MSP430 w przykładach (10)

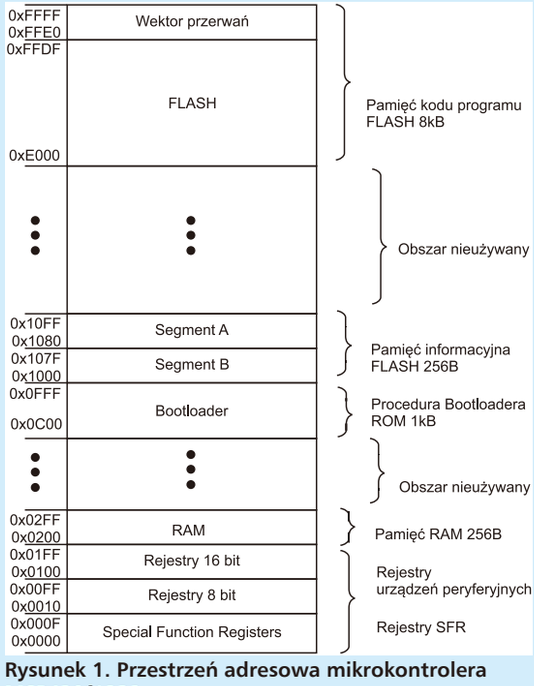
Obsługa wewnętrznej pamięci Flash

W tym artykule – kończącym cykl kursu programowania MSP430 – omówimy obsługę wewnętrznej pamięci Flash mikrokontrolera. Zaprezentujemy projekt zamka cyfrowego z pastylką cyfrową firmy Maxim-Dallas – DS19990A. Kurs zakończymy, krótkim podsumowaniem.

Mikrokontroler MSP430f1232, zainstalowany w module „Komputerek", wyposażono w 8 kB+256 B pamięci Flash. W procesorze dostępne są dwa rodzaje pamięci Flash: pamięć kodu programu (*Code Memory*, 8 kB), oraz pamięć informacyjna (*Info Memory*, 256 B). Oba rodzaje pamięci różnią się maksymalną liczbą cykli kasowania. Pamięć kodu programu (*Code Memory*) ma gwarantowane 10 tysięcy cykli kasowania. Pamięć informacyjna jest trwalsza i ma gwarantowane 100 tysięcy cykli kasowania. Po przekroczeniu maksymalnej, zdefiniowanej liczby cykli kasowania, pamięć może ulec uszkodzeniu. Dlatego też, to trwalsza pamięć informacyjna powinna być używana do przechowywania danych konfiguracji urządzenia.

Mikrokontrolery MSP430, w tym również MSP430f1232, mają architekturę von Neumanna (jedna pamięć danych i instrukcji, ciągła przestrzeń adresowa). Podstawowym słowem pamięci jest bajt. Dane o większej ilości bajtów zapisywane są w formacie „little endian". Dostęp do pamięci (odczyt/zapis) możliwy jest za pomocą bitów, bajtów (8 bitów), oraz słów (16 bitów). Mapę przestrzeni adresowej mikrokontrolera MSP430f1232 pokazano na **rysunku 1**.

Pamięć Flash mikrokontrolera MSP430f1232 podzielono na segmenty. Pamięć informacyjną tworzą dwa



Rysunek 1. Przestrzeń adresowa mikrokontrolera MSP430f1232

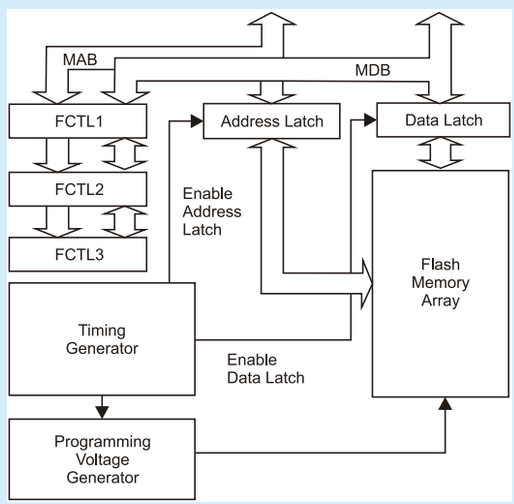
Dodatkowe materiały na CD/FTP:
ftp://ep.com.pl, user: 63241, pass: 741obq51

segmenty A i B. Pamięć kodu programu tworzy 16 segmentów. Każdy segment pamięci jest zbudowany z bloków pamięci o rozmiarze 64 bajtów. Segmenty A i B pamięci informacyjnej zbudowane są z dwóch bloków pamięci (rozmiar segmentu pamięci informacyjnej to128 bajtów). Segmenty 0…15 pamięci kodu programu zbudowane są z 8 bloków pamięci (rozmiar segmentu pamięci kodu programu to 512 bajtów). Segmentację pamięci w MSP430f1232 pokazano na **rysunku 2.**

