Emulacja pamięci EEPROM z wykorzystaniem kart MMC

Coraz tańsze i coraz bardziej popularne karty pamięci nieulotnej Flash najczęściej są wykorzystywane w aparatach cyfrowych oraz odtwarzaczach plików MP3. Ich niska cena zachęca też do stosowania tego rodzaju pamięci we własnych aplikacjach. Najchętniej wykorzystywane są do tego karty o mniejszej pojemności (poniżej gigabajta). Oprócz niskiej ceny zachętą do stosowania kart Flash jest również fakt, że do ich obsługi nie jest wymagana duża wiedza.

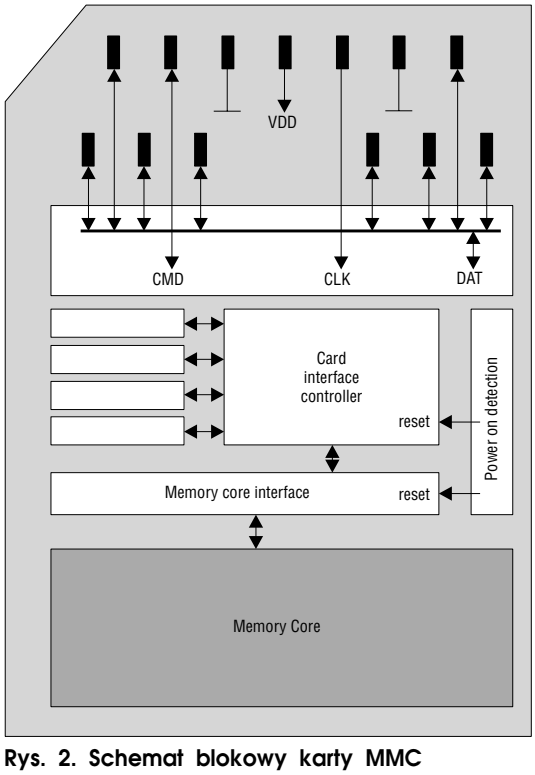
Najprostszymi do wykorzystania są karty MMC (*Multimedia Card*), które oprócz pracy w trybie MMC mogą pracować również w trybie komunikacji z wykorzystaniem szeregowego interfejsu SPI. W interfejs taki jest wyposażony aktualnie prawie każdy mikrokontroler. Wygląd przykładowej karty MMC pokazano na **fot. 1**. Również popularne są karty Flash typu CF (*Card Flash*), ale do komunikacji z nimi wymagana jest spora liczba linii portów mikrokontrolera.

W artykule zajmiemy się wyłącznie kartami MMC. Można je wykorzystać we własnych aplikacjach na wiele sposobów. Jedną z zalet kart MMC jest możliwość zaimplementowania własnych systemów plików lub standardowych: FAT16 lub FAT32. Można je również wykorzystywać bez jakiejkolwiek implementacji systemu plików. W większości urządzeń do zapamiętywania danych służą pamięci EEPROM, których pojemność nie jest zaskakująco duża. W przypadku większych ilości danych, do zapamiętania można wykorzystać pamięci DataFlash lub wymienione karty Flash (np. typu MMC). W przypadku pamięci EEPROM można w nich zarówno zapisać, jak i odczytać daną spod dowolnie wysłanego do pamięci adresu komórki. W przypadku pamięci DataFlash oraz kart Flash, dane są zapisywane i odczytywane w postaci bloku danych. Czas zapisywania jest równy dla każdego bloku. W przypadku kart MMC długości jednego bloku wynosi 512 bajtów. W artykule zostanie pokazany sposób emulacji pamięci EEPROM z wykorzystaniem karty MMC. Będzie więc możliwy łatwy odczyt i zapis danej spod dowolnego adresu nie przekraczającego pojemności wykorzystywanej karty. Jedyną wadą takiego rozwiązania będzie długi czas zapisu pojedynczej danej, który będzie równy czasowi zapisu jednego bloku danych. Z drugiej strony ułatwiony będzie jednak dostęp do dowolnej danej zapisanej na karcie. Kartę taką można traktować i wykorzystywać jak pamięć EEPROM o dużej pojemności. Do komunikacji z kartą MMC wykorzystano mikrokontroler ATmega128, a przykładową aplikację przygotowano w środowisku Bascom AVR. Emulację pamięci EEPROM można wykorzystać z zastosowaniem tylko takich mikrokontrolerów, które posiadają około 1 kB pamięci RAM, z której 512 bajtów przeznaczono na bufor wymiany danych. Zanim zostanie pokazany gotowy przykład, przypomnimy kilka informacji o kartach MMC i sposobach komunikacji z nimi z wykorzystaniem trybu SPI.

