biuletyn informacyjny

1-6
'80



MERA IMM OBIEKTOWE SYSTEMY KOMPUTEROWE

Veljko MILUTINOVIĆ

# PROJEKTOWANIE MIKROPROCESOROWYCH URZĄDZEŃ TELEKOMUNIKACYJNYCH

Zjednoczenie Przemysłu Automatyki i Aparatury Pomiarowej „MERA"
Instytut Maszyn Matematycznych „MERA IMM" Branżowy Ośrodek INTE

Tytuł oryginału: PROJEKTOVANJE TELEKOMUNIKACIONIH UREĐAJA POMOCU MIKRORAČUNARA

Wydawca: Zespół TELEKOMUNIKACJA Instytutu im. Mihailo Pupin w Belgradzie

Tłumaczyli:
Piotr DOBROŁĘCKI
Zbigniew HĄYDER

Recenzenci:
Andrzej KOJEMSKI
Czesław SYC

Opracowanie redakcyjne:
Hanna Drozdowska
Romana Nitkowska

Druk IMM zam. 100/81 nakł. 670 egz.

Biuletyn Informacyjny OBIEKTOWE SYSTEMY KOMPUTEROWE



Rok XVIII Nr 1-6 1980

Spis treści Содержание Contents

MILUTINOVIĆ V.: Projektowanie mikroprocesorowych urządzeń telekomunikacyjnych

МИЛЮТИНОВИЦЬ В.: Проектирование микропроцессорных телекоммуникационных устройств

MILUTINOVIĆ V.: Design of microprocessor telecommunication devices

D W U M I E S I Ę C Z N I K

Wydaje:
CENTRUM NAUKOWO-PRODUKCYJNE TECHNIK KOMPUTEROWYCH I POMIARÓW
I N S T Y T U T  M A S Z Y N  M A T E M A T Y C Z N Y C H
Branżowy Ośrodek Informacji Naukowej Technicznej i Ekonomicznej



Adres redakcji: ul.Krzywickiego 34, 02-078 Warszawa
tel.28-37-29 lub 21-84-41 w.244

Przedmowa

do wydania polskiego

Staraniem Instytutu Maszyn Matematycznych grono czytelników Biuletynu Informacyjnego otrzymuje do rąk narzędzie pomocne przy projektowaniu urządzeń wykorzystujących mikroprocesory. Mikroprocesory nie są już rozpatrywane jak "obce ciała" stosowane tylko w laboratoriach badawczych zajmujących się wymyślnymi ich aplikacjami. Stały się pełnowartościowymi składnikami tworzywa przemysłowego, które czynią produkty tego przemysłu wygodniejszymi i bardziej niezawodnymi, tym samym atrakcyjniejszymi.

Techniki mikrokomputerowe wyszły poza obszar projektowania różnego rodzaju programatorów czy sterowników. Zajęły poczesne miejsce w projektowaniu systemów komputerowych dobrze przystosowanych do potrzeb użytkowników.

Książka Veljko Milutinovića ilustruje tę sytuację oryginalnymi zastosowaniami mikroprocesorów oraz propozycjami nowych podejść do projektowania wydajniejszych układów mikrokomputerowych na przykładzie telekomunikacji. Wykład obejmuje podstawowe definicje, podaje wzmianki o technologiach unipolarnych i bipolarnych oraz omawia szczegółowo własności mikroprocesora Intel 8080 i innych układów LSI należących do jego rodziny.

Na uwagę zasługuje ciekawe podejście Autora do układów mikrokomputerowych jako naturalnego rozwinięcia pojęć układów logicznych małej skali integracji. Konsekwencją stało się rozróżnianie metod projektowania układów mikrokomputerowych bez sygnałów przerwań i układów wykorzystujących te sygnały. Zaakcentowane zostały zagadnienia projektowania układu przerwań zewnętrznych jak i obsługi programowej przerwań. Aparat języka symbolicznego ASSEMBLER stosowany jest wszędzie z widoczną swobodą. Ciekawe są rozważania poświęcone operacjom arytmetycznym przydatnym w zastosowaniach, a nie objętym listą rozkazów mikroprocesora Intel 8080. Zilustrowano je przykładami programów realizujących te operacje. Między innymi przedstawione zostały zadania protokołu SDLC i ich wykonywanie przy użyciu układów typu USART.

Wykład uzupełniają dodatki pomocne przy opracowywaniu oprogramowania i zabudowywaniu mikroprocesora.

Całość stanowi dobre, zwarte wprowadzenie w techniki komputerowe tak dla szerokiego kręgu czytelników, jak i dla osób z tą rozwijającą się dziedziną związanych.Stąd też postanowienie redakcji Biuletynu, aby książkę Veljko Milutinovića tą drogą przybliżyć polskiemu czytelnikowi.

Jestem przekonany, że publikacja ta spełni inspirującą rolę wśród pracowników inżynieryjno-technicznych przemysłu i pobudzi praktyczne jej wykorzystywanie zwłaszcza w odniesieniu do systemów o inteligencji rozproszonej.

/-/ prof.dr hab.inż. Andrzej JANICKI

Przedmowa do wydania jugosłowiańskiego

Opracowanie technologii wielkiej skali integracji - LSI umożliwia stworzenie jednostki procesorowej komputera na pojedynczym układzie scalonym - mikroprocesorze, który daje podstawę do utworzenia systemu mikroprocesorowego czyli mikrokomputera.

Telekomunikacja jest dziedziną stanowiącą szerokie pole dla zastosowania mikrokomputerów. Wykorzystanie mikrokomputerów daje możliwości uzyskania takich samych lub lepszych niż dotychczas efektów techniczno-technologicznych przy korzystniejszych warunkach ekonomicznych.

Mając na uwadze kierunki rozwojowe w technologii komputerowej i jej coraz szersze zastosowania w systemach komunikacyjnych Zespół Telekomunikacji Instytutu im. Mihailo Pupin z Belgradu ze szczególnym zainteresowaniem śledził i uczestniczył w rozwoju tej dziedziny. Uwzględniając również zainteresowanie specjalistów, którzy w różny sposób związani są z tym zagadnieniem, aktualność problematyki oraz stosunkowo niewielką literaturę fachową w języku serbochorwackim w tej dziedzinie, Instytut im. Mihailo Pupin podjął decyzję o wydaniu pracy mgr inż. Veljko Milutinovića.

Publikacja ta przeznaczona jest dla inżynierów telekomunikacji, pragnących zaznajomić się z podstawami techniki mikroprocesorowej; mogą z niej korzystać również inżynierowie elektrycy każdej innej specjalności. Zgodnie z taką koncepcją dokonano usystematyzowania treści.

W pierwszej części przedstawiono podstawowe pojęcia związane z architekturą, technologią i zastosowaniem mikrokomputerów, ze szczególnym uwzględnieniem zastosowań w telekomunikacji. W części drugiej, na dwóch konkretnych, pod względem programowym prostych przykładach, przedstawiono podstawowe pojęcia z dziedziny sprzętu i podstawowe zasady projektowania opartego na mikrokomputerach. W tej części nie poruszono jeszcze problemów telekomunikacyjnych, wychodząc z założenia, że czytelnika nie wprowadzonego w te zagadnienia nie należy obciążać problemami zastosowań, dopóki nie opanuje najbardziej podstawowych pojęć techniki mikroprocesorowej. Przedstawione przykłady nie mają praktycznej wartości, natomiast są dobrane pod względem dydaktycznym. W trzeciej części cała uwaga skierowana jest na projektowanie zespołów i urządzeń telekomunikacyjnych, odpowiednie oprogramowanie oraz konieczne dla tego celu zmiany i uzupełnienia w strukturze sprzętowej mikrokomputerów. Przedstawione przykłady są względnie proste, jednakże zawierają większość elementów potrzebnych do realizacji urządzeń telekomunikacyjnych.

W dalszym ciągu wprowadza się podstawowe pojęcia z techniki mikroprocesorowej i to równolegle z problemami, które za ich pomocą są rozwiązywane.

W części końcowej umieszczono kilka przydatnych załączników, między którymi wyróżnia się załącznik o rozwoju urządzeń opartych na mikrokomputerze oraz przegląd literatury dotyczącej zastosowania mikrokomputerów w telekomunikacji.

Całość pracy bazuje na wykorzystaniu mikroprocesora Intel 8080 oraz odpowiednich kompatibilnych układach towarzyszących. Wyrażamy podziękowanie wszystkim, którzy swoimi konstruktywnymi uwagami przyczynili się do podniesienia jakości tej pracy. Mamy nadzieję, że jej wydanie przyczyni się do rozwoju wiedzy fachowej w dziedzinie zastosowania mikrokomputerów, a szczególnie w dziedzinie telekomunikacji.

Zespół TELEKOMUNIKACJA
Instytutu Mihailo Pupin

/-/ inż. Djordje Rosić
Wicedyrektor

MICROCOMPUTER-BASED DESIGN OF TELECOMMUNICATION EQUIPMENTS

This book is one of the results of continuous efforts made by the Institute "Mihailo Pupin", Beograd, Yugoslavia, to follow the latest development trends in the field of telecommunications and is, also, the result of the work of the author himself, during several years, on the matters of application of microcomputers in telecommunications. It has been prepared with a view of serving to graduate telecommunication engineers who wish to prepare themselves for the task of designing microcomputer-based telecommunication equipments. The book has been executed in four parts.

Part One (pp.8-17) explains the basic notions dealing with architecture (pp.8-12), with application (pp.12-17) and with technology (pp.17-21) of microcomputers, having specially in view their application in telecommunication, in its broadest sense.

In Part Two (pp.23-83) basic notions of microcomputer technique have been presented making use of two simple examples.

In Part Three (pp.86-149) telecommunication problems have been introduced: automated branch exchange (pp.86-92), digital filter (pp.93-116), voiceband data modem (pp.117-148) and front-end processor (pp. 149-157).

Part Four (pp.160-172) includes the following Annexes: development of microcomputer-based equipments (pp.160-166), microcomputer protection (pp.167-168), prices of microcomputer family parts, support equipment and software (pp.168-170), current literature dealing with the application of microcomputers in telecommunications, more than 300 references (pp.172-186), etc.

Yugoslav publisher: Telecommunication Department of the Institute "Mihailo Pupin", Volgina 15, 11000 Beograd, Yugoslavia; 1978.

POLISH PUBLISHER: INSTYTUT MASZYN MATEMATYCZNYCH,
KRZYWICKIEGO 34, 02-078 WARSZAWA, POLAND, 1981.

## CZ.I. WSTĘP

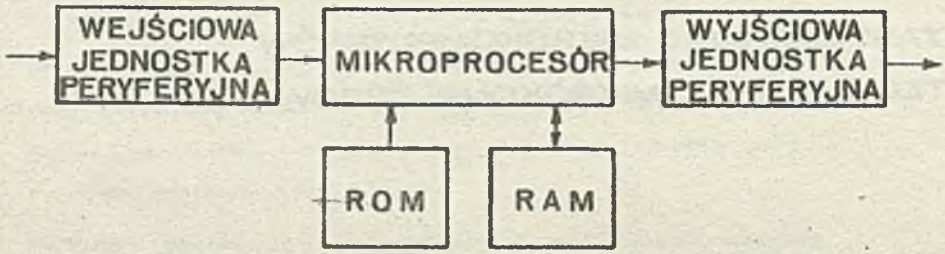
# WPROWADZENIE DO MIKROKOMPUTERÓW

W początku lat siedemdziesiątych w technologii półprzewodników powstała tzw. wielka skala integracji /Large Scale Integration/, w skrócie technologia LSI. Technologia LSI umożliwia wykonanie całej jednostki przetwarzającej /Central Processing Unit - CPU/ w tylko jednym układzie scalonym. Ten układ nazywa się mikroprocesorem.

## MIKROKOMPUTER

Mikroprocesor, rozpatrywany osobno, nie przedstawia wartości użytkowej, ponieważ nie istnieje zastosowanie, dla którego tylko sam mikroprocesor jest potrzebny i wystarczający. Mikroprocesor uzyskuje wartość praktyczną dopiero wówczas, gdy wokół niego zbuduje się cały system. System taki nazywa się mikrokomputerem, zaś mikroprocesor stanowi jego "serce". Zamiast terminu mikrokomputer stosuje się także termin system mikroprocesorowy. Terminologia w tej dziedzinie jeszcze ciągle nie jest ujednolicona. Pracą mikrokomputera steruje program. Program określa typ i kolejność operacji kontrolnych, logicznych i arytmetycznych, jakie mikroprocesor musi wykonać.

Poza mikroprocesorem, elementami mikrokomputera są jeszcze pamięci ROM i RAM, jak również jednostki peryferyjne /Input/ Output Units/ różnych typów. Schemat blokowy mikrokomputera przedstawiono na rys.1.




Rys.1. Schemat blokowy mikrokomputera

## MIKROPROCESOR

Jak już powiedziano, mikroprocesor stanowi jednostkę centralną komputera. Składa się z jednostki arytmetyczno-logicznej /Arithmetic-Logical-Unit - ALU/ i z jednostki sterującej /Control Unit - CU/. ALU wykonuje operacje arytmetyczne i logiczne na danych, natomiast CU generuje wszystkie potrzebne sygnały sterujące niezbędne do synchronizacji procesu wewnątrz i zewnątrz CPU.

Pierwszy mikroprocesor powstał w 1971 r. /Intel 4004/, podobny był do czterobitowego kalkulatora i nie był odpowiedni do przetwarzania ogólnego /General Purpose Computing/. Pierwszy ośmiobitowy mikroprocesor ogólnego zastosowania pojawił się w 1972 r. /Intel 8008/. Jego ulepszona wersja powstała w 1973 r. /Intel 8080/ i utrzymuje się do dzisiaj jako jeden z najczęściej stosowanych mikroprocesorów. Intel 8080 jest pierwszym z serii mikroprocesorów należących do "drugiej generacji". W 1974 r. powstał mikroprocesor Motorola 6800 i wiele innych. Na czele "trzeciej generacji" znajduje się Intel 8085 i Zilog Z80.

ROM

Pamięci przeznaczone wyłącznie do odczytywania znane są w literaturze pod nazwą pamięci ROM /Read Only Memory/. W pamięci ROM zastosowanej w mikrokomputerze, znajdują się program i dane, których wartość nie zmienia się w trakcie przebiegu programu.

Wpisania programu do pamięci ROM najefektywniej dokonuje się za pomocą urządzenia zwanego programatorem PROM. Przy wprowadzaniu programu, pamięć ROM umieszcza się w programatorze PROM. Bezpośrednie wprowadzenie programu do pamięci ROM nie jest możliwe. Dokonuje się tego za pośrednictwem specjalnego modułu /płytka drukowana z zamontowanymi elementami i wtykiem/ służącego do wprowadzania programu /Personality Card/. Moduły te nie są uniwersalne. Każdy typ pamięci ROM wymaga zastosowania odpowiedniego typu modułu dla wprowadzenia programu. Po wpisaniu całego programu do pamięci ROM wyjmuje się ją z programatora PROM i umieszcza w mikrokomputerze. W trakcie pracy mikroprocesor odczytuje z pamięci ROM kolejno rozkazy i wykonuje określone, wyspecyfikowane rozkazami, operacje sterujące, logiczne i arytmetyczne. Tak więc pamięć ROM stanowi "mózg" mikrokomputera.

Mikroprocesor nie może wprowadzać danych do pamięci ROM.

Pamięci ROM mają taką właściwość, dzięki której nie następuje zniszczenie programu przy wyłączeniu zasilania, tzn. pamięć ROM jest pamięcią z zapisem nieniszczącym /Non Volatile Memory/. Istnieją również pamięci nie posiadające tej właściwości /Volatile Memory/. Są to pamięci z zapisem niszczącym. Jeżeli program jest umieszczony w pamięci z zapisem niszczącym, wówczas przed wyłączeniem zasilania należy go umieścić w jakiejś innej pamięci z zapisem nieniszczącym. Takimi nośnikami pamięciowymi są poza pamięcią ROM: taśma papierowa /Paper Tape/, taśma magnetyczna /Magnetic Tape/, dysk /Disc/, domenowe pamięci magnetyczne z zapisem nieniszczącym /Magnetic Bubble Memories/ i inne.

Taśma papierowa jest najbardziej prymitywnym rodzajem nośnika pamięci, spotykanym w praktyce. Zamiast taśmy papierowej w ostatnim czasie coraz częściej stosowana jest taśma magnetyczna. Wiele zastosowań mikroprocesorów wymaga stosowania pamięci dyskowych. Istnieje kilka rodzajów jednostek dyskowych: z nieruchomą głowicą /Fixed-Head-Disk/, z głowicą ruchomą /Moving-Head-Disk/, dysk elastyczny /Floppy Disk/ i inne. Przy pracy z mikroprocesorem najczęściej stosuje się ostatni rodzaj, zwany także dyskietką /Diskette/. Oparty jest na dysku z głowicą ruchomą.

Wspomniane wyżej pamięci magnetyczne z zapisem nieniszczącym, wykonywane z materiału półprzewodnikowego, pojawiły się dopiero niedawno. Dotychczas nie mają one szerokiego zastosowania praktycznego, ale oczekuje się, że w przyszłości wyprą dyski elastyczne.

Po tej krótkiej dygresji powróćmy do pamięci ROM. Istnieją dwa rodzaje pamięci ROM.

Cechą charakterystyczną pierwszego rodzaju pamięci ROM jest wpisywanie programu na stałe. Oznacza to, że wpisany program musi być poprawny, ponieważ po wpisaniu nie można dokonać żadnych zmian. Rodzaj ten dzieli się na dwa podrodzaje. Do pierwszego podrodzaju należą tzw. elektrycznie programowalne pamięci ROM lub w skrócie pamięci PROM /Electrically Programmable ROM/. Do pamięci PROM program wpisywany jest przez samego użytkownika za pomocą wyżej wspomnianego programatora PROM.

Do drugiego podrodzaju należą pamięci ROM z maską lub w skrócie pamięci MPROM /Mask Programmable ROM/. Do pamięci MPROM program wpisywany jest przez producenta na życzenie użytkownika. W literaturze często zamiast skrótu MPROM spotyka się skrót ROM.

Cena jednostkowa pamięci MPROM jest znacznie niższa od ceny odpowiedniej pamięci PROM. Jednakże celem stworzenia pamięci MPROM należy uprzednio wykonać maskę, która jest bardzo droga. Cena jej wynosi od 600 do 1000 dolarów USA. Dlatego też pamięć MPROM stosowana jest w urządzeniach produkowanych wieloseryjnie, zaś pamięć PROM w urządzeniach produkowanych w krótkich seriach.

Drugi rodzaj pamięci ROM daje możliwość usunięcia wpisanego programu lub wpisania nowego programu. Są to tzw. pamięci ROM z wymazywaniem lub w skrócie pamięci EPROM /Erasable and Electrically Reprogrammable PROM/. Wymazywania dokonuje się przez poddanie pamięci ROM działaniu promieni ultrafioletowych o określonej długości fali. W tym celu używa się specjalnej lampy.

Pamięci EPROM są odpowiednie do prac rozwojowych i badawczych. Cena pamięci EPROM i PROM jest podobna.

Pamięci EPROM i pamięci MPROM zazwyczaj produkowane są w parach z odpowiadającymi sobie parametrami elektrycznymi oraz z odpowiednim rozmieszczeniem końcówek. Wtedy pierwsza służy do prac rozwojowych, a druga do eksploatacji.

W 1976 r. na rynku USA i Japonii pojawił się nowy typ pamięci ROM, w którym oprócz odczytywania możliwe jest również i wpisywanie. Jednakże, gdy proces odczytywania trwa około 1 μs, to proces wpisywania około 1 ms. Poza tym pamięci te są bardzo kosztowne i dlatego też stosowane są tylko w specjalnych zastosowaniach przemysłowych i wojskowych, gdzie występuje potrzeba wpisywania małej ilości danych. Znane są pod nazwą EAROM /Electrically Alterable ROM/ lub RMROM /Read-Mostly ROM/.

RAM

Program często wymaga, aby poszczególne dane były okresowo zapamiętywane. Oznacza to, że potrzebne są pamięci umożliwiające zarówno wpisywanie jak i odczytywanie. Taką własność posiadają tzw. pamięci RWM /Read Write Memory/.

Jednakże w praktyce rzadko używa się terminu RWM. Pamięci te częściej nazywane są pamięciami RAM /Random Access Memory/. W ten sposób podkreśla się ich jeszcze jedną własność. Dane można wpisać w dowolną komórkę pamięciową lub odczytać z dowolnej komórki pamięciowej.

W półprzewodnikowych pamięciach RAM, w odróżnieniu od pamięci ferrytowych, przy wyłączaniu zasilania zawartość pamięci RAM ulega zniszczeniu. Jeżeli zapamiętane dane należy przechować, do ponownego użycia stosuje się taśmę papierową, taśmę magnetyczną, dysk elastyczny itp.

Jednostki peryferyjne

Stosowanie mikroprocesorów wymaga zawsze wprowadzenia określonego sygnału cyfrowego lub skwantowanego sygnału analogowego z zewnątrz /External Logic/, jego opracowania i przekazania na zewnątrz. Opracowanie sygnału dokonywane jest w mikroprocesorze, jednak sygnału nie można bezpo-

średnio wprowadzić do mikroprocesora. Dokonywane jest to bezpośrednio za pomocą tzw. jednostki peryferyjnej.

UKŁADY SCALONE

Należy rozróżnić pojęcia: układ scalony - w rozumieniu kawałka półprzewodnika /Chip/ i układ scalony - w rozumieniu plastikowego lub ceramicznego opakowania z metalowymi końcówkami, we wnętrzu którego znajduje się obwód scalony /Dual-In-Line Package, Quad-In-Line Package/. W dalszej treści dla obu pojęć używać będziemy tego samego terminu, ale z kontekstu będzie wyraźnie wynikać różnica.

Wszystkie elementy mikrokomputera, zarówno mikroprocesor, jak i pamięci i jednostki peryferyjne produkowane są przede wszystkim w postaci układów scalonych /DIL Package, QIL Package/. Liczba końcówek w tych układach wynosi 16,18,20,24,28,40,42 lub 64. Wyjątek stanowią mikroprocesory modularne w obudowie zatapianej /TDY-52D firmy Teledyne/.

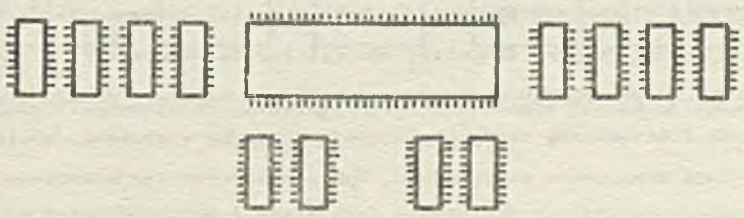
Chociaż istnieje możliwość zrealizowania całego procesora na jednym układzie /Chip/, w pierwszych mikroprocesorach tego nie robiono. W mikroprocesorze Intel 8080 przeważająca część procesora została zrealizowana w jednym dużym układzie scalonym. Jednak część funkcji sterujących jest realizowana przez dwa mniejsze układy. Fakt ten odpowiada stanowi technologii LSI w okresie, gdy był opracowywany Intel 8080. Pojawił się on w grudniu 1973 r. Natomiast mikroprocesor Intel 8085, opracowany w 1976 r. zrealizowany jest na jednym układzie.

W 1977 r. pojawił się pierwszy mikrokomputer w jednym układzie Intel 8748/8048. W jednym układzie zrealizowano procesor, 64 bajtową pamięć RAM, pamięć ROM o pojemności 1 kbajta, jak również wiele innych funkcji. Mikrokomputer Intel 8748 zawiera EPROM, zaś mikrokomputer Intel 8048 zawiera MPROM.

Zależnie od długości programu pamięć ROM tworzona jest w jednym, dwóch lub w kilku układach. Liczba układów, z których tworzy się pamięć RAM zależy od liczby danych, jakie należy okresowo zapamiętywać.

Jednostki peryferyjne wykonuje się w jednym układzie. Jednak realne zastosowania wymagają różnych typów jednostek peryferyjnych, jak również większej liczby jednostek peryferyjnych tego samego rodzaju.

Na rys.2 przedstawiono praktyczny przykład realizacji mikrokomputera.



Rys.2. Praktyczny przykład realizacji mikrokomputera

PROGRAM

Jak już wcześniej stwierdzono, pracą mikrokomputera steruje program. Program składa się z ciągu binarnie zakodowanych rozkazów, np. w mikroprocesorze Intel 8080 kombinacja kodowa

11001001

odpowiada rozkazowi realizującemu bezwarunkowy skok z podprogramu do programu głównego. Powyższy rozkaz składa się z jednego bajta. Jedna cyfra binarna nazywa się bit /Bit/, zaś uporządkowany ciąg składający się z ośmiu cyfr binarnych - bajt /Byte/. Na przykład w mikroprocesorze Intel 8080 rozkazy mogą składać się z jednego, dwóch lub z trzech bajtów. Przykładami rozkazów składających się z dwóch i trzech bajtów są:

| | | |
|---|---|---|
| | 11011011 | rozkaz dwubajtowy |
| | 00000000 | |
| i | | |
| | 11000011 | rozkaz trójbajtowy |
| | 00000000 | |
| | 00000000 | |

Kod dla rozkazów specyfikuje producent układu scalonego.

Program dokonuje obróbki danych w najszerszym znaczeniu tego słowa. W wypadku mikroprocesora Intel 8080 jedna dana może składać się z jednego lub kilku bajtów. Tak na przykład kombinacja kodowa

11001001

stanowi jedną daną o długości jednego bajta.

Z przedstawionego przykładu wynika, że taka sama kombinacja kodowa może przedstawiać zarówno rozkaz, jak i daną. Jednakże mikroprocesor dobrze rozróżnia rozkaz od danej. Dokładniej zajmiemy się tym później.

LITERATURA

D1, D5, D6, D8, F1, F2, F3, K1, K2.

## ZASTOSOWANIA MIKROKOMPUTERÓW

Mikrokomputer ma dwa główne zastosowania.

- Po pierwsze, opierając się na mikrokomputerze można zrealizować komputer ogólnego lub szczególnego przeznaczenia /General Purpose Computer, Special Purpose Computer/, a także można wykonać minikomputer standardowy. W tym wypadku do zestawu pokazanego na rys.1 dodaje się czytnik i dziurkarkę kart, czytnik i dziurkarkę taśmy papierowej, drukarkę wierszową, monitor ekranowy oraz w miarę potrzeby inne urządzenia peryferyjne. Takim zastosowaniem mikrokomputerów nie będziemy się tutaj zajmować.
- Po drugie, mikroprocesor może być traktowany jako cyfrowy element logiczny i jako taki może być wstawiony do każdego systemu cyfrowego w miejsce wielu innych cyfrowych układów logicznych /np. z rodziny 7400 lub innej/.

Jako układ logiczny mikroprocesor jest najczęściej wbudowywany w aparaturze pomiarowej, w urządzeniach dla automatyki i sterowania procesami, jak również w urządzeniach telekomunikacyjnych w najszerszym rozumieniu tego pojęcia.

Ze względu na charakter tej książki, w dalszym tekście uwaga zostanie głównie zwrócona na zastosowania mikroprocesorów w telekomunikacji, zaś wszystkie pozostałe zastosowania mikroprocesorów, nawet częściej spotykane w praktyce, będą tylko wspomniane.

PRZYRZĄDY POMIAROWE

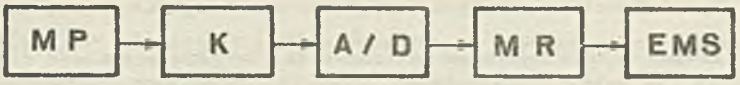
Schemat blokowy przyrządu pomiarowego opartego na mikrokomputerze przedstawiony jest na rys.3.


Rys.3. Schemat blokowy przyrządu pomiarowego opartego na mikrokomputerze

Przetwornik pomiarowy /MP/ służy do pomiarów określonej wielkości fizycznej możliwej do zmierzenia w sposób względnie prosty, np. ciśnienie w rurze. Zmiany tej wielkości w czasie są następnie przetwarzane /K/ na przebiegi czasowo napięcia elektrycznego. Po konwersji analogowo-cyfrowej /A/D/ otrzymuje się cyfrowe wartości napięcia elektrycznego w poszczególnych chwilach. Dla mikrokomputera /MR/ są to dane wejściowe. W mikrokomputerze, zgodnie z teorią kwantowania, podlegają one określonym operacjom arytmetycznym, charakteryzującym związek między wielkością fizyczną mierzoną bezpośrednio a wielkością fizyczną określoną pośrednio, której pomiar bezpośredni jest utrudniony. Może to być np. szybkość przepływu przez rurę. Wynik operacji arytmetycznych przekazuje się na ekran przyrządu pomiarowego /D/.

URZĄDZENIA AUTOMATYZACJI I STEROWANIA PROCESAMI

Schemat blokowy urządzenia do automatyzacji i sterowania procesem opartego na mikrokomputerze przedstawiono na rys.4.




Rys.4. Schemat blokowy urządzenia do automatyzacji i sterowania procesami opartego na mikrokomputerze

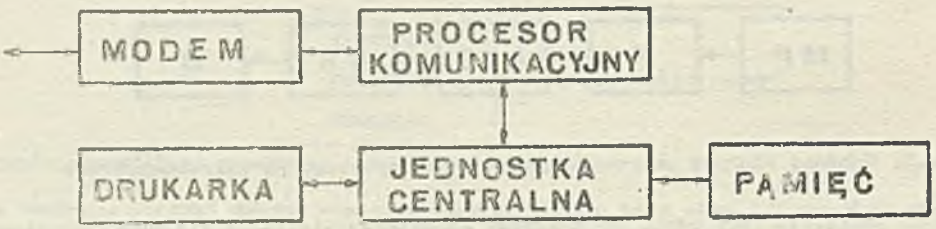
Opierając się na arytmetycznych i logicznych operacjach w mikrokomputerze przekazuje się określone komendy do różnych zespołów elektromechanicznych /EMS/, w wyniku których np. poszczególne zawory na różnych przewodach są otwierane lub zamykane.

URZĄDZENIA TELEKOMUNIKACYJNE OGÓLNEGO PRZEZNACZENIA

Mikrokomputery znalazły szerokie zastosowanie w telekomunikacji. Liczne urządzenia telekomunikacyjne mogą być wykonane za pomocą mikroprocesorów. Są to najczęściej: procesor komunikacyjny w sieci komputerowej służący do przekazywania danych kanałami telefonicznymi, automatyczna centrala telefoniczna, modem do przekazywania danych kanałem telefonicznym oraz filtr cyfrowy do obróbki sygnału telekomunikacyjnego.

Procesor komunikacyjny

Schemat blokowy jednego węzła w komputerowej sieci przesyłania danych przedstawiony jest na rys.5.




Rys.5. Schemat blokowy węzła komputerowej sieci przesyłania danych

Procesor komunikacyjny /Front-End-Procesor/ służy do połączenia głównego procesora /Host Computer/ z linią transmisji danych, na wejściu której znajduje się modem.
Procesor komunikacyjny może wykonywać następujące funkcje: wywołanie urządzenia końcowego /Polling/ na liniach z większą liczbą urządzeń końcowych /Multi-Drop Lines/, wykonywanie protokołu transmisji przy przyjmowaniu i wysyłaniu bloków rozkazów /Protocol Handling/, synchronizację bloków rozkazów, konwersja kodu, korekta błędów /Forward Error Control, FEC/ lub tylko detekcja z zadaniem retransmisji /Automatic Repeat reQuest, ARQ/, jak i czasowe oraz trwałe umieszczanie w pamięci pakietów poleceń /Buffering and Storage/, itd. Wszystkie te funkcje obecnie najczęściej wykonują minikomputery. Jednak zastosowanie minikomputerów jest nieracjonalne, gdyż mają one dużo większe możliwości i szybkość niż wymagają tego wyżej wymienione funkcje.

Zastosowanie mikrokomputerów jest znacznie bardziej opłacalne. Niektóre mikroprocesory są szczególnie przydatne do stosowania w sieciach komputerowych. Przykładem jest National IMP-16.

Intel 8080 jest mikroprocesorem ogólnego zastosowania. Ponadto producent tego mikroprocesora oferuje specjalny układ scalony stanowiący jednostkę peryferyjną, który odgrywa rolę połączenia między mikroprocesorem a modemem. Nazwa robocza tej jednostki peryferyjnej brzmi USART /Universal Synchronous Asynchronous Receiver Transmitter/. Taki układ scalony ma wszystkie połączenia z modemem, które określają zalecenia CCITT*). Stan tych połączeń może być programowo zmieniany. Taki układ scalony powoduje, że mikrokomputer oparty na mikroprocesorze Intel 8080 jest odpowiedni do stosowania w sieciach komputerowych.

*) CCITT - Comité Consultatif International Télégraphique et Téléphonique - Międzynarodowy Doradczy Komitet Telegrafii i Telefonii z siedzibą w Genewie /dop. red./.

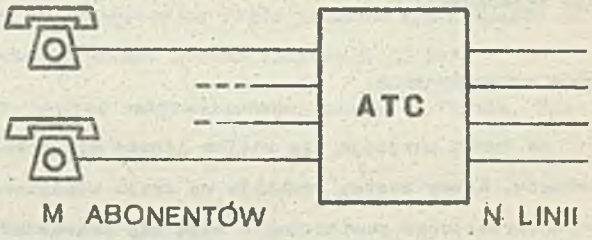
Szczegóły dotyczące zastosowania jednostki peryferyjnej USART są przedstawione w czwartym przykładzie trzeciej części tej książki.

Automatyczna centrala telefoniczna

Główno zastosowanie mikroprocesorów w telefonii związane jest z projektowaniem automatycznych central telefonicznych /ATC/. Schemat blokowy ATC podany jest na rys.6.

Gdy jeden z M abonentów wybiera jedną z N linii, mikroprocesor identyfikuje abonenta, jak również linię, na którą pragnie on "wejść" oraz działa na matrycę przełączającą i zapewnia dane połączenie.

Szczegóły dotyczące realizacji ATC opartej na mikrokomputerze są przedstawione w pierwszym przykładzie trzeciej części tej książki.

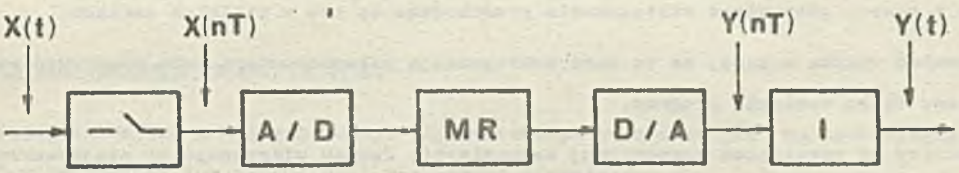



Rys.6. Schemat blokowy automatycznej centrali telefonicznej

Filtrowanie cyfrowe

Mikrokomputer jest szczególnie odpowiedni dla cyfrowej obróbki sygnałów telekomunikacyjnych. Najczęściej mamy do czynienia z cyfrowym filtrowaniem sygnałów o względnie wąskim paśmie częstotliwości. Takimi sygnałami są: mowa lub obraz w telewizji przewodowej /Videotelefon/. Dla sygnałów o szerszym zakresie częstotliwości mikrokomputer nie jest wystarczająco szybki, szczególnie jeżeli algorytm filtru cyfrowego zawiera większą liczbę operacji mnożenia.

Schemat blokowy filtra cyfrowego przedstawiony jest na rys.7.




Rys.7. Schemat blokowy filtra cyfrowego opartego na mikroprocesorze

Sygnał wejściowy X(t) jest mierzony X (nT) w określonych równomiernie rozłożonych momentach i kodowany za pomocą konwertera analogowo-cyfrowego. Zakodowane wartości są, zgodnie z teorią kwantowania wprowadzane do mikrokomputera i podlegają operacjom arytmetycznym, realizowanym przez algorytm filtra cyfrowego. Wynikiem tych operacji jest zakodowana wartość sygnału wyjściowego, która wprowadzona jest w równomiernie rozłożonych momentach do konwertera cyfrowo-analogowego. Na wyjściu konwertera D/A mamy sygnał Y (nT), natomiast na wyjściu interpolatora (We) sygnał wyjściowy Y(t).

Cyfrowy filtr dolnoprzepustowy przedstawiony jest w drugim przykładzie trzeciej części tej książki.

Modem do transmisji danych

W ostatnim czasie pojawiły się pierwsze dostępne na rynku modemy do przesyłania danych kanałem telefonicznym wykorzystujące mikrokomputery. Mikroprocesor dokonuje modulacji i demodulacji, kodowania statystycznego i ochronnego, a nawet wyrównywania. Jednakże nie jest on dostatecznie szybki, aby zsynchronizować nadajnik z odbiornikiem. Jest to realizowane w sposób tradycyjny. Modem z różnicową czterowstęgową modulacją fazową /Quaternary Differential Phase Shift Keying, QDPSK/ wykorzystujący mikrokomputer jest omówiony w trzecim przykładzie trzeciej części tego opracowania.

WYBÓR MIKROPROCESORA

Na rynku znajduje się wielka liczba mikroprocesorów. Każdy mikroprocesor ma własną listę rozkazów. Każdy zestaw rozkazów ma swoje właściwości. Właściwości jednego zestawu rozkazów w wypadku określonego zastosowania mogą się okazać dobre lub złe. Także i charakterystyka sprzętowa jednego mikroprocesora może być bardziej lub mniej odpowiednia dla danego zastosowania. Dlatego korzystne jest, gdy użytkownik zna więcej typów mikroprocesorów. Zawsze istnieje jeden mikroprocesor, który z uwagi na swoją charakterystykę sprzętową i w zakresie oprogramowania jest najodpowiedniejszy dla danego zastosowania, jednakże różnice między mikroprocesorami branymi pod uwagę są zazwyczaj bardzo małe.

ZALETY I WADY STOSOWANIA MIKROPROCESORÓW

Zalety projektowania urządzeń za pomocą mikrokomputerów są wielorakie.Urządzenie tak projektowane jest tańsze, bardziej elastyczne i ma mniejsze wymiary.

Cena mikroprocesora i innych układów, z których buduje się mikrokomputer jest względnie niska, ponieważ z powodu szerokiego zastosowania produkowane są one w wielkich seriach.

Elastyczność wynika z tego, że ta sama konfiguracja mikrokomputera może wykonywać różne funkcje. Należy tylko wymienić program.

Małe wymiary są rezultatem nowoczesnej technologii. Jednak mikrokomputer niejednokrotnie jest niedostatecznie szybki do wykonywania wszystkich postawionych mu zadań. Ma to miejsce szczególnie przy zastosowaniach w telekomunikacji. Również elementy, z których składa się mikrokomputer są wrażliwe na ładunki elektrostatyczne oraz na impulsowe zakłócenia w sieci, których moc nie powinna osiągać dużych wartości.

UWAGA

Mikroprocesory umożliwiły zastosowanie elektroniki w wielu dziedzinach, w których dawniej była ona stosowana w bardzo znikomym stopniu lub wcale. Ale mikroprocesor jest również nowym środkiem,za pomocą którego rozwiązuje się stare problemy. Jednak nowy środek przyniósł ze sobą i nowy sposób rozumowania. Na poparcie tego stwierdzenia posłużymy się przykładem.

"Zawsze" było wiadomo, że optymalny wektor współczynników wagowych adaptacyjnego filtra transwersyjnego wyraża się przez rozwiązanie układu równań liniowych. Układ taki można rozwiązywać bezpośrednio lub pośrednio, tzn. metodą iteracji.

Gdy na początku lat sześćdziesiątych rozpoczęto pracę nad teorią adaptacyjnego filtra transwersyjnego, poziom technologii pozwalał na stosowanie tylko metod iteracyjnych.
Po pierwszej pracy opartej na tej zasadzie nastąpiły inne, również o podobnych założeniach i zaczęto uważać, że adaptacyjny filtr transwersyjny może być opracowywany tylko metodami iteracyjnymi. Niedawno pojawił się artykuł, w którym opisuje się adaptacyjny filtr transwersyjny oparty na mikroprocesorach i bazujący na bezpośrednim rozwiązaniu układu równań liniowych /C 10/.

A więc zastosowanie mikroprocesorów umożliwiło zmianę zakorzenionego sposobu myślenia. Podobnych przykładów jest więcej.

UZASADNIENIE EKONOMICZNE ZASTOSOWANIA MIKROPROCESORÓW

Producenci sugerują, że zastosowanie mikroprocesora jest uzasadnione, gdy zastępuje on ponad 40 układów z rodziny układów scalonych 7400. Przy takiej kalkulacji wzięto pod uwagę tylko cenę składników.

Zastosowanie mikroprocesorów nie jest więc uzasadnione ekonomicznie. Prostsze urządzenia będą tańsze, jeżeli wykonane będą w sposób tradycyjny.

LITERATURA

A1, A2, A3, A4, B2, C1, C2, C3, C4, C5, C6, C7, C8, C9, C10, C13, C15, C23, C24, D1, D6, D8, E1, E4, E7, K1, K2.

## TECHNOLOGIA MIKROKOMPUTERÓW

Produkcja elementów, z których buduje się mikrokomputer oparta jest na technologii MOS /Metal Oxide Semiconductor/ lub technologii bipolarnej /Bypolar/.

MIKROPROCESORY

Produkcja większości dostępnych na rynku mikroprocesorów oparta jest na technologii MOS. Są to: Intel 8080, Motorola M 6800, Zilog Z-80, Fairchild F8, Rockwell PPS-8, Signetics 2650, RCA COSMAC, Mostek 3880, National Semiconductor SC/MP, Intersil 61000, Western Digital MCP 1600, American Microsystems S9209, Electronic Arrays 9002, General Instruments CP 1600, Hitachi, MOS Technology 6501, Panafacom PFL-16A, Texas Instruments TMS 9900, Toshiba TLCS-12 i inne.

Z technologią bipolarną związanych jest mniej mikroprocesorów. Są to: Intel 300 /przestarzały/, Advanced Micro Devices AMD 2900, Fairchild Macro Logic, Motorola M10800, Texas Instruments SBP0400, Transitron 1601, Scientific Micro Systems SMS-300, Monolithic Memories 6701, Fairchild 9405 i inne.

Mikroprocesory są oznaczane znakiem technologii, w której zostały zrealizowane.

Technologia MOS

Technologia MOS cechuje się dużą gęstością upakowania i niskim poborem mocy. Wadami technologii MOS są: mała szybkość oraz wrażliwość na ładunek elektrostatyczny.

Duża gęstość upakowania w technologii MOS umożliwia zrealizowanie całego procesora na jednym układzie, chociaż nie zawsze tak się dzieje /National IMP 16/. Procesory MOS mają zbiór rozkazów zdefiniowanych przez producenta. Liczba rozkazów wynosi od kilkudziesięciu do kilkuset, np. mikroprocesor Intel 8080 zawiera listę 91 rozkazów. Zadanie użytkownika jest względnie łatwe. Po pierwsze, należy połączyć mikroprocesor z układami pamięciowymi i peryferyjnymi. W ten sposób powstaje mikrokomputer. Po drugie, za pomocą załączonej listy rozkazów należy napisać program. Niestety, mikroprocesor MOS jest dla wielu zastosowań zbyt wolny.

Istnieją trzy podstawowe techniki, w których wykonuje się mikroprocesory MOS: PMOS /P-Channel MOS/, CMOS /Complementary MOS/ i NMOS /N-Channel MOS/. Pierwsze mikroprocesory /np. Intel 8080/ wykonano w technice PMOS. Jednakże techniki PMOS nie stosuje się dla nowych rozwiązań, dla których używana jest obecnie technika CMOS /rzadziej/ lub NMOS /częściej/. Podstawowymi cechami techniki CMOS /np. Intersil 6100, RCA COSMAC/ jest niski pobór mocy oraz większa odporność na zakłócenia. Najważniejszą cechą techniki NMOS /Intel 8080, Motorola 6800/ jest większa szybkość w porównaniu do innych technik MOS.

W przyszłości można oczekiwać, że mikroprocesory MOS będą nawet 10 razy szybsze niż obecnie. Dzisiejsza organizacja mikroprocesorów MOS nie jest całkowicie dostosowana do technologii MOS. Dlatego też osiąganie większych szybkości przez mikroprocesory MOS uwarunkowane jest zmianami w ich organizacji. Rezultatem wysiłków idących w tym kierunku jest mikroprocesor Intel 8085, dwa razy szybszy niż mikroprocesor Intel 8080. Poza tym oba te mikroprocesory są kompatybilne programowo.

Opanowywanie nowych technik w technologii MOS nie będzie istotnie wpływać na szybkość mikroprocesorów MOS, jednak będzie wywierać wpływ na dalszą obniżkę ich ceny.

Również proces produkcji układów scalonych ma określony wpływ na dalsze zwiększenie gęstości upakowania. Technologia LSI /100-20.000 tranzystorów w układzie/ wykorzystuje fotolitografię przy tworzeniu maski dla układów scalonych. Poza tym rozdzielczość promienia świetlnego jest ograniczona jej maksymalną wartość już w praktyce osiągnięto. Większa precyzja może być osiągnięta tylko przez zastosowanie laserów /Electron Beam Technique/. Na zastosowaniu laserów oparta jest nowa technologia VLSI /Very Large Scale Integration/ - ponad 10.000 tranzystorów w układzie. Niedawno opracowano technologię SLSI /Super Large Scale Integration/ - ponad 50.000 tranzystorów w układzie.

Dalsze zwiększenie gęstości upakowania umożliwiło pojawienie się dwóch typów nowych elementów. Z jednej strony są to kompletne 8-bitowe mikrokomputery w jednym układzie /Intel 8048/8748/, zaś z drugiej mikroprocesory 16-bitowe /Intel 8086/ oraz 32-bitowe /Zilog/ w jednym układzie.

Technologia bipolarna

Główną zaletą technologii bipolarnej jest znaczna szybkość. Wadami tej technologii jest mała gęstość upakowania i znaczny pobór mocy.

Mała gęstość upakowania w technologii bipolarnej nie pozwalała dotychczas na zrealizowanie całego procesora w jednym układzie. Dlatego starsze mikroprocesory bipolarne nie mają listy rozkazów na poziomie użytkownika /w języku asemblera/, zdefiniowanych przez samego producenta, wobec czego zadanie użytkownika jest bardziej skomplikowane.

