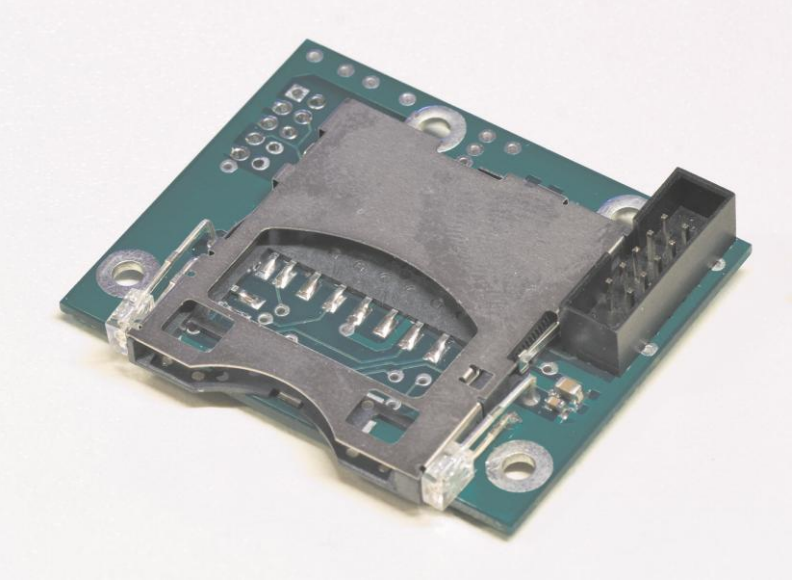
SimpleSD_RS

opis oprogramowania układowego do modułu karty SD



Kontakt:	**http://xyzyk.pl**	**pk@xyzyk.pl**

Informacje ogólne

Program przeznaczony jest do zorganizowania rejestracji danych w istniejącym urządzeniu. Dane można zapisać na kartę przez UART oraz SPI. Podstawowa transmisja odbywa się przez UART i umożliwia:

- odczyt stanu karty (jest, nie ma, nie sformatowania, wolne miejsce),
- otwarcie/zamknięcie/usunięcie/zmiana nazwy pliku,
- odczyt/zapis danych do pliku,
- odczyt/zapis danych do wewnętrznego bloku pamięci nieulotnej (ok 60kB przeznaczone do dowolnego wykorzystania),
- świecenia LEDami.

Transmisja przez SPI umożliwia tylko zapis danych do otwartego pliku. Konfiguracja modułu umożliwia zaprogramowanie go tak, aby pliki były tworzone automatycznie przy załączeniu zasilania lub wsadzeniu karty. W ten sposób do modułu można podpiąć tylko SPI bez potrzeby żadnej inicjalizacji przez UART.

UART (port A):

- maksymalna szybkość 24000000bps,
- bufor na dane: 8kB (+ 20 bajtów na dane związane z protokołem).

SPI:

- maksymalny zegar: 6MHz,
- bufor na dane: 16kB.

W testach przez SPI udało się uzyskać ciągły zapis ok 500kB/s do wyczerpania miejsca na karcie (Kingston Elite Pro, SD, 1GB, 50x, FAT16).

Konfiguracja - opis

Przed rozpoczęciem pracy moduł należy skonfigurować. W tym celu należy stworzyć plik tekstowy (opis poniżej) i nadać mu nazwę „**konfig.prg**”, skopiować na kartę, wsunąć do modułu i załączyć zasilanie. Podobnie jak przy zmianie oprogramowania układowego i tutaj moduł zmieni nazwę pliku (na „**konfig.pr_**”). Jeśli ta operacja nie jest pożądana, należy nadać plikowi nazwę „**konfig.prX**” (UWAGA! Po odczycie konfiguracji jest ona automatycznie zapisywana w pamięci wewnętrznej; Jeśli moduł będzie to robił przy każdym włączeniu zasilania to może dojść do uszkodzenia pamięci!).

Gdy moduł znajdzie plik „**konfig.prg**” lub „**konfig.prx**” przy uruchomieniu to najpierw naprzemiennie miga LEDami (zielony-czerwony-zielony), a gdy przetworzy plik to sygnalizuje zapis do pamięci wewnętrznej przez trzykrotne jednoczesne mignięcie oboma LEDami.