Fot. 1. Widok karty MMC

Komunikacja z kartą MMC w trybie SPI

Pojemność kart Flash ciągle jest zwiększana i już są dostępne karty MMC o pojemności kilkunastu gigabajtów. Na **rys. 2** pokazano schemat blokowy typowej karty MMC. Mogą one być zasilane napięciem od 2,7 V do 3,6 V, przy czym nie można ich dołączyć bezpośrednio do mikrokontrolerów zasilanych napięciem +5 V. Przy zasilaniu mikrokontrolera napięciem +5 V, należy dodatkowo zastosować odpowiedni konwerter poziomów logicznych. Komunikacja z kartą MMC odbywa się w sposób szeregowy, z wykorzystaniem trybu pracy MMC lub SPI. W przypadku współpracy karty z mikrokontrolerem najodpowiedniejszy będzie tryb pracy w wykorzystaniem interfejsu SPI. Jeśli w stosowanych mikrokontrolerach nie ma sprzętowego interfejsu SPI, można wykorzystać do tego celu jego programową emulację. Choć w trybie SPI nie ma dostępu do komend zdefiniowanych w standardzie MMC, to jest on bardzo prosty w zastosowaniu z mikrokontrolerem i zarazem w większości przy-



Rys. 2. Schemat blokowy karty MMC

Tab. 1. Funkcje styków karty MMC w trybie SPI

Numer styku	Nazwa	Opis
1	\CS	Sygnał wyboru układu
2	DI	Wejście danych
3	VSS1	Masa
4	VDD	Zasilanie 2,7...3,6 V
5	CLK	Sygnał zegarowy
6	VSS2	Masa
7	DO	Wyjście danych

padków wystarczający. Na **rys. 3** pokazano rozmieszczenie styków karty MMC. W trybie pracy SPI karty MMC wykorzystywane są tylko styki o numerach 1...7. Pozostałe styki są wykorzystywany przy pracy karty w trybie MMC, w którym linie po których są wysyłane dane pracują dwukierunkowo. W **tab. 1** pokazano funkcje styków karty w dla pracy w trybie SPI. Karty MMC posiadają kilka rejestrów specjalnych zawierających informacje o karcie. W trybie SPI jest dostęp tylko do trzech takich rejestrów. Dwa z nich składają się z 16 bajtów, czyli 128 bitów. Pierwszym z dostępnych rejestrów jest rejestr CID (*Card Identification Register*), który zawiera informacje o producencie, nazwie karty, numerze seryjnym oraz dacie produkcji. Są to dane tylko do odczytu. W aplikacji przykładowej, z tego rejestru odczytywana jest nazwa karty zapisana tekstowo na bitach 103...156. Zazwyczaj rejestr ten nie będzie wykorzystywany we własnych aplikacjach, jeśli nie będą potrzebne informacje o karcie MMC. Dokładne informacje o znaczeniu bitów rejestru CID można znaleźć w dokumentacji standardu MMC. Dużo ważniejszy jest drugi dostępny rejestr CSD (*Card Specific Data*), w którym są zawarte dane o konfiguracji danej karty i związanej z nią komunikacji. Niektóre dane tego rejestru można odczytywać i zapisywać. Z informacji zawartych w rejestrze CSD (16 bajtów) w przykładowej aplikacji zostały wykorzystane tylko te, na podstawie których możliwe jest obliczenie pojemności karty. Wartość pojemności karty będzie adresem końca danych. W **tab. 2** pokazano znaczenie bitów rejestru CSD na podstawie których jest możliwe obliczenie pojemności karty. Znaczenie pozostałych bitów rejestru CSD można znaleźć w specyfikacji standardu MMC. Pole *READ_BL_LEN* określa maksymalną długość pojedynczego bloku danych. Długość bloku podczas zapisu jest stała i wynosi 512 bajtów. Również wielkość bloku danych do odczytu jest identyczna jak wielkość bloku do zapisu. Pole *READ_BL_LEN* składa się z 4 bitów, a dopuszczalny zakres wartości tego pola wynosi od 0 do 11. Długość bloku jest obliczana jako $2^{READ\_BL\_LEN}$, czyli maksymalna długość bloku danych może wynosić 2 kB. Pojemność karty można obliczyć korzystając z następującego wzoru:

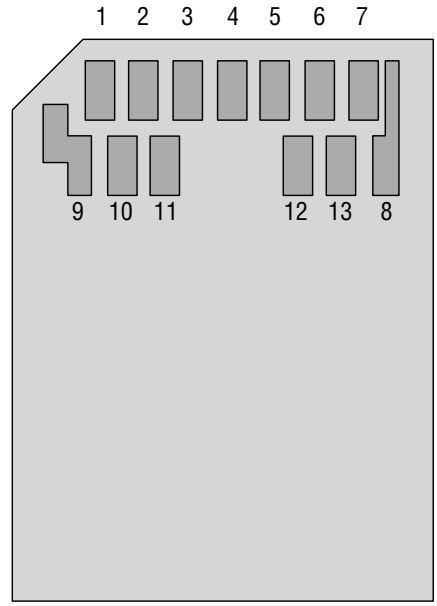
*Pojemność (w bajtach) = Liczba_Bloków * Długość_Bloku*

gdzie:

*Liczba_Bloków = (C_SIZE+1) * MULT*

$MULT = 2^{(C\_SIZE\_MULT+2)}$

Obliczenie pojemności karty wbrew pozorom nie jest trudne. Maksymalna pojemność karty zakodowana w ten sposób może wynosić 4 GB. Trzeci rejestr nazywany OCR składa się z 4 bajtów i zawiera informacje miedzy innymi o zakończeniu włączenia karty. Nie jest on wykorzystywany w przykładowej aplikacji. Komunikacja z kartą MMC w trybie SPI nie jest skomplikowana i będzie się rozpoczynać od podania na linię CS stanu niskiego. Po aktywacji karty sygnałem CS, należy wysłać 6–bajtową komendę, w skład której wchodzi kod komendy, argument i suma kontrolna CRC. Domyślnie karta MMC pracująca w trybie SPI ma wyłączone sprawdzanie sumy CRC, więc nie jest potrzebne jej obliczanie. Gdyby w aplikacji, w której pracuje karta MMC zależało na poprawności transmisji danych, należy włączyć sprawdzanie sumy kontrolnej. Wtedy na podstawie komendy i jej argumentów należy tę sumę obliczać. Wyłączenie sprawdzania sumy kontrolnej uprasza cały program, a przy małych prędkościach transmisji, komunikacja karty z procesorem będzie zawsze poprawna. Po wysłaniu komendy, należy odebrać potwierdzenie, które w zależności od komendy może być 1– lub 2––bajtowe. W zależności od komendy zależeć będzie, czy wysyłany lub odbierany jest blok danych, który rozpoczyna się bajtem startu o wartości &HFE. W przypadku, gdyby komenda odczytu paczki danych nie została wykonana poprawnie, karta zwraca 1–bajtowy kod błędu.



Rys. 3. Rozmieszczenie styków karty MMC

W przypadku zapisu bloku danych do karty należy czekać na zakończenie tej operacji sprawdzając stan zajętości karty (BUSY). Po zakończeniu komunikacji z kartą należy podać na linię CS stan wysoki i wysłać do niej bajt danych o wartości &HFF. Jak wiemy nie wszystkie dostępne w standardzie MMC komendy są dostępne w trybie SPI. W **tab. 3** pokazano komendy karty MMC jakie można stosować w trybie SPI. W przykładowej aplikacji jest wykorzystywanych tylko kilka z nich. Każda komenda składa się z 6 bajtów. Pierwszy bajt określa rodzaj komendy, przy czym 6 bit musi tu być zawsze ustawiony na 1. Oznacza to, że do numeru komendy z tab. 3 zawsze należy dodać wartość 64. Bajty od 2 do 5 komendy zawierają jej argument. Jeśli komenda nie posiada argumentu, należy wysłać 4 bajty o wartości &H00. Ostatni bajt komendy zawiera sumę kontrolną CRC, która nie jest wykorzystywana w przykładowej aplikacji i może mieć dowolną wartość. Reguła ta nie dotyczy