Z tego względu nie można bezpośrednio napisać programu. Użytkownik musi więc po pierwsze uformować procesor tzn. mikroprocesor, a dopiero potem mikrokomputer. W pierwszej kolejności powinien stworzyć takie rozkazy, które będzie wykorzystywać w swoim programie, a dopiero potem napisać sam program. Rozkazy dla użytkownika tworzy się przez kombinacje rozkazów mikroassemblera, określonych przez producenta. Proces ten nazywa się mikroprogramowaniem. W trakcie mikroprogramowania użytkownik ma możliwość preferowania ze względu na szybkość tych rozkazów, których najczęściej używa. Mikrokomputer bipolarny może być mikroprogramowany w taki sposób, aby pod względem programowym był kompatybilny z dowolnym mikrokomputerem MOS. Uzyskuje się w ten sposób rozwiązanie co najmniej 10 razy szybsze.

Istnieją trzy podstawowe techniki wytwarzania mikroprocesorów bipolarnych. Są to: TTL /Transistor Transistor Logic/, ECL /Emitter Coupled Logic/ i $I^2L$ /Integrated Injection Logic/.

Pierwsze mikroprocesory bipolarne zostały wykonane techniką Schottky TTL /np. Intel 3000/. Technikę ECL /Motorola M10800/ charakteryzuje wielka szybkość, ale również i duża wrażliwość na ładunek elektrostatyczny.

Technika $I^2L$ /np. Texas Instruments SBP 0400/ oparta jest na technice DCTL /Direct Coupled Transistor Logic/. Technika $I^2L$ jest dogodna do realizacji układów analogowych i cyfrowych na jednym układzie; cechuje ją niski pobór mocy i niezwykle duża gęstość upakowania. Powstanie techniki $I^2L$ dało podstawy do powstania pierwszego mikroprocesora bipolarnego na układzie /TI 9900/. Szybkość jest nieco mniejsza niż w innych technikach bipolarnych.

$I^3L$ /Isoplanar Integrated Injection Logic/ stanowi udoskonalenie techniki $I^2L$ pod względem szybkości. Firma Fairchild reklamuje 16-bitowy mikroprocesor wykonany w tej technice.

PODSUMOWANIE

Wszystkie modemy oparte na mikroprocesorach są realizowane za pomocą mikroprocesorów bipolarnych, ponieważ mikroprocesory MOS nie są wystarczająco szybkie. Natomiast w telefonii oraz sieciach służących do przesyłania danych kanałem telefonicznym mikroprocesory MOS spełniają swoje zadania. W wypadku filtrowania cyfrowego mikroprocesor MOS może, ale nie musi być bardzo szybki. Zależy to od zakresu częstotliwości sygnału wejściowego oraz od długości algorytmu obsługi funkcji przenoszenia filtru.

W wypadku urządzeń pomiarowych, jak również przy automatyzowaniu i sterowaniu procesami mikroprocesory MOS w zasadzie spełniają wymagania pod względem szybkości.

RAM

Istnieją dwa typy pamięci RAM: statyczne i dynamiczne.

Statyczne pamięci RAM

W pamięciach statycznych RAM wykorzystywane są przerzutnikowe elementy pamięciowe. Przy włączaniu zasilania w komórkach statycznej pamięci RAM powstaje przypadkowy, ale najczęściej przy każdym włączeniu taki sam, stan. Powodem tego jest to, że elementy przerzutnikowe, z których utworzona jest pamięć RAM nie są absolutnie symetryczne. Jeden z dwóch stanów jest zawsze bardziej uprzywilejowany.

Statyczne pamięci RAM są w obsłudze proste, ponieważ są bezpośrednio połączone z mikrokomputerem.

Maksymalna pojemność statycznych pamięci RAM wynosi obecnie 4 kbajtów w układzie /NMOS/.

Dynamiczne pamięci RAM

W dynamicznych pamięciach RAM elementami pamięciowymi są wejściowe pojemności tranzystorów MOS, a nie elementy przerzutnikowe. Z tego powodu dynamiczne pamięci RAM mogą mieć układy o większej pojemności pamięciowej. Jednak pojemności tranzystorów MOS są niedoskonałe, ładunek elektryczny w nich zawarty z biegiem czasu zanika i musi być okresowo odświeżany za pomocą specjalnego sygnału taktującego. Poza tym należy zapewnić specjalne poziomy napięcia i prądu. Dlatego też dynamiczne pamięci RAM są nieco skomplikowane w użytkowaniu. Nie można ich łączyć z mikrokomputerem bezpośrednio, lecz przez specjalne układy zmiany poziomu /TTL-to-MOS Level Shifter/ i układy sterujące /High Voltage Clock Driver/, stanowiące wzmacniacze prądu*).

W odróżnieniu od statycznych, dynamiczne pamięci RAM pobierają energię tylko w trakcie zapisu, odczytywania i odświeżania.

Przy wyłączaniu zasilania zawartość wszystkich komórek dynamicznej pamięci RAM wynosi zero.

Maksymalna pojemność dynamicznych pamięci RAM wynosi obecnie 16 kbajtów na układ /NMOS/.

W ostatnim czasie dla osiągnięcia wielkiej gęstości upakowania stosuje się techniki Double-Diffused MOS i VMOS /V-shaped MOS/.

ROM

Pamięci ROM również są produkowane w obu technologiach i noszą oznaczenie technologii, w której są wykonane. Maksymalna pojemność pamięci ROM wynosi obecnie 64 kbajtów na układ /NMOS/.

Pamięci EAROM produkowane są w technice MNOS /Metal-Nitride-Oxide-Semiconductor/. Maksymalna pojemność wynosi nieco ponad 4 kbajty.

---

*) Dane aktualne w czasie pisania książki, obecnie uległy dezaktualizacji.

JEDNOSTKI PERYFERYJNE

Najczęściej prostsze jednostki peryferyjne produkowane są w technologii bipolarnej, zaś bardziej skomplikowane w technologii MOS. Trzeba jednak mieć na uwadze, że na wybór technologii nie wpływa wyłącznie stopień skomplikowania jednostki peryferyjnej.

Dalszy rozwój jednostek peryferyjnych zmierza w kierunku tworzenia bardziej złożonych, uniwersalnych i programowalnych jednostek peryferyjnych /Inteligent Peripherals/. Taką jednostką jest Intel 8041/8741. Intel 8741 zawiera w sobie pamięć EPROM, natomiast Intel 8041 zawiera pamięć MPROM. Układy te przeznaczone są do projektowania inteligentnych terminali /Inteligent Terminals/.

LITERATURA

C41, C45, D1, D6, E1, E2, E3, F1, F2, F3, F4, F5, G3, G4, K1, K2.

# Cz.II. POJECIA PODSTAWOWE

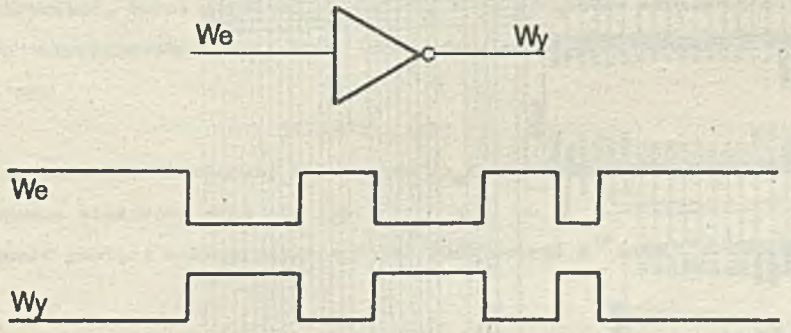
- INWERTER
- MULTIWIBRATOR MONOSTABILNY

## INWERTER

Podstawowe pojęcia związane z mikrokomputerem zostaną wprowadzone za pomocą dwóch prostych przykładów. Pierwszym z nich jest inwerter.

Oczywiście nie ma sensu realizacja zwykłego inwertera / 1/6 układu scalonego 7404/ za pomocą mikrokomputera. Jednak sprzętowa struktura mikrokomputera, służącego jako inwerter nie różni się w sposób istotny od struktury sprzętowej mikrokomputera służącego za filtr cyfrowy, modem do transmisji danych lub procesor komunikacyjny. Istotnie różnią się tylko programy. W wypadku inwertera program składa się tylko z kilku rozkazów. W wypadku wymienionych urządzeń profesjonalnych program składa się z kilkuset rozkazów.

Mając wszystko to na uwadze oraz mając na celu najbardziej przystępny sposób wykładu w tej części książki nie porusza się zagadnień telekomunikacyjnych. Będą one przedmiotem trzeciej części. Obecnie szczegółowo zostanie zanalizowana tylko struktura sprzętowa mikrokomputera opartego na mikroprocesorze Intel 8080. Później, gdy w trzeciej części będzie wprowadzony konkretny problem z dziedziny telekomunikacji nie będzie już potrzeby ponownie analizować sprzętu mikrokomputera. Wskazane będą tylko zmiany wymagane przez konkretny problem. Całą uwagę zwrócimy na problem telekomunikacyjny oraz program, za pomocą którego realizowano są funkcje danego urządzenia telekomunikacyjnego. Na rys. 8 podany jest symbol oznaczenia inwertera oraz jego przebiegi czasowe.

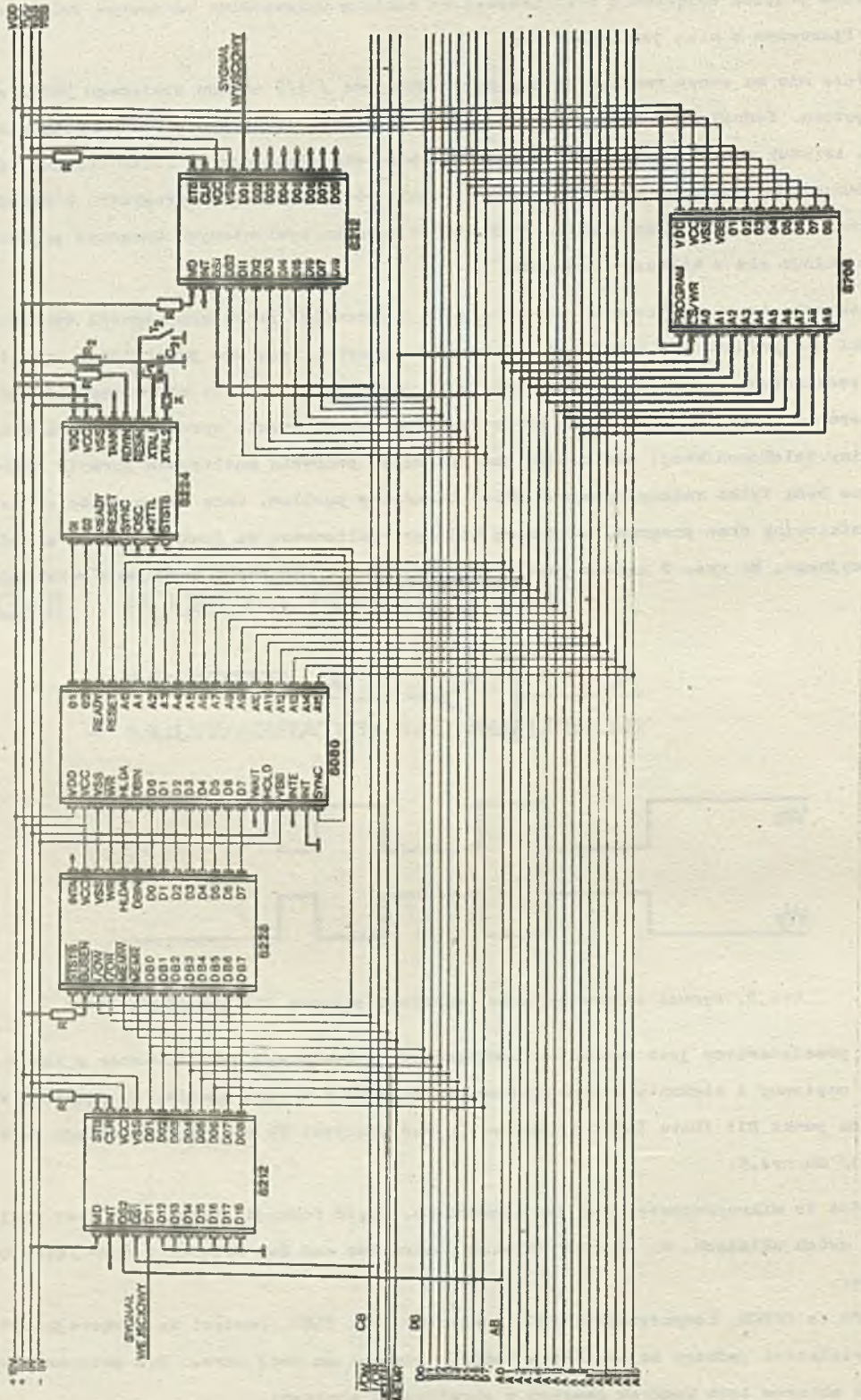



Rys.8. Symbol inwertera oraz przebiegi czasowe dla inwertera

Na rys.9 przedstawiony jest schemat mikrokomputera pracującego jako inwerter z rys.8. Program jest tak napisany i elementy mikrokomputera są połączone w taki sposób, że punktowi We na rys.8 odpowiada punkt DI1 /Data Input 1/ na rys.9, zaś punktowi Wy na rys.8 odpowiada punkt DO1 /Data Output 1/ na rys.9.

Układ 8080A to mikroprocesor. Jak już wspomniano, część funkcji sterujących jest realizowana jeszcze na dwóch układach. Są to 8228 /System Controller and Bus Driver/ i 8224 /Clok Generator and Driver/.

Układ 8708 to EPROM. Kompatybilny MPROM oznaczony jest 8308. Pamięci te zawierają 1024 komórki, każda wielkości jednego bajta. Każda komórka pamięci ma swój adres. Dla programisty pamięć ROM jest tylko zbiorem 1024 komórek pamięci z określonymi adresami.




Rys.9. Schemat mikrokomputera pracującego jako inwerter

Pamięć RAM w przykładzie na rys.9 nie jest potrzebna.

Układy 8212 są jednostkami wejścia/wyjścia. Sygnał wprowadzony jest do mikrokomputera przez górny lewy układ 8212 /końcówka DI1/, a wyprowadzony z mikrokomputera przez górny prawy układ 8212 /końcówka DO1/. Układ 8212 zawiera jednobajtowy rejestr buforowy. Każda komórka peryferyjna ma swój adres. Dla programisty cały układ 8212 jest tylko jedną komórką peryferyjną z określonym adresem.

Mikroprocesor jest w istocie bardzo skomplikowany. Jednak dla programisty mikroprocesor stanowi tylko zbiór 10 komórek. Komórki te nazywają się rejestrami. Rejestry nie mają adresów lecz nazwy.

SZYNA

Połączenia elementów mikrokomputera oznaczane są jako szyna /Bus/. W mikrokomputerze opartym na mikroprocesorze Intel 8080 istnieją trzy szyny: szyna adresowa /AB/, szyna danych /DB/ oraz szyna sterująca /CB/.

Szyna adresowa

Przez szynę adresową przekazywane są adresy komórek pamięci i peryferyjnych, do których zwraca się mikroprocesor. Szyna adresowa składa się z 16 przewodów oznaczonych symbolami od A0 do A15. W wypadku mikroprocesora Intel 8080 jeden adres pamięci składa się z szesnastocyfrowej liczby binarnej, np.

1011.0110.0000.1101

jest takim adresem pamięci. Najmłodszy bit adresu /$b_0$/, tzn. pierwszy z prawej, przekazywany jest przez połączenie oznaczone jako A0, itp.
Maksymalna pojemność pamięci mikroprocesora Intel 8080 wynosi $2^{16}$= 65.536 ośmiobitowych komórek pamięci.

W mikroprocesorze Intel 8080 jeden adres peryferyjny składa się z ośmiocyfrowej liczby binarnej. Np.

1111.0010

jest takim adresem peryferyjnym. Najmłodyza bit adresu /$b_0$/ przekazywany jest przez przewód A0 bądź A8, zaś najstarszy - przez przewód A7 bądź A15. Należy zauważyć, że adres komórki peryferyjnej pojawia się zarówno na dolnej, jak i na górnej połowie szyny adresowej. Pojemność pamięci mikrokomputera opartego na mikroprocesorze Intel 8080 wynosi $2^8$= 256 komórek peryferyjnych.

Heksadecymalny system liczbowy

Wyrażanie adresów, danych i zakodowanych rozkazów w dwójkowym systemie liczbowym jest niewygodne, gdyż liczby są wtedy rozciągnięte. Dlatego też cztery cyfry binarne łączy się w jedną cyfrę heksadecymalną /tab.1/. Pierwsza kolumna dotyczy dziesiętnego systemu liczbowego, druga binarnego, natomiast trzecia heksadecymalnego.

Tab.1.

| D | B | H |
|---|---|---|
| 0 | 0000 | 0 |
| 1 | 0001 | 1 |
| 2 | 0010 | 2 |
| 3 | 0011 | 3 |
| 4 | 0100 | 4 |
| 5 | 0101 | 5 |
| 6 | 0110 | 6 |
| 7 | 0111 | 7 |
| 8 | 1000 | 8 |
| 9 | 1001 | 9 |
| 10 | 1010 | A |
| 11 | 1011 | B |
| 12 | 1100 | C |
| 13 | 1101 | D |
| 14 | 1110 | E |
| 15 | 1111 | F |

Przedstawiony poprzednio adres pamięci może być obecnie zapisany jako B60D, zaś peryferyjny jako F2.

SZYNA DANYCH

Szyna danych ma 8 przewodów oznaczonych symbolami D0 i D7. Przez tę szynę przekazywane są zakodowane rozkazy z pamięci ROM do mikroprocesora, a także dane z komórek pamięci i peryferyjnych do mikroprocesora i odwrotnie. Z rysunku 9 wynika, że układ 8080 nie jest bezpośrednio połączony z szyną danych, lecz przez układ 8228. Tak więc część szyny danych znajduje się między układami 8080 i 8228. Przez tę część szyny danych przenoszone są również pewne sygnały sterujące. Dlatego mówi się, że szyna danych jest multipleksowana czasowo.

SZYNA STERUJĄCA

Szyna sterująca ma 4 połączenia oznaczone symbolami $\overline{\text{MEMR}}$, $\overline{\text{MEMW}}$, $\overline{\text{I/OR}}$ i $\overline{\text{I/OW}}$.

Gdy mikroprocesor zwraca się do pamięci pragnąc odczytać zawartość określonej komórki pamięci, na połączeniu $\overline{\text{MEMR}}$ /Memory Read/ pojawia się niski poziom sygnału. W ten sposób pamięć jest wybierana. Gdy mikroprocesor zwraca się do jednostki peryferyjnej w celu odczytania treści komórki peryferyjnej, na $\overline{\text{I/OR}}$ /Input/Output Read/ pojawia się niski poziom sygnału. W ten sposób dokonywany jest wybór jednostki peryferyjnej.

Podobnie na połączeniach $\overline{\text{MEMW}}$ /Memory Write/ i $\overline{\text{I/OW}}$ /Intput/Output Write/ pojawia się niski poziom sygnału przy wpisywaniu danych do komórki pamięci bądź peryferyjnej.

INTERPRETACJA DANYCH

Dane, o których mowa mogą być interpretowane na kilka sposobów. Jak będzie dana interpretowana zależy od programisty. W poniższym tekście jest przedstawione jak programista może interpretować jedną daną. Np. dana

11001001

może przedstawiać zbiór 8 niezależnych poziomów sygnałów /przykład przedstawiony na rysunku 9/.

Liczby arytmetyczne

Ta sama dana może stanowić binarny równoważnik dodatniej liczby dziesiętnej między 0 a 255. W mikrokomputerze stosuje się kod dwójkowy, wobec czego dana ta może być rozumiana jako liczba 201.

Liczby algebraiczne

Ta sama dana może stanowić binarny równoważnik algebraicznej liczby dziesiętnej w zakresie między 128 a + 127. W mikrokomputerach dla przedstawienia takich liczb stosuje się uzupełnienia dwójkowe kodu naturalnego /Two's Complement/.
Uzupełnienie liczby dodatniej między 0 a 127 jest identyczne z naturalnym kodem tej liczby. Np. liczba + 126 przedstawiana jest jako

01111110

Uzupełnienie liczby ujemnej między 0 a - 128 tworzone jest w następujący sposób.

1. Moduł danej liczby ujemnej przedstawia się w kodzie naturalnym, np. mamy zakodować liczbę -55. Modułem tej liczby jest 55, zaś binarnym równoważnikiem l-iczby 55 w kodzie naturalnym jest

   00110111

2. Znajduje się kod odwrotny /One's Complement/, tzn. zwykłe uzupełnienie jedynkowe ostatniej liczby dwójkowej. Wynikiem tej operacji jest

   11001000

3. Dodaje się binarnie zakodowaną liczbę 1. Po tej operacji otrzymuje się

   11001001

Oznacza to, że powyższą daną jest liczba - 55 przedstawiona w postaci kodu z uzupełnieniem dwójkowym.

Przy przedstawieniu liczb algebraicznych w postaci uzupełnienia dwójkowego najstarszym bitem przy liczbach dodatnich jest 0, zaś nastarszym bitem przy liczbach ujemnych jest 1. Bit 1 na miejscu najstarszej pozycji bitowej interpretuje się jako liczbę - 128. Pozostałe 7 bitów interpretuje się jako pewną binarnie zakodowaną liczbę dodatnią między 0 a 127. Ostateczną dziesiętną wartość liczby ujemnej otrzymuje się przez dodanie - 128 i dziesiętnego równoważnika pozostałych

bitów. Np. liczbę - 1 można przedstawić jako wyrażenie

$$-1 = -128 + 127$$

Liczba + 127 zakodowana z siedmioma cyframi dwójkowymi wynosi:

1111111

Gdy na ósme miejsce binarne dodamy - 128 łączna wartość wynosić będzie - 1:

1.1111111

Tworzenie binarnie zakodowanej wartości modułu liczby ujemnej wyrażonej w kodzie z uzupełnieniem dwójkowym dokonywane jest w następujący sposób:

1. znajduje się postać zanegowaną modułu danej liczby,
2. dodaje się binarnie zakodowaną liczbę 1.

Zalety takiego sposobu przedstawienia liczb algebraicznych są wielostronne.
Przy dodawaniu dwóch liczb algebraicznych przedstawionych w powyższy sposób wynik będzie również liczbą algebraiczną przedstawioną w podobny sposób. Przedstawiamy to na przykładzie liczb

$$+ 72D = 01001000B$$
$$- 55D = 11001001B$$

Litera D oznacza dziesiętny system liczbowy, litera B binarny system liczbowy, zaś litera H dotyczy heksadecymalnego systemu liczbowego. Po dodaniu otrzymamy:

| | | |
|---|---|---|
| - 55 | 01001000 | pierwszy składnik |
| + 72 | 11001001 | drugi składnik |
| + 17 | 00010001 | wynik |
| | 11 1 | przeniesienie |

Dziesiętnym równoważnikiem wyniku dodawania binarnego jest również + 17, a więc wynik jest poprawny.

Przeniesienie z pozycji bitowej $b_3$ do pozycji bitowej $b_4$ nazywa się przeniesieniem pomocniczym lub półprzeniesieniem /Auxiliary Carry, Intermediate Carry, Half Carry/, zaś przeniesienie pozycji bitowej $b_7$ nazywa się przeniesieniem końcowym /Final Carry/ lub tylko przeniesieniem /Carry/. Półprzeniesienie oznacza się jako AC, zaś przeniesienie końcowe jako C.

O mnożeniu liczb algebraicznych przedstawianych w postaci kodu z uzupełnieniem dwójkowym będzie mowa później.

Liczby BCD

Daną

10001001

można rozpatrywać również jako zakodowaną dwójkowo liczbę dziesiętną 89*).

Z 8 bitów może powstać 128 różnych kombinacji, jednak mogą one przedstawiać tylko 100 binarnie

*) Poszczególne czwórki bitów reprezentują tu binarnie kolejne cyfry dziesiętne /dop.red./.

zakodowanych liczb dziesiętnych. Dlatego też niektóre kombinacje nie mają sensu. Np. dana

11001001

nie nie oznacza przy tym sposobie przedstawienia dodatnich liczb dziesiętnych. Takie przedstawienie liczb jest wygodne, jeżeli wynik operacji w mikrokomputerze wyprowadza się w postaci cyfrowej np. na monitor ekranowy.

Kod ASCII

Często wyniki operacji w mikrokomputerze są drukowane. Gdy na wejściu na drukarkę pojawi się określona dana, drukarka wydrukuje określony znak alfanumeryczny.

Znaki alfanumeryczne można kodować różnymi sposobami. W drukarkach najczęściej jest stosowany kod ASCII /American Standard Code for Information Interchange/. Na końcu książki załączono kod ASCII wszystkich znaków alfanumerycznych stosowanych w języku angielskim. Zgodnie z załącznikiem znajdującym się na końcu książki drukarka wydrukuje znak %, jeżeli na jej wejściu pojawi się bajt

00100101

Również drukarka wydrukuje dużą literę M, gdy na jej wejściu pojawi się bajt

01001101

W załączniku umieszczono tablicę kodu ASCII z 7 oraz z 8 bitami. Gdy w minikomputerze opartym na mikrokomputerze znajduje się drukarka, wówczas stosuje się rozszerzony kod 8-bitowy BCD. Gdy drukarka jest oddalona, tzn. gdy znaki alfanumeryczne przesyłane są na odległość, wówczas stosuje się kod ASCII 7-bitowy. Ósmy bit służy wówczas do kontroli parzystości. W ramach kodu ASCII z 7 bitami znajdują się również kody znaków sterujących /Control Characters/ stosowane w sieciach komputerowych do przesyłania danych.

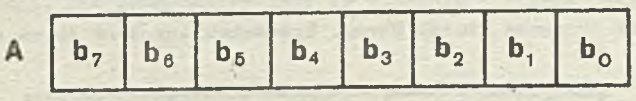
Z załącznika wynika, że niektóre kody nic nie oznaczają. Takim kodem jest np.

11001001

Niektóre drukarki stosują kod EBCDIC /Extended Binary Coded Decimal Interchange Code/.

AKUMULATOR

Główny rejestr mikroprocesora nazywa się akumulatorem pierwotnym lub po prostu akumulatorem. Oznacza się go symbolem A. W mikroprocesorach ośmiobitowych ma on pojemność 8 bitów. Schemat blokowy akumulatora jest przedstawiony na rys.10.




Rys.10. Schemat blokowy akumulatora

Stosowanie mikrokomputera często powoduje, że dana z zewnątrz, tzn. z określonej komórki peryferyjnej ma być przesłana do określonej komórki i odwrotnie. Wtedy daną przesyła się z komórki peryferyjnej do akumulatora, a następnie z akumulatora do komórki pamięci lub odwrotnie. To samo dotyczy przesyłania danych między dwoma komórkami pamięci lub peryferyjnymi. Przesyłanie z komó-

rok pamięci lub z innych rejestrów w mikroprocesorze do akumulatora oznacza się odpowiednio jako LOAD i MOVE. Przesyłanie z komórek peryferyjnych do akumulatora oznacza się jako IN i MOVE. Przesyłanie z akumulatora do komórek pamięci i innych rejestrów mikroprocesora oznacza się odpowiednio jako STORE i MOVE. Przesyłanie z akumulatora do komórek peryferyjnych oznacza się jako OUT i MOVE. Tak więc akumulator służy jako pośrednik przy przenoszeniu danych przez mikrokomputer. Jednak w akumulatorze dokonywane są również różne operacje logiczne i arytmetyczne, tzn. dokonywana jest obróbka danych.

REJESTRY

Poza akumulatorem pierwotnym mikroprocesor Intel 8080 zawiera jeszcze 6 akumulatorów wtórnych lub prościej - rejestrów. Ich pojemność wynosi również 8 bitów. Oznaczone są jako B, C, D, E, H i L. Przesyłanie danych między akumulatorem a tymi rejestrami trwa krócej niż przesyłanie danych między akumulatorem a pamięcią. Dlatego rejestry te są wygodne do tymczasowego zapamiętywania często stosowanych danych.

W mikroprocesorze Intel 8080 możliwe jest przesyłanie danych między pamięcią a rejestrami wtórnymi, jak również między samymi rejestrami wtórnymi. Takie przesyłanie oznacza się jako MOVE.

Rejestry B i C mogą być traktowane również jako jeden rejestr 16-bitowy, tzn. parę rejestrów BC. To samo dotyczy 16-bitowych rejestrów DE i HL. W rejestrach C, E, L znajduje się młodszy bajt liczby 16-bitowej. W rejestrach B, D i H znajduje się starszy bajt. Rejestry mają wiele innych funkcji, o czym będzie mowa później.

REJESTR STANU

Operacje logiczne i arytmetyczne wymagają zazwyczaj dwóch operandów. Pod terminem operand rozumie się wielkość, na której dokonuje się określonej operacji. Gdy w mikrokomputerze dokonywana jest operacja logiczna lub arytmetyczna jeden operand znajduje się w akumulatorze, zaś drugi w pewnej komórce pamięci lub w jednym z rejestrów. Wynik operacji zawsze znajduje się w akumulatorze.

Wynikiem operacji logicznej lub arytmetycznej może być 00000000 lub $\phi$. Wynik operacji może mieć parzystą liczbę jedynek lub $\phi$. Najstarszy bit może mieć wartość 1 lub $\phi$. Po operacjach arytmetycznych półprzeniesienie i przeniesienie końcowe mogą mieć wartość 1 lub $\phi$.

Rejestr, którego zawartość odzwierciedla wynik operacji w akumulatorze nazywa się rejestrem stanu i oznacza się jako PSW /Program Status Word/. Przedstawiony jest on na rys.11.


Rys.11. Rejestr stanu

Zawartość rejestru PSW składa się z 5 wskaźników /Status, Flag, Status Flag/.

Jeżeli wynik operacji jest zerem, wówczas wskaźnik Z /zero/ jest ustawiony w stanie 1. W innym razie jest zerowany.

Jeżeli wynik zawiera parzystą liczbę jedynek wskaźnik P /Parity/ jest ustawiony w stanie 1. W innym wypadku jest zerowany.

Jeżeli najstarszy bit wyniku ma wartość 1, wskaźnik S /Sign/ jest ustawiony w stanie 1. W innym wypadku jest zerowany.

Jeżeli przeniesienie ma wartość 1 wskaźnik C /Carry/ jest ustawiany w stanie 1. W innym wypadku jest zerowany.

Jeżeli półprzeniesienie ma wartość 1 wskaźnik AC /Auxiliary Carry/ jest ustawiany w stanie 1. W innym wypadku jest zerowany.

Bitów numer 1, 3 i 5 w rejestrze PSW nie używa się i oznaczone są one jako X.

Wskaźniki odgrywają wielką rolę w programowaniu. Jednak trzeba być ostrożnym w pracy z nimi, ponieważ istnieją rozkazy logiczne i arytmetyczne, po wykonaniu których zawartość rejestru statusowego nie odzwierciedla stanu w akumulatorze.

Tak więc sam wynik znajduje się w akumulatorze, zaś w rejestrze stanu znajdują się pewne parametry tego wyniku.

REJESTR PROGRAMU /LICZNIK ROZKAZÓW/

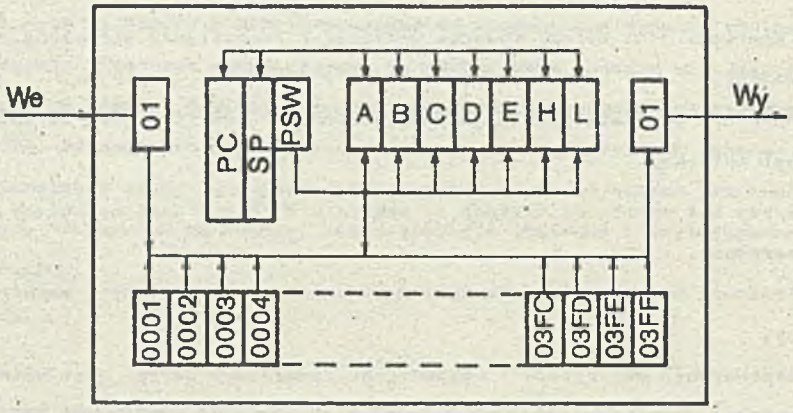
Rejestr programu /Program Counter/ oznaczany jest jako PC i ma pojemność 16 bitów. Zawiera on adres bieżąco wykonywanego rozkazu. Po wykonaniu rozkazu zawartość tego rejestru automatycznie zwiększa się o jeden. W ten sposób dokonuje się skoku do nowego rozkazu. Dlatego też często nazywa się ten rejestr licznikiem rozkazów.

REJESTR WSKAŹNIKA STOSU

Rejestr wskaźnika stosu /Stack Pointer/ oznaczany jest jako SP i ma pojemność 16 bitów. Będziemy się nim zajmować później.

PODSUMOWANIE

Mając na uwadze wszystko to, co przedstawiono dotychczas, mikrokomputer z rys.9 może być przedstawiony jak na rys.12, tzn. jako zbiór rejestrów lub komórek peryferyjnych i komórek pamięci z określonymi adresami. Dla programisty jest to zupełnie wystarczające.




Rys.12. Schemat blokowy mikrokomputera z uwidocznieniem jego budowy, składającej się z rejestrów oraz z komórek peryferyjnych i komórek pamięci.

WYKONYWANIE ROZKAZÓW

Przekazywanie danych przez mikrokomputer będzie przedstawione na przykładzie pobrania zakodowanego rozkazu z pamięci ROM do mikroprocesora /Instruction Fetch/.

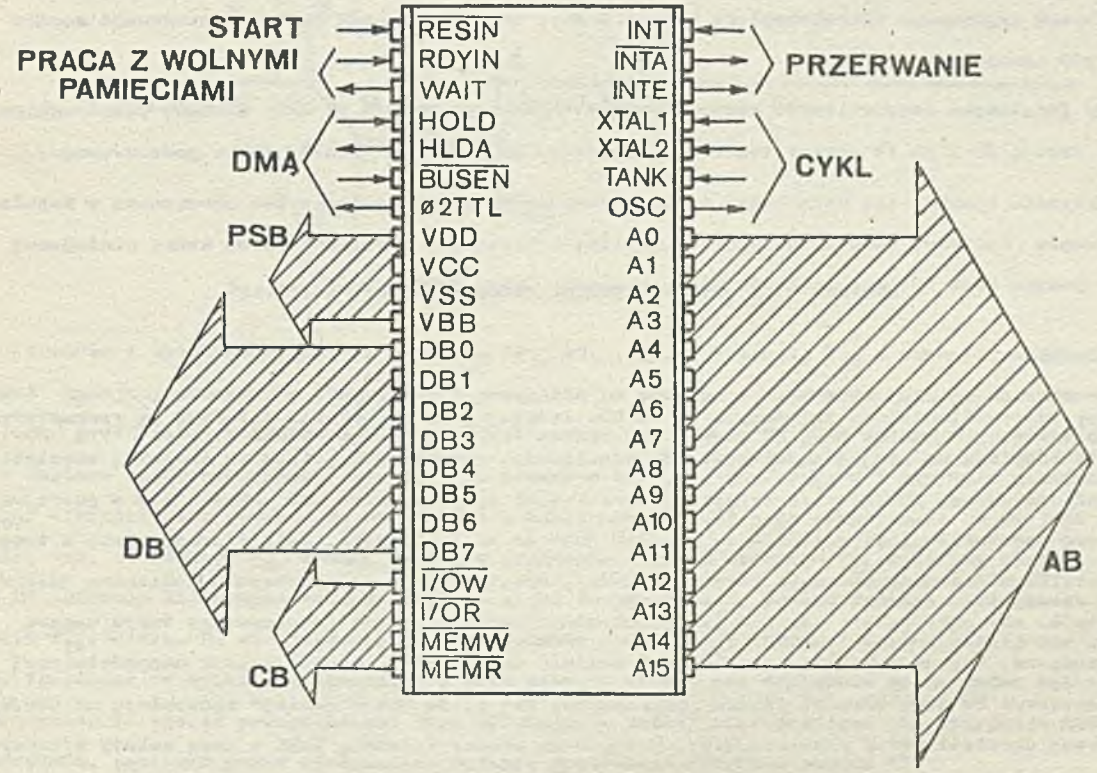
Załóżmy, że w komórce pamięci o adresie 03FBH znajduje się rozkaz C9H. W pewnym momencie, w trakcie wykonywania programu w rejestrze programu utworzona zostanie zawartość 03FBH. Liczba ta następnie "wychodzi" na szynę adresową, natomiast na połączeniu $\overline{MEMR}$ poziom spada do zera. W ten sposób dokonany zostaje wybór pamięci ROM. Zaraz potem zawartość komórki pamięci 03FBH, tzn. liczba C9H "wchodzi" na szynę danych i stąd dostaje się do mikroprocesora. Pobieranie rozkazu z pamięci ROM do mikroprocesora jest pierwszą częścią każdego rozkazu. Drugą częścią każdego rozkazu jest wykonanie operacji wyspecyfikowanej przez rozkaz /Instruction Execution/. Np. tworzy się uzupełnienie zawartości akumulatora, dana z wejściowej komórki pamięci jest umieszczona w akumulatorze itp.

ŁĄCZENIE UKŁADÓW MIKROKOMPUTERA

Połączenia na rys.9. są oznaczone liniami cienkimi i grubymi. Pierwsze z nich specyfikuje producent i muszą one być wykonane zgodnie z rysunkiem. Natomiast przy wykonaniu drugich, użytkownik ma określoną swobodę. Jak widać z rys.9 połączeń drugiego rodzaju jest nieporównywalnie mniej.

Mikroprocesor

Główne układy /8080A, 8228, 8224/ mają razem 84 końcówki, natomiast w stosunku do reszty mikrokomputera zachowują się jak jeden układ z 46 końcówkami. Taki układ przedstawiony jest na rys. 13. Pozostałych 38 końcówek służy do wzajemnych połączeń między układami 8080A, 8228 i 8224. Są to połączenia wyspecyfikowane przez producenta w jego katalogu sprzętu otrzymywanym razem z układami /Hardware Manual/. Przy wykonywaniu tych połączeń użytkownik nie ma swobody.




Rys.13. Mikroprocesor Intel 8080, przy założeniu, że jest zrealizowany na jednym układzie.

Użytkownik nie ma również swobody w zakresie połączeń mikroprocesora z rys.13 z szynami. Końcówki A0 do A15 powinny być wprowadzone na szynę adresową, końcówki DB0 do DB7 na szynę danych, końcówki $\overline{\text{MEMR}}$, $\overline{\text{MEMW}}$, $\overline{\text{I/OR}}$ i $\overline{\text{I/OW}}$ powinny być wprowadzone na szynę sterującą, natomiast końcówki VDD, VCC, VSS i VBB na tzw. szynę zasilającą /Power Supply Bus, PSB/.

Mikroprocesor Intel 8080 wymaga trzech źródeł zasilania: - 5 V, + 5 V i + 12 V z tolerancją mniejszą niż $\pm$ 5%. Dążeniem wszystkich producentów mikroprocesorów jest zrealizowanie mikroprocesora wymagającego tylko jednego źródła zasilania. Tak więc np. Intel 8085 potrzebuje tylko jednego źródła zasilania + 5 V. Przy łączeniu bądź pozostawieniu niepołączonych pozostałych 14 końcówek z rys.13 użytkownik ma określoną dowolność.

XTAL1, XTAL2

Między końcówkami XTAL1 i XTAL2 /eXternal crysTAL/ łączy się rezonator krystaliczny X. Podstawowa częstotliwość rezonatorów może wynosić 4,5 MHz do 18,432 MHz. Producenci rezonatorów oferują rezonatory każdej częstotliwości podstawowej, tzn. przykrawają rezonator według życzeń kupującego. Jeżeli podstawowa częstotliwość rezonatora wynosi ponad 10 MHz, należy w szereg z rezonatorem dodać kondensator $C_1$ o pojemności od 3 pF do 10 pF.

Podstawowa częstotliwość rezonatora krystalicznego dzielona jest przez 9 w układzie 8224. W ten sposób otrzymuje się podstawowy przebieg zegarowy mikrokomputera /Clock/. Np. jeżeli pod-

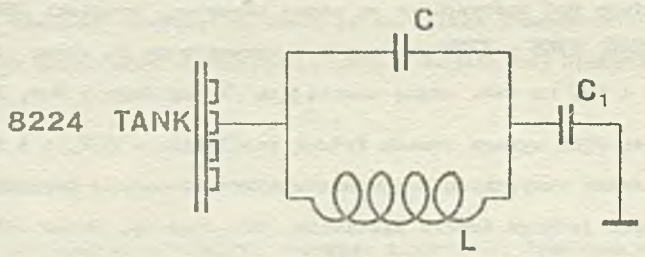
stawowa częstotliwość rezonatora krystalicznego wynosi 18 MHz, wówczas częstotliwość podstawowego przebiegu zegarowego mikrokomputera wynosi 2 MHz, natomiast okres taktu podstawowego zegara wynosi 500 nanosekund.

Gdy podstawowa częstotliwość rezonatora krystalicznego wynosi 18 MHz, wówczas poszczególne rozkazy trwają od 2 µs /4 okresy taktu podstawowego/ do 9 µs /18 okresów taktu podstawowego/.

Dokładnie wiadomo ile trwa każdy rozkaz. Jest to wyspecyfikowane przez producenta w katalogu programów /Software Manual/ otrzymywanym razem z układami. W załączniku na końcu niniejszej pracy podano czas trwania rozkazów mikroprocesora Intel 8080.

TANK

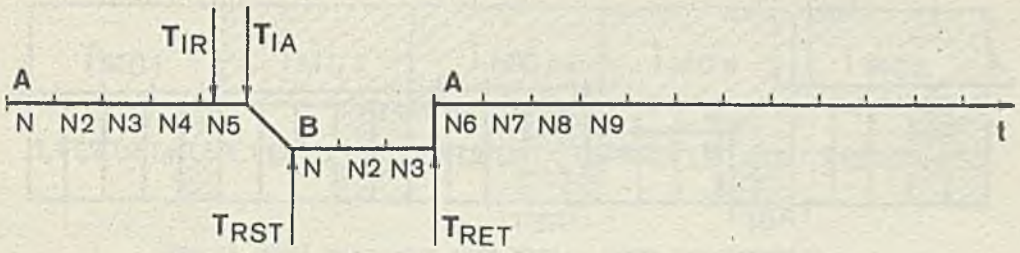
Przy częstotliwościach zbliżonych do 18 MHz łatwiej jest produkować i tańsze są rezonatory, które nie oscylują na swojej podstawowej częstotliwości rezonansowej, lecz na trzeciej częstotliwości nadpodstawowej /Overtone/. Trzecia częstotliwość nadpodstawowa nie pokrywa się z potrójną podstawową częstotliwością rezonansową, jednak jest do niej zbliżona. Jeżeli pracuje się z trzecią częstotliwością nadpodstawową /Overtone Mode/, częstotliwość nadpodstawowa dobierana jest w taki sposób, aby pokryła się ona z dziesięciokrotną częstotliwością podstawowego taktu zegara mikroprocesora. Aby jednak kwarc zaoscylował właśnie na trzeciej częstotliwości nadpodstawowej należy stworzyć ku temu warunki /kwarc pozostawiony sam sobie "najchętniej" zaoscyluje na swojej podstawowej częstotliwości rezonansowej/. W tym celu między końcówką TANK a masą należy stworzyć obwód oscylatorowy dostrojony do trzeciej częstotliwości nadpodstawowej. Kondensator $C_1$ służy do dokładnej regulacji. Przy stosowaniu kwarcu oscylującego na swojej podstawowej częstotliwości, końcówka TANK pozostaje "wisząca".




Rys.14. Sposób połączenia końcówki TANK w wypadku stosowania trzeciej częstotliwości nadpodstawowej oscylatora krystalicznego

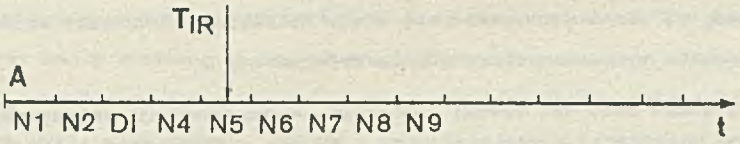
INTE, INT, INTA

Każdy mikroprocesor ma możliwość przerwań tj. wstrzymania na zewnętrzne żądanie /Interrupted Request/, wykonywania bieżącego programu /Interrupted Program/, i przejścia do początku innego programu /Interrupt Subroutine/, wykonania go i powrotu do programu, którego wykonanie zostało przerwane. Działanie takie nazywa się obsługą przerwania /Interrupt Service//rys.15/.




Rys.15. Przyjęcie żądania przerwania oraz jego obsługa

Program A wykonuje rozkaz po rozkazie /N1, N2, .../. W momencie $T_{IR}$ z zewnątrz nadchodzi sygnał żądający przerwania programu A i przejścia na program B. Jednakże żądanie to nadchodzi w środku przedziału, w którym wykonywany jest rozkaz N5. Rozkaz N5 jest wpierw wykonywany do końca i dopiero wtedy wykonywany jest skok do programu B /$T_{IA}$/. Gdy program B zostanie zakończony /$T_{RET}$/ mikroprocesor powraca do programu A i kontynuowane jest jego wykonywanie od miejsca zatrzymania, tzn. od rozkazu $N_6$. Jednak jeżeli w programie A przed momentem $T_{IR}$ wykonany zostanie rozkaz DI /Disable Interrupts/ nie dojdzie do skoku do programu B, pomimo żądania otrzymanego w momencie $T_{IA}$. Rozkaz DI nie powoduje ani przesyłania danych, ani operacji arytmetycznej czy logicznej. Wprowadza on tylko mikroprocesor w taki stan, w którym nie odpowiada on na żadne żądanie przerwania. Na rys.16 przedstawiona jest sytuacja, w której mikroprocesor nie przyjmuje żądania przerwania, ponieważ przed otrzymaniem żądania przerwania wykonano rozkaz DI.




Rys.16. Sytuacja, w której nie przyjmowane jest żądanie przerwania

Jednak, jeżeli w programie A przed momentem $T_{IR}$ wykonano rozkaz EI /Enable Interrupts/ to nastąpi skok do programu B, tak jak na rys.15.

Rozkaz EI wprowadza mikroprocesor w taki stan, w którym odpowiada on na żądanie przerwania.

