Emulator mikroprocesora 87C51, część 1

PROJEKT Z OKŁADKI

kit AVT-288

Każdy, kto chociaż raz budował urządzenie oparte na popularnym mikroprocesorze 87C51 (8751), wie jak żmudnym i czasochłonnym zajęciem jest wielokrotne weryfikowanie pisanego programu, poprzez kasowanie procesora promieniami UV i ponowne jego programowanie. Przedstawione urządzenie eliminuje te wady i umożliwia szybką, a co więcej pełną emulację 8751.

Wstęp

O popularności mikrokontrolerów z rodziny 8051 nie trzeba nikogo przekonywać. W handlu od kilku lat znajduje się szeroka gama tych uniwersalnych układów, począwszy od protoplasty rodziny układu 8051, aż po wyrafinowane wersje z obniżonym napięciem zasilającym, interfejsem UART, I2C, przetwornikiem A/C, wyjściami typu PWM (ang. „Pulse Width Modulation" - wyjścia o programowanej częstotliwości sygnału i jego wypełnieniu) i wiele innych. Wspólna ich cecha - język programowania, assembler oraz kompatybilność „w dół", powodują że procesory te zdominowały rynek mikrokontrolerów jednoukładowych. Generalnie producenci oferują trzy wersje handlowe rodziny '51, a mianowicie:

- układy bez wewnętrznej pamięci programu - tzw. ROMless,
- układy z wewnętrzną pamięcią programu typu EPROM,
- układy typu OTP (ang. „One Time Programmable" - programowalne tylko jeden raz). Są to tańsze wersje układów z pamięcią EPROM, nie wyposażone jednak w kwarcowe okienko, które umożliwia kasowanie uprzednio zaprogramowanej pamięci.

Najtańsze wersje wymagają do pracy podłączenia zewnętrznej pamięci programu w postaci pamięci EPROM. Traci się przy tym dwa ośmiobitowe porty (16 uniwersalnych wyprowadzeń), co często powoduje, że trzeba układ rozbudowywać o dodatkowe układy wejścia/wyjścia.

Wersja druga układów z wewnętrzną pamięcią EPROM, umożliwia wykorzystanie wszystkich portów procesora, jednak jej wysoka cena jest powodem tego, iż często amatora-elektronika po prostu nie stać na taki układ.

Kompromisem między pierwszą i drugą wersją są układy OTP. Ich cena jest niższa od układów z pamięcią EPROM, należy jednak pamiętać, że taki układ można zaprogramować tylko jeden raz. Najczęściej jednak pierwsza wersja programu w assemblerze nie jest ostateczna, zawsze znajdą się tam jakieś braki lub błędy. Toteż zaprogramowanie układu niedopracowaną wersją programu kończy się wyrzuceniem kosztownego mikroprocesora „do kosza".

Przedstawione w artykule urządzenie pozwala na emulację, czyli „udawanie" procesora typu 8751 (87C51). Nie jest przy tym zajęte żadne z jego wyprowadzeń. Specjalistyczne firmy oferują urządzenia tego typu, pozwalające nawet na pracę krokową procesora i pełny „debugging" programu, jednak ich cena.... Tańsze wersje takich urządzeń często zajmują jedno wyprowadzenie układu (najczęściej związane z przerwaniem zewnętrznym INT0), co ogranicza możliwość pełnej emulacji procesora.

Głównym celem przedstawionego rozwiązania była niska cena emulatora, przy jednoczesnym zapewnieniu pełnej emulacji 8751.

Emulator do pracy potrzebuje dowolnego komputera klasy PC ze standardowym portem szeregowym. Wbudowany w urządzenie interfejs RS-232C pozwala na transmisję emulatora z komputerem PC poprzez standardowy kabel RS (9 lub 3-żyłowy). Nad pracą urządzenia czuwa program komunikacyjny na komputer PC dołączany na dyskietce wraz z emulatorem.