Kontroler pamięci

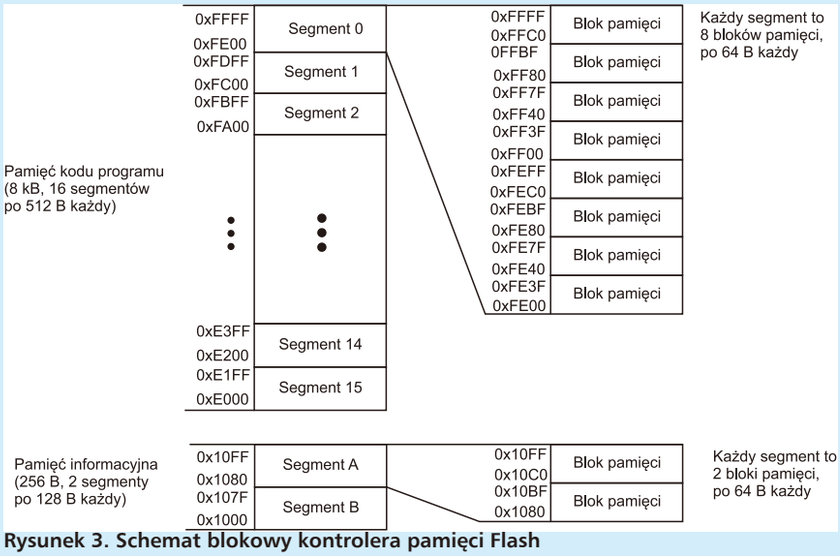
Mikrokontroler MSP430f1232 ma wewnętrzny kontroler pamięci Flash. Kontroler jest używany podczas operacji kasowania i programowania pamięci. W strukturę kontrolera wbudowano generator taktujący oraz generator napięcia. Budowę kontrolera ilustruje **rysunek 3**.

Kontroler pamięci Flash jest konfigurowany za pomocą 16-bitowych rejestrów: FCTL1, FCTL2, FCTL3. Opis rejestrów zamieszczono w materiałach dodatkowych na CD oraz na serwerze FTP. Dostęp do rejestrów konfiguracyjnych jest chroniony hasłem. W przypadku zapisu do rejestrów bez podania hasła, bądź z błędnym hasłem w rejestrze FCTL3, jest ustawiana flaga KEYV oraz wykonany restart PUC mikrokontrolera. Hasło dostępu do rejestrów konfiguracyjnych ma wartość 0x5A (w środowisku IAR definicja FWKEY). Hasło wpisujemy do bardziej znaczącego bajtu rejestru. Podczas odczytu rejestru bardziej znaczący bajt ma zawsze wartość 0x69, mniej znaczący ilustruje ustawienie bitów konfiguracyjnych.



Rysunek 2. Segmentacja pamięci w MSP430f1232

Kod programu sterującego pracą mikrokontrolera można umieścić albo w pamięci SRAM, albo Flash układu. W praktyce w 99 % przypadków oprogramowanie jest wgrywane do pamięci Flash. W artykule omówimy sposób obsługi kontrolera pamięci Flash w przypadku, gdy jest do niej wgrany program. Marginalny przypadek z programem w pamięci SRAM nie będzie dyskutowany.



Rysunek 3. Schemat blokowy kontrolera pamięci Flash

Generator taktujący

W moduł kontrolera pamięci Flash wbudowano generator taktujący. Wytwarza on sygnał *Fftg* używany do taktowania operacji kasowania i programowania pamięci Flash mikrokontrolera. Sygnał *Fftg* jest wytwarzany na podstawie jednego z sygnałów zegarowych mikrokontrolera: ACLK, MCLK, SMCLK. Źródło sygnału zegarowego konfiguruje bit *FSSELx* z rejestru *FCTL2*. Aby kontroler pamięci działał poprawnie, to częstotliwość sygnału *Fftg* musi być z przedziału 257…476 kHz. Jeśli częstotliwość sygnału wejściowego (sygnał zegarowy: ACLK/MCLK/SMCLK) jest zbyt wysoka, to należy ją podzielić. Dzielnik częstotliwości konfigurują bity *FNx* z rejestru *FCTL2*. Wartość dzielnika obliczamy ze **wzoru 10.1.** Minimalna wartość dzielnika wynosi 1, maksymalna 64.

