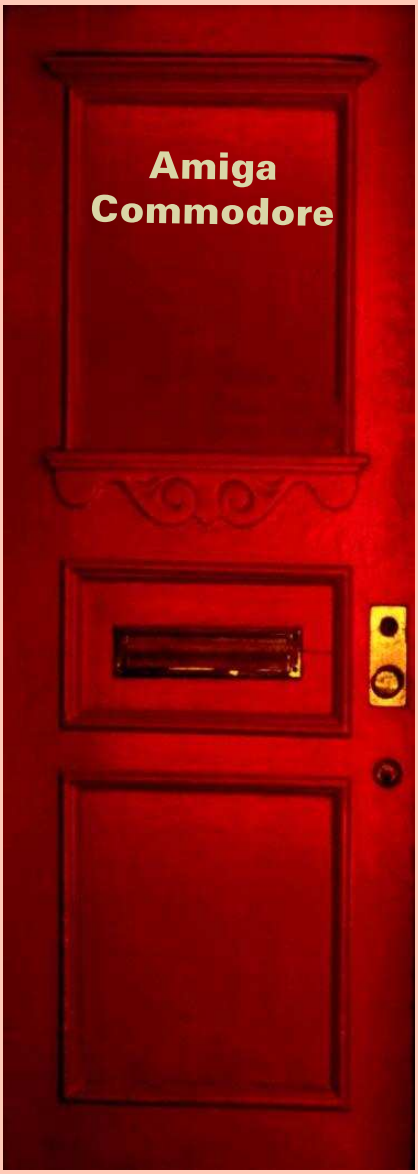
Pierwsze kroki

CZĘŚĆ 1

Witam w nowym cyklu poświęconym urządzeniom mikroprocesorowym, w szczególności urządzeniom współpracującym z C-64 i Amigą. Zapraszam wszystkich użytkowników tychże "maszyn". A może inni zainteresują się komputerem i zachęcą do jego kupna. Używany C-64 można obecnie nabyć za około 130 zł, napęd dyskowy za 150 zł. Najmniejsze Amigi sprzedawane są za około 350 zł. Jak widać, nie jest to wielki wydatek, a możliwości ogromne.



Do czego można wykorzystać "moc" naszych maszyn? Do wielu celów, np. sterowanie różnymi urządzeniami, takimi jak: oświetlenie choinkowe, węże świetlne, wyświetlanie czasu zsynchronizowane atomowym wzorcem czasu. Muzycy też znajdą coś dla siebie: możliwość generowania efektów takich jak: echo, pogłos. Inne możliwości to sterowanie urządzeniem pomiarowym, zbieraniem wyników i wyświetlanie ich na ekranie, bądź wydrukowanie na drukarce. Fani video mogą dodać do swoich nagrań efektowną czołówkę. Możliwości w zasadzie są ograniczone jedynie fantazją i wiedzą użytkownika. Tego pierwszego Wam nie dam, natomiast wiedzę mogę przekazać. No to zaczynamy.

Komputer jest urządzeniem zawierającym w swym wnętrzu (i tu następuje długa lista). A właściwie co nas interesuje, co jest w środku? A niech będzie nawet krasnoludek (jeśli one jeszcze żyją?). Mówmy o tym co można zobaczyć, dotknąć, przyłączyć i czasem zepsuć. Najważniejsze są dwie rzeczy: komputer z klawiaturą i monitor. No tak, ale nasze dzieło (tekst, program) trzeba gdzieś przechować. Idea papieru i ołówka odpada, gdy napiszemy program liczący ponad 100 linii, pozostaje pamięć masowa. Bądź to w postaci magnetofonu (C-64), bądź "miękiego" napędu dyskowego (C-64 i Amiga), czy też "twardego" dysku (w zasadzie tylko komputery 16 i więcej bitowe np. Amiga). Do czego służy klawiatura? Głównie do wprowadzania tekstów, programów, wydawania komend no i grania! Tym ostatnim nie będziemy się zajmować. A do czego służy monitor (czasem zwykły telewizor)? Na nim otrzymamy wyniki naszych prac, komunikaty o błędach, ładne obrazki.

Jak wspomniałem, opiszę urządzenia do C-64 i Amigi. Przykładowe listingii będą pisane w Basicu, nie dlatego, że innego języka nie znam, ale jest to jedyny, który można bez problemu przenosić pomiędzy różnymi komputerami. Zmusza mnie to więc do rezygnacji ze specjalnych rozkazów np. Amosa, czy "sztuczek" na C-64. Bardzej skomplikowane programy można będzie otrzymać na dyskietkach, często z kodami źródłowymi.