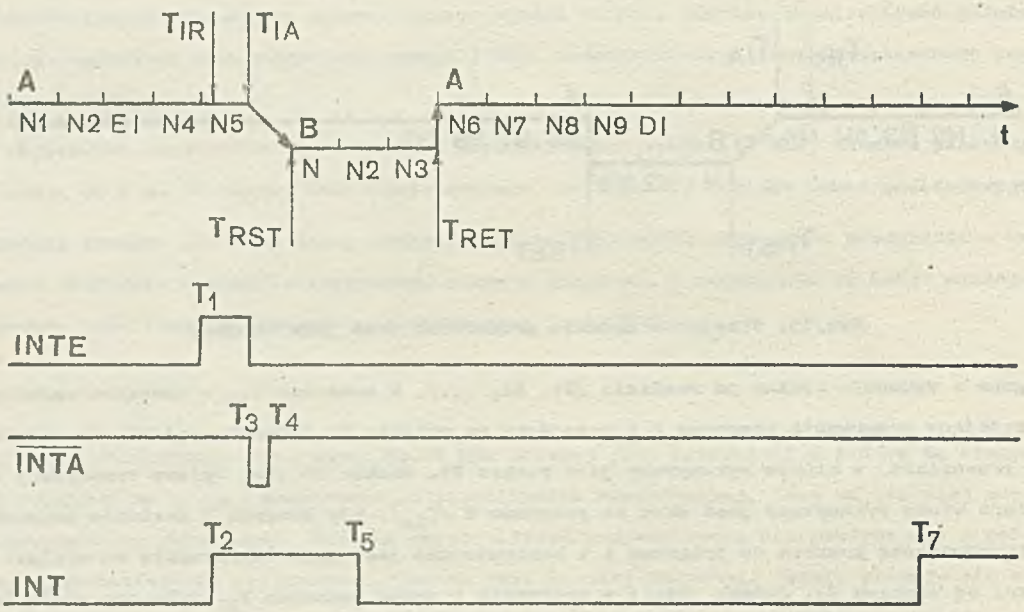
Końcówki INTE, INT i $\overline{\text{INTA}}$ wykorzystuje się przy obsłudze przerwania. Sygnały na tych końcówkach przedstawione są na rys.17.

Po przyjęciu przerwania na wyjściu INTE automatycznie tworzy się niski poziom sygnału, jak na rys.17. Jest to znak dla sieci logicznej poza mikrokomputerem /External Logic/, tzn. na zewnątrz, że mikroprocesor nie odpowiada na pierwsze żądanie przerwania, które nadejdzie.

Żądanie skoku do programu B dokonuje się przez utworzenie wysokiego poziomu sygnału 1 lub $\phi$ na wejściu INT /INTerrupt request/.

Na rys.17 są to momenty $T_2$ oraz $T_7$.

W momencie, gdy mikroprocesor zareaguje na przerwanie tzn. po wykonaniu do końca rozkazu N5 na rys.17, poziom sygnału na wyjściu $\overline{\text{INTA}}$ /INTerrupt Acknowledged/ spada do zera. Na rys.17 jest




Rys.17. Przebiegi czasowe na końcówkach INTE, $\overline{\text{INTA}}$ i INT w trakcie obsługi przerwania

to moment $T_3$. Niski poziom sygnału na wyjściu $\overline{\text{INTA}}$ utrzyma się tylko przez krótki czas /do momentu T/.

Końcówki układów, z których utworzony jest mikrokomputer mogą być wejściowe, wyjściowe lub wejściowo-wyjściowe. Może się zdarzyć, że jakaś wejściowa lub jakaś wyjściowa końcówka nie jest wykorzystana. Końcówki wyjściowe pozostawia się wtedy "wiszące", natomiast wejściowe łączy się z masą lub przez opornik o oporze od 1k do 10k ze źródłem + 5 V.

Aktywny poziom sygnału może być wysoki lub niski. W dokumentacji załączonej do mikroprocesora Intel 8080 końcówki odpowiadające sygnałom aktywnym o niskim poziomie sygnału mają oznaczoną nazwę.

Jeżeli jakiś sygnał nie jest użytkowany, wtedy najczęściej, ale nie zawsze, na odpowiednią końcówkę należy doprowadzić bierny poziom sygnału. Jednak w wypadku końcówki READY, o której będzie mowa dalej, sytuacja jest odwrotna.

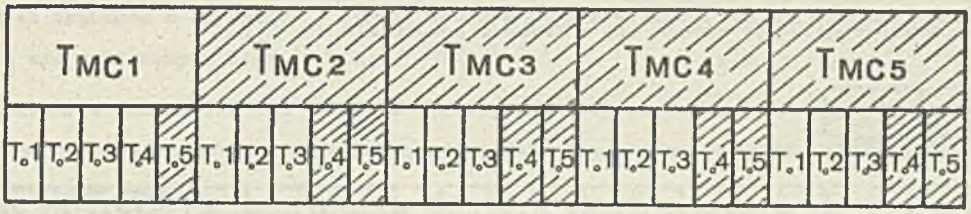
W sytuacji z rys.9 końcówki INT, INTE i $\overline{\text{INTA}}$ nie są wykorzystywane. Dlatego końcówka INT musi być połączona z masą, zaś końcówki INTE i $\overline{\text{INTA}}$ muszą "wisieć".

WAIT, RDYIN

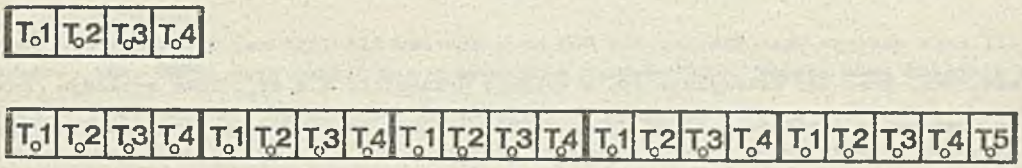
Wyżej stwierdzono, że jeden rozkaz trwa od 4 do 18 okresów taktu podstawowego. Od 3 do 5 okresów taktu podstawowego połączonych jest w jeden cykl maszynowy /Machine Cycle/ jak na rys.18.

Okresy taktu podstawowego $T_o$ oraz cykle maszynowe $T_{MC}$ - zakreślone na rysunku - mogą, ale nie muszą istnieć. Struktura najkrótszego oraz najdłuższego rozkazu przedstawiona jest na rys.19.

W jednym cyklu maszynowym mikroprocesor zwraca się do pamięci tylko raz i to w trakcie pierwszego, drugiego i trzeciego okresu taktu podstawowego /$T_1$, $T_2$, $T_3$/.




Rys.18. Cykl maszynowy oraz okresy taktu podstawowego




Rys.19. Struktura najkrótszego i najdłuższego rozkazu mikroprocesora Intel 8080

Istnieją rozkazy, w których mikroprocesor tylko raz zwraca się do pamięci. Takie rozkazy trwają tylko przez jeden cykl maszynowy czyli przez 4 lub 5 okresów taktu podstawowego i składają się z jednego bajtu. W trakcie trzech pierwszych okresów /$T_1$, $T_2$ i $T_3$/ rozkaz jest przenoszony z pamięci ROM do mikroprocesora /Instruction Fetch/, natychmiast w trakcie czwartego względnie czwartego i piątego okresu rozkaz jest wykonywany. Są to najkrótsze rozkazy.

W najdłuższych rozkazach mikroprocesor nawet pięć razy zwraca się do pamięci. W trakcie trzech pierwszych cykli maszynowych mikroprocesor trzy razy zwraca się do pamięci ROM. W ten sposób wszystkie trzy bajty rozkazu, jeden po drugim, przenoszone są do mikroprocesora. Jednak rozkaz może wymagać, aby teraz mikroprocesor zwrócił się jeszcze do pamięci RAM dwa razy, w trakcie dwóch następnych cykli maszynowych. Jednym z najważniejszych parametrów pamięci jest czas dostępu do komórki pamięci /Access Time/.

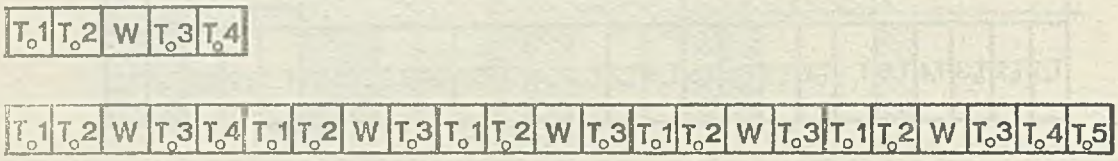
Czas dostępu do komórki pamięci jest definiowany jako czas trwający od momentu, gdy na szynie adresowej pojawia się adres komórki pamięci do momentu, gdy na szynie danych pojawi się treść komórki pamięci.

Jeżeli pracujemy z taktem, którego podstawowy okres wynosi 500 ns oraz z pamięciami, których czas dostępu wynosi 450 ns lub krócej, wówczas nie ma problemów. Takie są pamięci EPROM 8708 /450 ns/, MPROM 8308 /450 ns/ oraz statyczna pamięć RAM 8102A-4 /450 ns/.

Natomiast gdy pracujemy z pamięcią, której czas dostępu jest dłuższy od 450 ns, pamięć nie będzie dostatecznie szybka, aby w przewidzianym czasie /$T_1$, $T_2$ i $T_3$/ odpowiedzieć na żądania stawiane przez mikroprocesor. Dlatego między $T_2$ a $T_3$ należy wstawić jeden lub dwa okresy taktu podstawowego, tzn. mikroprocesor musi wejść w stan oczekiwania /Wait State/.

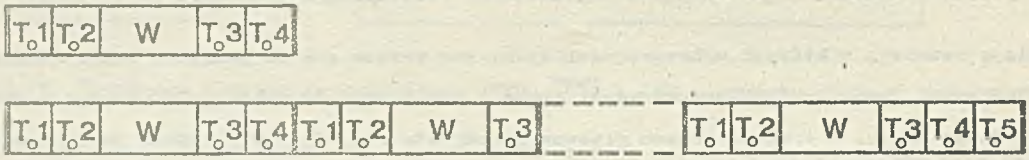
Jeżeli czas dostępu jest dłuższy niż 450 ns, a krótszy niż 900 ns, wstawia się tylko jeden okres oczekiwania /W/, tzn. procesor zatrzymuje się w trakcie przedziału W = $T_o$. Taka sytuacja ma miejsce przy pracy z pamięcią MPROM 8316A /850 ns/ oraz z pamięcią statyczną RAM 8101-2 /850 ns/.

Na rys.20 przedstawione są struktury najkrótszego i najdłuższego rozkazu mikroprocesora Intel 8080, gdy procesor zatrzymuje się w trakcie przedziału W = $T_o$.




Rys.20. Struktura najkrótszego i najdłuższego rozkazu mikroprocesora Intel 8080, gdy procesor zatrzymuje się w trakcie przedziału W = $T_o$

Jeżeli czas dostępu jest dłuższy niż 900 ns a krótszy niż 1350 ns, wprowadza się dwa okresy oczekiwania, tzn. procesor zatrzymuje się w trakcie przedziału W = $2T_o$. Taka sytuacja /rys.21/ ma miejsce przy pracy z pamięcią EPROM 8702A /1300 ns/ oraz ze statyczną pamięcią RAM 8101 /1300 ns/.

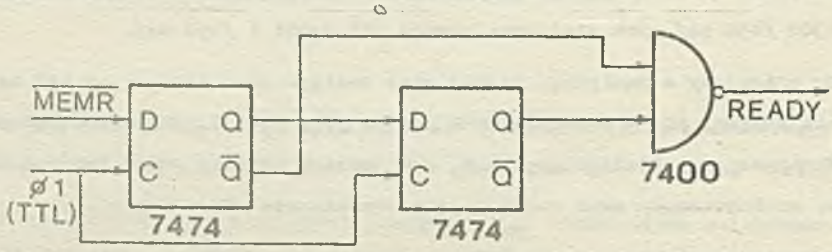



Rys.21. Struktura najdłuższego i najkrótszego rozkazu mikroprocesora Intel 8080, gdy procesor zatrzymuje się w trakcie przedziału W = 2 $T_o$

Takie problemy nie zaistnieją przy pracy z dynamiczną pamięcią RAM oraz z bipolarnymi pamięciami ROM. Można ich uniknąć również przez dobór rezonatora krystalicznego o wystarczająco niskiej częstotliwości. Do pracy z wolnymi pamięciami służą końcówki WAIT i RDYIN.

Przez wejście RDYIN /ReaDY logic INput/ dokonuje się zatrzymania procesora z zewnątrz, w tym również przez pamięć. Przez wyjście WAIT mikroprocesor zawiadamia, że znajduje się w stanie oczekiwania. Dopóki mikroprocesor znajduje się w stanie oczekiwania, wówczas na wyjściu WAIT jest wysoki poziom sygnału.

Jeżeli pragniemy przerzucić procesor do stanu oczekiwania na czas przedziału W = $T_o$, wówczas należy na wejściu RDYIN wygenerować niski poziom sygnału o czasie trwania $T_o$. Jeden z zespołów, za pomocą którego dokonuje się tego przedstawiony jest na rys.22.




Rys.22. Układ, za pomocą którego realizuje się zatrzymanie procesora na czas przedziału W = $T_o$

Ø 1 jest sygnałem o okresie $T_o$. Sygnał ten wyprowadzony jest z układu 8224 na rys.9, jednak jego poziom nie jest kompatybilny z TTL.

W przykładzie przedstawionym na rys.9 zastosowana jest szybka pamięć ROM, wobec czego nie używa się końcówek WAIT i RDYIN. Dlatego końcówka WAIT "wisi", zaś końcówka RDYIN połączona jest z wysokim poziomem sygnału.

Po przejściu procesora w stan oczekiwania efektywny czas trwania rozkazu wydłuża się. Jeżeli wstawiony jest jeden okres oczekiwania, wówczas rozkazy trwają od 5 do 23 okresów taktu podstawowego. Należy pamiętać, że przedłużane są tylko te cykle maszynowe, w ramach których procesor zwraca się do pamięci.

Procesor, tzn. 8080A oraz pamięć nie są sprzężone bezpośrednio. Między nimi znajduje się co najmniej układ 8228. Jednak, jeżeli mamy do czynienia z pamięcią o dużej pojemności wówczas jej realizacja opiera się na większej liczbie układów. Wtedy na szynie danych istnieje wielka liczba rozgałęzień i dlatego należy stosować wzmacniacze dla szyny danych. Ponieważ szyna danych jest dwukierunkowa to i wzmacniacze muszą być dwukierunkowe. Układy te wprowadzają określone opóźnienia. Opóźnienie to sumuje się z czasem dostępu do pamięci i wywołuje takie skutki,jakby pamięć była wolniejsza niż jest w rzeczywistości. Dlatego produkcja wzmacniaczy oparta jest na szybkiej technologii bipolarnej.

Opóźnienie przez układ 8228 wynosi 30 do 40 ns, zaś przez dwukierunkowe wzmacniacze dla szyny danych 25 do 30 ns, w zależności od kierunku rozchodzenia się sygnałów.

HOLD, HLDA, $\overline{\text{BUSEN}}$, Ø 2TTL

Istnieje możliwość uniknięcia pośrednictwa akumulatora przy przesyłaniu danych z zewnątrz do pamięci i odwrotnie. W ten sposób oszczędza się czas.

Proces bezpośredniej wymiany danych między pamięcią i światem zewnętrznym nazywa się bezpośrednim dostępem do pamięci lub w skrócie DMA /Direct Memory Access/. DMA dokonuje się w ten sposób, że między dwoma rozkazami procesor "odłącza się" od szyny danych, tzn. przechodzi do stanu wysokiej impedancji wyjściowej /Floating Mode/. Stan taki trwa przez określoną liczbę całkowitych okresów taktu podstawowego. W ten sposób procesor przechodzi w stan zawieszenia /Hold State/ Do bezpośredniej wymiany danych między procesorem a światem zewnętrznym wykorzystuje się specjalną jednostkę peryferyjną /DMA Controller/. W mikrokomputerze opartym na mikroprocesorze Intel 8080 DMA steruje układem scalonym 8257.

Żądanie DMA przeprowadza się przez doprowadzenie do wejścia HOLD wysokiego poziomu sygnału. Po zakończeniu wykonywania rozkazu bieżącego mikroprocesor przyjmuje żądanie DMA i zawiadamia o tym świat zewnętrzny przez ustalenie na wyjściu HLDA /HoLD Acknowledged/ wysokiego poziomu sygnału.

Następnie mikroprocesor "odłącza się" od szyny danych. Dzieje się to w wyniku działania sygnału $\overline{\text{BUSEN}}$ /BUS ENabled/ pochodzącego z jednostki peryferyjnej sterującej DMA. Dopóki trwa DMA na wejściu HOLD jest wysoki poziom sygnału.

Przy pracy jednostki peryferyjnej sterującej DMA stosuje się również sygnał Ø 2TTL. Ponadto sygnały $\overline{BUSEN}$ i Ø 2TTL stosowane są również w innych celach.

W przykładzie z rys.9 DMA nie jest stosowana i dlatego końcówka HOLD połączona jest z masą, końcówka $\overline{BUSEN}$ również z masą, natomiast końcówka Ø 2TTL "wisi".

$\overline{RESIN}$

Gdy włączy się zasilanie, w rejestrach mikroprocesora ustala się określony, najczęściej taki sam stan początkowy. Jest to skutek niesymetryczności przerzutników, z których utworzone są rejestry. Po włączeniu zasilania program rozpoczyna się wykonywać od rozkazu zawartego w komórce pamięciowej, której adres jest równy początkowej zawartości rejestru programu.

Jednakże program jest zazwyczaj tak napisany, że pierwszy rozkaz znajduje się w komórce pamięciowej o adresie 0000H. Mało jest prawdopodobne, aby początkowa zawartość rejestru programowego była właśnie 0000H. Oznacza to, że najprawdopodobniej przy włączaniu zasilania zaistnieje chaos w mikrokomputerze. Dlatego przy włączaniu zasilania należy rejestr programu automatycznie wyzerować. W tym celu wykorzystuje się końcówkę $\overline{RESIN}$ /RESest logic INput/.

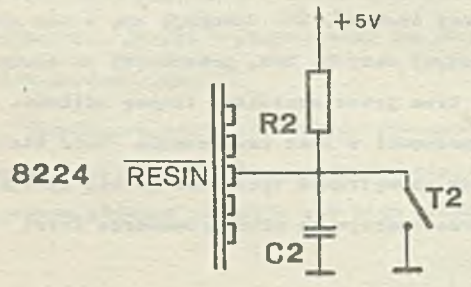
Dopóki na wejściu $\overline{RESIN}$ jest niski poziom sygnału program nie rozpoocznie działania. Schemat logiczny układu 8224 jest taki, że rejestr programu będzie wyzerowany, jeżeli niski poziom sygnału na wejściu $\overline{RESIN}$ trwa co najmniej przez 3 okresy taktu podstawowego.

Dlatego też końcówkę $\overline{RESIN}$ łączy się tak, jak przedstawiono na rys.23.

Elementy $R_2$ i $C_2$ dobiera się tak, aby:

$$\tau_2 = R_2 \cdot C_2 \gg 3T_o$$

Jeżeli pracuje się z $T_o$ = 500 ns, to wystarczy $R_2$ = 1K oraz $C_2$= 4,7 nF. Lepiej jest wybrać większy opornik, a mniejszy kondensator.




Rys.23. Sposób połączenia końcówki $\overline{RESIN}$

Dopóki zasilanie jest wyłączone kompensator $C_2$ jest nienaładowany. Po włączeniu zasilania napięcie w kondensatorze $C_2$ wolno narasta i w przybliżeniu można uważać, że jest ono przez 3 pierwsze okresy podstawowego taktu równe zeru. W tym czasie rejestr programowy jest zerowany. Program nie rozpoocznie się jednak od momentu wyzerowania rejestru programu, ale dopiero gdy napięcie na wejściu $\overline{RESIN}$ dostatecznie wzrośnie. Jeżeli w trakcie pracy zaistnieje potrzeba aby mikroprocesor ponownie wystartował od pierwszego rozkazu w programie, przycisk $T_2$ należy za chwilę nacisnąć. Wówczas kondensator $C_2$ chwilowo się rozładuje, napięcie na wejściu $\overline{RESIN}$ spadnie do zera, rejestr programu wyzeruje się, zaś mikroprocesor wystartuje, gdy napięcie na wejściu $\overline{RESIN}$ dostatecznie wzrośnie, tzn. gdy przycisk zostanie zwolniony.

Również w trakcie pierwszych 3 okresów zerowane są przerzutniki INTE i HLDA; wyjście INTE i HLDA są wyjściami właśnie tych przerzutników. Pozostałe rejestry nie są zerowane.

Istnieją również inne metody inicjalizacji programu.

ROM

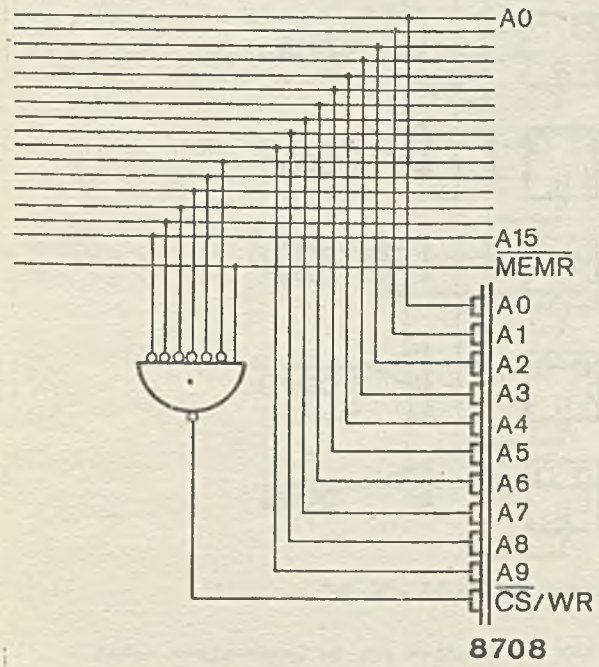
Pamięć ROM 8708 ma 24 końcówki. Końcówki O1 do O8 /Output 1 do Output 8/ powinny być połączone z szyną danych, końcówki VDD, VCC, VSS i VBB z szyną zasilającą, zaś końcówka $\overline{CS}$/WR /Chip Select/ z linią $\overline{MEMR}$ na szynie sterującej.

Końcówka PROGRAM odgrywa specjalną rolę przy wprowadzaniu programu do ROM. W trakcie eksploatacji programu nie jest ona wykorzystywana i łączy się ją z masą lub z + 5 V. Również końcówka $\overline{CS}$/WR /WRite/ ma określoną rolę przy wprowadzaniu programu.

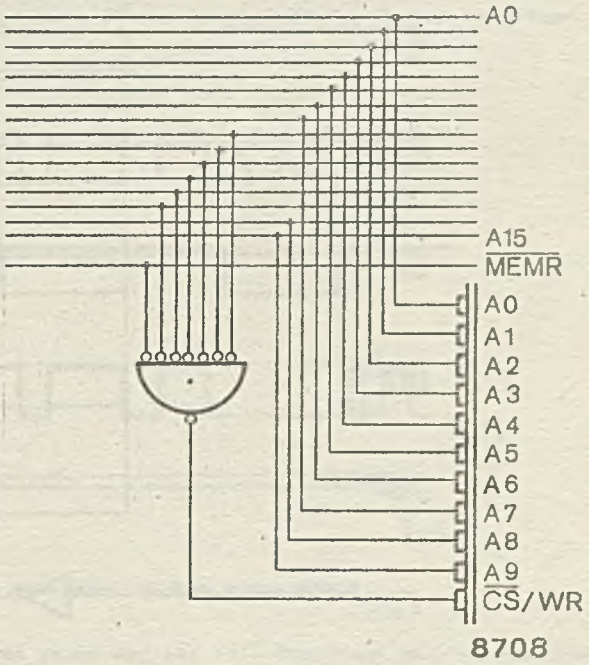
Końcówki A0 do A9 łączy się z szyną adresową. Teraz będzie opisany sposób ich połączenia.

Pojemność pamięci ROM 8708 komórki pamięci. Każda komórka pamięci ma swój adres; które 1024 adresy ze zbioru 65.536 adresów pamięci będą przyporządkowane do zbioru 1024 komórek pamięci - będzie zależało od sposobu połączenia końcówek od A0 do A9 z szyną adresową.

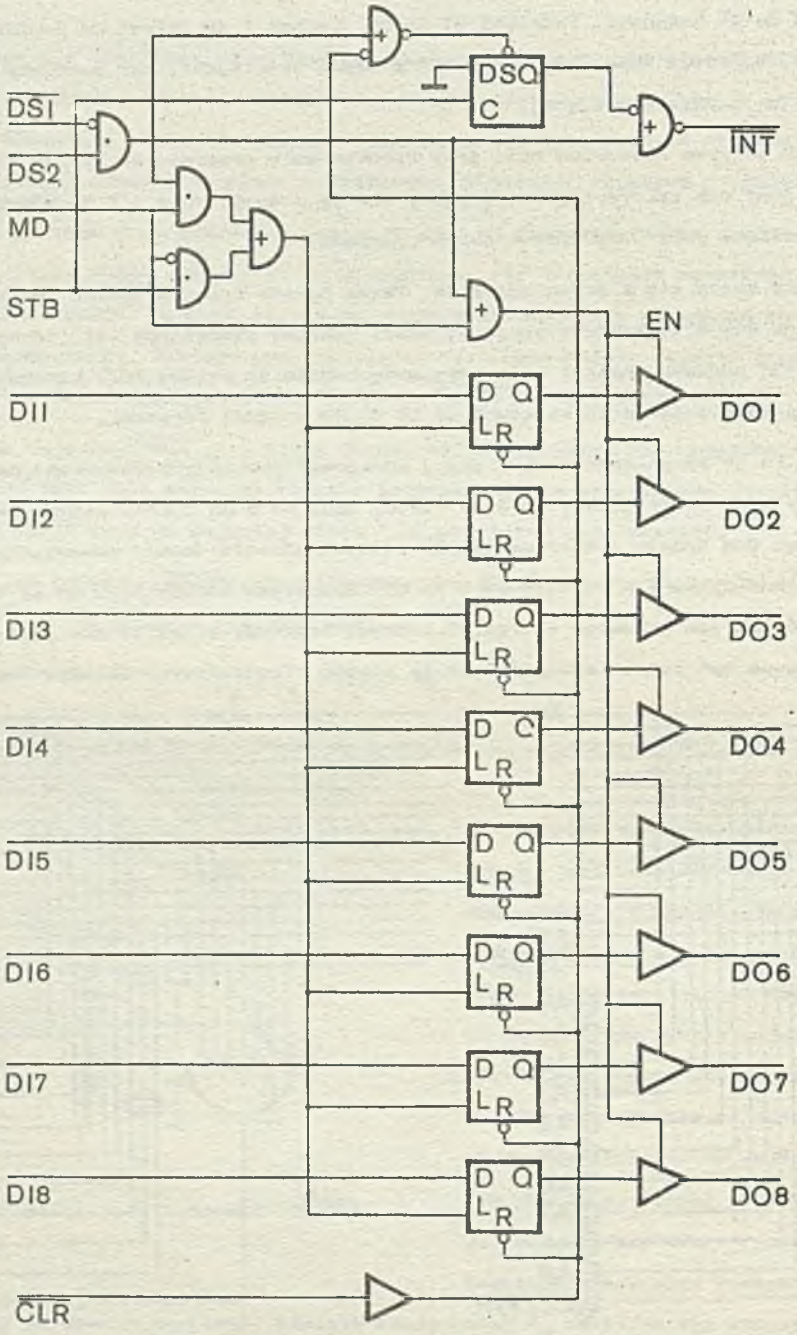
Jeżeli końcówki A0 do A9 połączone są z szyną adresową, jak to przedstawiono na rys.9, wówczas komórki adresowe będą miały adresy od 0 do 1023D, bądź od 0 do 03FFH. Jednak każda komórka tak połączonej pamięci ROM będzie wybierana dla 64 różnych adresów. Jeżeli chcemy, aby komórki pamięci ROM były wybierane tylko dla adresów od 0 do 03FFH, wówczas końcówki A0 do A9 oraz końcówkę $\overline{CS}$/WR należy połączyć tak jak pokazano na rys.24. Schemat logiczny z rys.24 nie jest standardowym rozwiązaniem, jednak może być łatwo zrealizowany za pomocą standardowych układów logicznych.




Rys.24. Jednoznaczne dekodowanie adresów pamięci od 0 do 03FFH




Rys.25. Jednoznaczne dekodowanie adresów pamięci od 0 do 00FFH, od 4000H do 40FFH, od 8000H do 80FFH oraz od 4000H do C0FFH

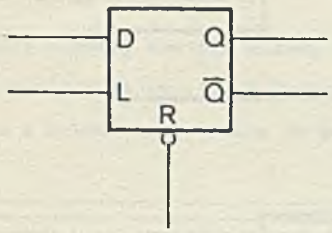



Rys.26. Schemat logiczny układu 8212

Jeżeli chcemy przypisać komórkom pamięci ROM tylko adresy między 0 a 00FFH, między 4000H a 40FFH, między 8000H a 80FFH, jak również między C000H a C0FFH, wówczas końcówkę A0 do A9 oraz końcówkę $\overline{CS}$/WR należy połączyć w sposób przedstawiony na rys.25.

Jednostki peryferyjne

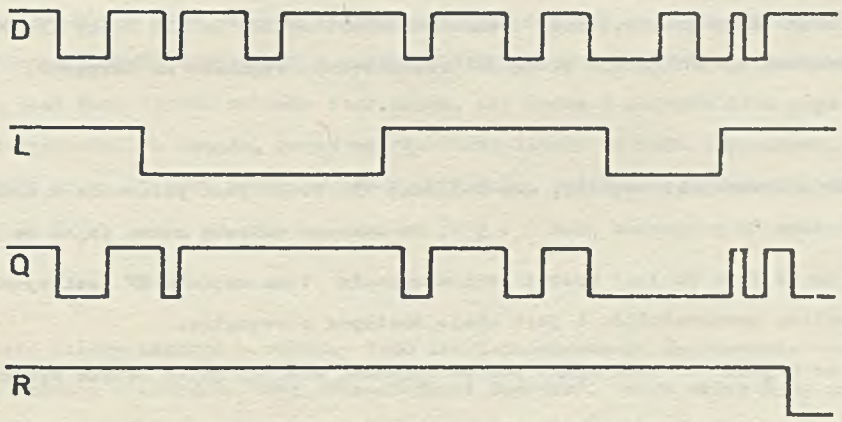
Jednostki peryferyjne 8212 mają 24 końcówki. Jeden układ 8212 wykorzystywany jest wyłącznie jako wejście lub wyłącznie jako wyjście. Schemat logiczny układu 8212 podany jest na rys.26. Rdzeń układu 8212 stanowi 8 elementów logicznych, które umownie nazwijmy przerzutnikami L /Latch/. Symboliczne oznaczenie przerzutnika L przedstawione jest na rys.27.




Rys.27. Symbol przerzutnika L

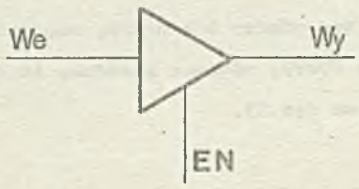
Przerzutnik L ma trzy wejścia /D, L i R/ oraz jedno wyjście /Q/. Przebiegi czasowe charakteryzujące przerzutnik L pokazane są na rys.28.

Gdy na wejściu L jest wysoki poziom sygnału, wówczas zawartość wejścia D jest bezpośrednio przenoszona na wyjście Q. Gdy na wejściu L jest niski poziom sygnału, poziom na wyjściu Q jest ustalony i równy poziomowi, jaki istniał na wyjściu Q w momencie zmiany poziomu logicznego z wysokiego na niski na wejściu L. Tak się dzieje, gdy na wejściu R jest wysoki poziom sygnału. Gdy jednak na wejściu R jest niski poziom sygnału,wówczas również na wyjściu Q jest zawsze niski poziom sygnału. Wyjścia przerzutników L są połączone z wejściami ośmiu buforów o trzech stanach /Three State Buffer/. Symbol trójstanowego bufora przedstawiony jest na rys.29.




Rys.28. Przebiegi czasowe charakteryzujące przerzutnik L

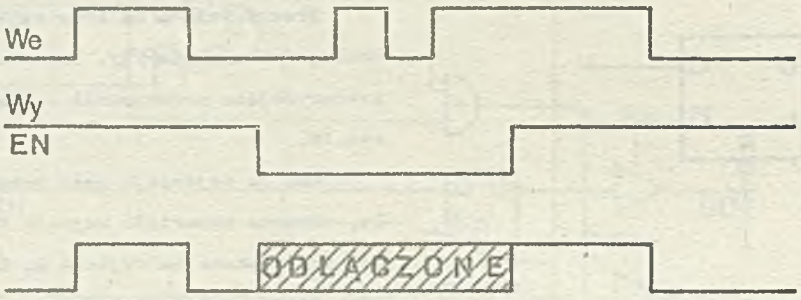
Bufor trójstanowy ma dwa wejścia /U, EN/ oraz jedno wyjście /I/. Przebiegi czasowe określające zachowanie się bufora trójstanowego przedstawione są na rys.30.
Dopóki na wejściu EN /Enable/ jest wysoki poziom sygnału, zawartość wejścia U przenoszona jest




Rys.29. Symbol bufora trójstanowego

bezpośrednio na wyjście I. Natomiast dopóki na wejściu EN jest niski poziom sygnału, wyjście I po prostu jest "odłączane", tzn. bufor przechodzi w stan wysokiej impedancji wyjściowej.

Każdy bufor, zwykły lub trójstanowy, odgrywa równocześnie rolę wzmacniacza prądu, np. w sytuacji przedstawionej na rys.26 wejście bufora pobiera maksymalnie 0,25 mA, zaś wyjście bufora daje 15 mA.




Rys.30. Przebiegi czasowe określające zachowanie się bufora trójstanowego

Poza układami logicznymi realizującymi przenoszenie danych, w układzie 8212 istnieją również określone układy logiczne sterowania. Pierwsze z nich wykorzystują końcówki od DI1 do DI8 oraz od DO1 do DO8, natomiast drugie końcówki MD, $\overline{DS1}$, DS2, STB, $\overline{CLR}$ oraz $\overline{INT}$.

Jeżeli 8212 służy jako wejście, wówczas końcówki od DO1 do DO8 łączy się z szyną danych, natomiast końcówki od DI1 do DI8 służą do przyjmowania sygnałów z zewnątrz.

Jeżeli 8212 służy jako wyjście, wówczas końcówki od DI1 do DI8 łączy się z szyną danych, natomiast końcówki od DO1 do DI8 służą do przekazywania sygnałów na zewnątrz.

MD

Układ 8212 służy jako wejście, gdy końcówka MD /Mode/ jest połączona z masą lub jako wyjście, gdy końcówka MD połączona jest z + 5 V. Do takiego wniosku można dojść na podstawie rys.26.

Jeżeli na wejściu MD jest wysoki poziom sygnału i na wejściu EN jest wysoki poziom sygnału, wówczas zawartość przerzutników L jest stale dostępna z zewnątrz.

Jeżeli na wejściu MD jest niski poziom sygnału, wówczas na EN będzie wysoki poziom sygnału tylko wtedy, gdy

$$\overline{DS1} \cdot DS2 = 1$$

tzn. jeżeli dany układ 8212 jest wybrany przez mikroprocesor. Oznacza to, że zawartość przerzutników L jest wyprowadzana na końcówki od DO1 do DO8 tylko wtedy, gdy dany element 8212 zostanie wybrany. Przez pozostały czas 8212 jest "odłączony" i nie przeszkadza przekazywaniu kodów rozkazowych przez szynę danych. Dlatego końcówki od DO1 do DO8 można łączyć z szyną danych.

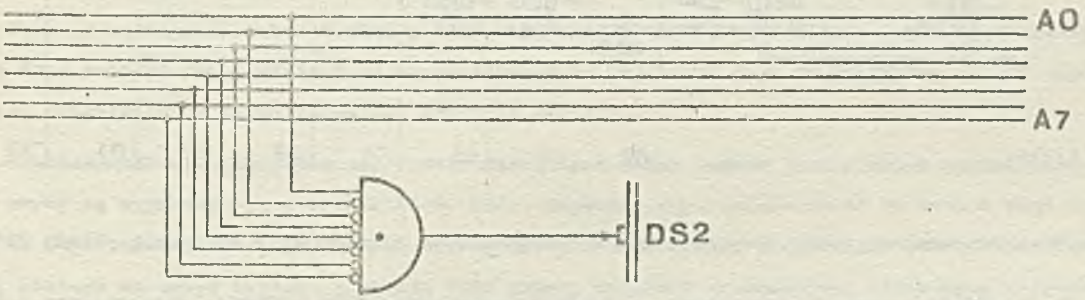
$\overline{DS1}$, DS2

Końcówki $\overline{DS1}$ i DS2 /Device Select/ służą do wybierania układu 8212. Jeżeli układ 8212 służy jako wejście, $\overline{DS1}$ łączy się z linią $\overline{I/OR}$ na szynie sterującej. Jeżeli 8212 służy jako wyjście, wówczas $\overline{DS1}$ łączy się z linią $\overline{I/OW}$.

Wejście DS2 służy do określania adresów komórki peryferyjnej, znajdującej się w układzie 8212.

Na rys.9 wejście DS2 w obydwóch układach 8212 dołączone jest linią A0 na szynie adresowej. Dlatego obydwa układy 8212 reagują na adres 1. Nie dojdzie do nieporozumienia, ponieważ sygnały różnią się na wejściu $\overline{DS1}$. Jednak oba układy 8212 reagują na jeszcze 128 innych adresów, tzn. na wszystkie adresy nieparzyste.

Jeżeli chcemy, aby obu układy 8212 reagowały tylko na adres 1, wówczas końcówkę DS2 należy połączyć tak jak na rys.31. 8-wejściowy układ z rys.31 nie jest obwodem standardowym, jednak może być on z łatwością wykonany za pomocą układów standardowych.




Rys.31. Jednoznaczne przypisanie adresu 1 układowi 8212

Jeżeli w mikrokomputerze jest znaczna liczba wejściowych lub/i wyjściowych układów 8212, wówczas wybór za pomocą układów logicznych z rodziny 7400 nie jest wygodny z dwóch powodów. Po pierwsze,konieczna jest duża liczba układów logicznych, aby każdy z układów 8212 mógł być jednoznacznie wybrany w systemie. Po drugie, przez użycie dużej liczby obwodów logicznych, liczba odgałęzień na wyjściach A0 do A7 może wzrosnąć ponad maksymalną wartość współczynnika wzmocnienia /Fan-Out/.

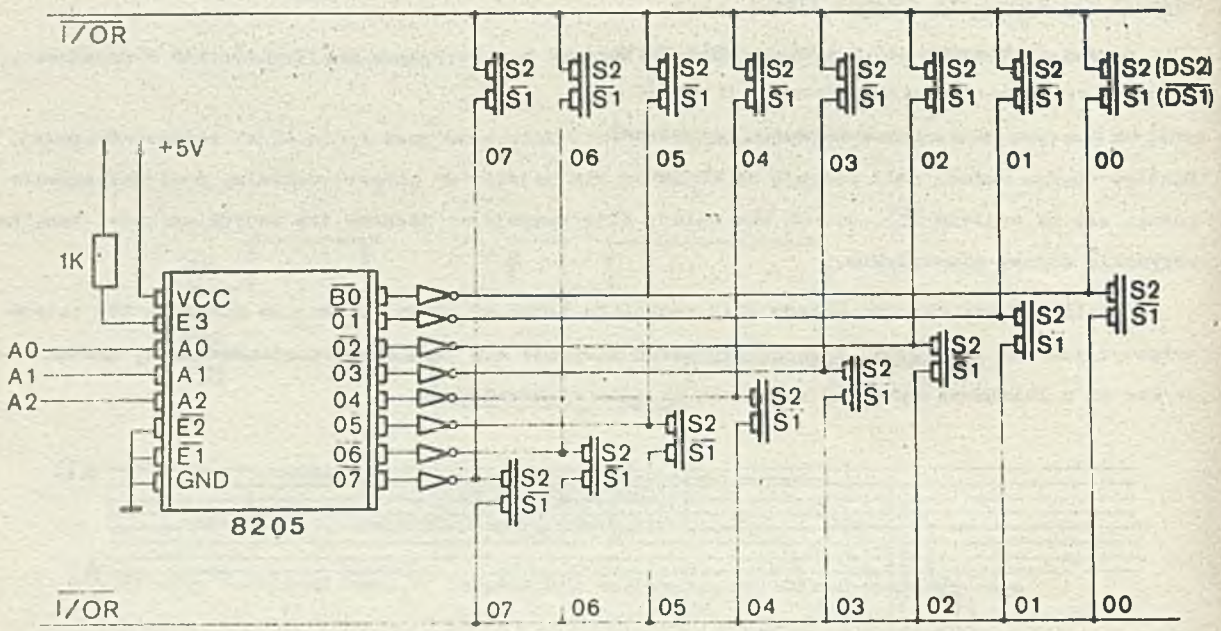
8205

Zamiast znacznej liczby układów z rodziny 7400 dla jednoznacznego dekodowania adresów stosuje się specjalny dekoder w układzie 8205 /One-of-Eight Decoder/. Jeden układ 8205 zapewnia jednoznaczne dekodowanie adresów 9-wejściowych i 8-wyjściowych jednostek peryferyjnych jakiegokolwiek rodzaju. Widać to na rys.32, na przykładzie 16 układów 8212.

Dla uproszczenia zamiast całych układów 8212 narysowano tylko końcówki $\overline{DS1}$ i DS2. Z powodu braku miejsca końcówki te oznaczone są jako $\overline{S1}$ i S2.

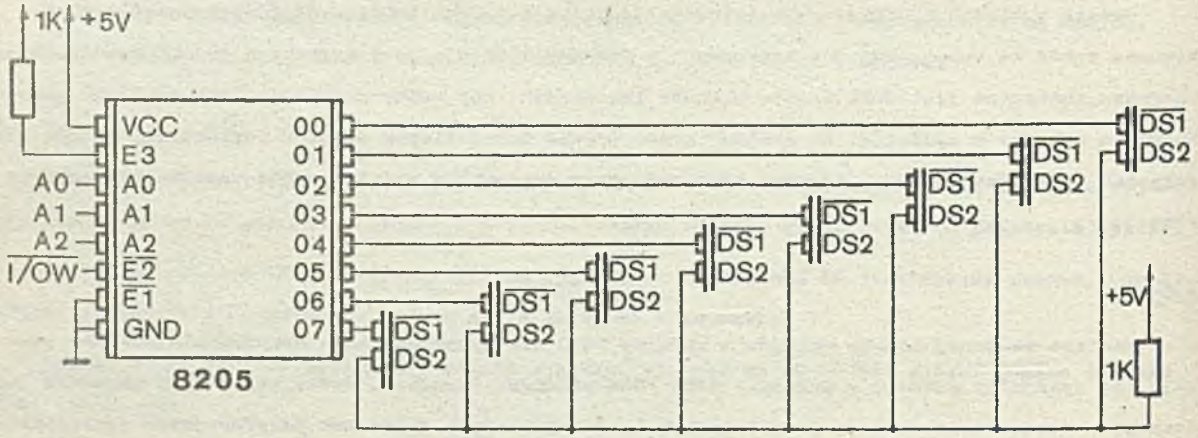
Wejścia A0, A1 i A2 łączy się z odpowiednimi liniami na szynie adresowej. Wejścia $\overline{E1}$, $\overline{E2}$ i E3 służą do wybierania układu 8205. Gdy w systemie wykorzystuje się tylko jeden układ 8205, wejś-

cia te łączy się tak jak na rys.32. Schemat logiczny w układzie 8205 jest taki, że przy jednej kombinacji poziomu na wejściach A0, A1 i A2 mamy niski poziom tylko na jednym z 8 wyjść od 01 do 08. Układy 8212 na rys.32, zarówno wejściowe, jak i wyjściowe, mają adresy od 00H do 07H. Adresy poszczególnych układów oznaczone są na samym rysunku.



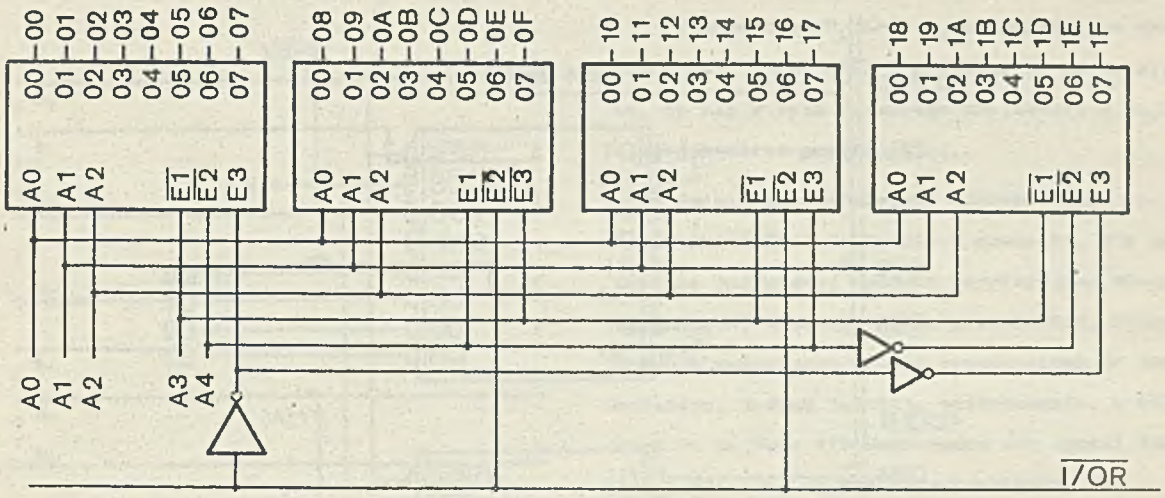

Rys.32. Jednoznaczny wybór 8-wejściowych i 8-wyjściowych układów 8212 za pomocą układu 8205

Na rys.33 inwertery są opuszczone, natomiast jeden układ 8205 wykonuje jednoznaczne dekodowanie tylko 8-wyjściowych jednostek peryferyjnych 8212.




Rys.33. Jednoznaczny wybór 8 układów wyjściowych 8212 za pomocą układu 8205

Jeżeli w systemie znajdują się 32 jednostki peryferyjne, wówczas potrzebne są 4 układy 8205. Łączy się je z szyną adresową tak jak na rys.34.




Rys.34. Jednoznaczny wybór 32 jednostek peryferyjnych za pomocą układów 8205

Oczywiście układy 8205 mogą być wykorzystywane również do jednoznacznego wyboru komórek pamięci.

Poza jednoznacznym wyborem układy 8205 zapewniają również wzmocnienie logiczne. Mianowicie układ 8205 pobiera maksymalnie 0,25 mA, natomiast na wyjściach daje 10 mA. W ten sposób współczynnik rozgałęzienia szyny adresowej znacznie wzrasta.