(**10.1**)

$$fdiv = 32 \times FN5 + 16 \times FN4 + 8 \times FN3 + 4 \times FN2 + 2 \times FN1 + 1 \times FN0 + 1$$

gdzie:

fdiv – dzielnik częstotliwości sygnału zegarowego.

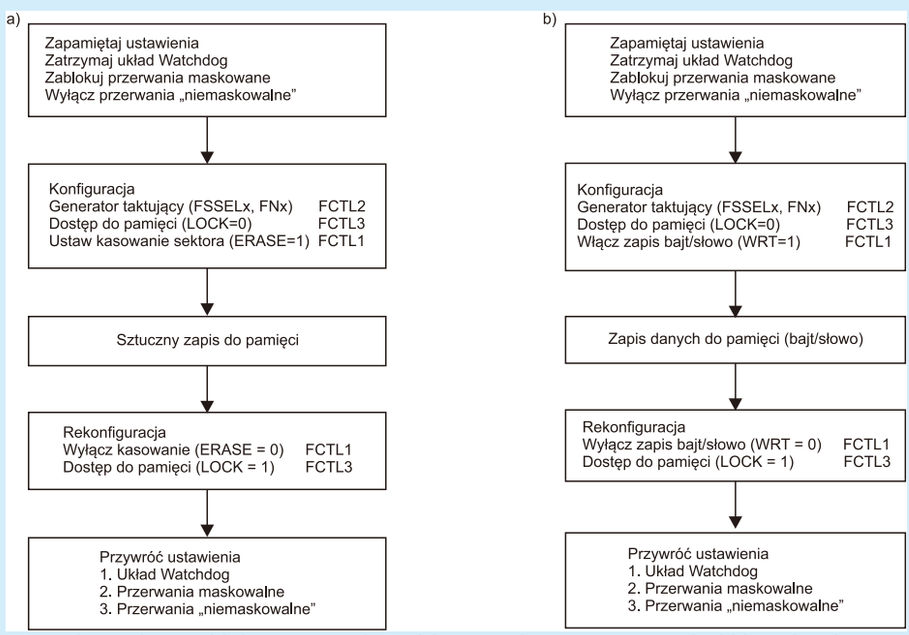
FN5…0 – bity konfiguracyjne FNx z rejestru FCTL2.

Generator napięcia

Moduł kontrolera pamięci Flash wyposażono w generator napięcia. Wytwarza on podwyższone napięcie elektryczne używane podczas kasowania i programowania pamięci. Generator jest włączany automatycznie na czas operacji kasowania i programowania pamięci. Aby moduł generatora napięcia działał prawidłowo, nasz mikrokontroler musi być zasilany napięciem o wartości powyżej 2,7 V! W mikrokontrolerach MSP430 z serii 2xx napięcie zmniejszono do 2,2 V, a w najnowszych układach z serii 5xx/6xx do 1,8 V. Jeśli napięcie zasilania mikrokontrolera będzie poniżej wartości 2,7 V, generator nie będzie działał poprawnie, a podczas kasowania i programowania pamięci Flash mogą wystąpić błędy.

Zabezpieczenia

Dostęp do rejestrów konfiguracyjnych kontrolera pamięci Flash jest chroniony hasłem. Dodatkowo, operacje zapisu i kasowania pamięci również zostały zabezpieczone (bit LOCK w rejestrze FCTL3). Aby kontroler pamięci Flash mógł zapisywać i kasować pamięć, bit LOCK musi być wyzerowany. Ustawiony bit blokuje dostęp do obu operacji. Pracując z kontrolerem pamięci Flash należy również pamiętać, że operacje zapisu do rejestrów konfiguracyjnych oraz kasowania i programowania pamięci Flash nie mogą zostać przerwane. Dlatego też, na czas pracy z kontrolerem pamięci, należy wyłączyć przerwania ma-



Rysunek 4. Schemat blokowy procedury a) kasowania segmentu pamięci b) zapisu do pamięci

