Obsługa kart pamięci FLASH za pomocą mikrokontrolerów, część 4
Karty MultiMedia Card (MMC)

W tej części kursu zajmiemy się drugim rodzajem opisywanych kart – kartami MultiMedia Card, czyli MMC.

W porównaniu do wcześniej opisywanych kart CF charakteryzują się one dużo mniejszymi wymiarami wynoszącymi tylko 32x24 mm przy grubości wynoszącej jedynie 1,4 mm oraz wagą nie przekraczającą 1,5 grama (**fot. 1**). Karty MultiMedia Card były do niedawna najmniejszymi z ogólnie dostępnych nośników pamięci. Rozmiarami przypominają znaczek pocztowy.

Standard MMC stworzono w roku 1997 przy współpracy firm SanDisk Corporation, Siemens AG and Infineon Technologies AG. W chwili obecnej dostępne są karty MMC o pojemności do 512 MB, a wkrótce oczekiwane są modele o pojemności 1 GB. Karty MMC mogą być zasilane napięciem od 2,7 do 3,6 V i nie mogą być bezpośrednio dołączane do systemów zasilanych napięciem 5 V. Nie stanowi to jednak zbyt wielkiego problemu, ponieważ urządzenia przenośne, w których najczęściej stosowane są tego typu karty są przeważnie zasilane napięciem 3,3 V, a jeśli zechcemy zastosować taką kartę we własnych konstrukcjach zasilanych napięciem o wartości 5 V, to zawsze można zastosować odpowiedni stabilizator napięcia zasilania oraz konwerter poziomów logicznych z 5 na 3,3 V.

Pobór prądu przez karty MMC jest mniejszy niż dla kart CF i wynosi typowo poniżej 40 mA przy zapisie, 30 mA przy odczycie i około 50 µA w trybie gotowości.

Interfejs

Karty MMC posiadają proste, płaskie 7-stykowe złącze. Do komunikacji karty z hostem wykorzystuje się szeregową transmisję danych. W zależności od użytego protokołu komunikacji – o czym piszę dalej – poszczególne styki karty mają nieco inne funkcje (**tab. 1**). Styk karty oznaczony numerem 1 znajduje się od strony ścięcia na obudowie karty.

W trybie MultiMedia Card obie linie po których przesyłane są dane pracują w trybie dwukierunkowym. Linia CMD może być wejściem lub wyjściem pracującym w trybie „otwarty dren" lub w trybie *push-pull*. Linia DAT pracuje jako wejście lub wyjście w trybie *push-pull*. W trybie SPI wszystkie linie są jednokierunkowe, co znacznie ułatwia stosowanie wszelkiego rodzaju konwerterów poziomów logicznych, oraz upraszcza procedury komunikacyjne. W trybie MMC możliwe jest bezpośrednie i jednoczesne podłączenie do jednej trójprzewodowej magistrali aż 65535 kart, a wybór aktywnej karty odbywa się przy pomocy mechanizmów czysto programowych zaimplementowanych w protokół komunikacji. W przypadku trybu SPI do jednej magistrali możemy również dołączyć wiele kart, lecz każda z nich wymaga podłączenia oddzielnego sygnału CS adresującego wybraną kartę. Ze względu na znaczne uproszczenie komunikacji, w dalszej części artykułu zajmę się głównie trybem SPI.