Czym właściwie jest Basic? Jest to język programowania dla (bez obrazy) początkujących. A czym są języki programowania? Jest to "pomost" pomiędzy maszyną a człowiekiem. Dzięki niemu można w łatwy sposób przekazać algorytm działania (program) do komputera. Program "tłumaczący" nasz tekst na język wewnętrzny procesora nazywamy translatorem. Translacja może przebiegać w dwojaki sposób. Jeśli kod źródłowy (nasz program) zostanie w całości przetłumaczony na język wewnętrzny, a potem wykonany - mamy do czynienia z kompilacją. Program jest uruchamiany dopiero po bezbłędnej kompilacji. Jeśli translacja wykonywana jest linia po linii, jednocześnie z wykonywaniem programu źródłowego - mamy do czynienia z interpretacją. Basic jest w zasadzie interpreterem, chodź i dla niego powstały kompilatory. Istnieją wersje Basic, które przed uruchomieniem są kompilowane (np. Amos).

Przejdźmy do rzeczy. Na początek, dla przypomnienia, parę rozkazów. Najczęściej używanym rozkazem jest instrukcja wyprowadzania informacji na monitor: PRINT. Można nią wydrukować tekst, wynik obliczeń. Spróbujmy! Włączamy komputer. C-64 zgłosi się nam interpreterem Basica, na Amidze natomiast musimy go uruchomić (np. GFA BASIC, Amos itp.) "klikając" na ikonie lub botując z dyskietki. Gdy komputer będzie gotowy do pracy napiszmy:

```
PRINT  2+3
```

i naciśnijmy Return (największy klawisz z prawej strony klawiatury) w przypadku C-64, na Amidze uruchamiamy kompilację (F1 dla Amosa). Na monitorze otrzymamy:

```
5
```

Można próbować innych działań np. mnożenia, dzielenia, potęgowania, pierwiastkowania, sinusów, logarytmów. Drugim potrzebnym nam rozkazem będzie POKE. Służy on do wpisania bajtu pod określony adres w pamięci. Spróbujmy na C-64 wpisać:

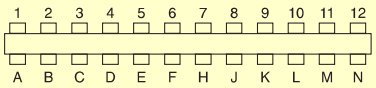
```
POKE  1024,65
```

potwierdzając klawiszem Return. W lewym górnym rogu pojawi się litera:

```
A
```

Użytkownicy starszych modeli C-64, tzw. "mydelniczek", prawdopodobnie nic nie zobaczą, jest to spowodowane błędem w systemie operacyjnym. Nie kasując zawartości ekranu wprowadźmy

Port C-64



widok od tyłu komputera

Pin	Sygnał	Uwagi
1	GND	
2	+5V	max 100mA
3	RESET\	
4	CNT1	
5	SP1	
6	CNT2	
7	SP2	
8	PC2\	
9	SER.ATN.IN	
10	9VAC	max 100mA
11	9VAC	max 100mA
12	GND	
A	GND	
B	FLAG2\	
C	PB0	
D	PB1	
E	PB2	
F	PB3	
H	PB4	
J	PB5	
K	PB6	
L	PB7	
M	PA2	
N	GND	

Port Amigi

Pin	Sygnał	Uwagi
1	STROBE\	
2	D0	
3	D1	
4	D2	
5	D3	
6	D4	
7	D5	
8	D6	
9	D7	
10	ACh\	
11	BUSY	
12	POUT	
13	SEL	
14	+5V	max 10mA
15	nie podłączone	
16	RESET\	
17-25	GND	

```
POKE    55296,7
```

i naturalnie Return. Litera zmieniła kolor na żółty. Zmieniając siódemkę na liczby z zakresu 0-15 uzyskamy różne kolory. Amigowcy wpiszą i uruchomią coś takiego:

```
POKE    $BFE001,2
```

I co? Dioda POWER przygasła. Natomiast

```
POKE    $BFE001,0
```

rozświetli ją pełną mocą. Tu zauważymy pierwszą wadę Basica C-64 (a może zaletę Basica Amigi): argumenty muszą być wprowadzane w systemie dziesiętnym (w Amidze dziesiętnym, szestnastkowym, binarnym). Znak $ przed liczbą wskazuje, iż jest to liczba zapisana w systemie szesnastkowym.

No, a jak pisze się program? Spróbujmy napisać na C-64:

```
10 PRINT "CZESC"
```

i naciśnijmy RETURN. Na ekranie nic się nie zmieniło (przesunął się kursor). Wpiszmy:

```
RUN
```

i naciśnijmy Return. Na ekranie zostanie wyświetlony napis CZESC. Jak łatwo zauważyć kod programu w Basicu rozpoczynamy numerem linii. Program jest wykonywany w kolejności od linii o najmniejszym numerze do największego. Dopiszmy:

```
5 PRINT     "SIEEEMA     " [Return]
```

(Umówmy się, że to co w nawiasie kwadratowym ma zostać naciśnięte, a nie napisane).

