MSP430
Kontrolery USART, FLASH i organizacja pamięci wewnętrznej

Kontynuujemy cykl artykułów na temat procesora MSP430 firmy Texas Instruments. Tytułowe zagadnienia, będą omówione w kontekście serii x1xx mikrokontrolerów MSP430. Uzupełnieniem treści artykułu są programy umieszczone na płycie CD. Wszystkie przykłady programów i konfiguracje sprzętowe wykonano w oparciu o dostępną w ofercie AVT płytkę ewaluacyjną eMeSPek (AVT–MSP430).

USART

Mniejsze mikrokontrolery MSP430 z serii 1xx (np.: x1121A,x1122) nie posiadają wbudowanego kontrolera interfejsów szeregowych. Dlatego też, gdy zachodzi potrzeba komunikowania się mikrokontrolera z otoczeniem, to projektant zmuszony jest samodzielnie zaimplementować interfejs komunikacyjny. Wydłuża to czas realizacji projektu, rosną koszty związane z implementacją i testowaniem. Rozwiązaniem alternatywnym jest zastosowanie mikrokontrolera mającego wbudowany sprzętowy kontroler interfejsów szeregowych USART. Spośród mikrokontrolerów serii x1xx projektant może wybrać układ mający jeden lub dwa kontrolery USART (np.: x1232, x149, x1611). Każdy USART to możliwość wyboru jednego z interfejsów komunikacyjnych: UART, SPI, bądź I^2C.

Mikrokontroler umieszczony na płytce eMeSPek (x1232) ma jeden kontroler USART. Dostępne interfejsy komunikacyjne to UART oraz SPI. Zmieniając nastawy bitu SYNC (rejestr U0CTL) dokonujemy wyboru interfejsu komunikacyjnego. Gdy, bit SYNC ma wartość „1", to aktywny jest UART, w przeciwnym wypadku SPI.

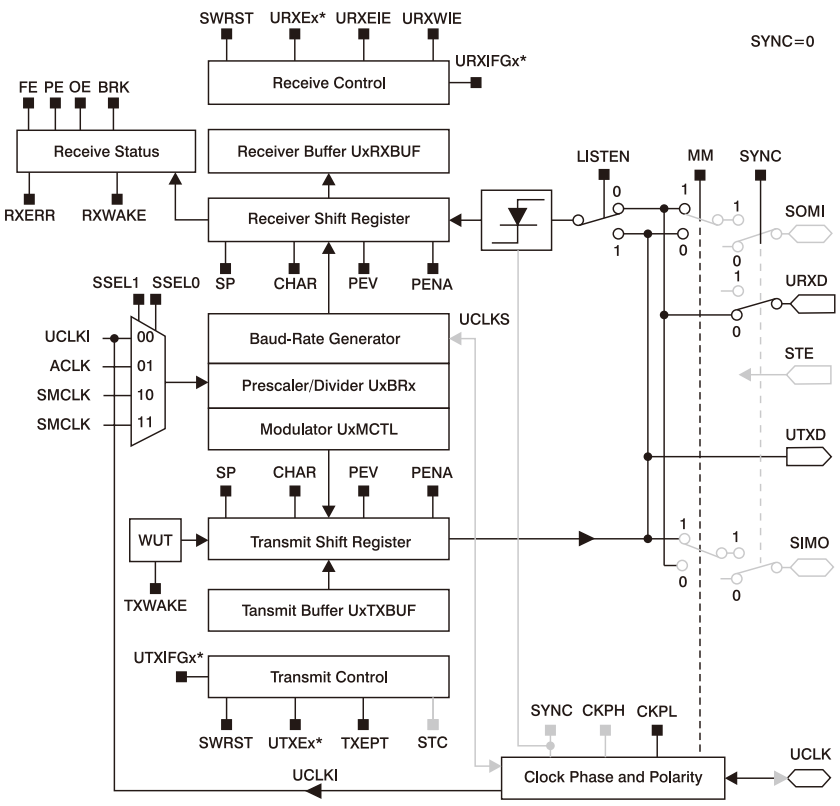
UART (SYNC=0)

Jak pokazano na schemacie blokowym (**rys. 1**) UART posiada niezależne bufory nadawczo-odbiorcze, oraz rejestry przesuwne danych przychodzących i wychodzących. Źródłem sygnału zegarowego sterującego transmisją (UCLK) może być jeden z wewnętrznych sygnałów zegarowych ACLK, SMCLK, albo zewnętrzny sygnał UCLKI doprowadzony do mikrokontrolera. O wyborze źródła decydują nastawy bitów SSEL1, SSEL0 z rejestru UxTCTL. Modyfikując rejestr UxCTL można określić format ramki danych (**rys. 2**). Dostępne nastawy to 7 lub 8 bitów danych (CHAR), kontrola parzystości (PENA), adresacja ramki (MM), jeden bądź dwa bity stopu (SP). Kompletna ramka rozpoczyna się bitem startu, a dane przesyłane są od bitu najmniej znaczącego.