Fot. 1

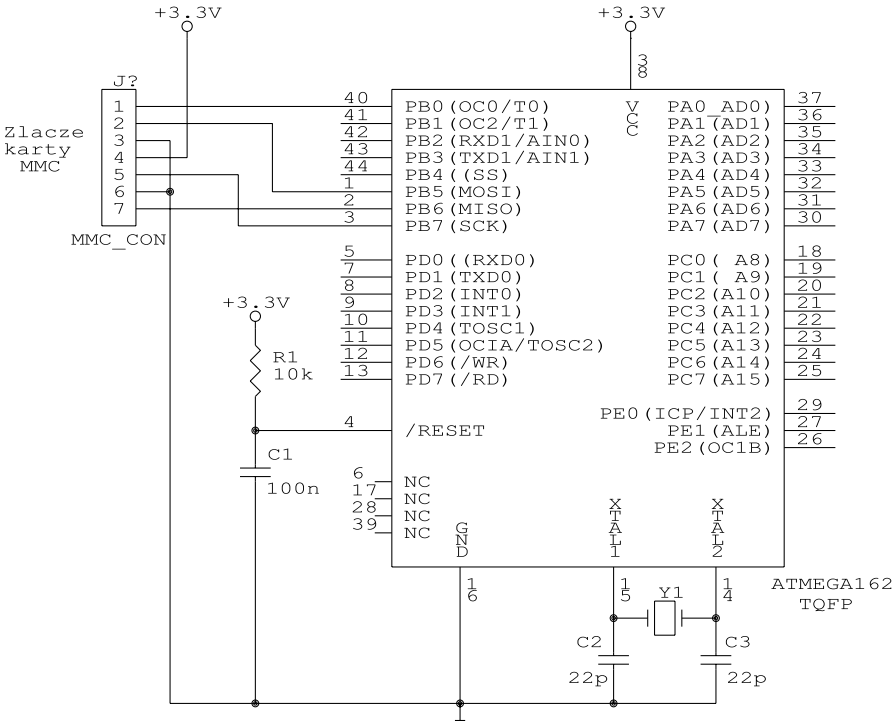
Podłączenie karty MMC do mikrokontrolera

Wbudowany w karty typu MMC kontroler może komunikować się z hostem w dwóch różnych trybach transmisji danych. Pierwszy z nich – zwany *MultiMedia Card Mode* jest podstawowym trybem pracy kart MMC i udostępnia wszystkie komendy zdefiniowane w standardzie MMC. Drugi – nieco okrojony tryb komunikacji jest sprzętowo kompatybilny z popularnym protokołem synchronicznej, szeregowej magistrali SPI, którego sprzętowy interfejs posiada wiele mikrokontrolerów jednoukładowych. Z tego też względu najłatwiej jest podłączyć kartę MMC do wbudowanego w mikrokontroler interfejsu SPI i wykorzystać ten tryb pracy karty. Jeśli mikrokontroler nie posiada wbudowanego interfejsu SPI, to można wykorzystać do komunikacji z kartą trzy linie portów wejścia-wyjścia, i w prosty sposób napisać procedury programowego interfejsu SPI typu Master, jakoż że karta zawsze działa jako urządzenie typu Slave.

Na **rys. 2** pokazano najprostszy sposób podłączenia karty MMC do mikrokontrolera AtMega162 zasilanego napięciem od 2,7 do 3,6 V. Wybrałem ten typ mikrokontrolera ze względu na to, że posiada dużo wewnętrznej pamięci RAM co umożliwia zapisanie całego sektora karty (512 B) naraz do pamięci. Jak widać sam układ jest banalnie prosty, ponieważ karta jest dołączona bezpośrednio do linii interfejsu SPI

Tab. 1. Funkcje styków kart MMC

Numer styku	Tryb MultiMedia Card		Tryb SPI	
	Nazwa	Opis	Nazwa	Opis
1	RSV	Nie używany lub podłączony do VDD	CS	Chip Select aktywny „0" logicznym
2	CMD	Komendy/Odpowiedzi – linia dwukierunkowa	DataIn	Komendy/Dane z hosta do karty
3	VSS1	Masa	VSS1	Masa
4	VDD	Zasilanie 2,7...3,6 V	VDD	Zasilanie 2,7...3,6 V
5	CLK	Sygnał zegarowy transmisji	CLK	Sygnał zegarowy transmisji
6	VSS2	Masa	VSS2	Masa
7	DAT[0]	Dane – linia dwukierunkowa	DataOut	Dane/Status z karty do hosta



Rys. 2. Schemat podłączenia karty MMC do mikrokontrolera AVR

mikrokontrolera (sygnały MOSI, MISO i SCK), oraz dodatkowo wykorzystano port PB0 jako sygnał *Chip Select* dla karty. W tym miejscu wypada dodać że złącze jak i karty MMC zostały zaprojektowane w ten sposób, że umożliwiają tak zwany *Hot Swap*, czyli wkładanie i wyjmowanie karty przy załączonym zasilaniu całego urządzenia. W tym celu styki złącza MMC doprowadzające zasilanie do karty są nieco bardziej wysunięte i zapewniają wcześniejszy kontakt z polami karty niż pozostałe styki.

