Pamięci nieulotne w systemach mikroprocesorowych, część 2

Konstruując urządzenia z mikrokontrolerami, często stajemy przed koniecznością zapewnienia przechowywania danych także po wyłączeniu lub zaniku zasilania. Łatwo jest, jeśli są to takie stałe jak: napisy menu (dla przykładu w różnych językach), obrazy, stałe parametry nastaw. Gorzej, jeśli musimy przechować zmienne. Jeszcze trudniej, gdy muszą one być zapamiętane również w przypadku awarii napięcia zasilania.

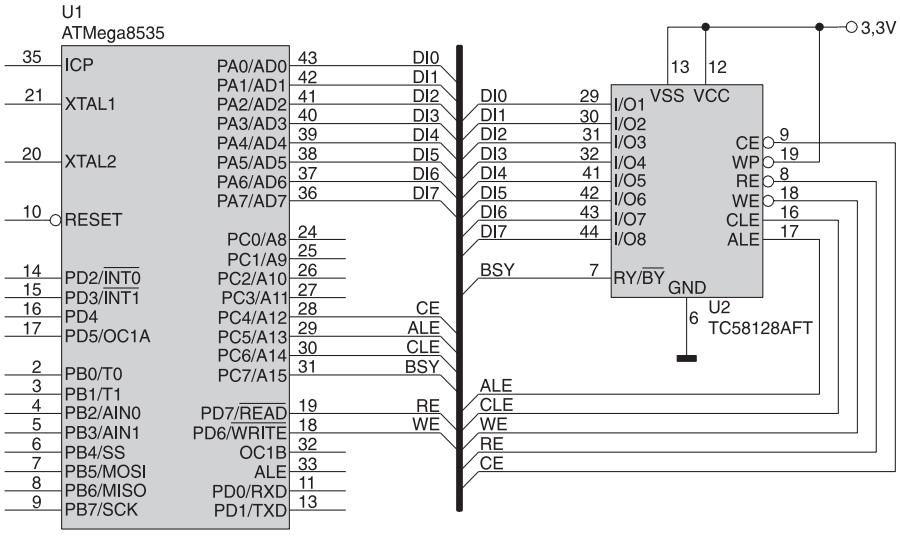
Przechowywanie danych w wewnętrznej pamięci EEPROM mikrokontrolera (AT89S8252, AT90S2313)

Do przechowywania danych nastaw, napisów, zmiennych definiowanych przez użytkownika i tym podobnych doskonale nadaje się wbudowana w strukturę mikrokontrolera AT89S8252 pamięć EEPROM. Ma ona dużą jak na świat mikrokontrolerów pojemności - „aż" 2 kB. Podobnie jak w przypadku wbudowanej w strukturę pamięci RAM, nie wymaga żadnych dodatkowych układów dekodera adresu, dodatkowego napięcia zasilającego itp. Producent gwarantuje co najmniej 100 tysięcy cykli zapisu/odczytu, co przy przeciętnym użytkowaniu wystarcza na co najmniej kilka lat.

Wewnętrzna pamięć EEPROM umieszczona została przez producenta w obszarze XDATA. Dostęp do niej odbywa się za pomocą rozkazu MOVX z adresem komórki w rejestrze DPTR. Oczywiście rozkaz MOVX służy również do dostępu do urządzeń podłączonych na zewnątrz mikrokontrolera i dlatego też producent wprowadził dodatkowy bit o nazwie EEMEN umożliwiający wybór zgodnego z intencją programisty sposobu działania MOVX. Ustawienie bitu powoduje, że rozkaz ten zapisuje/odczytuje dane do/z wewnętrznego EEPROM.

