Moduł Hosta USB, część 1

AVT–983

Cieszące się niezmiernie długo popularnością dyskietki 1,44 MB, których czytniki jeszcze do dziś są instalowane w komputerach, chyba powoli będą jednak odchodziły w zapomnienie. Do długotrwałego przechowywania danych wyparły je płytki CDROM i DVD, do chwilowego zapisania danych, np. w celu przeniesienia ich z komputera na komputer, już od dłuższego czasu służą pendrive'y. Wbudowana w nich pamięć półprzewodnikowa o pojemności dochodzącej do kilku gigabajtów nie daje szans dyskietce. Od niedawna, za sprawą układów Vinculum, pamięci pendrive można w bardzo prosty sposób wykorzystywać nawet w najprostszych systemach mikroprocesorowych.

Rekomendacje:

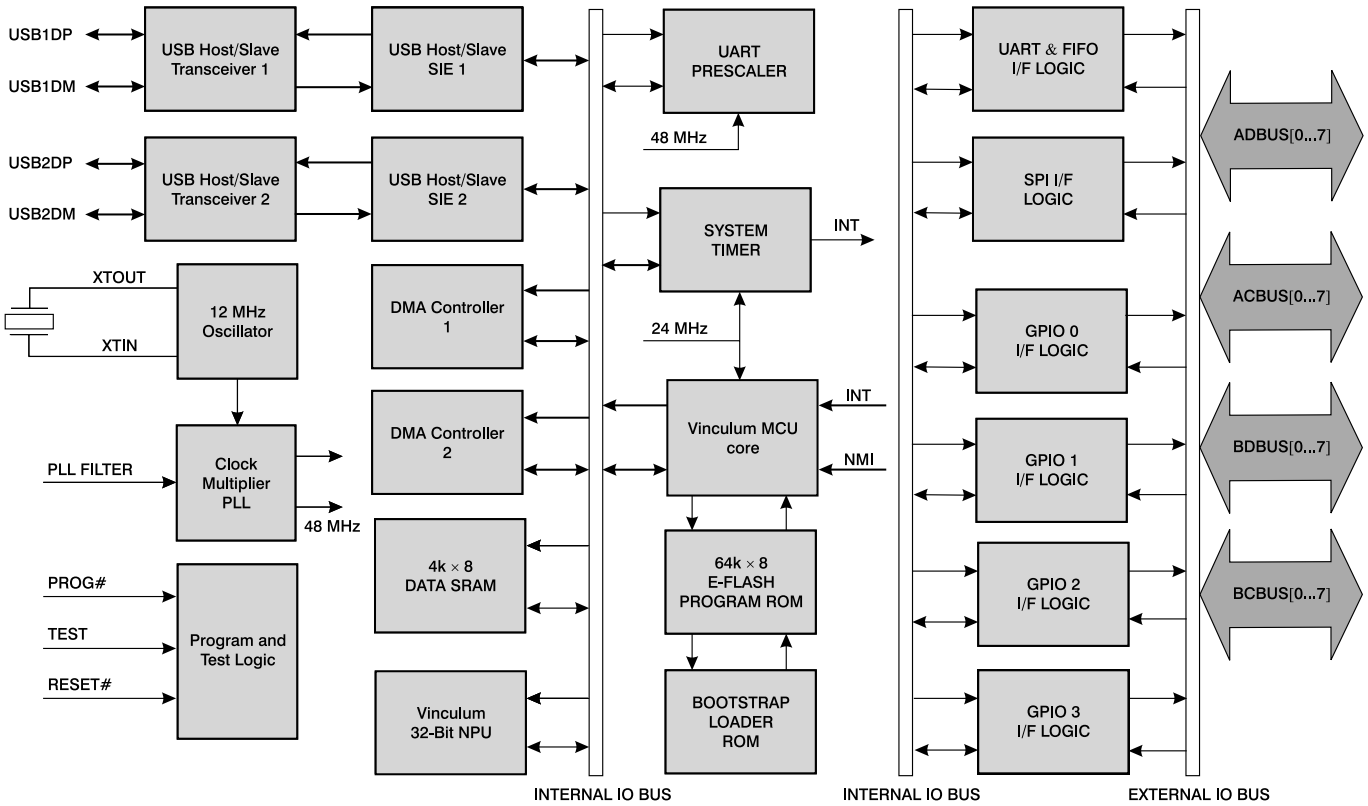
nowe układy rodziny Vinculum z pewnością szybko zdobędą popularność, tak jak stało się to w przypadku układów FT232 i FT245. Chętni ich stosowania nie mogą nie zrobić modułu Hosta USB.

PODSTAWOWE PARAMETRY

- Płytka o wymiarach 56x44 mm
- Zasilanie: +5 V, dostępne napięcie +3,3 V na pinach modułu
- Gniazdo: USB Host typu A
- Interfejsy: UART, SPI, FIFO wybierane dwoma zworkami, drugi interfejs USB dostępny na pinach modułu
- Wskazania o stanie modułu za pomocą diod LED
- Współpraca z pamięciami masowymi z systemem plików FAT
- Komunikacja za pomocą kilkunastu prostych komend przypominających komendy DOS

Większość dostępnych urządzeń pendrive wyposażono w interfejs USB. Tego typu tanie pamięci są idealne do zastosowań zarówno profesjonalnych, jak i amatorskich, w urządzeniach, które muszą gromadzić w nieulotnych pamięciach duże ilości danych. Obsługa takich pamięci za pomocą najprostszego mikrokontrolera jest od jakiegoś czasu możliwa dzięki układowi VNC1L produkowanemu przez znaną z konwerterów USB firmę FTDI. Układ VNC1L należy do nowej grupy układów nazywanych przez producenta Vinculum. Kontroler VNC1L pełni funkcje hosta USB. Za pomocą układu VNC1L (będącego pomostem pomiędzy pamięcią USB, a mikrokontrolerem) możliwa jest prosta obsługa urządzeń pamięci masowej (*Mass Storage*), czyli wszelkiego rodzaju dysków z interfejsem USB. Układ VNC1L umożliwia obsługę plików zapisanych w systemie FAT (FAT12, FAT16 oraz FAT32) i to za pomocą prostych komend. Dzięki zastosowaniu układu VNC1L można zmniejszyć koszt budowy i czas realizacji urządzeń, które mają umożliwiać dostęp do dysków USB. Poprzez możliwość zmiany oprogramowania układu VNC1L (wymagany prosty programator) można go dostosować do własnych potrzeb. Producent układu udostępnia obecnie kilka wersji oprogramowania (*firmware*). Umożliwiają one obsługę urządzeń pamięci masowej za pomocą interfejsu równoległego oraz szeregowych (SPI, UART, USB). Ponadto pamięci masowe można także obsługiwać za pomocą urządzenia USB Slave z rodziny FTDI, czyli za pomocą układów FT232 lub FT245.

