Dział "Projekty Czytelników" zawiera opisy projektów nadesłanych do redakcji EP przez Czytelników. Redakcja nie bierze odpowiedzialności za prawidłowe działanie opisywanych układów, gdyż nie testujemy ich laboratoryjnie, chociaż sprawdzamy poprawność konstrukcji.

Prosimy o nadsyłanie własnych projektów z modelami (do zwrotu). Do artykułu należy dołączyć podpisane ***oświadczenie, że artykuł jest własnym opracowaniem autora i nie był dotychczas nigdzie publikowany****. Honorarium za publikację w tym dziale wynosi 250,- zł (brutto) za 1 stronę w EP. Przysyłanych tekstów nie zwracamy. Redakcja zastrzega sobie prawo do dokonywania skrótów.*

Sprzętowy emulator mikrokontrolerów z rodziny PIC

Projekt 060

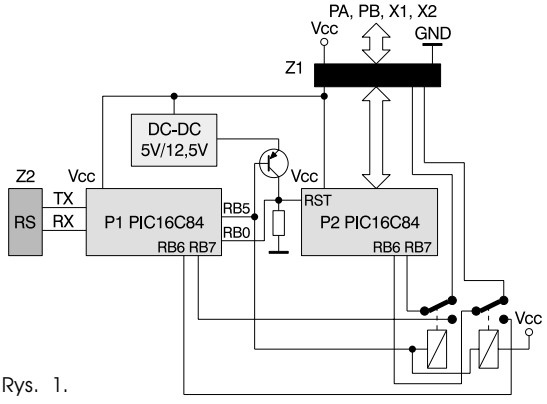
Czy próbowaliście uruchamiać układ mikroprocesorowy mając do dyspozycji asembler i symulator programowy? Wszyscy, którzy odpowiedzieli pozytywnie na to pytanie wiedzą ile godzin pracy zabierają poprawki programu, który początkowo wydawał się tak elegancko napisany. Bywa, że końcowym wynikiem naszej pracy jest stwierdzenie, że algorytm oprogramowania jest nieprawidłowy lub sprzętowe zasoby zastosowanego mikrokontrolera są niewystarczające. W takim przypadku nasza praca trafi do kosza.

Aby przyspieszyć proces uruchamiania systemów mikroprocesorowych niezbędny jest sprzętowy emulator mikrokontrolera. Urządzenie to umożliwia łatwe wprowadzanie poprawek do emulowanego mikrokontrolera i testowanie programu w czasie rzeczywistym.

Jeden z mikrokontrolerów z serii PIC dysponuje pamięcią reprogramowalną elektrycznie i możliwością programowania łączem szeregowym. Pozwala to na zbudowanie urządzenia zdolnego do emulacji wielu mikrokontrolerów z rodziny PIC16CXX.

Opis sposobu działania urządzenia

Emulator składa się z dwóch mikrokontrolerów (PIC16C84 lub PIC16F84): P1 i P2 (**rys. 1**).



Rys. 1.

Niestety, w chwili powstawania emulatora nie miałem do dyspozycji mikrokontrolera PIC16F84 i nie mogłem przetestować poprawności działania urządzenia z tym mikrokontrolerem. Sądząc jednak po jego opisie technicznym, emulator powinien działać poprawnie mając do dyspozycji więcej pamięci danych RAM.

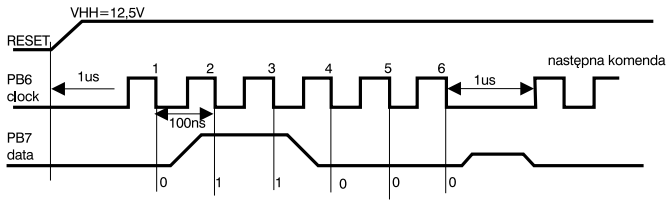
Mikrokontroler P1 odbiera dane w formacie INHX8M, za pośrednictwem łącza szeregowego, z komputera IBM PC, dekoduje je do postaci binarnej i programuje mikrokontroler P2 łączem szeregowym SPI (RB6, RB7).