Tab. 2. Znaczenie wybranych bitów rejestru CSD

Nazwa pola	Bity	Typ	Opis
READ_BL_LEN	83...80	R	Długość bloku danych przy odczycie
C_SIZE	73…62	R	Pojemność karty
C_SIZE_MULT	49…47	R	Mnożnik pojemności karty
R – tylko do odczytu			

Tab. 3. Komendy dostępne w trybie pracy SPI karty MMC

Numer komendy	Argument	Odpowiedź	Nazwa komendy	Opis
CMD0	Brak	R1	GO_IDLE_STATE	Zerowanie karty MMC
CMD1	Brak	R1	SEND_OP_COND	Inicjacja karty MMC
CMD9	Brak	R1	SEND_CSD	Odczyt rejestru CSD
CMD10	Brak	R1	SEND_CID	Odczyt rejestru CID
CMD12	Brak	R1	STOP_ TRANSMISSION	Komenda zatrzymania wielokrotnego odczytu bloków
CMD13	Brak	R2	SEND_STATUS	Odczyt statusu karty
CMD16	[31...0] Długość bloku	R1	SET_BLOCKLEN	Ustawienie długości bloku danych
CMD17	[31...0] Adres danych	R1	READ_SINGLE_ BLOCK	Odczyt bloku danych o długości ustawionej komendą CMD16
CMD18	[31...0] Adres danych	R1	READ_MULTIPLE_ BLOCK	Komenda wielokrotnego odczytu bloków danych
CMD24	[31...0] Adres danych	R1	WRITE_BLOCK	Zapis bloku danych do karty
CMD25	[31...0] Adres danych	R1	WRITE_MULTIPLE_ BLOCK	Komenda wielokrotnego zapisu danych do karty
CMD27	Brak	R1	PROGRAM_CSD	Programowanie możliwych do zapisu bitów rejestru CSD
CMD28	[31...0] Adres danych	R1b	SET_WRITE_PROT	Komenda włącza zabezpieczenie przed zapisem dla danej grupy bloków
CMD29	[31...0] Adres danych	R1b	CLR_WRITE_PROT	Komenda wyłącza zabezpieczenie przed zapisem dla danej grupy bloków
CMD30	[31...0] adres zabezpieczonych danych	R1	SEND_WRITE_PROT	Odczyt stanu zabezpieczeń karty przed zapisem
CMD32	[31...0] Adres danych	R1	TAG_SECTOR_START	Ustawia adres pierwszego bloku danych do skasowania
CMD33	[31...0] Adres danych	R1	TAG_SECTOR_END	Ustawia adres ostatniego bloku danych lub adres pojedynczego bloku do skasowania
CMD34	[31...0] Adres danych	R1	UNTAG_SECTOR	Anuluje wybrany sektor z grupy do skasowania
CMD35	[31...0] Adres danych	R1	TAG_ERASE_GROUP _START	Ustawia adres pierwszej grupy do skasowania
CMD36	[31...0] Adres danych	R1	TAG_ERASE_GROUP _END	Ustawia adres ostatniej grupy do skasowania
CMD37	[31...0] Adres danych	R1	UNTAG_ERASE_ GROUP	Anuluje wybraną grupę z zakresu do skasowania
CMD38	[31...0] Dowolne bity	R1b	ERASE	Kasowanie wszystkich wybranych wcześniej bloków danych
CMD42	[31...0] Dowolne bity	R1b	LOCK_UNLOCK	Blokuje/odblokowuje kartę. Wielkość bloku danych jest definiowana za pomocą komendy CMD16
CMD58	Brak	R3	READ_OCR	Odczyt rejestru OCR
CMD59	[31...1] Dowolne bity, [0] opcja CRC	R1	CRC_ON_OFF	Włączenie (bit 1) lub wyłączenie (bit 0) sprawdzania sum kontrolnych CRC