Rejestry karty MMC

Karty MMC posiadają 6 rejestrów specjalnych zawierających informacje o karcie, jej bieżącą konfigurację, aktualny adres karty (przy wykorzystaniu protokołu MMC) oraz rejestr statusu. Każdy z rejestrów ma inną wielkość, więc nazwa *rejestr*, która kojarzy się nam z jakąś konkretną wielkością (8, 16 czy więcej bitów), nie jest tu zbyt adekwatna i raczej należało by je nazwać *strukturami*. Ponieważ w ogólnodostępnych opisach przyjęto nazwę *rejestr*, będę się tej terminologii trzymał w artykule. W przypadku komunikacji z kartą w trybie SPI dostępne są tylko dwa z nich, więc opiszę je szczegółowo.

Rejestr CID – *Card Identification Register*

Rejestr ten zawiera dane o producencie, nazwie, wersji oraz typie danej karty. Ma on długość 128 bitów, czyli mieści się w 16 bajtach danych. Zawiera on dane tylko do odczytu. W **tab**. **2** podano szczegółowe informacje o znaczeniu poszczególnych pól rejestru CID.

Rejestr CSD – *Card Specific Data*

Rejestr ten zawiera wszystkie dane o konfiguracji danej karty niezbędne do prawidłowej komunikacji z kartą. Posiada on zarówno pola tylko do odczytu – czyli wartości ustalane przez producenta karty, jak i pola, których zawartość może być zmieniana przez użytkownika przy pomocy specjalnej komendy. Niektóre pola są jednokrotnie programowalne, tak wiec po jednokrotnej zmianie nie jest możliwy powrót do poprzedniej wartości. Przykładem może być pole *Permanent Write Protection*, które umożliwia zablokowanie możliwości zapisu na kartę, przez co karta staje się jakby kartą pamięci typu ROM bez możliwości dokonania jakichkolwiek zmian. W **tab**. **3** podano szczegółowe informacje o znaczeniu poszczególnych pól rejestru CSD.

Z wielu informacji zawartych w poszczególnych polach rejestru CSD dokładniej opisze tylko kilka z nich, które będą potrzebne do prawidłowej komunikacji z kartą lub zawierają informacje przydatne dla naszych celów. I tak kolejno zaczniemy od czasu dostępu do danych na karcie określonego polami TAAC, NSAC i R2W_FACTOR.

Pole TTAC określa składnik czasu dostępu do danych na karcie podczas odczytu, niezależny od częstotliwości przebiegu zegarowego podanego na linie CLK karty. W bitach od 0 do 2 tego pola zakodowany jest wykładnik czasu dostępu i wynosi odpowiednio: 0=1 ns, 1=10 ns, 2=100 ns, 3=1 µs, 4=10 µs, 5=100 µs, 6=1 ms, 7=10 ms. W bitach od 3 do 6 tego pola zakodowana jest podstawa czasu która wynosi odpowiednio: 1=1,0, 2=1,2, 3=1,3, 4=1,5, 5=2,0, 6=2,5, 7=3,0, 8=3,5, 9=4,0, 10=4,5, 11=5,0, 12=5,5, 13=6,0, 14=7,0, 15=8,0. Bit 7 tego pola jest zarezerwowany. Przykładowo, jeśli odczytamy wartość 0x26 z pola TAAC, oznacza to że czas dostępu wynosi 1,5 * 1 ms czyli 1,5 milisekundy.

Ale na tym nie koniec, ponieważ istnieje jeszcze pole NSAC które określa dodatkowo składnik czasu dostępu zależny od częstotliwości przebiegu zegarowego. Jednostką jest tu 100 cykli zegarowych, a więc wartość pola NSAC wynosząca 0x04 oznacza, że do czasu odczytanego z TAAC należy jeszcze dodać czas generowania 400 cykli zegarowych, a będzie on zależał od częstotliwości na linii CLK czyli inaczej od prędkości przesyłu danych. Tyle na temat czasu dostępu przy odczycie.