Format pliku konfiguracyjnego:
każda linia to zapis jednego parametru
`<nazwa_parametry>_spacje_<wartość>`
Średnik na początku linii oznacza że jest to komentarz. Parametry których nazwy nie występują w programie i tak zostaną ominięte. W linii w której jest jakiś parametr nie może być już komentarza.
Wielkość liter nie ma znaczenia.
Wartość liczbowa może być podana dziesiętnie (normalnie, tj „2”, „30”, itp) lub w systemie szesnastkowym („0x05”, „0xAB”).

Dostępne opcje:

Ogólne	
RESTOREDEFAULT	Przywraca parametry domyślne. Tu wyjątkowo nie podaje się wartości!
Ledy	
LED_AUTO_R	Bajt konfiguracyjny leda czerwonego. Ustawia sytuacje w których moduł ma migać tym LEDem (jeśli LED ma być sterowany przez UART to ten parametr musi wynosić „0”!) Znaczenie bitów: (opcja aktywna = 1) 0x01 – automatycznie świeć gdy nie ma karty, 0x02 – karta zła (jest wsadzona ale nieobsługiwana), 0x04 – karta wsadzona i działa, 0x08 – karta wsadzona i tylko do odczytu, 0x10 – migaj gdy odebrano poprawną transmisje przez UART, 0x20 – migaj gdy odebrano dane przez SPI
LED_AUTO_R_EMPTY **LED_AUTO_R_BAD** **LED_AUTO_R_OK** **LED_AUTO_R_READONLY** **LED_AUTO_R_UART** **LED_AUTO_R_SPI** Analogiczny zestaw jest do leda zielonego (**LED_AUTO_G_***).	Wartości 0 lub 1 kasują bądź ustawiają osobno powyższe znaczniki.
LEDY_AUTOFLASH_EMPTY **LEDY_AUTOFLASH_BAD** **LEDY_AUTOFLASH_OK** **LEDY_AUTOFLASH_READONLY**	Należy podać maksymalnie 10 bajtów zapisanych w HEX (20 znaków) zawierających motyw migania w danej sytuacji. Więcej informacji w dziale **Motywy migania LEDami**.
UART	
UART_BAUDRATE	Szybkość UARTu: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200..24000000. Pośrednie wartości mogą być dostępne ale wymagają sprawdzenia.
SPI	
SPI_CFG	Konfiguracja SPI. Tak jak w przypadku ledów, jest to 1 bajt gdzie poszczególne bity oznaczają: (1=opcja aktywna) 0x01 – stan SPI po włączeniu modułu 0x02 – patrz niżej 0x04 – patrz niżej

POL	CPH	Shift CK Edge	Capture CK Edge	CK Inactive Level
0	1	Falling	Rising	Low
0	0	Rising	Falling	Low
1	1	Rising	Falling	High
1	0	Falling	Rising	High

SPI_CFG_ONSTATE **SPI_CFG_POL** **SPI_CFG_CPH**	Wartości 0 lub 1 kasują bądź ustawiają osobno powyższe znaczniki.

SPI_AUTOFLUSH	Znaczniki: 0x01 – opcja AutoFlush włączona.
SPI_AUTOFLUSH_ON	Wartości 0 lub 1 kasują bądź ustawiają osobno powyższe znaczniki.
SPI_AUTOFLUSH_TIMEOUT	Ile od ostatniego zapisu robić flush danych (czas=wartość*10ms; wartość=0 oznacza zapis po każdej paczce).
Automatyczne tworzenie pliku	
AUTOFILE_CFG	Znaczniki: 0x01 – twórz plik automatycznie.
AUTOFILE_CFG_CREATE	Wartości 0 lub 1 kasują bądź ustawiają osobno powyższe znaczniki.
AUTOFILE_MAXSIZE	Maksymalna wielkość pliku w bajtach po której przekroczeniu nastąpi stworzenie kolejnego pliku. Maksymalna dostępna wartość to 1GB (1073741824). Wartość=0 oznacza bez limitu.