Następnie:

```
RUN     [Return]
```

w wyniku czego komputer powita nas słynnym okrzykiem Owsiaka. Ciekawe no nie? Wpiszmy:

```
LIST     [Return]
```

na ekranie zostanie wyświetlony kod naszego programu. Jak widać linie są uporządkowane. Nie ma więc konieczności wpisywania programu w kolejności, zostanie on uporządkowany samoczynnie. Jak jest możliwość wyprowadzania danych, to i musi istnieć wprowadzanie. Ci od C-64 napiszą:

```
NEW     [Return]
```

Dzięki temu kod naszego programu zostanie skasowany. Amigowcy muszą wykonać to w sposób zależny od wersji Basica (w Amosie należy nacisnąć równocześnie klawisz "Amiga" oraz "Q"). I małe wyjaśnienie, numery linii w Amosie są opcjonalne i nie brane pod uwagę podczas kompilacji. Wpiszmy program:

```
10 PRINT "JESTEM KOMPUTEREM
A TY JAK MASZ NA IMIE?"
[Return]
20  INPUT  T$  [Return]
30  PRINT  "MILO  SIE  Z TOBA
GAWEDZILO ";T$;" ALE MUSZE
KONCZYC.  CZESC!"  [Return]
```

Po jego uruchomieniu na ekranie zostanie wyświetlony tekst:

```
JESTEM KOMPUTEREM A TY JAK
MASZ NA IMIE?
?
```

Za znakiem zapytania pojawi się migający kursor. Wprowadzimy teraz imię np. WACEK i zatwierdzimy przez Return. Komputer odpowie nam na to tak:

```
MILO SIE Z TOBA GAWEDZILO
WACEK ALE MUSZE KONCZYC.CZESC!
```

To są najprostsze przykłady. Celem cyklu nie jest nauczenie Basica, dlatego zupełnie początkujących odsyłam do odpowiedniej literatury. My teraz przechodzimy do spraw interesujących elektronika.

Obydwa wspomniane komputery posiadają po dwa układy, nazywane CIA od Complex Interface Adapter. Układ CIA zawiera w swej strukturze dwa ośmiobitowe porty wejścia/wyjścia, timery, zegar czasu, rejestr przesuwający. Na razie najbardziej interesuje nas port równoległy (w C-64 USER Port).

Port jest dwukierunkowy, czyli może zarówno przesyłać dane z komputera do urządzenia współpracującego, jak i przyjmować dane od tego urządzenia.

Pracą portu sterujemy za pomocą dwóch rejestrów: danych i sterującego. Rejestr sterujący decyduje, które z ośmiu linii będą wejściowe, a które wyjściowe. Do rejestru danych wpisujemy liczbę jaka ma pojawić się na porcie (bitach ustawionych na wyjście). Możemy też odczytać bity ustawione jako wejście. Rejestry te można traktować jak komórki pamięci o określonych numerach (adresach). Adresy rejestrów portu podano poniżej:

Komputer	Rejestr danych	Rejestr sterujący
Amiga	$BFE101	$BFE301
C-64	56577	56579

W Amidze jest to gniazdo Cannon 25 styków. Wyjściami portu są piny (styki): 2 dla bitu 0, 3 dla bitu 1, 4 dla bitu 2, itd. do pinu 9. Razem osiem końcówek. Masa to wyprowadzenia od 17 do 25. W C-64 jest to złącze umiejscowione z lewej strony komputera. Interesują nas styki C, D, E i tak aż do L. Masa natomiast to 1, 12, A, N.

Sławomir Skrzyński

Przejdźmy do następnej części artykułu opisującej sposób przyłączenia do komputera różnych urządzeń. Na początek najprostszy przykład sterowania żarówką (**rysunek 1**). Ze względu na małą wydajność prądową portu zastosowano inwerter wchodzący w skład kostki 7406. Wyjście typu "otwarty kolektor" umożliwia sterowanie lampki o napięciu pracy do 30V. Wydajność prądowa układu wynosi 40mA. Jeżeli do zasilania odbiornika (lampki) wykorzystamy napięcie +5V, to można użyć układu 7404 (max 16mA) lub o większej wydajności prądowej 7437 (max 48mA). Warto wspomnieć, że pewne znaczenie ma tu seria układu scalonego (74LS, 74HCT). Różnią się one między sobą poborem mocy oraz maksymalną wydajnością prądową. Inwertery można zastąpić tranzystorami - **rysunek 2a**. Rezystor w obwodzie bazy nie jest konieczny, lecz gdy go nie będzie, odczyt portu da niewłaściwą wartość $00 i tym samym uniemożliwi stosowanie rozkazów OR, AND, BCHG, BCLR, BSET na porcie. Spowodowane jest to faktem, że napięcie baza-emiter tranzystora nie jest większe niż około 0.6V, czyli traktowane jest jako stan niski. Czasem mogą wystąpić problemy z przejściem tranzystora w stan zatkania. Należy wtedy zastosować układ z **rysunku 2b**. Porty są wykonane w technologi MOS, dzięki czemu napięcie linii portu w stanie niskim jest bliskie zeru i sytuacja taka nie powinna wystąpić. Gdy chcemy sterować większą mocą, należy użyć tranzystorów w układzie Darlingtona (**rysunek 2c**). Siedem takich tranzystorów zawierają układy ULN2001/2/3/4, osiem ULN2801/2/3/4/5. W skład układu wchodzą dodatkowo diody zabezpieczające. Do naszych zastosowań najkorzystniejsze są układy ULN2003 i ULN2803 przystosowane do układów CMOS i TTL. Maksymalne napięcie wyjściowe tych układów to 50V, prąd 500mA.

