Technologia GSM w elektronice (12)

Tryby oszczędzania energii

W tym artykule opiszemy tryby pracy, które są dostępne w programowalnych modułach Sierra Wireless™ AirPrime Q26 oraz korzyści wynikające z ich zastosowania.

Dodatkowe materiały na CD/FTP:
ftp://ep.com.pl, user: 10925, pass: 87thc181
• poprzednie części kursu

Różnorodność potencjalnych zastosowań, w których mogą być wykorzystane programowalne urządzenia Sierra Wireless, jest dość szeroka. Aby sprostać wymaganiom, jakie stawia się współczesnym układom programowalnym, szczególnie dotyczącemu optymalizacji zużycia energii, w programowalnych modułach Sierry Wireless mamy do dyspozycji kilka różnych trybów pracy. Każdy z nich cechuje się inną charakterystyką pracy, dostępnością interfejsów peryferyjnych oraz oczywiście poborem prądu.

ACTIVE mode

Podstawowym trybem pracy modułu jest tryb ACTIVE mode. W tym trybie wszystkie funkcje modułu są aktywne. Istnieją dwa warianty tego trybu: z włączonym i wyłączonym stosem GSM. Stos GSM to jeden z procesów w systemie operacyjnym modułu. Odpowiedzialny jest za obsługę części radiowej oraz za współpracę z siecią GSM. Wyłączenie stosu GSM pozwala na oszczędność energii, jednak wszystkie serwisy Open AT oraz komendy AT powiązane z obsługą sieci i obsługą SIM nie będą działały.

Aby wyłączyć stos GSM, należy wydać komendę AT+CFUN=0. Po tej komendzie nastąpi reset, moduł uruchomi się bez obsługi stosu GSM.

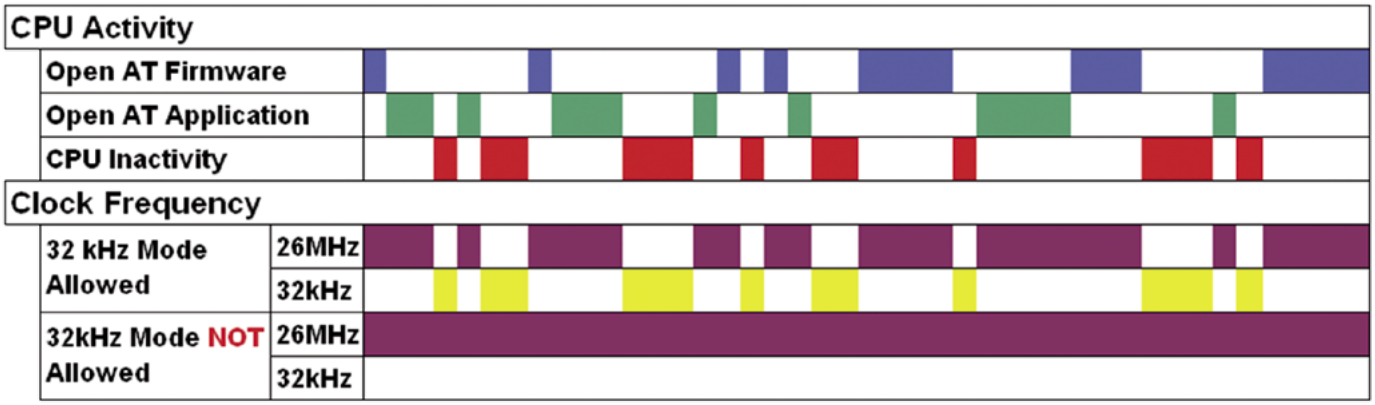
Aby ponownie włączyć stos GSM, wydajemy komendę AT+CFUN=1,1 (reset modułu) lub AT+CFUN=1,0

W trybie ACTIVE MODE procesor modułu jest taktowany domyślnie z prędkością 26 MHz, a jak pisaliśmy w jednym z wcześniejszych odcinków naszego cyklu, maksymalna prędkość taktowania procesora wynosi 104 MHz. Na **listingu 1** przedstawiamy, w jaki sposób kontrolować prędkość taktowania modułu.