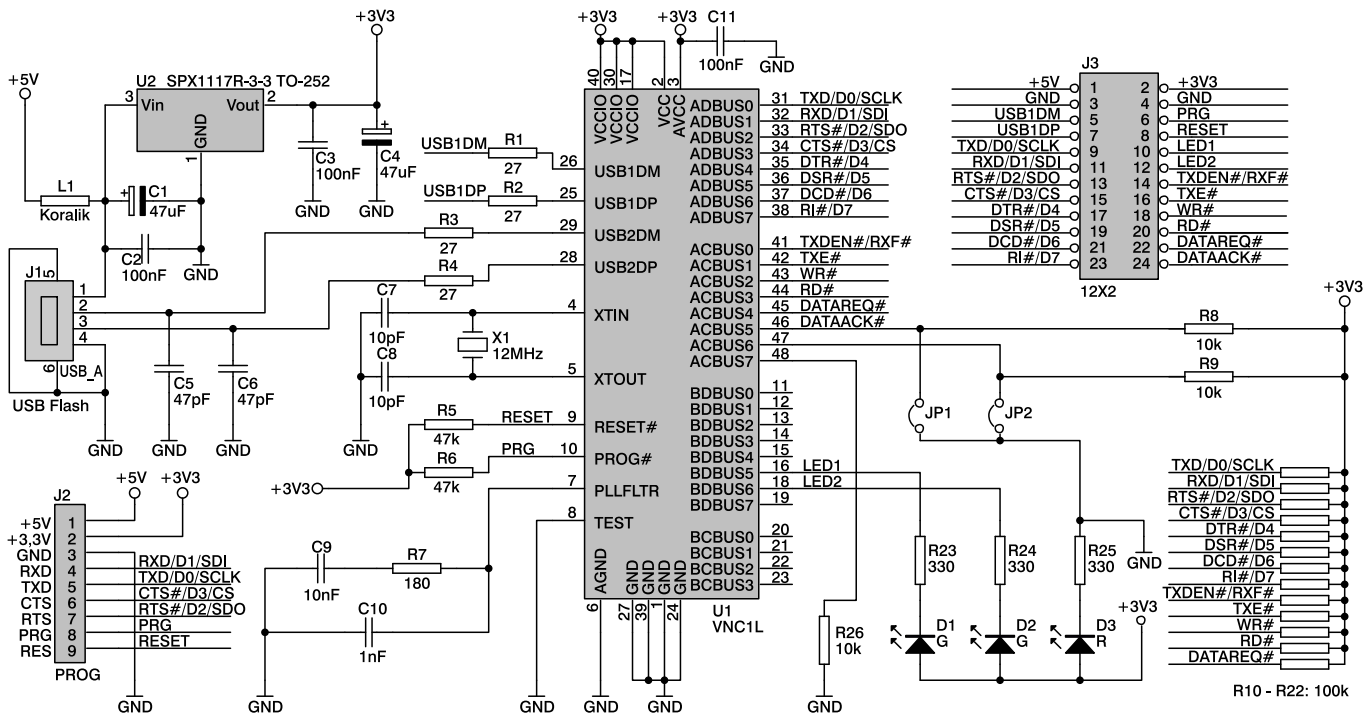
W artykule zostanie zaprezentowany uniwersalny moduł wykorzystujący układ VNC1L, który umożliwia dostęp do pamięci masowych za pośrednictwem już wymienionych interfejsów. Posiada on również złącze do podłączenia programatora, za pomocą którego można zmienić firmware. Programator, jak i sposób programowania zostanie przedstawiony w odrębnym artykule. Firmware można załadować za pomocą interfejsu UART, tak więc wspominany programator jest jedynie konwerterem USB<–>RS232 z wykorzystaniem układu FT232R. W artykule zostanie również zaprezentowana komunikacja z pamięcią, z wykorzystaniem komputerowego terminala, jak i mikrokontrolera komunikującego się za pośrednictwem interfejsu UART. Zostaną również przestawione przykładowe połączenia modułu do mikrokontrolera za pomocą pozostałych interfejsów SPI, równoległym FIFO oraz USB.



Rys. 1. Schemat blokowy układu VNC1L

Opis działania układu

Głównym układem modułu jest kontroler VNC1L, którego schemat blokowy pokazano na **rys. 1**. Wyróżnić w nim można dwa niezależne porty USB 2.0 pracujące z prędkościami Slow/Full Speed w konfiguracji Host/Slave, blok oscylatora z pętlą PLL, dwa kontrolery DMA, które poprawiają transmisję danych do interfejsów UART, FIFO, SPI z minimalnym udziałem mikrokontrolera. Układ posiada również 4 kB wewnętrznej pamięci SRAM, do której dostęp mają kontrolery DMA oraz mikrokontroler, mogący przechowywać w niej zmienne. Mikrokontroler układu VNC1L wykonuje operacje 8–bitowe, ale dodatkowy koprocesor arytmetyczny umożliwia szybkie obliczenia na danych 32–bitowych. Układ posiada wbudowany 8–bitowy rdzeń mikrokontrolera V–MCU, który jest oparty na architekturze



Rys. 2. Schemat elektryczny modułu Host USB

Tab. 1. Tryby pracy interfejsu

JP1	JP2	Tryb interfejsu
OFF	OFF	UART
ON	OFF	SPI
OFF	ON	FIFO
ON	ON	UART

Harvardzkiej (przestrzeń programu i danych są rozdzielone). Dla wbudowanego mikrokontrolera dostępna jest pamięć Flash o pojemności 64 kB. Pamięć Flash może być programowana z wykorzystaniem wbudowanego bloku *bootloadera* z wykorzystaniem interfejsu UART. Kontroler może być zasilany napięciem +3,3 V z tolerancją +5 V dla linii portów. Posiada on niski pobór prądu wynoszący 25 mA (w trybie Standby 2 mA). Schemat ideowy modułu został przedstawiony na **rys. 2**. Układ VNC1L potrzebuje do poprawnej pracy niewielu zewnętrznych komponentów. Pamięć z interfejsem USB jest dołączana do złącza J1. Linie drugiego portu USB, do którego można dołączyć np. FT232 lub FT245 zostały wyprowadzone na złącze J3. Układ VNC1L jest taktowany za pośrednictwem rezonatora kwarcowego X1 o częstotliwości 12 MHz. Elementy R5, R6 podciągają linie RESET oraz PROG do dodatniego napięcia zasilania. Linie te są wykorzystywane do uruchomienia wbudowanego *bootloadera*. Rezystor R26 informuje układ U1, że jest taktowany rezonatorem kwarcowym X1. Elementy R7, C9 i C10 współpracują z pętlą PLL układu U1. Układ U1 jest zasilany napięciem +3,3 V stabilizowanym przez U2. Dioda D1 sygnalizuje pracę urządzenia dołączonego do portu USB 1, a dioda D2 urządzenia dołączonego do portu USB 2. Dioda D3 sygnalizuje zasilanie modułu. Wykorzystując prosty programator można poprzez sygnały wyprowadzone na złącze J2 zapisać oprogramowanie do układu U1. Oprócz linii PROG i RESET, do programowania wykorzystywane są linie interfejsu UART. Zworki JP1 i JP2 umożliwiają wybór interfejsu, za pośrednictwem którego ma się odbywać komunikacja. W **tab. 1** pokazano możliwe konfiguracje zworek i zależne od ich ustawienia aktywne interfejsy. Linie interfejsów układu VNCIL zostały wyprowadzone na złącze J3. Dostępne są tam interfejsy USB, UART, SPI oraz równoległy FIFO. Przyporządkowane linie do danych interfejsów pokazano w **tab. 2**. Dodatkowo na złącze J3 zostały wyprowadzone linie zasilania. Moduł powinien być zasilany napięciem +5 V. W przypadku zasilania modułu napięciem +3,3 V, nie należy montować stabilizatora U2. Dodatkowe linie DATAREQ i DATAACK wskazują na tryb pracy (wysyłanie danych lub komend). Linie interfejsów zostały podciągnięte za pomocą rezystorów R10...R22 do dodatniego napięcia zasilającego.