Lampkę można zastąpić diodami LED wg **rysunku 3**. Wartość rezystora ograniczającego należy dobrać, by nie przekroczyć maksymalnego prądu diody LED, wynoszacego 30...50mA.

Sterować można także przekaźnikiem (**rys. 4a**) Pod symobolem inwertera należy rozumieć któreś z poprzednich rozwiązań (rys. 1 i 2a, b, c). Dioda włączona w kierunku zaporowym zabezpiecza inwerter przed przepięciami indukującymi się w momencie załączania i wyłączania przekaźnika, które mogłyby uszkodzić tranzystory. Jest to dioda krzemowa dowolnego typu (np. impulsowa). W miejsce diody można włączyć kondensator elektrolityczny spełniający tą samą funkcję. **Rys. 4b** przedstawia bardziej oszczędny energetycznie układ. Jak wiadomo do załączenia przekaźnika potrzebny jest określony prąd. Po zadziałaniu, prąd podtrzymujący może być znacznie mniejszy. W momencie załączenia, kondensator jest rozładowany i stanowi zwarcie. Przez przekaźnik płynie maksymalny prąd. Po chwili rozładowuje się i prąd jest ograniczony wartością rezystora. Układ ma pewną wadę objawiającą się tym, że szybkie wyłączenie i włączenie przekaźnika nie spowoduje jego zadziałania. Po prostu kondensator nie zdąży się rozładować. **Rys. 5** przedstawia układy z galwanicznym oddzieleniem obwodów za pomocą transoptorów. Jest to wręcz konieczne przy sterowaniu urządzeniami podłączonymi do sieci energetycznej. W układzie z rys. 5b rezystor w obwodzie kolektora transoptora nie jest konieczny przy współpracy z portami 8520, 6520, 6522, 6526. Dla dociekliwych dlaczego tak jest, na **rys. 6** przedstawiam układ wyjściwy portu. Jak widać gdy port pracuje jako wyjście, można na nim wymusić stan niski. Mikroprocesor odczyta go poprawnie. Sztuczka ta jest stosowana w C-64 do odczytu stanu joisticka (ten sam port obsługuje też klawiaturę). W przypadku układu 8255 sytuacja taka jest niemożliwa, ponieważ stan linii portu pracującego jako wyjście jest brany z wewnętrznego rejestru.

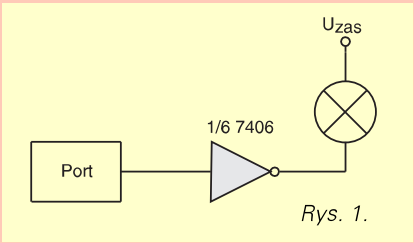
Na **rys. 7a** przedstawiono detektor przejścia napięcia sieci przez zero. Do obniżenia napięcia wykorzystano rezystor. Ze względu na dużą wydzielaną moc układ jest nieekonomiczny. Rezystor jest duży, mocno się nagrzewa , co zmusza do zapewnienia jego wentylacji.

Na **rys. 7b** przedstawiono inny obwód zmiejszający napięcie, gdzie rolę rezystancji spełnia reklantacja kondensatora. Waha się ona w granicach 100nF do 2,2uF. Rezystor R1 ogranicza prąd w momencie włączania do bezpiecznej wartości, kiedy to kondensator stanowi zwarcie. Wartość jego waha się w granicach kiluset omów. R2 o wartości około 1Mohm rozładowuje go po wyłączeniu zasilania. Układ reaguje na dodatnie połówki napięcia.

