Basic Stamp "Elektroniczny Znaczek", część 1

Opis sprzętu i języka

Mikrokomputerki rodziny Basic Stamp cieszą się dużym zainteresowaniem wśród naszych Czytelników. Trudno się temu dziwić, gdyż możliwości tych niewielkich mikrokomputerów, łatwość ich stosowania i programowania, pozwalają na tworzenie dość zaawansowanych sterowników przez projektantów nie posiadających zbyt wielkiego doświadczenia. Od chwili pojawienia się ich w ofercie handlowej AVT, otrzymaliśmy bardzo wiele listów i telefonów z pytaniami, na które postaramy się odpowiedzieć w serii artykułów. Rozpoczynamy od przybliżenia konstrukcji prostego mikrokomputerka Basic Stamp I.

Firma Parallax Inc. z Kalifornii oferuje bardzo ciekawy układ mikrokomputera, który nazwała po prostu „znaczkiem". Płytka tego urządzenia jest tak mała, że może być porównana wielkością ze zwykłym znaczkiem pocztowym. Jednak niewielkie rozmiary komputerka nie przesądzają o jego skromnych możliwościach. Zanim zajmiemy się przykładowymi zastosowaniami, przedstawimy jego skrótowy opis.

Basic Stamp z zewnątrz

Na płytce drukowanej o wymiarach 35.5x10mm umieszczono mikroprocesor PIC16C56 zamknięty w obudowie SMD i pamięć EEPROM typu 93LC56, która pełni rolę pamięci programu. Dodatkowo do mikroprocesora dołączono prosty układ zerowania, port I/O dla użytkownika oraz rezonator kwarcowy będący hybrydą kwarcu i dwóch pojemności odsprzęgających.

Na **rys. 1** pokazano schemat elektryczny całego komputerka.

Na dłuższym boku płytki (**fot. 1**) umieszczono złącze, za pomocą którego użytkownik może komunikować się ze Stampem. Na złączu dostępne są następujące sygnały:

PWR - napięcie zasilające niestabilizowane. Zalecana wartość napięcia wynosi 6..15V, przy czym dopuszczalne są wartości większe, jednak nie przekraczające 35V;

GND - masa;

PCO - wyjście do IBM PC, jest ono podłączane do linii BUSY (nóżka 11) portu równoległego Centronics;

PCI - wejście danych z IBM PC, jest ono podłączane do linii D0 (nóżka 2) portu równoległego Centronics, razem z PCO tworzą interfejs komunikacyjny z komputerem nadrzędnym, z którego można programować Stampa;

+5V - wejście/wyjście napięcia stabilizowanego, jest to wyjście układu lokalnego stabilizatora 5V, który znajduje się na płytce Stampa. Jeśli Stamp jest zasilany z napięcia niestabilizowanego przez nóżkę PWR, wtedy jest to wyjście napięcia stabilizowanego o stosunkowo niewielkiej wydajności. Kiedy PWR jest wolne, nóżka +5V staje się wejściem dla zewnętrznego napięcia stabilizowanego z zakresu 4.5V ~ 5.5V;

RES - wejście/wyjście sygnału zerowania procesora;

P0..P7 - piny we/wy, każdy z nich może przyjąć prąd o wartości ok. 25mA, może także stanowić źródło prądu o wydajności do 20mA. Ograniczeniem ze względów na skończoną moc strat jest sumaryczny prąd dla wszystkich pinów, który nie może przekroczyć 50mA dla prądu wpływającego i 40 mA dla prądu wypływającego.