WYKAZ ELEMENTÓW

Rezystory

R1...R4: 27 Ω SMD
R5, R6: 47 kΩ SMD
R7: 180 Ω SMD
R8, R9, R26: 10 kΩ SMD
R10...R22: 100 kΩ SMD
R23...R25: 330 Ω SMD

Kondensatory

C1, C4: 47 µF/16 V SMD
C2, C3, C11: 100 nF SMD
C5, C6: 47 pF SMD
C7, C8: 10 pF SMD
C9: 10 nF SMD
C10: 1 nF SMD

Półprzewodniki

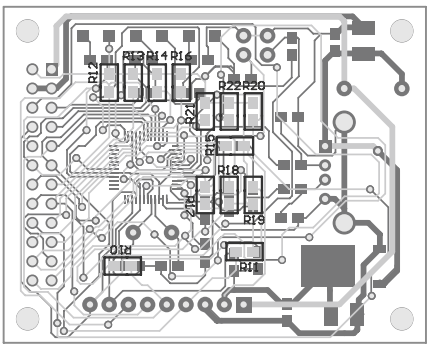
U1: VNCIL
U2: SPX1117R–3–3 TO–252
X1: Kwarc 12 MHz
D1, D2: LED SMD zielona
D3: LED SMD czerwona

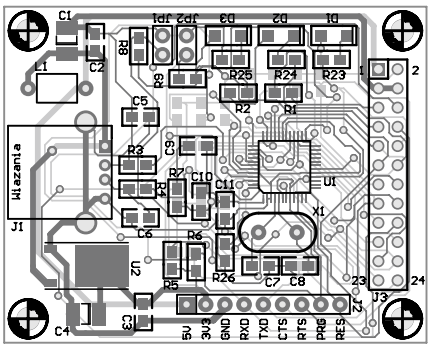
Inne

L1: koralik ferrytowy
J1: złącze USB A
J2: gniazdo na goldpin 1x9
J3: Goldpin 2x12
JP1, JP2: Goldpin 1x2 ze zworką

Tab. 2. Linie I/O dostępnych interfejsów

UART	Równoległy FIFO	SPI Slave
TXD	D0	SCLK
RXD	D1	SDI
RTS#	D2	SDO
CTS#	D3	CS
DTR#	D4	
DSR#	D5	
DCD#	D6	
RI#	D7	
TXDEN#	RXF#	
	TXE#	
	WR#	
	RD#	





Rys. 3. Schemat montażowy modułu Host USB

Montaż i uruchomienie

Schemat montażowy modułu przedstawiono na **rys. 3**. Zastosowano większość elementów w obudowach SMD. Są one montowane po obu stronach płytki. Największym problemem może być przylutowanie układu VNC1L. Wystarczy do tego celu lutownica z cienkim grotem, cyna o średnicy 0,25 mm i trochę ostrożności. Gdyby w trakcie lutowania zwarły się wyprowadzenia układu, można posłużyć się plecionką odsysającą. Złącze J3 należy przylutować od dolnej strony płytki, tak aby było możliwe umieszczenie modułu w innym urządzeniu. Moduł po poprawnym zmontowaniu (przy braku pomyłek w montażu) należy zasilić napięciem +5 V podanym na linię 1, masę dołączamy do linii 3 złącza J3. Przed rozpoczęciem użytkowania modułu należy go zaprogramować jednym z dostępnych *firmware*. Do zaprogramowania układu można wykorzystać programator, który będzie opisany w EP. Ma on już kompatybilne ze złączem J2 wyprowadzenia. Umożliwia również zasilenie modułu Hosta USB napięciem +5 V. Zaprogramowanie układu VNC1L jest bardzo proste przy użyciu programu VPROG (**rys. 4**). Należy wybrać jedynie programator oraz *firmware*. Dokładny przebieg

Tab. 3. Komendy modułu Hosta USB

Rozszerzone komendy ASCII przy pracy z terminalem	Skrócone (zapis szesnastkowy) komendy przy pracy z mikrokontrolerem	Funkcja komendy	Odpowiedź
Komendy przełączające pomiędzy komendami skróconymi, a rozszerzonymi			
„SCS"<cr>	$10,$0D	Włącza skrócony tryb komend	Zwraca znak zachęty „>",$0D informując, że urządzenie jest w trybie komend skróconych.
„ECS"<cr>	$11,$0D	Włącza rozszerzony tryb komend	Zwraca znak zachęty „D:\>",$0D informując, że urządzenie jest w trybie komend rozszerzonych.
„E"<cr>	„E"<cr>	Zwraca Echo	Zwrócony zostanie „E",$0D w celu synchronizacji.
„e"<cr>	„e"<cr>	Zwraca Echo	Zwrócony zostanie „e",$0D w celu synchronizacji.
Odpowiedzi wskazujące czy dysk jest włączony			
<cr>	$0D	Sprawdzenie czy dysk jest włączony	Zwrócony zostanie znak zachęty lub komunikat „no disk" dla wybranego trybu komend.
Odpowiedź sprawdzenia czy dysk jest włączony dla rozszerzonego trybu komend		Jeśli dysk nie znaleziony	„No Disk",$0D
		Jeśli dysk znaleziony	„D:\>",$0D
Odpowiedź sprawdzenia czy dysk jest włączony dla skróconego trybu komend		Jeśli dysk nie znaleziony	„ND",$0D
		Jeśli dysk znaleziony	„>",$0D
Operacje na katalogach			
„DIR"<cr>	$01,$0D	Wyświetla listę katalogów	Zostają zwrócone nazwy plików oraz katalogów. Każda nazwa jest kończona znakiem $0D. Katalog zawsze po nazwie ma znaki <sp>„DIR", ale przed znakiem $0D.
„DIR"<sp> <nazwa><cr>	$01,$20, <nazwa>,$0D	Wyświetla wielkość pliku o podanej nazwie. Wykorzystywany, aby wiedzieć ile danych odczytać z pliku.	$0D,<nazwa><sp><wielkość w hex(4 bajty), pierwszy LSB> $0D
„DLD"<sp> <nazwa><cr>	$05,$20,<nazwa>, $0D	Usuwa katalog	Usuwa katalog <nazwa> z wybranego katalogu. <ścieżka>$0D
„MKD"<sp> <nazwa><cr>	$06,$20, <nazwa>,$0D	Tworzy katalog	Tworzy nowy katalog <nazwa> w wybranym katalogu. <ścieżka>$0D
„CD"<sp> <nazwa><cr>	$02,$20,<nazwa> $0D	Umożliwia zmianę katalogu na nowy wybrany <nazwa>	<ścieżka>$0D
„CD"<sp>„.."<cr>	$02,$20,$2E,$2E,$0D	Wychodzi z katalogu	<ścieżka>$0D
Operacje na plikach			
„RD"<sp> <nazwa><cr>	$04,$20,<nazwa> $0D	Czyta plik <nazwa>	Ta komenda wysyła cały plik binarnie do monitora (terminal lub mikrokontroler). W pierwszej kolejności powinna zostać odczytana liczba bajtów w pliku używając komendy „DIR" <sp> <nazwa> <cr>. <ścieżka>$0D
„RDF"<sp> <liczba w hex (4 bajty)><cr>	$0B,$20, liczba w hex (4 bajty), $0D	Czyta dane <liczba w hex (4 bajty)> z aktualnie otwartego pliku.	Wysyła do monitora (terminal lub procesor) tylko wybraną liczbę danych. <ścieżka>$0D
„DLF"<sp> <nazwa><cr>	$07,$20,<nazwa> $0D	Usuwa plik <nazwa>	Usuwa plik z wybranego katalogu oraz zwalnia sektory FAT. <ścieżka>$0D
„WRF"<sp> <liczba w hex (4 bajty)><cr> <zapisywane dane w wybranej ilości><cr>	$08,$20, liczba w hex (4 bajty), $0D $dane,$0D	Zapisuje dane <liczba w hex (4 bajty)> do końca aktualnie otwartego pliku.	<ścieżka>$0D
„OPW"<sp> <nazwa><cr>	$09,$20, <nazwa>,$0D	Otwiera plik do zapisu za pomocą komendy „WRF"	<ścieżka>$0D
„OPR"<sp> <nazwa><cr>	$0E,$20, <nazwa>,$0D	Otwiera plik do odczytu za pomocą komendy „RDF"	<ścieżka>$0D
„CLF"<sp> <nazwa><cr>	$0A,$20, <nazwa>,$0D	Zamyka plik dla zapisu	<ścieżka>$0D