Nie przypadkowo na wstępie użyłem słowa „pełnej" emulacji, bowiem układ pozwala na pracę także z zewnętrznym rezonatorem kwarcowym o częstotliwości z przedziału 3,5...12 MHz. Możliwe jest rozszerzenie tej granicy do 16 MHz, trzeba jednak zastosować inną wersję głównego elementu urządzenia: procesora 80C451 z oznaczeniem „80C451CGA". Ta ważna cecha uniezależnia konstruktora od konieczności przystosowywania pisanych programów do częstotliwości zegara pracującego w emulatorze.

Prócz tego emulator umożliwia prace z wykorzystaniem wewnętrznego interfejsu RS-232C, a ściślej mówiąc, konwertera napięć TTL/RS232, zbudowanego na popularnym układzie MAX232, co w przypadkach budowy urządzeń wykorzystujących transmisję szeregową w standardzie RS-232C z komputerem PC zwalnia konstruktora od częstego przełączania wtyczki kabla RS z emulatora na uruchamiany układ. Tak więc np. możliwe jest uruchomienie projektowanego układu bez zamontowanego w nim układu translacji TTL/RS232.

Kolejną ważną opcją emulatora jest tryb pracy kontrolera 8751 z zewnętrzną pamięcią danych o rozmiarze maksymalnie do 64kB, a więc pokrywającym całą przestrzeń adresową procesora. Z punktu widzenia emulatora i pisanego programu, nie ma znaczenia czy potrzebne jest np. 8kB tej pamięci, czy więcej. Zawsze jest dostępne 64kB. Oczywiście dostępny jest tryb pracy procesora przy pełnym adresie, kiedy to pamięć danych jest adresowana poprzez instrukcje MOVX @DPTR,A (MOVX A,@DPTR), jak też w trybie stronicowania, gdy używamy instrukcji np. MOVX @R0,A kiedy procesor wystawia tylko młodszą część adresu do portu P0 procesora, nie zmieniając portu P2. Niewykorzystane do adresowania pamięci piny portu P2 są oczywiście dostępne dla programisty i mogą być dowolnie modyfikowane.

Wspomniana pamięć danych jest wbudowana w emulator toteż w tym trybie porty P0 i P2 oraz piny /WR i /RD nie są dostępne na wtyku emulacyjnym, a uruchamiany docelowo układ może być czasowo pozbawiony „kości" SRAM.

Procesor 80C451

Jest to rozszerzona wersja popularnego układu 8051. Procesor ten zawiera wewnątrz dokładnie to samo co jego protoplasta, z tą różnicą, że posiada 7 dwukierunkowych portów wejścia/wyjścia. Dodatkowo jeden z nich (port P6) może być wykorzystany do dwukierunkowej transmisji równoległej np. z drukarką (Centronics), bądź z innym procesorem 80C451 przy wymianie danych w układach zawierających kilka procesorów '451. Jest to jedyny procesor serii '51, posiadający wystarczającą ilość portów do emulacji 8751. W przedstawionym układzie porty P0 i P2 układu 80C451 współpracują z programem komunikacyjnym emulatora zapisanym w EPROMie oraz pamięciami typu SRAM. W trakcie emulacji porty te zostają „zastąpione" przez porty P5 i P6. Dodatkowy port P4 wykorzystywany jest do ustawienia konfiguracji pracy emulatora.

Opis układu

Przed przystąpieniem do szczegółowego opisu układu i jego działania wprowadźmy dwa określenia trybu pracy emulatora:

- tryb „LOAD", tryb pracy, w którym emulator komunikuje się z komputerem PC, umożliwiając tym samym przesłanie kodu programu użytkownika, ustalenie konfiguracji pracy emulatora oraz wiele innych przydatnych funkcji, które zostaną opisane później;
- tryb „EMU", tryb emulacji, w którym procesor wykonuje program użytkownika załadowany wcześniej do pamięci emulatora.