Nastawy domyślne równe są takiej konfiguracji:

```
LED_AUTO_R                0x30
LED_AUTO_G                0x0F
LEDY_AUTOFLASH_EMPTY      01000000000000000000
LEDY_AUTOFLASH_BAD        01010101010101010101
LEDY_AUTOFLASH_OK         01000000000100000000
LEDY_AUTOFLASH_READONLY   01000100000000000000
UART_BAUDRATE             115200
SPI_CFG                   0
SPI_AUTOFLUSH             1
SPI_AUTOFLUSH_TIMEOUT     10
AUTOFILE_CFG              0
AUTOFILE_MAXSIZE          1073741824
```

Konfiguracja – Przykładowe pliki konfiguracyjne

Taką czcionką jest zaznaczona zawartość pliku.	A to są komentarze na potrzeby tego opisu i ich w plikach nie ma.

Przywrócenie nastaw domyślnych. I nic więcej:

```
RestoreDefault
```

Ustawienie szybkości UARTu i włączenie SPI:

```
RestoreDefault                  Na samym początku przywracam nastawy
                                domyślne. Tylko żeby upewnić się że
                                wszystkie inne nastawy których tu nie
                                zmieniam będą zmienione na domyślne.

UART_BAUDRATE 24000000          Ustawiam szybkość portu na 24000000bps.
SPI_CFG 0x01                    SPI włączone przy uruchomieniu.
```

SPI mogę również włączyć ustawiając tylko sam bit:

```
SPI_CFG_ONSTATE 1               SPI włączone przy uruchomieniu.
```

Sterowanie LEDami:

```
LEDY_AUTOFLASH_EMPTY 0          Gdy nie ma karty – LED zgaszony
LEDY_AUTOFLASH_OK 01000000000100000000  Jest karta – mignięcie co sekundę.
LEDY_AUTOFLASH_READONLY FF      Karta tylko do odczytu – świecę cały
                                czas.
```

Pełna konfiguracja do trybu „**prostego**”:

```
RestoreDefault                  Przywracam nastawy domyślne.

UART_BAUDRATE 19200             Szybkość UARTu – 19200bps

SPI_CFG_ONSTATE 1               Włączam SPI.
SPI_CFG_POL 1                   Ustawiam polaryzacje SPI.

SPI_AUTOFLUSH_ON 1              Włączam AutoFlush.
SPI_AUTOFLUSH_TIMEOUT 0         Timeout opcji AutoFlush = 0s

AUTOFILE_CFG_CREATE 1           Włączam automatyczne tworzenie pliku.
AUTOFILE_MAXSIZE 1048576        Maksymalna wielkość pliku tworzonego
                                automatycznie – 1MB
```

UART – Protokół transmisji

Szybkość transmisji jest zadawana w nastawach. Pozostałe parametry obecnie nie są konfigurowalne i są takie: 8 bitów danych, bez parzystości.

Opis paczki danych:

```
Do modułu przychodzi:
        0x5A                    - 1 bajt identyfikatora (zawsze 0x5A)
        <rozkaz>                - 1 bajt
        <ile_danych>            - 2 bajty ilości danych
        <dane>                  - <ile_danych> bajtów

Moduł odpowiada:
        0xDA                    - 1 bajt identyfikatora (zawsze 0xDA)
        <kod_odpowiedzi>        - 1 bajt

        Jeśli kod_odpowiedzi to OK (=0) to zwraca pozostałe dane.
        W każdym innym wypadku zwraca tylko bajt kod_odpowiedzi!!!
```

Znaczenie bajtu <kod_odpowiedzi>:

= 0 – OK (operacja wykonana poprawnie),
= 1 – nieprawidłowe parametry,
= 2 – błąd,
= 3 – brak karty,
= 4 – plik nie otwarty.

UWAGA!
1. Na początku KAŻDEJ paczki jest bajt identyfikacyjny!!!