Tab. 4. Format potwierdzenia R1

Bit	Nazwa bitu	Opis
0	Idle State	Karta gotowa do przyjęcia komend
1	Erase Reset	Przerwano sekwencję kasowania
2	Illegal Command	Błędna komenda
3	Com CRC Error	Błąd sumy kontrolnej CRC
4	Erase Seq Error	Błąd sekwencji kasowania
5	Address Error	Podano błędny adres
6	Parameter Error	Nieprawidłowy argument
7	0	Bit zawsze posiada wartość 0

komendy o numerze 0, dla której suma wynosi &H95. W aplikacji emulacji pamięci EEPROM z wykorzystaniem karty MMC nie są wykorzystywane komendy kasowania danych, gdyż nie jest to wymagane. Możliwy jest zapis danych do karty bez wcześniejszego wykonywania komend kasowania.

Karta MMC pracująca w trybie SPI, wysyła kilka rodzajów potwierdzeń. Po wysłaniu komendy do karty, w zależności od jej rodzaju (**tab. 3**), może zostać odesłane potwierdzenie R1, R1b lub R3. Format jednobajtowego potwierdzenia R1 pokazano w **tab. 4**. Większość komend dostępnych w trybie SPI wysyła potwierdzenie typu R1. Potwierdzenie R1b jest to potwierdzenie typu R1 z dodatkowym sygnałem wskazującym na zajętość karty (BUSY). Może mieć ono dowolną liczbę bajtów. Jeśli z karty zostanie odebrany bajt zerowy, będzie on wskazywał na zajętość karty. Odebranie bajtu różnego od 0 wskazuje, że karta jest gotowa do odbioru kolejnej komendy. Dodatkowy sygnał BUSY to ściągnięcie linii DO karty do masy, co będzie powodować odczyt przez mikrokontroler wartości &H00. Dwubajtowe potwierdzenie R2 jest wysyłane w odpowiedzi na komendę żądania odczytu statusu karty i nie jest ono wykorzystywane w przykładowej aplikacji. Również nie jest wykorzystywane w przykładowej aplikacji potwierdzenie typu R3, które jest wysyłane w odpowiedzi na komendę odczytu rejestru OCR. Drugi bajt potwierdzenia R3 zawiera zawartość rejestru OCR. Zarówno w potwierdzeniach R2 i R3, pierwszy bajt jest identyczny jak w potwierdzeniu R1. Potwierdzenia nie muszą być natychmiast zwrócone przez kartę i dlatego wymagane jest wysłanie na linię zegarową od 8 do 64 impulsów, sprawdzając czy otrzymano potwierdzenie. Ponieważ najstarszy bit w potwierdzeniu ma zawsze wartość zerową, można to wykorzystać do sprawdzenia czy otrzymano potwierdzenie. Jest jeszcze jeden rodzaj potwierdzeń nazywanych w skrócie DR (*Data Response*) wysyłanych przez kartę po wysłaniu do niej kompletnego bloku danych przeznaczonych do zapisu. Format tego jednobajtowego potwierdzenia pokazano w **tab. 5**. W przypadku wystąpienia błędu podczas odczytu bloku danych, karta zwróci bajt danych nazywany DET (*Data Error Token*), którego zawartość przestawiono w **tab. 6**. Jeśli otrzymano bajt wskazujący początek danych różny od wartości &HFE, to oznacza, że otrzymano bajt DET.