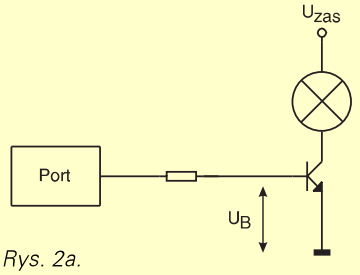
Dioda włączona równolegle do LED transoptora zabezpiecza ją przed spaleniem przy ujemnych połówkach.

Jeśli wymagana jest reakcja na obydwie połówki napięcia należy zastosować układ z rys. 7b.

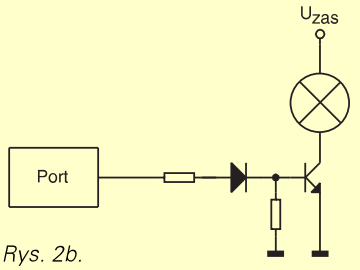
Triakami najwygodniej sterować układem z **rys. 8**. Optotriaki MOC3040 zawierają układ przejścia napięcia przez "zero", dzięki czemu minimalizujemy zakłócenia. Należy zwrócić uwagę na maksymalną wydajność prądową portu. W konsekwencji bez dodatkowych buforów można przyłączyć masymalnie dwa transoptory z rys. 5a i dwa optotriaki z rys. 8.



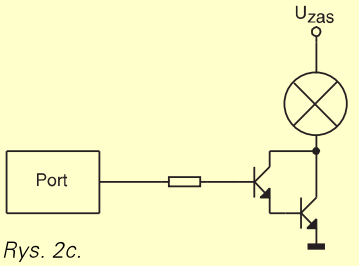
Rys. 1.



Rys. 2a.



Rys. 2b.

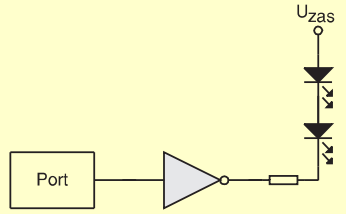


Rys. 2c.

Gdy liczba wyjść portu jest zbyt mała można ją zwiększyć używając rejestrów 4094. Układ jest statycznym rejestrem przesuwającym z możliwością podtrzymania informacji wprowadzonej szeregowo. Na **rys. 9** przedstawiono budowę wewnętrzną rejestru. Składa się on z trzech części:

1. Ośmiobitowego rejestru przesuwającego z wejściem szeregowym, wyjściem równoległym.
2. Ośmiobitowego zatrzasku (tzw. LATCH).
3. Ośmiu buforów trójstanowych.

Sterowanie rejestru odbywa się za pośrednictwem czterech wyprowadzeń. Do wprowadzania informacji służy wejście D. Przesuwana jest ona w takt narastających zboczy inpulsów zegarowych CL. Przepisanie danych z rejestru do zatrzasków nastąpi przy wysokim stanie wejścia ST. W niskim zatrzask pamięta ostatnio wprowadzoną informację. Wejście OE steruje buforami trójstanowymi. Niski poziom na nim powoduje



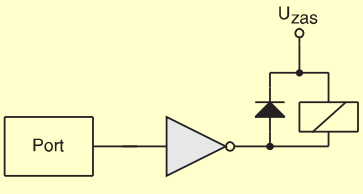
Rys. 3.

przejście buforów w stan trzeci. Wyjścia QS'~ są przeznaczone do kaskadowego łączenia rejestrów.

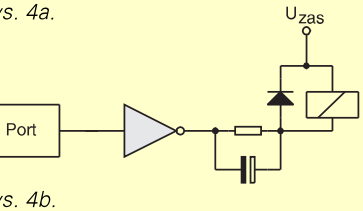
Na **rys. 10** przedstawiono przykładowy sposób połączenia rejestrów 4049.

Jeśli osim bitów portu to zdecydowanie za mało, można zastosować układ pokazany na **rysunku 11**. Pokazuje on rozbudowę portu przy użyciu zatrzasków i demultipleksera. Na bity D0-D3 portu wystawiamy daną do zatrzaśnięcia w 7475. Równocześnie na bity D4-D7 adres układu U2 do U5 (max U17). Powoduje to krótki ujemny impuls na linii STB. Demultiplekser U1 generuje na odpowiednim wyprowadzeniu (S0-S15) impuls dodatni. Powoduje on zatrzaśnięcie danych w zatrzaskach. Maksymalnie rozbudowując układ możemy sterować szesnastoma zatrzaskami co daje 16 x 4 = 64 linie (8 portów ośmiobitowych). Jeśli nie potrzeba nam tak dużej liczby można wykorzystać zamiast 4514 układ 4028. Bity D4, D5, D6 przyłączmy do wejść A, B, C układu 4028, natomiast STB do wejścia D. Tym sposobem można sterowć 10 układów 7475. Ze względu na technologię wykonania układu U1 zatrzaski muszą być serii 74HC, 74HCT, ostatecznie 74LS.