Odczyt danych jest bardzo szybki, gorzej jest z zapisem. Wewnętrzny układ kontrolera EEPROM umożliwia dostęp do pojedynczego bajtu danych. Każda operacja zapisu związana jest prawdopodobnie z koniecznością ustawienia wszystkich bitów w słowie. Piszę prawdopodobnie, ponieważ nie posiadam informacji na temat szczegółów funkcjonowania operacji zapisu i nie wiem, czy ustawiane są wszystkie bity słowa (przed zapisem komórka musi zawierać „1" na pozycjach „1" zapisywanego słowa danych), czy tylko te, dla których operacja ta jest niezbędna. Tak czy inaczej, operacja zapisu pojedynczego bajtu trwa około 1 ms. Jej zakończenie sygnalizowane jest przez wyzerowanie bitu BSY/RDY umieszczonego w rejestrze WMCON. Na **list. 2** (EP10/2003) przedstawiono fragment programu odpowiedzialny za zapis i odczyt danych w pamięci EEPROM mikrokontrolera AT89S8252, a na **list. 3** (EP10/2003) przykłady ich użycia.

W podobny sposób można uzyskać dostęp do pamięci EEPROM wbudowanej w strukturę mikrokontrolera AT90S2313. Bity kontrolne, sposób ich funkcjonowania oraz nazwy są takie same, jak dla AT89. Zupełnie inne są jednak nazwy rejestrów specjalnych. Na **list. 3** pokazano przykładowe funkcje zapisujące i odczytujące bajt do/z pamięci EEPROM. Niestety - ten typ mikrokontrolera posiada wbudowane tylko 128 bajtów EEPROM.



Rys. 5. Przykład połączenia TC58128-AFT z mikrokontrolerem AVR

Tab. 1. Zestawienie pamięci Flash-NAND produkcji firmy Toshiba

Nazwa	Org.	Obudowa	Zasilanie	Rozmiar strony [B]	Rozmiar bloku [kB]
TC58V64BFT	8M X 8	TSOP-II 44 wypr.	2,7V do 3,6V	528	8
TC58V64BFTI	8M X 8	TSOP-II 44 wypr.	2,7V do 3,6V	528	8
TC58DVM72F1FT00	8M X 16	TSOP-I 48 wypr.	2,7V do 3,6V	264	8
TC58128AFT	16M X 8	TSOP-I 48 wypr.	2,7V do 3,6V	528	16
TC58128AFTI	16M X 8	TSOP-I 48 wypr.	2,7V do 3,6V	528	16
TC581282AXB	16M X 8	TFBGA-56	2,7V do 3,6V	528	16
TC58DVM72A1FT00	16M X 8	TSOP-I 48 wypr.	2,7V do 3,6V	528	16
TC58DVM82F1FT00	16M X 16	TSOP-I 48 wypr.	2,7V do 3,6V	264	8
TC58256AFT	32M X 8	TSOP-I 48 wypr.	2,7V do 3,6V	528	16
TC58256AFTI	32M X 8	TSOP-I 48 wypr.	2,7V do 3,6V	528	16
TC582562AXB	32M X 8	TFBGA-56	2,7V do 3,6V	528	16
TC58DVM82A1FT00	32M X 8	TSOP-I 48 wypr.	2,7V do 3,6V	528	16
TC58DVM82A1XBJ1	32M X 8	TFBGA-56	2,7V do 3,6V	528	16
TC58512FT	64M X 8	TSOP-I 48 wypr.	2,7V do 3,6V	528	16
TC58512FTI	64M X 8	TSOP-I 48 wypr.	2,7V do 3,6V	528	16
TC58DVM92A1FT00	64M X 8	TSOP-I 48 wypr.	2,7V do 3,6V	528	16
TH58100FT	128M X 8	TSOP-I 48 wypr.	2,7V do 3,6V	528	16
TC58DVG02A1FT00	128M X 8	TSOP-I 48 wypr.	2,7V do 3,6V	528	16
TC58NVG0S3AFT05	128M X 8	TSOP-I 48 wypr.	2,7V do 3,6V	2112	128
TH58100FTI	128M X 8	TSOP-I 48 wypr.	2,7V do 3,6V	528	16
TH58NVG1S3AFT05	256M x 8	TSOP-I 48 wypr.	2,7V do 3,6V	2112	128k

Podobnie jak w przypadku pamięci RAM, również i EEPROM może przechowywać zmienne i nastawy ważne dla funkcjonowania aplikacji. Ze względu jednak na ograniczoną liczbę cykli zapisu/odczytu, wszelkich zmian powinno się dokonywać na zmiennej znajdującej się w pamięci RAM a jedynie w przypadku zaniku napięcia zasilania czy też jego załączenia, operacje zmienne powinny zostać zapamiętane czy odtworzone z pamięci EEPROM. Dla wydłużenia żywotności urządzenia, pamięci tej powinno się używać tak rzadko, jak to tylko jest możliwe.