Tab. 3. Komendy modułu Hosta USB c.d.

Rozszerzone komendy ASCII przy pracy z terminalem	Skrócone (zapis szesnastkowy) komendy przy pracy z mikrokontrolerem	Funkcja komendy	Odpowiedź
„REN"<sp> <oryginalna nazwa> <sp> <nowa nazwa><cr>	$0C,$20, <oryginalna nazwa>,$20, <nowa nazwa> <cr>	Zmienia nazwę pliku lub katalogu	<ścieżka>$0D
„FS"<cr>	$12,$0D	Zwraca w bajtach ilość wolnego miejsca na dysku	<liczba wolnych bajtów w hex (4 bajty) pierwszy LSB> $0D
Komendy tylko przy pracy z interfejsem UART			
„SBD"<sp><dzielnik (3 bajty) pierwszy LSB><cr>	$14, $20,dzielnik (3 bajty) pierwszy LSB >,$0D	Szybkość transmisji danych (patrz tab. 6)	<ścieżka>$0D
Komendy zarządzania poborem mocy			
„SUD"<cr>	$15,$0D	Powoduje uśpienie dysku, gdy nie jest używany. Dysk będzie automatycznie budzony, gdy będzie do niego wysyłana komenda.	<ścieżka>$0D
„WKD"<cr>	$16,$0D	Wyprowadza dysk z uśpienia	<ścieżka>$0D
„SUM"<cr>	$17,$0D	Zawiesza pracę monitora i wyłącza zegar	<ścieżka>$0D
Pozostałe komendy			
„SD"<sp> <numer sektora w ASCII hex><cr>	$0,$20,...$0D	Zwalnia wybrany sektor. Używany do debatowania programu.	Wysyła 512 bajtów sektora wyszczególnionego w hex konwertowanego do ASCII. Każde 16 bajtów kończone jest znakiem $0D. <ścieżka>$0D
„IDD"<cr>	$0F,$0D	Identyfikuje dysk. Pokazuje informacje o dysku.	Wysyła dane bloku IDD i <ścieżke>$0D
„FWV"<cr>	$1,$0D	Pobiera wersje firmware	Pokazuje wersje głównego oprogramowania oraz reprogramowanego firmware VNC1L „MAIN x.xx"$0D „RPRG x.xx"$0D i <ścieżka>$0D
Komendy związane z FT232/FT245 dołączanymi do portu USB 1			
„FBD"<sp><dzielnik(3 bajty) pierwszy LSB><cr>	$18,$20,<divisor (3 bajty) pierwszy LSB>$0D	Szybkość transmisji danych (patrz tab. 6)	<prompt>$0D
„FMC"<sp><wartość (2 bajty) > <cr>	$19, $20,<wartość (2 bajty)>,$0D	Ustawienie kontroli dla sygnałów RTS/DTR (patrz tab. 7)	<prompt>$0D
„FSD"<sp><wartość (2 bajty) pierwszy LSB><cr>	$1A, $20,wartość (2 bajty) pierwszy LSB>,$0D	Ustawienie parametrów ramki danych (patrz tab. 8)	<prompt>$0D
„FFC"<sp><wartość (1 bajt)><cr>	$1B, $20,wartość (1 bajt),$0D	Ustawienie parametrów kontroli strumienia danych (patrz tab. 9)	<prompt>$0D
„FGM"<cr>	$1C,$0D	Pobranie statusu (patrz tab. 7)	Zwraca status (2 bajty),$0D
Gdzie: <sp> znak spacji			

programowania został przedstawiony przy opisie programatora układów Vinculum.

Tab. 4. Zwracane błędy

Błąd	Tryb komendy	Odpowiedź
Jeśli komenda nierozpoznana	Rozszerzone komendy	„Bad Command",$0D
	Skrócone komendy	„BC",$0D
Jeśli błąd wykonania komendy	Rozszerzone komendy	„Command Failed",$0D
	Skrócone komendy	„CF",$0D

Oprogramowanie i dostępne komendy

Dla układu VNC1L mającego komunikować się z mikrokontrolerem dostępne są dwa rodzaje oprogramowania. Oprogramowanie VDIF realizuje funkcje interfejsu Host USB umożliwiającego obsługę pamięci masowych z interfejsem USB. Dostęp do pamięci dołączanej do portu Host USB jest możliwy za pomocą interfejsów: UART, równoległego FIFO, SPI oraz z urządzenia z interfejsem USB, które pozwala wykonywać operacje na dołączonej pamięci USB (mogą to być telefony, PDA, MP3 itp.). Oprogramowanie VDIF przyjmuje, że pamięć USB będzie dołączana

Rys. 4. Okno programu VPROG służącego do programowania układów VNC1L

mocą komunikować się z dołączoną do portu USB 2 pamięcią USB. Moduł z tym oprogramowaniem umożliwi komunikację urządzeń już wyposażonych w interfejsy USB firmy FTDI. Do komunikacji z dołączoną pamięcią USB można wykorzystać komendy zapisane w kodach ASCII. Przypominają one komendy systemu DOS (DIR, CD, MKD, itp.). Są również dostępne skrócone komendy (zapisywane w kodzie szesnastkowym) do obsługi pamięci USB poprzez dołączony mikrokontroler. W dalszej części artykułu zostanie pokazana obsługa pamięci poprzez dostępne interfejsy również z wykorzystaniem terminala. Zostanie pokazany przykład założenia katalogu, w którym następnie tworzony będzie plik. Zostanie do niego zapisany przykładowy tekst, który następnie w celu weryfikacji będzie odczytywany. W **tab. 3** pokazano dostępne komendy w przypadku oprogramowania VDAP. Dostępne są komendy rozszerzone oraz skrócone, wykorzystywane przy komunikacji z mikrokontrolerem. Komendy można podzielić na kilka grup. Dostępne są komendy przełączające pomiędzy rozkazami rozszerzonymi a skróconymi, komendy operacji na katalogach, na plikach, wskazujące czy dysk USB jest włączony itp. Są również komendy związane z wybranym interfejsem. W przypadku interfejsu UART można wybrać jego prędkość transmisji. Dostępne są komendy związane z poborem mocy, komendy z układami FT232 i FT245 dołączanymi do drugiego portu USB. Komend nie ma wiele i są bardzo łatwe w użyciu. W **tab. 4** pokazano zwracane błędy podczas wystąpienia nierozpoznanej komendy lub braku jej wykonania. W **tab. 5** pokazano zwracane informacje o dołączonym dysku USB po wykonaniu komendy *IDD*. Dostępne są wszystkie podstawowe informacje jak system plików czy pojemność dysku. W **tab. 6** pokazano możliwe do wyboru prędkości interfejsu UART, jak i w przypadku wykorzystywania układu FT232. W **tab. 7...9** pokazano parametry bajtów związanych z układem FT232 dołączanym do drugiego portu USB. W przypadku oprogramowania VDIF nie ma komend związanych z układami FT232 i FT245.