2. Moduł odsyła całą odpowiedź dopiero PO zakończeniu wykonywania rozkazu. Np odpowiedź na rozkaz zapisania danych do pliku odeśle dopiero gdy dane te zostaną faktycznie zapisane (lub umieszczone w buforze wewnętrznym karty – w każdym razie PO wykonaniu rozkazu).

3. Po otrzymaniu pełnego i poprawnego rozkazu moduł odpowiada bajtem identyfikacyjnym jeszcze ZANIM przetworzy całą paczkę i ją wykona. Dopiero po zakończeniu przetwarzania danych (np zapisu na kartę jak w pkt 2) moduł odsyła resztą paczki odpowiedzi.

Rozkazy:

SimpleSD_Exe_GetProgVer = 0x01
Zwraca wersję programu modułu.

```
parametry:
        <ile_danych>          - 2 bajty (=0)

odpowiedź:
        <kod_odpowiedzi>      - 1 bajt (zawsze OK)
        <prog_id>             - 1 bajt (ID programu)
                              = 0x03 - SimpleSD
        <prog_ver>            - 1 bajt (wersja programu)
        <asm_y>               - 1 bajt (data asemblacji: rok)
        <asm_m>               - 1 bajt (data asemblacji: miesiąc)
        <asm_d>               - 1 bajt (data asemblacji: dzień)
        <asm_h>               - 1 bajt (data asemblacji: godzina)
        <asm_n>               - 1 bajt (data asemblacji: minuta)
        <asm_s>               - 1 bajt (data asemblacji: sekunda)
```

SimpleSD_Exe_GetFree = 0x02
Zwraca ilość wolnego miejsca na karcie.

```
parametry:
        <ile_danych>          - 2 bajty (=0)

odpowiedź:
        <kod_odpowiedzi>      - 1 bajt
        [wielkosc__LL]        -  \
        [wielkosc__LH]        -   \ 5 bajtów
        [wielkosc__HL]        -   / ilości wolnego miejsca w bajtach
        [wielkosc__HH]        -  /
        [wielkosc_HHH]        - /
```

SimpleSD_Exe_Format = 0x03
Sprawdza poprawność systemu plików na karcie.

```
parametry:
        <ile_danych>          - 2 bajty (=0)

odpowiedź:
        <kod_odpowiedzi>      - 1 bajt
        [wielkosc__LL]        -  \
        [wielkosc__LH]        -   \ 5 bajtów
        [wielkosc__HL]        -   / ilości wolnego miejsca w bajtach
        [wielkosc__HH]        -  /
        [wielkosc_HHH]        - /
```

SimpleSD_Exe_GetStatus= 0x04
Zwraca status karty.

```
parametry:
        <ile_danych>   - 2 bajty (=0)

odpowiedź:
        <kod_odpowiedzi>            - 1 bajt
        [status]                    - 1 bajt statusu (kodowany bitowo)
```

```
                Zwracane wartości:    (4 najmłodsze bity)
                0x_X = 0              - brak karty
                0x_X = 1              - karta wsunięta i zainicjalizowana
                0x_X = 2              - jest karta ale ma nieobsługiwany format
                0x_X = 3              - karta wsunięta i zainicjalizowana, ale tylko do odczytu

                dodatkowo: (maski bitowe)
                0x80 = 1              - plik otwarty
                0x40 = 1              - otwarty plik jest tylko do odczytu (otwarto z
parametrem "r" lub karta tylko-do-odczytu)
```

SimpleSD_Exe_SetLedR = 0x05
SimpleSD_Exe_SetLedG = 0x06
Ustawia świecenie LEDem.
UWAGA! Opcja działa jeśli dla danego LEDa opcja "LED_AUTO_x" jest =0!
Więcej o miganiu w rozdziale **„Motywy migania LEDami”.**

Dostępne są dwa zestawy parametrów:
1.
parametry:
- <ile_danych> - 2 bajty (=1)
- <maska świecenia> - 1 bajt maski świecenia który zostanie powielony na cały bufor LEDa

odpowiedź:
- <kod_odpowiedzi> - 1 bajt

2.
parametry:
- <ile_danych> - 2 bajty (=10)
- <bufor_leda_0> - \ 10 bajtów
- ... - / pełnego bufora
- <bufor_leda_9> - / leda

odpowiedź:
- <kod_odpowiedzi> - 1 bajt

SimpleSD_Exe_SPI = 0x07
Włącza/wyłącza SPI.