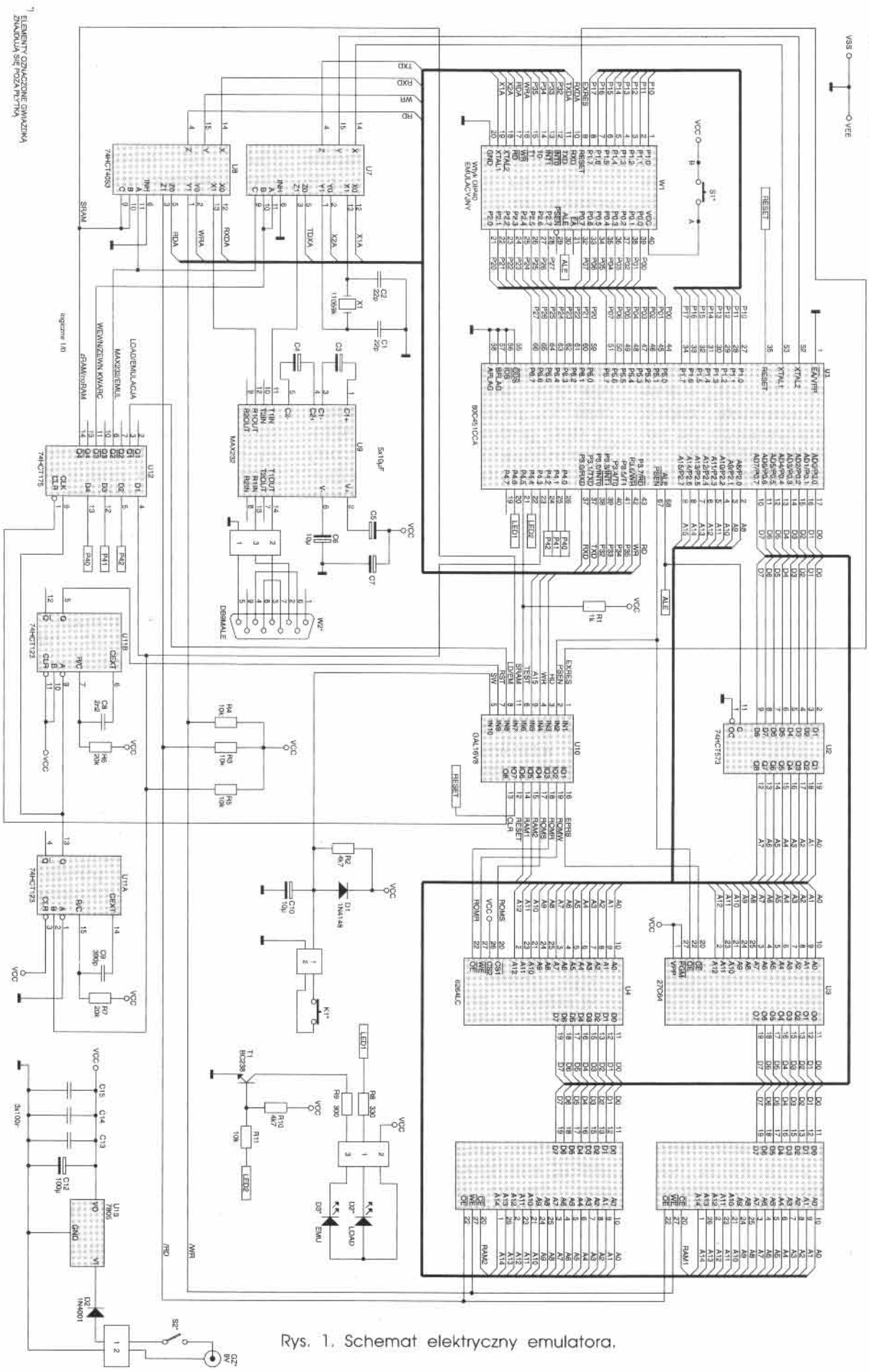
Schemat ideowy urządzenia przedstawia **rys.1**. Procesor U1 współpracuje z pamięcią EPROM U3 w której znajduje się program obsługi emulatora w trybie LOAD, oraz dwoma blokami pamięci SRAM. Pierwszy blok pamięci, układ U4, to 8kB pamięci programu użytkownika. Co prawda procesory 8751 zawierają tylko 4kB pamięci wewnętrznej, ale ta „nadwyżka" umożliwia „niepełną" emulację także procesora 8752 (bez wykorzystania trzeciego timera T2), który posiada pamięć o pojemności 8 kB. Drugi blok składa się z układów U5 i U6 o łącznej pojemności 64kB, wykorzystywanej przy pracy w trybie z zewnętrzną pamięcia danych. Układ U2 zatrzaskuje młodszą część adresu wystawianego na szynie danych procesora.