Listing 1. Procedura kasowania segmentu pamięci Flash

```
/* procedura kasowania segmentu pamięci Flash
mikrokontrolera */
void FlashKasujSegment(char * adres)
{
  unsigned int  wdtctl;  // zmienne lokalne
  unsigned char ie1;
// zapamiętaj konfigurację
  wdtctl  = WDTCTL;  // układu Watchdog
  ie1     = IE1;  // oraz przerwań niemaskowalnych
  WDTCTL =  WDTPW + WDTHOLD;  //zatrzymaj układ Watchdog
Timer
  __bic_SR_register(GIE);  // zablokuj przerwania
maskowalne
  IE1 &=~ (NMIIE + OFIE + ACCVIE);  // wyłącz przerwania
niemaskowalne
// konfiguracja generatora taktującego
// ustaw źródło SMCLK=DCOCLK~740 kHz
// podziel częst. SMCLK przez 2
//(~740 kHz/2)=~370 kHz jest w wymaganym zakresie
256..476 kHz
  FCTL2 = FWKEY + FSSEL_3 + FN0;
  FCTL3 = FWKEY;  // odblokuj dostęp do pamięci (LOCK=0)
  FCTL1 = FWKEY + ERASE;  // włącz kasowanie segmentu
(ERASE=1)
  *adres = 0x00;  // sztuczny zapis,
  FCTL1 = FWKEY;  // wyłącz kasowanie segmentu (ERASE=0)
  FCTL3 = FWKEY + LOCK;  // zablokuj dostęp do pamięci
(LOCK=1)
// przywróć konfigurację
  wdtctl &= 0x00FF;  // wyzeruj hasło odczytu rej. WDTCTL
  wdtctl |= WDTPW;  // ustaw hasło zapisu rej. WDTCTL
  WDTCTL = wdtctl;  // ustaw konfigurację uk. Watchdog
  __bis_SR_register(GIE);  // odblokuj przerwania
maskowalne
    IE1    = ie1;  // przywróć konfigurację źródeł
przerwań
}
```

Listing 2. Procedura zapisu do pamięci Flash

```
/* procedura zapisu danych do pamięci Flash
mikrokontrolera */
void FlashZapisz(char* adres, char* dane, int ilosc)
{
  unsigned int  indeks;  // zmienne lokalne
  unsigned int  wdtctl;
  unsigned char ie1;
// zapamiętaj konfigurację
  wdtctl = WDTCTL;  // układu Watchdog
  ie1 = IE1;  // oraz przerwań „niemaskowalnych”
  WDTCTL = WDTPW + WDTHOLD;  // zatrzymaj układ Watchdog
Timer
  __bic_SR_register(GIE);  // zablokuj przerwania
maskowalne
  IE1 &=~ (NMIIE + OFIE + ACCVIE);  // wyłącz przerwania
niemaskowalne
// konfiguracja generatora taktującego
// ustaw źródło SMCLK=DCOCLK= ~740 kHz i podziel częst.
SMCLK przez 2
  FCTL2 = FWKEY + FSSEL_3 + FN0;
  FCTL3 = FWKEY;  // odblokuj dostęp do pamięci (LOCK=0)
  FCTL1 = FWKEY + WRT;  // włącz zapis bajt/słowo (WRT=1)
  for(indeks=0; indeks < ilosc; indeks++)  // zapisz dane
(bajtowo)
    *((char*)adres ++) = *((char*)dane ++);
  FCTL1 = FWKEY;  // wyłącz zapis bajt/słowo (WRT=0)
  FCTL3 = FWKEY + LOCK;  // zablokuj dostęp do pamięci
(LOCK=1)
// przywróć ustawienia
  wdtctl &= 0x00FF;  // wyzeruj hasło odczytu rej. WDTCTL
  wdtctl |= WDTPW;  // ustaw hasło zapisu rej. WDTCTL
  WDTCTL =  wdtctl;  // ustaw konfigurację uk. Watchdog
  __bis_SR_register(GIE);  // odblokuj przerwania
maskowalne
  IE1    = ie1;  // przywróć ustawienia źródeł przerwań
}
```

skowalne i niemaskwalne), oraz zatrzymać pracę układu Watchdog.

Kasowanie pamięci

Operacja kasowania pamięci Flash polega na ustawieniu bitów pamięci (zamazanie pamięci jedynkami). W MSP430 pamięć można kasować segmentami, można również od razu skasować całą pamięć kodu programu oraz dodatkowo, pamięć informacyjną. Jeśli program jest wgrany do pamięci SRAM, to pamięć Flash można kasować blokami. Sposób kasowania pamięci konfigurują bity *ERASE* i *MERAS* z rejestru *FCTL1*. Operacja kasowania wymaga „sztucznego” zapisu danych pod adres kasowanego obszaru pamięci. Procedurę kasowania segmentu pamięci Flash pokazano na **listingu 1**. Schemat blokowy procedury ilustruje **rysunek 4a**.