Nastawa prędkości transmisji wykonywana jest przy pomocy rejestrów UBR1x, UBR0x, UxMCTL. Do rejestrów UBR1x, UBR0x wprowadzany jest wynik operacji dzielenia częstotliwości UCLK (Hz) przez prędkość transmisji (bps), natomiast w rejestrze UxMCTL umieszczany jest współczynnik modulacji sygnału UCLK. Na płycie CD zapisano program `BaudrateCalculator.exe` obliczający nastawy rejestrów konfiguracyjnych UBR1x, UBR0x, UxMCTL. Maksymalna prędkość transmisji to 1/3 częstotliwości UCLK. Poza tym jednym ograniczeniem programista posiada pełną swobodę przy jej wyborze.

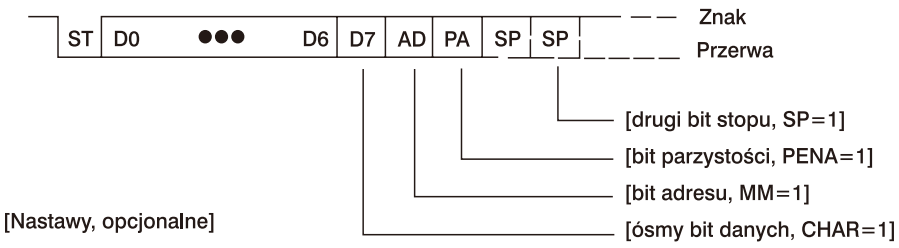
Konfigurując interfejs komunikacyjny UART należy włączyć moduł odbiorczy, nadawczy, bądź oba modułu jednocześnie (UTXEx, URXEx, rejestru MEx). Nie wolno zapominać również o aktywowaniu linii transmisyjnych (UTXDx, URXDx) mikrokontrolera. W tym celu należy wyprowadzenia P3.4, P3.5 (UART0), P3.6, P3.7 (UART1) skonfigurować jako dedykowane funkcjom UART. W określeniu konfiguracji UART pomogą kroki opisane w **tab. 1**.

Wśród materiałów będących uzupełnieniem treści artykułu, umieszczono program `UART.eww`. Jego uruchomienie wymaga zaprogramowania płyty ewaluacyjnej eMeSPek i podłączenia niej komputera z uruchomionym terminalem portu szeregowego. Komunikacja z układem odbywa się przy pomocy komend tekstowych.



**Refer to the device-specific datasheet for SFR locations*

Rys. 1. Schemat blokowy kontrolera USART – tryb UART



Rys. 2. Ramka danych interfejsu UART

Tab. 1. Konfiguracja kontrolera USART (UART, SPI)

1. Wprowadzenie kontrolera USART w tryb restartu (ustawienie bitu SWRST z rejestru UxCTL na wartość „1”).
2. Modyfikacja rejestrów konfiguracyjnych (interfejs komunikacyjny UART, albo SPI).
3. Włączenie modułów odbiorczego, nadawczego, bądź obu jednocześnie (bity URXEx, UTXEx – UART, USPIEx – SPI, rejestru MEx).
4. Opuszczenie trybu restartu (ustawienie bitu SWRST z rejestru UxCTL na wartość „0”).
5. Ewentualne aktywowanie obsługi przerwań, od danych przychodzących, wychodzących (bity URXIEx, UTXIEx, rejestru IEx – UART, SPI).

Każda komenda musi być zakończona znakiem CR. Dostępne komendy to *SetD1*, oraz *ClrD1*. Efektem ich realizacji jest zmiana stanu portu P1.3, co w przypadku płyty eMeSPek przekłada się na włączenie, bądź wyłączenie diody D1.

Kontroler USART ma dwa oddzielne wektory przerwań dla danych odebranych oraz wysyłanych (**list. 1**). Flaga UTXIFGx ustawiana jest po restarcie PUC, w wyniku restartu USART (SWRST=1) oraz gdy moduł transmisyjny jest gotowy do wysłania danych (pusty bufor nadawczy UxTXBUF). Jeśli w mikrokontroler ma ustawiony bit GIE w rejestrze SFR i aktywne przerwania od modułu nadawczego (bit UTXIEx w rejestrze IEx), to ustawienie flagi UTXIFGx skutkuje wywołaniem procedury obsługi przerwania USART0TX, lub ewentualnie USART1TX). Wówczas flaga UTXIFGx jest automatycznie zerowana.

Automatyczne zerowanie flagi UTXIFGx ma również miejsce w momencie wpisu danych do bufora nadawczego UxTXBUF. Dodatkowo, w rejestrze UxTCTL jest bit TXEPT, który przyjmuje wartość „1” w momencie opróżnienia bufora nadawczego. Monitorując stan tego bitu użytkownik ma możliwość kontrolowania zajętości bufora transmisyjnego.

Ustawienie flagi URXIFGx informuje o odebraniu danych i załadowaniu ich do bufora odbiorczego UxRXBUF. Jeśli w mikrokontrolerze zostały aktywowane globalne przerwania maskowane oraz przerwania od modułu odbiorczego (bit URXIEx, rejestru IEx), to wywoływana jest procedura obsługi przerwania USART0RX, ewentualnie USART1RX. Flaga URXIFGx zerowana jest automatycznie w momencie rozpoczęcia wykonywania procedury obsługi przerwania oraz podczas odczytu danych z buforu odbiorczego UxRXBUF. Wyjątkiem jest opisywany dalej tryb wykrywanie zbocza transmisji.