Wykorzystanie zewnętrznej pamięci Flash do przechowywania danych

O ile opisywane wcześniej rozwiązania mogą posłużyć do zapamiętywania danych w sytuacji awaryjnej, o tyle użycie pamięci Flash o dużej pojemności wiąże się najczęściej z koniecznością budowy czegoś w rodzaju „dysku krzemowego" do przechowywania: obrazów, przetworzonego na postać cyfrową dźwięku i innych. Gwałtowny rozwój pamięci Flash w ostatnich latach został zapoczątkowany głównie przez aplikacje związane z telekomunikacją, a ściślej z telefonami GSM. Współczesny telefon wyposażony jest bardzo często w kilka lub kilkadziesiąt MB pamięci Flash służącej do przechowywania komunikatów SMS, nagrań, plików zawierających muzykę czy obrazy. Jako przykład układów przeznaczonych do takich właśnie zastosowań, chcę zaprezentować rodzinę układów firmy Toshiba (TAEC) oznaczonych symbolem TC58. Wykaz pamięci Flash-NAND produkowanych przez tę firmę znajduje się w **tab. 1**.

Wbrew pozorom, wykonanie interfejsu do obsługi pamięci Flash nie jest trudne. Innym zagadnieniem jest ewentualne wykonanie systemu plików na wzór tego stosowanego przy zapisie dysków (czy dyskietek) w systemach operacyjnych „dużych" komputerów, jak na

Tab. 2. Wykaz poleceń akceptowanych przez kontroler pamięci TC58

Opis polecenia	1-szy bajt	2-gi bajt	Akceptowane w stanie zajętości?
Wprowadzanie danych (*Serial Data Input*)	0x80	-	nie
Tryb odczytu numer 1 (*Read Mode 1*)	0x00	-	nie
Tryb odczytu numer 2 (*Read Mode 2*)	0x01	-	nie
Tryb odczytu numer 3 (*Read Mode 3*)	0x50	-	nie
Zerowanie kontrolera (*Reset*)	0xFF	-	tak
Programowanie strony (*Auto Program*)	0x10	-	nie
Kasowanie bloku danych (*Auto Block Erase*)	0x60	0xD0	nie
Odczyt statusu (*Status Read*)	0x70	-	tak
Odczyt sygnatury (*ID Read*)	0x90	-	nie