W drugiej części artykułu zostaną przedstawione przykładowe, praktyczne sposoby wykorzystania układu VNC1L.

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

Tab. 5. Rezultat komendy IDD

IDD – Identify Disk Drive Results
„USB VID = $", 2 bajty w ASCII, $0D
„USB PID = $", 2 bajty w ASCII, $0D
„Vendor Id = ", 8 bajty w ASCII, $0D
„Product Id = ", 16 bajty w ASCII, $0D
„Revision Level = ", 4 bajty w ASCII, $0D
„I/F = „„SCSI" lub „ATAPI" w ASCII, $0D
„FAT12" lub „FAT16" lub „FAT32" w ASCII, $0D
„Bajty/Sector = $", 2 bajty w ASCII, $0D
„Bajty/Cluster = $", 3 bajty w ASCII, $0D
„Pojemność = $", 4 bajty w ASCII, $0D
„Wolna przestrzeń = $", 4 bajty w ASCII, $0D

Tab. 6. Prędkości interfejsu UART

Prędkość	Pierwszy bajt	Drugi bajt	Trzeci bajt
300	$10	$27	$00
600	$88	$13	$00
1200	$C4	$09	$00
2400	$E2	$04	$00
4800	$71	$02	$00
9600*	$8	$41	$00
19200	$9C	$80	$00
38400	$4E	$C0	$00
57600	$34	$C0	$00
115200	$1A	$00	$00
230400	$0D	$00	$00
460800	$06	$40	$00
921600	$03	$80	$00
1000000	$03	$00	$00
1500000	$02	$00	$00
2000000	$01	$00	$00
000000	$00	$00	$00
Uwaga: prędkość domyślna – 9600 bodów			

do portu USB 2, a pozostałe dostępne interfejsy będą służyć do komunikacji z dołączoną pamięcią USB. Dla opisywanego urządzenia bardziej odpowiednie będzie oprogramowanie VDAP, które różni się od oprogramowania VDIF tym, że do drugiego interfejsu USB można dołączyć układy Slave FTDI, takie jak FT232 lub FT245 i za ich po-

Tab. 7. Ustawienie kontroli dla sygnałów RTS/DTR dla FT232B lub FT232R

Pierwszy bajt	Operacje
Bit 0	DTR# Stan 0 = off, 1 = on
Bit 1	RTS# Stan 0 = off, 1 = on
Bits 7...2	Zarezerwowane „0"
Drugi bajt	Operacje
Bit 0	1 = zmiana DTR, 0 = brak zmiany DTR
Bit 1	1 = zmiana RTS, 0 = brak zmiany RTS
Bits 7...2	Zarezerwowane „0"

Tab. 8. Ustawienie parametrów ramki danych dla FT232B lub FT232R

Pierwszy bajt	Operacje
Bit 7...0	Liczba przesyłanych bajtów – 7 lub 8
Drugi bajt	Operacje
Bit 2...0	Bit parzystości: 0 – none 1 – odd 2 – even 3 – mark 4 – space
Bit 5...3	Liczba bitów Stop: 0 – 1 bit stopu 1 – 1 bit stopu 2 – 2 bity stopu
Bit 6	1 = Wysłanie break, 0 = Stop break
Bit 7	Zarezerwowane „0"

Tab. 9. Ustawienie parametrów kontroli strumienia danych dla FT232B lub FT232R

Pierwszy bajt	Operacje
Bit 0	Sprzętowy handshake RTS/CTS
Bit 1	Sprzętowy handshake DTR/DSR
Bity 2	Programowy handshake XOFF/XOFF
Bity 7...3	Zarezerwowane „0"

Moduł Hosta USB, część 2

AVT–983

Cieszące się niezmiernie długo popularnością dyskietki 1,44 MB, których czytniki jeszcze do dziś są instalowane w komputerach, chyba powoli będą jednak odchodziły w zapomnienie. Do długotrwałego przechowywania danych wyparły je płytki CDROM i DVD, do chwilowego zapisania danych, np. w celu przeniesienia ich z komputera na komputer, już od dłuższego czasu służą pendrive'y. Wbudowana w nich pamięć półprzewodnikowa o pojemności dochodzącej do kilku gigabajtów nie daje szans dyskietce. Od niedawna, za sprawą układów Vinculum, pamięci pendrive można w bardzo prosty sposób wykorzystywać nawet w najprostszych systemach mikroprocesorowych.

Rekomendacje:

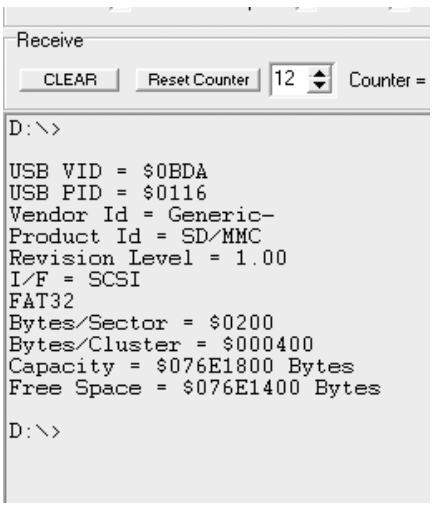
nowe układy rodziny Vinculum z pewnością szybko zdobędą popularność, tak jak stało się to w przypadku układów FT232 i FT245. Chętni ich stosowania nie mogą nie zrobić modułu Hosta USB.

PODSTAWOWE PARAMETRY

- Płytka o wymiarach 56x44 mm
- Zasilanie: +5 V, dostępne napięcie +3,3 V na pinach modułu
- Gniazdo: USB Host typu A
- Interfejsy: UART, SPI, FIFO wybierane dwoma zworkami, drugi interfejs USB dostępny na pinach modułu
- Wskazania o stanie modułu za pomocą diod LED
- Współpraca z pamięciami masowymi z systemem plików FAT
- Komunikacja za pomocą kilkunastu prostych komend przypominających komendy DOS

Komunikacja z wykorzystaniem komputerowego terminala

Stosując komendy rozszerzone, komunikację modułu Hosta USB z komputerowym terminalem można zrealizować w najprostszy sposób wykorzystując interfejs UART. Do tego celu niezbędny będzie programator układów Vinculum, który jak już wiemy, pełni jedynie funkcję konwertera USB<–>RS232. Zainstalowano dla niego sterowniki wirtualnego portu COM. Za pomocą programatora można podać napięcie zasilania dla modułu Hosta USB. Zgodnie z tab. 1, aby wybrać interfejs UART, zworki JP1 i JP2 mogą być założone lub zdjęte. Do portu USB został dołączony pendrive o pojemności 128 MB. Komunikacja będzie się odbywać za pomocą komend rozszerzonych. Domyślna prędkość transmisji wynosi 9600 bodów, 1 bit startu, 1 bit stopu, brak bitu parzystości z włączoną kontrolą transmisji za pomocą sygnałów RTS/CTS. RTS to linia żądania nadawania, a CTS to linia gotowości do wysłania danych. Aby zrezygnować ze sprzętowej kontroli transmisji, można linie RTS i CTS połączyć ze sobą. Wtedy do transmisji będą wykorzystywane tylko linie TXD i RXD. W terminalu należy ustawić identyczne parametry transmisji. Przy poprawnej komunikacji, po umieszczeniu dysku w złączu USB modułu, w oknie terminala powinien pojawić się znak zachęty *D:\>*. Po wysłaniu rozkazu *IDD* zosta-