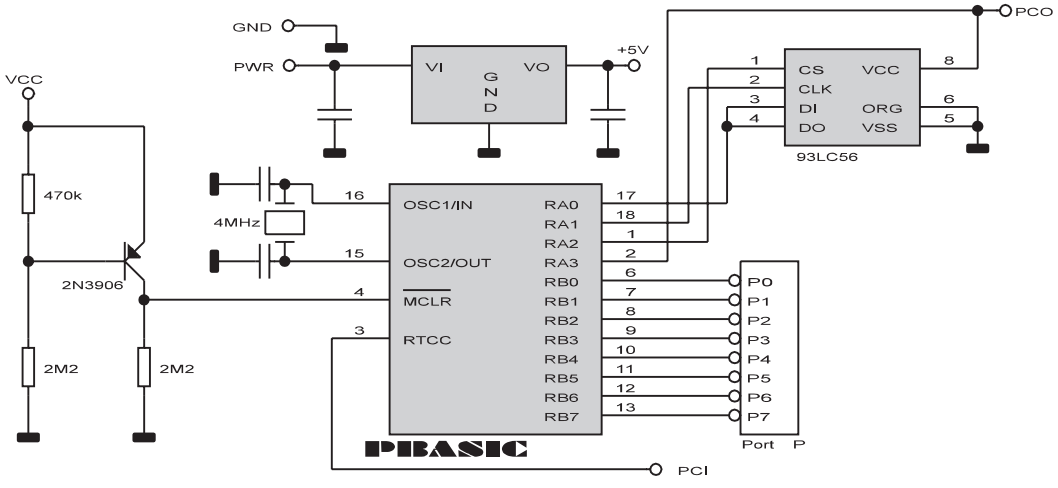
Zasoby programowe

Do pamięci programu mikroprocesora Basic Stamp wpisano program zawierający procedury komunikacji z komputerem nadrzędnym oraz interpreter prostego języka PBasic.

Basic Stamp dla celów programu w PBasicu rezerwuje 16 bajtów pamięci danych RAM. Dwa pierwsze bajty są przeznaczone do obsługi pinów P0..P7, z czego pierwszy o nazwie Pins odpowiada stanom na poszczególnych nóżkach portu we/wy, zaś drugi o nazwie Dirs decyduje o kierunku przesyłania informacji dla poszczególnych nóżek portu. Owe dwa bajty tworzą słowo o nazwie Port. Pozostałe 14 bajtów zostało zgrupowanych w siedem dwubajtowych słów, numerowanych od W0 do W6. Słowo W0 oraz słowo Port mogą być adresowane bitowo, czyli mogą być traktowane jako indywidualne bity, np. w celu przechowywania flag pewnych uruchamianych procesów oraz zmiany stanu osobnych linii portu P. Te 32 bity mają swoje nazwy symboliczne w interpreterze PBasica jako Pin0..Pin7, Dir0..Dir7, Bit0..Bit15.

Ponadto słowa W0..W6 zostały podzielone na dwa bajty oznaczone B0..B13. Podział pamięci został zbiorczo przedstawiony w **tab. 1**.

Słowo Port składa się z dwóch bajtów, Pins i Dirs. Bajt Pins oraz odpowiadające mu bity Pin0..Pin7 to linie portu P. Odczyt tych zmiennych jest równoważny z od-



Rys. 1.

Tabela 1.

Słowo	Bajt	Nazwy bitów	Uwagi
Port	Pins	Pin0..Pin7	Piny I/O; możliwość osobnego adresowania
	Dirs	Dir0..Dir7	Piny sterowania kierunkiem transmisji w porcie P; możliwość osobnego adresowania
W0	B0	Bit0..Bit7	możliwość adresowania bitowego
	B1	Bit8..Bit15	możliwość adresowania bitowego
W1	B2		
	B3		
W2	B4		
	B5		
W3	B6		
	B7		
W4	B8		
	B9		
W5	B10		
B11			
W6	B12		Używane przez instrukcję GOSUB
	B13		Używane przez instrukcję GOSUB

czytem stanu wejść portu P. Zapis do zmiennej Port zawartej w pamięci RAM użytkownika oznacza jednoczesne przepisanie zawartości tej zmiennej na fizyczne linie portu P.