parametry:
- <ile_danych> - 2 bajty (=1)
- <rozkaz> - 1 bajt
  - =F0 - wyłącz SPI
  - =F1 - włącz SPI
  - każda inna wartość nic nie zmienia

odpowiedź:
- <kod_odpowiedzi> - 1 bajt
- <stan SPI> - 1 bajt
  - =F0 - SPI wyłączone
  - =F1 - SPI włączone

SimpleSD_Exe_FileOpenRead = 0x10
SimpleSD_Exe_FileOpenWrite = 0x11
SimpleSD_Exe_FileOpenAppend = 0x12
Otwiera plik w danym trybie:
- *Read - otwiera plik do odczytu i ustawia pozycję na początku pliku
- *Write - otwiera plik do zapisu i ustawia pozycję na początku pliku
  - jeśli plik nie istnieje to go tworzy,
  - jeśli plik istnieje to go NADPISUJE;
- *Append - otwiera plik do zapisu i ustawia pozycję na końcu pliku
  - jeśli plik nie istnieje to go tworzy,
  - nie nadpisuje zawartości.

parametry:
- <ile_danych> - 2 bajty (>0)

```
        <dane>                - nazwa pliku

odpowiedź:
        <kod_odpowiedzi>      - 1 bajt
```

SimpleSD_Exe_FileClose= 0x13
Zamyka plik.

```
parametry:
        <ile_danych>          - 2 bajty (=0)

odpowiedź:
        <kod_odpowiedzi>      - 1 bajt
```

SimpleSD_Exe_FileWrite= 0x14
SimpleSD_Exe_FileWriteFlush = 0x15
Zapisuje dane do pliku. Plik musi być otwarty do zapisu.
Rozkaz "SimpleSD_Exe_FileWriteFlush" wymusza fizyczny zapis danych na kartę.

```
parametry:
        <ile_danych>          - 2 bajty (>0, <= 1024)
        <dane>...

odpowiedź:
        <kod_odpowiedzi>      - 1 bajt
```

Dla rozkazu "SimpleSD_Exe_FileWriteFlush" istnieje jeszcze możliwość samego wymuszenia zapisu buforowanych danych na kartę:

```
parametry:
        <ile_danych>          - 2 bajty (=0)

odpowiedź:
        <kod_odpowiedzi>      - 1 bajt
```

SimpleSD_Exe_FileRead = 0x16
Czyta dane z pliku. Plik musi być otwarty.

```
parametry:
Dostępne są dwa prawidłowe zestawy parametrów:
        1. Odczyt max 0xFFFF bajtów:
        <ile_danych>          - 2 bajty (=2)
        <ile_L>               - młodszy bajt ilości
        <ile_H>               - starszy bajt ilości

        2. Odczyt max 0xFFFFFFFF bajtów:
        <ile_danych>          - =4
        <ile_LL>              -
        <ile_LH>              -
        <ile_HL>              -
        <ile_HH>              -

        Przesłana ilość danych musi >0!


odpowiedź:
        <kod_odpowiedzi>      - 1 bajt
        [ile_danych]          - 2/4 bajty (tyle ile było parametrów dla rozkazu)
        [dane]                - 0..<ile> bajtów
```

SimpleSD_Exe_GetFileSize   = 0x17
Zwraca wielkość otwartego pliku.

```
parametry:
        <ile_danych>          - 2 bajty (=0)

odpowiedź:
        <kod_odpowiedzi>      - 1 bajt
```

```
        [wielkosc_LL]         - \
        [wielkosc_LH]         -  \ 4 bajty
        [wielkosc_HL]         -  / wielkości pliku w bajtach
        [wielkosc_HH]         - /
```

