Obsługa kart pamięciowych SD, część 3

W przedostatniej części cyklu przedstawiamy wybrane zagadnienia związane z transmisją danych do i z kart SD. Autor omawia także najważniejsze polecenia obsługiwane przez te karty, a które służą do obsługi transmisji danych.

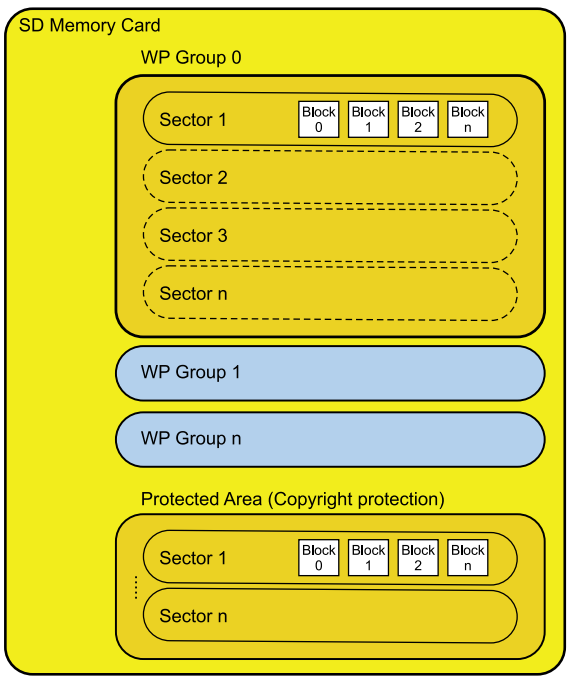
Podstawową jednostką wymiany informacji z pamięcią karty SD jest bajt. Każdy transfer wymagający przesłania danych o większej długości wymaga zdefiniowania bloku danych.

Dla komend zapisu, odczytu i kasowania są zdefiniowane:

- *Blok danych*. Blok o rozmiarze określonym w bajtach jest zapisywany lub odczytywany z karty w trakcie wykonywania komend operujących na bloku danych. Długość bloku może być zmienna (negocjowana) lub stała. Informacja o długości bloku jest zawarta w rejestrze CSD.
- *Sektor danych*. Ilość danych (w bajtach) przeznaczonych do skasowania. Długość sektora jest stała dla każdego typu karty (nie podlega negocjacji) i jest wielokrotnością długości bloku. Informacje o długości bloku są zawarte w rejestrze CSD.
- *Grupa WP* (*WP Group*) – minimalny rozmiar pamięci (od 1 bajtu), która może być objęta protekcją zapisu. Grupa WP jest stała dla każdego typu karty. Informacja o długości grupy WP jest zawarta w rejestrze CSD.

Organizacja pamięci karty SD jest oparta na systemie plików. Przestrzeń pamięci można podzielić na dwie oddzielnie formatowane w systemie DOS partycje:

- Przestrzeń użytkownika (*User Area*). Chroniona lub niechroniona przestrzeń pamięci, do której użytkownik ma dostęp używając standardowych komend zapisu i odczytu.
- Przestrzeń chroniona (*Security Protected Area*) używana do ochrony praw autorskich informacji zapisywanych w przestrzeni użytkownika. Dostęp do przestrzeni chronionej jest możliwy po wykonaniu autoryzacji opisanej w specyfikacji *SD Security Specification*.



Rys. 10. podział pamięci SD na partycje

Z każdą informacją (plikiem) zapisywaną w przestrzeni użytkownika jest skojarzona informacja zapisywana w przestrzeni chronionej. Dla każdego dostępu do pliku (zapisywanie, odczytywanie, lub kasowanie) jest określany stopień ochrony. Jeżeli plik nie jest chroniony, to układ ochrony karty jest przeźroczysty dla operacji na nim wykonywanej. Jeżeli jest chroniony, to układy karty wykonują procedury autoryzacji pozwalając, lub nie na dostęp do pliku.

Przestrzeń użytkownika nie może zajmować całej pamięci karty. Można przyjąć z pewnym przybliżeniem, że ok. 1% całkowitej pojemności zajmuje przestrzeń chroniona, w której użytkownik nie zapisuje swoich danych – szczegółowe informacje można uzyskać w dokumentacji producenta lub przez odczytanie rejestrów CSD i SD Card Status.