Bajt Dirs oraz odpowiadające mu bity Dir0..Dir7 decydują o dozwolonym kierunku transmisji informacji. Zapis zera na określoną pozycję bajtu Dirs oznacza ustawienie odpowiadającej mu pozycji bajtu Port jako wejścia, zaś zapis jedynki oznacza zdefiniowanie odpowiadającej mu pozycji jako wyjścia.

Podkreślone słowa wymagają dodatkowego wyjaśnienia. Jak niektórym czytelnikom wiadomo, w mikroprocesorach PIC definicja portu pokrywa się w swej istocie z tym, co jest dostępne w tym interpreterze PBasica. Na podstawie analizy schematu, bajt Port odpowiada portowi PB i komórce pamięci o adresie 6h, zaś bajt Dirs jest odpowiednikiem rejestru sterującego TRISB, który jest dostępny poprzez rozkaz TRIS. Różnica pomiędzy reprezentacją w mikroprocesorze a rozwiązaniem występującym w PBasicu Stampa polega na inwersji wartości pomiędzy rejestrem TRIS a bajtem Dirs. Zapis zera do bitu TRIS oznacza ustawienie odpowiadającej mu linii jako wyjścia, zaś zapis jedynki daje efekt w postaci ustawienia odpowiadającej mu linii jako wejścia, dokładnie odwrotnie niż ma to miejsce w PBasicu Stampa. Należy więc zapamiętać tę różnicę.

Zatem zapis w PBasicu „dirs=%01010101“ oznacza, że bity 7, 5, 3 i 1 portu P to wejścia, zaś pozostałe to wyjścia.

Podstawowa symbolika PBasica

Znaki tego języka to wszystkie znaki 7-bitowego alfabetu ASCII. Wielkość liter nie odgrywa tutaj roli, czyli wielkie i małe litery są traktowane jednakowo. Wyjątek stanowią definicje stałych znakowych (np. “Ala ma koTa“ różni się od “Ala ma kota“).

Linia programowa jest podstawową interpretowaną jednostką programową. Linia programowa może zawierać jedną lub więcej instrukcji języka. W danej linii programowej poszczególne instrukcje są oddzielone od siebie znakiem dwukropka (:). Na przykład poniższe dwie wersje programu są sobie równoważne:

- wersja z jedną instrukcją w jednej linii:

```
dirs=255
for b2=0 to 100
pins=b2
next
```

- wersja z jedną linią:

```
dirs=255 : for b2=0 to 100 : pins=b2 : next
```

Komentarze zaczynają się po znaku apostrofu (') i trwają do znaku końca danej linii programu. Stosowanie komentarzy jest dobrym zwyczajem każdego szanującego się programisty. Innym oznaczeniem komentarza jest dyrektywa REM znana z innych wersji PBasica, tutaj też dostępna. Użycie REM wyklucza całą linię z procesu interpretacji programu.

Wartości stałe są deklarowane na cztery sposoby: wartości dziesiętne, szesnastkowe, binarne oraz znaki ASCII. Wartości dziesiętne są zapisane wprost, wartości szesnastkowe są poprzedzone znakiem dolara ($), wartości binarne są poprzedzone znakiem procentu (%), a znaki ASCII oraz ciągi znaków ASCII są objęte znakami cudzysłowu („). Na przykład:

```
200          - wartość dziesiętna
$2f          - wartość szesnastkowa
%11100111    - wartość binarna
“A“          - znak ASCII
“Abecadlo“   - ciąg znaków ASCII
```

W praktyce, ze względu na nasze przyzwyczajenia, przede wszystkim używamy wartości zapisanych dziesiętnie. Jednak w niektórych sytuacjach użycie liczb zapisanych szesnastkowo czy binarnie daje bardziej przejrzysty zapis, np. opisując stany pojedynczych linii portu P.