List. 5. Przykład realizacji programowego interfejsu do obsługi pamięci Flash TC58128

```
#define PAGE_SIZE  528             // rozmiar strony
#define PAGES_PER_BLOCK   32       // liczba stron w bloku dla danego typu NAND
#define BLOCK_PER_FLASH 1024;      // liczba bloków na pamięć

at 0x00 pdata FLASH_BASE;          // adres bazowy pamięci FLASH

#define FLASH_READY    (PINC & 0x80)           // wyprowadzenie BUSY pamięci FLASH
#define FLASH_CLE_SET  PORTC = PORTC | 0x40    // ustawienie sygnału CLE pamięci FLASH
#define FLASH_CLE_CLR  PORTC = PINC & (~0x20) // zerowanie sygnału CLE pamięci FLASH
#define FLASH_ALE_SET  PORTC = PORTC | 0x20    // ustawienie sygnału ALE pamięci FLASH
#define FLASH_ALE_CLR  PORTC = PINC & (~0x20) // zerowanie sygnału CLE pamięci FLASH
#define WAIT_4_READY while (!FLASH_READY)      // pętla - oczekiwanie na gotowość FLASH
#define FLASH_CE_SET   PORTC = PORTC | 0x10    // ustawienie sygnału CE pamięci FLASH
#define FLASE_CE_CLR   PORTC = PINC & (~0x10) // zerowanie sygnału CE pamięci FLASH

#define WORD unsigned int
#define BYTE unsigned char

/* odczyt kodu wytwórcy i kodu pamięci FLASH (odczyt sygnatury, PDF file str.14);
starszy bajt zawiera kod wytwórcy, młodszy kod urzšdzenia (rodzaju FLASH) */
WORD F_ReadStatus(void)
{
int temp;
   FLASH_ALE_CLR;          // wyzerowanie sygnału ALE
   FLASH_CLE_SET;          // ustawienie sygnału CLE
   FLASH_CE_CLR;           // zerowanie sygnału CS pamięci FLASH, wybór pamięci
   FLASH_BASE = 0x90;      // zapis bajtu 0x90 do pamięci FLASH-1
   FLASH_CLE_CLR;          // wyzerowanie sygnału CLE
   FLASH_ALE_SET;          // ustawienie sygnału ALE
   FLASH_BASE = 0;         // zapis bajtu o wartości 0x00 do FLASH
   FLASH_ALE_CLR;          // wyzerowanie sygnału ALE
   temp = (int)FLASH_BASE;// zapis do zmiennej temp wartości bajtu pobranego z FLASH
   temp <<= 8;             // kod wytwórcy
   temp |= (int)FLASH_BASE;   // dodanie do słowa sygnatury
   FLASH_CE_SET;           // wyłączenie pamięci FLASH poprzez ustawienie sygnału CE
   return (temp);          // funkcja zwraca bajt sygnatury FLASH
}

// odczyt strony o rozmiarze PAGE_SIZE z pamięci FLASH (sequential read (1), str.11 w PDF)
short FLASH_ReadPage(WORD wPageNum, char *pBuff)
{
   int temp;

   if (!FLASH_READY) return (-1); // jeśli pamięć nie gotowa funkcja kończy prace i zwraca -1
   FLASH_ALE_CLR;      // zerowanie sygnału ALE
   FLASH_CE_CLR;       // załączenie wyboru pamięci FLASH-1 (zerowanie sygnału CE)
   FLASH_CLE_SET;      // ustawienie sygnału CLE
   FLASH_BASE = 0;     // zapis bajtu o wartości 0x00 do FLASH, wybór trybu 1
   FLASH_CLE_CLR;      // wyzerowanie sygnału CLE
   FLASH_ALE_SET;      // ustawienie sygnału ALE
   FLASH_BASE = 0;     // ustawienie adresu kolumny w pamięci FLASH na 0x00
   FLASH_BASE = (unsigned char)wPageNum;          // zapis młodszego bajtu adresu strony
   FLASH_BASE = (unsigned char)(wPageNum>>8);     // zapis starszego bajtu adresu strony
   FLASH_ALE_CLR;      // zerowanie sygnału ALE
   WAIT_4_READY;       // oczekiwanie na zakończenie operacji przez FLASH
   for (temp = 0; temp < PAGE_SIZE; temp++) *pBuff++ = FLASH_BASE;
                       // pobranie kolejnych bajtów
   FLASH_CE_SET;       // wyłączenie pamięci poprzez ustawienie sygnału wyboru
   return (0);         // jeśli wykonanie zakończyło się sukcesem f.zwraca 0x00
}

/* odczyt części strony pamięci FLASH w trybie 1,2 lub 3; tryb wybierany jest w zależności
   od rozmiaru parametru OFFSET (PDF strony 11 i 12) */
short FLASH_ReadPartialPage(WORD wPageNum, void *pBuff, WORD offset, WORD length)
{
   short temp;         // zmienna tymczasowa; wykorzystana do inkrementacji adresu
   BYTE command;       // zmienna zawierająca komendę
   *pBuffer = (BYTE*)pBuff;   // wskaźnik do bufora w pamięci RAM

   if (offset + length > PAGE_SIZE) return (-1);  // jeśli offset i liczba bajtów do
                       // odczytu są większe od rozmiaru strony, funkcja kończy prace
                       // zwracając kod błędu
   if (offset < 256)           // jeśli offset jest mniejszy od 256, wybierany jest tryb 1
       command = 0;            // (command = 0)
   else if (offset < 512)      // jeśli offset jest większy od 256 ale mniejszy od 512,
       command = 1;            // wybierany jest tryb 2 (command = 1)
   else                        // w innych przypadkach wybierany jest tryb 3 (command = 50H)
       command = 0x50;

   if (!FLASH_READY) return (-1); // pamięć jest zajęta, f.kończy pracę zwracając kod błędu
   // poniższa część kodu jest wspólna dla wszystkich trybów pracy - różnią się one tylko
   // 1-szym bajtem komendy
   FLASH_ALE_CLR;      // wyzerowanie sygnału ALE
   FLASH_CE_CLR;       // wybór (załączenie) pamięci FLASH
```