Kasowanie segmentu pamięci Flash w MSP430f1232 trwa 4819 taktów przebiegu *Fftg*. Kasowanie całej pamięci programu, plus ewentualne kasowanie pamięci informacyjnej, trwa 5297 taktów *Fftg*. W trakcie kasowania pamięci pobór prądu mikrokontrolera wynosi 3…7 mA.

Zapis do pamięci

Zapisując dane do pamięci Flash zawsze istnieje możliwość wyzerowania bitu pamięci, nigdy jego ustawienia (zmiana z 0 na 1). Aby ustawić bit pamięci, należy ją skasować. Dlatego też, w wielu aplikacjach, zapis do pamięci jest poprzedzany jej kasowaniem.

Dane do pamięci mogą być zapisywane jako bity, bajty lub słowa. Mogą być też zapisywane w formie bloków 64-bajtowych. Tryb zapisu do pamięci konfigurują bity *BLKWRT*, *WRT* z rejestru *FCTL1*. Zapis bloku pamięci nie jest obsługiwany, gdy program znajduje się w pamięci Flash i nie będzie omawiany. Procedurę zapisu danych do pamięci Flash pokazano na **listingu 2**. Schemat blokowy procedury ilustruje **rysunek 4b**. Zapis bajtu lub słowa do pamięci Flash w MSP430f1232 trwa 35 taktów *Fftg*. W trakcie zapisu danych pobór prądu mikrokontrolera wynosi 3…5 mA.

Przykłady

Zaprezentujemy dwa przykłady obsługi kontrolera pamięci Flash w MSP430f1232. Kody źródłowe programów oraz filmy ilustrujące ich działanie zamieszczono na płycie CD i serwerze FTP.

W przykładzie „Obsługa pamięci Flash” zademonstrujemy sposób użycia procedur obsługi kontrolera pamięci Flash (kasowanie pamięci/zapis danych do pamięci/odczyt danych z pamięci). Działanie kontrolera pamięci przećwiczymy pracując z segmentem B pamięci informacyjnej mikrokontrolera.

W przykładzie „Zamek cyfrowy” obsłużymy odczyt pastylki identyfikacyjnej Dallasa – DS19990A. Każda pastylka iButton ma unikalny, 48-bitowy numer seryjny zapisany w pamięci ROM. Odczyt kodu pastylki jest możliwy za pomocą interfejsu 1-Wire. W programie raz na sekundę będziemy sprawdzali stan linii interfejsu 1-Wire, a po wykryciu pastylki iButton będziemy odczytywali jej kod. Odczytanego kodu pastylki użyjemy jako klucza cyfrowego do otwarcia zamku. W przykładzie, naszym „zamkiem” będzie dioda LED koloru zielonego podłączona do linii wyjścia mikrokontrolera. Ustawienie na linii na czas 0,5 s stanu odpowiada zaświeceniu się diody i symuluje otwarcie zamka.

Program działa w dwóch trybach: programowania i sterowania zamkiem. W trybie programowania odczytany kod pastylki iButton jest zapisywany w pamięci Flash mikrokontrolera (programowanie klucza cyfrowego w pamięci Flash mikrokontrolera). W trybie sterowania zamkiem odczytany kod pastylki jest porównywany z kodem zapisanym w pamięci Flash mikrokontrolera. Jeśli oba kody są identyczne, to zamek jest otwierany.

Obsługa pamięci Flash

Program „Obsługa pamięci Flash” uruchamiamy korzystając z modułu „Komputerek”. Wszystkie zworki układu należy ustawić w pozycji IO/Off.