Kolejnym układem jest U10, którego zadaniem jest przełączanie bloków pamięci po przejściu układu do pracy w tryb emulacji. Ponadto układ steruje sygnałami zerowania procesora: wewnętrznym, klawisz K1, jak i zewnętrznym dostępnym poprzez sondę W1 z zewnętrznego, uruchamianego układu. Wykorzystano tu programowany układ logiczny w postaci GALía 16V8, bardzo popularnego na rynku. Kombinacyjny układ logiczny wpisany w jego strukturę zaoszczędza zastosowanie kilku dodatkowych układów scalonych TTL, które musiałyby go zastąpić.

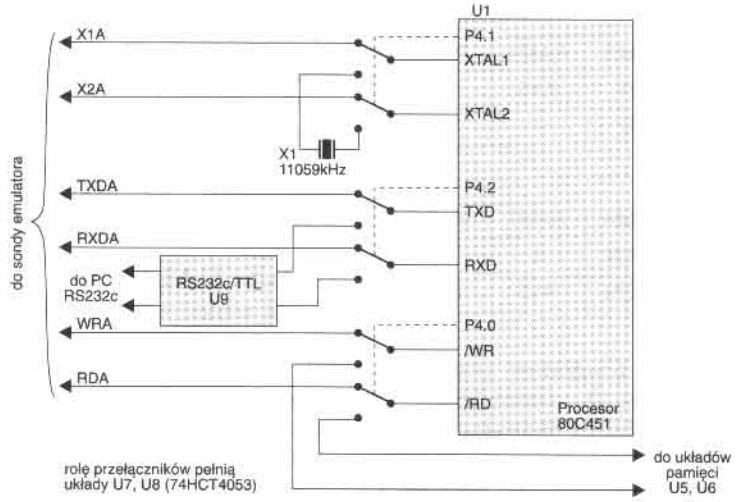
Układy U7, U8 to przełączniki analogowe przełączające wyprowadzenia TXD, RXD, /WR, /RD oraz rezonator kwarcowy procesora U1 na wtyk sondy emulacyjnej W1, bądź na wewnętrzne układy emulatora, zależnie od ustawionego trybu pracy. Jak wynika z definicji, emulator powinien udostępniać wszystkie sygnały procesora 8751. Te ostatnie, wymienione wyżej są wykorzystywane do komunikacji z komputerem PC, toteż dzięki tym przełącznikom, w trybie emulacji użytkownik ma możliwość wyprowadzenia ich na zewnątrz. Rozwiązanie tego problemu przedstawiono na **rysunku 2.**

Układ U9 przetwarza sygnały transmisji szeregowej z poziomów TTL na poziomy standardu RS-232C i odwrotnie. Dzięki temu poprzez gniazdo W2, za pośrednictwem standardowego kabla RS, możliwa jest współpraca emulatora z komputerem PC. Układ U12 (4-krotny zatrzask z wyjściami komplementarnymi) odpowiada za „zapamiętanie" konfiguracji pracy układu po przejściu w tryb emulacji.

Podwójny uniwibrator U11A, U11B generuje odpowiednie sygnały zapisu konfiguracji w układzie U12 oraz zeruje procesor U1. Zespół K1, D1, R2, C10 jest



Rys. 1. Schemat elektryczny emulatora.



Rys. 2. Sposób dublowania wejść sygnałów procesora.

obwodem kasowania emulatora (przejście z trybu EMU w tryb LOAD), a R8...R11 oraz T1 sterują diodami LED D2 i D3. Świecenie diody D2 - zielonej oznacza, że układ znajduje się w trybie „LOAD", natomiast świecenie D3 - czerwonej sygnalizuje tryb emulacji.