Na **rys. 10** pokazano podział pamięci karty na partycje z uwzględnieniem przestrzeni chronionej.

Dla tych obu trybów pracy magistral (SDBus i SPI Bus) takie same są tylko rejestry karty. Inny jest sposób przesyłania danych po magistrali. Inaczej też są interpretowane niektóre komendy. Za korzystaniem z trybu SDBus przemawia większa szybkość przesyłania danych. Tylko z magistralą danych o długości 4 bitów można uzyskać maksymalną gwarantowaną przez producenta prędkość transferu. Poza tym, w trybie SDBus można wykorzystać wszystkie możliwości karty. Jednak jest to okupione bardziej rozbudowanym interfejsem fizycznym i skomplikowanym protokółem wymiany informacji.

Za użyciem magistrali SPI Bus przemawia to, że sprzętowy interfejs SPI jest bardzo popularny. To znacząco upraszcza programowy i sprzętowy interfejs wymiany danych. Wydaje się, że projektanci

Tab. 11. Komendy CMD obsługiwane przez kartę w trybie SPI

Indeks	Argument	Potwierdzenie	Nazwa	Opis komendy
CMD0	Brak	R1	GO_IDLE_STATE	Zerowanie karty
CMD1	Brak	R1	SEND_OP_COND	Aktywacja procesu inicjalizacji karty
CMD9	Brak	R1	SEND_CSD	Polecenie przesłania rejestru CSD
CMD10	Brak	R1	SEND_CID	Polecenie przesłania rejestru CID
CMD12	Brak	R1b	STOP_TRANSMISSION	Wymusza zakończenie transmisji w czasie wykonywania odczytu multiblokowego
CMD13	Brak	R2	SEND_STATUS	Polecenie przesłania rejestru statusu
CMD16	[31:0]długość bloku	R1	SET_BLOCK_LEN	Ustawienie długości bloku (w bajtach) dla operacji zapisu i odczytu danych
CMD17	[31:0] adres	R1	READ_SINGLE_BLOCK	Odczytanie jednego bloku pamięci o długości określonej komendą SET_BLOCK_LEN
CMD18	[31:0] adres	R1	READ_MULTIPLE_BLOCK	Ciągłe odczytywanie bloków pamięci aż do odebrania komendy STOP_TRANSS-MISION
CMD24	[31:0] adres	R1	WRITE_BLOCK	Zapisanie bloku pamięci o długości określonej komendą SET_BLOCK_LEN
CMD25	[31:0] adres	R1	WRITE_MULTIPLE_ _BLOCK	Ciągłe zapisywanie bloków pamięci, aż do odebrania bajtu „stop transmission token”
CMD27	Brak	R1	PROGRAM_CSD	Programowanie zapisywalnych bitów rejestru CSD
CMD28	[31:0] adres	R1b	SET_WRTE_PROT	Jeżeli karta miała określone własności protekcji, to ta komenda ustawia protekcję od adresu określonego w argumencie na liczbę bloków określonych w WP_GRP_SIZE
CMD29	[31:0] adres	R1b	CLR_WRTE_PROT	Jeżeli karta miała określone własności protekcji, to ta komenda kasuje protekcję od adresu określonego w argumencie na liczbę bloków określonych w WP_GRP_SIZE
CMD30	[31:0] write protect adres	R1	SEND_WRITE_PROTECT	Jeżeli karta miała określone własności protekcji, to ta komenda przesyła stan bitów protekcji
CMD32	[31:0] adres	R1	ERASE_WR_BLK_ _START_ADDR	Ustawienie adresu początku pierwszego bloku do skasowania.
CMD33	[31:0] adres	R1	ERASE_WR_BLK_ _END_ADDR	Ustawienie adresu początku ostatniego bloku do skasowania.
CMD38	[31:0] bez znaczenia	R1b	ERASE	Kasowanie grupy bloków ustawionych komendami CMD32 i CMD33
CMD55	[31:0]	R1	APP_CMD	Następna komenda jest komendą aplikacji ACMD
CMD56	[31:0], bit [0] RD/WR	R1	GEN_CMD	Używane zarówno do przesyłania Data Block, jak i odczytywania Data Block z karty. Długośc bloku jest określona w SET_BLOCK_LEN, Bit[0]=1 dane odczytywane z karty Bit[0]=0 dane przesyłane do karty
CMD58	Brak	R3	READ_OCR	Odczytanie rejestru OCR
CMD59	[31:1] bez znaczenia [0] opcja CRC	R1	CRC_ON_OFF	Bit[0]=1 włączone sprawdzanie CRC Bit[0]=0 wyłączone sprawdzanie CRC

kart SD celowo dodali możliwość komunikacji prze interfejs SPI, żeby uprościć komunikację z hostem zbudowanym na bazie popularnych mikrokontrolerów. Z tych powodów opiszemy wyłącznie protokół SD w trybie SPI.