Etykiety adresowe

Etykiety adresowe w tym języku symbolicznie odpowiadają adresom, do których program będzie się odwoływał. Trzeba wiedzieć, że ta wersja PBasica nie dopuszcza numerowania poszczególnych linii. Etykieta adresowa jest ciągiem liter, cyfr i znaku podkreślenia (_). Pierwszym znakiem tej etykiety nie może być cyfra. Etykieta definiująca adres jest zakończona znakiem dwukropka (:), zaś odwołanie do niej w programie znaku dwukropka nie wymaga. Oczywiście etykieta nie może być równoznaczna jakiemukolwiek słowu kluczowemu języka (np. serin, toggle, goto itd.). Poniższy prosty program ilustruje sposób użycia etykiety:

```
petla:
toggle 0   'inwersja wartości pinu 0
for b0=1 to 10
toggle 1   ' dziesięciokrotna
           ' zmiana wartości na pinie 1
next
goto petla        ' powtórz ten proces
```

Zmienne i stałe użytkownika

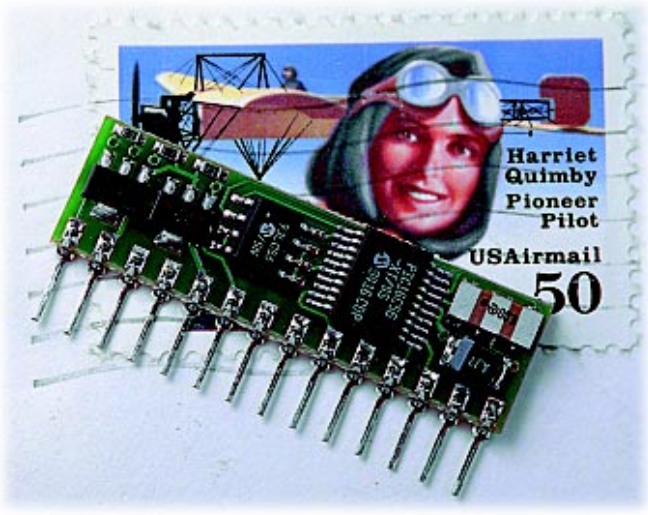
Ponieważ cała pamięć użytkownika została w tym języku symboliczne predefiniowana, trudno byłoby mówić o zmiennej w takim rozumieniu, jakie spotyka się w innych językach. Zmienne tutaj zostały sprowadzone do czynności wskazania konkretnego adresu symbolicznego z pamięci użytkownika (patrz tab. 1). W celu odróżnienia od etykiety adresowej definicja zmiennej czy stałej polega na poprzedzeniu słowa definiującego dyrektywą SYMBOL. Stałą definiujemy jako nazwę połączoną znakiem równości z liczbą zapisaną w jednym z czterech dopuszczalnych formatów zapisu liczby. Zmienną definiujemy jako nazwę połączoną znakiem równości z jednym z predefiniowanych symboli wymienionych w tab. 1. Oto przykład:

```
symbol poczatek=1 ' definicja stałej
                  ' początek
symbol koniec=100 ' definicja stałej
                  ' koniec
symbol licznik=b3 'definicja zmiennej
       ' licznik jako bajtu B3
petla:
for licznik=poczatek to koniec
toggle 1     ' stukrotna zmiana stanu
             ' linii Pin1
next
```

Operatory matematyczne

Są to symbole reprezentujące wykonanie określonej operacji w wyrażeniach. PBasic Stampa dopuszcza następujące operatory:

```
+   dodawanie
-   odejmowanie
*   mnożenie - wynikiem jest młodsze słowo iloczynu
**  mnożenie - wynikiem jest starsze słowo iloczynu
/   dzielenie - wynikiem jest iloraz dzielenia
//  dzielenie - wynikiem jest reszta dzielenia
min wartość nie większa
max wartość nie mniejsza
&   logiczne AND
```



Fot. 1.

|	logiczne	OR
^	logiczne	XOR
&/	logiczne	NAND
|/	logiczne	NOR
^/	logiczne	XNOR

Instrukcje języka

Instrukcje języka PBasic można zebrać w kilka funkcjonalnych grup. Oto one:

Skoki
IF...THEN
BRANCH
GOTO
GOSUB
RETURN
Pętla
FOR...NEXT
Instrukcje numeryczne
LET
LOOKUP
LOOKDOWN
RANDOM
Instrukcje we/wy cyfrowe
OUTPUT
LOW
HIGH
TOGGLE
PULSOUT
INPUT
PULSIN
REVERSE
BUTTON
Transmisji szeregowej
SERIN
SEROUT
Instrukcje we/wy analogowe
PWM
POT
Dźwięku
SOUND
Dostępu do pamięci EEPROM
EEPROM
READ
WRITE
Czasu
PAUSE
Poboru mocy
NAP
SLEEP
END
Uruchamiania programu
DEBUG

Instrukcje skoków obejmują pięć różnych skoków. Skokiem najprostszym jest oczywiście skok bezwarunkowy GOTO. GOSUB jest odwołaniem do podprogramu, zaś RETURN oznacza powrót z niego. PBasic ma dwa skoki warunkowe: znany IF...THEN oraz BRANCH, który jest odpowiednikiem w innych wersjach Basica instrukcji ON...GOTO.

Instrukcja pętli FOR...NEXT zapewnia automatyczne powtarzanie pewnych sekwencji czynności zadaną liczbę razy.

Instrukcje numeryczne dotyczą obsługi prostych tablic danych (LOOKUP i LOOKDOWN), generatora liczb losowych (RANDOM). Do nich zaliczono instrukcję przypisania LET, która w zasadzie została umieszczona dla porządku, bowiem nie ma ona praktycznego znaczenia i może być pominięta.

Instrukcje we/wy zostały podzielone na cyfrowe i analogowe. Podział ten jest sztuczny, bowiem mikroprocesor PIC16C56 nie posiada wejść analogowych. Uczyniono tak dlatego, żeby uświadomić użytkownikowi, że wykorzystując właściwości progowe wejść cyfrowych można śledzić wybrane procesy dziejące się w świecie analogowym.

Cyfrowe instrukcje we/wy dotyczą pojedynczych pinów portu P. Można więc dany pin ustawić (HIGH), wyzerować (LOW), ustawić je jako wejście (INPUT) bądź wyjście (OUTPT), zmienić kierunek przesyłu informacji na przeciwny (REVERSE), zmienić jego stan na przeciwny (TOGGLE), wygenerować krótki impuls o zadanej długości (PULSOUT), zmierzyć długość występującego impulsu, testować stan przycisku doń podłączonego (BUTTON).

Analogowe instrukcje we/wy pozwalają wytworzyć falę prostokątną o zadanym wypełnieniu i liczbie okresów (PWM) oraz odczytać proporcje podziału napięcia na dzielniku rezystancyjnym (POT).

Instrukcje komunikacji szeregowej zapewniają transmisję szeregową w obu kierunkach z jedną z czterech prędkości i tylko w jednym formacie.

Instrukcja SOUND zapewnia wytworzenie fali prostokątnej o zadanym czasie trwania. Sygnał ten może z powodzeniem zasilać brzęczyk lub mały głośniczek.

Instrukcje dostępu do pamięci EEPROM umożliwiają potraktowanie pamięci programu jako przedłużenia pamięci danych. Oczywiście dostęp do tak przechowywanych danych jest wolniejszy niż do pamięci RAM procesora, ale za to pamięć ta jest wielokrotnie bardziej pojemna.

Instrukcja PAUSE zatrzymuje działanie programu na zadany okres czasu.

Instrukcje sterowania poborem mocy wykorzystują właściwości samego mikroprocesora PIC16C56. Można zatem wprowadzić Stampa w stan chwilowego uśpienia albo w stan głębokiego uśpienia aż do zewnętrznego zerowania.

Instrukcja DEBUG przesyła stan wybranych zmiennych do nadrzędnego komputera PC.

Mirosław Lach, AVT

Czytelnicy posiadający dostęp do Internetu mogą skorzystać z serwisu WWW firmy Parallax, który znajduje się pod adresem: http://www.parallaxinc.com.