Opis działania

Po włączeniu zasilania emulatora ładowany jest kondensator C10 co poprzez układ U10 wymusza stan wysoki na wejściu RESET procesora U1 . Jednocześnie na wyjściu CLR U10 panuje wtedy stan niski kasując układ U12. Na wyjściach U12: 3,6,11,14 ustala się poziom wysoki przełączając w ten sposób za pośrednictwem układów U7,U8 :

- obwód rezonatora kwarcowego X1,C1,C2 do końcówek 52,53 procesora U1,
- obwód interfejsu szeregowego U9 do końcówek TXD,RXD układu U1
- wejścia zapisu/odczytu zewnętrznej pamięci danych (układy U5 i U6): /WE,/OE do końcówek /RD i /WR U1.

Wyzerowanie procesora ustawia porty P1, P5, P6 i P4 w stan wysokiej impedancji (jako wejściowe). Wymuszony przez rezystor R14 stan wysoki na wejściu TEST układu U10 blokuje wejścia wyboru układu /CE układów U5, U6 tym samym uniemożliwiając ich odczyt. Odblokowany zostaje zaś układ U4 (/CS1=0) umożliwiając zapis programu użytkownika, który będzie później emulowany. Na tym etapie, U4 „widziany" jest przez procesor U1 jako zewnętrzna pamięć danych o adresach: 0000h-1FFFh. Na wejściu wyboru /CE układu U3 panuje stan niski, procesor zaczyna wykonywanie programu komunikacyjnego zawartego w EPROMie U3. Emulator znajduje się w trybie LOAD.

Rysunek 3 wyjaśnia „konfigurację" pamięci emulatora w obydwu trybach jego pracy.

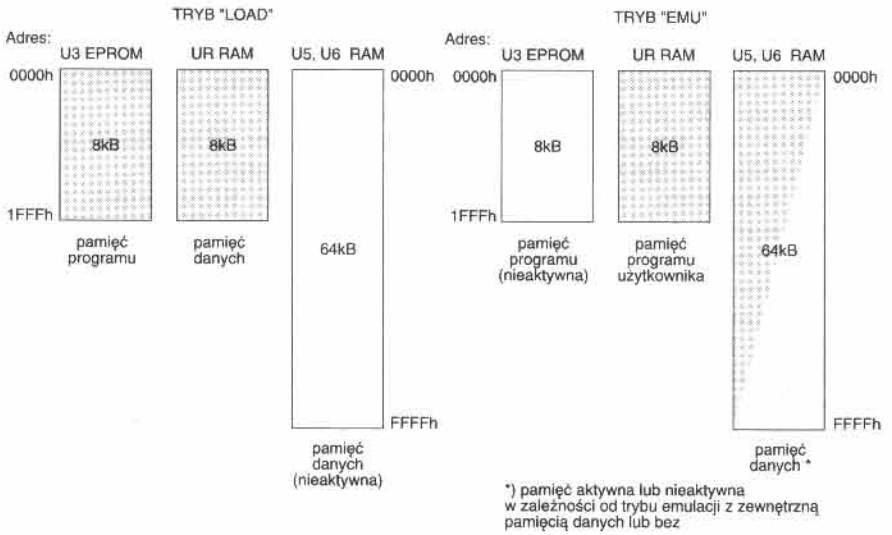
Jak widać z rysunku, pamięć EPROM z kodem programu nadzorującego, który jest wykonywany po włączeniu emulatora (tryb „LOAD"), zostaje „zastąpiona" przez układ U4 w trybie „EMU", który staje się pamięcią programu w przestrzeni adresowej procesora U1.

Po zapisaniu kodu programu użytkownika w pamięci U4 ustalana jest konfiguracja pracy w trybie emulacji. Ustalają ją stany logiczne na końcówkach portu P4. I tak:

- dla wyprowadzenia P4.0 stan zapisany później w U12 (końcówka 14) przełączy końcówki /RD, /WR procesora (piny 42,43) na wtyk emulacyjny (logiczne „1"), bądź na wejścia pamięci U5,U6 (logiczne „0"),
- podobnie stan P4.1 podłącza końcówki 52,53: XTAL1, XTAL2 U1 do kwarcu X1 (logiczne „0"), lub do wtyku emulacyjnego W1 (logiczne „1"), co umożliwi pracę z zewnętrznym rezonatorem kwarcowym;
- i wreszcie stan P4.2 przełącza końcówki 36,37: TXD,RXD U1 pomiędzy wtykiem W1 (logiczne „1") a układem U9 (logiczne „0").