Magistrala jest zorganizowana według zasady *Master–Slave*. Masterem jest zawsze host, czyli zewnętrzny mikrokontroler. Host jest źródłem sygnału zegarowego i inicjuje początek każdej transakcji z kartą SD wymuszając stan niski na linii CS. Wymiana informacji pomiędzy sterownikiem karty SD i hostem odbywa się według schematu:

- Host inicjuje transmisję wysyłając komendę,
- Sterownik karty odpowiada potwierdzeniem,
- Zależnie od wysłanej komendy albo karta wysyła do hosta dane (zawartość rejestru, lub zawartość sektora pamięci), albo host przesyła dane do sterownika (najczęściej sektor danych).

Każda komenda karty SD ma długość 6 bajtów i składa się z:

- kodu komendy o długości 1 bajtu,
- argumentu komendy o długości 4 bajtów,
- sumy kontrolnej CRC o długości 7 bitów (1 bajtu).

W bajcie kodu komendy na sześciu młodszych bitach jest zapisany numer komendy, siódmy bit ma zawsze wartość jeden, a najstarszy ósmy bit jest wyzerowany. Na przykład kod komendy CMDO jest równy 0x40, komendy CMD9 jest równy 0x49, komendy CMD12 jest równy 0x4C i tak dalej.

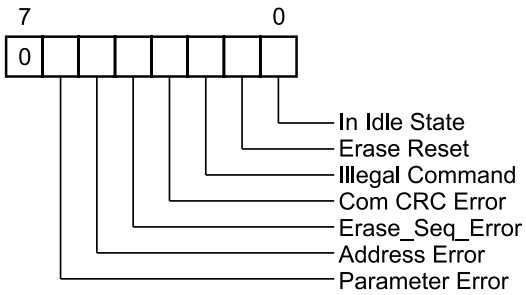
W specyfikacji SD oprócz standardowych komend CMDn zdefiniowano także zestaw komend aplikacji oznaczanych jako ACMDn. Bajt kodu komendy CMDn i ACMDn jest identyczny. (CMD13=0x4D i ACMD13=0x4D). Karta identyfikuje przesłaną komendę jako ACMD, jeżeli bezpośrednio przed nią odebrała prawidłowo komendę CMD55 (APP_CMD).

W czterobajtowe pole argumentu jest wpisywany fizyczny adres początku bloku danych do odczytania, zapisania, lub początek sektora do skasowania. Pole argumentu musi być wysyłane nawet wtedy, kiedy komenda nie wymaga formalnego argumentu. Wysyłane są wtedy same zera. Siedmiobitowa suma kontrolna CRC jest umieszczona w bajcie na siedmiu starszych bitach. Najmłodszy bit jest zawsze jedynką. Po przejściu w tryb SPI karta domyślnie nie sprawdza poprawności CRC i w to pole można wpisać dowolną wartość.

W **tab. 11** zestawiono komendy CMDn, a w **tab. 12** komendy

Tab. 12. Komendy ACMD obsługiwane przez kartę w trybie SPI

Indeks	Argument	Potwierdzenie	Nazwa	Opis komendy
ACMD13	[31:0]	R2	SD_STATUS	Odczytanie rejestru SD Card Status
ACMD18	Zarezerwowana dla trybu ochrony			
ACMD22	[31:0]	R1	SEND_NUM_WR_ _BLOCKS	Odczytanie liczby prawidłowo zapisanych bloków
ACMD23	[31:23] [22:0] liczba bloków	R1	SET_WR_BLK– _ERASE_COUNT	ustawienie liczby bloków zapisywanych we wcześniej skasowanych blokach (szybszy zapis danych)
ACMD24	Zarezerwowana dla trybu ochrony			
ACMD25				
ACMD26				
ACMD41	Brak	R1	SEND_OP_COND	Aktywacja procesu inicjalizacji karty
ACMD42	[31:1] [0]Cd_set	R1	SET_CLR_CARD_ _DETECT	Podłączenie (bit[0]=1), lub odłączenie (bit[0]=0) rezystora podciągającego do wyprowadzenia 1 karty
ACMD43 ACMD49	Zarezerwowana dla trybu ochrony			
ACMD51	[31:0]	R1	SEND_SCR	Odczytanie rejestru OCR