Mikrokontroler P2 emuluje działanie uruchamianego mikrokontrolera. Jest on połączony z układem uruchamianym za pośrednictwem złącza Z1, które wtyka się w miejsce docelowego mikrokontrolera. Za pośrednictwem tego złącza emulator jest zasilany z obwodów układu uruchamianego. Przetwornica DC-DC z 5V na 12,5V zapewnia zasilanie oraz inicjację trybu programowania mikrokontrolera P2.

Na **rys. 2** przedstawiono sposób szeregowego progra-

Tab. 1. Polecenia szeregowego programowania mikrokontrolerów PIC.

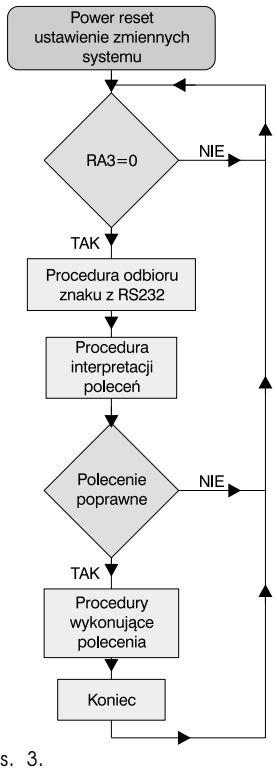
L.p.	Polecenie	Format polecenia MSB.. LSB	Dane
1	Ładuj konfigurację mikrokontrolera	0 0 0 0 0 0	0,dane (14),0
2	Ładuj dane do pamięci programu	0 0 0 0 1 0	0,dane (14),0
3	Czytaj dane z pamięci programu	0 0 0 1 0 0	0,dane (14),0
4	Inkrementacja adresu	0 0 0 1 1 0	
5	Rozpocznij programowanie	0 0 1 0 0 0	
6	Ładuj dane do pamięci danych	0 0 0 0 1 1	0,dane (14),0
7	Czytaj dane z pamięci danych	0 0 0 1 0 1	0,dane (14),0
8	Kasuj zawartość pamięci programu	0 0 1 0 0 1	
9	Kasuj zawartość pamięci danych	0 0 1 0 1 1	



Rys. 2.

mowania mikrokontrolerów PIC16C84. Przed programowaniem wyprowadzenie RB0 mikrokontrolera P1 wymusza sygnał reset mikrokontrolera P2. Następnie, wyprowadzeniem RB5 mikrokontrolera P1 wymusza przyłączenie poprzez przekaźniki wyprowadzeń RB6, RB7 mikrokontrolera P2 do odpowiednich wyprowadzeń mikrokontrolera P1. Na wyprowadzeniach tych był uprzednio ustawiony stan niski.

Załączenie przekaźników powoduje jednocześnie wyprowadzenie napięcia programującego VHH=12,5V na wejście RESET mikrokontrolera P2, powodując wprowadzenie go w tryb programowania. Po upływie minimum 1µs możemy przesłać do mikrokontrolera P2 polecenia związane z procesem programowania. Polecenia przesyłane są począwszy od najstarszego bitu. Wszystkie polecenia są sześciobitowe (**tab. 1**).



Rys. 3.

Proces programowania

W tab. 1 przedstawiono polecenia dostępne podczas programowania szeregowego. Są one - mam nadzieję - czytelne i nie wymagają komentarza. Wyjaśnienia wymaga natomiast format przesyłania danych.

Po poleceniach związanych z odczytem i zapisem danych następuje przesyłanie danych. Dane z pamięci programu mikrokontrolerów rodziny PIC16XX są czternastobitowe, dane pamięci danych ośmiobitowe. Podczas procesu odczytu-zapisu danych wymagane jest wygenerowanie szesnastu taktów zegara, lecz pierwszy i ostatni takt w wypadku odczytu są jałowymi, a w wypadku zapisu do mikrokontrolera wymaga się, aby był wpisywany bit 0. Stąd pochodzi forma zapisu danych „0,dane(14),0“. Podczas zapisu i odczytu z pamięci danych uwzględniamy tylko pierwsze osiem bitów, mimo tego taktować musimy 16 bitów.