List. 5 - cd.

```
   FLASH_CLE_SET;      // ustawienie sygnału CLE
   FLASH_BASE = command;  // wysłanie do pamięci FLASH bajtu komendy wybranego na
                       // podstawie wartości argumentu OFFSET
   FLASH_CLE_CLR;      // zerowanie sygnału CLE
   FLASH_ALE_SET;      // ustawienie sygnału ALE
   FLASH_BASE = offset;   // zapis bajtu adresu kolumny do pamięci FLASH
   FLASH_BASE = (unsigned char)wPageNum;      // zapis młodszego bajtu numeru strony FLASH
   FLASH_BASE = (unsigned char)(wPageNum>>8); // zapis starszego bajtu numeru strony FLASH
   FLASH_ALE_CLR;      // wyzerowanie sygnału ALE
   WAIT_4_READY;       // oczekiwanie na ustawienie sygnału BUSY
   length += offset;   // dodanie do wartości LENGHT wartości OFFSET, to będzie
                       // adres liniowy w pamięci FLASH
   for (temp = offset; temp < length; temp++)
   {                   // odczyt LENGTH bajtów z pamięci FLASH
       *pBuffer++ = FLASH_BASE;
       WAIT_4_READY;   // po odczycie każdego bajtu czekamy na ustawienie READY
   }
   FLASH_CE_SET;       // wyłączenie pamięci
   return(0);
}

/* zapis danych o rozmiarze pojedynczej strony do pamięci FLASH */
short FLASH_WritePage(WORD wPageNum, char *pBuff)
{
   int temp;
   BYTE status;

   if (!FLASH_READY) return (-1); // pamiec nie gotowa, f.kończy prace i zwraca kod błędu
   FLASH_ALE_CLR;      // zerowanie sygnału ALE
   FLASH_CLE_SET;      // ustawienie sygnału CLE
   FLASH_CE_CLR;       // załączenie układu pamięci FLASH
   FLASH_BASE = 0x80;  // zapis komendy 80H do FLASH-1 (serial input)
   FLASH_ALE_SET;      // ustawienie sygnału ALE
   FLASH_CLE_CLR;      // wyzerowanie sygnału CLE
   FLASH_BASE = 0;     // ustawienie adresu kolumny na wartość 0
   FLASH_BASE = (unsigned char)wPageNum;      // zapis młodszego bajtu numeru strony do FLASH
   FLASH_BASE = (unsigned char)(wPageNum>>8); // zapis starszego bajtu numeru strony do FLASH
   FLASH_ALE_CLR;      // wyzerowanie sygnału ALE
   WAIT_4_READY;       // oczekiwanie na zgłoszenie gotowości przez FLASH
                       // zapis PAGE_SIZE bajtów do pamięci FLASH
   for (temp = 0; temp < PAGE_SIZE; temp++) FLASH_BASE = *pBuff++;
   FLASH_CLE_SET;      // ustawienie sygnału CLE
   FLASH_BASE = 0x10;  // zapis komendy "koniec programowania"
   WAIT_4_READY;       // oczekiwanie na gotowość FLASH
   FLASH_BASE = 0x70;  // zapis komendy "status read", ustawienie FLASH w
                       // "status read mode"
   FLASH_CLE_CLR;      // zerowanie sygnału CLE
   status = FLASH_BASE;       // pobranie bajtu statusu
   FLASH_CE_SET;       // wyłączenie FLASH poprzez ustawienie jej sygnału wyboru
   if (status&1) return (-2); // jeśli ustawiony jest bit FAIL (strona 20 PDF),
                       // funkcja zwraca kod bledu
   return (0);         // w przeciwnym wypadku zwracane jest 0x00
}

/* kasowanie bloku o numerze wBlockNum w pamięci FLASH (auto block erase, PDF str. 13) */
short FLASH_EraseBlock(WORD wBlockNum)
{
    BYTE status;

/* adresy bloków dla pamięci TC58128 podawane sa na liniach od A14 do A23; wymaga to
przesunięcia argumentu wBlockNum o 5 miejsc w lewo; inaczej jest dla pamięci TC5832 i 64; tu
adres podawany jest na liniach A13 do A21 - wymaga to przesunięcia adresu w lewo o 4 miejsca
(PDF strona 16); w tej sytuacji opisywany w PDF "NAND address in block" jest równy 0x00 */
   wBlockNum<<=5;      // przesuniecie dla TC58128
   FLASH_ALE_CLR;      // zerowanie sygnału ALE
   FLASH_CLE_SET;      // zerowanie sygnału CLE
   FLASH_CE_CLR;       // wybór (załączenie) pamięci FLASH
   FLASH_BASE = 0x60;  // zapis instrukcji "erase setup"
   FLASH_ALE_SET;      // ustawienie sygnału ALE
   FLASH_CLE_CLR;      // zerowanie sygnału CLE
   FLASH_BASE = (unsigned char)wBlockNum;         // zapis młodszego bajtu numeru bloku
   FLASH_BASE = (unsigned char)(wBlockNum>>8);    // zapis starszego bajtu numeru bloku
   FLASH_ALE_CLR;      // zerowanie sygnału ALE
   FLASH_CLE_SET;      // zerowanie sygnału CLE
   FLASH_BASE = 0xD0;  // zapis polecenia "erase start"
   FLASH_CLE_CLR;      // zerowanie sygnału CLE
   WAIT_4_READY;       // oczekiwanie na gotowość pamięci NAND
   FLASH_CLE_SET;      // ustawienie sygnału CLE
   FLASH_BASE = 0x70;  // zapis polecenia "status read"
   FLASH_CLE_CLR;      // zerowanie sygnału CLE
   status = FLASH_BASE;   // odczyt bajtu statusu
   FLASH_CE_SET;       // wyłączenie pamięci FLASH poprzez ustawienie sygnału wyboru
   if (status & 1) return (-2);   // jeśli ustawiony jest bit FAIL w słowie statusu, to
                       // funkcja zwraca błąd
   return (0);         // w przeciwnym wypadku zwracane jest 0x00
}
```