Rys. 11. Format potwierdzenia R1

ACMDn obsługiwane przez karty SD w trybie SPI.

Przy zapisywaniu danych prawidłowy jest tylko blok danych o długości 512 bajtów. Mimo, iż przy odczytywaniu danych blok może mieć długość od 1 do 512 bajtów, to ustawienie długości bloku mniejszej niż 512 bajtów spowoduje błąd przy zapisywaniu danych. Komenda CMD16 nie powinna być używana, jeżeli karta akceptuje domyślną długość bloku 512 bajtów.

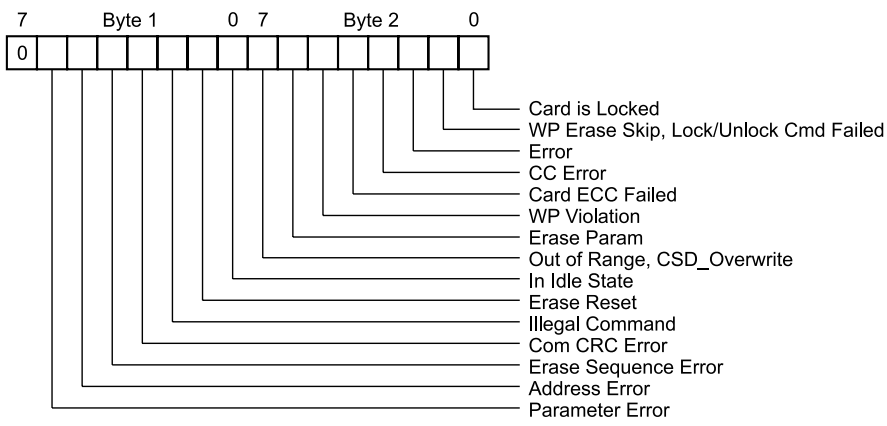
Karty SD wysyłają 3 rodzaje potwierdzeń przesyłanych po odebraniu komendy:

- Potwierdzenie R1 i jej odmiana R1b,
- Potwierdzenie R2,
- Potwierdzenie R3.

Potwierdzenia R1 i R1b są wysyłane przez kartę po wszystkich komendach z wyjątkiem SEND_STATUS (CMD13), SD_STATUS (ACMD13) i READ_OCR(CMD58). Potwierdzenie R1 ma zawsze długość 1 bajtu, a jego format pokazano na **rys. 11**.

Ustawienie bitu w bajcie potwierdzenia R1 oznacza występowanie błędu:

- In Idle State – karta jest w trybie Idle State i wykonuje procedurę inicjalizacji,
- Erase Reset – kasowanie pamięci nie zostało zakończone, bo odebrano komendę, która przerwała kasowanie,
- Illegal Command – odebrano nieprawidłowy kod komendy,
- Com CRC Error – błąd CRC ostatniej komendy,
- Erase_Seq_Error – wystąpił błąd w sekwencji kasowania pamięci,
- Address Error – w argumencie komendy wystąpił adres, który nie był początkiem bloku danych (adresowanie wewnątrz bloku danych),
- Parametr Error – argument komendy (adres, długość bloku) był poza zakresem dla tego typu karty.



Rys. 12. Format potwierdzenia R2

Potwierdzenie R1b ma taki sam format jak potwierdzenie R1, ale po odebraniu kodu R1 karta może dodatkowo sygnalizować zajętość (*busy*) przez wymuszenie na linii danych DO stanu niskiego. Przejście DO w stan wysoki oznacza koniec sygnału zajętości i możliwość przesłania kolejnej komendy. Komenda CMD24 WRITE_BLOCK, mimo iż odpowiada potwierdzeniem R1, ale po wysłaniu do karty bloku danych również na linii DO wprowadza sygnał zajętości do momentu zakończenia zapisywania bloku.