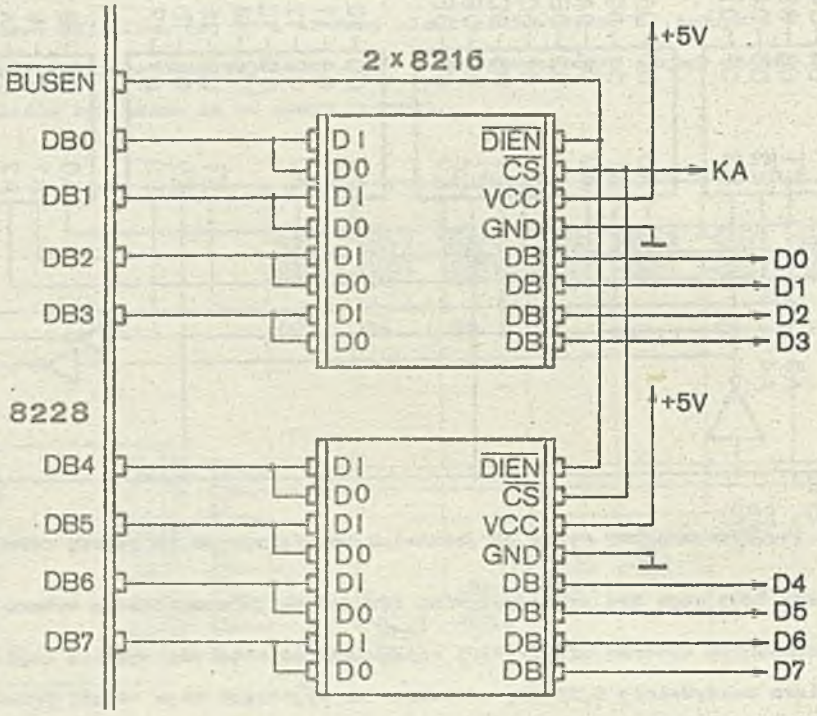
Współczynnik rozgałęzienia szyny sterującej oraz szyny danych jest również ograniczony. Te dwie szyny są wyprowadzone z układu 8228, który zapewnia maksymalnie 10 mA na swoich wyjściach. Jest to zbyt mało wówczas, gdy mikrokomputer zawiera dużą liczbę komórek pamięci lub peryferyjnych. Dlatego na samym wyjściu z układu 8228 należy umieścić wzmacniacze, które będą zwiększać możliwość obciążenia prądowego.

Przez szynę sterującą sygnały przenoszone są tylko w jednym kierunku. Dlatego stosuje się tu zwykłe wzmacniacze jednokierunkowe. Jednym z układów z 6 wzmacniaczami tego rodzaju jest 7407.

8216, 8226

Przez szynę danych sygnały przenoszone są w obu kierunkach. Dlatego stosuje się tu wzmacniacze dwukierunkowe /Bidirectional Bus Driver/. Wzmacniacze dwukierunkowe spotyka się też pod nazwą transywerów /Transciever/. Układy 8216 i 8226 zawierają po 4 takie wzmacniacze. Obydwa wymienione układy mają po 16 końcówek i różnią się tym, że pierwszy z nich nie dokonuje inwersji sygnału natomiast drugi to robi. Poza tym drugi wymaga mniejszego prądu zasilania i wprowadza mniejsze opóźnienie przy przechodzeniu sygnałów.

Szyna danych wymaga stosowania dwóch układów 8216. Sposób połączenia tych układów z szyną danych przedstawiony jest na rys.35.




Rys.35. Sposób łączenia układów 8216

Podsumowanie

Problemy związane z jednoznaczną selekcją i podnoszeniem poziomu prądu na szynie adresowej rozwiązują wzmacniające układy wybierania np. 8205. Problemy związane z podnoszeniem poziomu w szynie danych rozwiązują dwukierunkowe wzmacniacze, np. 8216. Problemy związane z podnoszeniem poziomu na szynie sterującej rozwiązują zwykłe wzmacniacze jednokierunkowe, np. 7407.

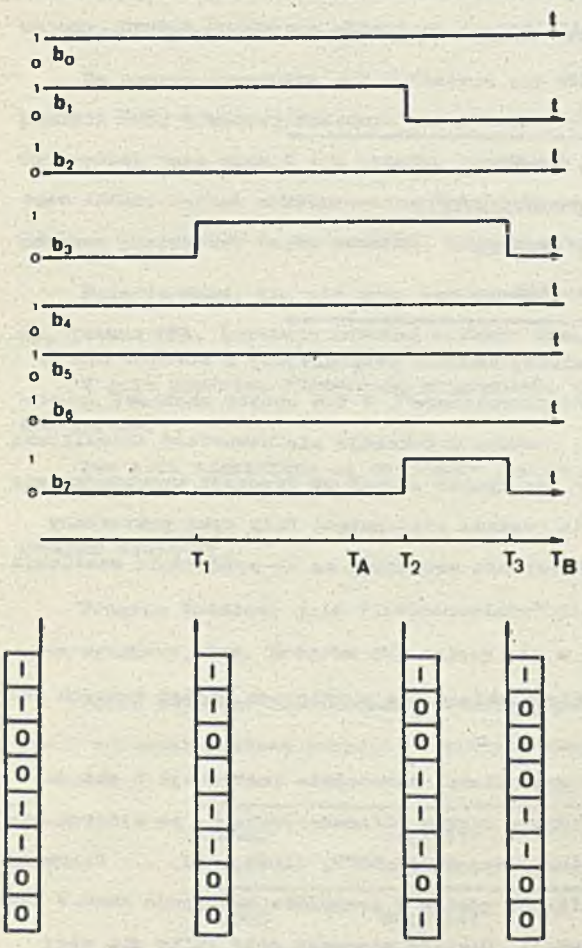
Wszystkie te układy produkuje się w technologii bipolarnej. Dlatego też powodują małe opóźnienia sygnałów. W wypadku układów 8216 i 8226 opóźnienie to wynosi odpowiednio 30 ns i 25 ns.

Im większa jest podstawowa częstotliwość rezonatora krystalicznego, tym czynnik rozgałęzienia wyjścia jest mniejszy !

STB

Na rys.26 widać, że sygnał na wejściu STB /STroBe/ działa na przerzutniki L tylko wtedy, gdy na wejściu MD jest niski poziom sygnału, tzn. gdy 8212 służy jako wejście.

Sygnały z zewnątrz przenoszone są na wejście bufora trójstanowego tylko wówczas, gdy na wejściu STB jest wysoki poziom sygnału. Jeżeli końcówkę STB połączymy z + 5 V zawartość wejściowej komórki peryferyjnej będzie bezpośrednio odpowiadała wejściowym sygnałom cyfrowym /rys.36/. W momentach $T_1$, $T_2$ i $T_3$ na rys.36 zmieniają się poziomy sygnałów niektórych sygnałów wejściowych. W tych samych momentach ulega zmianie również zawartość wejściowej komórki peryferyjnej, tzn. zawartość na wyjściach przerzutników L.




Rys.36. Sposób wprowadzania sygnałów zewnętrznych do mikrokomputera

W momentach $T_A$ oraz $T_B$ mikroprocesor wybiera dany układ 8212 i przenosi do akumulatora to, co się w tych momentach znajdowało w wejściowej komórce peryferyjnej.

Jeżeli zdarzy się, że mikroprocesor wybiera układ 8212 dokładnie w momencie, gdy zawartość wejściowej komórki peryferyjnej ulega zmianie, wówczas nie można przewidzieć, która z dwóch zawartości będzie przeniesiona do akumulatora. Jednak istnieją zastosowania, w których na wejście STB doprowadza się sygnał taktu lub jakiś inny sygnał asynchroniczny.

CLR

Sygnał $\overline{CLR}$ /CLeaR/ działa na wejścia R wszystkich przerzutników i zeruje te przerzutniki. W przykładzie przedstawionym na rys.9 nie zachodzi potrzeba zerowania przerzutników L i dlatego końcówka $\overline{CLR}$ połączona jest z + 5 V.

INT

Końcówkę $\overline{INT}$ wykorzystuje się przy obsłudze przerwania. W przykładzie z rys.9 nie stosuje się przerwania i dlatego końcówka $\overline{INT}$ "wisi".

VGG, VSS

Końcówki VGG oraz VSS łączy się szyną zasilającą.

DI, DO

W przykładzie przedstawionym na rys.9 wykorzystuje się tylko końcówkę DI1 w wejściowym układzie 8212 oraz końcówkę DO1 w wyjściowym układzie 8212. Końcówki od DO2 do DO8 "wiszą". Dowolnie przyjęto, że końcówki od DI2 do DI8 połączone są z masą. Zawartość wejściowej komórki peryferyjnej będzie zatem następująca.

0000000X

Najmłodszy bit /$b_0$ = X/ ulega zmianie w zależności od wejściowego sygnału cyfrowego, natomiast wartość pozostałych bitów jest stale równa zeru.

Tak samo można przyjąć, że końcówki od DI2 do DI8 połączone będą z + 5 V. W takiej sytuacji zawartość wejściowej komórki peryferyjnej byłaby następująca

1111111X

PROGRAM ŹRÓDŁOWY

Program napisany w języku programowania asembler nazywa się programem źródłowym. Program źródłowy dokonujący inwersji ma następującą postać:

```
JEDEN:   IN      01H      ; WEJŚCIE
         CMA               ; UZUPEŁNIENIE
         OUT     01H       ; WYJŚCIE
         JMP     JEDEN     ; SKOK
```

Przez pierwszy rozkaz wprowadza się zawartość wejściowej komórki peryferyjnej z adresem 01H do akumulatora. Drugim rozkazem uzupełnia się zawartość akumulatora*). W ten sposób zachodzi inwersja sygnału wejściowego. Za pośrednictwem trzeciego rozkazu przekazuje się zawartość akumulatora do wyjściowej komórki peryferyjnej z adresem 01H. W ten sposób sygnał po inwersji wyprowadza się na zewnątrz. Czwartym rozkazem dokonuje się skoku do rozkazu pierwszego. Cały cykl powtarzany jest od momentu, gdy napięcie na wejściu $\overline{\text{RESIN}}$ dostatecznie wzrośnie, aż do wyłączenia zasilania.

Instrukcje w języku asemblera

Ogólnie biorąc każda instrukcja w języku asemblera składa się z czterech części zwanych polami /Field/.

Pierwsze pole stanowi etykietę /Label/, czyli symboliczne oznaczenie instrukcji i składa się z symbolu alfanumerycznego o 5 lub mniejszej liczbie znaków alfanumerycznych, po których następują dwie kropki. Pierwszym znakiem musi być litera. Np. ADR1, LOOP, JEDEN, itd. ... Pierwsze pole wypełnia się tylko dla takich instrukcji, do których gdzieś w programie następuje skok. W naszym przykładzie skok następuje do pierwszej instrukcji. Dlatego pierwsze pole tylko dla niej jest wypełnione. Nie będzie błędem wypełnienie pierwszego pola również w wypadku instrukcji, do których nigdy nie dokonuje się skoku. A więc pierwsze pole może, ale nie musi być wypełnione.

Drugie pole /Operation/ określa wykonywaną operację i składa się z nazwy dwu-, trzy- i czteroliterowej, nazywanej mnemonikiem, np. IN, CMA, OUT, JMP itd. Pole to jest zawsze wypełnione.

Poszczególne operacje wymagają, aby poza operacją była zdefiniowana jeszcze jedna wielkość związana z operacją. Wielkość ta nazywa się operandem i wprowadzana jest do trzeciego pola /Operand/, składającego się z symbolu alfanumerycznego o 5 lub mniejszej liczbie znaków. Pierwszym znakiem może być litera albo cyfra, np. mnemonik IN definiuje przekazywanie danych z komórki peryferyjnej do akumulatora. Jednak w mikrokomputerze opartym na mikroprocesorze Intel 8080 może istnieć 256 wejściowych jednostek peryferyjnych. Dlatego należy również zdefiniować adres wejściowej jednostki peryferyjnej. Wtedy operand odpowiada adresowi komórki peryferyjnej. W naszym przykładzie jest to 01H. Po każdej cyfrze w operandzie powinna znajdować się litera B, D /nie

*) Tzn. zera są zamieniane na jedynki, a jedynki na zera /dop.red./.

obowiązkowo/, Q lub H. W ten sposób specyfikuje się, do jakiego systemu liczbowego dana liczba należy. Symbol Q dotyczy oktalnego systemu liczbowego.

Za pomocą mnemonika JMP definiuje się skok programowy. Jednak trzeba zdefiniować adres w pamięci ROM, w której znajduje się pierwszy /może jednocześnie jedyny/ bajt rozkazu, do którego wykonywany jest skok.W tym wypadku operandem jest adres pamięci. W naszym przykładzie operandem jest JEDEN. Jednak mikroprocesor "wie", że operand JEDEN jest w rzeczywistości liczbą 0000H, tzn. adresem pierwszego bajtu rozkazu. Operandem może być dana, nazwa rejestru itp.

Pole to może, ale nie musi być wypełnione. Niektóre rozkazy nie wymagają istnienia operanda np. rozkaz CMA. Istnieją również rozkazy wymagające dwóch operandów.

W polu czwartym /Commment/ programista wpisuje dowolny komentarz. Pole to rozpoczyna się średnikiem.

Dwa pola oddzielone są od siebie jednym lub kilkoma pustymi miejscami.

PROGRAM WYNIKOWY

Program źródłowy jest "niezrozumiały" dla mikroprocesora. Mikroprocesor "rozumie" tylko program wynikowy, tzn. program składający się z ciągu binarnie zakodowanych rozkazów.

Celem lepszego zrozumienia powtórzymy program źródłowy dla inwersji razem z programem wynikowym i adresami komórek pamięci, w których znajdują się poszczególne rozkazy.

| | | | | |
|---|---|---|---|---|
| 0000 | 11011011 | JEDEN | IN | 01H |
| 0001 | 00000001 | | | |
| 0002 | 00101111 | | CMA | |
| 0003 | 11010011 | | OUT | 01H |
| 0004 | 00000001 | | | |
| 0005 | 11000011 | | JMP | JEDEN |
| 0006 | 00000000 | | | |
| 0007 | 00000000 | | | |

Zamiana programu źródłowego na program wynikowy dokonywana jest za pomocą "tłumacza" programowego, który również nazywa się asembler. Używa się go, wykorzystując w jednym z urządzeń stosowanych do wspomagania projektowania mikrokomputera. O asemblerze i o urządzeniach wspomagania projektowania będzie mowa w jednym z załączników.

Ogólna postać pierwszej instrukcji jest następująca:

IN ADRP

gdzie ADRP jest konkretną wartością adresu komórki peryferyjnej, której zawartość wprowadzana jest do akumulatora. Rozkaz ten składa się z dwóch bajtów. Pierwszy bajt definiuje operację i ma postać:

11011011

Bajt ten jest określony przez producenta.

Na końcu książki załączone są kody wszystkich rozkazów mikroprocesora Intel 8080, jest ich 91

Drugi bajt definiuje operand, tzn. konkretną wartość adresu komórki peryferyjnej, której zawartość wprowadzana jest do akumulatora. Ponieważ adresem komórki peryferyjnej jest 01H, drugi bajt ma postać:

00000001

Pierwszy rozkaz umieszczony jest w komórkach pamięci ROM, których adresami są 0000H i 0001H.

Rozkazem CMA /Complement Accumulator/ uzupełnia się zawartość akumulatora. Składa się on z jednego bajtu. Bajt ten definiuje operację i ma postać:

00101111

Rozkaz CMA jest umieszczony w komórce pamięci ROM z adresem 0002H.

Trzeci rozkaz wygląda następująco:

OUT ADRP

Rozkaz ten składa się również z dwóch bajtów. Pierwszy definiuje operację i ma postać:

11010011

Drugi bajt definiuje operand. Ponieważ adres wyjściowy komórki peryferyjnej w naszym przykładzie ma postać 001H, to drugi bajt jest następujący:

00000001

Do rozkazów IN ADRP oraz OUT ADRP podobne są rozkazy LDA ADRM i STA ADRM.

Rozkazem LDA ADRM wprowadza się do akumulatora zawartość komórki pamięci, której adresem jest ADRM. Rozkaz ten składa się z trzech bajtów. Pierwszy bajt definiuje operację. Operand natomiast jest obecnie 16-cyfrową liczbą binarną, ponieważ definiuje adres komórki pamięci. Dlatego operand składa się z dwóch bajtów. Drugi bajt zawiera młodszy bajt adresu, zaś trzeci zawiera starszy bajt adresu.

Rozkazem STA ADRM wprowadza się do akumulatora zawartość komórki pamięci, której adresem jest ADRM.

Kody tych dwóch rozkazów, jak i wszystkich pozostałych rozkazów można znaleźć w załączniku.

Jak już wspomnieliśmy, mikroprocesor Intel 8080 dopuszcza przesyłanie danych między pamięcią a pozostałymi rejestrami. Przesyłanie danych między pamięcią a parą rejestrową HL dokonywane jest za pomocą rozkazów LHLD ADRM oraz SHLD ADRM. Pierwszym rozkazem zawartość komórki pamięci o adresie ADRM przenosi się do rejestru L, a zawartość komórki pamięci o adresie ADRM + 1 przenosi się do rejestru H. Drugim rozkazem dokonuje się przesłania w przeciwnym kierunku. Ogólna postać ostatniego rozkazu jest następująca:

JMP ADRM

Przez ten rozkaz JMP /JuMP/ dokonuje się skoku do rozkazu, którego pierwszy /a może i jedyny/ bajt znajduje się w komórce pamięci z adresem ADRM. Również i ten rozkaz składa się z 3 bajtów. Pierwszy bajt definiuje operację i ma postać:

11000011

Drugi i trzeci bajt definiują operand. Drugi bajt zawiera młodszą połowę, a trzeci bajt starszą połowę adresu pamięci, do którego dokonuje się skoku. W naszym przykładzie rozkazem

JMP JEDEN

dokonuje się skoku do adresu 0000H. Dlatego drugi i trzeci bajt mają postać:

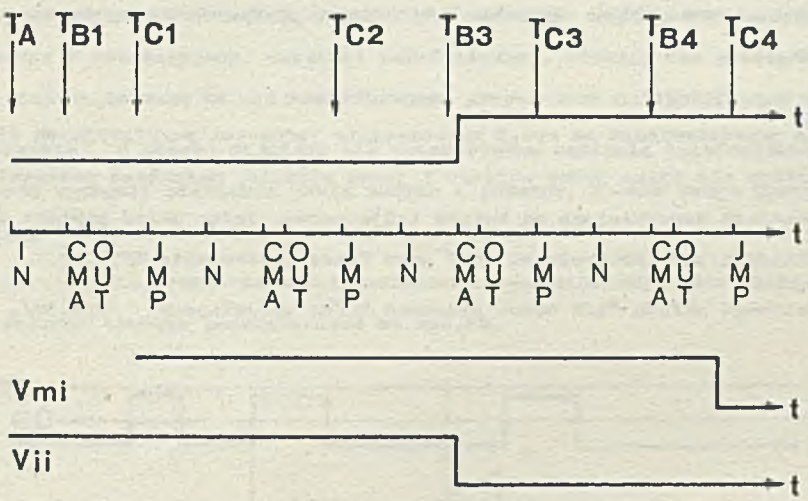
00000000

00000000

Rozkaz JMP umieszczony jest w komórkach pamięci ROM, których adresy mają postać 0005H, 0006H oraz 0007H.

PRZEBIEG PROGRAMU

Przebieg opisanego programu przedstawiono na rys.37.




Rys.37. Przebieg programu realizującego inwersję sygnału wejściowego

W załączniku umieszczonym na końcu książki można zauważyć, że rozkaz IN ADRP trwa przez 10 okresów taktu podstawowego. Do akumulatora przenosi się taki stan, jaki zaistniał na wyprowadzeniach wejściowych układu 8212 w połowie dziewiątego okresu taktu podstawowego /$T_{B1}$, $T_{B2}$, $T_{B3}$/. Po pierwszym wykonaniu tego rozkazu /$T_{B1}$/ zawartość akumulatora jest następująca:

00000000

Rozkaz CMA trwa przez 4 okresy taktu podstawowego. Po wykonaniu tego rozkazu zawartość akumulatora jest następująca:

11111111

Rozkaz OUT ADRP trwa przez 10 okresów taktu podstawowego. Zawartość akumulatora "wychodzi" na wyprowadzenia wyjściowe układu 8212 w połowie dziewiątego okresu taktu podstawowego /$T_{C1}$/. Stan ten utrzymuje się na wyprowadzeniach wyjściowych układu 8212 aż do następnego wykonywanego rozkazu OUT. Wówczas to zawartość układu 8212 może, ale nie musi ulec zmianie. Jest to skutek faktu, że końcówki MD oraz $\overline{CLR}$ połączone są z + 5 V.

Po wykonaniu tego rozkazu zawartość akumulatora w dalszym ciągu jest następująca:

11111111

Rozkaz JMP trwa przez 10 okresów taktu podstawowego. Po wykonaniu tego rozkazu zawartość akumulatora jest następująca:

11111111

Następnie wszystko się powtarza. W momentach $T_{B2}$, $T_{B3}$, $T_{B4}$... "pobiera" się zawartość wejściowego układu 8212, zaś w momentach $T_{O2}$, $T_{C3}$, $T_{C4}$... "odnawia" się zawartość wyjściowego układu 8212.

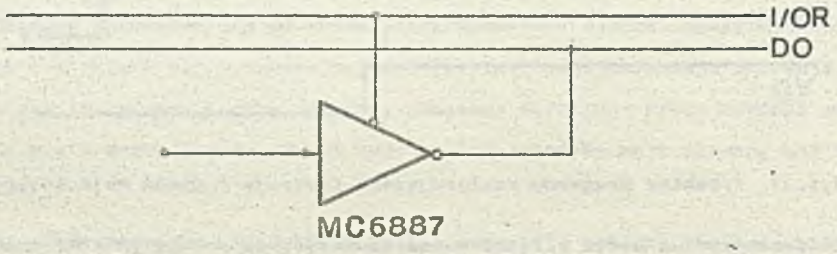
Sygnał wyjściowy z mikrokomputera pracującego jako inwerter /$V_{mi}$/ opóźnia się w stosunku do sygnału wyjściowego z inwertera idealnego /$V_{ii}$/ w najgorszym wypadku nawet o 48 okresów taktu podstawowego.

Jeżeli podstawowa częstotliwość rezonatora kwarcowego wynosi 18 MHz, wówczas opóźnienie to wynosi 2⅔ μs. Oczywiście, jeszcze raz podkreślamy, że mikrokomputer nie będzie nigdy pracował jako inwerter. Niezależnie od tego, opóźnienie reakcji na sygnał pobudzający wystąpi zawsze. Jest to skutek skończonego czasu obróbki sygnału. W niektórych zastosowaniach to przeszkadza, natomiast w innych nie.

UWAGA

W przykładzie przedstawionym na rys.9 zastosowanie jednostek peryferyjnych 8212 jest nieracjonalne. Wykorzystuje się tylko jedno wejście i jedno wyjście, natomiast pozostałe 7 wejść "wisi". W takich sytuacjach dogodniejsze są bufory trójstanowe. Jeden układ zawiera 6 buforów tego rodzaju. Takimi układami są - Motorola MC 6887 oraz Texas Instruments SN74367.

Zamiast wejściowego układu 8212 można stosować bufor trójstanowy /rys.38/.

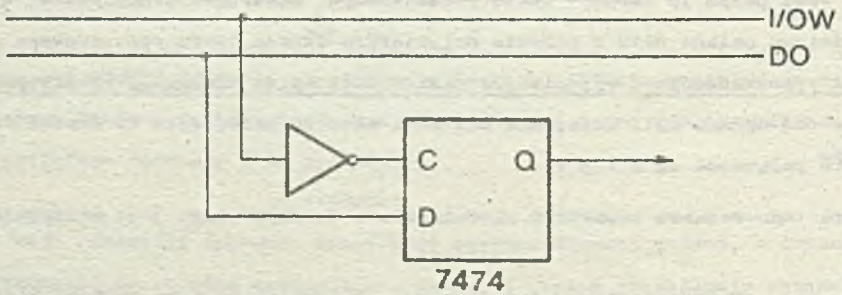



Rys.38. Wprowadzanie sygnału cyfrowego do mikrokomputera za pomocą bufora trójstanowego

Zamiast wyjściowego układu 8212 można stosować przerzutnik D /rys.39/.

Bufory trójstanowe produkowane są w technologii bipolarnej /Schottky Bipolar/ i powodują opóźnienie wynoszące tylko 8 ns.

Jeżeli występuje znaczne obciążenie pojemnościowe, wówczas zaleca się stosowanie specjalnych układów z dwoma wzmacniaczami w układzie. Tego rodzaju układem jest Motorola MMH0026.




Rys.39. Wyprowadzanie sygnału cyfrowego z mikrokomputera za pomocą przerzutników D

LITERATURA

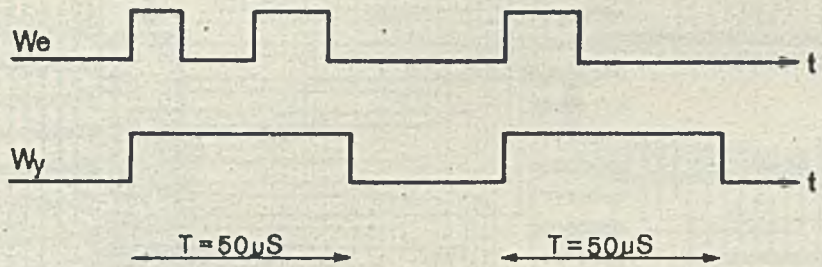
D1, D2, D3, F1, F2, K1, K2.

## MULTIWIBRATOR MONOSTABILNY

Obecnie zajmiemy się drugim z kolei przykładem wstępnym. Oczywiście, obowiązują takie same zastrzeżenia jak w przykładzie pierwszym. Mija się z celem projektowanie multiwibratora monostabilnego za pomocą mikrokomputera. Jednak struktura mikrokomputera, który służy jako multiwibrator monostabilny ze zmienną szerokością impulsu nie różni się zbytnio od struktury mikrokomputera wykorzystywanego w automatycznej centrali telefonicznej. Dlatego też szczegóły sprzętowe tej struktury będą przeanalizowane na nieskomplikowanym przykładzie multiwibratora monostabilnego.

W części trzeciej, w której zajmiemy się automatyczną centralą telefoniczną, zagadnienia sprzętowe nie będą wymagały omawiania wielu nowych elementów, a cała uwaga będzie skierowana na konkretne problemy telekomunikacyjne i szczegóły programowe.

Przebiegi czasowe dla konkretnego multiwibratora monostabilnego /One shots/, którym będziemy się zajmować w dalszym takście przedstawiono na rys.40.

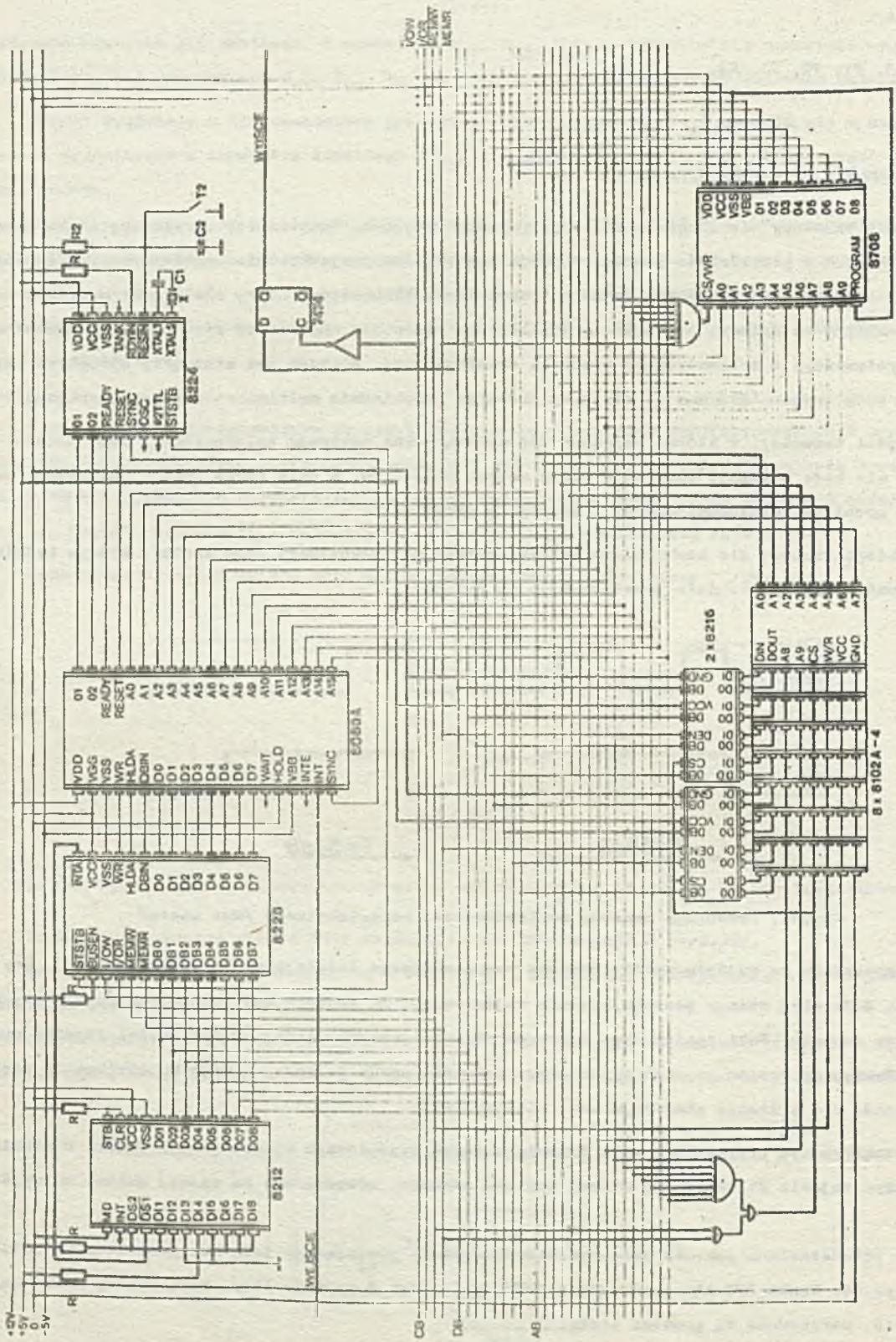



Rys.40. Przebiegi czasowe multiwibratora monostabilnego /One shots/

W stanie spoczynku na wyjściu multiwibratora monostabilnego istnieje niski poziom sygnału. Gdy dojdzie do dodatniej zmiany poziomu sygnału wejściowego, na wyjściu multiwibratora pojawi się wysoki poziom sygnału. Taki poziom trwa przez określony czas odpowiadający szerokości impulsu wyjściowego. Następnie poziom sygnału wyjściowego ponownie spada do zera. A więc multiwibrator monostabilny może być w stanie stabilnym oraz niestabilnym.

W rozpatrywanym przykładzie czas trwania impulsu wyjściowego wynosi 50 μs. Jeżeli w stanie niestabilnym dojdzie do dodatniej zmiany poziomu sygnału, wówczas nie ma żadnej zmiany na wyjściu.

Na rys.41 przedstawiono schemat połączeń mikrokomputera pracującego jako multiwibrator monostabilny z rys.40. Punkt INT stanowi wejście /WE/, zaś punkt Q wyjście /WY/. Połączenia, które należy objaśnić, narysowane są grubszą linią.




Rys. 41. Schemat połączeń mikrokomputera pracującego jako multiwibrator monostabilny

MIKROKOMPUTER Z JEDNYM POZIOMEM PRZERWANIA

Głównym zagadnieniem w przypadku multiwibratora monostabilnego jest obsługa zgłoszenia żądania przerwania. Jeśli zgłoszenie żądania przerwania przychodzi z jednego źródła, wówczas mówimy, że jest jeden poziom przerwania /Interrupt Level/.

W rozpatrywanym przykładzie multiwibratora monostabilnego stan stabilny odpowiada programowi A, zaś stan niestabilny - programowi B z rys.15.

Program A jest obecnie trywialny. Mianowicie, na początku programu A na wyjściu Q wytwarza się niski poziom sygnału. Następnie program A wchodzi w zamkniętą pętlę /Idle State/. W czasie gdy program "biega" na zamkniętej pętli, na wyjściu Q istnieje niski poziom sygnału. Mikroprocesor może wyjść z zamkniętej pętli tylko pod wpływem żądania przerwania, które nadchodzi do wejścia INT. Gdy żądanie przerwania nadejdzie, mikroprocesor przechodzi na wykonywanie programu B. Program B ustala wysoki poziom sygnału na wyjściu Q i utrzymuje go przez 50 μs. Po upływie 50 μs program B zmienia poziom sygnału na wyjście Q na niski. Następnie program B zatrzymuje się, lub program A kontynuowany jest tam, gdzie został zatrzymany, tzn. w zamkniętej pętli. W ten sposób za pomocą mikrokomputera realizowany jest impuls wyjściowy z multiwibratora monostabilnego o określonym czasie trwania.

Program A ma następującą postać:

| | | | | | |
|---|---|---|---|---|---|
| /A/ | 0000 | | LXI | SP 0800H | 00110001 |
| | 0001 | | | | 00000000 |
| | 0002 | | | | 00001000 |
| | 0003 | | XRA | A | 10101111 |
| | 0004 | | OUT | 01H | 11010011 |
| | 0005 | | | | 00000001 |
| | 0006 | | EI | | 11111011 |
| | 0007 | ADI: | NOP | | 00000000 |
| | 0008 | | IMP | AD1 | 11000011 |
| | 0009 | | | | 00000111 |
| | 000A | | | | 00000000 |

Przyjęto, że program A rozpoczyna się od adresu 000H; można było umieścić go gdziekolwiek w pamięci ROM. Jednak wybrane rozwiązanie jest najdogodniejsze ze względów praktycznych. Przypomnijmy sobie, że rejestr programu automatycznie zeruje się w momencie włączenia zasilania pod warunkiem, że końcówka $\overline{\text{RESIN}}$ jest połączona jak na rys.23.

Program B jest następujący:

| | | | | | |
|---|---|---|---|---|---|
| /B/ | 0010 | | CMA | | 00101111 |
| | 0011 | | OUT | 01H | 11010011 |
| | 0012 | | | | 00000001 |
| | 0013 | | MVI | A,05D | 00111110 |
| | 0014 | | | | 00000101 |
| | 0015 | AD2: | DCR | A | 00111101 |
| | 0016 | | JNZ | AD2 | 11000010 |
| | 0017 | | | | 00010101 |
| | 0018 | | | | 00000000 |
| | 0019 | | NOP | | 00000000 |
| | 001A | | NOP | | 00000000 |

| | | | |
|---|---|---|---|
| 001B | OUT | 01H | 11010011 |
| 001C | | | 00000001 |
| 001D | EI | | 11111011 |
| 001E | RET | | 11001001 |

Przy wyborze początkowego adresu programu B nie mieliśmy całkowitej swobody. Później wyjaśnimy z jakiego powodu. Również do sprawy szczegółowej analizy programów A i B, czyli do zaznajomienia się z nowymi rozkazami, wrócimy w dalszym tekście. Obecnie musimy dokładniej zaznajomić się z obsługą przerwania.

RST N

W momencie $T_{IA}$ na rys.47 mikroprocesor przyjmuje żądanie przerwania. Po przyjęciu żądania przerwania mikroprocesor przechodzi do programu B. Aby móc przejść do programu B, mikroprocesor powinien "znać" adres pierwszego rozkazu programu B. A więc zachodzi potrzeba istnienia odrębnego układu, który po przyjęciu żądania przerwania przekaże mikroprocesorowi adres pierwszego rozkazu programu B. Tym układem jest dobrze nam znany układ 8212. Oznacza to, że obsługa przerwania wymaga odrębnego układu 8212, który nie służy do komunikowania się ze światem zewnętrznym, a do obsługi przerwania. Ten układ 8212 znajduje się w górnym lewym rogu rys.41.

W naszym przykładzie adres początkowy programu B równy jest 0010H. W przedziale między momentem $T_{IA}$ a $T_{RST}$, na rys.17, układ 8212 przekazuje mikroprocesorowi adres 0010H. Przez ten cały czas poziom sygnału $\overline{INTA}$ jest równy zeru. Adres 0010H staje się zawartością rejestru programu, a program kontynuuje swój przebieg od rozkazu, którego adresem początkowym jest 0010H lub od rozkazu N1 w programie B na rys.17.

Jeżeli układ 8212 służy do obsługi przerwania, wówczas końcówki $\overline{DS1}$ w układzie 8212 i końcówka $\overline{INTA}$ w układzie 8228 powinny być połączone. Ten sposób umożliwił mikroprocesorowi wybranie 8212 w momencie $T_{IA}$, gdy będzie potrzebował adres pierwszego rozkazu programu B. Układ 8212 jest wybierany aż do momentu, gdy kończy się wprowadzanie adresu. Końcówka $\overline{DS2}$ łączy się z + 5 V. Dlatego tylko sygnał DS1 wybiera układ 8212.

Końcówki D11, D12, D16, D17 i D18 m u s z ą być połączone z + 5 V. Końcówki D13, D14 i D15 łączy się z masą lub +5 V. Sposób połączenia tych 3 końcówek jednoznacznie definiuje adres pierwszego rozkazu wykonywanego po przyjęciu żądania przerwania. Trzy końcówki mogą być połączone na 8 różnych sposobów. Oznacza to, że pierwszy bajt pierwszego rozkazu programu B może znajdować się tylko w 8 komórkach pamięci ROM. Adresami tych komórek są 0000H, 0008H,0010H,0018H,0020H,0028H, 0030H i 0038H. W naszym przykładzie /patrz program B/ pierwszy i jedyny bajt pierwszego rozkazu programu B ma adres 0010H. W tab.2. przedstawiono zależności między sposobem łączenia końcówek D13, D14 i D15 a adresem pierwszego rozkazu, wykonywanego po przyjęciu żądania przerwania.

Jak widać, nie ma dużej dowolności przy wyborze adresu pierwszego rozkazu programu B. W naszym przykładzie pierwszy rozkaz programu B ma adres 0010H. Dlatego końcówki D13 i D15 połączone są z masą, zaś końcówka D14 z +5 V. Praktycznie w przedziale między momentami T3 a T4 dokonywane jest sprzętowe wytwarzanie kodu dla rozkazu RST N /ReSTard/. Rozkaz RST N nie występuje ani w programie A, ani B. Tak więc rozkaz RST N nie jest wytwarzany w sposób programowy, tak jak to się

| ADRES | D15 | D14 | D13 | N |
|---|---|---|---|---|
| 0000H | 0 | 0 | 0 | 0 |
| 0008H | 0 | 0 | 1 | 1 |
| 0010H | 0 | 1 | 0 | 2 |
| 0018H | 0 | 1 | 1 | 3 |
| 0020H | 1 | 0 | 0 | 4 |
| 0028H | 1 | 0 | 1 | 5 |
| 0030H | 1 | 1 | 0 | 6 |
| 0038H | 1 | 1 | 1 | 7 |

Tab.2

dzieje ze wszystkimi rozkazami w programach A i B. Jest on wytwarzany sprzętowo za pomocą układu 8212 i sygnału $\overline{INTA}$. Kod rozkazu RST N jest następujący:

111XXX11

Cyfry binarne XXX definiują adres pierwszego rozkazu programu B i stanowią binarny odpowiednik liczby dziesiętnej N. Po wykonaniu rozkazu RST N zawartość wskaźnika stosu jest następująca:

0000000000XXX000B

Np. dla rozkazu RST 2 obowiązuje:

N = 2 i XXX = 010

Dlatego kod rozkazu RST 2 jest następujący:

11101011

Po wykonaniu rozkazu RST 2 zawartość rejestru programowego jest następująca:

0000000000010000B

Dlatego w praktyce za pomocą rozkazu RST 2, dokonuje się skoku do rozkazu, którego adresem jest 0010H.

Pamięć RAM i jej rola przy obsłudze przerwania

Przy wpisywaniu adresu pierwszego rozkazu programu B uprzednia zawartość rejestru programu ulega skasowaniu. Jednak po zakończeniu programu B mikroprocesor musi wrócić do programu A. Powrót do programu A może być dokonany jeżeli zostanie zachowana zawartość wskaźnika stosu przy wejściu do programu B. Temu celowi służy pamięć RAM. Tak więc obsługa przerwania wymaga bezwzględnego stosowania pamięci RAM. Czas dostępu statystycznej pamięci RAM z rys.41 wynosi 450 ns zaś pojemność 1024 bajtów. Składa się ona z 8 układów o kodzie 8101A-4. W każdym układzie znajduje się tylko jeden bit wszystkich 1024 komórek. Pierwszy układ z prawej zawiera 1024 najmłodszych bitów itd. Jeden układ pamięci RAM może zawierać 2, 4 lub nawet wszystkie 8 bitów jednej komórki pamięci. Pamięć RAM jest tańsza, gdy zostanie zrealizowana na ośmiu kostkach, niż na jednej kostce, ale w pierwszym wypadku zajmuje więcej miejsca na płycie drukowanej.

Końcówki od A0 do A9 związane są z szyną adresową przez układy dekodujące, tak więc z pamięciami RAM powiązane są tylko adresy od 1024D = 0400H do 2047D = 07FFH. Adresy od 0000D = 0000H do 1023D = 03FFH wiąże się tylko z pamięciami ROM.

Końcówkę W/R łączy się linią $\overline{MEMW}$ szyny sterującej.

Końcówki DIN /Data IN/ oraz DOUT /Data OUT/ służą do wprowadzania i wyprowadzania danych. Szyna danych jest w naszym przykładzie prosta, tzn. nie ma oddzielnej szyny wejściowej ani wyjściowej.

Końcówki DIN oraz DOUT połączone są z szyną danych przez kostki 8216, jak to przedstawiono na rys.41.

Końcówki VCC oraz GND połączone są szyną zasilającą. Mogliśmy również zdecydować się na zastosowanie dynamicznej pamięci RAM. Wówczas połączenie pamięci RAM z mikrokomputerem następuje za pośrednictwem układu 8210 /TTL-to-MOS Level Shifter and High Voltage Clock Driver/ bądź 8222 /Dynamic Memory Refresh Controller/.

Stos

Umieszczanie zawartości wskaźnika stosu w pamięci RAM dokonywane jest automatycznie w ramach rozkazu RST N. Programista nie musi się tym zajmować. Jednak programista musi zdefiniować obszar w pamięci RAM, w której tymczasowo będzie przechowywana zawartość wskaźnika stosu. Obszar ten nazywa się stosem /STACK/. Tak się dzieje w mikroprocesorze Intel 8080. Natomiast w innych rodzinach mikroprocesorów bywają stosy zrealizowane jako odrębny układ /PACE-National/.

Stos w naszym przykładzie definiuje się na samym początku programu A. W praktyce istotne jest zdefiniowanie stosu przed umieszczeniem pierwszej danej na stosie, co w zasadzie nie musi występować na samym początku programu. Stos definiuje się w ten sposób, że do wskaźnika stosu wprowadza się adres pierwszej komórki pamięciowej w stosie zwiększony o 1. Załóżmy, że pierwsza komórka pamięciowa w stosie ma adres 07FFH. Dlatego więc do wskaźnika stosu wprowadza się liczbę 0800H, co dokonywane jest za pomocą rozkazu LXI SP, 0800H.

Ogólna postać takiego rozkazu jest następująca:

LXI RP, P16

Taki rozkaz ma dwa operandy. Pierwszy z nich definiuje parę rejestrową /SP, BC, DE lub HL/. Drugi - definiuje daną 16-bitową przekazywaną do tej pary rejestrowej. Rozkaz LXI RP, P16 składa się zatem z trzech bajtów. Pierwszy bajt określa operację względem pierwszego operandu i ma postać:

00RR0001

Para bitów RR definiuje jedną z 4 par rejestrowych. W załączniku umieszczonym na końcu książki można sprawdzić, że wskaźnikowi stosu /SP/ odpowiada RR = 11. Dlatego pierwszy bajt rozkazu LXI SP, 0800H ma następującą postać:

00110001

Dwa następne bajty definiują drugi operand. W przypadku rozkazu LXI SP, 0800H wyglądają one następująco:

00000000
00001000

W naszym przykładzie początek stosu jest stały, tzn. nie ulega zmianie w trakcie programu. Oczywiście w praktyce mamy sytuacje, gdy początek stosu jest redefiniowany w trakcie programu. Stos nie ma ustalonej pojemności. Jego szerokość zależy od tego ile danych znajduje się w nim. Tak więc stos rozszerza się kosztem pozostałych części pamięci RAM. Na początku, po rozkazie LXI SP,

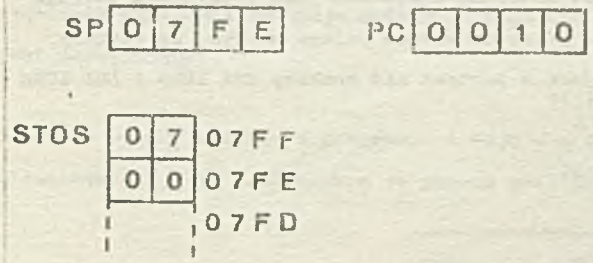
0800H pojemność stosu wynosi zero. Przez umieszczenie danych w stosie, jeśli mikrokomputer oparty jest na mikroprocesorze Intel 8080, stos rozszerza się w kierunku niższych adresów pamięci. Dlatego w naszym przykładzie stos znajduje się na samym szczycie pamięci RAM, tzn. początek stosu pokrywa się z najwyższym adresem ze zbioru adresów przyznanych pamięci RAM. A więc stos nie jest niczym innym niż częścią pamięci RAM.

Natomiast przesyłanie danych między stosem a mikroprocesorem dokonywane jest w inny sposób niż przesyłanie danych między pozostałymi częściami pamięci RAM a mikroprocesorem.

Przesyłanie danych między procesorem a stosem może być dokonywane dwoma sposobami: automatycznie i programowo.

Jednym z rozkazów, w ramach którego dokonywane jest przesyłanie automatyczne między procesorem a stosem jest RST N. Tymczasowe zapamiętywanie zawartości rejestru programu nie jest dokonywane w dowolnym miejscu pamięci RAM, lecz na stosie.

Dla ilustracji załóżmy, że adres rozkazu N6 w programie A z rys.17 równy jest 0007H. Gdy wykona się rozkaz RST N, wskaźnik stosu, rejestr programu i stos mają taką zawartość, jak na rys.42.