przykład antyczne już dziś CP/M czy DOS. To może być dosyć skomplikowane. Jeśli jednak chcemy przechowywać dane w postaci rekordów adresowanych liniowo, czy wreszcie dużych bitmap przechowywanych pod określonym adresem - można to zrobić szybko i efektywnie.

Pamięci - jak łatwo zorientować się z tab. 1 - zasilane są napięciem od 2,7 do 3,6V. Taki wybór zakresu napięć zasilania podyktowany był przede wszystkim przeznaczeniem pamięci do urządzeń przenośnych zasilanych z baterii. W przeszłości firma Toshiba produkowała również pamięci zasilane napięciem 5 V, jednak ich produkcja została zaniechana. Detekcja poziomów logicznych jest zgodna ze standardem CMOS - tak więc stan wysoki na wejściu czy wyjściu pamięci to około 95% wartości napięcia zasilającego (przeciętnie około 3,1 V dla Ucc=3,3 V), a stan niski to napięcie mniejsze niż 0,4 V.

Czasami podłączając TC58 do systemu z mikrokontrolerem należy zbudować dekoder adresów i umieścić ją w obszarze adresowym XDATA przydzielając jej pewien, zależny od rozmiaru pamięci, zakres adresów. Niektóre układy mikrokontrolerów mają dekoder adresowy wbudowany w strukturę, dla innych trzeba go wykonać na układach dyskretnych. Popularny mikrokontroler z rodziny 8051/8052 może również wymagać zastosowania dodatkowego rejestru zapamiętującego młodszą część adresu. Dodatkowego sterowania wymagają również wyprowadzenia pamięci takie jak: RE (*Read Enable*), WE (*Write Enable*), CLE (*Command Latch Enable*), ALE (*Address Latch Enable*) i RY/BY (*Ready/Busy*). Należy również pamiętać o tym, że pamięć nie może być zasilana napięciem wyższym niż 3,6 V, a optymalną wartością jest 3,3 V. O ile w przypadku mikrokontrolerów AVR nie jest to problem, o tyle dla układów z serii 8051 może nim być. Używany we wcześniejszych przykładach programowania AT89S8252 ma minimalne dopuszczalne napięcie zasilające 4 V, w związku z czym konieczny jest układ translacji poziomów napięć.