Pole R2W_FACTOR określa mnożnik przez który trzeba pomnożyć otrzymany czas dostępu przy odczycie, aby uzyskać czas zapisu jednego bloku na kartę. Wartości tego mnożnika są na-

Tab. 2. Znaczenie poszczególnych bitów rejestru CID

Nazwa	Typ pola	Wielkość w bitach	Pozycja	Opis
Manufacturer ID	Binarny	24	[127–104]	Numer Identyfikacji producenta nadany przez MMCA
Product name	Tekstowy	56	[103–48]	Nazwa karty w postaci jawnego tekstu
HW Revision	Binarny	4	[47–44]	Wersja sprzętu karty
FW Revision	Binarny	4	[43–40]	Wersja oprogramowania karty (wewnętrznego kontrolera)
Serial Number	Binarny	24	[39–16]	Unikalny numer seryjny karty
Month code	Binarny	4	[15–12]	Data produkcji – miesiąc
Year code	Binarny	4	[11–8]	Data produkcji – rok w postaci offsetu od roku 1997
CRC7 checksum*	Binarny	7	[7–1]	Wyliczona suma kontrolna danych rejestru CID
Not used		1	[0]	Dopełnienie do 128 bitów czyli do 16 bajtów (zawsze „1”)

Tab. 3. Znaczenie poszczególnych bitów rejestru CSD

Nazwa pola	Wielkość w bitach	Pozycja	Typ komórki	Opis
CSD_STRUCTURE	2	[127–126]	R	Wersja struktury rejestru CSD
MMC_PROT	4	[125–122]	R	Wersja protokołu MMC
–	2	[121–120]	R	Zarezerwowany
TAAC	8	[119–112]	R	Czas dostępu przy odczycie – część niezależna od zegara
NSAC	8	[111–104]	R	Czas dostępu przy odczycie – część zależna od zegara
TRAN_SPEED	8	[103–96]	R	Maksymalna szybkość transmisji (zegara)
CCC	12	[95–84]	R	Klasa komend
READ_BL_LEN	4	[83–80]	R	Maksymalna długość bloku przy odczycie
READ_BL_PARTIAL	1	[79]	R	Czy możliwy odczyt fragmentu bloku
WRITE_BLK_MISALGIN	1	[78]	R	
READ_BLK_MISALGIN	1	[77]	R	
DSR_IMP	1	[76]	R	Czy zaimplementowano DSR
–	2	[75–74]	R	Zarezerwowany
C_SIZE	12	[73–62]	R	Pojemność karty
VDD_R_CURR_MIN	3	[61–59]	R	Max. pobór prądu przy odczycie przy minimalnym VDD
VDD_R_CURR_MAX	3	[58–56]	R	Max. pobór prądu przy odczycie przy maksymalnym VDD
VDD_W_CURR_MIN	3	[55–53]	R	Max. pobór prądu przy zapisie przy minimalnym VDD
VDD_W_CURR_MAX	3	[52–50]	R	Max. pobór prądu przy zapisie przy maksymalnym VDD
C_SIZE_MULT	3	[49–47]	R	Mnożnik pojemności karty
SECTOR_SIZE	5	[46–42]	R	Wielkość kasowanego sektora
ERASE_GRP_SIZE	5	[41–37]	R	Wielkość kasowanych grup sektorów
WP_GRP_SIZE	5	[36–32]	R	Wielkość grupy sektorów które można zabezpieczyć przed zapisem
WP_GRP_ENABLE	1	[31]	R	Czy możliwe jest zabezpieczenie grupy przed zapisem
DEFAULT_ECC	2	[30–29]	R	Fabrycznie ustawiony typ korekcji błędów
R2W_FACTOR	3	[28–26]	R	Stosunek czasu dostępu do czasu zapisu (mnożnik)
WRITE_BL_LEN	4	[25–22]	R	Maksymalna długość bloku przy zapisie
WRITE_BL_PARTIAL	1	[21]	R	Czy możliwy zapis fragmentu bloku
–	5	[20–16]	R	Zarezerwowane
–	1	[15]	R/W	Zarezerwowany
COPY	1	[14]	R/W	Flaga określająca kopię
PERM_WRITE_PROTECT	1	[13]	R/W	Stała blokada zapisu na kartę
TMP_WRITE_PROTECT	1	[12]	R/W/E	Tymczasowa blokada zapisu na kartę
–	2	[11–10]	R/W	Zarezerwowany
ECC	2	[9–8]	R/W/E	Typ korekcji błędów
CRC	7	[7–1]	R/W/E	Wyliczona suma kontrolna danych rejestru CSD
–	1	[0]	–	Zarezerwowany

Typ Komórki:
R – tylko do odczytu
R/W – odczyt i jednokrotny zapis
R/W/E – dowolnie zmieniana

stępujące: 0=1, 1=2, 2=4, 3=8, 4=16, 5=32, 6 i 7 – zarezerwowane. Tak więc jeśli pole R2W_FACTOR ma wartość 4, oznacza to, że czas zapisu bloku jest 16-krotnie większy niż czas dostępu określony polami TAAC i NSAC.