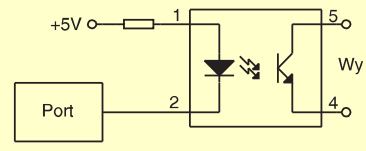
Zajmijmy się teraz sposobem zwiększenia liczby wejść. Na **rys. 12** przedtawiono przykład wykorzystania rejestrów 74165. Jest to rejestr ośmiobitowy z wejściem równoległym, wyjściem szeregowym. Niski poziom logiczny na wejściu LOAD wpisuje informację z wejść A...H do wewnętrznych rejestrów. Na wyjściu QH pojawi się stan z wejścia H.



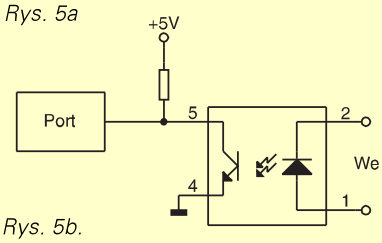
Rys. 4a.



Rys. 4b.



Rys. 5a



Rys. 5b.

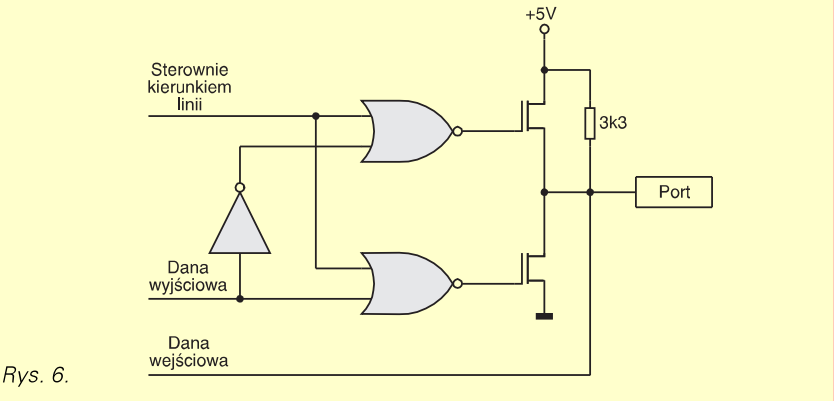
Narastające zbocze na linii zagarowej CK przy wysokim stanie wejścia LOAD przesuwa informację w rejestrze. W ten sposób po jednym impulsie zegarowym odczytamy stan, jaki był na wejściu G, po dwóch - F, po trzech -E, itd. Podczas przesuwania i wyprowadzania zawartości, do rejestru A wpisywany jest stan wejścia SER. Osiem impulsów na CK daje nam możliwość odczytania danych ze wszystkich wejść A...H. Wejście SER służy do kaskadowego łączenia rejestrów, jeśli potrzeba byłoby więcej niż osiem wejść. Układy powinny być serii 74HC, 74HCT. Jeśli użyjemy 74LS należy buforować linie CK i LOAD np. układem 7408.

Do zwiększenia liczby wejść można też wykorzystać multipleksery (**rys. 13**). Wystawiając daną na linie D0-D3 decydujemy, z którego wejścia (E0-E15) informacja zostanie przepisana na wyjście W.

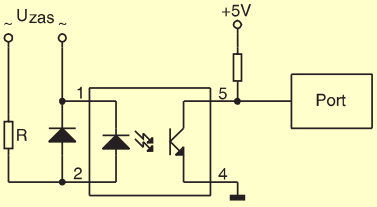
Zaznaczam, że rozwiązania tu przedstawione nie są jedyne i nie zawsze są najlepsze. Artykuł miał na celu przybliżenie czytelnikom sposobów rozwiązywania niektórych problemów. Na zakończenie mała przyjemność. Zbudujmy prosty interfejs do sterowania żarówką lub diodami LED. Pamiętajmy także, aby ZAWSZE wszelkie urządzenia przyłączać przy wyłączonym zasilaniu.

"Wklepmy" poniższy program. Dla C-64:

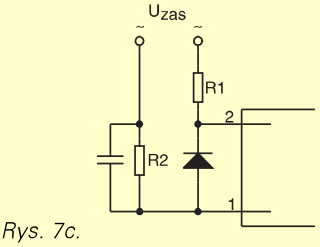
```
10  DIM  T(14)
30  FOR  X=1  TO  14  : REM
WCZYTANIE DANYCH
40  READ  A
50  T(X)=A
60  NEXT  X
70  POKE  56579,255  : REM
USTAWIENIE  LINI  JAKO  WYJSCIA
90  FOR  X=1  TO  14  : REM
WYSWIETLANIE
100  POKE  56577,T(X)
```



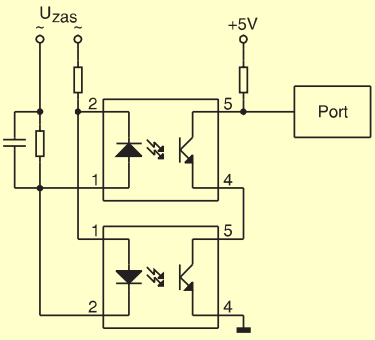
Rys. 6.