Cechą szczególną mikrokontrolerów rodziny PIC jest to, że programujemy nie tylko sposób działania mikrokontrolera, lecz również jego konfigurację. Możemy zaprogramować rodzaj zegara systemowego, watchdog, zablokować dostęp do odczytu naszego programu. Wyczerpującą literaturę, dane techniczne, algorytmy programowania, wiele przykładowych procedur oraz podstawowe oprogramowanie uruchomieniowe znajdziesz drogi Czytelniku pod adresem *http://www.microchip.com*. Dlatego nie będę tutaj opisywał algorytmów programowania konfiguracji oraz pamięci danych i programu zwłaszcza, że firma prezentuje wiele różnych algorytmów poprawiających pewność i trwałość zapisu.

Proces programowania pamięci programu przebiega następująco:
- inicjalizacja poprzez RB6=0 i RB7=0, a następnie RESET=VHH (w tym momencie zerowany jest licznik programu),
- polecenie „Ładuj dane do pamięci programu“,
- ładuj dane,
- polecenie „Rozpocznij programowanie“,
- czekaj 10ms,
- polecenie „Czytaj dane z pamięci programu“,
- czytaj dane,
- weryfikacja poprawności zapisu,
- polecenie „Inkrementacja adresu“,
- powtarzamy powyższe polecenia aż do wyczerpania adresów.

Proces programowania pamięci danych przebiega analogicznie, lecz występują polecenia dotyczące pamięci danych.

Programowanie konfiguracji mikrokontrolera wiąże się z programowaniem specjalnego obszaru pamięci, który jest wywoływany poleceniem „Ładuj konfigurację“ i przesłaniem pod adresem (0x2007H) słowa konfiguracji.

Podczas procesu programowania szeregowego nie musimy kasować uprzednio zapisanych informacji.

Oprogramowanie emulatora

Program obsługi emulatora wykonuje następujące zadania:
- programowa emulacja łącza szeregowego RS232 19200,N,8,1,
- interpretacja przychodzących danych i poleceń,
- konwersja odebranych danych z formatu INHX8M na binarny,
- komunikacja z mikrokontrolerem emulującym program,
- zapis i odczyt danych programu oraz konfiguracji mikrokontrolera emulującego.

Interesującym jest fakt, że udało się zmieścić w pamięci operacyjnej mikrokontrolera dosyć złożone oprogramowanie. Należy to przypisać niezwykle efektywnemu kodowi mikrokontrolerów rodziny PIC. Drugą ciekawostką jest możliwość uzyskania stosunkowo dużej prędkości komunikacji (19200bit/s), ze sprzętową obsługą tego procesu: RTS-CTS, przy wolnym zegarze taktującym mikrokontroler - częstotliwość zaledwie 4MHz.

Omówienie wszystkich procedur programu emulatora i programu komunikacyjnego, ze względu na ich liczbę, raczej nie wchodzi w rachubę. Muszę się ograniczyć do przedstawienia algorytmu oprogramowania mikrokontrolera emulatora. Myślę, zdając się na dociekliwość Czytelników, że przedstawienie algorytmu i źródeł będzie wystarczające do zrozumienia działania emulatora.

Program komunikacyjny napisałem w języku PDS 7.1. Jest to bardzo rozbudowany QBASIC. Dlaczego w QBASICU? Otóż dlatego, że można w nim napisać bardzo szybko dobre oprogramowanie. Oczywiście, każdy język jest tak dobry, jakie są dostępne do niego biblioteki. Biblioteki piszę sam w asemblerze i aby skompilować mój program trzeba z nich skorzystać.