Wysoki stan logiczny na wejściu 8 U10 informuje GAL,a że układ jest w trybie LOAD. Stan ten jest wymuszony przez układ U1 po jego wcześniejszym „resecie".

W trybie LOAD możliwe jest przetestowanie układów pamięci U5,U6 pomimo że są one „nieaktywne" jak wynika z **rys. 3**. Aby to zrealizować procesor wystawia stan niski na końcówce P4.5 (pin 21), który to pojawiając się na wejściu TEST układu U10 powoduje zablokowanie układu U4 (sygnał ROMS=0), a odblokowuje pamięci U5 i U6 (sygnały RAM1=0,RAM2=0). Umożliwia to odczyt i zapis tych układów przez procesor za pośrednictwem sygnałów /WR i /RD, które są dołączone jak wcześniej wspomniano do pinów 42 i 43 procesora U1. Końcówki P4.4 i P4.6 sterują diodami LED tak, że w trybie LOAD



Rys. 3. Mapy pamięci emulatora w dwóch trybach pracy.

Tab.1.

LD_EM	SRAM	TEST	EPRS	ROMS	ROMW	ROMR	RAM1	RAM2
1	X	0	0	1	WR	RD	A15	/A15
1	X	1	0	0	WR	RD	1	1

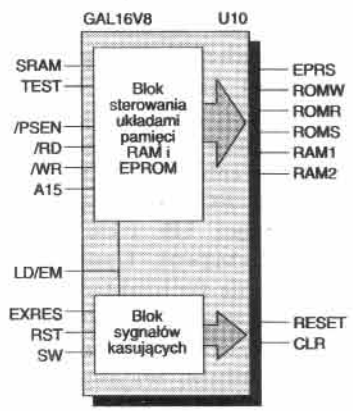
Tab.2.

LD_EM	SRAM	TEST	EPRS	ROMS	ROMW	ROMR	RAM1	RAM2
0	0	X	1	0	1	PSEN	1	1
0	1	X	1	0	1	PSEN	A15	/A15

pali się dioda D2 (zielona), a w trybie EMU pali się tylko dioda D3 (czerwona). Na wyjściu P4.3 U1 (pin 23) w trybie „LOAD" panuje stan niski.

Po załadowaniu programu użytkownika i ustaleniu konfiguracji pracy w trybie emulacji , procesor przechodzi do trybu EMU wymuszając stan wysoki na końcówce portu P4.3 (pin 23,U1). Narastające zbocze tego sygnału powoduje wyzwolenie uniwibratora U11A. W konsekwencji na wyjściu Q (końc.13, U11A) pojawia się dodatni impuls, którego narastające zbocze zapisuje konfigurację (stany pinów U1: P4.0, P4.1, P4.2) w układzie U12, zaś opadające zbocze wyzwala drugi uniwibrator U11B. Dodatni impuls na wyjściu Q tego drugiego (końc.5, U11B) poprzez układ U10 generuje sygnał zerujący procesor U1 (końc.12, U10).

W układzie U12, w momencie przechodzenia procesora w tryb emulacji, zapisane zostaje także logiczne zero z portu P4.3 U1, co poprzez końcówkę 3 układu U12 (zanegowany stan z P4.3), informując układ GAL U10 o przejściu w tryb EMU. Układ U10 na tej podstawie blokuje EPROM U3 (EPRS=1), a pamięć U4 ustawia



Rys. 4. Budowa układu GAL16V8 stosowanego w emulatorze.