SimpleSD_Exe_GetFilePos = 0x18
Zwraca pozycję w otwartym pliku.

```
parametry:
        <ile_danych>          - 2 bajty (=0)

odpowiedź:
        <kod_odpowiedzi>      - 1 bajt
        [wielkosc_LL]         - \
        [wielkosc_LH]         -  \ 4 bajty
        [wielkosc_HL]         -  / pozycji w pliku w bajtach licząc od początku pliku
        [wielkosc_HH]         - /
```

SimpleSD_Exe_SetFilePos = 0x19
Ustawia pozycję w pliku. Plik musi być otwarty do odczytu!

```
parametry:
        <ile_danych>          - 2 bajty (=5)
        <początek>            - z jakiego miejsca liczyć pozycje
                                    Dostępne wartości:
                                    =0      - od początku pliku
                                    =1      - od bieżącej pozycji
                                    =2      - od końca pliku
        <wielkosc_LL>         - \
        <wielkosc_LH>         -  \ 4 bajty
        <wielkosc_HL>         -  / pozycji w pliku
        <wielkosc_HH>         - /

odpowiedź:
        <kod_odpowiedzi>      - 1 bajt
```

SimpleSD_Exe_FileDelete = 0x1A
Usuwa plik o podanej nazwie.

```
parametry:
        <ile_danych>          - 2 bajty (>0)
        <dane>                - nazwa pliku

odpowiedź:
        <kod_odpowiedzi>      - 1 bajt
```

SimpleSD_Exe_FileRename = 0x1B
Zmienia nazwę pliku.

```
parametry:
        <ile_danych>          - 2 bajty (>0)
        <dane>                - nazwa pliku źródłowego
        <przerwa>             - bajt=0
        <dane>                - nazwa pliku docelowego

odpowiedź:
        <kod_odpowiedzi>      - 1 bajt
```

SimpleSD_Exe_EE_GetSize = 0x30
Zwraca ilość wolnego miejsca w eepromie (w bajtach).

```
parametry:
        <ile_danych>          - 2 bajty (=0)

odpowiedź:
        <kod_odpowiedzi>      - 1 bajt
        [wielkosc__L]         -  \ 2 bajty
        [wielkosc__H]         -  / ilości wolnego miejsca
```

SimpleSD_Exe_EE_Write = 0x31
Zapisuje dane w eepromie.

```
parametry:
       <ile_danych>          - 2 bajty (>2)
       <adres_L>             - adres w eepromie (młodszy bajt)
       <adres_H>             - adres w eepromie (starszy bajt)
       <dane>...

odpowiedź:
       <kod_odpowiedzi>      - 1 bajt
```

SimpleSD_Exe_EE_Read = 0x32
Czyta dane z eeproma.

```
parametry:
       <ile_danych>          - 2 bajty (=4)
       <adres_L>             - adres w eepromie (młodszy bajt)
       <adres_H>             - adres w eepromie (starszy bajt)
       <ile_czytac_L>        - ile danych czytać (młodszy bajt)
       <ile_czytac_H>        - ile danych czytać (starszy bajt)

odpowiedź:
       <kod_odpowiedzi>      - 1 bajt
       <dane>                - ile_czytac_H:ile_czytac_L bajtow danych
```

UART – Przykładowe transmisje

Dokładny opis danych jest w dziale „**UART – Protokół transmisji**”.
Podane poniżej dane pochodzą z programu „Moduł karty SD – Tester” dostępnego na stronie xyzyk.pl.

Format przykładów jest wspólny:
```
Tx: <dane wysłane do modułu>
Rx: <dane odebrane z modułu>
Odp: <kod odpowiedzi już zdekodowany - tekst>
<i tutaj jakieś dane zależne od rozkazu>
```

Pobranie wersji programu:
```
Tx: 0x5A 0x01 0x00 0x00
Rx: 0xDA 0x00 0x03 0x00 0x0D 0x08 0x06 0x16 0x07 0x07
Odp: OK
ID programu: 3
Wersja:      0
Data asm:    2013.08.06 22:07:07
```

Zapytanie o status: (sytuacja: karta nie wsadzona)
```
Tx: 0x5A 0x04 0x00 0x00
Rx: 0xDA 0x00 0x00
Odp: OK
Status karty: brak karty
Status pliku: zamknięty
```