Rys.42. Stan stosu po przyjęciu pierwszego żądania przerwania

Młodsza połowa adresu rozkazu N6 znajduje się na stosie w komórce adresowej, której adresem jest 07FFH. Starsza połowa tego adresu znajduje się w komórce pamięci o adresie 07FEH. Wskaźnik stosu zawsze wskazuje szczyt stosu +1. Dlatego nowa zawartość wskaźnika stosu równa jest 07FEH. Po tym rozkazie pojemność stosu wynosi dwa bajty. Programowe wpisywanie danych na szczyt stosu dokonywane jest za pomocą rozkazu PUSH RP. Kod tego rozkazu jest następujący:

11RR0101

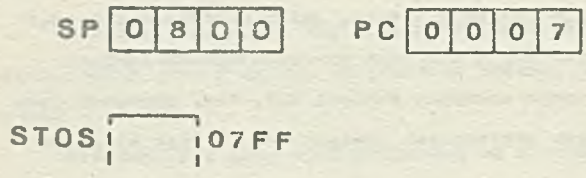
Para bitów, jak i uprzednio, definiuje parę rejestrową, której zawartość przerzucana jest do stosu.

Programowe odczytywanie danych ze szczytu stosu może być również automatyczne lub programowe. Jednym z rozkazów, w ramach którego dokonuje się automatycznego przesłania danych ze szczytu stosu do procesora jest rozkaz RET /RETurn/. Rozkaz RET musi być ostatnim rozkazem w programie B. Rozkazem tym zawartość dwóch komórek pamięci ze szczytu stosu umieszcza się w rejestrze programu. W ten sposób dokonuje się powrotu do programu A, tzn. do rozkazu N6 z rys.17. Na rys.17 wykonanie rozkazu RET kończy się w momencie $T_{RET}$. W tym momencie zawartość rejestru programu wynosi znowu 0007H, a zawartość wskaźnika stosu wynosi znowu 0800H. Taka sytuacja przedstawiona jest na rys.43. Pojemność stosu wynosi znowu zero.

Programowe odczytywanie danych ze szczytu stosu dokonywane jest za pomocą rozkazu POP RP. Kod dla tego rozkazu jest następujący:

11RR0001

Rozkazem tym zawartość szczytu stosu jest przesyłana do określonej pary rejestrowej.




Rys.43. Stan stosu po zakończeniu obsługi przerwania

Praca ze stosem ma swoje zalety i wady, ale zalety przeważają.

Załóżmy, że w stosie znajduje się kilka tymczasowo zapamiętanych danych. Jeżeli zaistnieje potrzeba, procesor za pomocą tylko jednego rozkazu nie może pobrać dowolnej danej, lecz tylko taką, która ostatnio została umieszczona na stosie. Jest to jedna z wad stosu. W wypadku pozostałej części pamięci RAM przesyłanie danych może być dokonywane między procesorem a jakąkolwiek komórką pamięci, mianowicie rozkazy LDA ADRM oraz STA ADRM zawierają konkretny adres komórki pamięci, do której zwraca się procesor.

Pamięć stosu nie jest pamięcią RAM w dosłownym znaczeniu tego słowa. Stos jest w rzeczywistości pamięcią z dostępem sekwencyjnym /Sequentially Assessible Memory/ bądź pamięcią typu LIFO /Last-In-First-Out/. Dla porządku zaznaczamy, że istnieją również pamięci typu FIFO /First-In First-Out/.

Główna zaleta pracy ze stosem polega na tym, że znacznie ułatwiona jest programowa obsługa przerwania i przetwarzanie podprogramu. Inna zaleta pracy ze stosem polega na tym, że rozkazy PUSH RP i POP RP trwają krócej i zajmują mniej miejsca w pamięci niż rozkazy STA ADRM i LDA ADRM /zob. załącznik znajdujący się na końcu tej pracy/.

Obecnie przechodzimy do analizy programów A i B.

Program A

Rozkazem

LXI SP, 0800H

definiuje się początek stosu. Rozkaz ten nie musi znajdować się na samym początku programu A, ale musi być wykonany przed zwróceniem się procesora do stosu, tzn. przed przyjęciem pierwszego żądania przerwania.

Rozkazem

XRA A /eXclusive oR Accumulator/

zawartość akumulatora jest zerowana, mianowicie przy włączaniu zasilania w akumulatorze powstanie jakaś przypadkowa zawartość. Oddziaływanie na wejściu $\overline{\text{RESIN}}$ zeruje tylko rejestr programu. Dlatego na początku programu, jeżeli jest to potrzebne, musi się znajdować rozkaz zerujący zawartość akumulatora.

Ogólna postać powyższego rozkazu jest następująca:

XRA R

Rozkazem tym wykonuje się operację logicznej różnicy symetrycznej. Jeden operand znajduje się w akumulatorze, zaś drugi w jednym z 7 rejestrów. Kod tego rozkazu jest następujący:

10101RRR

RRR definiuje jeden z 7 rejestrów /A,B,C,D,E,H lub L/. W wypadku akumulatora obowiązuje RRR = 111. /zob. załącznik znajdujący się na końcu tej pracy/.

Przy zerowaniu akumulatora wykorzystuje się następujące własności logicznej różnicy symetrycznej:

$$X \oplus X = 0;\ X = 0,1$$

Do rozkazu XRA R podobne są rozkazy

ANA R /ANd Accumulator/

ORA R /OR Accumulator/

Za pomocą pierwszego wykonuje się operację logicznego iloczynu I, za pomocą drugiego operację logiczną sumy logicznej LUB.

Za pomocą rozkazu OUT 01H zawartość akumulatora /ZERO/ przekazuje się na wyjście Q /rys.41/. Ze względu na sposób w jaki wyjściowy przerzutnik D z rys.41 jest połączony z szyną mikrokomputera, za pomocą każdego rozkazu OUT osiąga się przeniesienie zawartości akumulatora na wyjście Q. W ten sposób na wyjściu Q otrzymuje się niski poziom sygnału odpowiadający stabilnemu stanowi multiwibratora monostabilnego. Poziom ten utrzymuje się na wyjściu Q aż do przyjęcia pierwszego żądania przerwania.

Jednak żądanie przerwania /jeżeli nadejdzie/ nie zostanie przyjęte dopóki nie wykonany będzie rozkaz EI. Na rys.17 jest to moment T1. Jak już powiedzieliśmy, układ na wejściu RESIN po włączeniu zasilania dokonuje automatycznego zablokowania przyjmowania przerwania. Kod rozkazu EI jest następujący:

11111011

Pierwsze cztery rozkazy w programie A mają charakter przygotowawczy. "Rdzeniem" programu A jest nieskończona pętla realizowana za pomocą piątego i szóstego rozkazu.

```
AD1:    NOP
        JMP     AD1
```

Rozkaz

NOP /No OPeration/

nie powoduje wykonania żadnej operacji, "traci" się tylko czas trwający 4 okresy taktu podstawowego czyli 2 $\mu$s. Kod tego rozkazu jest następujący:

00000000

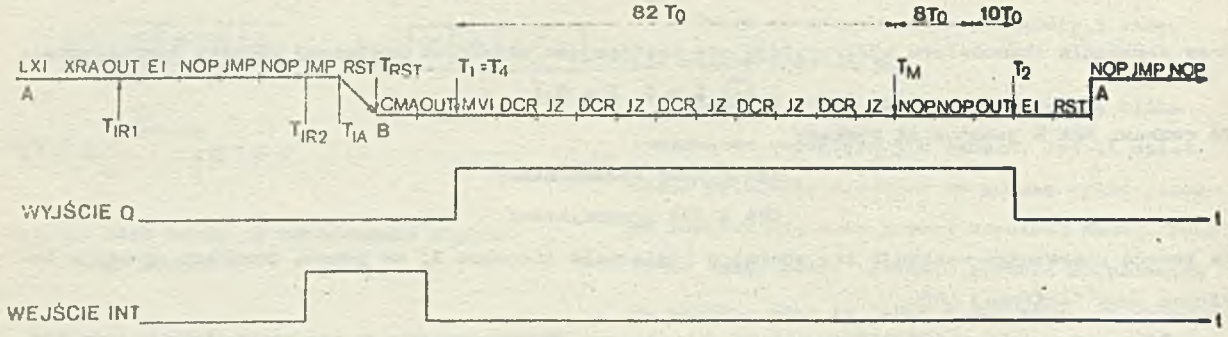
Za pomocą rozkazu JMP ADRM dokonuje się skoku do rozkazu NOP. W ten sposób rozkazy NOP i JMP ADRM wykonywane są na zmianę czyli komputer "chodzi na pusto". Przez ten cały czas na wyjściu Q jest niski poziom sygnału.

Program B

Przebieg programu B w funkcji czasu przedstawiony jest na rys.44. Żądanie przerwania nadchodzi w momencie $T_{IR2}$. Mikroprocesor przechodzi do programu B dopiero, gdy wykonany zostanie rozkaz bieżący /JMP/. Jak już stwierdziliśmy wytwarzanie sprzętowe i wykonanie rozkazu RST 2 dokonywane jest między momentami $T_{IA}$ & $T_{RST}$.

Po wykonaniu rozkazu CMA zawartość akumulatora jest następująca:

11111111




Rys.44. Przebieg programu B w funkcji czasu

Po wykonaniu rozkazu OUT 01H /$T_1$/ na wyjściu Q pojawia się wysoki poziom sygnału. Poziom ten trwa przez 50 µs czyli 100 okresów taktu podstawowego. A więc należy zrealizować odcinek programu /Routine/ trwający dokładnie 100 okresów taktu podstawowego, na końcu którego powstaje niski poziom sygnału na wyjściu Q.

Zazwyczaj mówi się, że taki odcinek programu wprowadza opóźnienie /Delay; Timeout/.

Opóźnienie

Odcinek programu wprowadzający opóźnienie 41 µs /$82T_3$/ ma następującą postać:

```
        MVI   A,   05D
AD2:    DCR   A
        JNZ   AD2
```

Wszystkie trzy rozkazy spotykamy po raz pierwszy. Sieć działań przebiegu powyższego odcinka programu przedstawiona jest na rys.45.

Rozkazem

MVI A, 05D /MoVe Imidiatelly/

umieszcza się w akumulatorze liczbę 05D. Ogólna postać tego rozkazu jest następująca

MVI R, P8

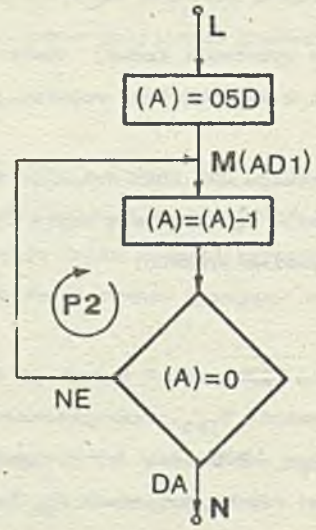
Pierwszy operand definiuje jeden z 7 rejestrów, a drugi operand definiuje daną ośmiobitową umieszczaną w danym rejestrze. Rozkaz ten składa się z dwóch bajtów. Pierwszy z nich definiuje operację oraz pierwszy operand i jest następujący:

00RRR110

Drugi bajt definiujący drugi operand:

00000101

Jak wynika z załącznika znajdującego się na końcu tej pracy wykonanie tego rozkazu trwa




Rys.45. Sieć działań przebiegu odcinka programu wprowadzającego opóźnienie

7 okresów taktu podstawowego. Po wykonaniu tego rozkazu zawartość akumulatora wynosi 05D. Na rys. 45 przedstawiono to symbolicznie jako

/A/ = 05D

Symbolem /R/ w dalszym tekście będzie oznaczana zawartość rejestru R. Symbolem //RP// będzie oznaczana zawartość komórki pamięci, której adres jest równy zawartości pary rejestrowej RP.

Rozkazem

DCR A /DeCRement/

zawartość akumulatora zmniejsza się o 1. Ogólna postać tego rozkazu jest:

DCR R

Operand określa rejestr, którego zawartość zmniejsza się o 1. Kod tego rozkazu jest następujący:

00RRR101

Wykonywanie tego rozkazu trwa 5 okresów taktu podstawowego. Do tego rozkazu podobny jest rozkaz

DCX RP

za pomocą którego zawartość pary rejestrowej RP zmniejsza się o 1.

Istnieją również rozkazy, za pomocą których zawartość rejestru lub pary rejestrowej zwiększa się o 1. Są to rozkazy

INR R /INcRement/

oraz

INX RP

Gdy rozkaz DCR A wykonany zostanie po raz pierwszy, wówczas zawartość akumulatora będzie wynosić 04D.

Rozkazem

JNZ AD2 /Jump if Non Zero/

testuje się stan wskaźnika Z. Jeżeli Z = 1, tzn., gdy zawartość akumulatora równa jest zeru, program jest kontynuowany z następnym rozkazem zapisanym niżej w ciągu. Jeżeli Z ≠ 1, program jest kontynuowany z rozkazem, którego etykietą jest AD2.

Ogólna postać tego rozkazu jest następująca:

JNZ ADRM

Rozkaz ten składa się z trzech bajtów. Pierwszy z nich definiuje operację i jest następujący:

11000010

Drugie dwa bajty definiujące operand:

00010101

00000000

Wykonywanie tego rozkazu trwa 10 okresów taktu podstawowego. Do rozkazu JNZ ADRM jest podobny rozkaz

JZ ADRM /Jump if Zero/

Za pomocą tego rozkazu również testuje się wskaźnik Z, zaś program kontynuuje się z następnym rozkazem zapisanym niżej w ciągu, jeżeli Z = 0.

W podobny sposób rozkazy

JC ADRM /Jump if Carry/

i

i

JNC ADRM /Jump if Non Carry/

testują wskaźnik C, rozkazy

JM /Jump if Minus/

i

JP /Jump if Plus/

wskaźnik S, zaś rozkazy

JPO /Jump if Parity Odd/

i

JPE /Jump if Parity Even/

wskaźnik P.

Oczywiście program przejdzie 4 razy przez pętlę P2 i jeden raz przez "drogę" MN na rys.45. Jedno przejście przez pętlę P2 lub przez "drogę" MN trwa 15 okresów taktu podstawowego. Oznacza to, że odcinek programu między punktami L a N na rys.45 trwa 82 okresy taktu podstawowego. Widać to również na rys.44.

Dwa rozkazy NOP służą tylko do "dopasowania" opóźnienia. Przez ich wykonanie dodaje się 8 okresów taktu podstawowego czyli 4 μs.

Po wyjściu z pętli zawartość akumulatora jest równa zeru. Za pomocą rozkazu OUT 01H zawartość tę przenosi się na wyjście Q. Na wyjściu Q pojawia się niski poziom sygnału, dopiero w momencie zakończenia wykonywania rozkazu OUT 01H.

Ponieważ rozkaz OUT 01H trwa 10 okresów taktu podstawowego wysoki poziom sygnału na wyjściu Q trwa dokładnie 100 okresów taktu podstawowego czyli 50 μs.

W momencie, gdy żądanie przerwania zostaje przyjęte, blokuje się przyjmowanie nowych żądań przerwania. W naszym przykładzie jest to bardzo korzystne, ponieważ przewiduje się, że w trakcie stanu niestabilnego nie przyjmuje się żadnego żądania przerwania. Jednak w trakcie stanu stabilnego musi być przyjęte każde żądanie przerwania i dlatego przedostatni rozkaz programu B musi być rozkazem EI.

Powrót do programu A dokonywany jest za pomocą rozkazu RET. Rozkaz ten musi być ostatnim rozkazem programu B. Jednakże przyjmowanie nowych żądań przerwania jest zabronione aż do momentu zakończenia wykonywania rozkazu RET. Zezwolenie przyjmowania przerwania możliwe jest dopiero po zakończeniu rozkazu następującego po rozkazie EI. Rozkazem następującym jest najczęściej właśnie rozkaz RET.

UWAGI

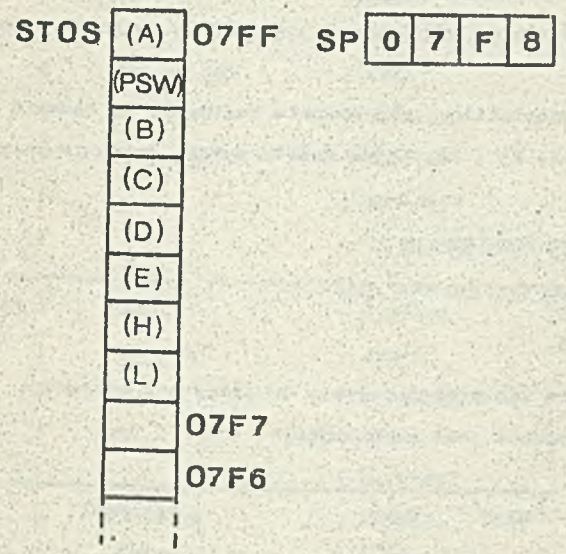
Może się zdarzyć, że program A posiada ważne informacje w rejestrach PSW, A,B,C,D,E,H lub L. Istnieje prawdopodobieństwo wykorzystywania niektórych z tych rejestrów również w programie B. W trakcie programu B zawartość tych rejestrów zmienia się, zaś program A będzie kontynuował działanie z fałszywymi danymi. Dlatego zawartość tych rejestrów, używanych zarówno przez program A, jak i program B, musi być przeniesiona w bezpieczne miejsce przed rozpoczęciem programu B. Najczęściej takim miejscem jest stos, chociaż można użytkować jakąkolwiek część pamięci RAM. A więc zaraz po rozkazie RST N należy umieścić następujący odcinek programu:

| RST | N |
|---|---|
| PUSH | PSW |
| PUSH | BC* |
| PUSH | DE |
| PUSH | HL |

Rozkazem

PUSH PSW

przenosi się zawartość akumulatora i rejestru stanu na stos. Zawartość stosu oraz rejestru wskaźnika stanu po wykonaniu rozkazu PUSH HL przedstawiona jest na rys.46.




Rys.46. Zawartość rejestru wskaźnika stosu oraz stosu po przeniesieniu zawartości rejestru, stanu akumulatora oraz innych rejestrów na stos

Gdy program B kończy się, tymczasowo zapamiętana zawartość wspomnianych rejestrów powinna wrócić do procesora. W tym celu przed rozkazami EI oraz RET umieszcza się następujący odcinek programu:

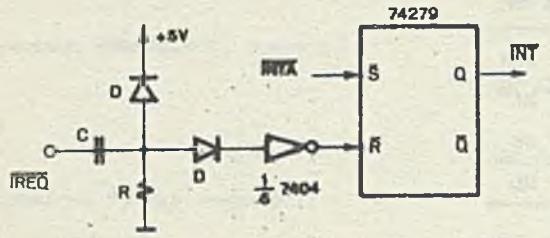
| | |
|---|---|
| POP | HL |
| POP | DE |
| POP | BC |
| POP | PSW |
| EI | |
| RET | |

Należy uwzględnić, że dane ze stosu wracają do procesora w odwrotnej kolejności w stosunku do kolejności, w jakiej były przenoszone na stos.

Istnieje niebezpieczeństwo, że żądanie przerwania trwa dłużej niż 50 μs. Wówczas to samo żądanie przerwania wytwarza dwa impulsy na wyjściu, z których drugi jest fałszywy. Unika się tego za pomocą układu przedstawionego na rys.47.

Żądanie przerwania /IR/ wprowadzane jest przez przerzutnik, który jest przełączany przednim zboczem impulsu. Nadejście żądania przerzutnik ustawia w stanie 1. Natomiast po przyjęciu żądania przerwania przerzutnik jest zerowany. W ten sposób żądanie przerwania przestaje działać na wejście INT, chociaż jeszcze ciągle istnieje. Przerzutnik RS z rys.47 nie jest układem standardowym, jednak może być łatwo wykonany za pomocą układów standardowych.

* Prawidłowa postać tego rozkazu jest PUSH B. Jednakże tutaj operand B, ze względów dydaktycznych zastąpiony jest operandem BC. W ten sposób podkreślono, że chodzi o parę rejestrową BC, a nie o rejestr B. To samo dotyczy niektórych innych rozkazów /w powyższym odcinku programu PUSH DE i PUSH HL/. W każdym razie, w trakcie konkretnej pracy należy korzystać z podręcznika programowania, otrzymanego od producenta.




Rys.47. Układ, za pomocą którego eliminuje się możliwość dwukrotnego zadziałania tego samego żądania przerwania

W naszym przykładzie żądanie przerwania nadchodzi tylko z jednego źródła. Można wówczas zrezygnować z układu 8212: jeżeli końcówkę $\overline{INTA}$ połączy się z +12 V przez opornik o wartości 1K, to po przyjęciu żądania przerwania automatycznie wykonuje się rozkaz RST 7. Oznacza to, że program jest automatycznie kontynuowany od rozkazu, którego adresem jest 0038H.

MIKROKOMPUTER Z 8 POZIOMAMI PRZERWANIA

Systemy z jednym poziomem przerwania rzadko występują; dlatego też pokażemy jak formuje się system z 8 lub mniejszą liczbą poziomów przerwań.

W dalszym ciągu rozpatrujemy multiwibrator monostabilny, ale obecnie założymy, że żądanie przerwania nadchodzi z 7 źródeł: R0, R1, R2, R3, R4, R5 i R6. Każde źródło wymaga czasu trwania impulsu określonej długości /tab.3/.

| $\overline{R0}$ | $\overline{R1}$ | $\overline{R2}$ | $\overline{R3}$ | $\overline{R4}$ | $\overline{R5}$ | $\overline{R6}$ |
|---|---|---|---|---|---|---|
| 55 | 70 | 85 | 100 | 115 | 130 | 145 /us/ |

Tab.3

Rys.48 przedstawia część komputera zmieniającą się przy przejściu do pracy z 8 poziomami przerwania. Program, który zaspokaja powyższe wymagania jest następujący:

```
0000    LXI SP,   0800H   ; PIERWSZA
   1                      ; CZĘŚĆ
   2    JMP AD1           ; PROGRAMU A
   3
   4
   5                      ; PUSTE
   6                      ; PUSTE
   7                      ; PUSTE

0008    CMA               ; PIERWSZA
   9    OUT 01H           ; CZĘŚĆ
   A                      ; PROGRAMU B
   B    MV1 A,13H         ; KTÓRY
   C                      ; WYTWARZA
   D    JMP AD3           ; IMPULS
   E                      ; TRWAJĄCY
   F                      ; 145 MIKROSEKUND

0010    CMA               ; PIERWSZA
   1    OUT 01H           ; CZĘŚĆ
   2                      ; PROGRAMU B
   3    MV1 A, 11H        ; KTÓRY
   4                      ; WYTWARZA
   5    JMP AD3           ; IMPULS
   6                      ; TRWAJĄCY
   7                      ; 130 MIKROSEKUND

0018    CMA               ; PIERWSZA
   9    OUT 01H           ; CZĘŚĆ
   A                      ; PROGRAMU B
   B    MV1 A, ODH        ; KTÓRY
   C                      ; WYTWARZA
   D    JMP AD3           ; IMPULS
   E                      ; TRWAJĄCY
   F                      ; 115 MIKROSEKUND
```

```
0020    CMA             ; PIERWSZA
   1    OUT 01H         ; CZĘŚĆ
   2                    ; PROGRAMU B
   3    MVI A, 0BH      ; KTÓRY
   4                    ; WYTWARZA
   5    JMP AD3         ; IMPULS
   6                    ; TRWAJĄCY
   7                    ; 100 MIKROSEKUND

0028    CMA             ; PIERWSZA
   9    OUT 01H         ; CZĘŚĆ
   A                    ; PROGRAMU B
   B    MVI A, 09H      ; KTÓRY
   C                    ; WYTWARZA
   D    JMP AD3         ; IMPULS
   E                    ; TRWAJĄCY
   F                    ; 85 MIKROSEKUND

0030    CMA             ; PIERWSZA
   1    OUT 01H         ; CZĘŚĆ
   2                    ; PROGRAMU B
   3    MVI A, 07H      ; KTÓRY
   4                    ; WYTWARZA
   5    JMP AD3         ; IMPULS
   6                    ; TRWAJĄCY
   7                    ; 70 MIKROSEKUND
```

```
0040    AD1/: XRA A     ; CIĄG DALSZY
   1        OUT 01H     ; PROGRAMU A
   2        MVI A, 0FH
   3
   4        OUT 02H
   5
   6        EI
   7 AD2:   NOP
   8        JMP AD2
   9
   A

   B    AD3:    DCR A       ; CIĄG DALSZY
   C            JNZ AD3     ; PROGRAMU B
   D
   E
   F            NOP
0050            NOP
   1            OUT 01H
   2
   3            MVI A, 0FH
   4
   5            OUT 02H
   6
   7            EI
   8            RET
```

Program będzie bardziej zrozumiały po kilku słowach wprowadzenia

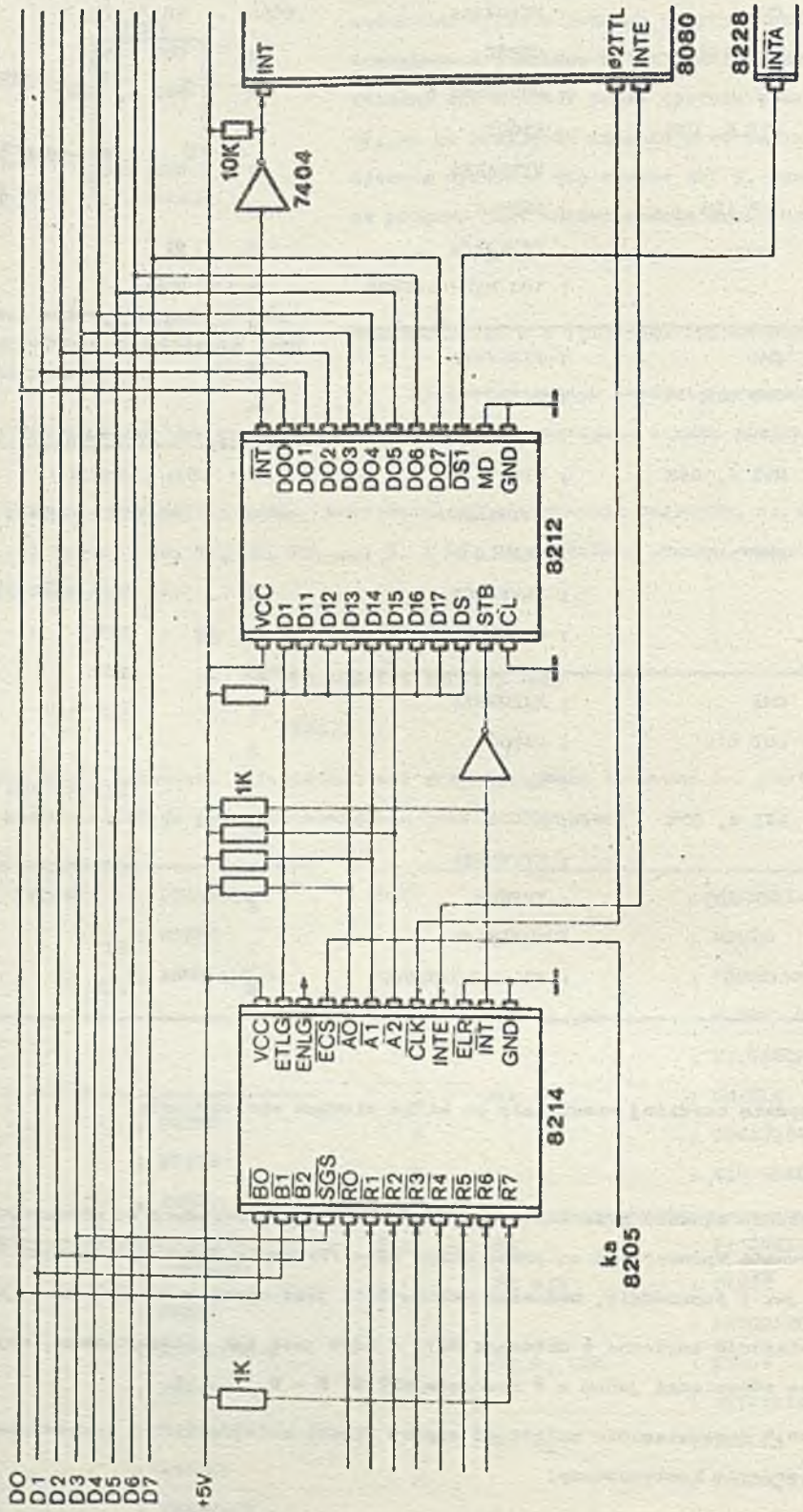
8214

Uprzednio żądanie przerwania wprowadzane było bezpośrednio do mikroprocesora. Obecnie żądania przerwania wprowadzane są przez układ 8214 /Priority Interrupt Control Unit, PICU/. I obecnie, tak jak i poprzednio, zadaniem układu 8212 jest sprzętowe wytwarzanie kodu dla rozkazu RST N.

Rozwiązanie logiczne z układami 8212 i 8214 jest tak skonstruowane, aby każdemu z 8 żądań przerwania odpowiadał jeden z 8 rozkazów RST N; N = 0, ..., 7.

W tab.4 przedstawiono zależność między liczbą kolejną źródła przerwania a adresem, od której jest program kontynuowany.

Przy łączeniu układów 8214 użytkownik ma określoną swobodę tylko w odniesieniu do końcówek ETLG, ENLG, $\overline{ELR}$ i $\overline{ECS}$.




Rys.48. Sposób łączenia układów 8212 i 8214 w mikrokomputerze z 8 poziomami przerwania

| N | Źródło | Adres |
|---|---|---|
| 7 | $\overline{R7}$ | 0000H |
| 6 | $\overline{R6}$ | 0008H |
| 5 | $\overline{R5}$ | 0010H |
| 4 | $\overline{R4}$ | 0018H |
| 3 | $\overline{R3}$ | 0020H |
| 2 | $\overline{R2}$ | 0028H |
| 1 | $\overline{R1}$ | 0030H |
| 0 | $\overline{R0}$ | 0038H |

Tab.4

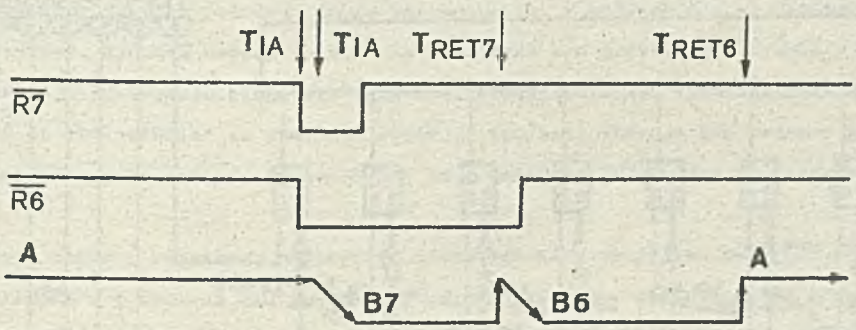
Końcówki ETLG /Enable This Logic Group/ i ENLG /Enable Next Logic Group/ służą do łączenia kilku układów 8214. W naszym przykładzie jest tylko jeden układ 8214. Wówczas końcówkę ETLG łączymy przez opornik o wartości 1K z +5 V, zaś końcówkę ENLG zostawiamy "wiszącą".

Niski poziom sygnału na wejściu $\overline{ELR}$ /Enable Level Read/ powoduje zakaz pojawiania się sygnałów na wyjściach A0, A1 oraz A2 z rys.48.

W naszym przykładzie nie wykorzystuje się tej możliwości i dlatego końcówka $\overline{ELR}$ połączona jest z masą.

Dla zrozumienia roli wejścia $\overline{ECS}$ /Enable Current Status/ potrzebne jest krótkie wprowadzenie.

Źródła przerwania mają różny priorytet. W naszym przykładzie najwyższy priorytet ma żądanie nadchodzące na wejście $\overline{R7}$. Jeżeli dwa źródła o różnym priorytecie jednocześnie zgłaszają żądanie przerwania, wówczas mikroprocesor odpowie na żądanie o wyższym priorytecie. Żądanie o niższym priorytecie, jeżeli nie będzie wycofane, zostanie przetworzone dopiero po zakończeniu przetwarzania przerwania o wyższym priorytecie /rys.49/.

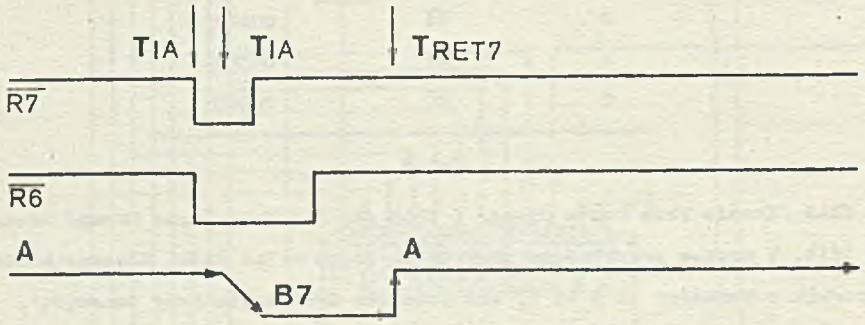



Rys.49. Sposób przetwarzania żądań przerwania, gdy dwa żądania przerwania o różnym priorytecie nadchodzą jednocześnie

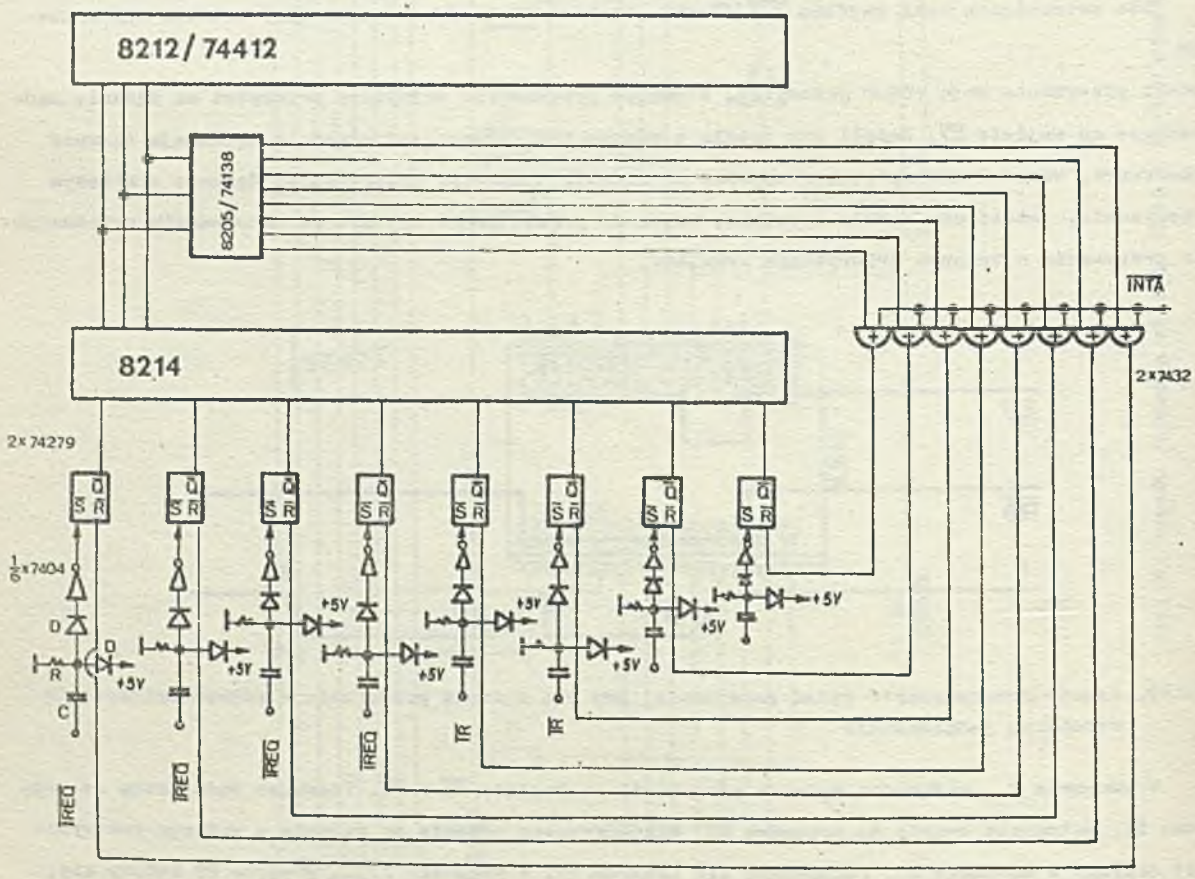
W momencie $T_{IR}$ nadchodzą żądania przerwania na wejścia $\overline{R6}$ i $\overline{R7}$. Pierwsze żąda skoku do programu B6, natomiast drugie do programu B7. Mikroprocesor odpowie na żądanie o wyższym priorytecie; dlatego w momencie $T_{IA}$ rozpocznie się program B7. W momencie $T_{RET7}$ program B7 kończy się, ale żądanie na wejściu $\overline{R6}$ jeszcze trwa i mikroprocesor natychmiast przechodzi do programu B6.

Gdyby źródło $\overline{R6}$ wycofało żądanie przerwania przed momentem $T_{RET7}$, wówczas nie doszłoby do skoku do programu B6 /rys.50/.

Istotne jest wycofanie żądania przerwania przez źródło $\overline{R7}$ przed momentem $T_{RET7}$, gdyż w przeciwnym razie dochodzi do fałszywego żądania przerwania. Zjawisku temu możemy zapobiec za pomocą układu z rys.51.

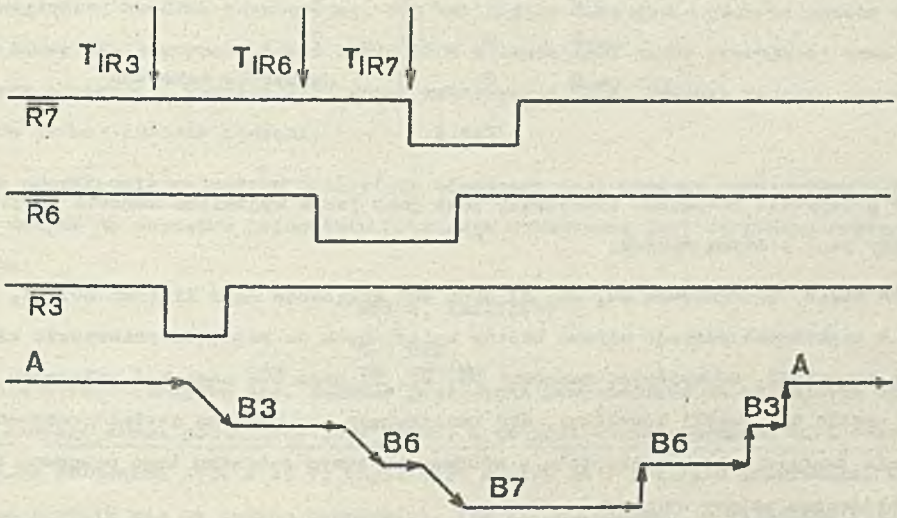



Rys.50. Sposób przetwarzania żądań przerwania, gdy dwa żądania przerwania o różnym priorytecie nadchodzą jednocześnie, jednak żądanie przerwania o niższym priorytecie zostaje wycofane przed zakończeniem przetwarzania żądania o wyższym priorytecie




Rys.51. Schemat logiczny układu, za pomocą którego eliminuje się możliwość podwójnego zadziałania tego samego żądania przerwania w systemie z 8 poziomami przerwania

Schemat logiczny przedstawiony na rys.51 jest tak skonstruowany, że żądanie przerwania, nie wycofane po przyjęciu, przestaje działać na w e j ś c i e układu 8214 w momencie, gdy poziom sygnału $\overline{INTA}$ zmieni się skokowo. Jak powiedzieliśmy już, w tym momencie kończy się wczytywanie kodu dla rozkazu RST N. Może się zdarzyć, że źródło o wyższym priorytecie zgłosi żądanie przerwania w momencie, w którym program już się realizuje /jako wynik żądania przerwania o niższym priorytecie/. Wówczas zachodzi sytuacja przedstawiona na rys.52. Tak więc program B3 będący skut-




Rys.52. Sposób przetworzenia żądania przerwania, gdy żądanie o wyższym priorytecie nadchodzi w momencie przetwarzania żądania o niższym priorytecie

kiem żądania przerwania o niższym priorytecie zostaje przerwany i będzie kontynuowany po zakończeniu programu B6, będącego wynikiem żądania przerwania o wyższym priorytecie. Są zastosowania, w których nie chcemy, aby pracujący program był przerwany w momencie nadejścia żądania przerwania wyższego priorytetu. Również istnieją takie zastosowania, w których nie ma zapotrzebowania na układy priorytetowe. Wówczas wszystkie żądania mają taki sam priorytet, to znaczy każde żądanie przerwania powoduje przerwanie bieżącego programu. W związku z tym istnieje możliwość, aby układowi 8214 "dać do zrozumienia" co powinien robić. W tym celu stosuje się rejestr priorytetu bieżącego /Current Status Register/. Znajduje się on w układzie 8214. Jest to rejestr czterobitowy /rys.53/.

Jeden z czterech bitów w rejestrze priorytetu bieżącego oznaczony jest jako $\overline{SGS}$ /Status Group Select/. Jeżeli $\overline{SGS}$ = 1 wówczas nie obowiązują priorytety, tzn. uważa się, że wszystkie żądania przerwania mają taki sam priorytet. Wtedy każde nowe żądanie przerwania powoduje przerwanie programu bieżącego.

Jeżeli $\overline{SGS}$ = 0, to system działa z priorytetami. Wówczas przerywanie programu bieżącego wywołane jest tylko przez takie żądania przerwania, których poziom jest wyższy od N1. Zawartość pozycji bitowych $\overline{B0}$, $\overline{B1}$ i $\overline{B2}$ jednoznacznie wskazuje na wielkość N1 /tab.5/.

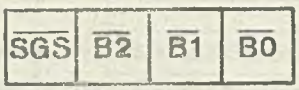
| $\overline{B2}$ | $\overline{B1}$ | $\overline{B0}$ | $\overline{N1}$ |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 2 |
| 1 | 0 | 1 | 3 |
| 1 | 0 | 0 | 4 |
| 0 | 1 | 1 | 5 |
| 0 | 1 | 0 | 6 |
| 0 | 0 | 1 | 7 |
| 0 | 0 | 0 | wszystko zakazane |

Tab.5.

Rejestr priorytetu bieżącego traktowany jest jako jedna wyjściowa komórka peryferyjna i dlatego połączony jest z szyną danych.

Na rysunku 48 widać, że końcówki $\overline{B0}$, $\overline{B1}$, $\overline{B2}$ oraz $\overline{SGS}$ połączone są z liniami D0, D1, D2 i D3 szyny danych, zaś 4 najstarsze pozycje bitowe bajtów wpisywanych do rejestru priorytetu bieżącego mogą mieć dowolną zawartość. Oczywiście, końcówki $\overline{B0}$, $\overline{B1}$, $\overline{B2}$ oraz $\overline{SGS}$ mogą być połączone z szyną danych w inny sposób np. jeżeli zezwalamy, aby realizujący się program został przerwany przez żądanie przerwania jakiegokolwiek priorytetu - wówczas na samym początku tego programu do rejestru priorytetu bieżącego należy wprowadzić daną

XXXX1XXX




Rys.53. Rejestr priorytetu bieżącego

Jeżeli zezwalamy, aby wykonywany program został przerwany tylko przez te żądania przerwania, których priorytet jest większy od 5, wówczas na samym początku tego programu należy do rejestru priorytetu bieżącego wprowadzić daną

XXXX0011

Jeżeli nie dopuszczamy do przerywania bieżącego programu - wówczas na samym początku tego programu w rejestr priorytetu bieżącego należy wprowadzić daną

XXXX0000

Taki sam rezultat osiągniemy również jeżeli nie odnowimy zawartości rejestru priorytetu bieżącego. Przyczyna tego jest następująca: po przyjęciu żądania przerwania mikroprocesor przechodzi do nowego programu. Jednocześnie dokonuje się zakazu przyjmowania wszystkich nowych żądań przerwania. Zakaz ten wynika z dwóch źródeł, tzn. procesor /8080/ oraz układ PICU /8214/ realizują ten zakaz. Procesor przeprowadza to w ten sposób, że ustawia w stanie pewien wewnętrzny przerzutnik.

Aby procesor po przyjęciu żądania przerwania odblokował zakaz przyjmowania nowych żądań przerwania należy, jak to już powiedziano, zrealizować rozkaz EI.

Aby układ PICU po przyjęciu żądania przerwania przestał blokować przyjmowanie nowych żądań przerwania - należy zmienić stan wspomnianego wewnętrznego przerzutnika, tzn. należy wpisać do rejestru priorytetu bieżącego jakąkolwiek zawartość, oprócz XXXX0000. Odblokowanie nie jest uwarunkowane wpisaniem nowej zawartości do rejestru priorytetu bieżącego, lecz wybieraniem rejestru

priorytetu bieżącego, poprzedzającym wpisanie nowej zawartości. Przy wybieraniu rejestru priorytetu bieżącego równocześnie zmienia się stan wspomnianego wewnętrznego przerzutnika.

W pierwszej kolejności należy odnowić zawartość rejestru priorytetu bieżącego, a dopiero potem zrealizować rozkaz EI. W przeciwnym wypadku istnieje możliwość nadejścia oraz przyjęcia nowego żądania przerwania - przed ustaleniem pożądanego stanu priorytetów. Wówczas może zaistnieć "chaos". Jak już wcześniej powiedzieliśmy, dla mikroprocesora rejestr priorytetu bieżącego jest tylko jedną wyjściową komórką peryferyjną. Jej wybieranie dokonywane jest za pomocą wejścia $\overline{ECS}$. Końcówkę $\overline{ECS}$ łączy się zazwyczaj przez układ 8205 z linią $\overline{I/OW}$ szyny sterującej oraz z niższymi 8 liniami szyny adresowej. Zamiast $\overline{I/OW}$ można wykorzystać $\overline{MEMW}$. Wówczas rejestr priorytetu bieżącego odpowiada jednej komórce pamięci.

W naszym przykładzie rejestrem priorytetu bieżącego jest komórka peryferyjna z adresem 02H. Wprowadzanie danych do rejestru priorytetu bieżącego dokonywane jest za pomocą następujących dwóch rozkazów:

```
MVI A, XXXXXXXXB
OUT 02H
```

Tutaj ze źródła $\overline{R7}$ nie korzysta się. Żądanie przerwania przychodzące na to wejście wymaga skoku do programu, którego adres początkowy jest 0000H, a to jest właśnie program A. Z tego powodu linia $\overline{R7}$ na rys.48 połączona jest z +5 V, czyli nasz system ma 7 poziomów przerwania. W łączeniu układu 8212 użytkownik nie ma żadnej dowolności. Wszystkie końcówki, z wyjątkiem D13, D14, D15, $\overline{INT}$ oraz STB, łączone są tak jak poprzednio.