jako pamięć programu procesora U1 (patrz rys.3). Ustawienie to polega na podaniu sygnału odczytu procesora z zewnętrznej pamięci programu /PSEN na wejście / OE pamięci U4, z jednoczesnym zablokowaniem jej zapisu (/ WE=1). Tak więc pamięć danych jak do tej pory, z zapisanym w niej programem użytkownika, staje się pamięcią programu. Na wejściu wyboru /CS1 U4 ustala się stan niski, przez co procesor, przed chwilą wyzerowany, zaczyna wykonywać zawarty w niej program. Emulator jest w trybie EMU. W zależności od ustalonej konfiguracji emulatora układ U10 steruje wejściami wyboru (/CE), zapisu i odczytu (/WR, /RD) pamięci U5 i U6, tak że pokrywa ona cały obszar adresowy procesora, 64kB. W trybie emulacji rolę portu P0 mikroprocesora pełni port P5 U1, a portu P2 - port P6 układu U1. Porty P1 i P3 są połączone bezpośrednio z wtykiem emulacyjnym W1.

Przełącznik S1 umożliwia zasilanie płytki emulowanego układu, należy jednak zwracać uwagę aby nie przekroczyć maksymalnego prądu pobieranego przez ten układ (ok. 300 mA).

W trybie EMU możliwe są dwa sposoby zerowania procesora. Pierwszy sposób poprzez przycisk znajdujący się na płytce układu użytkownika, powoduje „gorący restart" emulatora, tzn. emulator po tej operacji dalej znajduje się w trybie emulacji i rozpoczyna wykonywanie programu użytkownika od początku (z układu pamięci U4). Natomiast naciśnięcie przycisku K1 (RESET) powoduje „zimny" restart i po nim emulator przechodzi do trybu LOAD. Jeżeli emulacja odbywała się w trybie z zewnętrzną pamięcią danych, to zawartość pamięci U5,U6 nie ulega zmianie. Pozwala to na późniejsze załadowanie jej zawartości do komputera PC celem ewentualnej analizy przez użytkownika.

Generowaniem sygnału RESET procesora zajmuje się także układ GAL U10 na podstawie stanów logicznych na jego wejściach RST,SW, EXRES i LD/EM.

Układ GAL16V8 (U10)

Jak widać z wcześniejszego opisu, układ ten pełni po procesorze U1 kluczową rolę w pracy emulatora. Dla ułatwienia analizy na **rys. 4** przedstawiony jest jego schemat blokowy.

Pierwszy blok odpowiada za sterowanie blokami pamięci SRAM oraz pamięci EPROM układu emulatora, zgodnie z wcześniejszym opisem, drugi blok generuje sygnał zerowania procesora RESET oraz sygnał kasowania CLR układu U12.

Stan logiczny na wejściu LD_EM odpowiada za tryb pracy emulatora LOAD/EMU. Stan na wejściu SRAM informuje układ U10 o trybie emulacji: z zewn. pamięcią RAM lub bez niej. Sygnały /PSEN, /RD, /WR, A15 są potrzebne do sterowania blokami pamięci. Wejście EXRES jest połączone z pinem 9 wtyku emulacyjnego W1, a więc odpowiedzialnego za zewnętrzny sygnał „gorącego zerowania" procesora.

Wejście SW jest wejściem sygnału „zimnego" restaru emulatora, a RST to wejście sygnału kasującego U1 z uniwibratora U11B. Znaczenie sygnałów wyjściowych układu U10 jest następujące:

EPRS - (EPROM Select) sygnał wyboru pamięci U3,
ROMW- (User ROM Write) sygnał zapisu do pamięci programu użytkownika U4,
ROMR - (User ROM Read) sygnał odczytu z pamięci U4,
ROMS - (User ROM Select) sygnał wyboru układu U4,
RAM1 - sygnał wyboru układu U5 (adres 0000h-7FFFh),
RAM2 - sygnał wyboru układu U6 (adres 8000h-FFFFh),
RESET - sygnał zerowania procesora U1,
CLR - sygnał zerowania układu U1.

Jeżeli ktoś nie rozumie idei stosowania w układzie struktur programowalnych typu GAL, na-

leży wyjaśnić, że zastosowany tu GAL16V8 (U10) zaprogramowano jako układ kombinacyjny, tak aby spełniał wszystkie funkcje wymienione wcześniej. Sposób jego działania najlepiej przeanalizować przy pomocy tabeli prawdy **(tab.1** i **tab.2)**.