Zapytanie o status: (sytuacja: karta wsadzona i plik otwarty do zapisu)
```
Tx: 0x5A 0x04 0x00 0x00
Rx: 0xDA 0x00 0x81
Odp: OK
Status karty: karta wsunięta i zainicjalizowana
Status pliku: otwarty
```

Zapytanie o status: (sytuacja: karta wsadzona i plik otwarty, ale tylko do odczytu)
```
Tx: 0x5A 0x04 0x00 0x00
Rx: 0xDA 0x00 0xC1
Odp: OK
Status karty: karta wsunięta i zainicjalizowana
```

```
Status pliku: otwarty, tylko do odczytu
```

Pobranie ilości wolnego miejsca na karcie:

```
Tx: 0x5A 0x02 0x00 0x00
Rx: 0xDA 0x00 0x00 0xC0 0xEF 0x3B 0x00
Odp: OK
Wolne miejsce: 958,98 MB (1005568000 b)
```

Otwarcie pliku o nazwie „test.txt”:

```
Tx: 0x5A 0x10 0x08 0x00 0x74 0x65 0x73 0x74 0x2E 0x74 0x78 0x74
Rx: 0xDA 0x00
Odp: OK
```

Próba zapisania tekstu „XXX” do pliku który jest otwarty tylko do odczytu:

```
Tx: 0x5A 0x14 0x03 0x00 0x58 0x58 0x58
Rx: 0xDA 0x02
Odp: Błąd
```

Zapis tekstu „XXX” do pliku otwartego do zapisu:

```
Tx: 0x5A 0x14 0x03 0x00 0x58 0x58 0x58
Rx: 0xDA 0x00
Odp: OK
```

Motywy migania LEDami

W zależności od stanu modułu (jest karta, nie ma karty, jest ale tylko do odczytu, nieprawidłowy format karty) można ustalić sposób migania LEDów. Aby modulik w danej sytuacji przejął świecenie LEDem należy ustawić stosowne bity w opcjach LED_AUTO_R/G. Możliwość wymuszenia sposobu świecenia jest dostępna także przez UART, wtedy LED_AUTO_R/G musi być ustawione na 0.

Sposób w jaki moduł ma migać LEDami jest zadawany przez 10 bajtowy bufor, osoby dla każdego LEDa i każdej sytuacji. Bufor ten to nic innego jak stan LEDa w danych przedziałach czasowych (1 bit = 25ms; 1 bajt = 200ms) w okresie kolejnych 2 sekund. Program co 25ms przesuwa się o 1 bit (zaczyna od najmłodszego) w buforze i w zależności od jego wartości zapala lub gasi LEDa.

Przykłady:
(bufor zapisany w HEXie; spacje mięszy bajtami dodane dla czytelności)

```
 1  2  3  4  5  6  7  8  9 10        (numer bajtu – tylko dla czytelności)
00 00 00 00 00 00 00 00 00 00        (taki bufor ma same zera – LED będzie zawsze zgaszony)
FF FF FF FF FF FF FF FF FF FF        (same jedynki – LED świeci cały czas)
01 00 00 00 00 01 00 00 00 00        (jedynka w 1 i 6 bajcie oznacza mignięcie co sekundę)
FF 00 FF 00 FF 00 FF 00 FF 00        (co 200ms led zostanie zapalony na 200ms)
```

Przy wysyłce motywu przez UART wystarczy podać pierwszy bajt – zostanie on powielony na pozostałe. Tak łatwo można LEDa zapalić i zgasić: wystarczy wysłać odpowiednio **FF** lub **00**.
W przypadku czytania nastaw z pliku konfiguracyjnego wszystkie brakujące znaki zostaną zastąpione wartościami **00** (podając 10 bajtów, czyli 20 znaków w HEX, można wypełnić cały bufor, ale podanie np tylko ABCD spowoduje zapisanie tylko pierwszych dwóch bajtów bufora; reszta zostanie nadpisana zerami).