Tab. 5. Format potwierdzenia Data Response

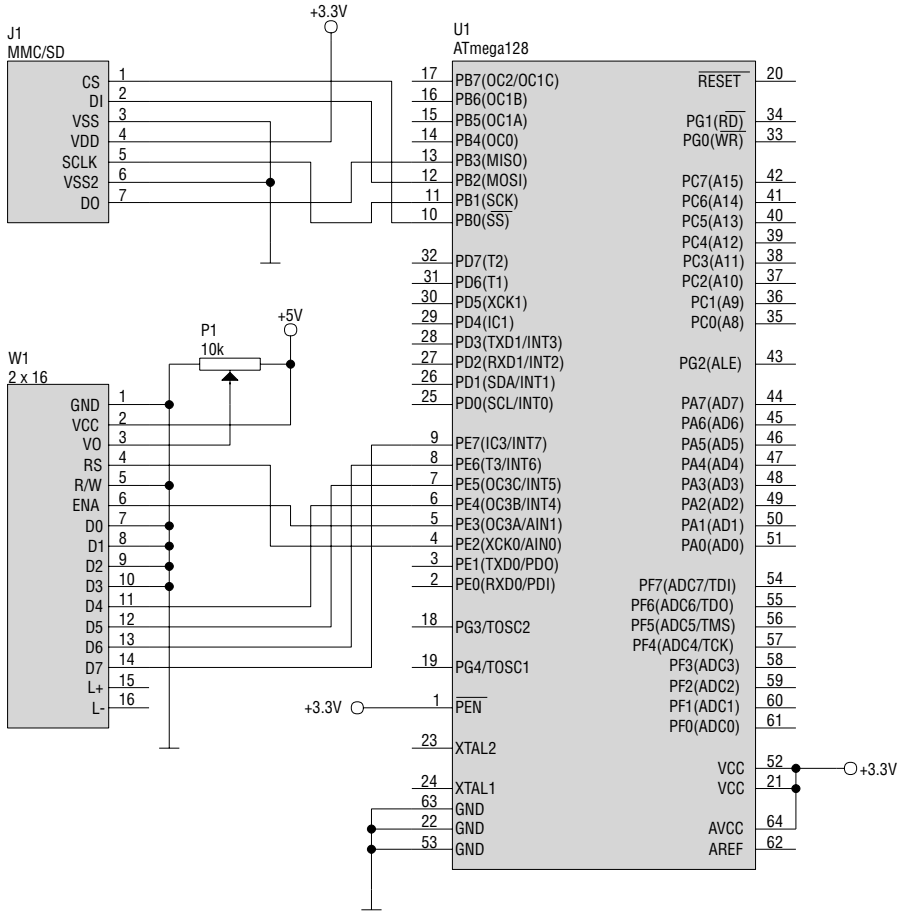
Bit	Wartość bitu
0	1
1	Status
2	Status
3	Status
4	0
5	-
6	-
7	-

Znaczenie bitów Status:
010 – Dane zaakceptowane
101 – Dane odrzucone ze względu na błąd CRC
110 – Dane odrzucone ze względu na błąd zapisu

Aplikacja emulacji pamięci EEPROM w kartach MMC

Do testów wykorzystano kartę MMC o pojemności 1 GB, której producentem jest Kingston. Na **rys. 4** pokazano prosty sposób dołączenia karty MMC do interfejsu SPI mikrokontrolera ATmega128. Zarówno karta, jak i mikrokontroler są zasilane tym samym napięciem +3,3 V. Pin PB0 mikrokontrolera steruje linią /CS karty MMC. Dodatkowo do mikrokontrolera został dołączony wyświetlacz LCD, na którym będą przedstawiane informacje działania przykładowej aplikacji emulacji pamięci EEPROM. Jak było wspomniane, ze względu na potrzebny bufor 512 bajtów na blok danych, zastosowany mikrokontroler do komunikacji z kartą MMC powinien posiadać pamięć RAM o pojemności około 1 kB.

Przykładowy program emulujący pamięć EEPROM z wykorzystaniem karty MMC został pokazany na **list. 1**. Pierwszą czynnością jest skonfigurowanie sprzętowego interfejsu SPI mikrokontrolera, do czego służy polecenie *Config SPI*. Częstotliwość linii zegarowej interfejsu SPI została ustalona na 2 MHz, przy czym transmisja rozpoczyna się od najbardziej znaczącego bitu. Po konfiguracji wyświetlacza LCD inicjowany jest interfejs SPI mikrokontrolera (polecenie *Spiinit*), w wyniku czego zostają odpowiednio ustawione linie komunikacyjne interfejsu SPI. Następnie wywoływana jest procedura zerowania karty MMC jednocześnie konfigurująca kartę do pracy z interfejsem SPI. Przed rozpoczęciem jakiejkolwiek komunikacji z kartą, po włączeniu do niej zasilania, należy wysłać około 80 impulsów zegarowych na linii CLK, przy wysokim