Na **rys. 5** przedstawiono propozycję podłączenia pamięci Flash TC58128-AFT do mikrokontrolera Atmega8535. Jak widzimy, sprzętowy interfejs pamięci TC58128 (128 Mb) nie nastręcza zbyt wielu kłopotów przy wykonaniu. Szyna danych pamięci jest multipleksowana z szyną adresową. Słowo adresowe, wprowadzane jest w 3 „porcjach" po 8 bitów. Słowo danych ma długość 8 bitów, toteż 8-bitowy AVR nie ma żadnego kłopotu z jego odczytem. Oba układy (mikrokontroler i pamięć Flash) zasilane są z napięcia 3,3 V. Dzięki temu nie jest konieczne wykonywanie translacji poziomów napięć logicznych.

W projektowaniu i implementacji interfejsu programowego pomaga uważna lektura kart katalogowych. Struktura pamięci używanej w przykładzie to 528 bajtów x 32 strony x 1024 bloki. Pamięć posiada wewnętrzny rejestr statyczny o długości 528 bajtów służący jako bufor do wymiany danych podczas operacji zapisu/odczytu. Funkcja kasowania została

Tab. 3. Wybrane pamięci Flash produkcji AMD (obecnie Spansion)

Pojemność	Nazwa	Opis	Obudowa
16Mb	Am29LV160MT/B	z częścią Boot; magistrala danych x8 lub x16; organizacja 1x16kB, 2x8kB, 1x32kB, 15x64kB; 70, 90 i 120ns	48 FBGA, 64 Fortified BGA
16Mb	Am29LV017M	16Mb, magistrala danych x8; jednolity blok; 70, 90 i 120ns	48 FBGA, 40 TSOP/RTSOP
16Mb	Am29LV116M	z częścią Boot; magistrala danych x8 lub x16; organizacja 1x16kB, 2x8kB, 1x32kB, 15x64kB; 70, 90 i 120ns	40 TSOP/RTSOP 26008
32Mb	Am29LV320MH/L	64kB, magistrala danych x8 lub x16; jednolity blok; 90, 100, 110 i 120ns	64 Fortified BGA, 56 TSOP/RTSOP
32Mb	Am29LV320MT/B	z częścią Boot; magistrala danych x8 lub x16; organizacja 64kB (boot 2x8kB na "górze" i "dole" przestrzeni adresowej); 90, 100, 110 i 120ns	48 FBGA, 64 Fortified BGA, 48 TSOP
32Mb	Am29LV033MU	64kB, magistrala danych x8 lub x16; jednolity blok; 90, 100, 110 i 120ns	48 FBGA, 40 TSOP/RTSOP
64Mb	Am29LV640MH/L	64kB, magistrala danych x8 lub x16; jednolity blok; 90, 100, 110 i 120ns	Fortified BGA, 56 TSOP/RTSOP
64Mb	Am29LV640MT/B	z częścią Boot; magistrala danych x8 lub x16; organizacja 64kB (boot 2x8kB na "górze" i "dole" przestrzeni adresowej); 90, 100, 110 i 120ns	48 FBGA, 64 Fortified BGA,48 TSOP
64Mb	Am29LV640MU	32kWord, x16; jednolity blok; 90, 100, 110 i 120ns	63 FBGA, 64 Fortified BGA
64Mb	Am29LV641MH/L	32kWord, x16; jednolity blok; 90, 100, 110 i 120ns	48 TSOP/RTSOP
64Mb	Am29LV065MU	64kB, x8; jednolity blok; 90, 100, 110 i 120ns	63 FBGA, 48 TSOP/RTSOP
128Mb	Am29LV128MH/L	64kB, magistrala danych x8 lub x16; jednolity blok; 90, 100, 110 i 120ns	64 Fortified BGA, 56 TSOP/RTSOP
256Mb	Am29LV256MH/L	64kB, magistrala danych x8 lub x16; jednolity blok; 90, 100, 110 i 120ns	64 Fortified BGA, 56 TSOP/RTSOP
512Mb	Am29LV512NH/L	128kB, magistrala danych x8 lub x16; jednolity blok; 90, 100, 110 i 120ns	64 Fortified BGA, 56 TSOP/RTSOP