Tabela 1. Programy prezentowane w kolejnych artykułach cyklu

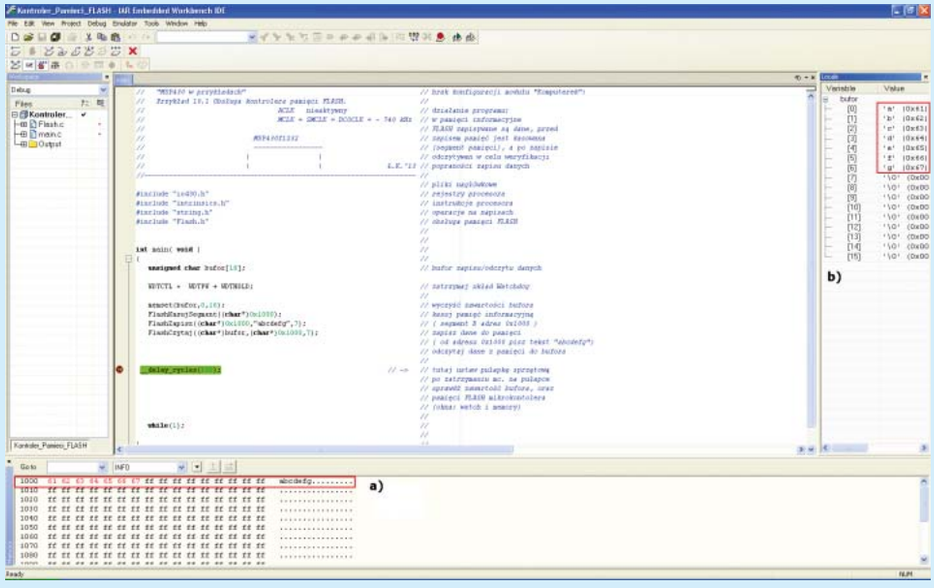
Nr	Tytuł artykułu	Opis	Przykłady		Publikacja
1	Wprowadzenie.	Budowa modułu „Komputerek". Właściwości MSP430f1232.	xxx		EP09/2012
2	Konfiguracja układu zegarowego.	Działanie modułu zegarowego Basic Clock. Przykłady konfiguracji układu zegarowego.	2.1 – Częstotliwości sygnału DCOCLK.	Pomiar częstotliwości sygnału DCOCLK.	EP10/2012
			2.2 – Obsługa układu zegarowego.	Przykłady konfiguracji układu zegarowego.	
3	Projektowanie urządzeń "low power".	Zasady projektowania urządzeń o niskim poborze mocy. Tryby uśpienia LPMx w MSP430.	xxx		EP11/2012
4	Obsługa portów wejścia/wyjścia.	Konfiguracja oraz sterowanie liniami we/wy MSP430.	4.1– Sterowanie diodą RGB.	Obsługa wyjścia.	EP12/2012
			4.2 – Dzwonek do drzwi.	Obsługa wejścia.	
			4.3 – Obsługa wyświetlacza LCD.	Obsługa wyświetlacza alfanumerycznego 2×16 ze sterownikiem Hitachi HD44780	
5	Konfiguracja układu Watchdog.	Moduł Watchdog Timer w trybie pracy restartu, oraz zegara.	5.1 – Symulator latarni morskiej.	Układ Watchdog w trybie restartu.	EP01/2013
			5.2 – Sekundnik.	Układ Watchdog w trybie zegara.	
6	Timer w trybie odmierzania czasu.	Użycie licznika TAR do odmierzania czasu.	6.1 – Zegarek elektroniczny.	Licznik TAR w trybie pracy „licz do".	EP02/2013
			6.2 – Pozytywka.	Licznik TAR w trybie pracy „ciągłej".	
7	Timer w trybie wejścia-wyjścia.	Licznik TAR oraz rejestry TACCRx w trybie Compare/ Capture (porównaj/przechwyć).	7.1 – Generator przebiegu prostokątnego.	Licznik TAR i rejestry TACCRx w trybie Compare. Generator sygnału PWM.	EP03/2013
			7.2 – Pomiar czasu trwania impulsu.	Licznik TAR i rejestry TACCRx w trybie Capture. Pomiar czasu/częstotliwości sygnału.	
8	Transmisja szeregowa UART, SPI.	Obsługa modułu kontrolera transmisji szeregowej USART.	8.1 – Komunikacja mikrokontrolera z PC.	Obsługa interfejsu UART.	EP04/2013
			8.2 – Gra zręcznościowa.	Obsługa interfejsu SPI.	
9	Pomiary analogowe.	Pomiary analogowe z wykorzystaniem modułu przetwornika ADC10.	9.1 – Termometr pokojowy.	Obsługa wbudowanego czujnika temperatury.	EP05/2013
			9.2 – Woltomierz cyfrowy.	Pomiar napięcia na wejściu analogowym.	
			9.3 – Pomiar zużycia baterii.	Pomiar napięcia zasilania mikrokontrolera.	
10	Obsługa wewnętrznej pamięci Flash. Podsumowanie kursu.	Budowa i działanie kontrolera wewnętrznej pamięci Flash.	10.1 – Obsługa pamięci Flash.	Użycie procedur obsługi pamięci Flash.	EP06/2013
			10.2 – Zamek cyfrowy.	Odczyt klucza cyfrowego DS1990A.	