Rys. 4. Przykładowy sposób dołączenia karty MMC do mikrokontrolera ATmega128 wykorzystujący interfejsu SPI

stanie na linii /CS. Jest to realizowane poprzez wysłanie 10 bajtów o wartości &HFF z wykorzystaniem procedury *Wysl_p_cykl*, której parametr określa liczbę wysyłanych bajtów. Aby przełączyć kartę do pracy w trybie SPI, należy do niej wysłać komendę *CMD0* przy niskim stanie na linii /CS. Procedura *Wysl_kom* umożliwia wysłanie dowolnej komendy do karty. Posiada ona trzy parametry, z których pierwszy wskazuje na numer komendy, a pozostałe dwa parametry typu word (2 bajty) są argumentem komendy. Wysyłana komenda w procedurze *Wysl_kom* zawsze jest uzupełniana o stałą sumę CRC, która jest prawidłowa tylko dla komendy *CMD0*. Dla pozostałych komend, mimo przesłania nie jest ona uwzględniana. Karta potwierdza odebranie komendy wysyłając odpowiedź typu R1, na którą oczekiwanie odbywa się w procedurze *Wysl_kom*. Sprawdzany jest bit 0 odpowiedzi R1, wskazujący na zakończenie wykonywania operacji przez kartę. Po wysłaniu komendy *CMD0* w procedurze *Res_mmc*, wywoływana jest procedura *Zak_kom*, która kończy komunikację z kartą MMC. Ustawiana jest w niej linia /CS, po czym do karty wysyłane jest 8 wymaganych impulsów zegarowych. W procedurze *Res_mmc* sprawdzana jest zawartość odpowiedzi R1. Jeśli któryś z jej bitów ma wartość 1, to ustawiana jest zmienna błędu *Err*. Następnie wysyłana jest komenda *CMD1*, rozpoczynająca inicjalizację karty. Po jej wykonaniu karta jest już gotowa do zapisu lub odczytu danych. Po inicjalizacji karty, wywoływana jest procedura *Odcz_cid_csd* umożliwiająca w zależności od wartości zmiennej *Temp* (wskazuje na numer komendy) odczyt rejestru CIS lub CSD. Zwracana przez kartę zawartość rejestru CIS lub CSD rozpoczyna się od bajtu startowego o wartości &HFE. Jeśli ma on inną wartość, to znaczy, że otrzymano DET z zawartością rodzaju błędu. W pierwszej kolejności odczytywany jest do bufora rejestr CIS, z którego pobierana jest nazwa karty wyświetlana następnie w pierwszej linii wyświetlacza LCD. W celu oblicze-

Tab. 6. Format potwierdzenia Data Error Token

Bit	Nazwa bitu	Opis
0	Error	Nieznany błąd
1	CC Error	Błąd kontrolera karty
2	Card ECC failed	Błąd próby wewnętrznej korekcja danych
3	Out of range	Przekroczono granicę fizycznego bloku karty
4	Card Locked	Karta zablokowana
5	0	Bit zawsze posiada wartość 0
6	0	Bit zawsze posiada wartość 0
7	0	Bit zawsze posiada wartość 0