Program

Program A jest obecnie umieszczony w obszarze między adresami 0000H a 0004H, a następnie między adresami 0040H a 004AH. Wynika to z tego, że program A nie jest dostatecznie krótki, aby mógł być w całości umieszczony w obszarze między adresami 0000H a 0007H. Jednakże oddziaływanie wejścia $\overline{RESIN}$ wymaga, aby początkowym adresem programu A było 0000H. Dlatego więc w tej komórce pamięci znajduje się pierwszy bajt rozkazu LXI SP, 0800H. W zasadzie program A ma taką samą postać, jak w systemie z jednym poziomem przerwania, z tym, że ma dwa dodatkowe rozkazy, za pomocą których tworzy się początkową zawartość rejestru priorytetu bieżącego /MVI, OUT/.

Obecnie mamy 8 programów B. Programy B podzielone są na dwie części. W pierwszej części wszystkich programów B tworzy się wysoki poziom sygnału na wyjściu Q /CMA, OUT/. Następnie tworzy się początkową wartość licznika rewersyjnego /MVI/ oraz dokonuje się skoku do drugiej części programu B /JMP/. Druga część jest wspólna dla wszystkich programów B.
W ramach tej części licznik liczy wstecz /DCR, JZ/. Gdy licznik dojdzie do zera, wówczas na wyjściu Q pojawia się niski poziom sygnału /OUT/.

Przez czas trwania stanu niestabilnego żadne nowe żądanie przerwania nie zostanie przyjęte. Mianowicie na początku programu B zawartość rejestru priorytetu bieżącego nie podlega odnowieniu. Jednak na końcu programu B dochodzi do odnowienia zawartości rejestru priorytetu bieżącego /MVI, OUT, EI/. W ten sposób umożliwia się przyjmowanie nowych żądań przerwania po zakończeniu stanu niestabilnego.

MIKROKOMPUTER Z PONAD 8 POZIOMAMI PRZERWANIA

W niektórych zastosowaniach zachodzi potrzeba większej niż 8 liczby źródeł przerwania, np. przy projektowaniu automatycznej centrali telefonicznej. Wówczas układy 8214 łączy się w łańcuch /Daisy Chain/.

Łańcuch układów 8214

Na rys.54 przedstawiony jest łańcuch składający się z dwóch układów 8214 i umożliwiający pracę z 16 poziomami przerwania. Gdy układy połączone są w łańcuch, wówczas obsługa przerwania w istotny sposób różni się od dotychczas poznanego. Oczywiście, również obecnie wejścia różnią się pod względem priorytetu. Na rys.54 najwyższy priorytet ma wejście $\overline{R15}$.

Różne źródła przerwania zazwyczaj wymagają, aby mikroprocesor wykonywał różnorodne operacje, tzn. po przyjęciu żądania przerwania mikroprocesor przechodzi do jednego z 16 odcinków programowych. Adres odcinka programowego również obecnie jest pobierany z układu 8212, jednak w sposób, jak to już podkreślano, różniący się od dotychczas przedstawionego. Po przyjęciu żądania przerwania automatycznie wykonywany jest rozkaz RST7, bez względu na to, które ze źródeł zgłosiło żądanie przerwania. Oznacza to, że po przyjęciu żądania przerwania program b e z w a r u n k o w o dokonuje skoku do adresu 0038H. A więc układ 8212 nie służy już do sprzętowego wytwarzania kodu dla rozkazu RST N.

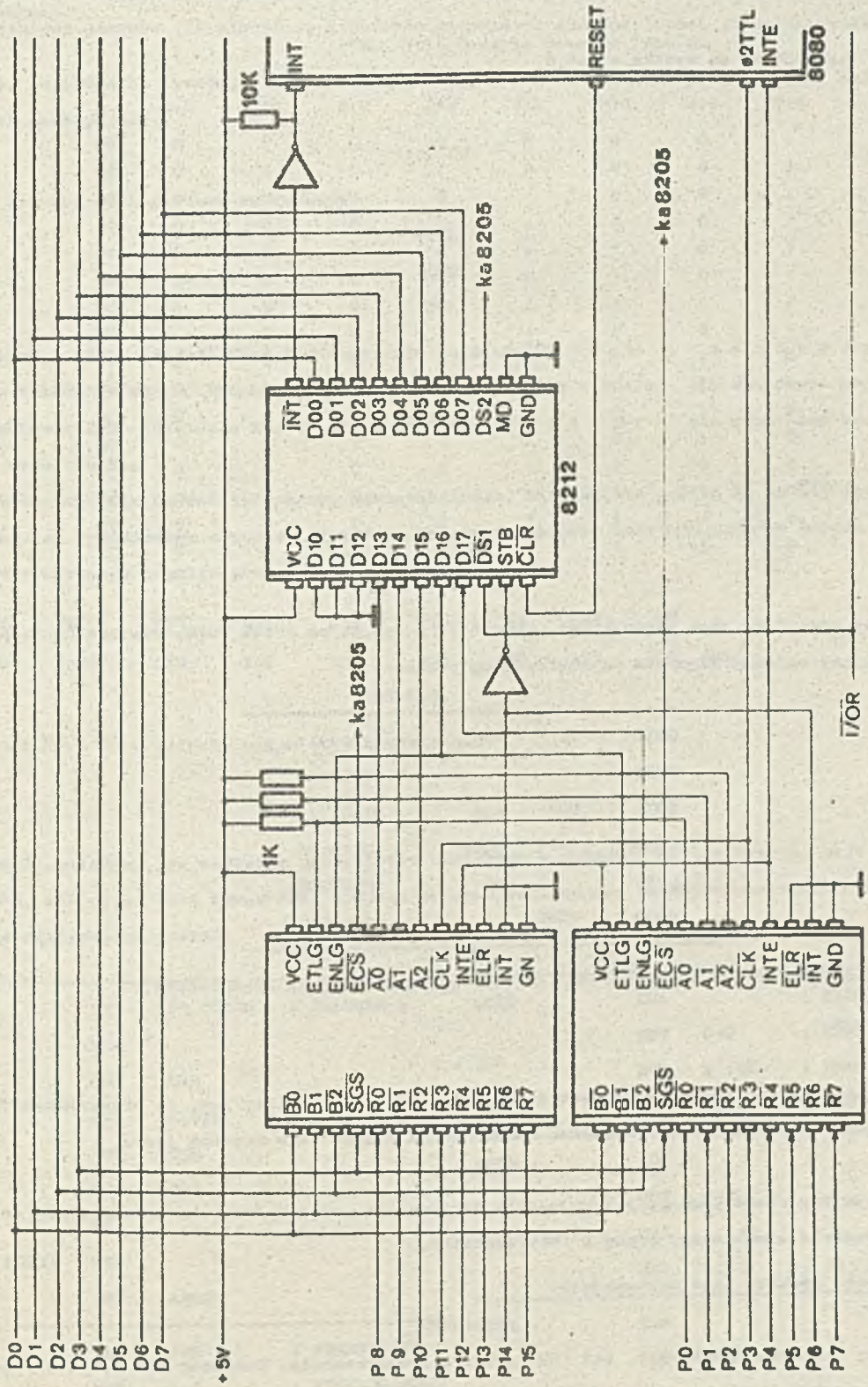
W momencie, gdy mikroprocesor przyjmie żądanie przerwania, na wejściach układu 8212 pojawi się zawartość jednoznacznie definiująca źródło, z którego nadeszło żądanie. W tab.6 podano uzależnienia między źródłami zgłaszającymi żądanie przerwania, a zawartością pojawiającą się na wejściach układu 8212. Tab.6 obowiązuje tylko, gdy układy 8214 oraz układ 8212 połączone są w sposób przedstawiony na rys.54.

Tak więc zawartością układu 8212 jest jedna dwucyfrowa liczba heksadecymalna. Ta liczba heksadecymalna stanowi mniej znaczącą połowę adresu początkowego odcinka programowego, który następuje po przyjęciu żądania przerwania. Bardziej znacząca połowa adresu początkowego wybierana jest dowolnie. My zdecydujemy się na 00H, chociaż moglibyśmy zdecydować się również na 01H, 02H lub 03H, ponieważ z pamięcią ROM sprzęgnięte są adresy od 0000H do 03FFH.

Aby mikroprocesor przeszedł do wykonywania odpowiedniego odcinka programowego, zawartość układu 8212 należy przesłać jako młodszy bajt do rejestru programu. W starszym bajcie rejestru programu należy umieścić 00H. Tak więc począwszy od adresu 0038H musi się znajdować odcinek programu, za pomocą którego zawartość układu 8212 przenosi się do młodszego bajtu rejestru programu, a 00H umieszcza się w starszym bajcie rejestru programu.

Bezpośrednie przenoszenie zawartości układu 8212 do rejestru programu jest niemożliwe, nie istnieje mianowicie rozkaz, za pomocą którego przenosi się zawartość komórki peryferyjnej do mniej znaczącej połowy rejestru programu oraz umieszcza się określoną daną w bardziej znaczącej połowie rejestru programu. Dlatego też zawartość rejestru programu tworzona jest bezpośrednio.

W pierwszej kolejności zawartość komórki peryferyjnej przenosi się do rejestru L. Mianowicie




Rys.54. Sposób łączenia układów 8212 i 8214 w mikrokomputerze z 16 poziomami przerwania

rejestr L stanowi młodszy bajt 16-bitowej danej w parze rejestrowej HL. Następnie wprowadza się daną 00H do rejestru H. Jeżeli żądanie przerwania nadchodzi ze źródła R5, wówczas zawartość rejestru HL jest 0030H, co wynika z tab.6.

| | D17 | D16 | D15 | D14 | D13 | D12 | D11 | D10 | H |
|---|---|---|---|---|---|---|---|---|---|
| $\overline{R15}$ | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 80 |
| $\overline{R14}$ | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 88 |
| $\overline{R13}$ | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 90 |
| $\overline{R12}$ | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 98 |
| $\overline{R11}$ | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | A0 |
| $\overline{R10}$ | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | A8 |
| $\overline{R9}$ | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | B0 |
| $\overline{R8}$ | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | B8 |
| $\overline{R7}$ | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 40 |
| $\overline{R6}$ | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 48 |
| $\overline{R5}$ | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 50 |
| $\overline{R4}$ | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 58 |
| $\overline{R3}$ | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 60 |
| $\overline{R2}$ | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 68 |
| $\overline{R1}$ | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 70 |
| $\overline{R0}$ | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 78 |

Tab.6.

Jeżeli spełniony jest warunek, że układ 8212 z rys.54 ma adres 002H, wówczas odcinek programu realizujący opisany algorytm ma następującą postać:

| | | | |
|---|---|---|---|
| 0038 | IN | 02H | 11011011 |
| 0039 | | | 00000010 |
| 003A | MOV | L,A | 01101111 |
| 003B | MOI | H,00H | 10110100 |
| 003C | | | 00000000 |
| 003D | PCHL | | 11101001 |

Za pomocą rozkazu IN 02H zawartość układu 8212 wprowadza się do akumulatora.

Rozkazem

MOV L,A

zawartość akumulatora przenoszona jest do rejestru L. Wspomnieliśmy już, że przesyłanie danych między rejestrami oznacza się za pomocą MOVE. Ogólna postać tego rozkazu jest:

MOV D,S

Rozkazem tym zawartość jednego z 7 rejestrów oznaczonego przez S /Source/ przenosi się do jednego z 7 rejestrów oznaczonego przez D /Destination/.

Kod tego rozkazu jest następujący:

01DDDSSS

SSS dotyczy rejestru źródłowego, zaś DDD rejestru przeznaczenia. Rozkazem MVI H,00H w rejestrze umieszcza się 00H.

Rozkazem

PCHL /load Program Counter from HL/

zawartość rejestru HL zostaje p r z e n i e s i o n a do rejestru programowego. Kod tego rozkazu jest następujący

11101001

Do rozkazu PCHL podobne są rozkazy

SPHL

XCHG

oraz XTHL

Rozkazem SPHL /load SP from HL/ zawartość rejestru HL zostaje p r z e n i e s i o n a do rejestru wskaźnika stosu. Rozkazem XCHG /eXCHanGe/ w y m i e n i a się zawartość rejestrów DE i HL. Rozkazem XTHL /eXchange Top of stack with GL/ w y m i e n i a się zawartość szczytu stosu oraz rejestru HL.

Wrócimy obecnie do multiwibratora monostabilnego. Załóżmy, że źródła $\overline{R7}$ do $\overline{R15}$ żądają realizacji impulsu wyjściowego o różnej długości. Zależności między kolejnym numerem źródła a szerokością wytwarzanego impulsu przedstawiono w tab.7.

| $\overline{R15}$ | $\overline{R14}$ | $\overline{R13}$ | $\overline{R12}$ | $\overline{R11}$ | $\overline{R10}$ | $\overline{R9}$ | $\overline{R8}$ | $\overline{R7}$ | |
|---|---|---|---|---|---|---|---|---|---|
| 280 | 265 | 250 | 232 | 220 | 205 | 190 | 175 | 160 | /µs/ |

Tab.7.

Dla źródeł $\overline{R0}$ i $\overline{R6}$ w dalszym ciągu obowiązuje tab.3.

Program

Jeżeli założymy, że rejestry priorytetów bieżących w układach 8214 z rys.54, mają adresy 03H i 04H, zaś układ 8212 adres FFH, wówczas program realizujący określony multiwibrator monostabilny ma następującą postać:

```
0000        LXI   SP,0800H  ; PROGRAM A
            XRA
            OUT   01H
            MVI   A,0FH
            OUT   03H
            OUT   04H
            EI
      ADR2: NOP
            JMP   ADR2

0038        IN    FFH       ; FORMOWANIE
            MOV   L,A       ; POCZĄTKOWEGO
            MVI   H,00H     ; ADRESU
            PCHL            ; PROGRAMU B

0040        CMA             ; PIERWSZA
            OUT   01H       ; CZĘŚĆ
            MVI   A,13H     ; PROGRAMU B
            JMP   ADR3      ; 160 MIKROSEKUND

0048        CMA
            OUT   01H
            MVI   A,11H
            JMP   ADR3      ; 145 MIKROSEKUND

0050        CMA
            OUT   01H
            MVI   A,0FH
            JMP   ADR3      ; 130 MIKROSEKUND
```

```
0058    CMA
        OUT     01H
        MVI     A,0DH
        JMP     ADR3    ; 115 MIKROSEKUND

0060    CMA
        OUT     01H
        MVI     A,0BH
        JMP     ADR3    ; 100 MIKROSEKUND

0068    CMA
        OUT     01H
        MVI     A,09H
        JMP     ADR3    ; 85 MIKROSEKUND

0070    CMA
        OUT     01H
        MVI     A,07H
        JMP     ADR3    ; 70 MIKROSEKUND

0078    CMA
        OUT     01H
        MVI     A,05H
        JMP     ADR3    ; 55 MIKROSEKUND

0080    CMA
        OUT     01H
        MVI     A,23H
        JMP     ADR3    ; 280 MIKROSEKUND

0088    CMA
        OUT     01H
        MVI     A,21H
        JMP     ADR3    ; 265 MIKROSEKUND

0090    CMA
        OUT     01H
        MVI     A,1FH
        JMP     ADR3    ; 250 MIKROSEKUND

0098    CMA
        OUT     01H
        MVI     A,1DH
        JMP     ADR3    ; 235 MIKROSEKUND

00A0    CMA
        OUT     01H
        MVI     A,1BH
        JMP     ADR3    ; 220 MIKROSEKUND

00A8    CMA
        OUT     01H
        MVI     A,19H
        JMP     ADR3    ; 205 MIKROSEKUND

00B0    CMA
        OUT     01H
        MVI     A,17H
        JMP     ADR3    ; 190 MIKROSEKUND

00C0    ADR3:   DCR     A       ; CIĄG DALSZY
                JNZ     ADR3    ; PROGRAMU B
                NOP
                NOP
                OUT     01H
                MVI     A,FFH
                out     03H
                OUT     04H
                EI
                RET
```

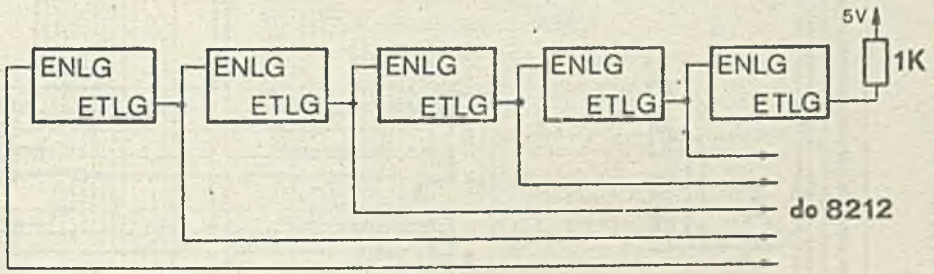
Program A ma taką samą postać jak uprzednio, jednak teraz tworzy się dwa rejestry priorytetu bieżącego.

Obecnie istnieje 16 programów B i ich postać jest taka sama jak w poprzednim przykładzie, z takim wyjątkiem, że na końcu wspólnej części dla wszystkich programów B odnawia się zawartość dwóch rejestrów priorytetu bieżącego. Jedyną nowością jest odcinek programowy służący do formowania adresu początkowego.

MIKROKOMPUTER Z 40 POZIOMAMI PRZERWANIA

Układy 8214 oraz układ 8212 z rys.54 połączone są w taki sposób, że wejścia D13, D14 i D15 jednoznacznie definiują numer kolejny źródła, z którego nadchodzi żądanie przerwania, mianowicie, końcówki D13, D14 i D15 połączone są z końcówkami A0, A1, A2 obu układów 8214. Wejścia D16 i D17 jednoznacznie definiują układ 8214, przez który nadchodzi żądanie przerwania. Końcówki te połączone są z końcówkami ENLG obu układów 8214. Do zdefiniowania układu, przez który nadchodzi żądanie przerwania mogą służyć również wejścia D10, D11 i D12. Oznacza to, że w łańcuch może być połączonych maksimum 5 układów 8214, czyli, że mikrokomputer może zawierać maksimum 40 poziomów przerwania.

Jak powiedzieliśmy, do łączenia większej liczby układów 8214 służą końcówki ENLG /Enable Next Level Group/ i ETLG /Enable This Level Group/. Sposób łączenia układów 8214 w łańcuch przedstawiony jest na rys.55.




Rys.55. Sposób łączenia układów 8214 w łańcuch

Na wyjściach ENLG wszystkich układów 8214 w łańcuchu jest wysoki poziom sygnału. Na wyjściu ENLG tego układu 8214, przez który nadeszło żądanie przerwania znajduje się niski poziom sygnału. A więc, po przyjęciu żądania przerwania, na wyjściach ENLG tworzy się jeden z 5 stanów przedstawionych w tab.8.

11110

11100

11000

10000

00000

Tab.8

Oznacza to, że stan na wyjściach ENLG jednoznacznie definiuje układ 8214, przez który nadeszło żądanie przerwania i dlatego końcówki ENLG łączy się z końcówkami D10, D11, D12, D16 i D17 w układzie 8212. Końcówki D13, D14 i D15 nadal łączy się z końcówkami A0, A1 i A2. Dlatego też po przyjęciu żądania przerwania na wejściu do układu 8212 pojawia się jedna z 40 różnych zawartości.

MIKROKOMPUTER Z PONAD 40 POZIOMAMI PRZERWANIA

Istnieje kilka sposobów tworzenia systemu z ponad 40 poziomami przerwania.

Pierwszy sposób dotyczy kombinacji kilku łańcuchów obejmujących po 5 układów 8212.

Drugi sposób to sprzętowe generowanie kodu rozkazu CALL ADRM, co omówimy w dalszej części pracy.

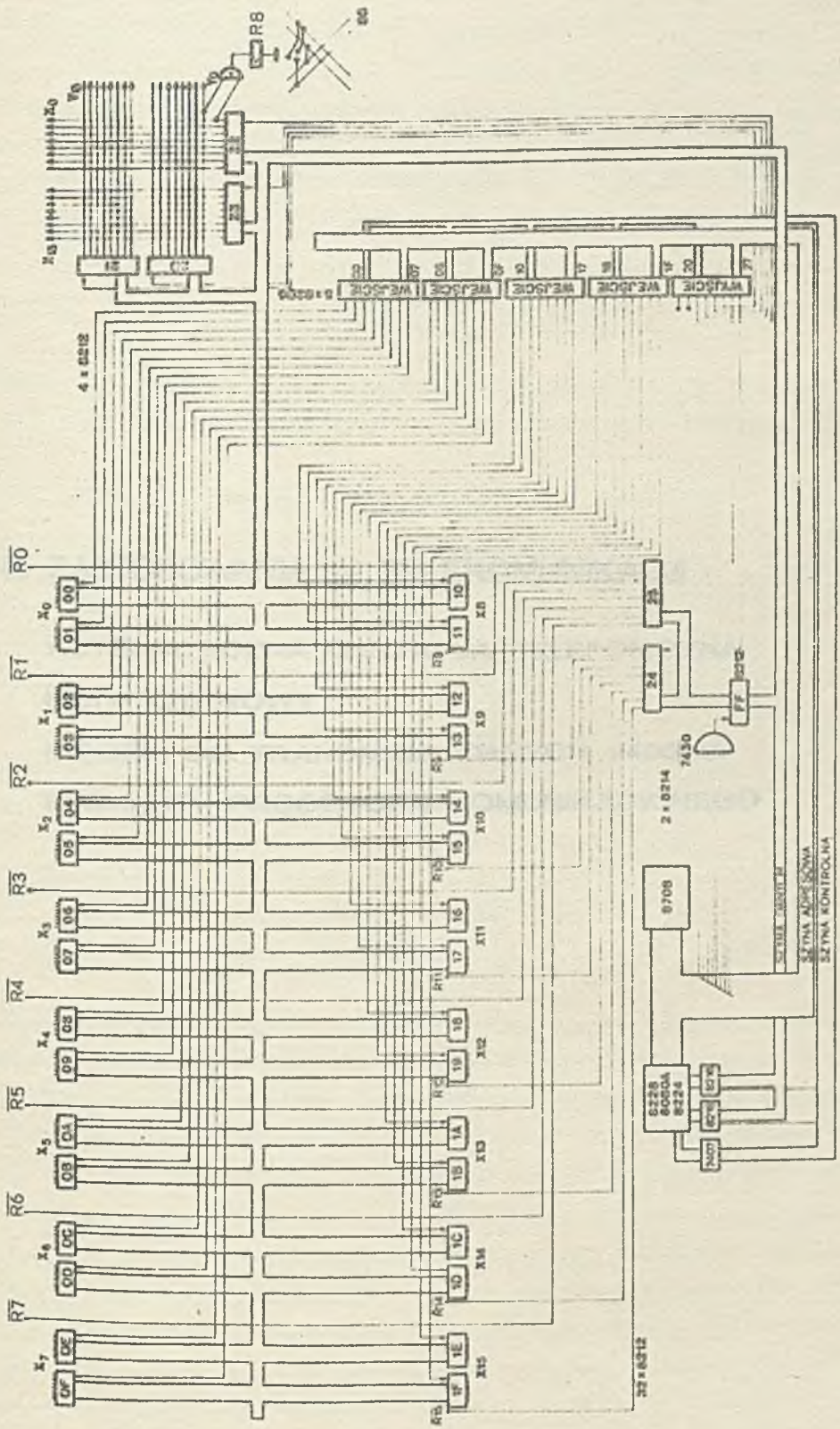
Trzeci sposób polega na zastosowaniu układu 8259 /Programmable Interrupt Controller/, umożliwiającego bezpośrednio utworzenie systemu z 64 poziomami przerwania.

Czwarty sposób dotyczy programowej obsługi przerwania /Polling Interrupt/. W tym wypadku, w określonej kolejności, testuje się kolejno stan wszystkich jednostek peryferyjnych, przez które może wpłynąć żądanie przerwania. Jest to powolny sposób i dlatego jest rzadko stosowany.

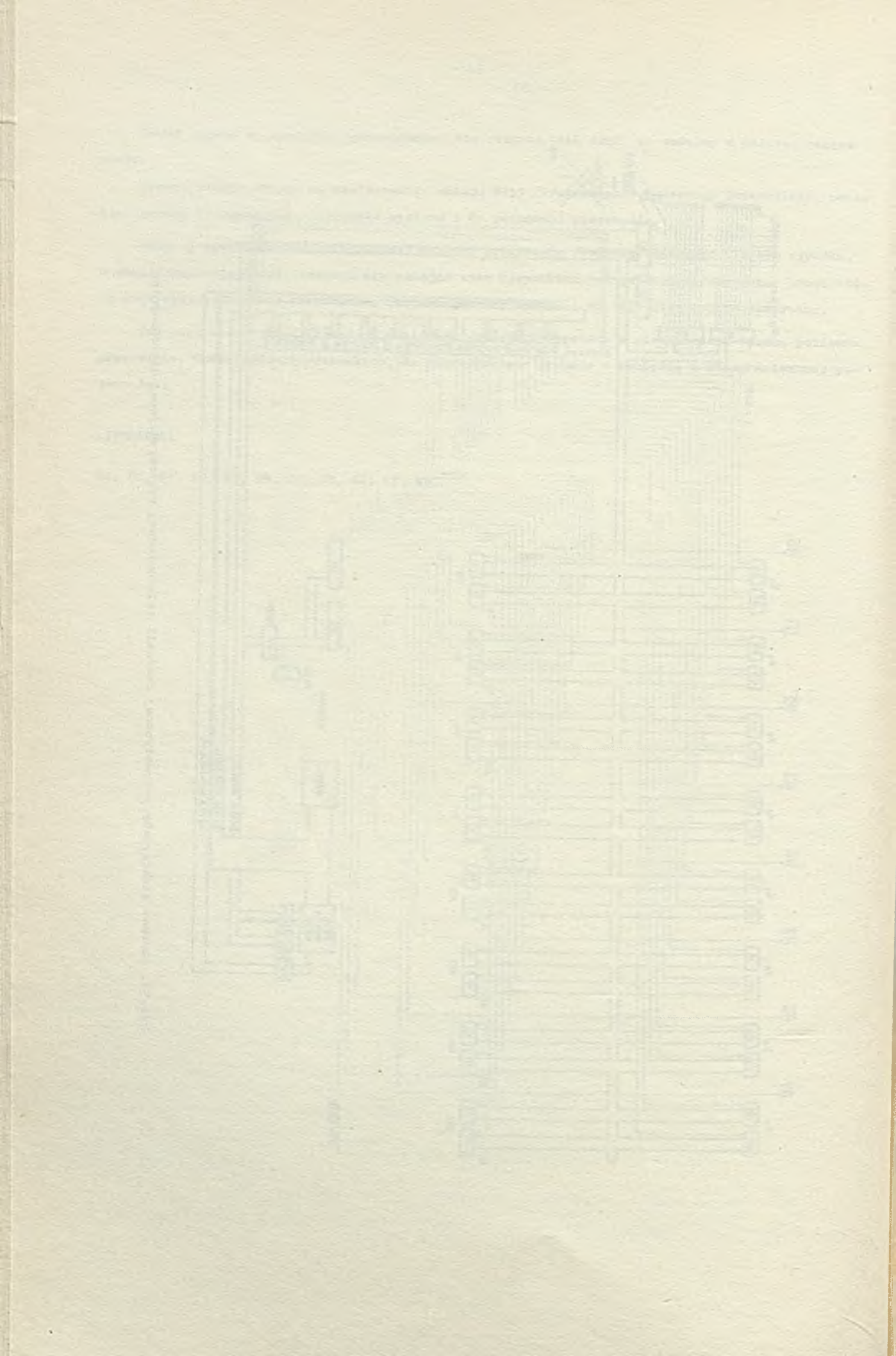
Tak więc teoretycznie istnieje możliwość utworzenia systemu z nieskończoną liczbą poziomów przerwania. Oczywiście pod warunkiem, że mikroprocesor pracuje z pamięcią o nieograniczonej pojemności.

LITERATURA

C1, C2, D1, D2, D3, D4, F1, F2, G6; K1, K2.




Rys.56. Schemat elektryczny automatycznej centrali telefonicznej wykorzystującej mikrokomputer



## CZ. III. ZASTOSOWANIA W TELEKOMUNIKACJI

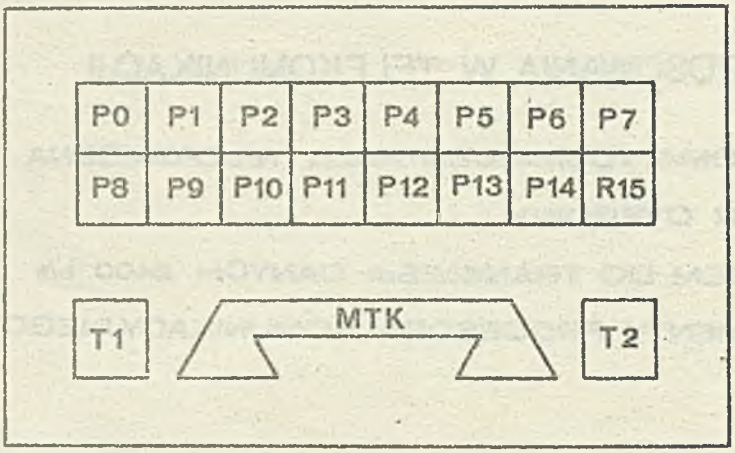
## AUTOMATYCZNA CENTRALA TELEFONICZNA

Na rys.56 przedstawiono schemat elektryczny automatycznej centrali telefonicznej, która jest przedmiotem naszych rozważań.

Zajmujemy się specyficzną, uproszczoną formą automatycznej centrali telefonicznej. Brakuje systemu wybierania oraz wielu innych funkcji, bez których nie można sobie wyobrazić prawdziwej automatycznej centrali telefonicznej. Jednak przykład ten zawiera większość elementów niezbędnych przy projektowaniu urządzeń handlowych /seryjnych/ tego typu wykorzystujących procesor.

Liczba abonentów tej centrali wynosi 16. Liczba linii zewnętrznych wynosi również 16. Przewiduje się, że każdy abonent może połączyć się przez dowolną linię, ale abonenci nie mogą łączyć się ze sobą.

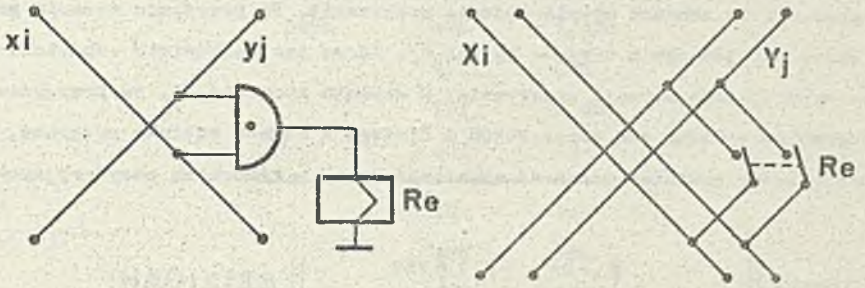
Każdy z abonentów posiada aparat telefoniczny i tablicę sterowniczą. Tablica sterownicza z dwoma przyciskami /T1 i T2/ oraz szesnastoma wyłącznikami /P0, ..., P15/ przedstawiona jest na rys.57.




Rys.57. Tablica sterownicza abonenta

Nawiązując połączenie z jedną określoną linią abonent wciska jeden z 16 wyłączników, odpowiadający tejże linii, a następnie przyciska przycisk T1. Przyciskając odpowiedni wyłącznik abonent definiuje linię, z którą pragnie się połączyć. Przyciskając natomiast przycisk T1 abonent "zawiadamia" mikrokomputer, że chce uzyskać połączenie. Po uzyskaniu połączenia abonent wciskając przycisk T2 wysyła wezwanie do swojego rozmówcy na drugim końcu linii.

W górnym prawym rogu rys.56 znajduje się matryca przełącznikowa. Przepływ przez matrycę przełącznikową może być dwu- lub czteroprzewodowy. Załóżmy, że przesyłanie przez matrycę przełącznikową z rys.56 jest dwuprzewodowe. Jeden węzeł takiej matrycy przełącznikowej przedstawiony jest na rys.58. Jeden z 256 takich węzłów jest również wyodrębniony na rys.56.




Rys.58. Węzeł matrycy przełącznikowej

Przewody $x_i$ /i = 0,..., 15/ i $y_j$ /j = 0,..., 15/ służą do tworzenia połączenia między parami telefonicznymi $X_i$ / i = 0,...,15/ oraz $Y_j$ /j = 0,..., 15/. Para $X_i$ prowadzi do abonenta, natomiast para $Y_j$ do linii.

Przekaźnik /Re/ z rys.58 uaktywnia się dopiero, gdy na obu przewodach $x_i$ i $y_j$ wystąpi wysoki poziom sygnału. Gdy przekaźnik działa, styki przekaźnika zamykają się. W ten sposób tworzy się połączenie, w tym wypadku dwuprzewodowe, między abonentem $X_i$ i linią $Y_j$. Przewody $x_i$ /i = 0,... ,15/ oraz przewody $y_j$ /j = 0,...,15/ połączone są z końcówkami wyjściowymi 4 jednostek peryferyjnych typu 8212. Adresy komórek peryferyjnych w tych jednostkach są 22H i 23H, oraz 20H i 21H /rys.56/. Adresy te dobrane są dowolnie.

Uzyskanie połączenia zilustrujemy na przykładzie abonenta $X_5$ i linii $Y_0$. Aby takie połączenie uzyskać, należy w cztery wspomniane komórki peryferyjne wpisać następujące zawartości:

01H = 00000001B do komórki peryferyjnej z adresem 20H

00H = 00000000B do komórki peryferyjnej z adresem 21H

20H = 00100000B do komórki peryferyjnej z adresem 22H

00H = 00000000B do komórki peryferyjnej z adresem 23H

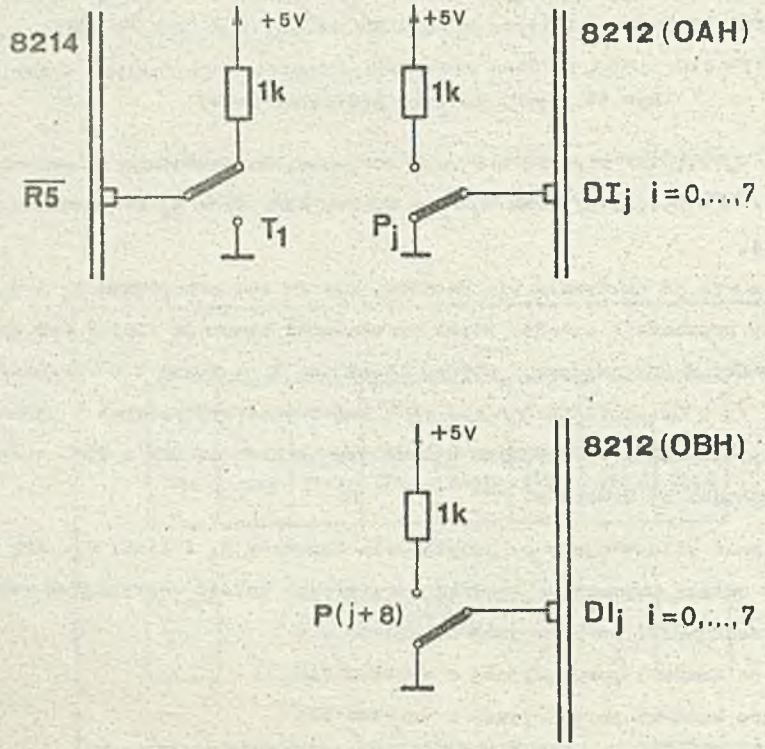
Jedynki w komórkach peryferyjnych odpowiadają pozycjom $X_5$ i $Y_0$. Wróćmy obecnie do tablicy sterowniczej.

Na tablicy sterowniczej każdego abonenta znajdują się dwie wejściowe jednostki peryferyjne typu 8212 oraz zacisk przewodu, przez który wysyła się żądanie obsługi przerwania. Adresy komórek peryferyjnych w tych jednostkach peryferyjnych, jak i poziomy priorytetów przerwania odpowiednich przewodów oznaczone są na rys.56. Np. dla abonenta $X_5$ adresami komórek peryferyjnych w przyłączonych jednostkach peryferyjnych są 0AH i 0BH. Abonentowi $X_5$ przyznane jest żądanie obsługi przerwania, którego poziom priorytetu wynosi 5 /przewód R5/.

Na rys.59 przedstawiono sposób połączenia wyłączników na tablicy sterowniczej abonenta $X_5$ /P0,...,P15/ z wyjściowymi końcówkami dwóch przyłączonych układów 8212 oraz sposób połączenia przycisku T1 z przewodem R5. Naciskając na jeden z 16 wyłączników $P_i$ /i = 0,...,15/ abonent tworzy na wejściach swoich dwóch jednostek peryferyjnych właśnie taki stan, jaki należy stworzyć w komórkach peryferyjnych z adresami 20H i 21H. Oznacza to, że zadanie mikroprocesora jest proste. Gdy abonent $X_5$ uzyskuje połączenie, wówczas tylko należy odpowiednio przerzucić zawartość komórek peryferyjnych z adresami 0AH i 0BH do komórek peryferyjnych z adresami 20H i 21H. Należy tak-

że umieścić zawartość 0020H w komórkach peryferyjnych z adresami 22H i 23H.

Naciskając na przycisk T1 abonent wysyła żądanie przerwania. Po przyjęciu żądania przerwania, program dokonuje skoku /do jednego z adresów wg tab.6/. Adres ten całkowicie odpowiada kolejnemu numerowi abonenta wysyłającego żądanie przerwania. W wypadku abonenta $X_5$, po przyjęciu żądania przerwania, program przechodzi pod adres 0050H i dlatego w ramach odcinka programu, jaki wówczas występuje, mikroprocesor powinien umieścić zawartość 0020H w komórkach peryferyjnych z adresami 22H i 23H.




Rys.59. Sposób połączenia przycisków i wyłączników z tablicy sterowniczej abonenta z układami 8212 i 8214

Krótko mówiąc, opierając się na tym, kto wysyła żądanie przerwania, mikroprocesor "wie" kto chce uzyskać połączenie.

PROGRAM

Odcinek programowy, który następuje po przyjęciu żądania przerwania wykonuje dwie operacje.

Po pierwsze, w komórkach peryferyjnych z adresami 22H i 23H umieszcza określoną liczbę 16-bitową. Liczba ta ma jedynkę tylko na tej pozycji bitowej, która odpowiada numerowi abonenta.

Po drugie, do komórek peryferyjnych z adresami 20H i 21H przenosi zawartość komórek adresowych znajdujących się w końcówce abonenta.

Odcinek programu, realizowany przy nawiązywaniu połączenia przez abonenta $X_5$, tzn. po przyciśnięciu przez abonenta $X_5$ na jego tablicy sterowniczej wyłącznika P0 i przycisku T1, ma następującą postać:

| | | |
|---|---|---|
| 0050 | MVI | A,02H |
| | OUT | 22H |
| | MVI | A,00H |
| | OUT | 23H |
| | IN | 0AH |
| | OUT | 20H |
| | IN | 0BH |
| | OUT | 21H |
| | MVI | A,FFH |
| | OUT | 24H |
| | OUT | 25H |
| | EI | |
| | RET | |

Ten odcinek programu odpowiada jednemu z 16 programów B z rozdziału o multiwibratorze monostabilnym z 16 poziomami przerwania. Każdy z abonentów ma przydzielony odpowiedni odcinek programu, który jest realizowany po przyciśnięciu przez danego abonenta przycisku T1 na swojej tablicy sterowniczej. Początkowe adresy tych odcinków programu podane są w tab.6. Powyższy odcinek programu jest za długi, aby mógł zmieścić się w obszarze między dwoma początkowymi adresami z tab.6. Dla uproszczenia wyjaśnień przedstawiony odcinek programu nie został "rozerwany" na dwie części.

Pierwsze cztery rozkazy realizują pierwszą operację. Następne cztery rozkazy realizują drugą operację. Dalsze cztery rozkazy odnawiają zawartości rejestrów priorytetu bieżącego, znajdujące się w układach 8214. Ostatni rozkaz realizuje powrót do programu A. Program A oraz odcinek programu tworzący zawartość rejestru programu mają taką samą postać, jak w multiwibratorze monostabilnym z 16 poziomami przerwania.

MIKROKOMPUTER

Układy 8080A, 8228, 8224 i 8708 są między sobą połączone jak na rys.9. Układy 8214 /adresy 24H i 25H/ oraz układ 8212 /adres FFH/ połączone są jak na rys.54.

W mikrokomputerze z rys.56 istnieją 33 wejściowe komórki peryferyjne i 6 wyjściowych komórek peryferyjnych. Przyjęto, że rejestry priorytetów bieżących traktowane są jako komórki peryferyjne, a nie komórki pamięci. Wybieranie jednostek peryferyjnych dokonywane jest za pomocą 5 dekoderów 8205 i jednego 8-wejściowego układu $\overline{7430}$. Górne 4 dekodery z rys.56 połączone są jak na rys. 34. Razem z 8-wejściowym obwodem I dekodery te służą do wybierania 33 wejściowych jednostek peryferyjnych. Najniższy dekoder z rys.56 połączony jest jak na rys.33. Służy on do wybierania 6 wyjściowych jednostek peryferyjnych. Dwa układy 8216 służą do zwiększenia współczynnika rozgałęzienia szyny danych. Połączone są jak na rys.35. Zapotrzebowanie na te układy wynika z faktu, że układ 8228 zapewnia tylko 10 mA na wyjściach DB0 do DB7, jeden układ 8212 pobiera w najgorszym razie 0.25 mA, zaś w mikrokomputerze z rys.56 jest 37 takich układów.

Układ 7407 służy do zwiększenia współczynnika rozgałęzienia na liniach $\overline{I/OR}$ i $\overline{I/OW}$ szyny sterującej.

UWAGI

Scharakteryzowana tu automatyczna centrala telefoniczna ma wiele wad; chcieliśmy w możliwie najprostszy sposób przedstawić istotę projektowania automatycznej centrali telefonicznej.

Po pierwsze, abonent może "wejść" na linię bez względu na to, czy jest ona wolna, czy też nie. W praktyce zezwala się na to tylko pewnej ograniczonej liczbie abonentów. Przez zmianę programu B można zabronić określonej liczbie abonentów "wejścia" na zajętą linię.

Po drugie, przy wejściu na łącze abonent rozłącza wszystkie pozostałe połączenia. Również ta niedogodność może być usunięta programowo, tzn. bez dodawania nowych składników.

Po trzecie, przyciski T1 i T2 nie są niezbędne. Przez skomplikowanie końcówki abonenckiej można osiągnąć taką sytuację, że wysyłanie żądań przerwania oraz wezwań dokonywane jest jednocześnie z wyborem rozmówcy.

Po czwarte, nie przedstawiono jak dokonuje się rozłączenia połączenia. W tym celu należałoby dodać jeszcze dwa układy 8214, przez które możnaby przesyłać żądanie rozłączenia połączenia.

Liczba przewodów na zewnętrznych zaciskach automatycznej centrali telefonicznej nie jest wielka. W naszym przykładzie jest ich tylko 56. Z tego 8 przewodów należy do szyny danych, 16 przewodów służy do wysyłania żądań przerwania, zaś 32 przewody służą do wybierania jednostek peryferyjnych. Skomplikowanie budowy końcówki abonenckiej, wynikające z głębszego poznania elementów mikrokomputera, może zmniejszyć liczbę przewodów.

Na zakończenie dokonajmy podsumowania. Są trzy podstawowe zalety projektowania automatycznej centrali telefonicznej z wykorzystaniem mikroprocesorów. Po pierwsze, zmniejsza się znacznie liczbę przewodów w kablu łączącym centralę z abonentami. Po drugie, system jest bardzo elastyczny. Przez zmianę programu można pewnej liczbie abonentów zabronić "wychodzenia" na jedną z linii, można zmieniać konfigurację systemu itd. Po trzecie, cena jednostki sterującej jest obecnie znacznie niższa.

Dysponując wiedzą dotychczas przedstawioną można automatyczną centralę telefoniczną z rys.56 rozszerzyć na 40 abonentów i na 40 linii. W praktyce realizowane są już automatyczne centrale telefoniczne oparte na mikrokomputerach z ponad tysiącem abonentów. Jednak w tych wielkich centralach wykorzystano znaczną liczbę mikroprocesorów.

Opisane projektowanie automatycznej centrali telefonicznej oparte jest na wykorzystaniu przerwań. Drugi sposób projektowania, który nie został tutaj opisany, ale stosowany jest w praktyce, zakłada wykorzystanie metody przeglądania końcówek abonenckich /polling/. Wówczas mikroprocesor sukcesywnie wywołuje końcówki abonenckie w celu "stwierdzenia", czy któraś z nich zgłosiła żądanie połączenia, "udziela" połączenia odpowiedniemu abonentowi i "wędruje" dalej. Jest to tzw. "programowa obsługa przerwania". Taka metoda obsługi przerwania wymaga dłuższego programu i jest względnie powolna, jednakże stosując ją uzyskuje się oszczędności w zakresie sprzętu /nie stosuje się układów 8214, itd./.

ZUŻYCIE ENERGII ELEKTRYCZNEJ

Interesująca jest sprawa zasilania, którego wymaga sterownicza część automatycznej centrali telefonicznej z rys.56.

W tab.9 podano maksymalne i typowe wartości zużycia prądu /Power Supply Current/ dla wszystkich składników z rys.56.

Ponieważ rys.56 zawiera wszystkie wymienione do tej pory składniki z rodziny mikroprocesora Intel 8080, to tab.9 umożliwia uzyskanie poglądu na wymagania dotyczące zasilania przy projektowaniu za pomocą mikroprocesora automatycznej centrali telefonicznej, lub jakiegoś innego urządzenia.

| Układ | Źródło | Maks. | Typ |
|---|---|---|---|
| 8080A | + 5 V | 80mA | 60mA |
| | +12 V | 70mA | 40mA |
| | - 5 V | 1mA | 0.01mA |
| 8708 | + 5 V | 10mA | 6mA |
| | +12 V | 65mA | 50mA |
| | - 5 V | 45mA | 30mA |
| 8224 | + 5 V | 115mA | brak danych* |
| | +12 V | 12mA | brak danych |
| 8205 | + 5 V | 70mA | brak danych |
| 8212 | + 5 V | 130mA | 90mA |
| 8214 | + 5 V | 130mA | 90mA |
| 8216 | + 5 V | 130mA | 95mA |
| 8228 | + 5 V | 190mA | 140mA |

Tab.9.

Mikrokomputer z rys.56 wymaga, w najgorszym razie źródeł następujących mocy:

źródło +5 V mocy 20,00W

źródło +12 V mocy 1,80W

źródło -5 V mocy 0,24W

Typowy pobór mocy mikrokomputera z rys.56 wynosi

źródło +5 V mocy 12,00W

źródło +12 V mocy 1,20W

źródło -5 V mocy 0,18W

Stosunek zużycia maksymalnego do typowego wynosi zatem 1,5.