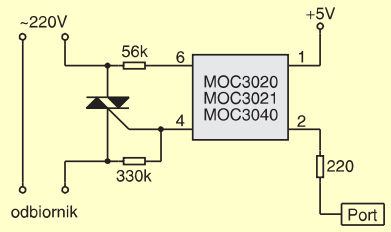
Rys. 7a.



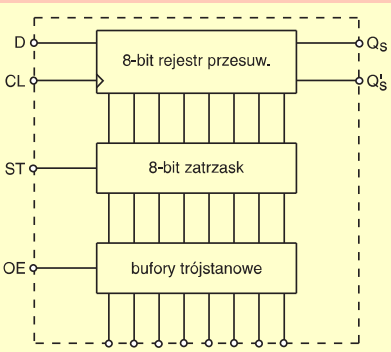
Rys. 7c.



Rys. 7b.



Rys. 8.



Rys. 9.

```
110 FOR Y=1 TO 50 :NEXT : REM
PETLA CZASOWA
120 NEXT
130 GOTO 90
140       Data      1,2,4,8,16,32,64,128
150       Data      64,32,16,8,4,2
```

Uruchomijmy go (RUN). Na przyłączonych zarówkach lub diodach ujrzymy tzw. płynący punkt. Zmieniając liczbę 50 w linii 110 wpływamy na prędkość poruszania się punktu. Program przerywamy naciskając STOP. Dla Amosowców natomiast program wygląda tak:

```
10  Dim  T(14)
20   Restore   140
30  For  X=1  To  14
     40  Read  A
     50  T(X)=A
60  Next  X
70   Poke   $BFE301,255
80  Do
     90  For  X=1  To  14
       100  Poke  $BFE101,T(X)
      110  Wait  5
      120  Next
130   Loop
140       Data      1,2,4,8,16,32,64,128
150       Data      64,32,16,8,4,2
```

Parametrem instrukcji Wait w linii 110 regulujemy szybkość poruszania punktu. Naciskając równocześnie kombinację CTRL i C przerywamy program.

Dla niecierpliwych proponuję wykonanie układu z **rys. 14**. W programie należy zmienić jedną z linii. W wersji dla C-64 linię 100 na:

```
POKE  56577,T(X)  XOR  255
```

w przypadku Amigi:

```
Poke  $BFE101,T(X)  Xor  255
```

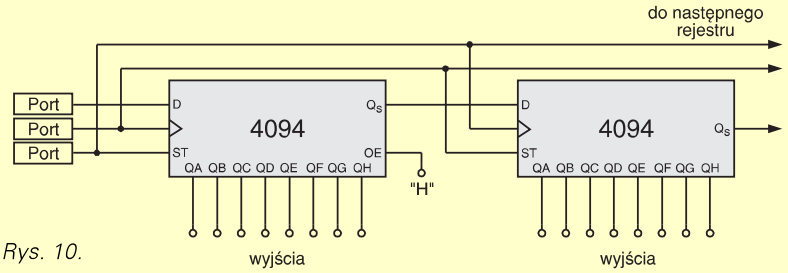
Jest to spowodowane odwrotnym włączeniem diod. Przy zmianach w programie (linie DATA) nie należy przesadzać z ilością zaświeconych diod (max 2). Jest to spowodowane maksymalną wydajnością prądową portu.

To był prościutki wstęp. Commodore i Amiga potrafią dużo, dużo więcej. Czy słyszeliście kiedyś o zegarze DCF? A jeśliby nasz komputer był zegarem o dokładności +/- 1 sekunda na 100000 lat i sterowałby dużym wyświetlaczem, wyświetlał temperaturę? A może stanie się samouczącym "pilotem", włączającym o wybranej godzinie, co tylko dusza zapragnie? Może wykonać z jego pomocą programator EPROMów? Możliwości jest wiele.