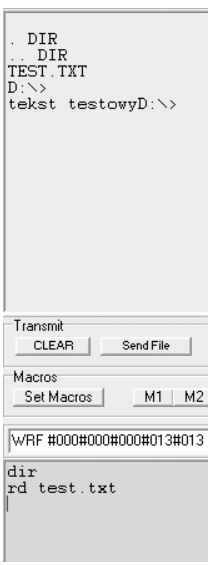


Rys. 5. Informacje o dysku wyświetlone po wysłaniu rozkazu *IDD*

Rys. 6. Informacja wyświetlona po założeniu katalogu *TEST*

Rys. 7. Informacja wyświetlona po założeniu do zapisu pliku *TEKST.TXT*

Rys. 8. Zapisanie danych do pliku *TEST.TXT* przy użyciu komendy WRF

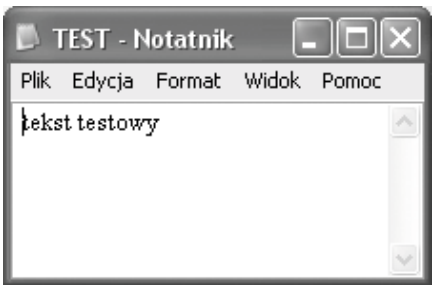


Rys. 9. Odczytanie zawartości pliku *TEST.TXT* przy użyciu komendy *RD*

ną wyświetlone informacje o dysku (**rys. 5**). Można odczytać z tych informacji pojemność dysku, wielkość sektora oraz typ systemu plików, w tym przypadku jest nim FAT32. W pierwszej kolejność w ramach testu został założony katalog *TEST* (**rys. 6**) z wykorzystaniem komendy *MKD*. Listę katalogów i plików można wyświetlić wysyłając komendę *DIR*. Następnie po wejściu do katalogu *TEST* (komendą *CD*) zostaje założony do zapisu plik *TEKST.TXT* (**rys. 7**). Do założenia pliku otwartego do zapisu wykorzystano komendę *OPW*. Do zapisu danych do pliku *TEST.TXT* wykorzystano komendę *WRF* (**rys. 8**), której jednym z parametrów jest informacja o liczbie zapisywanych bajtów. Składa się ona z 4 bajtów zapisanych szesnastkowo. Po zapisaniu do pliku przykładowego stringu *„tekst testowy”*, plik jest zamykany z wykorzystaniem komendy *CLF*. Do odczytu zawartości pliku *TEST.TXT* wykorzystano komendę *RD* (**rys. 9**). Jak można się przekonać, obsługa pamięci USB jest bardzo prosta. Zapisane pliki można również odczytywać umieszczając dysk USB w komputerze (**rys. 10**). W ten sposób można bez większych problemów odczytać dane za pośrednictwem komputera, po zapisaniu ich przez inne urządzenie.

Komunikacja z wykorzystaniem interfejsu UART

Na **rys. 11** przedstawiono przykładowy sposób podłączenia mikrokontrolera do interfejsu UART modułu Host USB. Z modułem komunikuje się mikrokontroler AVR ATmega88. Linie interfejsu RS232 zostały pokazane w **tab. 10**. Do komunikacji wykorzystano tylko linie RXD i TXD. Linie kontroli transmisji CTR i RTS zostały ze sobą połączone. Oczywiście jest możliwe, aby tymi liniami sterował mikrokontroler. Do mikrokontrolera został dołączony również wyświetlacz LCD, na którym będą prezentowane informacje z działania programu. Program sterujący mikrokontrolerem został napisany w Bascom AVR i będzie realizował identyczne operacje



Rys. 10. Odczytanie zawartości pliku *TEST.TXT* po włożeniu dysku USB w komputerze

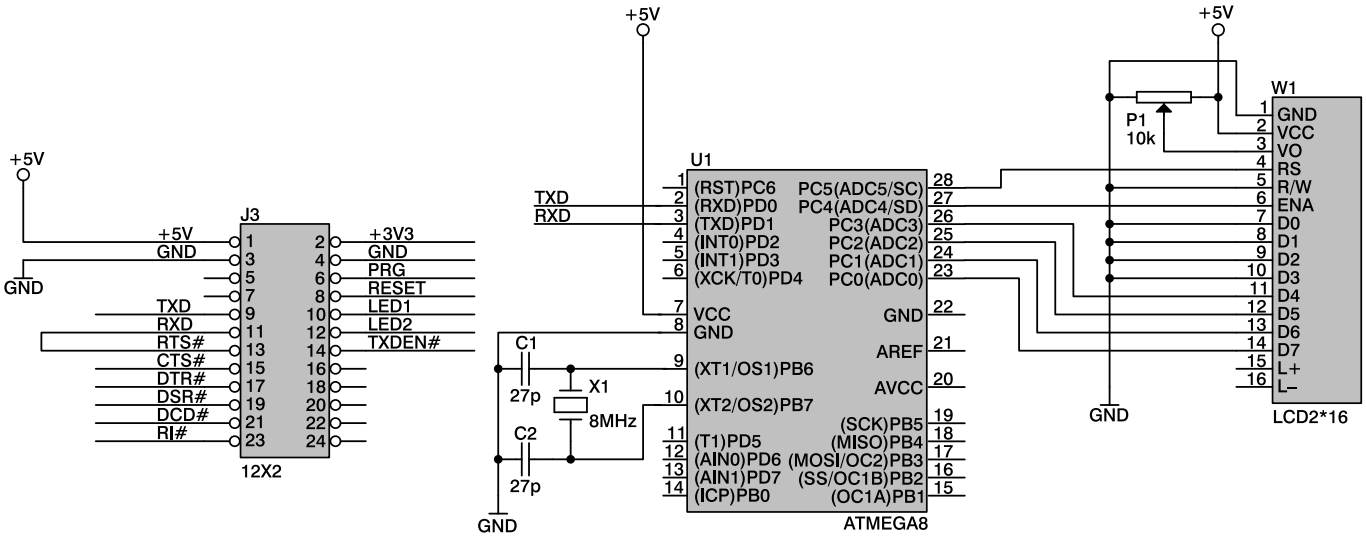
Tab. 10. Sygnały interfejsu UART

Nazwa	Typ	Opis
TXD	Wyjście	Dane wysyłane
RXD	Wejście	Dane odbierane
RTS#	Wyjście	Żądanie nadawania
CTS#	Wejście	Gotowość do wysyłania danych
DTR#	Wyjście	Gotowość do odbierania danych
DSR#	Wejście	Odbiornik gotowy do odbioru danych
DCD#	Wejście	Odbiór fali nośnej
RI#	Wejście	Wskaźnik wywołania
TXDEN	Wyjście	Odblokowuje transmisję przez RS485