Dotychczas opisana funkcjonalność wektorów przerwań odnosi się zarówno do interfejsu UART, jak również SPI. UART wzbogacono jednak o pewną, dodatkową funkcjonalność. Możliwa jest taka konfiguracja układu, aby do bufora odbiorczego UxRXBUF nie trafiały przychodzące dane, a flaga URXIFGx nie była ustawiana. Tryby pracy, o których właśnie mowa to: tryb wykrywania błędów transmisji (URXEIE=0), oraz tryb adresacji danych (URXWIE=1).

Interfejs komunikacyjny UART w MSP430 można wykorzystać do komunikacji typu punkt-punkt lub w pomiędzy wieloma procesorami. W tym drugim trybie pakiety danych opatrzone są adresem. Transmisja z adresacją wspiera dwa tryby pracy: *idle-line* (MM=0) oraz adresowy (MM=1).

Oba tryby mają identyczny mechanizm odbioru, natomiast różnią się sposobem adresowania wysyłanych danych. Podczas odbioru danych, gdy odebrany zostanie adres, to automatycznie ustawiany jest bit RXWAKE (rejestr UxRCTL). Ciężar weryfikacji adresu i ewentualnego odbioru danych spoczywa na użytkowniku. Dane odbiera się w „tradycyjny” sposób, natomiast odrzucenie transmisji realizowane jest przy pomocy modyfikacji wartości bitu URXWIE (rejestr UxRCTL). Jeśli ma on wartość „1”, to wszystkie dane nie będące adresem są przez UART ignorowane.

W trybie *idle-line* (MM=0), koniec pakietu danych wykrywany jest, gdy po bicie stopu pojawi się co najmniej 10 bitów o wysokim stanie logicznym, bądź w przypadku ramki z dwoma bitami stopu, po wystąpieniu pierwszego z nich. Adres pakietu danych jest pierwszą daną wysyłaną po wykryciu końca pakietu. Adresacja pakietu odbywa się dwuetapowo. Najpierw należy ustawić bit TXWAKE (rejestry UxTCTL), a do bufora nadawczego wpisać dowolna wartość. Następnie należy do bufora nadawczego wpisać adres pakietu danych. Dane wpisane do bufora w pierwszym kroku, tak naprawdę nie są wysyłane i użytkownik nie powinien się nimi przejmować.

W trybie adresowym (MM=1), adresacja pakietów danych odbywa się przy pomocy dodatkowego bitu, AD, umieszczanego w ramce danych. Gdy, bit ma wartość „1”, to dane przesyłane w ramce są adresem pakietu. Adresacja ramki odbywa się przy pomocy bitu TXWAKE. Wartość bitu jest przepisywana do bitu adresu AD, po czym jest automatycznie czyszczona. Chcąc ustawić adres pakietu, należy ustawić bit TXWAKE, a następnie do bufora nadawczego wpisać adres.

List.1. Kontroler USART -szablony procedur obsługi przerwań

```
// wektor USARTxTX danych wychodzących
// UART, SPI  (x - id kontrolera - 0,1)
#pragma vector=USARTxTX_VECTOR
__interrupt void usartx_tx (void)
{
   …
}

// wektor USARTxRX danych przychodzących
// UART, SPI  (x - id kontrolera - 0,1)

#pragma vector=USARTxRX_VECTOR
__interrupt void usartx_rx (void)
{
   …
}
```

Tryb pracy idle-line bez adresowania pakietów, przeważnie jest stosowany do komunikowania się pomiędzy dwoma urządzeniami np.: mikrokontroler – komputer PC.

Interfejs komunikacyjny UART wyposażono w mechanizm wykrywania początku odbioru danych. Aby móc wykorzystywać ten mechanizm, należy aktywować globalne przerwania maskowane (GIE), przerwania od modułu odbiorczego (URXIEx), oraz ustawić bit URXSE rejestru UxTCTL. Wtedy, wraz z wykryciem początku odbioru danych, ustawiany jest wewnętrzny sygnał URXS. Wymusza to wywołanie procedury obsługi przerwania. Flaga URXIFGx nie jest jednak ustawiana i w ten sposób istnieje możliwość rozróżnienia, czy odebrano dane (URXIFGx=1), czy procedura została wywołana dzięki wykryciu początku transmisji (URXIFGx=0). W tym przypadku programista powinien wyzerować oraz ponownie ustawić bit URXSE. Spowoduje to wyzerowanie wewnętrznego sygnału URXS i reaktywuje wykrywanie początku odbioru danych.

Omawiany mechanizm stosowany jest w aplikacjach korzystających z energooszczędnych trybów pracy LPMx. W sytuacji, gdy w trybie uśpienia sygnał taktujący transmisję (UCLK) jest wyłączony, po wykryciu początku odbioru danych, programista powinien wyjść z trybu LPMx do aktywnego AM, bądź do trybu LPMx, w którym UCLK jest aktywny. W przeciwnym wypadku przychodząca transmisja zostanie utracona.