Kolejnym polem mającym związek z szybkością karty jest pole TRAN_SPEED określające maksymalną częstotliwość przebiegu zegarowego na linii CLK, czyli inaczej: szybkość przesyłu danych. Podobnie jak w przypadku TAAC, pole to składa się z podstawy i wykładnika określających częstotliwość. I tak, bity od 0 do 2 oznaczają wykładnik, którego wartość wynosi: 0=100 kb/s, 1=1 Mb/s, 2=10 Mb/s, 3=100 Mb/s, 4...7=zarezerwowane. Wartości podstawy zakodowane w bitach od 3 do 6 tego pola są identyczne jak podstawy w polu TAAC czyli od 1.0 dla wartości 1 do 8.0 dla wartości równej 15. Bit 7 pola TRAN_SPEED jest zarezerwowany. I znowu przykładowo jeśli odczytana wartość pola wynosi 0x2A, to oznacza że dana karta może pracować z szybkością do 20 Mb/s czyli częstotliwość sygnału zegarowego może wynosić maksymalnie 20 MHz.

Pole READ_BL_LEN określa maksymalną długość pojedynczego bloku danych możliwych do odczytania jedną komendą odczytu. Długość tą obliczamy jako $2^{READ\_BL\_LEN}$, a dopuszczalny zakres tego pola wynosi od 0 do 11. Oznacza to że długość bloku może wynosić od 1 bajtu (READ_BL_LEN = 0) do 2048 bajtów (READ_BL_LEN =11). Typowa wartość tego pola to 9, czyli blok o długości 512 bajtów. Te same zasady tyczą się pola WRITE_BL_LEN, przy czym w tym przypadku dotyczą komendy zapisu bloku na kartę.

Pola VDD_R_CURR_MIN, VDD_R_CURR_MAX, VDD_W_CURR_MIN, VDD_W_CURR_MAX określają typowy pobór prądu przez kartę podczas odczytu i zapisu, przy minimalnym (pola z MIN) i maksymalnym (pola z MAX) napięciu zasilania karty. Pola te mają wielkość 3 bitów a więc można określić 8 różnych wartości. I tak przy odczycie poszczególne wartości prądów wynoszą: 0=0,5 mA, 1=1 mA, 2=5 mA, 3=10 mA, 4=25 mA, 5=35 mA, 6=60 mA, 7=100 mA. Pola dotyczące zapisu mają nieco inaczej zakodowane wartości: 0=1 mA, 1=5 mA, 2=10 mA, 3=25 mA, 4=35 mA, 5=45 mA, 6=80 mA, 7=200 mA.

W końcu dochodzimy do najważniejszej rzeczy, czyli do pojemności karty. Nie wiadomo czemu jest ona zakodowana w dość dziwny i zawiły sposób, i żeby ją uzyskać musimy się posłużyć wartościami aż trzech pól: C_SIZE, C_SIZE_MULT, oraz READ_BL_LEN. Zacznijmy od obliczenia pojemności karty w blokach, czyli od ilości bloków na karcie:

$$BLOCKNR = (C\_SIZE + 1) * 2^{(C\_SIZE\_MULT + 2)}$$

Skomplikowane, prawda? Ale nie szkodzi, bo jak się okaże podczas opisywania procedur komunikacji – dość proste do obliczenia. Stąd już tylko krok do poznania pojemności karty wyrażonej w bajtach, bo obliczoną ilość bloków BLOCKNR wystarczy pomnożyć przez odczytaną z READ_BL_LEN długość pojedynczego bloku. Oczywiście ta długość to wartość otrzymana po rozszyfrowaniu pola READ_BL_LEN.

Na koniec tych skomplikowanych obliczeń dodam że maksymalna pojemność karty zakodowana w ten sposób wynosi 4096 * 512 * 2048 bajtów co daje 4 gigabajty.

Jako że w trybie SPI mamy dostęp jedynie do tych dwóch rejestrów karty, mogę na tym zakończyć ich opis. W następnej części kursu zajmiemy się protokołem komunikacji oraz opisem komend dostępnych w trybie SPI.

Romuald Biały