Przytoczony przykład tworzy dwie komendy AT, których zadaniem jest przełączenie prędkości taktowania zgodnie ze swoją nazwą, a następnie wywołanie prostej pętli opóźniającej. Po jej wykonaniu wyświetlany jest komunikat OK. Obserwując czas wykonania pętli opóźniającej, możemy porównać moc obliczeniową dla obu częstotliwości taktowania. Według informacji producenta moduł pracujący z częstotliwością 26 MHz dysponuje mocą obliczeniową na poziomie 21 MIPS-ów, natomiast jeśli pracuje z częstotliwością 104 MHz, to moc obliczeniowa równa jest 87 MIPS-ów. Oczywiście wyższa

Listing 1. Kontrola częstotliwości taktowania

```
#include "adl_global.h"
#include "generated.h"

s32 VariHandle ;
//------------------------------------------------------------------------
void Empty (u32 Data) {

        static u8 a =0;
        a++;
}
//------------------------------------------------------------------------
void Fun_slow (adl_atCmdPreParser_t * paras) {

    u16 Cnt, Cnt1 ;

    TRACE((2,"Start Normal mode"));
    adl_vsSetClockMode (VariHandle, ADL_VS_MODE_STANDARD); //26MHz

    Cnt = 0xffff;           //opoznienie
    while (Cnt--) {
        Cnt1 = 50 ;
        while (Cnt1--) {
            Empty (Cnt) ;
        }
    }
    adl_atSendStdResponse(ADL_AT_RSP,ADL_STR_OK); //wyswietl OK
}

//------------------------------------------------------------------------
void Fun_fast(adl_atCmdPreParser_t * paras) {

    u16 Cnt, Cnt1 ;

    TRACE((2, "Start in VariSpeed mode")) ;
    adl_vsSetClockMode (VariHandle, ADL_VS_MODE_BOOST) ; //104MHz

    Cnt = 0xffff ;              ///opoznienie
    while (Cnt--) {
        Cnt1 = 50 ;
        while (Cnt1--) {
            Empty (Cnt ) ;
        }
    }

    adl_atSendStdResponse(ADL_AT_RSP,ADL_STR_OK); //wyswietl OK
}
//------------------------------------------------------------------------
void main_task ( void )
{
    // TODO Insert your task initialization code here
        VariHandle = adl_vsSubscribe () ;
    TRACE((1,"VariHandle = %d", VariHandle));

    adl_atCmdSubscribe("AT+FAST", Fun_fast, ADL_CMD_TYPE_ACT);
    adl_atCmdSubscribe("AT+SLOW", Fun_slow, ADL_CMD_TYPE_ACT);
}
```



Rysunek 1. Schemat zmiany częstotliwości taktowania w funkcji zmian obciążenia procesora

prędkość taktowania oznacza większe zużycie prądu przez moduł.

SLEEP mode

Kolejnym, bardzo pożytecznym trybem jest tryb SLEEP mode. W trybie tym wewnętrzny procesor jest w trybie obniżonego poboru energii i jest taktowany częstotliwością 32 kHz. Dostęp do interfejsów zewnętrznych jest ograniczony, w związku z czym interfejsy UART, USB, SPI, I²C, GPIO, ADC oraz Buzzer nie są dostępne. Przetwarzanie aplikacji Open AT jest wstrzymane. Wznowienie jej działania następuje w przypadku zaistnienia poniższych zdarzeń:

- pojawienie się alarmu RTC,
- zdarzenie od sprzętowego Timera TCU,
- pojawienie się zewnętrznego przerwania na pinie INT,
- naciśnięcie klawisza z matrycy 5×5,
- zmiana stanu linii DTR portu UART1 na niski.

Po wykonaniu zadań związanych ze zdarzeniem, które było powodem wybudzenia, moduł ponownie przełącza się w tryb SLEEP. W okresie wybudzenia funkcjonalność modułu jest zgodna z trybem ACTIVE mode i wszystkie interfejsy znowu są dostępne. Schemat zmian częstotliwości najlepiej zobrazuje **rysunek 1**.