Na płycie CD można znaleźć program `UART_URXSE.eww` będący zmodyfikowaną wersją `UART.eww`. Modyfikacja polega między innymi na zmianie źródła sygnału UCLK, z ACLK na SMCLK. Źródło, sygnału taktującego SMCLK jest konfigurowane w ciele programu. Dostępne nastawy źródła SMCLK to LFXT1CLK=32768 Hz i DCO=720 kHz. Oprogramowanie oczekuje na dane przychodzące w trybie uśpienia LPM3. W tym trybie sygnał zegarowy SMCLK jest nieaktywny!

Kontroler UART wyposażono w mechanizm wykrywania błędów odbioru danych. Wykrywane są nieprawidłowości w strukturze ramki, błędy parzystości, przepełnienia oraz przerwanie odbioru danych (**tab. 2**). W momencie, wystąpienia któregoś z wyżej wymienionych błędów, w rejestrze UxRCTL ustawiany jest bit opisujący

błąd. Dodatkowo, ustawiany jest bit RXERR informujący użytkownika o wystąpieniu któregoś z błędów odbioru danych. Można wówczas wyzerować bity błędów oraz bit RXERR. Można również wykorzystać fakt, że są one zerowane automatycznie w momencie odczytu rejestru UxRXBUFF. Jeśli w mikrokontrolerze bit URXIEI ma wartość „1", to dane obarczone błędem odbierane są przez kontroler (UxRXBUF). Dodatkowo błąd przerwania transmisji BRK powoduje ustawienie flagi URXIFGx i może spowodować uruchomienie procedury obsługi przerwania. W praktyce, zazwyczaj bit URXIEI jest zerowany, a dane zabezpiecza się sumą kontrolną. Wówczas dane obarczone błędem są pomijane, a CPU nie jest zaangażowany w ich odbiór.

Tab. 2. UART – błędy transmisji danych przychodzących

Typ błędu.	Opis błędu.
FE (błąd ramki)	Błąd ramki jest wykrywany, gdy bit stopu jest w stanie niskim. Jeśli ramka posiada dwa bity stopu (SP), to wyłącznie pierwszy z nich jest sprawdzany pod kątem błędu ramki.
PE (błąd parzystości)	Parzystość ramki jest obliczana dla danych z ramki, oraz dla bitu adresu – jeśli został on użyty (MM). Błąd parzystości jest wykrywany, gdy programista aktywuje sprawdzanie parzystości ramki (PENA), oraz wystąpi przekłamanie monitorowanego ciągu bitów.
OE (błąd przepełnienia)	Błąd przepełnienia bufora odbiorczego jest zgłaszany, w momencie, gdy w buforze odbiorczym UxRXBUF zostaną nadpisanie dane. Jeśli programista nie odczyta z bufora odebranego znaku, następnie do bufora trafi nowy znak, który nadpisze poprzednią daną, wówczas zostanie wykryty błąd przepełnienia.
BRK (błąd przerwania odbioru danych)	W sytuacji, gdy nie zostanie odebrany bit stopu, a następnie na linii URXDx zostanie odebranych, co najmniej 10 bitów o niskim poziomie logicznym, zostanie zgłoszony błąd przerwania odbioru danych. Błąd ten jest powiązany z błędem FE, oraz jako jedyny z grupy opisywanych błędów powoduje ustawienie flagi obsługi przerwania URXIFGx.

SPI (SYNC=1, I²C=0)

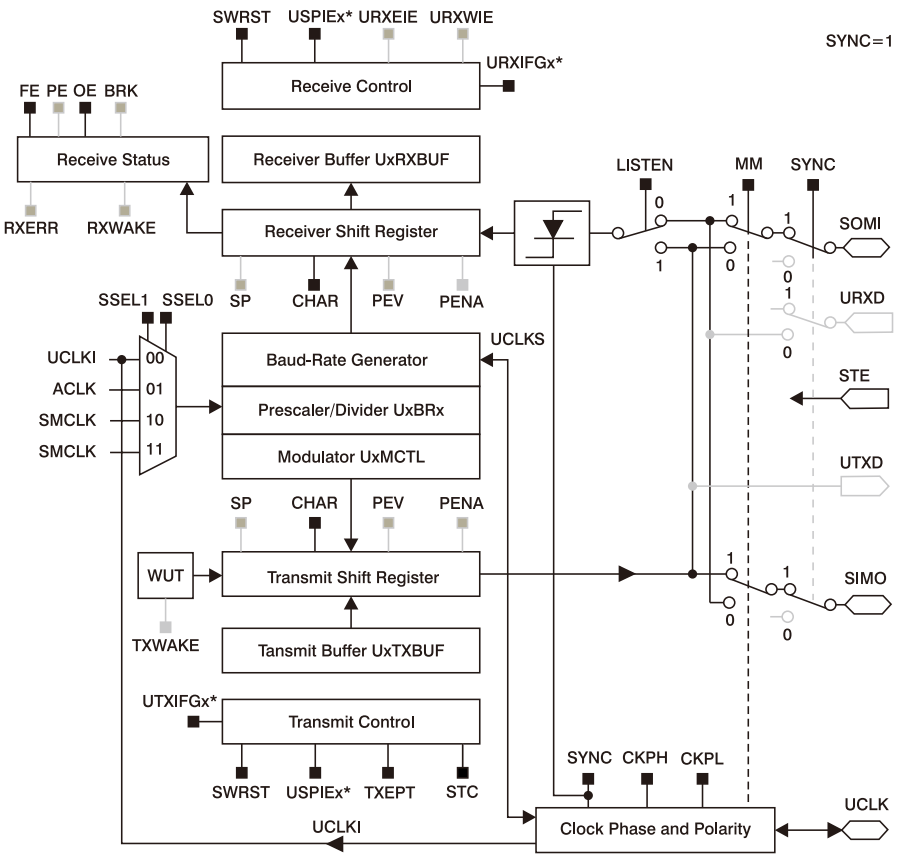
Aby przełączyć USART w tryb pracy jako SPI, należy ustawić bit SYNC (rejestr UxCTL). Dodatkowo, gdy kontroler obsługuje interfejs I²C (mikrokontrolery x15x, x16x), to aktywacja trybu SPI wymaga, wyzerowania bitu I²C w rejestrze U0CTL.