Działanie programu przeanalizujemy korzystając z emulatora środowiska IAR. Uruchamiamy emulator (*Project -> Download and Debug* lub skrót klawiszowy *Ctrl+D*). W linii programu wskazanej w komentarzu ustawiamy pułapkę sprzętową (*Edit -> Toggle Breakpoint* lub klawisz *F9*). Następnie uruchamiamy program (*Debug -> Go*, klawisz F5). W programie jest zerowana zawartość bufora odczytu pamięci oraz kasowany segment B pamięci informacyjnej mikrokontrolera. W pamięci Flash, pod adresem 0x1000 (początek segmentu B) jest zapisywany ciąg 7 znaków: „abcdefg". Następnie, zawartość pamięci jest odczytywana do bufora odczytu (7 bajtów pamięci począwszy od adresu 0x1000). W kolejnej instrukcji działanie programu jest zatrzymywane na pułapce sprzętowej. W oknie podglądu zmiennych lokalnych programu (*View -> Locals*) możemy sprawdzić zawartość danych w buforze odczytu. Ewentualnie możemy podejrzeć zawartość pamięci Flash mikrokontrolera (*View -> Memory*). Jeśli w buforze odczytu i w pamięci mikrokontrolera jest ciąg znaków „abcdefg", to programowanie pamięci wykonało się prawidłowo. Okna emulatora z włączonym podglądem zmiennych oraz zawartości pamięci Flash mikrokontrolera pokazano na **rysunku 5**.

Zamek cyfrowy

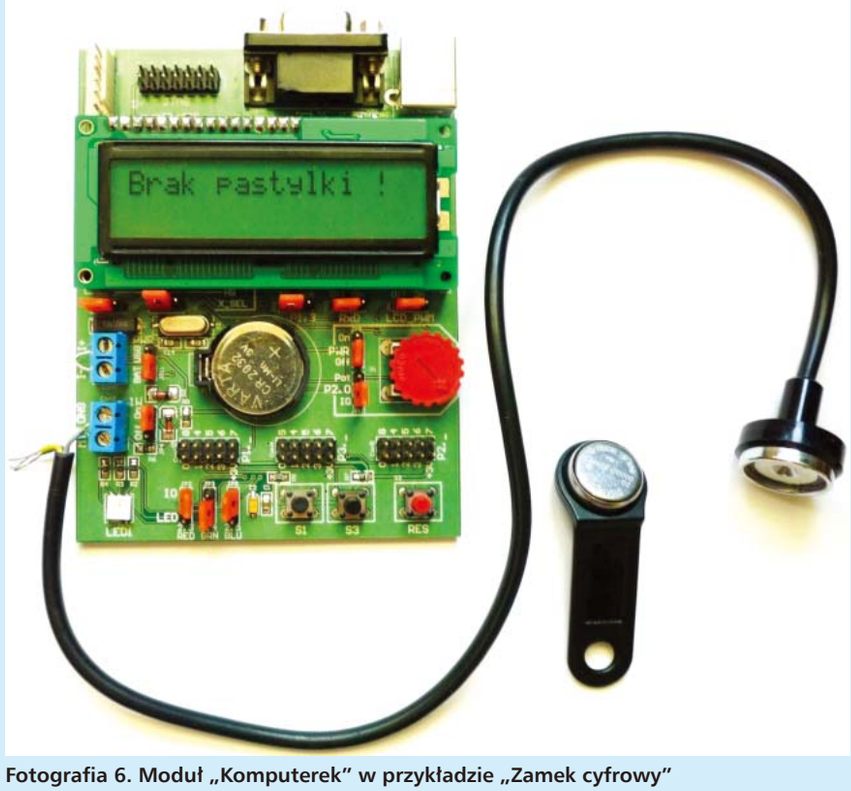
Program „Zamek cyfrowy" uruchamiamy korzystając z modułu „Komputerek". Zworki JP7 i JP8 należy ustawić w pozycji *LF*. Zworkę JP4 (złącze interfejsu 1-Wire) należy ustawić w pozycji *On*. Zworkę JP3 (dioda LED) należy ustawić w pozycji LED. Pozostałe zworki układu należy ustawić w pozycji IO/Off. W złączu szpilkowym Dis1 montujemy wyświetlacz LCD. Natomiast w złączu 1-Wire czytnik pastylek iButton. Wygląd zmontowanego modułu „Komputerek" pokazano na **fotografii 6**.