Wyjście ze stanu obniżonego poboru energii następuje również regularnie, zgodnie z pagingiem sieci GSM (oczywiście o ile stos GSM nie został wyłączony). Wygląda to tak, że co określony czas (zależny od sieci GSM, z reguły co 2 s lub 0,5 s) moduł budzi się na bardzo krótko, żeby posłuchać sieci GSM, a następnie ponownie przełącza się w tryb obniżonego poboru energii. Dzięki temu możliwa jest reakcja modułu na przychodzące połączenie oraz możliwe jest odbieranie SMS-ów.

Wprowadzenie modułu w tryb SLEEP mode uzyskujemy, wydając komendę AT+W32K=1 lub AT+W32K=1,0 (ignorowany jest stan linii DTR). Moduł po wydaniu komendy potrzebuje od 1 s do 15 s czasu na przejście do trybu SLEEP. Trwałe opuszczenie trybu SLEEP mode następuje po wydaniu komendy AT+W32K=0.

Wykorzystanie linii DTR jako jednego z warunków wejścia w tryb SLEEP mode jest podyktowane tym, że w niektórych zastosowaniach moduł pracuje jedynie jako zwykły modem odpowiadający na komendy AT (bez aplikacji Open AT), wysyłane przez system nadrzędny(Host). W takim przypadku, gdy Host ma jakieś komendy do wydania do modułu, blokuje on możliwość przejścia w tryb SLEEP mode, ustawiając linie DTR w stan wysoki. Dzięki temu UART nie zostanie wyłączony w trakcie przesyłania informacji. Po przesłaniu całej informacji Host zmienia stan na linii DTR, pozwalając modułowi ponownie wejść w tryb SLEEP mode. W przypadku wywołania trybu SLEEP mode ignorującego linię DTR, wyjście z trybu SLEEP kontrolowane jest przez aplikację Open AT (np. po doliczeniu timera lub po wystąpieniu zdarzenie GSM).

Tak samo jak w przypadku ACTIVE mode, tu też istnieje możliwość wyboru trybu SLEEP z wyłączoną obsługą stosu GSM. W tym celu wydajemy komendę AT+WBHV=1,2. Opuszczenie trybu SLEEP w takim przypadku realizowane jest przez komendę AT+WBHV=1,0.

Na koniec jedna uwaga dotycząca trybu SLEEP mode – w trybie tym zostaje dezaktywowany wewnętrzny, hardware'owy wachdog systemowy. W warunkach normalnych przerywa on działanie każdej pętli trwającej dłużej niż 5 s, co zabezpiecza nasz system przed wystąpieniem tzw. martwych pętli. Zatem przed uruchomieniem trybu SLEEP mode musimy zadbać o takie warunki aplikacji, aby żadna tego typu sytuacja nie zaistniała.

Alarm mode

Tryb ALARM mode oparty jest na zegarze RTC modułu i polega na wyłączaniu modemu i automatycznym jego włączaniu się w momencie wystąpienia wcześniej zaprogramowanego alarmu. Zasilanie RTC jest realizowane za pomocą osobnego pinu BAT-RTC, dzięki czemu kiedy modem jest wyłączony, pobór energii będzie ograniczony do prądu pobieranego przez pin RTC (2-3uA). W przypadku modemu Fastrack zasilanie RTC zrealizowane jest wewnątrz modemu w postaci baterii pastylkowej, natomiast w przypadku projektów opartych na module Q26, o zasilanie dla RTC musimy zadbać samodzielnie.