Kontroler SPI ma niezależne bufory do nadawania i odbioru oraz rejestry przesuwne danych przychodzących i wychodzących (**rys. 3**). Procedura konfiguracji interfejsu przebiega analogicznie, jak w przypadku interfejsu UART (tab. 1). Identyczny jest również mechanizm obsługi wektorów przerwań.

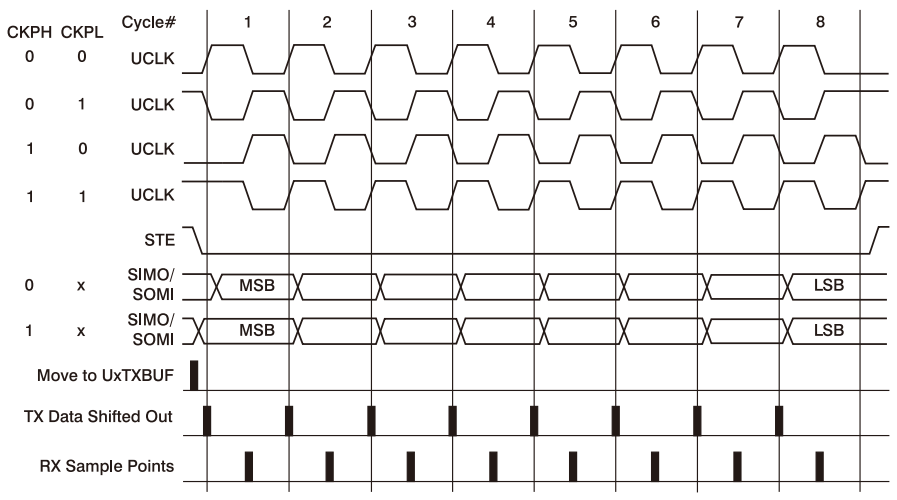
Transmisja danych odbywa się przy wykorzystaniu linii sterujących SIMO, SOMI, UCLK, STE. W zależności od trybu pracy SPI, wykorzystywane są 3 lub 4 linie sterujące (bit STC rejestru UxTCTL). W trybie 3-przewodowym (STC=1), aktywne są linie danych SIMO, SOMI, oraz linia sygnału zegarowego UCLK taktującego transmisję. Tryb 4-przewodowy (STC=0) charakteryzuje się dodatkowym sygnałem STE, służącym do synchronizacji transmisji oraz wykrywanie konfliktów. Aktywacja linii transmisyjnych, odbywa się po przez nadanie pinom mikrokontrolera P3.0…P3.3 (SPI0), P5.0…P5.3 (SPI1) funkcjonalności doprowadzeń SPI.

W przypadku interfejsu SPI zaimplementowanego w MSP430, dane transmitowane są w ramkach 7 (CHAR=0) bądź 8-bitowych (CHAR=1), począwszy od bitu najbardziej znaczącego. Konfiguracji podlega również polaryzacja linii zegarowej w stanie spoczynku, oraz moment nasłuchu/zmiany stanu linii danych. Nastawy bitów CKPH, CKPL, umożliwiają uzyskanie czterech możliwych sytuacji próbkowania linii danych (**rys. 4**).

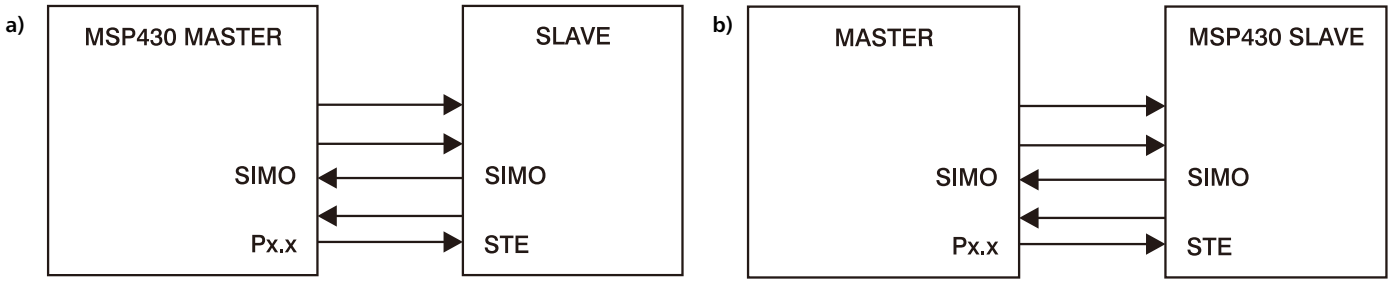
SPI może pracować w jednym z dwóch trybów pracy. Nadrzędnym – master (MM=1). Podrzędnym – slave (MM=0). W trybie nadrzędnym (**rys. 5a**), linie SIMO i SOMI, są odpowiednio wyjściem i wejściem transmitowanych danych. Sygnał taktujący transmisje UCLK, generowany jest przez kontroler SPI. Wyboru źródła sygnału (ACLK lub SMCLK) dokonuje się programowo, modyfikując nastawy bitów SSEL0, SSEL1. Użytkownik ma możliwość ustawienia prędkości transmisji. Zgodnie ze wzorem *UxBR=UCLK(Hz)/*