Opisy niektórych funkcji i pozostałe ważne informacje

Zapis danych przez SPI, linie sterujące

Moduł umożliwia szybki zapis danych przez interfejs SPI. Działa w trybie SLAVE i **MUSI zostać aktywowany (=zwarcie do masy) na linii CS**. Linia ta może być cały czas zwarta do masy. Należy tylko pamiętać że bez niej SPI w module nie zadziała!

Transmisja przez SPI jest jednokierunkowa i moduł nie może poinformować MASTERa o braku możliwości przyjęcia danych. W tym celu wykorzystana została linia „**USER**". Moduł zgłasza sygnałem aktywnym (=zwarciem do masy) możliwość odbioru danych. **Linia ta działa w ten sposób cały czas, nie tylko przy SPI.** Sygnał aktywny oznacza że jest jest otwarty plik z możliwością zapisu i że moduł ma miejsce w buforze na przyjęcie danych. Wyjęcie karty, zamknięcie pliku (np przez komendę wysłaną przez UART) czy inne okoliczności spowodują zmianę stanu linii.

AutoFlush

W trybie SPI moduł odbiera dane i zapisuje je na karcie. Aby zwiększyć wydajność dane te buforowane są w pamięci i zapisywane na karcie dopiero po zebraniu zoptymalizowanej ilości danych. Jednak takie działanie jest narażone na pewne niebezpieczeństwa utraty danych, np w wyniku wyjęcia karty podczas zapisu lub zaniku zasilania. W normalnej sytuacji dane z bufora zapisywane są na karcie po wysłaniu rozkazu (przez UART) **`FileWriteFlush`** lub **`FileClose`**. Jeśli jednak istnieje niebezpieczeństwo utraty danych można włączyć opcję AutoFlush.

Działanie tej opcji polega na wymuszeniu, co zadany czas bezczynności (czas od ostatniej transmisji), fizycznego zapisu bufora na kartę niezależnie od ilości danych w buforze. Skrajnym przypadkiem jest możliwość ustawienia zerowego czasu, co z jednej strony może znacząco spowolnić szybkość odbioru danych ale z drugiej zapobiega utracie danych nawet gdy użytkownik niespodziewanie wyciągnie kartę lub nastąpi utrata zasilania.

AutoFile

Opcja została wprowadzona aby maksymalnie uprościć integracje modułu z urządzeniem. Jej działanie jest proste – tworzy plik gdy tylko się da. Jeśli opcja ta jest włączona to moduł automatycznie tworzy nowy plik w momencie gdy jest taka możliwość (załączenie zasilania, włożenie karty bez blokady zapisu). Tym samym można zupełnie nie przejmować się transmisją sterującą przez UART tylko po pojawieniu się sygnału na linii **USER** (patrz „**Zapis danych przez SPI, linie sterujące**„) może od razu wysyłać dane. I wysłać może je zarówno przez UART (sytuacja jak przy normalnym zapisie do pliku) jak i przez SPI (jeśli jest włączone i aktywowane linią **CS**).
Opcja AutoFile posiada jedną opcję (patrz dział **Konfiguracja**): **`AutoFile_MaxSize`**. Jest to maksymalna wielkość pliku w bajtach po której osiągnięciu moduł założy kolejny plik. Maksymalna wartość to 1GB (1073741824 bajtów).

Uwagi:

- Jeśli tryb AutoFile jest włączony to moduł automatycznie próbuje stworzyć plik o ile żaden nie jest otwarty. W każdej chwili można, przy pomocy rozkazów wysłanych przez UART, zamknąć bieżący plik i otworzyć inny. Nawet w trybie tylko-do-odczytu. Po takiej operacji moduł nie będzie próbował stworzyć pliku aż do zamknięcia obecnego.
- Format nazw automatycznie tworzonych plików to „**d0000000.dat**", „**d0000001.dat**", „**d0000002.dat**", itd.
- Moduł zawsze tworzy plik którego nazwa jest wolna; zawsze (po załączeniu zasilania) zaczyna od **0** i inkrementuje wartość aż plik z danym numerem nie będzie istniał.

Wersje dokumentu

2013.08.06

- Pierwsze wydanie