Ustawienie wewnętrznego zegara RTC najczęściej realizowane jest komendą AT+CCLK="yy/mm/dd,hh:mm:ss", ale może być też wykonane przez aplikację Open AT, wykorzystując funkcję ADL-a o nazwie *adl_rtcSetTime()*. Jeżeli w układzie zostało zrealizowane podtrzymanie RTC, to zegar będzie pracował niezależnie od tego, czy modem jest włączony, czy nie. System RTC ma dodatkowo możliwość ustawienia do 16 alarmów. Alarm ustawiamy komendą AT+CALA="yy/mm/dd,hh:mm:ss" (na razie nie mam możliwości ustawienia go za pomocą funkcji ADL). Po ustawieniu zegara oraz alarmu można wyłączyć moduł komendą AT+CPOF. Wydając komendę AT+CPOF, należy się upewnić, że pin ON/OFF jest w stanie niskim. Stan wysoki na tym pinie będzie powstrzymywał moduł przed wyłączeniem. W momencie nadejścia czasu alarmu moduł zostaje automatycznie włączony. Przykładowe zastosowanie przedstawionych możliwości zostało wykorzystane w **listingu 2**.

Działanie przedstawionej aplikacji polega na ustawieniu alarmu za pięć minut od odczytanego czasu RTC. Jednocześnie uruchomione zostają dwa timery: jeden wykonujący cyklicznie wyświetlanie tekstu (może to być oczywiście każda inna funkcja), a drugi to timer, który spowoduje wyłączenie się modułu. Po pięciu minutach, czyli kiedy uruchomi się alarm, moduł ponownie się włączy, wykona te same zadania i sam się wyłączy.

W aplikacji zastała również zamaskowana systemowa komenda AT+CCLK. Własna interpretacja tej komendy ma na celu ponowne ustawienie alarmu, w momencie kiedy czas RTC zostanie przedstawiony za pomocą tej komendy. Warto zauważyć, że funkcja *ustaw_alarm()* najpierw kasuje poprzedni

Listing 2. Przykład użycia funkcji opisanych w artykule

```
#include "adl_global.h"
#include "generated.h"

/*************************************************************************/
/*  Local variables                                                      */
/*************************************************************************/
adl_rtcTime_t time;
adl_rtcTimeStamp_t timestamp;
bool set_alarm;
/*************************************************************************/
/*  Local functions                                                      */
/*************************************************************************/
bool Resp_Handler(adl_atResponse_t * strc);

void ustaw_alarm(){

         ascii komenda[40];
         ascii data[20];
         TRACE((1,"ustaw alarm"));

         adl_rtcGetTime(&time);
         adl_rtcConvertTime(&time,&timestamp,ADL_RTC_CONVERT_TO_TIMESTAMP);
         TRACE((1,"TimeStamp = %d",timestamp.TimeStamp));
         timestamp.TimeStamp+=5*60;                  //dodaj 5min
         TRACE((1,"TimeStamp = %d",timestamp.TimeStamp));
         adl_rtcConvertTime(&time,&timestamp,ADL_RTC_CONVERT_FROM_TIMESTAMP);
         TRACE((1,"DATA : = %d/%d/%d",time.Year-2000,time.Month,time.Day));
         //+CCLK: „11/04/04,11:46:07"

    wm_sprintf(komenda,"AT+CALA=\"\",1;+CALA=\"");
    wm_sprintf(data,"%02d/%02d/%02d,%02d:%02d",time.Year-2000,time.Month,time.Day,time.Hour,time.Minute);
    wm_strcat(komenda,data);
    wm_strcat(komenda,"\"");
    TRACE((1,komenda));
    set_alarm=TRUE;
    adl_atCmdCreate ( komenda , ADL_AT_PORT_TYPE( ADL_PORT_OPEN_AT_VIRTUAL_BASE, FALSE) , Resp_Handler , „*" , NULL);

}
//--------------------------------------------------------------------------------------------
bool Resp_Handler(adl_atResponse_t * strc){

     TRACE (( 1, „Embedded : Response Handler %d",strc->RspID ));
    if(strc->RspID==ADL_STR_OK) {

        if(set_alarm)  //wywolanie AT+CALA
         set_alarm=FALSE;
        else     //wywolanie AT+CCLK
        {
         adl_atSendStdResponse(ADL_AT_RSP,ADL_STR_OK);
         ustaw_alarm();
        }
    }
    else {
         adl_atSendStdResponse(ADL_AT_RSP,ADL_STR_ERROR);
    }

  return FALSE;
}
//--------------------------------------------------------------------------------------------
void Fun_cclk(adl_atCmdPreParser_t * param) {

         ascii * parametr;
         ascii komenda[40];

    TRACE((1,"W Fun_cclk"));

    if (param->Type == ADL_CMD_TYPE_PARA){
         parametr = ADL_GET_PARAM ( param, 0 );
        wm_sprintf(komenda,"AT+CCLK=\"");
        wm_strcat(komenda,parametr);
        wm_strcat(komenda,"\"");
        TRACE((2,komenda));
        adl_atCmdCreate ( komenda , ADL_AT_PORT_TYPE( ADL_PORT_OPEN_AT_VIRTUAL_BASE, FALSE) , Resp_Handler , „*" ,
NULL);
    }
}
//--------------------------------------------------------------------------------------------

void Hello_timer ( u8 ID, void * Context ){

         adl_atSendResponse ( ADL_AT_UNS, „Hello from timer\r\n");

}

void Sleep_timer ( u8 ID, void * Context ){

         adl_atCmdSend ( „AT+CPOF" , NULL, NULL , NULL);
}

void main_task ( void )
{
    // TODO Insert your task initialization code here

    TRACE (( 1, „Embedded Application : Main" ));

    adl_atCmdSubscribe(„at+cclk", (adl_atCmdHandler_t *) Fun_cclk, ADL_CMD_TYPE_PARA | 0x0011);
    adl_tmrSubscribe( TRUE, 10, ADL_TMR_TYPE_100MS, Hello_timer );
    adl_tmrSubscribe( FALSE, 50, ADL_TMR_TYPE_100MS, Sleep_timer );
    ustaw_alarm();
}
```