Rys. 3. Schemat blokowy kontrolera USART – tryb SPI



Rys. 4. Polaryzacja, przesuniecie sygnału zegarowego UCLK



Rys. 5. SPI pracujące w trybie Master (a) i Slave (b)

prędkość transmisji (bps) obliczane są wartości rejestrów UBR1x, UBR0x. Rejestr modulacji sygnału UxMCTL, nie jest wykorzystywany i powinien zostać wyzerowany. Maksymalna dostępna prędkość transmisji, to połowa częstotliwości UCLK. W trybie nadrzędnym (Master) układy odbiorczy oraz nadawczy pracują równocześnie, dlatego też chcąc odebrać dane należy wymusić transmisję danych.

W trybie podrzędnym (Slave, **rys. 5b**), linie SIMO i SOMI są odpowiednio wyjściem i wejściem transmitowanych danych. Źródłem sygnału taktującego transmisję jest zewnętrzny sygnał doprowadzony do linii UCLK. Układ nadrzędny taktuje transmisję, a nastawy bitów SSEL0, SSEL1 nie maja znaczenia. Prędkość transmisji jest równa częstotliwości sygnału taktującego transmisję.

Wśród materiałów uzupełniających treść artykułu umieszczono szablony obsługi interfejsu SPI w trybie master, oraz slave.

I²C (SYNC=1, I²C=1)

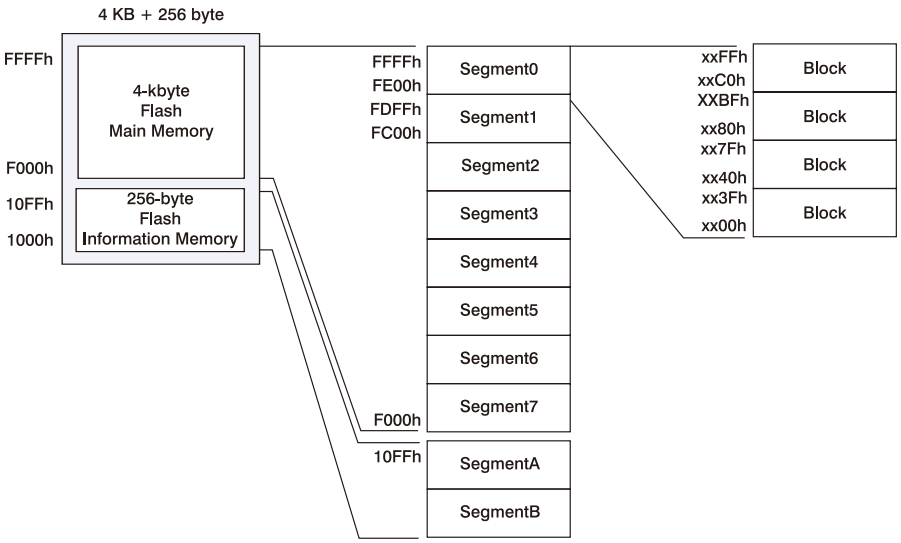
Mikrokontrolery serii x1xx mające dwa USART (x15x, x16x), mają też możliwość skonfigurowania pierwszego z nich w tryb pracy interfejsu I²C. W mikrokontrolerze z płyty eMeSPek – x1232, interfejs I²C nie jest dostępny.

Wewnętrzna pamięć mikrokontrolera

Mikrokontrolery MSP430 oparte są o architekturę von Neumanna i w związku z tym cała pamięć to pojedyncza, ciągła przestrzeń adresowa. W 64 kB przestrzeni adresowej odwzorowane są wektor SFR, urządzenia peryferyjne, pamięć RAM, pamięć FLASH/ROM oraz wektory przerwań (patrz CD – Map.pdf). Podstawowym słowem pamięci jest bajt, a dane o większej ilości bajtów zapisywane są w formie *little endian*.

Cała pamięć FLASH podzielona jest na segmenty. Rozróżniane są dwa rodzaje segmentów o długości 128 i 512 bajtów. Pierwsze mogą występować w liczbie jednego (segment B), bądź dwóch (segmenty A, B). W nazewnictwie określane są jako pamięć informacyjna (*Info*), a umiejscowione są od adresu 0x1000. Segmenty 512 bajtowe, określane jako pamięć programu, dostępne są od adresu 0xFFFF. Każdy z segmentów składa się z bloków 64-bajtowych (**rys. 6**). W zależności od tego, czy jest to pamięć informacyjna, czy pamięć programu, to segmenty są 2 lub 8-blokowe.