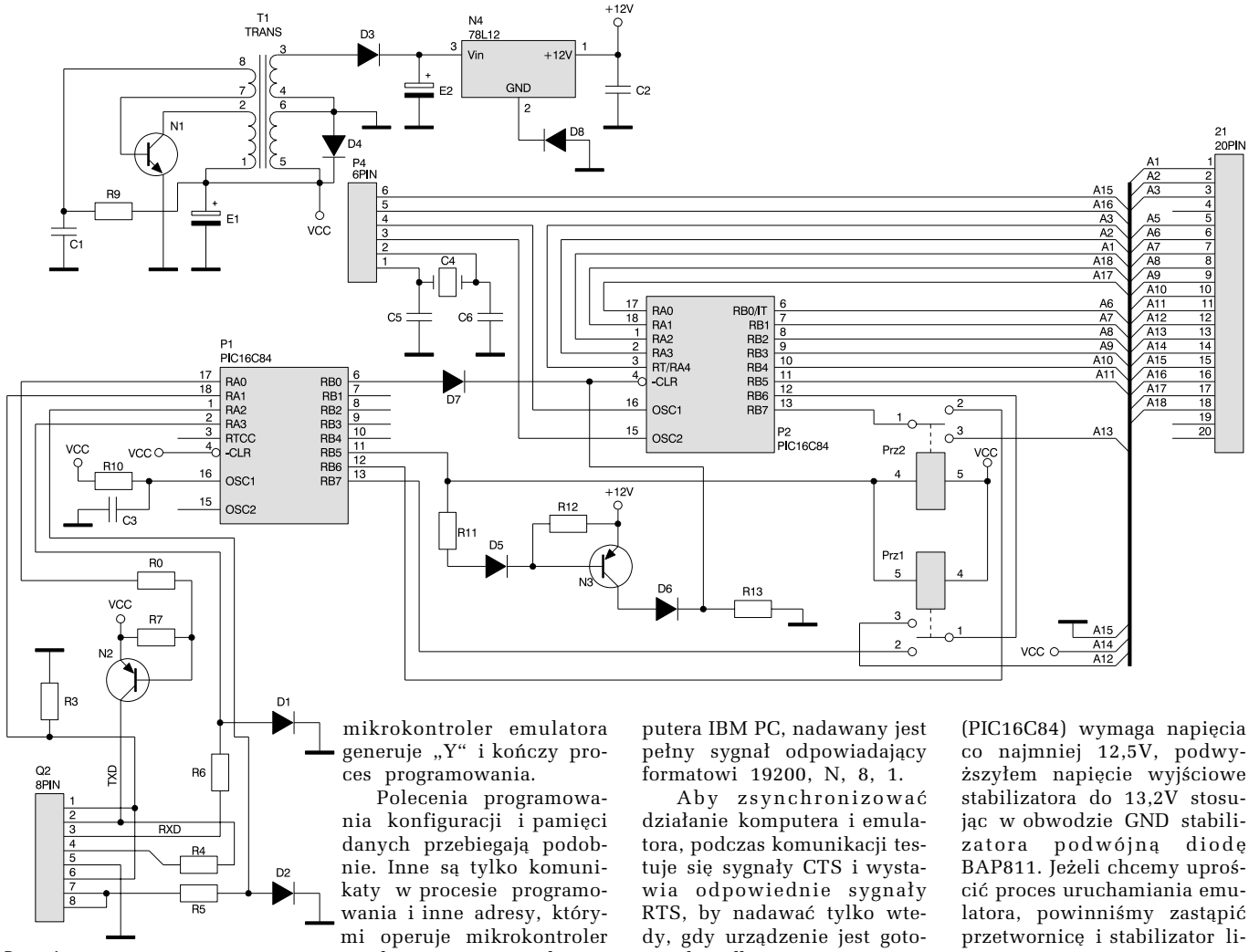
Oprogramowanie mikrokontrolera emulatora czyta polecenia z komputera IBM PC, nadawane za pośrednictwem programu komunikacyjnego i wykonuje szereg operacji.

Polecenia stanowią pojedyncze litery ASCII:
- „H“ - ładuj program,
- „I“ - czytaj program,
- „J“ - czytaj konfigurację,
- „L“ - czytaj dane,
- „M“ - pisz dane,
- „Y“ - koniec danych.

Przykładowo, emulator po odebraniu polecenia „H“ wykonuje inicjalizację trybu programowania mikrokontrolera emulującego, przełącza odpowiednio przekaźniki i czeka pewien czas na ustalenie się styków. Następnie oczekuje znaku „:“. Przesłanie dowolnego innego znaku lub zakłócenie powodujące zmianę informacji przesłanej powodują koniec procesu programowania i wygenerowanie przez emulator komunikatu o błędzie programowania.