---

* Brakuje wartości, których producent nie podaje.

Największy udział w stratach mocy mają układy 8212. Po pierwsze, jest ich dużo, po drugie wyprodukowane są w technologii bipolarnej. Ale zamiast układów 8212 można stosować układy 8255.

8255

Jednostki peryferyjne mogą być programowalne lub nieprogramowalne. Układ 8212 jest jednostką nieprogramowalną tzn. jest zwykłym rejestrem wejściowo/wyjściowym. Wśród elementów mikrokomputera opartego na mikroprocesorze Intel 8080 istnieją również programowalne jednostki peryferyjne. Taką jednostką jest układ 8255 /Programmable Peripheral Interface, PPI/. Układ 8255 zawiera 3 komórki peryferyjne oraz jeden rejestr sterujący. Dlatego też zakres stosowania tej jednostki peryferyjnej jest bardzo szeroki. Układ 8255 produkuje się w technologii MOS, stąd wymagania tego układu pod względem zasilania są niewielkie. Układ 8255 wymaga zasilania +5 V. Typowe zużycie mocy tego układu wynosi 40 mA. Przez zastosowanie układu 8255 wymagana moc typowa źródeł +5 V spada do około 7,5 W. Niedawno na rynku pojawiła się również ulepszona wersja tego układu tj.8255A.

LITERATURA

C1, D1, D6, F1, F2, K1, K2.

# FILTR CYFROWY

Filtr jest podstawowym elementem każdego urządzenia telekomunikacyjnego. Nie można więc wyobrazić sobie urządzenia telekomunikacyjnego bez określonej liczby filtrów.

Teoria filtru przeszła w swoim rozwoju przez 3 fazy:

bierny /pasywny/ filtr analogowy

aktywny filtr analogowy

filtr cyfrowy

W biernych filtrach analogowych wymagana funkcja przenoszenia realizowana jest za pomocą cewek i kondensatorów.

Pojawienie się filtrów aktywnych umożliwiło zrealizowanie wymaganej funkcji przenoszenia za pomocą wzmacniaczy, oporników i kondensatorów. Pozwala to na oszczędność kosztów i objętości. Z tego powodu aktywne filtry znalazły szerokie zastosowanie, szczególnie w niskoczęstotliwościowej części urządzeń telekomunikacyjnych do przekazywania telegrafii, danych i mowy.

Pojawienie się filtrów cyfrowych umożliwia zrealizowanie wymaganej funkcji przenoszenia za pomocą procesora cyfrowego. Daje to zwiększenie elastyczności filtru. Zmianę impulsowej odpowiedzi filtru osiąga się przez zmianę programów. Nie jest potrzebna żadna zmiana w strukturze schematu elektrycznego.

Do czasu pojawienia się układów kalkulatorowych i mikroprocesorów, filtry cyfrowe nie były dostępne szerokiemu kręgowi użytkowników. Filtrowanie cyfrowe w czasie rzeczywistym mogło być przeprowadzone tylko za pomocą problemowo ukierunkowanych komputerów. Takie komputery są drogie i duże. Po pojawieniu się układów kalkulatorowych i mikroprocesorów filtry cyfrowe zdobyły popularność. Cena i objętość obecnie znacznie się zmniejszyły.

Filtry cyfrowe mają jeszcze dwie ważne cechy. To pierwsze, ich projektowanie może być na tyle precyzyjne, na ile tego potrzebujemy. Osiąga się to w taki sposób, że mierzony sygnał wejściowy kodujemy nie 8, lecz 16 lub większą liczbą bitów. Jeżeli algorytm opiera się na rozwinięciu w szereg, wówczas weźmiemy tyle członów szeregu, ile będzie potrzebnych do osiągnięcia wymaganej dokładności.

Po drugie, klasa transmitancji, które mogą być realizowane za pomocą filtrów cyfrowych jest szersza od klasy transmitancji, jakie mogą być realizowane za pomocą filtru analogowego.

Istota filtru cyfrowego polega na tym, że analogowy sygnał wejściowy próbkowany jest w równych odstępach czasu /$t_i$/ z częstotliwością określoną przez teorię o kwantowaniu. Ostatnia próbka wejściowa

$$X\,/t_i = nT/$$

poprzedzająca M próbek wejściowych

$$X/t_i = nT - T/,\ldots,X/t_i = nT - MT/$$

oraz poprzedzająca N skwantowanych wartości wyjściowych

$$Y/t_i = nT - T/,\ldots,Y/t_i = nT - NT/$$

połączona jest algorytmem realizującym transmitancję filtru. Za pomocą tego algorytmu wytwarza się ostatnią skwantowaną wartość wyjściową

$$Y/t_j = nT/$$

Wyjściowy sygnał analogowy otrzymuje się po interpolacji skwantowanych wartości wyjściowych. Sygnał ten, Y/t/, ma taką samą postać jak sygnał na wyjściu filtra analogowego o takiej samej transmitancji.

A więc na wejściu do mikrokomputera musi znajdować się konwerter analogowo-cyfrowy zaś na wyjściu konwerter cyfrowo-analogowy.

A/D i D/A

Niedawno pojawiły się pierwsze hybrydowe konwertery A/D i D/A, specjalnie zaprojektowane do stosowania z mikrokomputerami. Są to konwertery A/D: MP20 i MP21, i konwertery D/A - MP10 i MP11 produkcji firmy Burr Brown. Konwertery MP20 i MP10 przeznaczone są do stosowania w ramach mikrokomputera opartego na mikroprocesorze Intel 8080. Konwertery te połączone są bezpośrednio z szynami mikrokomputera. Sposób połączenia jest wyspecyfikowany przez ich producenta i użytkownik nie ma w tym zakresie żadnej dowolności.

Dla programisty konwertery te stanowią tylko jedną komórkę pamięci bądź peryferyjną, w zależności od sposobu w jaki konwerter jest połączony z szyną kontrolną.

Jedna obudowa konwertera A/D zawiera 2 konwertery A/D i ma 80 końcówek /QIL/. Dwa konwertery A/D mogą być stosowane również jako jeden konwerter A/D z wejściami różnicowymi.

Przedział między dwoma kolejnymi odczytami danych z konwertera A/D nie może być krótszy od czasu konwersji A/D, który waha się od 20 do 200 $\mu$s i regulowany jest za pomocą specjalnego potencjometru. Przy pracy z konwerterem A/D mikroprocesor musi zapewnić sterowanie dla początku konwersji /Conversion Strobe, Start-of-Conversion Signal/. Cyfrowy pomiar może być odczytany z przetwornika A/D dopiero po upływie okresu konwersji. Koniec konwersji oznaczony jest pojawieniem się specjalnego sygnału /End-of-Conversion Signal/.

Jedna obudowa zawiera dwa konwertery D/A i ma 32 końcówki /DIL/. Czas konwersji konwertera D/A wynosi 25 mikrosekund.

Niedawno pojawił się również pierwszy monolityczny konwerter A/D spełniający taką samą rolę, a który jest kompatybilny z mikroprocesorem Intel 8080 /National 8292/.

Oczywiście można stosować również tradycyjne konwertery A/D i D/A, jednakże niemożliwe jest wówczas bezpośrednie sprzężenie z szynami mikrokomputera. Doświadczony projektant częściej stosuje tradycyjny konwerter A/D, ponieważ różnice między tradycyjnym a mikroprocesorowo ukierunkowanym konwerterem A/D są znacznie mniejsze od różnicy ich cen.

Tak więc przez konwerter A/D wprowadza się sygnał analogowy, nad którego wartościami wyjściowymi wykonywać się będzie operacje obliczeniowe, wyspecyfikowane przez algorytm danego filtra cyfrowego. Ponieważ mnożenie i dodawanie stanowi podstawę algorytmu każdego filtra cyfrowego, poświęcimy szczególną uwagę mnożeniu i dodawaniu liczb binarnych, jak również odcinkom programu realizującym te operacje.

MNOŻENIE ARYTMETYCZNE

Mnożenie dodatnich liczb binarnych, których wartość zawarta jest między 0 a 255 przedstawione jest na przykładzie liczb

121D = 01111001B

10D = 00001010B

Przy mnożeniu liczb binarnych obowiązują te same zasady, co przy mnożeniu liczb dziesiętnych. Tak więc:

| | Liczba D | Liczba C |
|---|---|---|
| | 01111001 | . 00001010 = |
| 1 | 00000000 | |
| 2 | 01111001 | |
| 3 | 011110010 | |
| 4 | 00000000 | |
| 5 | 0011110010 | |
| 6 | 01111001 | |
| 7 | 10010111010 | |
| 8 | 00000000 | |
| 9 | 010010111010 | |
| 10 | 00000000 | |
| 11 | 0010010111010 | |
| 12 | 00000000 | |
| 13 | 00010010111010 | |
| 14 | 00000000 | |
| 15 | 0000010010111010 | |

Wynik znajduje się w 15 wierszu i stanowi binarny równoważnik liczby dziesiętnej 1210.

Pierwszy wiersz otrzymujemy przez przemnożenie liczby D przez najmłodszy bit liczby C. Ponieważ najmłodszy bit liczby C jest równy 0, to wynikiem tego mnożenia jest bajt zero /00000000/.

Drugi wiersz otrzymujemy przez przemnożenie liczby D przesuniętej o jedno miejsce na lewo przez bit najbliższy najmłodszemu bitowi liczby C. Ponieważ ten bit jest równy jedności, wynikiem tego mnożenia jest sama liczba D. A więc składnikiem dodawania jest albo bajt zero albo sama liczba D.

Trzeci wiersz otrzymujemy przez dodanie pierwszego i drugiego wiersza. Wszystko to powtarza się jeszcze 6 razy. Na końcu otrzymujemy wynik 16-bitowy.

Oznacza to, że mnożenie liczb binarnych sprowadza się do sukcesywnego przesuwania liczby D w lewo i dodawania do wyniku poprzedniego dodawania, lub do pominięcia tego dodawania. Przesuwanie liczby D w lewo odpowiada przesuwaniu wyniku poprzedniego dodawania w prawo. Odcinek programu wykonujący mnożenie w ten sposób ma następującą postać:

```
1   A4:   MVI   B,00H
2         MVI   E,09H
3   A5:   MOV   A,C
4         RAR
5         MOV   C,A
6         DCR   E
```

```
 7              JZ      A7
 8              MOV     A,B
 9              JNC     A6
10              ADD     D
11      A6:     RAR
12              MOV     B,A
13              JMP     A5      PIERWSZY ROZKAZ NASTĘPNEGO ODCINKA PROGRAMU
14      A7:
```

Na początku czynniki znajdują się w rejestrach C i D. Na końcu starszy bajt wyniku znajduje się w rejestrze B, zaś młodszy w rejestrze C.

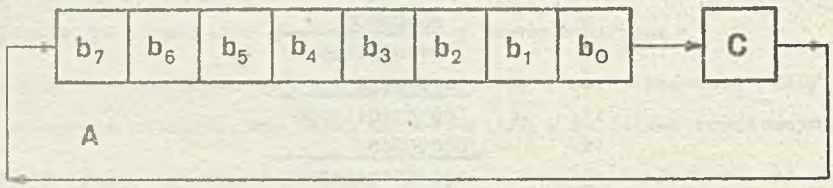
Z następującymi rozkazami spotykamy się pierwszy raz

RAR /Rotate Accumulator Right through carry/

i

ADD R

Rozkazem RAR realizuje się operację przedstawioną na rys.60.




Rys.60. Ilustracja operacji realizowanej w trakcie wykonywania rozkazu RAR

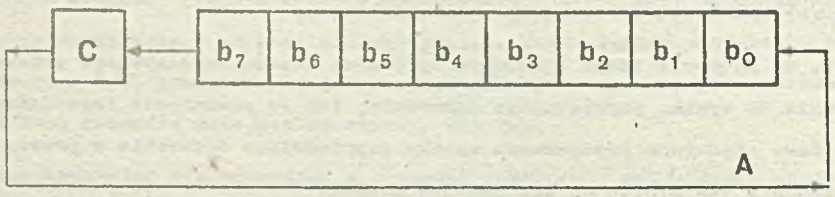
Zawartość akumulatora przesuwa się o jedno miejsce w prawo. Najmłodszy bit akumulatora umieszcza się jako wskaźnik C. Poprzedni wskaźnik C staje się najstarszym bitem akumulatora. Kodem tego rozkazu jest

00011111

Do tego rozkazu podobny jest rozkaz

RAL /Rotate Accumulator Left through carry/

Rozkazem RAL realizuje się operację przedstawioną na rys.61.




Rys.61. Ilustracja operacji realizowanej w trakcie wykonywania rozkazu RAL

Za pomocą rozkazów RAR i RAL dokonuje się przesunięcia zawartości akumulatora w prawo i lewo p r z e z  p r z e n i e s i e n i e.

Te dwa rozkazy należy odróżnić od rozkazów

RRC /Rotate accumulator Right with Carry/

oraz

RLC /Rotate accumulator Left with Carry/

Za pomocą rozkazów RRC i RLC dokonuje się przesunięcia zawartości akumulatora w prawo i w lewo z p r z e n i e s i e n i e m.

Ostatnie dwie operacje przedstawiono na rys.62.

Rozkazem

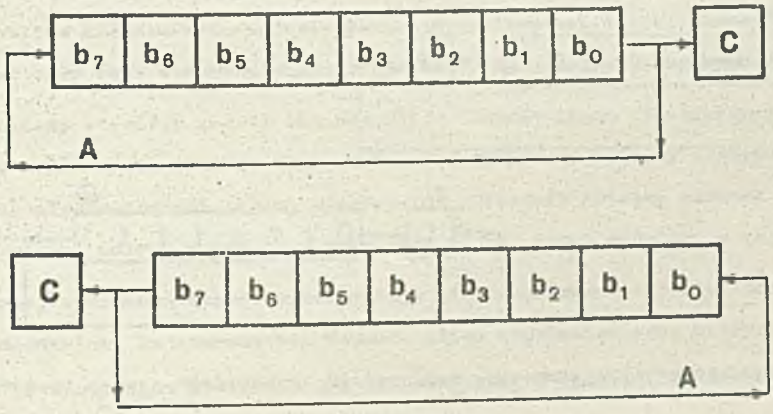
ADD D

dodaje się zawartość rejestru D do zawartości akumulatora. Ogólna postać tego rozkazu jest

ADD R

Kod dla tego rozkazu jest następujący

10000RRR




Rys.62. Ilustracja operacji realizowanych w trakcie wykonywania rozkazów RRC /na górze/ oraz RLC /na dole/

Do rozkazu ADD R podobny jest rozkaz

SUB R /SUBtract/

Rozkazem SUB R zawartość rejestru R odejmuje się od zawartości akumulatora.

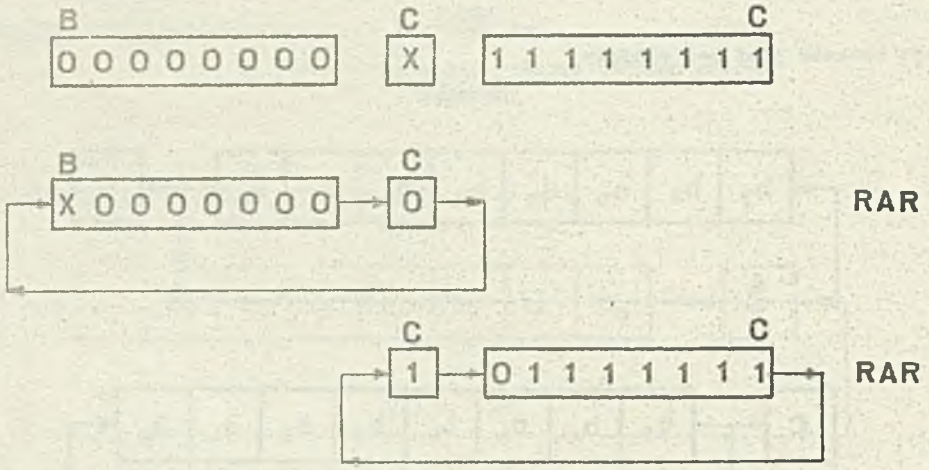
Wróćmy obecnie do odcinka programu obejmującego mnożenie liczb dodatnich.

Zamiast liczbę D przesuwać w lewo obecnie wynik poprzedniego dodawania przesuwa się w prawo. Z załączonego przykładu mnożenia liczb binarnych widać, że wynik poprzedniego dodawania na początku składa się z 8 bitów /1 wiersz/, zaś na końcu z 16 bitów /15 wiersz/. Dlatego w podanym odcinku programu wynik poprzedniego dodawania, na początku zajmuje tylko rejestr B. Rejestr ten musi być opróżniony na samym początku odcinka programu /1 rozkaz/. W trakcie pracy wynik poprzedniego dodawania rozszerza się w prawo i powoli zajmuje rejestr C. Na końcu wynik znajduje się w rejestrach B /starszy bajt/ i C /młodszy bajt/.

Czy dodawać się będzie zawartości rejestrów B i D na początku, to zależeć będzie od wartości najmłodszego bitu w rejestrze C. Wgląd do wartości najmłodszego bitu w rejestrze C osiąga się przez przesunięcie zawartości rejestru C przez wskaźnik C. Bezpośrednie przesunięcie rejestru C nie jest możliwe. Dlatego też przesunięcie rejestru C dokonuje się za pośrednictwem akumulatora

/3, 4 i 5 rozkaz/. Przesuwając zawartość rejestru C w prawo stwarza się jednocześnie miejsce na rozszerzenie wyniku poprzedniego dodawania z rejestru B. Jeżeli najmłodszy bit w rejestrze C jest równy jedności, wówczas dokonane zostanie dodawanie zawartości rejestrów B i D. W przeciwnym razie dodawanie takie nie będzie wykonane. Bezpośrednie dodawanie rejestrów B i D jest niemożliwe. W tym celu znowu wzywa się na pomoc akumulator /8, 9 i 10 rozkaz/. Następnie wynik poprzedniego dodawania przesuwa na prawo /8, 11, 12 i 4 rozkaz/.

Na rys.63 przedstawiono przenoszenie zawartości rejestru B do rejestru C, za pomocą wskaźnika C i dwóch rozkazów RAR.




Rys.63. Ilustracja sposobu, w jaki przenosi się zawartość rejestru B do rejestru C

Oczywiście takie przenoszenie dokonywane jest za pośrednictwem akumulatora.

Wszystko to powtarza się aż do opróżnienia rejestru E, służącego jako licznik. Stan początkowy licznika tworzy się za pomocą rozkazu 2. Realizacja programu kończy się, gdy zawartość rejestru E staje się równa zeru /6 i 7 rozkaz/. Odcinek programu obejmujący mnożenie liczb całkowitych trwa od 623 do 655 okresów taktu podstawowego. Czas trwania tego odcinka programu jest zmienny, gdyż rozkaz ADD D czasem się opuszcza. Czas trwania odcinka programu zależy od liczby jedynek w rejestrze C, jednak może być stały. Rozkaz ADD D trwa przez 4 okresy taktu podstawowego. Aby czas trwania omawianego tu odcinka programu był stały, należy dorzucić 4 okresy taktu podstawowego wtedy, gdy rozkaz ADD D nie jest wykonywany. Osiąga się to przez dodanie rozkazu nie spełniającego żadnej innej roli poza realizacją opóźnienia o 4 okresy taktu podstawowego /Dummy Instruction/.

Przy zachowaniu warunku, że okres taktu podstawowego wynosi 500 ns, jedno mnożenie trwa w najgorszym razie 327,5 µs. Przy kwantowaniu cyfrowym sygnału mowy pomiary sygnału wejściowego pobierane są co 125 µs. Oznacza to, że mikroprocesor Intel 8080 jest zbyt wolny dla cyfrowego kwantowania sygnału głosowego. Jak już wspomnieliśmy, jest to właściwość wszystkich mikroprocesów MOS.

Jednak do mikrokomputera można "dorzucić" również jeden układ kalkulatorowy. Dopiero wówczas można przeprowadzać kwantowanie cyfrowe sygnału głosowego za pomocą mikroprocesora MOS. Jednym z

najnowszych układów tego typu jest układ mnożący Monolythic Memories 57558/67558. Jego zastosowanie powoduje, że mnożenie dwóch liczb 8-bitowych trwa 100 ns!

MNOŻENIE ALGEBRAICZNE

Opisany odcinek programu nie może być bezpośrednio wykorzystywany do mnożenia liczb algebraicznych w postaci uzupełnienia dwójkowego, których wartość wynosi od -128 do +127. Jednak może być on wykorzystany pośrednio. Wówczas, po pierwsze, określa się znak wyniku. Znak wyniku jest dodatni, jeżeli najstarszy bit czynników jest jednakowy. W przeciwnym razie znak wyniku jest ujemny. Aby określić znak wyniku należy dokonać na czynnikach operacji logicznej różnicy symetryczne j. Jeżeli najstarszy bit wyniku tej operacji jest równy zeru, wówczas wynik mnożenia algebraicznego jest dodatni, i odwrotnie. Tak więc najstarszy bit wyniku tej operacji daje informację o znaku wyniku mnożenia algebraicznego i należy go chwilowo zapamiętać. Jednak w mikroprocesorze 8080 nie jest możliwe krótkotrwałe zapamiętywanie wyizolowanych bitów. Dlatego też należy krótkotrwale zapamiętać cały bajt stanowiący wynik operacji logicznej różnicy symetrycznej.

Po drugie, należy określić moduły obu czynników. Moduły liczb algebraicznych są zawsze dodatnie, wobec czego można w stosunku do nich zastosować uprzednio opisany algorytm. Jeżeli do mnożenia modułów liczb algebraicznych chcemy wykorzystać uprzednio opisany odcinek programu, wówczas moduł jednego czynnika należy umieścić w rejestrze C, zaś moduł drugiego w rejestrze D.

Dopiero teraz, w trzecim etapie wykorzystuje się bezpośrednio odcinek programu obejmujący mnożenie liczb dodatnich. Najstarszy bit modułów liczb algebraicznych, których wartość leży między -127 a + 127 jest zawsze równy zeru. Dlatego też uprzednio opisany odcinek programu może być skrócony o 67 okresów taktu podstawowego. Czyli rozkaz

MVI E, 09H

może być zastąpiony rozkazem

MVI E, 08H

Po czwarte - tworzy się ostateczny wynik. W tym celu testuje się najstarszy bit chwilowo zapamiętanego wyniku operacji logicznej różnicy symetrycznej. Jeżeli bit ten równy jest jedności, czyli jeśli wynik mnożenia algebraicznego jest ujemny, wówczas wynik mnożenia algebraicznego przekształca się do postaci uzupełnienia dwójkowego.

Określanie znaku wyniku

Załóżmy, że czynniki

-01D = FFH

+10D = 0AH

znajdują się w komórkach pamięciowych, których adresy to: 03FEH i 0401H. Załóżmy także, że wierzchołek stosu ma adres 07FFH. Odcinek programu, służący do określania i krótkotrwałego zapamiętywania znaku wyniku mnożenia algebraicznego może mieć następującą postać:

```
LDA     03FEH
MOV     B,A
```

```
LDA     0401H
XRA     B
PUSH    PSW
```

Jak już stwierdzono, za pomocą rozkazu

```
LDA ADRM
```

do akumulatora przenosi się zawartość komórki pamięciowej, której adres specyfikuje operand, zaś za pomocą rozkazu

```
MOV     B,A
```

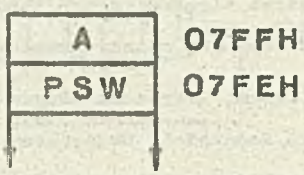
zawartość akumulatora przenosi się do rejestru B. Rozkazem

```
XRA B
```

dokonuje się operacji logicznej różnicy symetrycznej wobec czynników. Wynik tej operacji znajduje się w akumulatorze i musi być, jak to już poprzednio stwierdzono, chwilowo zapamiętany.

Daną można krótkotrwale zapamiętać w jednym z rejestrów lub w jednej z komórek pamięci. Jak już wspomnieliśmy, przenoszenie danych do pamięci może być dokonywane za pomocą rozkazów STA ADRM i PUSH RP. Rozkaz PUSH RP jest wydajniejszy. Z tego powodu w ostatnim odcinku programu zastosowany jest stos do chwilowego zapamiętywania danych. Na rys.64 przedstawiono wygląd stosu po wykonaniu rozkazu PUSH PSW.

Zauważmy, że do stosu w pierwszej kolejności przenosi się zawartość akumulatora. Tak więc akumulator jest traktowany jako starszy bajt pary rejestrowej A-PSW. Czyli, na stos przenosi się zawsze najpierw starszy bajt rejestru szesnastobitowego.




Rys.64. Stos po wykonaniu rozkazu PUSH PSW

Ostatni odcinek programu może być także zapisany w następujący sposób:

```
LXI     DE,03FEH
LXI     HL,0401H
LDAX    DE
XRA     M
PUSH    PSW
```

Za pomocą trzeciego rozkazu zawartość komórki pamięciowej, której adres jest równy zawartości rejestru DE przenosi się do akumulatora. Ogólna postać tego rozkazu jest następująca:

```
LDAX RP /Load Accumulator/
```

Kodem tego rozkazu jest

```
000R1010
```

Bit R definiuje rejestr 16-bitowy, którego zawartość jest równa adresowi komórki pamięciowej, której zawartość przenoszona jest do akumulatora. R dotyczy rejestru BC lub rejestru DE.

Przed wykonaniem rozkazu LDAX DE zawartość rejestru DE jest 03FEH - na podstawie działania pierwszego rozkazu LXI z poprzedniego programu.

Odwrotny względem LDAX RP jest rozkaz

STAX RP /STore Accumulator/

Za pomocą tego rozkazu zawartość akumulatora przenoszona jest do komórki adresowej, której adres równy jest zawartości rejestru BC lub rejestru DE.

Do rozkazów LDAX RP i STAX RP podobne są rozkazy

MOV R,M

MOV M,R

Za pomocą pierwszego rozkazu przenosi się zawartość komórki pamięciowej, której adres równy jest zawartości rejestru HL do jednego z 7 istniejących rejestrów.

Drugim rozkazem dokonuje się przesyłania w przeciwnym kierunku.

Za pomocą ozwartego rozkazu ostatniego odcinka programu realizuje się operację logicznej różnicy symetrycznej. Ogólna postać tego rozkazu jest następująca:

XRA M

Jeden operand znajduje się w akumulatorze. Drugi operand jest w komórce pamięci, której adres równy jest zawartości rejestru HL. Kodem tego rozkazu jest

10101110

Przed wykonaniem ozwartego rozkazu zawartością rejestru HL jest 0401H. Dlatego ozwarty rozkaz wykonuje taką samą operację, jak następujące trzy rozkazy:

```
-------------
MOV     B,A
LDA     0401H
XRA     B
-------------
```

Do rozkazu XRA M podobne są rozkazy

ORA M

i ANA M

Pierwszy rozkaz służy do wykonania operacji logicznej LUB, zaś drugi do realizowania operacji logicznej I.

Ostatnie dwa odcinki programu spełniają taką samą funkcję, jednak różnią się pod względem sposobu, w jaki specyfikuje się adresy komórek pamięci, tzn. pod względem stosowanego sposobu adresowania /Memory Addressing/.

W pierwszym odcinku programu stosuje się bezpośrednie adresowanie komórek adresowych /Direct Addressing/. W drugim stosowane jest adresowanie pośrednie /Implied Addressing/. Poza tym termin "adresowanie pośrednie" używany jest w praktyce również w odniesieniu do innych typów adresowania.

W adresowaniu bezpośrednim adres komórki pamięciowej, w której umieszczony jest operand, znajduje się w samym rozkazie /np. LDA 03FEH/.

W adresowaniu pośrednim, tak jak zostało ono tutaj zdefiniowane, adres komórki pamięciowej znajduje się w rejestrach BC, DE lub HL /np. LDAX RP i MOV R,M/. Z wyjątkiem rozkazów LDAX RP i STAX RP, symbol M w polu operanda wskazuje, że mamy do czynienia z adresowaniem pośrednim.

Wprowadzenie adresowania pośredniego komplikuje "życie" początkującemu programiście, jednak

dzięki temu można osiągnąć znaczne oszczędności czasu komputera i objętości pamięci. Wówczas program trwa krócej oraz zajmuje mniej miejsca w pamięci ROM.

W załączniku umieszczonym na końcu tej pracy można zauważyć, że rozkaz LDA 03FEH trwa przez 13 okresów taktu podstawowego i zajmuje 3 bajty pamięci ROM. Rozkazy LDAX RP i MOV R,M trwają 7 okresów taktu podstawowego i zajmują tylko 1 bajt pamięci ROM.

W naszym przykładzie odcinek programu, w którym stosuje się adresowanie bezpośrednie trwa 35 okresów taktu podstawowego i zajmuje 8 bajtów pamięci ROM. Odcinek programu, w którym wykorzystuje się adresowanie pośrednie trwa 34 okresy taktu podstawowego i zajmuje również 8 bajtów pamięci ROM. Tak więc nie uzyskuje się prawie żadnej oszczędności. Ma to znaczenie dopiero w przy dłuższych odcinkach programu, a szczególnie przy pracy z większą ilością danych, umieszczonych w komórkach pamięci mających sąsiednie adresy. Oznacza to, że adresowanie pośrednie nie jest lepsze.

Przy adresowaniu pośrednim rejestr HL jest "uprzywilejowany". Rejestry BC i DE umożliwiają tylko przesyłanie między komórkami pamięciowymi a akumulatorem, natomiast rejestr HL umożliwia przesyłanie między komórkami pamięciowymi i każdym z rejestrów. Następnie, w odróżnieniu od rejestrów BC i DE, rejestr HL umożliwia również wykonywanie określonych operacji logicznych i arytmetycznych.

Mikroprocesor Intel 8080 ma niewielkie możliwości różnorodnego adresowania komórek pamięciowych. Wśród mikroprocesorów można spotkać wiele innych typów adresowania: adresowanie indeksowe /Indexed Addressing/ z odmianami /Pre-Indexed Addressing, Post-Indexed Addresing/, adresowanie stronami /Page Addressing/ z odmianami /Base Page Addressing, Relative Page Addressing.../, jak również typ adresowania pośredniego, w którym adres komórki pamięci nie znajduje się w parze rejestrowej, lecz w parze komórek pamięciowych /Indirect addressing/ z różnymi odmianami i inne.

Określanie modułów czynników

Załóżmy, że zawartość rejestrów DE i HL pozostała niezmieniona. Przypomnijmy, że zawartością rejestru DE jest 03FEH, zaś rejestru HL jest 0401H.

Odcinek programu, za pomocą którego określa się moduły obu czynników ma wówczas następującą postać:

```
        LDAX    DE
        ANA     A
        JP      A1
        CMA
        INR     A
A1:     MOV     D,A
        MOV     A,M
        ANA     A
        JP      A2
        CMA
```

```
        INR     A
A2:     MOV     C,A
```

Pierwszy czynnik wprowadza się do akumulatora za pomocą rozkazu

LDAX DE

Niestety, po tym rozkazie nie ustala się zawartości rejestru stanu, tzn. po tym rozkazie wskaźniki nie odzwierciedlają stanu w akumulatorze. W załączniku umieszczonym na końcu tej książki można zauważyć, że dotyczy do wszystkich rozkazów, za pomocą których dokonuje się przesyłania danych przez mikrokomputer. Dlatego rozkaz

ANA A

ma jedynie za zadanie ustalenie zawartości rejestru stanu. Przez wykonanie rozkazu ANA A nie zmienia się zawartości akumulatora. Istnieją również inne rozkazy, za pomocą których można ustalić zawartość rejestru stanu, nie zmieniając przy tym zawartości akumulatora. Rozkaz ANA A jest najkrótszy. Trwa tylko przez 4 okresy taktu podstawowego.

Rozkazem

JP A1

testuje się znak pierwszego czynnika. Jeżeli pierwszy czynnik jest dodatni, wówczas nie określa się modułu tego czynnika.

Rozkazy

CMA

i

INR A

realizują określanie modułu liczby znajdującej się w akumulatorze.

Za pomocą rozkazu

MOV D,A

przenosi się pierwszy czynnik do rejestru D. Następnie powtarza się wszystko z drugim czynnikiem.

Na końcu tego odcinka programu moduły czynników znajdują się w rejestrach C i D. Nie przypadkowo przyjęliśmy, że moduły czynników zostaną umieszczone w rejestrach C i D. Jak już powiedziano, po określeniu modułów następuje mnożenie liczb dodatnich, zaś poprzednio opisany odcinek programu obejmujący mnożenie liczb dodatnich wymaga, aby czynniki znajdowały się w rejestrach C i D.

Mnożenie liczb dodatnich

Trzecia faza mnożenia algebraicznego sprowadza się do mnożenia liczb dodatnich, czym zajmowaliśmy się wcześniej. Dlatego przejdziemy od razu do ostatniej fazy, tzn. do tworzenia wyniku końcowego.

Tworzenie wyniku końcowego

Wynik mnożenia arytmetycznego znajduje się w rejestrze BC. Odcinek programu, za pomocą którego - jeżeli istnieje potrzeba - znajduje się uzupełnienie dwójkowe zawartości rejestru BC, ma następującą postać:

```
        POP     PSW
        ANA     A
        JP      A3
        MOV     A,B
        CMA
        MOV     B,A
        MOV     A,C
        CMA
        MOV     C,A
        INX     B,C
A3:                     ; PIERWSZY ROZKAZ NASTĘPNEGO ODCINKA
                        ; PROGRAMU
```

Za pomocą 3 pierwszych rozkazów testuje się znak wyniku mnożenia algebraicznego. Jeżeli znak mnożenia algebraicznego jest ujemny, wówczas za pomocą 7 następnych rozkazów tworzy się uzupełnienie dwójkowe 16-bitowej liczby, znajdującej się w rejestrze BC.

Podsumowanie

Uwzględniając wszystko, co do tej pory zostało powiedziane, kompletny odcinek programu obejmujący mnożenie liczb algebraicznych ma następującą postać:

```
        LDAX    DE      ; OKREŚLANIE ZNAKU
        XRA     M       ; WYNIKU
        PUSH    DE
        PUSH    PSW
        LDAX    DE      ; OKREŚLANIE MODUŁU
        ANA     A       ; PIERWSZEGO CZYNNIKA
        JP      A1
        CMA
        INR     A
A1:     MOV     D,A
        MOV     A,M     ; OKREŚLANIE MODUŁU
        ANA     A       ; DRUGIEGO CZYNNIKA
        JP      A2
        CMA
        INR     A
A2:     MOV     C,A
A4:     MVI     B,00H   ; MNOŻENIE LICZB
                E,09H   ; DODATNICH
A5:     MOV     A,C
        RAR
        MOV     C,A
        DCR     E
        JZ      A7
        MOV     A,B
        JNC     A6
        ADD     D
A6:     RAR
        MOV     B,A
```

```
        JMP     A5
A7:     POP     PSW     ; OKREŚLANIE UZUPEŁNIENIA
        ANA     A       ; DWÓJKOWEGO
        JP      A3
        MOV     A,B
        CMA
        MOV     B,A
        MOV     A,C
        CMA
        MOV     C,A
        INX     BC
A3:     POP     DE*
```

Ten odcinek programu musi być poprzedzony rozkazami, za pomocą których definiuje się początek stosu oraz początkową zawartość rejestrów DE i HL:

```
        LXI     SP,0800H
        LXI     DE,03FEH
        LXI     HL,0401H
```

Przypomnijmy, że czynniki znajdują się w komórkach pamięciowych, których adresy równe są zawartościom rejestrów DE i HL. Starszy bajt wyniku znajduje się w rejestrze B, zaś młodszy w rejestrze C.

Rozkazy PUSH DE i POP DE oznaczone są gwiazdkami. Rejestr DE wykorzystuje się wpierw do adresowania pośredniego, a następnie rejestry D i E wykorzystuje się przy mnożeniu liczb całkowitych. Na końcu rejestr DE znowu wykorzystuje się do adresowania pośredniego i dlatego przed przejściem do mnożenia liczb całkowitych należy zawartość rejestru DE "przechować w pewnym miejscu". Osiąga się to za pomocą rozkazu PUSH DE. Po zakończeniu mnożenia należy odnowić zawartość rejestru DE. Czyni się to za pomocą rozkazu POP DE.

Należy podkreślić, że istnieje również krótszy algorytm algebraicznego mnożenia liczb wyrażonych w postaci uzupełnienia dwójkowego. Algorytm ten umożliwia bezpośrednie mnożenie liczb algebraicznych, ponieważ przy przesuwaniu zawartości akumulatora mikroprocesor dba o to, aby najstarszy bit liczb ujemnych był traktowany jako -128. Niektóre mikroprocesory mają w ogóle rozkazy asemblera, za pomocą których znacznie upraszcza się i przyspiesza bezpośrednie mnożenie liczb algebraicznych. Są to rozkazy arytmetycznego przesunięcia w prawo /Arithmetic Shift Right/ i w lewo /Arithmetic Shift Left/. Przy przesunięciu w prawo, zwolnione miejsce wypełnia się wartością odpowiadającą najstarszemu bitowi. Niestety, mikroprocesor Intel 8080 nie ma takiego rozkazu.

DODAWANIE ALGEBRAICZNE

Algebraiczne dodawanie dwóch liczb 8-bitowych w mikroprocesorze Intel 8080 wykonywane jest bezpośrednio. Jeden ze składników powinien znajdować się w akumulatorze. Drugi składnik powinien znajdować się w jednym z 7 rejestrów lub w komórce pamięci, której adres jest równy zawartości rejestru HL. W pierwszym wypadku stosuje się rozkaz

```
ADD R
```

który już poznaliśmy, zaś w drugim wypadku rozkaz

```
ADD M
```

To samo dotyczy odejmowania. Wówczas stosuje się rozkazy

SUB R

i

SUB M

Przy dodawaniu dwóch liczb 16-bitowych należy uważać na przeniesienie z najstarszej pozycji bitowej młodszego bajtu do najmłodszej pozycji bitowej starszego bajtu. Zilustrujemy to na następnym przykładzie:

| | | |
|---|---|---|
| 11* | | Przeniesienie |
| 01000011 | 10000000 | Pierwszy składnik |
| 00000000 | 10101010 | Drugi składnik |
| 01000100 | 00101010 | Wynik |

Gwiazdką oznaczono przeniesienie z młodszego do starszego bajtu.

A więc należy dwa starsze bajty dodać razem z przeniesieniem, jakie zachodzi przy dodawaniu młodszych bajtów. Dlatego też istnieją rozkazy, które przy dodawaniu uwzględniają przeniesienia, tzn. o wskaźnik C. Są to rozkazy

ADC R

i

ADC M /ADd with Carry/

Za pomocą tych rozkazów zawartość akumulatora sumuje się ze wskaźnikiem C i z drugim składnikiem. Drugi składnik znajduje się w jednym z 7 rejestrów lub w komórce pamięci, której adres równy jest zawartości rejestru HL. To samo dotyczy odejmowania dwóch liczb 16-bitowych. Wówczas stosuje się rozkazy:

SBB R

i

SBB M /SuBtract with Borrow/

Załóżmy, że młodszy bajt pierwszego składnika znajduje się w komórce pamięci, której adres jest równy zawartości rejestru HL a starszy bajt pierwszego składnika znajduje się w komórce pamięci, której adres równy jest zawartości rejestru HL zwiększonej o 1. Załóżmy również, że młodszy i starszy bajt drugiego czynnika znajduje się odpowiednio w rejestrach C i B. Wówczas odcinek programu dotyczący dodawania dwóch liczb 16-bitowych ma następującą postać:

```
MOV     A,C
DDD     M
MOV     C,A
INX     HL
MOV     A,B
ADC
```

Młodszy bajt wyniku znajduje się w rejestrze C, natomiast starszy bajt wyniku znajduje się w akumulatorze.

PODPROGRAM

Każdy odcinek programu może być przekształcony w podprogram /Subroutine/. Odcinek programu dotyczący dodawania algebraicznego po przekształceniu w podprogram ma postać:

```
DODAW:  MOV   A,C
        ADD   M
        MOV   C,A
        INX   HL
        MOV   A,B
        ADC   M
        RET
```

Tak więc do pierwszego rozkazu dodaje się etykietę, zaś za ostatnim rozkazem dodaje się rozkaz RET. To wszystko. Podprogram może być umieszczony w dowolnym miejscu pamięci ROM.

Nasz podprogram wywołuje się rozkazem

CALL DODAW

Ogólna postać tego rozkazu jest następująca:

CALL ADRM

Składa się on z 3 bajtów. Pierwszy bajt definiuje operację skoku do podprogramu, zaś pozostałe dwa bajty definiują adres pierwszego bajtu pierwszego rozkazu w podprogramie.

W ramach tego rozkazu poprzednia zawartość rejestru programu automatycznie jest umieszczana na stosie. Adres pierwszego bajtu rozkazu staje się nową zawartością rejestru programu. Przypomnijmy, że taką samą operację wykonywał wytworzony sprzętowo rozkaz RST N. Rozkaz RST N nie znajdował się w programie. Wykonywany on był po przyjęciu żądania przerwania i przebiegał pod kontrolą układu 8212. Jednak rozkaz RST N może być wytworzony również programowo, tzn. rozkaz ten może znajdować się w samym programie, wtedy zostanie on wykonany, gdy nadejdzie jego kolejność. Nie konieczne są ani żądanie przerwania ani układ 8212.

Tak więc podprogram może być wywoływany za pomocą programowo wytwarzanego rozkazu RST N. Podprogram nie może wówczas znajdować się w dowolnym miejscu pamięci ROM. Początkowym adresem podprogramu musi być jeden z 8 adresów z tab.4.

Oznacza to, że stosując rozkaz RST N traci się na elastyczności, zyskuje się jednak miejsce w pamięci i skraca czas działania. Mianowicie, rozkaz RST N trwa przez 11 okresów taktu podstawowego i zajmuje 1 bajt pamięci ROM. Natomiast rozkaz CALL ADRM trwa przez 17 okresów taktu podstawowego i zajmuje 3 bajty pamięci ROM.

Podobnie jak rozkaz RST N, tak samo rozkaz CALL ADRM można wytwarzać zarówno programowo jak i sprzętowo. Sprzętowe wytwarzanie rozkazu CALL ADRM stosuje się w mikrokomputerach z ponad 40 poziomami przerwania. Jest to najbardziej skuteczna, ale nie jedyna metoda rozróżniania ponad 40 poziomów przerwania.

Jak widać istnieje wielkie podobieństwo między podprogramem i odcinkiem programu, który następuje po przyjęciu żądania przerwania. Odcinek programu następujący po przyjęciu żądania przerwania nie jest niczym innym jak podprogramem zwanym podprogramem obsługi przerwania /Interrupt Service Routine/. Jedyna różnica polega na tym, że skok do podprogramu przerwania dokonywany jest asynchronicznie, natomiast skok do "klasycznego" podprogramu - synchronicznie.

Bieżący podprogram obsługi przerwania może być przerwany przez inny podprogram obsługi przerwania. Może to się powtarzać kilkakrotnie, tak długo, dopóki w pamięci RAM jest miejsce na rozsze-

rzanie stosu /Nesting/. Również zwykły podprogram może być przerwany przez inny podprogram. Innymi słowy, w ramach jednego podprogramu może być wezwany inny podprogram. To również może być wielokrotnie powtarzane.

Warunkowy skok do podprogramu

Za pomocą rozkazów CALL ADRM oraz RST N dokonuje się bezwarunkowego skoku do podprogramu. Jednakże często zachodzi potrzeba umożliwienia warunkowego skoku do podprogramu. Tym warunkiem jest zazwyczaj wynik jakiejś operacji. Jak już stwierdziliśmy, wynik prawie wszystkich operacji arytmetycznych i logicznych wpływa na stan rejestru stanu. Jedynymi wyjątkami są rozkazy INX RP oraz DCX RP. Dlatego istnieje wiele rozkazów testujących przed skokiem do podprogramu stan określonych wskaźników. W zależności od stanu danego wskaźnika skok do podprogramu jest dokonywany lub nie.

W taki sposób za pomocą rozkazów

CC ADRM /Call if Carry/

CNC ADRM /Call if Non Carry/

dokonywany jest warunkowy skok do podprogramu uzależniony od stanu wskaźnika C. Za pomocą rozkazów

CPE ADRM /Call if Parity Even/

i CPO ADRM /Call if Parity Odd/

dokonywany jest warunkowy skok do podprogramu uzależniony od stanu wskaźnika P. Za pomocą rozkazów

CZ ADRM /Call if Zero/

i CNZ ADRM /Call if Non Zero/

dokonywany jest warunkowy skok do podprogramu - uzależniony od stanu wskaźnika Z, natomiast rozkazy

CM ADRM /Call if Minus/

i CM ADRM /Call if Plus/

przed skokiem do podprogramu testują stan wskaźnika S.

Warunkowy powrót z podprogramu

Za pomocą rozkazu RET dokonywany jest bezwarunkowy powrót z podprogramu. Na podstawie stanu określonych wskaźników może być również dokonywany powrót warunkowy z podprogramu. Tak więc rozkazy

RC /Return if Carry/

i RNC /Return if Non Carry/

testują wskaźnik C. Rozkazy

RPE /Return if Parity Even/

i RPO /Return if Parity Odd/

testują wskaźnik P. Rozkazy

RZ /Return if Zero/

i RNZ /Return if Non Zero/

testują wskaźnik Z, natomiast rozkazy

RM /Return if Minus/

i RP /Return if Plus/

przed powrotem z podprogramu testują stan wskaźnika S.

Celowość stosowania podprogramu

Im częściej są wzywane i im są dłuższe podprogramy, tym bardziej są potrzebne. Wówczas przy ich stosowaniu oszczędza się miejsce w pamięci ROM.

Celowość przekształcania określonego odcinka programu w podprogram uzasadnimy również obliczeniami.

Odcinek programu o długości L bajtów, powtarzający się K razy zajmuje K . L komórek pamięci ROM. Ten sam odcinek programu może być przekształcony w podprogram, który będzie wywoływany rozkazem CALL ADRM. Podprogram wywoływany K razy zajmuje

$$3K + (L + 1) + 3$$

komórki pamięci ROM oraz

2

komórki pamięci RAM. Czynnik 3K wynika z K rozkazów CALL ADRM, z których każdy zajmuje 3 bajty. Czynnik L + 1 wynika z odcinka programu, do którego dodany jest rozkaz RET o długości jednego bajtu. Składnik 3 wynika z inicjalizacji stosu. Rozkaz LXI SP, P16 definiujący początek stosu zajmuje 3 bajty.

Jak wiadomo, w mikrokomputerze opartym na mikroprocesorze Intel 8080 stos może egzystować tylko w pamięci RAM i dlatego praca z podprogramami bezwzględnie wymaga pamięci RAM w mikrokomputerze. Dwie komórki w pamięci RAM potrzebne są do chwilowego zapamiętywania zawartości rejestru programowego.