Cechą, która różni pamięć informacyjną od pamięci programu, jest maksymalna liczba cykli



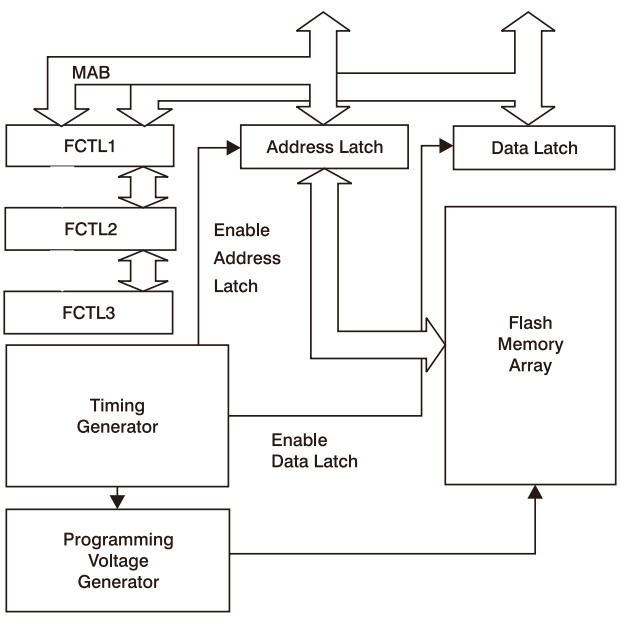
Rys. 6. Segmentaja pamieci FLASH (x1xx, 4kB + 256B)

kasowania. Pamięć informacyjna ma gwarantowane 100 tysięcy cykli kasowania, natomiast pamięć programu 10 tysięcy. W wyniku kasowania, pamięć zostaje wypełniona „1". Zapisując dane zawsze istnieje możliwość wyzerowania bitu, nigdy jego ustawienia. Aby uzyskać wymagany efekt należy skasować pamięć. Dlatego też w wielu aplikacjach zapis do pamięci poprzedzany jest jej kasowaniem.

Kasując i zapisując wewnętrzną pamięć FLASH mikrokontrolera należy mieć na uwadze minimalne wymagania związane z napieciem zasilania układu. W przypadku serii x1xx nie może być ono mniejsze, niż 2,7 V.

Kontroler pamięci FLASH

W strukturę mikrokontrolerów MSP430 wbudowano wewnętrzny kontroler pamięci (**rys. 7**) używany podczas zapisu i kasowania. Dane mogą być zapisywane jako pojedyncze bajty, słowa i bloki. Kasowanie danych obywa się segmentami. Możliwe jest także skasowanie całego obszaru pamięci programu, oraz dodatkowo pamięci informacyjnej.



Rys. 7. Schemat blokowy kontrolera pamieci FLASH

Konfiguracja kontrolera odbywa się przy pomocy rejestrów FCTL1, FCTL2, FCTL3. Dostęp do rejestrów chroniony jest hasłem (0xA5 – odczyt, 0x96 – zapis). Hasło musi być wpisywane do bardziej znaczącego bajtu rejestru. W przypadku dostępu do rejestru bez hasła, bądź też z niepoprawnym hasłem, zostanie ustawiona flaga KEYV oraz wykonany będzie restart PUC mikrokontrolera. Druga z flag, ACCVIFG, ustawiana jest w momencie naruszenia zasad dostępu do pamięci FLASH. Sytuacja taka ma miejsce, gdy programista rozpocznie zapis do rejestru FCTL1, FCTL2, a kontroler zajęty będzie wykonywaniem operacji kasowania bądź zapisu (bit BUSY=1).

Przed zapisem do rejestrów należy sprawdzać stan bitu BUSY. Wyjątkiem od tej reguły jest zapis do rejestru FCTL1 w trakcie operacji zapisu bloku pamięci. Wówczas, monitorowanie bitu BUSY nie jest konieczne. W tej sytuacji konieczna jest kontrola bitu WAIT. Zapis do rejestru FCTL1 przy wyzerowanym bicie WAIT spowoduje ustawienie flagi ACCVIFG. Jeśli programista aktywował przerwania od flagi ACCVIFG ustawiając bit ACCVIE w rejestrze IE, to w momencie jej ustawienia zostanie wywołana procedura obsługi przerwania.

Kontroler wyposażono w układ wewnętrznego generatora sygnału zegarowego. Konfigurując kontroler pamięci należy zdefiniować parametry pracy generatora (rejestr FCTL2). Nastawom podlegają wybór źródła sygnału taktującego generator oraz częstotliwość sygnału. Źródłem sygnału taktującego operacje kasowania i zapisu pamięci może być jeden z sygnałów zegarowych ACLK, MCLK, SMCLK (bit FSSELx). Częstotliwość sygnału taktującego generator Fftg musi być z przedziału 257…476 kHz. Dlatego też, jeśli częstotliwość źródła jest zbyt duża, należy ją podzielić (1…64, bit FNx). Tylko częstotliwość sygnału taktującego mieszcząca się w zdefiniowanym zakresie gwarantuje prawidłową pracę kontrolera pamięci FLASH!