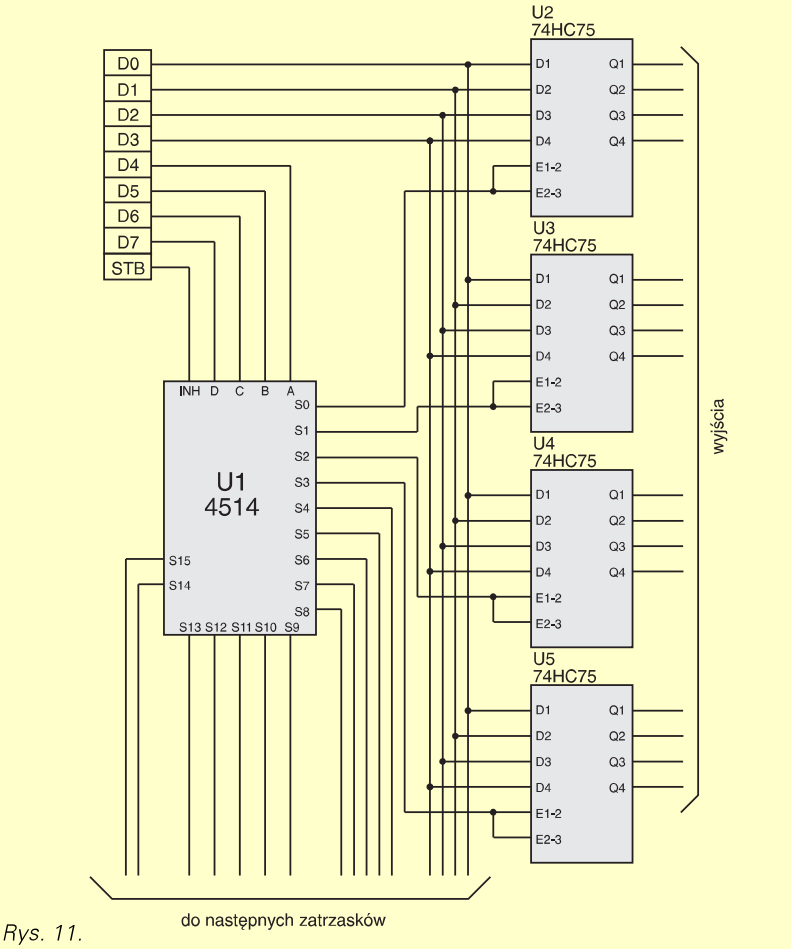
Proszę o listy z uwagami na temat cyklu oraz z propozycjami układów do zrealizowania z użyciem komputera.

Sławomir Skrzyński

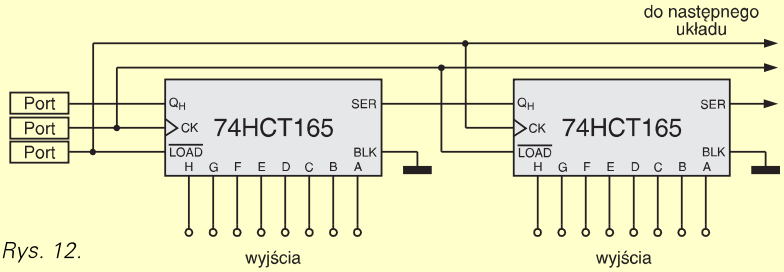
Literatura:
Układy scalone serii UCA64/UCY74. Parametry i zastosowania - WKŁ Warszawa 1990.
USKA Układy Analogowe 2/1994.
Mikroprocesor 6502 i jego rodzina - NOT SIGMA Warszawa 1989.
Kurs asemblera dla początkujących - HELION 1994.
Commodore 64 od środka - FET 1992.
User's Guide A1200 - Commodore.
Commodore 64 Bedienungshandbuch - Commodore.



Rys. 10.

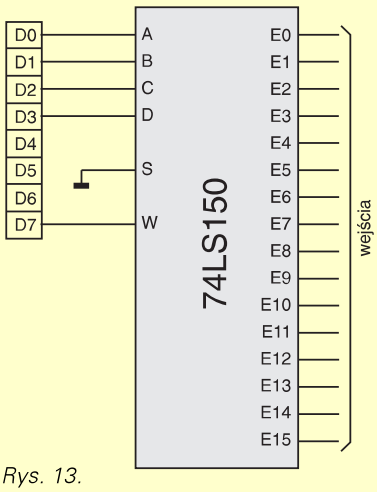


Rys. 11.

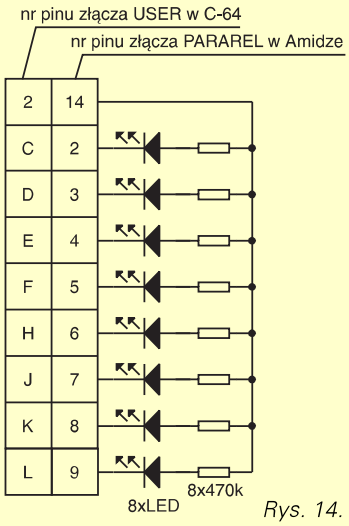


Rys. 12.

Dołączanie jakichkolwiek urządzeń do portu komputera musi być przeprowadzone po wyłączeniu zasilania.



Rys. 13.



Rys. 14.