zaimplementowana dla pojedynczego bloku (528 bajtów x 32 strony). Tutaj słowo wyjaśnienia. Komórka pamięci Flash przed zapisem wymaga, aby wszystkie bity słowa lub co najmniej te, które mają mieć po zapisie wartość logiczną „1“, były ustawione. W tym celu każda z pamięci Flash ma zaimplementowaną funkcję sprzętowego kasowania, najczęściej bloku lub strony, powodującą ustawienie wszystkich bitów w zadanym obszarze na wartość „1“. Pamiętajmy - operacja zapisu musi być poprzedzona kasowaniem!

Każda z pamięci TC58 posiada wbudowany w strukturę kontroler zdolny do interpretacji i wykonywania prostych rozkazów. On jest odpowiedzialny za kasowanie bloku oraz za właściwą interpretację wprowadzanego bloku danych. Lista realizowanych poleceń nie jest zbyt obszerna, jednak do pełnego zrozumienia wymaga lektury sekwencji czasowych. Nie umieszczę ich w tym artykule ze względu na szczupłość miejsca - proponuję sięgnąć do karty katalogowej.

W **tab. 2** znajduje się uproszczony wykaz dostępnych rozkazów. Przykładową implementację interfejsu programowego do obsługi pamięci Flash typu TC58128 w języku C zawiera **list. 5**.

Perspektywy

Znając tempo rozwoju pamięci Flash sądzę, że od momentu napisania do ukazania się tego artykułu, pojawią się na rynku nowe produkty, a pewne informacje staną się nieaktualne. Moim zdaniem, pamięci Flash zastąpią dyski twarde w komputerach PC. Pierwsze produkty tego typu można już kupić w sklepach, jednak dyski Flash o dużej pojemności są ciągle zbyt drogie w porównaniu z ich tradycyjnymi odpowiednikami. Sądzę, że to jednak tylko kwestia czasu i opracowania odpowiednio tanich technologii. Ogromne zapotrzebowanie na pamięci typu Flash jest powodem, dla którego stale są one udoskonalane. Nie mam tu na myśli bynajmniej tylko komputerów PC, ale również różne nowoczesne „zabawki“: telefony komórkowe, komputery typu PalmTop, kamery wideo i cyfrowe aparaty fotograficzne oraz wiele, wiele innych. Wystarczy przejrzeć strony internetowe czy katalogi producentów podzespołów. Na przykład firma AMD (obecnie Spansion) wprowadziła ofertę podzespołów pod wspólną nazwą MirrorBit. Można w niej znaleźć pamięci Flash zasilane napięciem od 1,8 do 5 V o pojemnościach od 16 do 256 Mb (**tab. 3**).

Firma STM wprowadziła linię produktów pod wspólną nazwą LightFlash (**tab. 4**). Podobne działania prowadzą również inni producenci, jak na przykład Winbond, Atmel czy Renesas - prześcigając się w miniaturyzacji cel oraz liczbie cykli zapisu/odczytu. Przyszłość pamięci Flash zapowiada się bardzo ciekawie.

Jacek Bogusz, AVT
jacek.bogusz@ep.com.pl

Tab. 4. Wybrane pamięci Flash produkcji firmy STM

Pojemność	Nazwa	Opis	Obudowa
16Mb	M29KW016E	16Mb (x16), 90ns, jednolity blok	TSOP48, SO44, TFBGA48
32Mb	M29KW032E	32Mb (x16), 90ns, jednolity blok	TSOP48, TFBGA48
64Mb	M29KW064E	32Mb (x16), 90ns, jednolity blok	TSOP48, TFBGA48