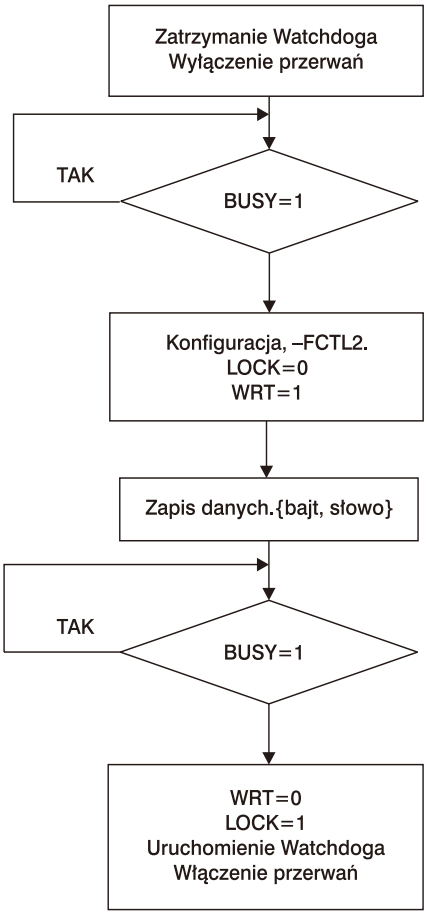
Kasowanie pamięci (**rys. 8a**) może być przeprowadzane sektorami (4819 cykli Fftg). Skasować można zawartość zarówno pamięci programu, jak i pamięci informacyjnej (5297 cykli Fftg). Konfiguracji trybu kasowania dokonuje się modyfikując wartości bitów ERASE, MERAS w rejestrze FCTL1. Operacja kasowania wymaga sztucznego zapisu danych pod adres kasowanego obszaru pamięci. Tylko wówczas uruchamiany jest wewnętrzny generator częstotliwości, a proces kasowania pamięci jest kontynuowany. W środowisku IAR, w trakcie programowania mikrokontrolera, domyślnie kasowana jest pamięć programu i pamięć informacyjna. Użytkownik ma jednak możliwość rezygnacji z kasowania pamięci informacyjnej. W tym celu, w opcjach projektu w kategorii FET Debugger, zakładce *Download*, należy wybrać opcję kasowania tylko pamięci programu (*Erase main memory*).

Dane do pamięci mogą być zapisywane jako bajty, słowa (rys. 8b), bądź też w postaci 64 bajtowych bloków (**rys. 8c**). Za wybór trybu zapisu odpowiadają bity BLKWRT, oraz WRT w rejestrze FCTL1. Przed każdym zapisem bajtu lub słowa uruchamiany jest wewnętrzny generator napięcia. Sprawia to, że czas zapisu jest wydłużony i trwa 35 cykli Fftg. W przypadku zapisu bloku danych, generator napięcia uruchamiany jest przed zapisem całego 64 bajtowego bloku, a zatrzymywany po jego zapisie. Czas zapisu pierwszego bajta/słowa w bloku wynosi 30 cykli Fftg (zapis + włączenie generatora), kolejne 63 bajty są zapisywane w czasie 21 cykli Fftg, a operacja zapisu kończy się wyłączeniem generatora napięcia (6 cykli Fftg).

W trybie debuggera, środowisko IAR zezwala na dostęp do pamięci mikrokontrolera (*View –> Memory*). Użytkownik może przeglądać jej zawartość oraz, w razie potrzeby, modyfikować dane w pamięci RAM.

W programie `FLASH.eww` dostępnym na płycie CD, zaimplementowano procedury obsługi odczytu, zapisu pamięci (bajt/słowo), zapisu pamięci blokami, oraz kasowanie sektorów pamięci. Program jest zmodyfikowaną wersją programu UART.eww. Informacja o zmianie stanu diody D1 zapisywana jest w pamięci FLASH mikrokontrolera (sektor B, pamięci informacyjnej). Po starcie mikrokontroler odczytuje pamięć FLASH i jeśli odnajdzie w niej znacznik SetD1 (przed startem dioda D1 była włączona) to ustawi diodę w stan aktywny.

Wewnętrzna pamięć FLASH mikrokontrolera może być programowana nie tylko z poziomu aplikacji użytkownika, ale również przy wykorzystaniu interfejsu JTAG, jak również przy użyciu wbudowanego w strukturę mikrokontrolera układu *Bootstrap Loader* (BSL, pod adresem 0x0C00). Istotną kwestią jest możliwość zabezpieczenie kodu programu przed odczytem, a nazywając rzeczy po imieniu, przed kradzieżą. Programista ma możliwość nieodwracalnego wyłączenia interfejsu JTAG (wypalenie bitu FUSE). Dostęp do BSL chroniony jest 256-bitowym hasłem. W omawianej serii x1xx, niepoprawne podanie hasła dostępu do BSL nie pociąga za sobą żadnych konsekwencji. Jest to spore niedociągnięcie, poprawione w serii x2xx.



Rys. 8. Algorytm zapisu do pamięci Flash bajtu lub słowa

Łukasz Krysiewicz

R E K L A M A