Następny bajt informuje o ilości danych „LL“, które będą przesłane. Ta informacja jest ważna, ponieważ przesyłane rekordy mogą maksymalnie mieć 16 bajtów, lecz mogą być krótsze. Dzięki tej danej wiemy kiedy kończy się rekord i odebrany bajt nie jest daną do zapisu, lecz sumą kontrolną.

Następnie jest przesyłany adres „AAAA“ informujący, gdzie w pamięci programu mają być umiejscowione dane, potem typ danych „00“oraz dane „DD“. Ostatnim bajtem jest ośmiobitowa liczba „SS“ liczona jako zero minus suma wszystkich poprzedzających danych za wy-



Rys. 4.

mikrokontroler emulatora generuje „Y“ i kończy proces programowania.

Polecenia programowania konfiguracji i pamięci danych przebiegają podobnie. Inne są tylko komunikaty w procesie programowania i inne adresy, którymi operuje mikrokontroler emulatora. Program komunikacyjny nie zawiera obsługi programowania pamięci danych.

Na **rys. 3** przedstawiono ogólny algorytm programu. Program testuje stan pinu RA3 mikrokontrolera PIC16C84. Jeżeli wystąpi niski poziom logiczny, program przechodzi do obsługi procedury łącza szeregowego. Odebrany znak jest interpretowany i weryfikowana jest jego poprawność. W zależności od odebranego znaku wykonywane są procedury omówione powyżej.

Procedura obsługi łącza szeregowego po wykryciu niskiego stanu RA3 bada poprawność sygnału START. Procedura jest wykonywana, jeżeli sygnał START trwa odpowiednio długo. Po poprawnym odebraniu sygnału START procedura próbkuje stan pinu RA3 w odpowiednich momentach osiem razy. Procedura nie bada bitu kontroli parzystości i sygnału stopu.

W trakcie nadawania przez mikrokontroler do komputera IBM PC, nadawany jest pełny sygnał odpowiadający formatowi 19200, N, 8, 1.

Aby zsynchronizować działanie komputera i emulatora, podczas komunikacji testuje się sygnały CTS i wystawia odpowiednie sygnały RTS, by nadawać tylko wtedy, gdy urządzenie jest gotowe do odbioru.

jątkiem znaku „:“. Wszystkie elementy pliku HEX są typu znakowego. Suma odnosi się do wartości jaką reprezentują.

„ : L L A A - AA00DDDDDDDDDDDDDDDDSS“ - format „HEX“

Po poprawnym odebraniu całego rekordu i umiejscowieniu go w pamięci danych (DAN0..DAN15), dane są kolejno konwertowane do postaci binarnej i zapisywane w pamięci mikrokontrolera emulującego. Jeżeli suma kontrolna odebranych danych jest nieprawidłowa generowany jest błąd sumy kontrolnej „A“.

Po dokonaniu zapisu jest on weryfikowany z danymi odebranymi. Jeżeli zapis jest poprawny, generowany jest znak sukcesu „S“ i mikrokontroler emulatora oczekuje kolejnego rekordu. W przeciwnym wypadku generowany jest komunikat „G“ - błąd zapisu. Po odebraniu ostatniego rekordu „:000001FF“, jeżeli wszystkie poprzednie operacje przebiegły pomyślnie,

Rozwiązania sprzętowe emulatora

Emulator został zaprojektowany z myślą o maksymalnej prostocie rozwiązań sprzętowych (schemat na **rys. 4**). Niestety, wiąże się z tym dosyć uciążliwy proces uruchamiania sprzętu.