Potwierdzenie R2 ma długość 2 bajtów i jest odpowiedzią na komendę SEND_STATUS i SD_STATUS. Format potwierdzenia R2 pokazano na **rys. 12**. Pierwszy bajt jest identyczny z potwierdzeniem R1. Bity drugiego bajtu jeżeli są ustawione sygnalizują:

- Erase param – nieprawidłowy sektor do kasowania,
- Write protect violation – komenda próbuje zapisać blok zabezpieczony protekcją zapisu,
- Card ECC failed – korekcja błędów ECC nie zdołała naprawić błędu,
- CC error – błąd wewnętrznego mikrokontrolera,
- Error – wystąpił wewnętrzny nieznany błąd karty,
- Write protect erase skip – tylko część przestrzeni z zaprogramowanej przez WP blocks zostało skasowanej,
- Card is locked – karta zablokowana.

Potwierdzenie R3 jest odpowiedzią na komendę READ_OCR i ma długość 5 bajtów. Pierwszy bajt R3 ma identyczną strukturę jak potwierdzenie R1, pozostałe 4 zawierają kopię rejestru OCR karty.

Po włączeniu zasilania karta przechodzi w domyślny tryb pracy z magistralą SDBus. Magistrala karty po włączeniu zasilania i przed wysłaniem pierwszej komendy wymaga zainicjowania przez wysłanie na

linię zegarową SCLK przynajmniej 74 cykli zegara.

Żeby ją przełączyć w tryb pracy z magistralą SPI, trzeba wymusić na wyprowadzeniu CS stan niski i wysłać do karty komendę CMD0. Jeżeli wewnętrzny kontroler karty uzna, że niezbędny jest tryb SDBus, to nie wysyła odpowiedzi na CMD0 i nie przełączy się na tryb SPI. Host musi powtarzać komendę CMD0, aż do uzyskania potwierdzenia R1.

Po przejściu w tryb SPI domyślnie jest wyłączane sprawdzanie poprawności CRC, ale w trakcie wysyłania CMD0 po włączeniu zasilania karta jest w trybie SDBus i musi wysłać poprawny CRC. CMD0 jest komendą, której kod (0x40) i pole argumentu (0x0000) mają zawsze stałą wartość. Wyliczona dla CMD0 suma CRC jest stała i wynosi 0x4A. Dodając jedynkę na najmłodszym bicie otrzymujemy bajt CRC o wartości 0x95.

Sekwencja zerowania zaczyna się opisywaną już komendą CMD0 wysyłaną w celu przełączenia w tryb SPI po włączeniu zasilania, lub rozpoczęcia procedury zerowania karty w trakcie normalnej pracy. Po prawidłowym wykonaniu komendy CMD0 karta przechodzi w stan *idle state*. W stanie *idle state* możliwe jest wysłanie jednej z komend:

- CMD1 SEND_OP_COND
- ACMD41 SD_SEND_OP_COND
- CMD59 CRC_ON_OFF
- CMD58 READ_OCR

Host musi powtarzać wysyłanie komendy CMD1 SEND_OP_COND i odczytywać potwierdzenie R1, aż do wyzerowania bitu zerowego R1 InIdleState. Wtedy zerowanie (inicjalizacja) karty jest zakończona i można wprowadzać inne komendy. Takie zerowanie jest identyczne z zerowaniem karty MMC pracującej w trybie SPI. Jednak są karty SD, które nie akceptują komendy CMD1 do kończenia stanu *idle state*. Zamiast komendy CMD1 trzeba wysłać komendę ACMD41 pamiętając, że przed nią trzeba wysłać komendę CMD55 (APP_CMD). W stanie *idle state* można wysłać komendę CMD58 i odczytać wartość rejestru OCR. Komenda CMD58 może być również wysłana w trakcie normalnej pracy karty.

Rozpoczęcie przesyłania danych magistralą SPI musi być poprzedzone zerowaniem karty. Po zakończeniu zerowania, karta musi być zasilana napięciem określonym w rejestrze OCR.

W kartach SD blok danych ma domyślny rozmiar 512 bajtów i nie jest możliwe zapisanie bloku o innej długości. Również sektor kasowania danych ma długość 512 bajtów.