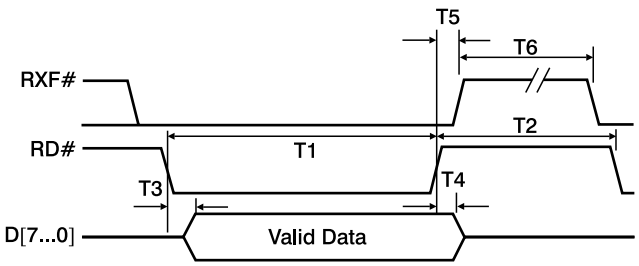
jak w przypadku testowania modułu poprzez terminal. Będzie więc zakładany katalog, a w nim plik, w którym zostaną zapisane dane. Następnie utworzony plik zostanie odczytany i jego zawartość zostanie wyświetlona na wyświetlaczu LCD. Aby moduł pracował z interfejsem UART, zgodnie z tab. 1 należy odpowiednio ustawić zworki JP1 i JP2. Do portu USB został dołączony również pendrive o pojemności 128 MB. Komunikacja będzie się odbywać za pomocą skróconych komend, zalecanych przy współpracy z mikrokontrolerem. Komendy skrócone są prostsze i sto-

Tab. 11. Sygnały interfejsu równoległego FIFO

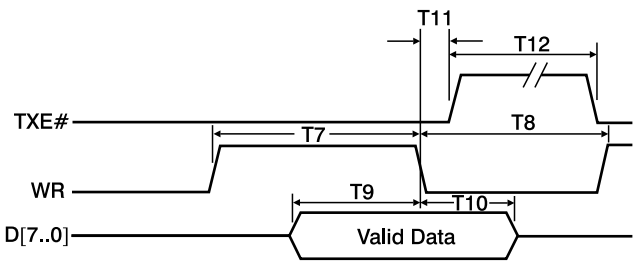
Nazwa	Typ	Opis
D0	We/Wy	Bit 0 magistrali danych D
D1	We/Wy	Bit 1 magistrali danych D
D2	We/Wy	Bit 2 magistrali danych D
D3	We/Wy	Bit 3 magistrali danych D
D4	We/Wy	Bit 4 magistrali danych D
D5	We/Wy	Bit 5 magistrali danych D
D6	We/Wy	Bit 6 magistrali danych D
D7	We/Wy	Bit 7 magistrali danych D
RXF#	Wyjście	Kiedy stan wysoki, nie można czytać danych z FIFO. Kiedy stan niski dane są w FIFO. Dane można odczytać strobując linię RD#.
TXE#	Wyjście	Kiedy stan wysoki, nie można zapisywać danych do FIFO. Kiedy stan niski można zapisywać dane do FIFO. Dane można zapisywać strobując linię WR.
WR#	Wejście	Dane z linii D0...D7 są zapisywane do FIFO kiedy sygnał WR zmienia stan z wysokiego na niski.
RD#	Wejście	Dane z FIFO pojawiają się na liniach D0...D7 kiedy sygnał RD# zmienia stan z wysokiego na niski.



Rys. 11. Przykładowy sposób dołączenia mikrokontrolera do interfejsu UART modułu Host USB

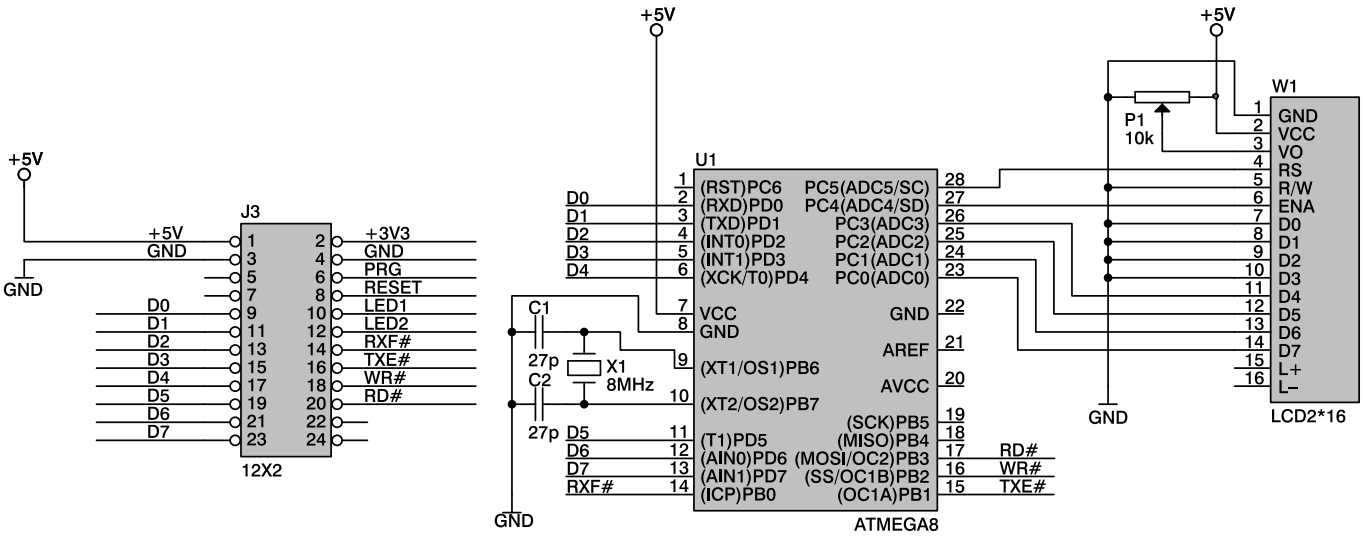


Rys. 12. Przebiegi występujące podczas odczytu danych z FIFO



Rys. 13. Przebiegi występujące podczas zapisu danych do FIFO

sując je przesyła się mniej danych. Domyślna prędkość transmisji wynosi 9600 bodów, 1 bit startu, 1 bit stopu, brak bitu parzystości, z włączoną kontrolą transmisji danych za pomocą sygnałów RTS/CTS. Również domyślnie wykonywany jest rozszerzony zestaw komend. Na **list. 1** przedstawiono program testowy komunikujący się z modułem Hosta USB. W programie wykorzystano buforową transmisję przez UART, zarówno danych odbieranych, jak i wysyłanych. W pierwszej kolejności oczekuje się na zgłoszenie się dysku USB (oczekiwany jest odbiór znaków *D:\>*). Jeśli dysk zostanie wykryty, wysyłany jest rozszerzony rozkaz *SCS*, który powoduje przełączenie modułu do komunikacji ze skróconymi komendami. Po jego wysłaniu, dysk USB będzie się zgłaszał znakiem „>”. Znak „>” będzie również świadczyć o poprawności wykonania danej komendy. Jego otrzymanie jest sprawdzane po każdej komendzie. Jeśli nie zostanie on otrzymany, zgłaszany jest błąd. W dalszej kolejności komunikacja z dyskiem odbywa się za pomocą instrukcji *Print* i *Input*. Zakładany jest katalog, następnie plik z danymi. Na końcu programu następuje odczyt zawartości pliku i jego wyświetlenie na wyświetla-



Rys. 14. Schemat dołączenia modułu Host USB do mikrokontrolera z wykorzystaniem interfejsu równoległego FIFO

czu LCD. Procedura *Odp* sprawdza poprawność wykonania komend. Jak widać założenie katalogu i pliku jest bardzo proste. Nie jest przy tym wymagana duża liczba operacji. Do obsługi dysku wystarczy nawet niewielki mikrokontroler, który nie musi posiadać dużo pamięci i wyprowadzeń.