Powielacz napięcia wykonano w oparciu o generator astabilny z elementów N1 i T1. Uzwojenie pierwotne 1-2 transformatora zawiera 50 zwojów, uzwojenie wtórne 3-4 180 zwojów, uzwojenie sprzęgające 7-8 10 zwojów, uzwojenie zwrotne 5-6 50 zwojów. Całość nawinięto na rdzeniu kubkowym o średnicy 12 mm, o AL=340. Uruchomienie generatora wymaga odpowiedniego wyfazowania uzwojeń. Napięcie z przetwornicy jest prostowane i stabilizowane stabilizatorem N4 zbudowanym z układu 78L12. Ponieważ mikrokontroler do inicjalizacji procesu programowania (PIC16F84) i do procesu programowania (PIC16C84) wymaga napięcia co najmniej 12,5V, podwyższyłem napięcie wyjściowe stabilizatora do 13,2V stosując w obwodzie GND stabilizatora podwójną diodę BAP811. Jeżeli chcemy uprościć proces uruchamiania emulatora, powinniśmy zastąpić przetwornicę i stabilizator li-

WYKAZ ELEMENTÓW

Rezystory
R3, R4, R5, R6, R7, R8, R9, R10, R11, R12, R13: 4,3kΩ

Kondensatory
C1: 4,7nF
C2, C3, C5, C6: 22pF
E1, E2: 4,7µF/6,3V

Półprzewodniki
P1, P2: PIC16C84
D1, D2, D4, D6, D7: 1N4148
D3: 1N4002
D5: DZ6,2V
D8: BAP811 Z1,2V
stabilizator 1,2V
N1: BC239
N2, N3: BC309
N4: UA78L12CP

Różne
P4: złącze 6-pinowe
Q2: złącze 9-pinowe (RS232)
20PIN: złącze emulacyjne
Prz1, Prz2: przekaźniki Matsushita HD1-M-12DC
T1: transformator wg opisu
C4: dobierany oscylator kwarcowy

niowy nowoczesnym, impulsowym stabilizatorem podwyższającym napięcie.

W obwodzie RB6 i RB7 mikrokontrolera emulującego P2 zastosowałem dwa miniaturowe, jednostykowe przekaźniki firmy Matsushita typu HD1-M-DC12, które jak się okazało sprawnie pracują również przy napięciu zasilającym 5V.

Standard RS232 wymaga dla linii transmisyjnych napięcia bipolarnego o amplitudzie od 3 do 15V [1]. Praktyka wykazała, że ogromna większość łączy w komputerach IBM PC zadowala się dodatnim napięciem unipolarnym. Możemy więc zrezygnować z tranzystora N2 i rezystora R4, pamiętając aby wprowadzić w oprogramowaniu mikrokontrolera negację bitów nadawanych. Amplituda sygnałów nadawanych na poziomie 4,5V w zupełności wystarcza do transmisji sygnału na odległość do 2m.

Zegar mikrokontrolera emulatora P1 zbudowałem stosując generator RC. Stabilność takiego generatora wystarcza do utrzymania łączności w warunkach laboratoryjnych. Niestety wiąże się z tym konieczność dokładnego dobrania R i C generatora tak, aby otrzymać wartość częstotliwości zegara 4MHz. Wartość tę musimy zmierzyć na pinie OSC2 mikrokontrolera P1 i powinna wynosić 1MHz.

Sposób posługiwania się emulatorem

Po podłączeniu emulatora do gniazda *com1* lub *com2* wydajemy polecenie *picemul <nazwa pliku hex>*. Powinniśmy również zadbać o konfigurację mikrokontrolera. Parametry konfiguracji mikrokontrolera wpisujemy dowolnym edytorem tekstowym do pliku „econfig.dat“. Format konfiguracji powinien być dla osób znających rodzinę PIC zupełnie jasny. Emulator posiada złącze P4 służące do ustawienia zegara mikrokontrolera. Należy zwrócić uwagę, aby sprzętowa konfiguracja generatora zegara była zgodna z ustawieniem w pliku „econfig.dat“. Kwarc generatora zegara znajduje się w specjalnym gniazdku. Możemy łatwo wymienić rezonator na pożądany przez emulowany układ. Najistotniejszą uwagą jest ta, aby nie uruchamiać urządzeń prototypowych na sprzęcie ostatniej generacji.

Janusz Raniszewski

Literatura

1. Wojciech Mielczarek „Szeregowe interfejsy cyfrowe“ wyd. Helion 1993
2. Microchip Technology Inc. „PIC16/17 data book“ 1995/1996
3. Atmel Corporation „AVR enhaced RISC microcontroler data book“ 1997

Źródła

1. http://www.microchip.com
2. http://www.atmel.com