Tabela 1. Dostępne funkcje w zależności od trybu pracy

Funkcja	Tryb aktywny (Active)	Tryb czuwania (Sleep)	Tryb alarmowy (Alarm)	Tryb wyłączenia (Off)
Alarm	√	√	√	–
Uruchomienie aplikacji OpenAT po wystąpieniu zdarzenia	√	√	–	–
Żądanie obsługi przez sieć na skutek wywołania, odbioru komunikatu SMS lub transmisji danych GPRS	Tak, jeśli interfejs GSM jest aktywny	Tak, jeśli interfejs GSM jest aktywny	–	–
SIM	√	–	–	–
UART	√	–	–	–
USB	√	–	–	–
SPI	√	–	–	–
I^2C	√	–	–	–
GPIO	√	–	–	–
ADC	√	–	–	–
Buzzer	√	–	–	–
Klawiatura	√	√	–	–
Przerwanie zewnętrzne	√	√	–	–
Mrugająca dioda LED	√	√	–	–

alarm o indeksie 1(komenda AT+CALA="",1), a dopiero potem ustawia właściwy alarm. Dzięki takiemu rozwiązaniu, w momencie przestawiania czasu, alarm zawsze jest ustawiany za pięć minut oraz pojawia się nam kilka różnych, niepotrzebnych alarmów.

Na koniec, w celu podsumowania informacji, jakie zostały omówione w tym odcinku, przedstawiam tabelkę, która zawiera informacje na temat dostępności interfejsów zewnętrznych w każdym z trybów (**tabela 1**).

Opisując tryby pracy, celowo unikałem podawania wartości prądów dla poszczególnych trybów. Wartości te są różne dla różnych urządzeń, więc inna byłaby wartość poboru prądu dla modułu Q2687 (2G), a inna dla modułu Q26Extreme (3G) pracującego w tym samym trybie. Zupełnie inne byłyby też wartość pobieranego prądu dla modemu Fastrack Xtend. Te wartości są dostępne w dokumentacji hardware'owej (Product Technical Specification) każdego z tych urządzeń.

Więcej informacji na temat produktów Sierra Wireless można znaleźć na stronach producenta: *www.sierrawireless.com* lub kontaktując się z firmą ACTE Sp. z o.o., która jest oficjalnym dystrybutorem opisywanych produktów oraz zapewnia pełne wsparcie techniczne.

Adrian Chrzanowski
Acte Sp. z o.o.