Komunikacja z wykorzystaniem interfejsu równoległego FIFO

Komunikacja z wykorzystaniem portu równoległego FIFO znacznie przyspiesza transmisję danych w odniesieniu do interfejsów szeregowych. Interfejs równoległy wymusza jednak stosowanie kilkunastu linii mikrokontrolera. W module Host USB interfejs równoległy FIFO można aktywować, zakładając tylko zworkę JP2. W **tab. 11** pokazano opis linii interfejsu równoległego FIFO. Na **rys. 12** pokazano przebiegi podczas odczytu danych z FIFO. Nieodebrane dane są wskazywane niskim stanem linii RXF. Na **rys. 13** pokazano przebiegi podczas zapisu danych do FIFO. Zapełnienie FIFO danymi jest wskazywane stanem wysokim linii TXE. Interfejs FIFO modułu Host USB działa identycznie, jak interfejs FIFO układu FT245. Na **rys. 14** pokazano przykładowy schemat dołączenia modułu Host USB do mikrokontrolera z wykorzystaniem interfejsu równoległego FIFO.

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

List. 1. Program testowy wykorzystujący interfejs UART do komunikacji z pamięcią USB

```
'Program wykorzystujacy interfejs UART do komunikacji z pamiecia USB
'z wykorzystaniem ukladu VNCIL (Vinculum)
'Zaklozony zostaje katalog w ktorym nastepnie tworzony jest plik.
'Do pliku zapisywany jest tekst, ktory podczas odczytu wyswietlany jest na LCD
'Program mozna rowniez wykorzystac do komunikacji mikrokontrolera z pamiecia USB
'z wykorzystaniem interfejsu USB Slave jak FT232 lub FT245.
'Marcin Wiązania
'marcin.wiazania@ep.com.pl
$prog &HFF , &HFF , &HDF , &HF9                          'konfiguracja bitow FUSE
$regfile = „m88def.dat"                                  'informuje kompilator o pliku dyrektyw
wykorzystywanego mikrontrolera
$crystal = 8000000                                       'informuje kompilator o czestotliwo-
sci rezonatora kwarcowego
$baud = 9600                                             'informuje kompilator o predkosci
transmisji
$hwstack = 40                                            'konfiguracja wartosci stosow
$swstack = 40                                            'konfiguracja wartosci stosow
$framesize = 40                                          'konfiguracja wartosci stosow
Config Lcd = 16 * 2                                      'konfiguracja organizacji znakow wy-
swietlacza LCD
Config Lcdpin = Pin , Db4 = Portc.3 , Db5 = Portc.2 , Db6 = Portc.1 , Db7 = Portc.0 , E = Portc.4
, Rs = Portc.5       'konfiguracja pinow mikrokontrolara do ktorych dolaczone zostaly linie wy-
swietlacza
Config Serialin = Buffered , Size = 50                   'konfiguracja bufora wejsciowego
Config Serialout = Buffered , Size = 50                  'konfiguracja bufora wejsciowego
Declare Sub Odp                                          'procedura sprawdzania poprawnosci
wykonania komendy
Dim Znaki As String * 40                                 'definicja zmiennej znaki typu string
Wait 2                                                   'opoznienie 2 sekundy
Echo Off                                                 'wylaczenia echa dla instrukcji input
Enable Interrupts                                        'odblokowanie przerwan globalnych
Cursor Off                                               'wylaczenie kursora
Cls                                                      'czysc lcd
Lcd „Brak dysku"                                         'wyswietlenie komunikatu
Do                                                       'poczatek petli
  Input Znaki                                            'odczyt danych z UART
Loop Until Znaki = „D:\>"                                'petla wykonywana az odebrane zostana
znaki D:\>
Print „SCS" ; : Print Chr(&H0d);                         'przelacznie na skrocony zestaw ko-
mend
Call Odp                                                 'sprawdzenie wykonania komendy
Cls                                                      'czysc lcd
Lcd „Dysk aktywny"                                       'wyswietlenie komunikatu
Wait 1                                                   'opoznienie 1 sekundy
Cls                                                      'czysc lcd
Lcd „MKD TEST"                                           'wyswietlenie komunikatu
Print Chr(&H0d);                                         'wyslanie znaku CR w ramach testu
Call Odp                                                 'sprawdzenie wykonania komendy
Print Chr(&H06) ; : Print Chr(&H20) ; : Print „TEST" ; : Print Chr(&H0d);       'zalozenie kata-
logu TEST
Call Odp                                                 'sprawdzenie wykonania komendy
Wait 1                                                   'opoznienie 1 sekundy
Cls                                                      'czysc lcd
Lcd „CD TEST"                                            'wyswietlenie komunikatu
Print Chr(&H02) ; : Print Chr(&H20) ; : Print „TEST" ; : Print Chr(&H0d);       'wejscie do kata-
logu TEST
Call Odp                                                 'sprawdzenie wykonania komendy
Wait 1                                                   'opoznienie 1 sekundy
Cls                                                      'czysc lcd
Lcd „OPW TEST.TXT"                                       'wyswietlenie komunikatu
Print Chr(&H09) ; : Print Chr(&H20) ; : Print „TEST.TXT" ; : Print Chr(&H0d);       'zalozenie
pliku do zapisu TEXT.TXT
Call Odp                                                 'sprawdzenie wykonania komendy
Wait 1                                                   'opoznienie 1 sekundy
Cls                                                      'czysc lcd
Lcd „WRF TEST.TXT"                                       'wyswietlenie komunikatu
Print Chr(&H08) ; : Print Chr(&H20) ; : Print Chr(&H00) ; : Print Chr(&H00);       'zapis do pli-
ku TEXT.TXT
Print Chr(&H00) ; : Print Chr(&H0d) ; : Print Chr(&H0d);     'tekstu majacego 13 znakow o tresci
Print „tekst testowy" ; : Print Chr(&H0d);               'tekst testowy
Call Odp                                                 'sprawdzenie wykonania komendy
Call Odp                                                 'sprawdzenie wykonania komendy
Wait 1                                                   'opoznienie 1 sekundy
Cls                                                      'czysc lcd
Lcd „CLF TEST.TXT"                                       'wyswietlenie komunikatu
Print Chr(&H0a) ; : Print Chr(&H20) ; : Print „TEST.TXT" ; : Print Chr(&H0d);       'zamkniecie
pliku TEXT.TXT do zapisu
Call Odp                                                 'sprawdzenie wykonania komendy
Wait 1                                                   'opoznienie 1 sekundy
Cls                                                      'czysc lcd
Lcd „RD TEST.TXT"                                        'wyswietlenie komunikatu
Print Chr(&H04) ; : Print Chr(&H20) ; : Print „TEST.TXT" ; : Print Chr(&H0d);       'odczyt za-
wartosci pliku TEXT.TXT
Input Znaki                                              'odczyt danych z UART
Lowerline                                                'druga linia LCD
Lcd Left(znaki , 13)                                     'wyswietlenie tesktu odczytanego
z pliku TEXT.TXT
Home                                                     'pierwsze linia LCD
Do                                                       'nieskonczona petla do-loop
Loop                                                     'koniec nieskonczonej petli
End                                                      'koniec programu
Sub Odp                                                  'podprogram sprawdzania poprawnosci
wykonania komendy
  Input Znaki                                            'odczyt danych z UART
  If Znaki <> „>" Then                                   'jesli znak inny niz > to
    Cls                                                  'czysc LCD
    Lcd „Blad komendy"                                   'wyswietlenie komunikatu
    Do                                                   'nieskonczona
    Loop                                                 'petla do-loop
  End If
End Sub                                                  'koniec procedury
```