nia pojemności karty, odczytywany jest do bufora rejestr CSD, z którego korzysta procedura *Oblicz_poj*. Obliczana jest w niej pojemność karty przy wykorzystaniu pól *READ_BL_LEN*, *C_SIZE* oraz *MULT* i przedstawionego wcześniej wzoru. Obliczona pojemność zapisywana jest w zmiennej *Poj* typu Long. Pojemność karty w bajtach jest wyświetlana w drugiej linii wyświetlacza LCD. Po 2 sekundach opóźnienia rozpoczyna się prosty test emulowanej z wykorzystaniem karty MMC pamięci EEPROM. Pierwszy krok polega na zapisaniu do karty 5 kB danych o wartościach od 0 do 255. Zmienna *Licz* jest adresem komórki emulowanej pamięci EEPROM, natomiast do zapisywana jest zawartość zmiennej *K*. Do zapisania jednego bajtu emulowanej pamięci EEPROM służy procedura *Zap_byte*, której pierwszy parametr to adres, a drugi wartość do zapisania. Parametr adresu jest typu Long (składa się z 4 bajtów). W procedurze *Zap_byte* w pierwszej kolejności obliczany jest adres bloku karty MMC. Następnie jest wywoływana procedura *Odcz_bloku*, której parametrem jest obliczony wcześniej numer bloku. Przed zapisem danej do karty MMC konieczny jest odczyt bloku danych, w którym będzie modyfikowany tylko jeden bajt. Chodzi o to, aby podczas zapisu jednego bajtu zachować pozostałe dane danego bloku danych. W procedurze odczytu bloku danych (*Odcz_bloku*), adres bloku jest rozdzielany na dwie zmienne, które będą parametrem komendy o numerze *CMD17*, czyli komendy odczytu bloku danych. Jeśli otrzymano prawidłową odpowiedz R1, odbierany jest pierwszy bajt startowy o wartości &HFE, po którym można odebrać 512 bajtów danych zapisywanych następnie do bufora *Buf*. Odebranie bajtu startowego o wartości innej niż &HFE będzie świadczyć o błędzie. Powracając do procedury *Zap_byte*, po odczycie bloku danych, obliczany jest adres modyfikowanej komórki w buforze *Buf*, do której jest zapisywana zawartość zmiennej *Dana*. Zmodyfikowany blok danych jest zapisywany do karty MMC z wykorzystaniem procedury *Zap_bloku*, która posiada identyczne parametry jak procedura *Odcz_bloku*. W procedurze *Zap_bloku*, po rozdzieleniu adresu bloku na dwie zmienne, wysyłana jest komenda o numerze *CMD24* umożliwiająca zapisanie bloku danych do karty MMC. Po wysłaniu komendy, wysyłany jest bajt startowy o wartości &HFE, po którym jest wysyłana zawartość bufora *Buf*. Następnie jest wysyłana dowolna suma kontrolna CRC. Dalej w procedurze *Zap_bloku* następuje oczekiwanie na potwierdzenie otrzymania przez kartę bloku danych. Dodatkowo w procedurze zapisu bloku danych następuje oczekiwanie na zakończenie zapisywania bloku danych. Podczas zapisywania danych do karty MMC, na wyświetlaczu LCD prezentowany jest adres oraz zapisywana dana. Po zapisaniu do karty 5 kB danych, przeprowadzana jest weryfikacja z wykorzystaniem procedury *Odcz_byte* posiadającej parametry identyczne jak procedura *Zap byte*. Odczytana dana spod podanego adresu umieszczana jest w zmiennej *Wart*. Budowa procedury *Odcz_byte* jest podobna do procedury *Zap_byte*, z tym że w procedurze *Odcz_byte* nie jest wywoływana procedura zapisu bloku danych. Jeśli podczas weryfikacji, odczytana wartość jest różna od obliczanej wartości *K*, to zgłaszany jest błąd. Po poprawnej weryfikacji bloku danych o wielkości 5 kB przeprowadzany jest jeszcze jeden test, w którym pod adres 130000 zostaje zapisywana wartość zmiennej *K*, czyli wartość 128. Zarówno adres, jak i zapisywana do pamięci wartość jest wyświetlana w pierwszej linii wyświetlacza LCD. Następnie spod adresu 130000 odczytywana jest wartość. Adres jak i odczytana wartość z pamięci są wyświetlane w drugiej linii wyświetlacza LCD. Jeśli wartość zapisana jest rożna od odczytanej, komunikacja z kartą jest błędna. Zapis danej trwa kilkanaście do kilkudziesięciu milisekund zależnie od typu karty MMC. Dzieje się tak, gdyż podczas emulacji pamięci EEPROM wymagany jest odczyt i zapis bloków danych, a nie pojedynczych danych. Jak wiemy, nie można do karty MMC bezpośrednio zapisać pojedynczej danej pod dowolny adres.

Podsumowanie

Z przedstawionego przykładu wynika, że emulacja łatwych do stosowania pamięci EEPROM z wykorzystaniem karty MMC nie jest skomplikowana. Zazwyczaj karty o pojemnościach rzędu 1 GB nie będą wymagane do własnych aplikacji i z pewnością wystarczą do tego celu bardzo tanie karty o pojemnościach do 128 czy 256 MB. W karcie MMC nie zaimplementowano żadnego systemu plików. Jeśli jednak zainteresowanie Czytelników tematyką stosowania kart Flash we własnych aplikacjach będzie duże, zostanie zaprezentowany artykuł przedstawiający wykorzystywanie ich z wykorzystaniem systemu plików. Może to być np. karta pamięciowa zastępująca dysk twardy.

Marcin Wiązania, EP
marcin.wiazania@ep.com.pl