Przy przesyłaniu danych magistralą pracującą w trybie SD SPI trzeba pamiętać o kilku ograniczeniach:

- Częstotliwość taktowania magistrali może się zmieniać w dowolnym momencie, ale nie może przekroczyć maksymalnej częstotliwości podanej w parametrach karty.
- Wymagane jest, żeby sygnał zegara na linii SCLK był przesyłany do karty w czasie odczytywania danych i oczekiwania na potwierdzenia i bajtu startu.
- Host musi wykrywać stan zajętości. Karta sygnalizuje stan zajętości w trakcie wykonywania zapisywania lub kasowania danych przez wymuszenie stanu niskiego na linii danych DO. W czasie, kiedy karta jest w stanie zajętości host musi taktować linię zegarową SCLK. Jeżeli karta nie będzie taktowana, to linia DO przejdzie w stan niski i pozostanie w tym stanie nawet po zakończeniu zapisywania.

Po zakończeniu transakcji na magistrali SPI wymagane jest wysłanie 8 cykli zegarowych w następujących sytuacjach:

- W sekwencji komenda – potwierdzenie. Osiem cykli zegara jest wysyłanych po odebraniu ostatniego bitu potwierdzenia.
- Zakończenie sekwencji odczytywania danych. Osiem cykli zegara jest wysyłanych po odczytaniu ostatniego bitu bloku danych (ostatniego bitu CRC).
- Zakończenie sekwencji zapisywania danych. Osiem cykli zegara jest wysyłanych po odebraniu ostatniego bitu potwierdzenia Data Response.

W czasie wysyłania tych cykli stan linii CS nie jest istotny. Dane z karty są odczytywane w blokach o długości określonej w polu READ_BL_LEN rejestru CSD. Długość bloku danych może mieć minimalną długość 1 bajta, a maksymalną 512 bajtów, i jest ustawiana komendą CMD16 SET_BLOCK_LENGHT.

Do odczytywania danych karty SD wykorzystywane są 2 komendy: CMD17 i CMD18. Komenda CMD17 odczytuje pojedynczy blok. Jeżeli karta zaakceptuje komendę odczytu danych, to wysyła potwierdzenie R1 z wyzerowanymi wszystkimi bitami. Host musi odczytywać dane z karty, aż odbierze bajt startu 0xEF. Może się zdarzyć, że z jakichś powodów karta nie odpowie bajtem startu. Procedura odczytywania bajtu startu powinna odczytywać bajt startu z karty przez określony czas i jeżeli nie odbierze 0xEF, to kończy odczytywanie danych i zgłasza błąd.

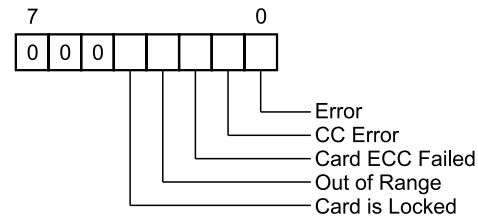
Po odebraniu bajtu startu odczytywany jest blok danych. Blok danych jest uzupełniany dwoma bajtami sumy kontrolnej CRC wyliczanej według zależności:

$$x^{16}+ x^{12}+ x^{5}+1$$

Odczytanie CRC kończy odczytywanie pojedynczego bloku danych i karta jest gotowa na przyjęcie nowej komendy.

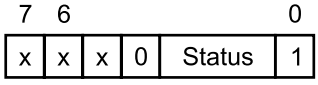
Parametrem komendy odczytywania bloku danych jest adres danych. Podawany adres nie może mieć dowolnej wartości. Na przykład, jeżeli blok ma długość 512 bajtów, to adres odczytywania pierwszego bloku ma wartość 0, adres odczytywania drugiego bloku ma wartość 512, trzeciego 1024 itd. Wysłanie w tym wypadku komendy CMD17 z adresem innym niż wielokrotność długości bloku zostanie zinterpretowane przez kartę jako błąd *AddressError* sygnalizowany w potwierdzeniu R1. Wysłanie adresu większego niż to wynika z pojemności karty sygnalizowane będzie błędem *ParameterError* w potwierdzeniu R1.

