RTCvar1; ) \
X("RTC1addr", RTC1addr, PUSH (uintptr_t)&RTCvar1; ) \
X("RTC2!", savetoRTC2, savetoRTC2(n0); DROP; ) \
X("RTC2@", fromRTC2, PUSH RTCvar2; ) \
X("RTC2addr", RTC2addr, PUSH (uintptr_t)&RTCvar2; ) \
X("RTCstr!", savetoRTCstr, savetoRTCstr(c0); DROP; ) \
X("RTCstr@", fromRTCstr, PUSH RTCstr; ) \
X("RTCstraddr", RTCstraddr, PUSH (uintptr_t)&RTCstr; )

```

And simple test, all in hexadecimal:

```
--> hex
 ok
--> 123 RTC1! RTC1@ RTC1addr
 ok
123 50000224 --> 456 RTC2! RTC2@ RTC2addr
 ok
123 50000224 456 50000220 --> z" Hello FORTH" RTCstr! RTCstr@ RTCstraddr
 ok
123 50000224 456 50000220 50000200 50000200 -->
 ok
123 50000224 456 50000220 50000200 50000200 --> z>s type
Hello FORTH ok
123 50000224 456 50000220 50000200 -->
```

What is there - addresses of declared RTC vars start at 0x5000:0200, according datasheet it is area of RTC slow memory. First is area for 32 bytes of RTCstr, next twice 4 bytes for 32bit integers. It is now possible safely use this data places in RTC memory directly if we need, not only use created RTC words.

Now we try deepsleep.

```
--> esp 1000000 deepsleep
ets Jul 29 2019 12:21:46

rst:0x5 (DEEPSLEEP_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:1344
load:0x40078000,len:13964
load:0x40080400,len:3600
entry 0x400805f0
ESP32forth v7.0.7.20 - rev fe6acd0bb6b6ea5485788f4b26f97624bba9b0fe
ESP32-D0WD-V3   240 MHz   2 cores   4194304 bytes flash
     System Heap: 174204 free + 314708 used = 488912 total (35% free)
                  86004 bytes max contiguous
Forth dictionary: 70464 free + 33868 used = 104332 total (67% free)
3 x Forth stacks: 2048 bytes each
 ok
--> hex RTC1@ RTC2@
 ok
123 456 --> RTCstr@ z>s type
Hello FORTH ok
123 456 -->
```

Data remains in memory and can be used consistently between *deepsleep* commands.
And now reset with RESET button.

```
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:1344
load:0x40078000,len:13964
load:0x40080400,len:3600
entry 0x400805f0
ESP32forth v7.0.7.20 - rev fe6acd0bb6b6ea5485788f4b26f97624bba9b0fe
ESP32-D0WD-V3   240 MHz   2 cores   4194304 bytes flash
     System Heap: 174204 free + 314708 used = 488912 total (35% free)
                  86004 bytes max contiguous
Forth dictionary: 70464 free + 33868 used = 104332 total (67% free)
3 x Forth stacks: 2048 bytes each
 ok
--> hex RTC1@ RTC2@
 ok
0 0 --> RTCstr@ z>s type
 ok
```

Data in RTC SRAM are lost. Exactly according specs, here is visible limited usability of RTC SRAM memory. Every button reset, power-off reset will erase stored data. So for safe storage in construction using Deep Sleep is necessary to use more robust memory as SPIFFS or NVM flash with *preferences.h* as I explained in previous article. Flash has teoretical limit in Write cycles, it can wear out, but practicaly? I think this RTC memory is there for usage with ULP coprocessor able to do something even in Deep Sleep mode.
If it helps somebody I will be pleased, if there are errors in my explanation I will be also pleased to be corrected.

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