W pierwszych liniach programu dołączane są bibliotek obsługi interfejsu 1-Wire oraz pastylki iButton DS1990A (biblioteki napisane w języku C). W programie głównym jest zatrzymywana praca układu Watchdog, konfigurowane linie I/O mikrokontrolera, wyświetlacz LCD oraz sygnały zegarowe. Częstotliwość sygnału zegarowego MCLK jest konfigurowana na 3 **MHz (**kalibracja wewnętrznego generatora DCO). W programie sygnał MCLK będzie używany podczas transmisji 1-Wire. Częstotliwość sygnału musi być stabilna, dlatego też w pętli głównej programu wywoływana jest procedura kalibracji sygnału.



Rysunek 5. Emulatora środowiska IAR. Podgląd a) pamięci Flash b) zmiennych lokalnych

W programie głównym, raz na sekundę odczytywany jest stan linii 1-Wire. Gdy mikrokontroler wykryje pastylkę iButton, to odczytywany jest numer identyfikacyjny pastylki. Poprawnie odczytany kod pastylki jest wyświetlany na ekranie LCD. **Żeby zaprogramować zamek**, wciskamy przyciski SW1 i SW2, a do czytnika przykładamy pastylkę iButton. Odczytany numer identyfikacyjny pastylki zostanie zapisany w pamięci Flash mikrokontrolera (pamięć informacyjna, segment B). Teraz zamek cyfrowy „otworzy się" (stan wysoki na wyjściu przez 0,5 s, włączona dioda LED koloru zielonego) skoro tylko zostanie odczytany kod z pastylki, której numer identyfikacyjny został zapisany w pamięci mikrokontrolera. Pozostałe pastylki/klucze są ignorowane.



Fotografia 6. Moduł „Komputerek" w przykładzie „Zamek cyfrowy"

Podsumowanie kursu

Przez ostatnie 10 miesięcy publikowaliśmy kurs programowania MSP430. Przykłady prezentowane w trakcie kursu były tworzone w języku C przy użyciu środowiska programistycznego IAR EW (wersja darmowa z ograniczeniem wielkości kodu wynikowego programu do 4 kB). Platformą sprzętową kursu był moduł „Komputerek" z mikrokontrolerem MSP430f1232 (moduł można kupić w sklepie AVT, oznaczenie kitu to: AVTMSP430/2). W materiałach dodatkowych na CD oraz na serwerze FTP zamieszczamy kody źródłowe programów prezentowanych podczas kursu, dokumentację techniczną modułu „Komputerek" oraz mikrokontrolera MSP430f1232, opis jego rejestrów i filmy ilustrujące działanie przykładów. Opis programów prezentowanych podczas kursu oraz listę odcinków kursu prezentuje **tabela 1**.

Zainstalowany w module „Komputerek" mikrokontroler MSP430f1232 jest układem z serii 1xx. W literaturze seria 1xx jest nazywana podstawową/bazową serią MSP430 i dlatego też do przeprowadzenia kursu został wybrany układ z serii 1xx. W praktyce, znając działanie mikrokontrolera z serii 1xx, można „szybko" opanować działanie mikrokontrolerów MSP430 z innych, nowszych serii. Różnice pomiędzy serią 1xx, a seriami 2xx, 4xx, G2xx są nieznaczne. Przejście z serii 1xx na najnowsze układu z serii 5xx/6xx oraz FR5xx (mikrokontrolery z pamięcią FRAM) również nie jest kłopotliwe.

Na zakończenie warto dodać, że MSP430 to ponad 400 typów mikrokontrolerów i aż 9 serii produkcyjnych układów. W bogatej ofercie MSP430 można znaleźć moduły wyposażone w kontroler LCD, interfejs komunikacyjny USB i transceiver radiowy RF pracujący w paśmie ISM do 1 GHz. Nowością w ofercie są mikrokontrolery z pamięcią FRAM. Praktycznie każdy projekt można zrealizować korzystając z MSP430. A wszystko przy znikomym, prawie zerowym poborze energii!

Łukasz Krysiewicz, EP