Trzy pierwsze kolumny to sygnały wejściowe: LD_EM, SRAM, i TEST. Pozostałe to wyjściowe opisane wcześniej. Przypatrzmy się bliżej tabeli. Pierwszy wiersz określa tryb testowania pamięci RAM U5 i U6. Drugi wiersz zaś tryb ładowania i weryfikacji programu użytkownika do pamięci U4. Znak „X" oznacza dowolny poziom logiczny na wejściu SRAM. I tak np. wartość „WR" w kolumnie ROMW oznacza że na wyjściu ROMW układu U10 w tym przypadku pojawia się sygnał WR procesora - zapisu do pamięci, innymi słowy GAL „przenosi" sygnał WR na wyjście ROMW. Dotyczy to także pozostałych wartości.

Pierwszy wiersz określa przypadek pracy bez zewnętrznej pamięci danych. Jak widać sygnały RAM1,RAM2 blokują dostęp do bloku pamięci U5 i U6, sygnał

```
/* Dekoder na GAL16V8 dla emulatora sprzętowego procesora 8751 */
#define DESIGNER Slawomir Surowinski
#define PARTNUM GAL16V8
#define COMMENTS Dekoder dla emulatora sprzetowego uP 8751 ver 2.0

/* deklaracje sygnalow wejsciowych i|wyjsciowych */

emu_dek (in  PSEN, RD, WR, A15, TEST, SRAM, LD_EM, EXRES, RST, SW :
              out  EPRS, RAM1, RAM2, ROMS, ROMR, ROMW, CLR, RESET )

/* zadeklarowanie wyjsc ukladu G16V8 jako prostych */
{
          EPRS.oe   =|1;        RAM1.oe   =|1;
          RAM2.oe   =|1;        ROMS.oe   =|1;
          ROMR.oe   =|1;        ROMW.oe   =|1;
          RESET.oe  =|1;        CLR.oe        =|1;

/* równania opisujace blok sygnalow kasujacych */

          CLR =|SW;                                     /* sygnał zerowania U12 */
          RESET =|!!((LD_EM ||!EXRES) &|SW &|!RST);     /* sygnał zerowania U1   */

/* tabele opisujace blok sterowania ukladow pamieci RAM i EPROM */

      table ||LD_EM, SRAM, TEST -> EPRS, ROMS, ROMW,  ROMR,  RAM1,  RAM2  ]||
/****************** tryb pracy EMU ******************/
/* no SRAM */   (|0,    0,    ?|=> 1,       0,        1,        PSEN,     1,      1||||);
/* ze SRAM */   (|0,    1,    ?|=> 1,       0,        1,        PSEN,     A15,    !A15 );
/****************** tryb pracy LOAD ******************/
/* test SRAM */ (|1,    ?,    0 -> 0,       1,        WR,       RD,       A15,    !A15 );
/* normal    */   (|1,    ?,    1|-> 0,     0,        WR,       RD,       1,      1||||);
}

/* |deklaracja zastosowanego układu scalonego, w|tym przypadku G16V8 */

   putpart (|"gl6v8","emu_dek",
         EXRES, PSEN,  RD,   WR,   SW,   TEST,   RST,    LD_EM, A15,   GND,
         SRAM,  RESET, CLR,  RAM1, RAM2, EPRS, ROMS, ROMR, ROMW, VCC );      ||
/*-------------------------------------------------------------------------*/
```

List.1. Program opisujący konfigurację układu GAL16V8 (przygotowany dla kompilatora Tango PLD).

ROMR „przekazuje" /PSEN na wejście odczytu układu U4, a EPROM U3 jest zablokowany (EPRS=1). Poziom logiczny na wejściu TEST jest w tym przypadku nieistotny.

Sygnały zerowania procesora RESET i CLR można zapisać równaniami przedstawionymi na list.1 (opis dla programu Tango PLD). *Dokończenie artykułu w EP1/96.*

Sławomir Surowiński