Komenda odczytywania bloku danych może powodować błąd wewnętrznego sterownika, lub karta z jakichś przyczyn nie może odesłać danych. Wtedy karta zamiast



Card ECC failed - korekcja błędów ECC nie zdołała naprawić błędu
CC error - błąd wewnętrznego mikrokontrolera
Error - wystąpił wewnętrzny nieznany błąd karty
Card is locked - karta zablokowana
Out of range - adres odczytu poza zakresem

Rys. 13. Potwierdzenie Data Error Token



pole STATUS
010 - dane zaakceptowane
101 - dane odrzucone z powodu błędu CRC
110 - dane odrzucone z powodu błędu zapisywania

Rys. 14. Potwierdzenie *Data Response*

danych odeśle bajt *Data Error Token* – **rys. 13**.

Komenda CMD18 odczytuje wiele bloków danych. Tak jak w przypadku komendy odczytywania pojedynczego bloku, po wysłaniu komendy karta odpowiada potwierdzeniem R1, a potem bajtem startu 0xFE. Również w razie wystąpienia błędu karta odpowiada bajtem *Data Error Token*.

Po prawidłowym potwierdzeniu R1 (wszystkie bity wyzerowane) i odebraniu bajtu startu, host może odczytywać kolejne bloki danych od bloku, którego adres został umieszczony w argumencie komendy. Każdy blok danych jest uzupełniany dwoma bajtami CRC. Karta kończy wysyłanie danych, kiedy odbierze komendę CMD12 STOP_TRANSMISSION.

W trybie SPI karta SD może zapisywać pojedynczy blok danych komendą CMD24 i wiele bloków danych komendą CMD25.

Zapisywanie pojedynczego bloku danych rozpoczyna się od wysłania komendy CMD24. Karta odpowiada potwierdzeniem R1 z wyzerowanymi wszystkimi bitami. Host wysyła bajt startu 0xFE i 512 bajtowy blok danych plus 2 bajty CRC. Karta potwierdza odebrany blok danych bajtem *Data Response* – **rys. 14**.

Dane zostaną zapisane poprawnie, kiedy pole STATUS ma wartość 010. Zapisywanie danych w pamięci Flash musi trwać jakiż czas. Nie można wysłać komendy zapisywania nowych danych, jeżeli zapis poprzednich jeszcze się nie zakończył. Karta sygnalizuje, że trwa proces zapisu przez wymuszenie stanu niskiego na linii danych – host musi czekać, aż linia DO przejdzie w stan wysoki. Po zakończeniu zapisu, kiedy wystąpił błąd zapisywania (Status=110), host powinien odczytać rejestr statusu komendą CMD13 SEND_STATUS, żeby określić jego przyczynę.

Komenda CMD25 może zapisywać wiele bloków danych. Po wysłaniu kodu komendy z adresem pierwszego bloku danych do zapisania karta odpowiada potwierdzeniem R1 z wyzerowanymi wszystkimi bitami. Host musi wtedy wysłać bajt startu 0xFC i zapisywany blok danych. Karta potwierdza odebranie bloku danych bajtem *Data Response* i wymusza na linii danych DO stan zajętości (stan niski). Wykrycie błędu sygnalizowane w *Data Response* wymaga zatrzymania zapisu wielkoblokowego przez host komendą CMD12. Dodatkowe informacje o przyczynach błędu zapisu znajdują się w rejestrze statusu odczytywanym komendą CMD13 SEND_STATUS. Komenda ACMD22 umożliwia określenie liczby poprawnie zapisanych bloków. Żeby zatrzymać zapisywanie wielkoblokowe trzeba wysłać do karty bajt stopu 0xFD.

Gdy karta podczas zapisywania danych jest zajęta, host może ustawić linię CS w stan wysoki. Linia DO przejdzie w stan wysokiej impedancji, ale nie przerwie to zapisywania danych w pamięci Flash.

W protokole SPI rejestry statusu i OCR są przesyłane do hosta jako potwierdzenia komend. Zawartości rejestrów CID i CSD są odczytywane tak jak blok danych o długości 16 bajtów uzupełniony o 2 bajty sumy kontrolnej CRC. Czas dostępu do danych w trakcie odczytywania rejestru CSD nie może być określony przez pole TAAC, bo to pole jest zawarte w rejestrze CSD. Do określenia czasu dostępu w tym przypadku stosowany jest standardowy czas dostępu definiowany w parametrach karty jako *Ncr*.

Tomasz Jabłoński, EP
tomasz.jablonski@ep.com.pl