Przekształcanie odcinka programu w podprogram jest uzasadnione dopiero wtedy, gdy spełniona jest następująca nierówność:

$$3K + L + 6 < K . L$$

Rozwiązując tę nierówność dla L otrzymuje się wyrażenie:

$$L_{min} > (3K + 6)/(K - 1)$$

$L_{min}$ jest minimalną długością odcinka programu, przy której przekształcanie odcinka programu w podprogram jest uzasadnione. Wartości $L_{min}$ dla różnych wartości parametru K gdy podprogram jest wywoływany rozkazem CALL ADRM, podane są w tab.10.

| K | $L_{min}$ |
|---|---|
| 1 | ∞ |
| 2 | 13 |
| 3 | 8 |
| 4 | 7 |
| 5 | 6 |
| 6 | 5 |
| 7 | 5 |
| 8 | 4 |
| ∞ | 4 |

Tab.10

Tak więc bezcelowe jest przekształcanie w podprogram odcinków programu zajmujących 4 lub mniej bajtów.

Jeżeli przy wywoływaniu podprogramu stosowany jest rozkaz RST N, wówczas przekształcanie odcinka programu w podprogram jest uzasadnione dopiero po spełnieniu następującej nierówności:

$$K + L + 6 < K \cdot 1$$

Rozwiązując tę nierówność dla L otrzymujemy wyrażenie:

$$L_{min} < (K + 6) / (K - 1)$$

Wartości $L_{min}$ dla różnych wartości parametru K, gdy podprogram jest wzywany rozkazem RST N podane są w tab.11

| K | $L_{min}$ |
|---|---|
| 1 | $\infty$ |
| 2 | 9 |
| 3 | 5 |
| 4 | 4 |
| 5 | 3 |
| 6 | 3 |
| 7 | 3 |
| 8 | 3 |
| 9 | 2 |
| $\infty$ | 2 |

Tab.11.

Obecnie, przy określonych warunkach, celowe jest przekształcenie jakiegokolwiek odcinka programu w podprogram.

MAKROROZKAZY

Każdy odcinek programu może być przekształcony w jakiś makrorozkaz asemblera, wówczas cały odcinek programu traktowany jest jako jeden rozkaz.

Tworzenie makrorozkazu dokonuje się w ten sposób, że programowi translującemu zwanemu asemblerem należy "dać do zrozumienia", że wszystkie rozkazy z jednego obszaru pamięci ma rozumieć jako jeden makrorozkaz. Obszar ten jest zazwyczaj definiowany za pomocą początkowego i końcowego adresu. Np. aby odcinek programu dotyczący dodawania, o którym poprzednio mówiliśmy, był rozumiany jako jeden makrorozkaz, należy go zapisać w następujący sposób:

```
ADR1    MACRO
        MOV     A,C
        ADD     M
        MOV     C,A
        INX     HL
        MOV     A,B
        ADC     M
        ENDM
```

MACRO i ENDM /END of Macro/ nie są rozkazami asemblera lecz dyrektywami makroasemblera. Nie są one tłumaczone, tzn. nie mają swojego kodu wynikowego. Służą tylko do zaznaczenia, że zawarty między nimi odcinek programu należy traktować jako jeden makrorozkaz.

Jeżeli chcemy, aby dany odcinek programu został wykonany, wówczas zamiast całego odcinka programu piszemy tylko jeden rozkaz zwany makrorozkazem:

ADR1

Makrorozkazy są ogromnym ułatwieniem dla programisty. Jednak stosując je nie uzyskuje się żadnych oszczędności miejsca w pamięci. Gdy tylko pojawi się makrorozkaz , wówczas program translatora asemblera[*] dokonuje kodowania i wpisu do pamięci wszystkich rozkazów asemblera wchodzących w skład danego makrorozkazu. Natomiast nie zdarza się to w podprogramie. Podprogram wpisywany jest tylko w jednym miejscu w pamięci, bez względu na to ile razy jest wywoływany.

FILTR DOLNOPRZEPUSTOWY

Powyższa dygresja miała na celu wprowadzenie kilku nowych pojęć niezbędnych do zrozumienia dalszego tekstu. Obecnie wracamy do filtra dolnoprzepustowego /przepuszczającego niskie częstotliwości/.

Algorytm realizujący funkcję transmitancji filtra dolnoprzepustowego ma następującą postać:

$$Y(nT) = k_1 \cdot Y(nT - T) + k_2 \cdot X(nT);\ n = 1,2,\ldots$$

Stałe $k_1$ i $k_2$ określone są następującymi wyrażeniami

$$k_1 = 2 - p - \sqrt{(p^2 - 4p + 3)}$$

$$k_2 = 1 - k_1$$

$$0 < k_1, k_2 < 1$$

gdzie

$$p = \cos(2\pi F_g T)$$

oraz

$$T < 1 / 2 F_g$$

Filtr ten ma jeden biegun oraz częstotliwość graniczną $F_g$. Jeżeli filtr nie magazynuje energii początkowej, wówczas obowiązuje:

$$Y(0) = 0$$

Oznacza to, że kondensator, z którego wyprowadza się sygnał wyjściowy, jest na początku rozładowany.

Na rys.65 przedstawiony jest przebieg na wyjściu filtra cyfrowego dolnoprzepustowego, którego częstotliwość graniczna wynosi $F_g$= 4 kHz i który nie magazynuje energii początkowej. Sygnał wejściowy składa się ze zbioru dwóch sinusoid o takiej samej amplitudzie i różnej częstotliwości: $F_1$= 1 kHz i $F_2$= 2 kHz. Okres kwantowania jest 20 razy krótszy niż zakłada to teoria o kwantowaniu.

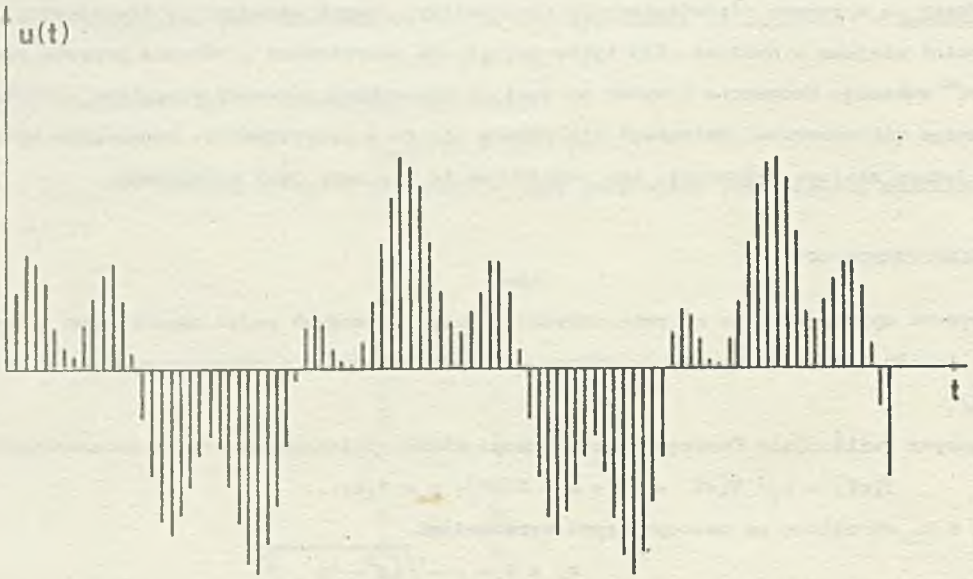
Załóżmy, że sygnał wejściowy jest losowy i aperiodyczny, z częstotliwością graniczną $F_1$ = 250 Hz. Oznacza to, że kwanty wejściowe należy pobierać z minimalną częstotliwością wynoszącą 500 Hz.

Załóżmy, że graniczna częstotliwość filtru wynosi $F_g$= 100 Hz. Oznacza to, że wyjściowe kwan-

---

[*] W tym wypadku program ten nazywa się często makroasemblerem, celem odróżnienia od programu translującego, który nie ma tych właściwości i nazywa się asemblerem.

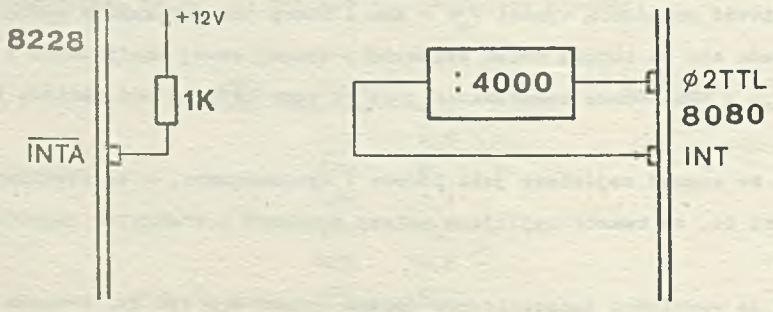
ty należy tworzyć z minimalną częstotliwością wynoszącą 200 Hz. Kwanty wyjściowe powinny być również tworzone z częstotliwością 500 Hz.




Rys.65. Kwanty sygnału wejściowego z filtru dolnoprzepustowego o górnej granicznej częstotliwości $F_G$= 4 kHz, gdy sygnał wejściowy składa się ze zbioru dwóch sinusoid o takiej samej amplitudzie, a których częstotliwości wynoszą $F_1$= 1 kHz i $F_2$= 2 kHz.

Tak więc co 2 ms należy powtarzać cały algorytm. Dlatego do mikrokomputera doprowadzimy jedno źródło przerwania, które co 2 ms będzie żądać, aby mikroprocesor pobrał nowy wejściowy kwant i ponownie zrealizował cały algorytm filtru dolnoprzepustowego. Oczywiście program dokonujący pobierania nowego kwantu wejściowego oraz realizujący odpowiedni algorytm powinien rozpoczynać się od adresu 0038H.

Jeżeli okres taktu podstawowego wynosi 500 ns, wówczas układ, który co 2 ms zgłasza żądanie przerwania, wygląda tak, jak na rys.66.




Rys.66. Układ zgłaszający żądanie przerwania co 2 ms

Dzielnik przez 4000 zrealizowany jest za pomocą 12 zwykłych przerzutników /7474/ z określonymi sprzężeniami zwrotnymi.

Należy zauważyć, że współczynniki $k_1$ oraz $k_2$ są mniejsze od 1. Znormalizujemy je do 256, wobec czego wynik Y(n) będzie 256 razy większy niż powinien być. Dlatego więc Y(n) należy podzielić przez 256. Ponieważ Y(n) składa się z dwóch bajtów, to dzielenie przez 256 jest równoznaczne z uznaniem starszego bajtu za wynik. Zostanie to wykorzystane w następnym odcinku programu.

Struktura mikrokomputera

Niech do pamięci ROM zostaną przydzielone adresy od 0000H do 03FFH, zaś pamięci RAM adresy od 0400H do 07FFH. Osiąga się to przez połączenie RAM i ROM z szyną mikrokomputera w sposób jak na rys.41.

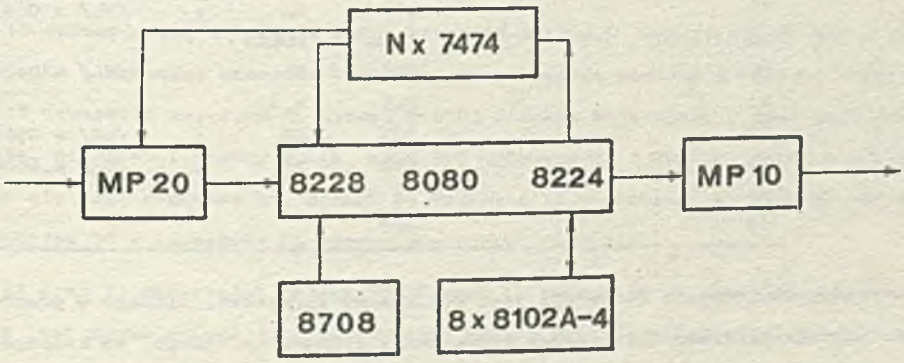
Stałe $k_1$ i $k_2$ będą wcześniej obliczone i umieszczone w pamięci ROM jako ustalone dane. Uzależnienia między poszczególnymi danymi oraz adresami przynależnych im komórek pamięciowych podane są w tab.12.

| Adres | Dana | Miejsce |
|---|---|---|
| 03FE | $k_1$ | ROM |
| 03FF | $k_2$ | ROM |
| 0400 | Y(n) | D/A |
| 0401 | Y(n-1) | RAM |
| 0402 | starszy bajt iloczynu $k_1 \cdot Y(n-1)$ | RAM |
| 0403 | młodszy bajt iloczynu $k_1 \cdot Y(n-1)$ | RAM |
| 0404 | X(n) | A/D |

Tab.12

Tak więc konwertery A/D i D/A traktowane są jak komórki pamięci z adresami odpowiednio 0404H i 0400H. Należy być bardzo ostrożnym przy traktowaniu komórki peryferyjnej na równi z komórką pamięci. Konkretnie w naszej sytuacji należy zastosować układy dekodujące, które uniemożliwią wybieranie pamięci RAM dla adresów 0401H i 0404H.

Schemat blokowy mikrokomputera, za pomocą którego realizuje się algorytm dowolnego filtra cyfrowego przedstawiony jest na rys.67.




Rys.67. Schemat blokowy mikrokomputera, za pomocą którego realizuje się algorytm filtra cyfrowego

Program

W dalszych rozważaniach wykorzystamy poprzednio utworzony podprogram dotyczący dodawania, pod nazwą DODAW. Wcześniej przedstawiony odcinek programu obejmujący mnożenie algebraiczne jest również przekształcony w podprogram. Podprogram ten będzie nosił nazwę ALGMN.

Program, za pomocą którego realizuje się algorytm filtra dolnoprzepustowego składa się z dwóch części. Pierwszą część możemy umownie nazwać programem głównym. Wykonywane są w nim czynności przygotowawcze oraz wejście do niekończącej się pętli. Program ten odpowiada programowi A w multiwibratorze monostabilnym. Druga część stanowi podprogram obsługi przerwania. Zrealizowany jest w nim algorytm filtra cyfrowego dolnoprzepustowego. Ta część programu odpowiada programowi B w multiwibratorze monostabilnym.

Główny program A ma następującą postać:

```
1   0000        LXI   DE,03FEH;   /DE/ = 03FEH
2                 LXI   HL,0400H;   /HL/ = 0400H
3                 LXI   SP,0400H
4                 XRA
5                 EI
6           A0:   NOP
7                 JMP   A0
```

Podprogram obsługi przerwania B ma następującą postać:

```
8   0038        MOV   M,A   ;
9                 INX   HL    ;   /HL/ = 0401H
10                CALL  ALGMN ;
11                INX   HL    ;   /HL/ = 0402H
12                MOV   M,B
13                INX   HL    ;   /HL/ = 0403H
14                MOV   M,C
15                INX   DE    ;   /DE/ = 03FFH
16                INX   HL    ;   /HL/ = 0404H
17                CALL  ALGMN
18                DCX   HL    ;   /HL/ = 0403H
19                CALL  DODAW
20                DCX   HL    ;   /HL/ = 0401H
21                MOV   M,A
22                DCX   DE    ;   /DE/ = 03FEH
23                DCX   HL    ;   /HL/ = 0400H
24                EI
25                RET
```

Załóżmy, że rozkaz rozpoczęcia konwersji jest wytwarzany sprzętowo. Dlatego w programie B nie przewidziano odcinka programu dotyczącego wytwarzania sygnału cyfrowego, od którego rozpoczyna się konwersja A/D. Zakłada się, że kwant w postaci cyfrowej jest przygotowany do odczytania przy każdym "zwróceniu się" mikroprocesora do konwertera A/D.

Załóżmy, że początkowe adresy podprogramów ALGMN i DODAW równe są odpowiednio 004FH i 0081H.

Dlatego rozmieszczenie w pamięci jest takie jak na rys.68. Nawiasy zapisane w komentarzach rozpatrywanego programu oznaczają zawartości rejestru wpisanego wewnątrz.

Program ten dobrze ilustruje skuteczność pośredniego adresowania.

Zakładamy, że filtr nie magazynuje energii początkowej i dlatego na samym początku programu akumulator musi być wyzerowany. Dokonuje się tego rozkazem 4.

Rozkazy 6 i 7 realizują niekończącą się pętlę. Można z niej wyjść tylko po przyjęciu żądania przerwania. Aby żądanie przerwania zostało przyjęte rozkaz 5 musi być EI.

Zamiast rozkazów NOP i JMP ADRM może być zastosowany rozkaz

HALT

Tym rozkazem zatrzymuje się mikroprocesor /Halt State/. Kod tego rozkazu jest

01110110

Przed rozkazem HALT musi również znajdować się rozkaz EI. Jeżeli nie byłoby rozkazu EI, wówczas mikroprocesor mógłby być ponownie uruchomiony tylko przez wyłączenie i ponowne włączenie zasilania.

Rozkazem 8 zawartość akumulatora przenoszona jest do konwertera D/A.

Rozkazem 10 mnoży się $k_1$ i Y/n-1/. Z tab.12 widać, że Y/n-1/ przechowywany jest w komórce pamięci, której adresem jest 0401H. Natomiast po wykonaniu rozkazu 8 zawartość rejestru HL wynosi 0400H. Dlatego też rozkazem 9 zawartość rejestru HL powiększa się o 1.

Rozkazem 12 starszy bajt wyniku umieszcza się w"bezpiecznym miejscu", tzn. w komórce pamięci z adresem 0402H. Dlatego przed wykonaniem tego rozkazu zawartość rejestru HL musi wynosić 0402H. Osiąga się to za pomocą rozkazu 11.

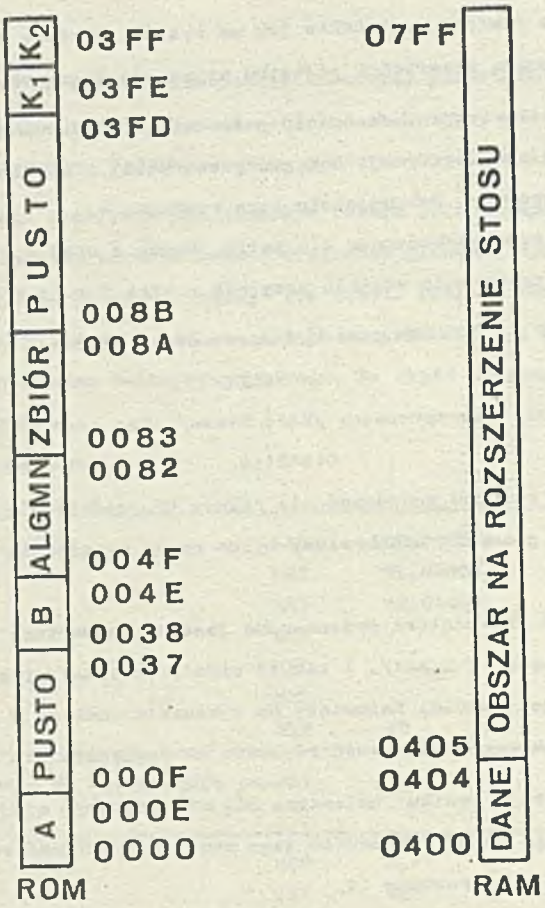
Rozkaz 14 młodszy bajt wyniku umieszcza się w "bezpiecznym miejscu", tzn. w komórce pamięci z adresem 0403H. Dlatego rozkazem 13 w rejestrze HL tworzy się zawartość 0403H.

Rozkazem 17 mnoży się $k_2$ i X/n/. Współczynnik $k_2$ znajduje się w komórce adresowej z adresem 03FFH, natomiast X/n/ znajduje się w komórce peryferyjnej z adresem 0404H /konwerter A/D/. Po wykonaniu rozkazu 15 zawartość rejestru DE wynosi 03FFH. Po wykonaniu rozkazu 16 zawartość rejestru HL wynosi 0404H.

Rozkazem 19 dokonuje się dodawania dwóch liczb 16-bitowych. Wpierw dodaje się młodsze bajty. Młodszy bajt wyniku pierwszego mnożenia znajduje się w komórce pamięci z adresem 0403H. Po wykonaniu rozkazu 18 zawartość rejestru HL wynosi 0403H. Starszy bajt wyniku, tzn. bajt przedstawiający rzeczywisty, nieznormalizowany wynik, musi być umieszczony w komórce pamięci z adresem 0401H. Dokonuje się tego rozkazem 21. Jednak po rozkazie 19 zawartość rejestru HL wynosi 0401H. Dlatego też rozkazem 20 w rejestrze HL tworzy się zawartość 0401H.

Rozkazem 22 oraz 23 zawartości rejestrów DE oraz HL powracają do stanu początkowego. W ten sposób wszystko jest znowu gotowe do przetworzenia nowego kwantu wejściowego.

Ostatni kwant Y/n/ znajduje się w akumulatorze. W momencie nadejścia następnego żądania przerwania, kwant ten zostanie wpierw przeniesiony do konwertera D/A. Dokonuje się tego rozkazem 8. Dopiero wtedy przechodzi się do przetworzenia nowego kwantu wejściowego.




Rys.68. Rozmieszczenie przestrzeni pamięciowej

Niektóre mikroprocesory mają rozkazy, za pomocą których zawartość rejestrów 16-bitowych automatycznie powiększa się /Auto Increment/ lub zmniejsza /Auto Decrement/. Niektóre mikroprocesory mają również rozkazy dotyczące automatycznego powiększania /zmniejszania oraz warunkowego skoku (Auto Increment/Decrement and Skip on Condition). Za pomocą takich rozkazów adresowanie pośrednie jest jeszcze bardziej skuteczne, gdyż rozkazy INX RP i DCX RP nie są wówczas potrzebne. Mikroprocesor Intel 8080 nie ma, niestety, takich rozkazów.

LITERATURA

A1, C3, D1, D6, F1, F2, K1, K2.

## MODEM DO TRANSMISJI DANYCH - 2400 b/s

Sygnał danych w zakresie podstawowym /Baseband Signal/ składający się z ciągu prostokątnych impulsów jest nieodpowiedni do przesyłania przez kanał telefoniczny /Voice-Grade Channel/.

Po pierwsze, zasadnicza część widma, czyli spektralnej gęstości mocy tego sygnału, znajduje się w niskich częstotliwościach. Po drugie, teoretyczne widmo tego sygnału jest nieskończenie szerokie. Z drugiej strony kanał telefoniczny zachowuje się jak filtr przepuszczający ograniczony zakres niskich, ale nie najniższych częstotliwości. Tak więc sygnał, którym się zajmujemy, musimy dostosować do środków przesyłu. W tym celu przy nadawaniu przeprowadza się modulację i kształtowanie widma. Oczywiście przy odbiorze muszą być dokonywane demodulacje. Urządzenie, za pomocą którego przeprowadza się demodulacje nazywa się modemem /MOdulator DEModulator/ i będzie przedmiotem obecnych rozważań.

### CZTEROWSTĘGOWA RÓŻNICOWA MODULACJA FAZOWA

Przy średnich szybkościach przesyłania danych /od 2400 b/s do 4800 b/s/ najczęściej stosuje się różnicową modulację fazy /Differential Phase Shift Keying, DPSK/. Czterowstęgowa /Quaternary/ różnicowa modulacja fazowa /QDPSK/ zapewnia optymalny stosunek między ozułością na zakłócenia a liczbą przesyłanych informacji.

W wypadku czterowstęgowej różnicowej modulacji fazy, przy modulacji rozpatruje się dwa sąsiednie bity, tzn. jeden dibit /Dibit/. W zależności od wartości tego dibitu do sygnału liniowego wprowadza się określone przesunięcia fazowe /dP/. W tab.13 przedstawiono schemat modulacyjny stosowany w praktyce /CCITTV.26 bis, Alternative A/.

| Dibit | $dP^{(o)}$ |
|---|---|
| 00 | 0 |
| 01 | +90 |
| 11 | +180 |
| 10 | +270 |

Tab.13.

Jeżeli częstotliwość bitów wynosi

$$f_B = 1/T_B = 2400 \text{ b/s}$$

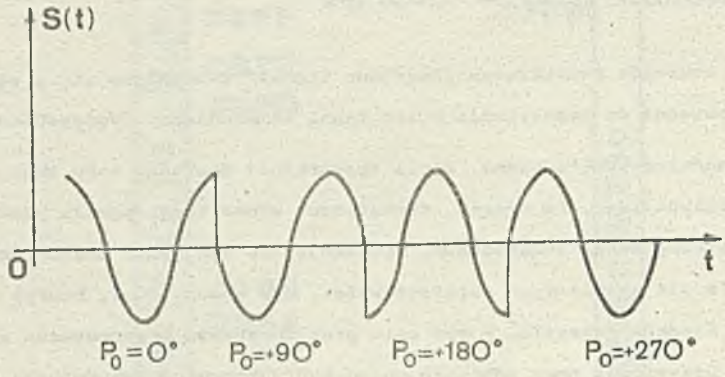
wówczas częstotliwość dibitów wynosi

$$f_D = 1/T_D = 1200 \text{ b/s}$$

Dyskretny skok fazy na granicach przedziałów dibitowych bardzo niekorzystnie odbija się na szerokości widma sygnału liniowego, nawet gdy obwiednia dibitu ma kształt podniesionego cosinusa /Rised Cosine/. Zmodulowany sygnał z dyskretnymi skokami fazy na granicach dibitowych przedstawiony jest na rys.69.

Symbol $P_0$ na rys.68 dotyczy początkowej fazy zmodulowanego sygnału w jednym przedziale dibitowym.

Szerokość widma sygnału liniowego może ulec zmniejszeniu, jeżeli wprowadzi się pewne nakładanie się przebiegów czasowych z sąsiednich przedziałów dwubitowych. Dla oznaczenia tych przebiegów czasowych w dalszym tekście używa się terminu symbol dibitowy. Tak więc pod pojęciem "symbol dibitowy" rozumie się dibit zmodulowany. Częściowe nakładanie się sąsiednich symboli dibito-

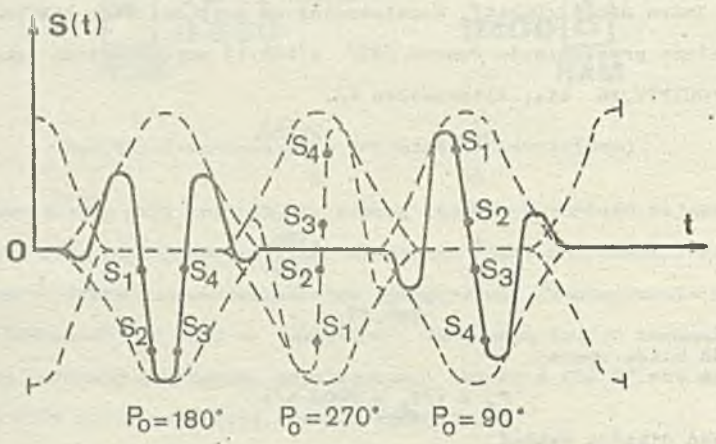



Rys.69. Zmodulowany sygnał ze skokowymi zmianami fazy na granicach przedziałów dibitowych dla czterowstęgowej różnicowej modulacji fazy

wych przedstawione jest na rys.70. Analityczne przedstawienie symboli dibitowych z rys.70 jest następujące:

$$S(t)=\begin{cases}\left\{b+\sin\left[\left(\frac{2\pi}{T_E}\right)(t-c)\right]\right\}\cdot\cos\left[\omega_o t+(n-1)\frac{\pi}{2}\right]; & 0\leqslant t\leqslant T_E-2c\\ 0; & T_E-2c\leqslant t\leqslant T_E\end{cases}$$

$n = 1,2,3,4.$

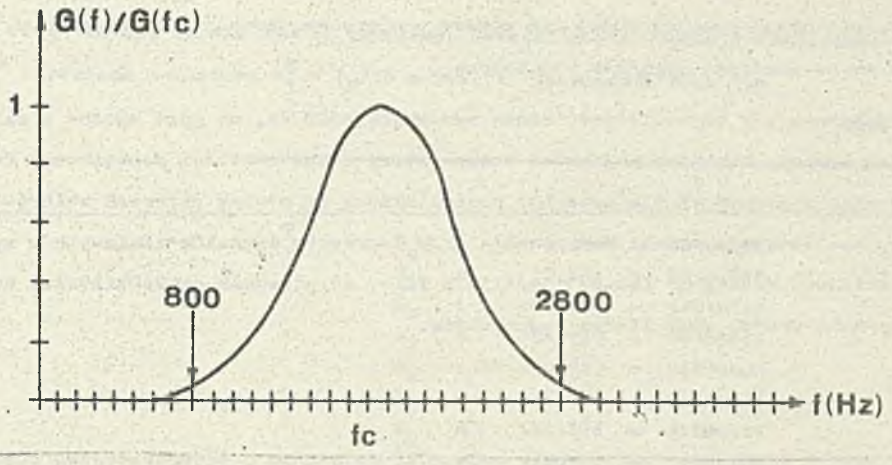



Rys.70. Zmodulowany sygnał z dibitową obwiednią o kształcie podniesionego cosinusa, gdy sąsiednie symbole dibitowe nakładają się przy czterowstęgowej różnicowej modulacji fazy /B2/

Stałe b i c definiują szerokość przedziału, w którym sąsiednie symbole dibitowe nakładają się /B5/. Stała $T_E = 1/f_E$ dotyczy okresu obwiedni sygnału na rys.70. Dla sygnału z rys.70 spełniona jest następująca relacja:

$$T_E = 2\,T_D = 4\,T_B$$

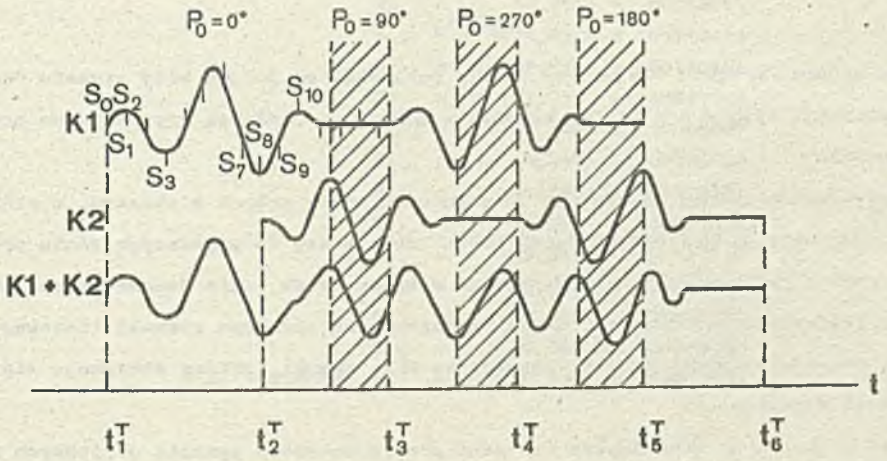
Stała $\omega_o = 2\pi f_o$ przedstawia prędkość kątową częstotliwości nośnej. Przez wprowadzenie nakładania się między sąsiednimi symbolami dibitowymi realizuje się stałą zmianę fazy na granicach przedziałów dibitowych, co odbija się na spektralnej gęstości mocy, której znormalizowany kształt $G(f)G(f_o)$ przedstawiony jest na rys.71.




Rys.71. Znormalizowana spektralna gęstość mocy zmodulowanego sygnału z obwiednią dibitową o kształcie podniesionego cosinusa oraz ze stałymi zmianami fazy na granicach przedziałów dibitowych przy czterowstęgowej różnicy modulacji fazy /B5/

Jeżeli do nakładania się sąsiednich symboli dibitowych dochodzi w 50% czasu, jak ma to miejsce na rys.70, wówczas ponad 99% mocy sygnału liniowego znajduje się w obszarze między 800hZ a 2800Hz. Widać to na rys.71.

Aby zrealizować nakładanie, o którym mówimy, sąsiednie dibity modulowane są w różnych kanałach, zaś sygnał liniowy otrzymuje się po zsumowaniu sygnałów z tych kanałów. Przedstawiono to na rys.72. Kanały te oznaczone są symbolami K1 i K2, zaś sygnał liniowy oznaczony jest sygnałem K1 + K2.




Rys.72. Sposób tworzenia sygnału liniowego /C6/

Obwiednia sygnałów w kanałach, tzn. przed zsumowaniem ma częstotliwość /$f_E$/, która jest, jak to powiedzieliśmy, dwa razy niższa od częstotliwości dibitowej /$f_D$/. W naszym przykładzie częstotliwość obwiedni wynosi 600 Hz.

Zgodnie z teorią kwantowania /Sampling Theorem/ sygnał, którego spektrum leży poniżej 2800 Hz może być jednoznacznie przedstawiony za pomocą próbek, których częstotliwość wynosi 5600 Hz. Teoretycznie oznacza to, że wystarczy 2,3 próbki na jeden przedział bitowy. Z przyczyn

praktycznych, nie można przyjąć mniej niż cztery pomiary na jeden przedział bitowy. Oznacza to 16 próbek na okres obwiedni sygnałów kanałowych.

Jeżeli dobierze się częstotliwość nośną wynoszącą 1800 Hz, co jest zgodne z zaleceniami CCITT, wówczas sygnały kanałowe mogą mieć tylko cztery różne wartości początkowej fazy $P_0$. Tak więc dowolny stan w jednym kanale może być przedstawiony za pomocą czterech zbiorów po 16 próbek. Innymi słowy, zastosowanie teorii kwantowania przy tworzeniu sygnałów liniowych w wypadku czterowstęgowej różnicowej modulacji fazowej umożliwia fakt, że stosunek częstotliwości nośnej do częstotliwości sygnału danych jest liczbą rzeczywistą.

Nadajnik

Opisany sposób umożliwia realizowanie nadajnika do modemu z czterowstęgową różnicową modulacją fazy za pomocą stosunkowo wolnego mikroprocesora Intel 8080 wykonanego w technologii MOS.

Zakodowane wartości czterech grup próbek /po 16 próbek w każdej grupie/ przechowywane są w pamięci ROM. Są to obszary $A_T$, $B_T$, $C_T$ i $D_T$ na rys.76.

W tab.14 przedstawiono zależności między obszarami w pamięci ROM a początkowymi fazami symbolu dibitowego, którego próbki znajdują się w tej pamięci.

| Obszar w pamięci ROM | $P_0$ (°) |
|---|---|
| $A_T$ | 0 |
| $B_T$ | +90 |
| $C_T$ | +180 |
| $D_T$ | +270 |

Tab.14.

Z wejścia do modemu /przewód 103 według CCITT/ pobierane są po dwa bity sygnału danych. W zależności od kombinacji tych dwóch bitów, zgodnie z tab.13 na linię są wyprowadzane próbki z odpowiedniego obszaru.

Pierwszych osiem próbek dodaje się do ostatnich ośmiu próbek z obszaru, w którym program znajdował się uprzednio. Następnych osiem próbek dodaje się do pierwszych ośmiu próbek z obszaru, do którego program "skoczy" po detekcji nowego dibitu. Przez takie dodawanie realizuje się nakładanie sąsiednich symboli dibitowych, czyli kształtowanie spektrum sygnału liniowego. Tak utworzone próbki są następnie wprowadzane do konwertora D/A. Sygnał liniowy otrzymuje się po interpolacji tych próbek wyjściowych.

W tab.15 umieszczono skwantowane wartości próbek czterech symboli dibitowych z rys.72. Wartości te podane są w dziesiętnym systemie liczbowym oraz w postaci binarnej z uzupełnieniem dwójkowym. Otrzymane są z analitycznej postaci dla symboli dibitowych z rys.70.Symbolem $S_i/i=0,\ldots,15/$ oznaczono kolejne numery próbek. W tab.15 oznaczone są również miejsca z rys.72, odpowiadające poszczególnym próbkom. Pozycje wszystkich 16 próbek w symbolu dibitowym z początkową fazą $P_0 = 0^\circ$ oznaczone są na rys.72 pionowymi kreskami.

Przyjęto, że pierwsza próbka symbolu dibitowego w kanale K1 tzn., symbolu kanałowego z początkową fazą $P_0 = 0^\circ$ odpowiada momentowi $t_1^T$ z rys.72. Dla symbolu dibitowego związanego z kana-

łem będzie stosować się termin symbol kanałowy. Ostatni pomiar symbolu kanałowego z początkową fazą $P_0 = 0^o$ odpowiada momentowi $t_3^T - T_E/16$ z rys.72. Obowiązuje przy tym

Tab.15

| | | | | |
|---|---|---|---|---|
| | | | $P_o = 0^o$ | |
| $s(t_1^T)$ | $= s_o$ | = | 0.000 | = 00000000 |
| | $s_1$ | = | 10.461 | = 00001010 |
| | $s_2$ | = | -42.045 | = 11010110 |
| | $s_3$ | = | -84.793 | = 10101011 |
| | $s_4$ | = | -0.223 | = 00000000 |
| | $s_5$ | = | 126.704 | = 01111111 |
| | $s_6$ | = | 101.886 | = 01100110 |
| | $s_7$ | = | -52.117 | = 11001100 |
| | $s_8$ | = | -119.031 | = 10001001 |
| | $s_9$ | = | -35.460 | = 11011101 |
| | $s_{10}$ | = | 41.886 | = 00101010 |
| | $s_{11}$ | = | 25.281 | = 00011001 |
| | $s_{12}$ | = | 0.000 | = 00000000 |
| | $s_{13}$ | = | 0.000 | = 00000000 |
| | $s_{14}$ | = | 0.000 | = 00000000 |
| $s(t_3^T - T_E/16) =$ | $s_{15}$ | = | 0.000 | = 00000000 |
| | | | $P_o = +90^o$ | |
| $s(t_2^T)$ | $= s_o$ | = | 0.000 | = 00000000 |
| | $s_1$ | = | -25.223 | = 11100111 |
| | $s_2$ | = | -42.124 | = 11010110 |
| | $s_3$ | = | 34.983 | = 00100011 |
| | $s_4$ | = | 114.032 | = 01110111 |
| | $s_5$ | = | 52.831 | = 00110101 |
| | $s_6$ | = | -101.314 | = 10011011 |
| | $s_7$ | = | -127.000 | = 10000001 |
| | $s_8$ | = | -0.447 | = 00000000 |
| | $s_9$ | = | 84.595 | = 01010101 |
| | $s_{10}$ | = | 42.281 | = 00101010 |
| | $s_{11}$ | = | -10.319 | = 11110110 |
| | $s_{12}$ | = | 0.000 | = 00000000 |
| | $s_{13}$ | = | 0.000 | = 00000000 |
| | $s_{14}$ | = | 0.000 | = 00000000 |
| $s(t_4^T - T_E/16) =$ | $s_{15}$ | = | 0.000 | = 00000000 |
| | | | $P_o = +180^o$ | |
| $s(t_4^T)$ | $= s_o$ | = | 0.000 | = 00000000 |
| | $s_1$ | = | -10.461 | = 11110110 |
| | $s_2$ | = | 42.045 | = 00101010 |
| | $s_3$ | = | 84.793 | = 01010100 |
| | $s_4$ | = | 0.223 | = 00000000 |
| | $s_5$ | = | -126.704 | = 10000001 |
| | $s_6$ | = | -101.886 | = 10011001 |
| | $s_7$ | = | 52.117 | = 00110100 |
| | $s_8$ | = | 119.031 | = 01110111 |
| | $s_9$ | = | 35.460 | = 00100011 |

| | | | | |
|---|---|---|---|---|
| | $S_{10}$ | = | -41.886 | = 11010110 |
| | $S_{11}$ | = | -25.000 | = 11100111 |
| | $S_{12}$ | = | 0.000 | = 00000000 |
| | $S_{13}$ | = | 0.000 | = 00000000 |
| | $S_{14}$ | = | 0.000 | = 00000000 |
| $S(t_6^T - T_E/16) =$ | $S_{15}$ | = | 0.000 | = 00000000 |

$P_o = +270^o$

| | | | | |
|---|---|---|---|---|
| $S(t_3^T) =$ | $S_o$ | = | 0.000 | = 00000000 |
| | $S_1$ | = | 25.223 | = 00011001 |
| | $S_2$ | = | 42.124 | = 00101010 |
| | $S_3$ | = | -34.983 | = 11011101 |
| | $S_4$ | = | -119.032 | = 10001001 |
| | $S_5$ | = | -52.832 | = 11001011 |
| | $S_6$ | = | 101.314 | = 01100101 |
| | $S_7$ | = | 127.000 | = 01111111 |
| | $S_8$ | = | 0.447 | = 00000000 |
| | $S_9$ | = | -84.595 | = 10101011 |
| | $S_{10}$ | = | -42.281 | = 11010110 |
| | $S_{11}$ | = | 10.319 | = 00001010 |
| | $S_{12}$ | = | 0.000 | = 00000000 |
| | $S_{13}$ | = | 0.000 | = 00000000 |
| | $S_{14}$ | = | 0.000 | = 00000000 |
| $S(t_5^T - T_E/16) =$ | $S_{15}$ | = | 0.000 | = 00000000 |

$$t_3^T - t_1^T = T_E$$

Jak już podkreślono, symbol $T_E = 1/f_E$ dotyczy okresu obwiedni sygnału kanałowego. Wartości dla symbolu kanałowego z początkową fazą $P_0 = +90^o$ dotyczą przedziału czasowego między momentami $t_2^T$ i $t_4^T - T_E/16$ z rys.72. Wartości dla symbolu kanałowego z początkową fazą $P_0 = +270^o$ dotyczą przedziału czasowego między momentami $t_3^T$ i $t_5^T - T_E/16$ z rys.72. Wartości dla symbolu kanałowego z początkową fazą $P_o = +180^o$ dotyczą przedziału czasowego między momentami $t_4^T$ i $t_6^T - T_E/16$. Ponieważ nakładanie się symboli kanałowych wykonywane jest w 50% czasu, to ostatnie cztery próbki każdego symbolu kanałowego muszą być równe zeru /rys.72/.

Źródło danych musi być zsynchronizowane z modemem. Taka synchronizacja może być przeprowadzona dwoma sposobami. Urządzenie końcowe może być synchronizowane z zewnątrz przez modem lub też modem może być z zewnątrz synchronizowany przez urządzenie końcowe. Proponujemy pierwszy sposób. Dlatego też mikrokomputer na specjalnym wyjściu /przewód 114 według CCITT/ powinien generować sygnał taktowy o częstotliwości 2400H i o określonych wymaganiach. Sygnał ten zawiera dokładną informację o częstotliwości i fazie początkowej oscylatora krystalicznego w mikrokomputerze i może być wykorzystany do omawianej synchronizacji.

DEMODULACJA SYGNAŁU PO MODULACJI RÓŻNICOWEJ FAZY

Demodulacja sygnału po modulacji różnicowej fazy może być dokonywana kilkoma sposobami. Jeden z nich opiera się na zastosowaniu odpowiednio dopasowanych filtrów. W modulacji czterowstęgowej, na wejściu muszą znajdować się cztery odpowiednio dopasowane filtry. Parametrami

takiego filtru są: obwiednia, częstotliwość oraz faza sygnału, do której filtr jest dopasowany. W naszym przykładzie filtry przystosowane są do obwiedni typu podniesionego cosinusa, do częstotliwości nośnej wynoszącej 1800Hz oraz do początkowych faz $0^{o}$, $+90^{o}$, $+180^{o}$ i $+270^{o}$.

Sygnały na wyjściach wszystkich czterech dopasowanych filtrów osiągają maksimum na końcu przedziału dibitowego, z których jedno jest największe. Dopasowany filtr, na wyjściu którego pojawia się największe maksimum jednoznacznie definiuje początkową fazę sygnału w poprzednim przedziale dibitowym.

Ponieważ chodzi o modulację różnicową, to za dopasowanymi filtrami musi znajdować się układ logiczny, który na podstawie różnic faz ostatnich dwóch detektowanych symboli dibitowych ustala informacje o dibicie, który jest rzeczywiście przekazywany. Granice przedziałów dibitowych badane są przez określenie średnich chwil, w których pojawiają się największe maksima na wyjściach dopasowanych filtrów. W ten sposób dokonuje się synchronizacji odbiornika i nadajnika.

Jak już podkreślono, mikroprocesory MOS są za wolne dla cyfrowego filtrowania sygnału o względnie szerokim zakresie częstotliwości, jakim jest różnicowo modulowany w fazie sygnał o częstotliwości nośnej wynoszącej 1800 Hz. Z tego powodu będziemy dalej rozpatrywać uproszczoną postać dopasowanego filtrowania, tzn. binarnie dopasowane filtrowanie.

Przedział, w którym nie ma nakładania się symboli kanałowych trwa u nas 416 µs. Przedział ten jest na rys.72 zakreskowany. W naszym przykładzie sygnał liniowy próbkuje się cztery razy w ciągu tego przedziału. Te miejsca pomiaru są zaznaczone na rys.70 punktami i symbolami $S_i$ /i = 1,2,3,4/. Ich znaki jednoznacznie definiują początkową fazę sygnału liniowego w tym samym przedziale dibitowym. Zależności między wartościami znaków tych próbek a początkową fazą sygnału liniowego w tym samym przedziale dibitowym przedstawione są w tab.16.

Tab.16

| $P_o$ | $S_1$ | $S_2$ | $S_3$ | $S_4$ |
|---|---|---|---|---|
| $0^{o}$ | + | + | + | + |
| $+90^{o}$ | + | + | - | - |
| $+180^{o}$ | - | - | - | - |
| $+270^{o}$ | - | - | + | + |

Tak więc binarnie dopasowane filtrowanie sprowadza się do próbkowania sygnału liniowego w określonych momentach i do porównywania rozmieszczenia znaków tych pomiarów z uporządkowanymi czwórkami znaków z tab.16. Gdy w kanale nie ma szumu i innych zakłóceń, rozmieszczenie znaków pomiarów próbnych sygnału wejściowego pokryje się z jedną z czterech uporządkowanych czwórek znaków z tab. 16. Taka zbieżność jednoznacznie wskazuje na fazę sygnału liniowego w trakcie danego przedziału dibitowego.

Detekcja oparta na znakach tylko czterech pomiarów jest bez wątpienia bardzo uczulona na działanie szumu i innych zakłóceń w kanale. Z tego powodu można oczekiwać względnie dużego prawdopodobieństwa błędu przy odbiorze. Jednak nie ma żadnych przeszkód teoretycznych, aby powiększyć liczbę próbek, na podstawie znaków, którymi dokonuje się detekcji początkowej fazy. Istnieje tylko jedno niebezpieczeństwo przy zbyt dużym powiększeniu liczby tych próbek, mianowicie, mikroprocesor MOS może wówczas nie być wystarczająco szybki dla omawianego przetwarzania.

Odbiornik

Opisany sposób umożliwia również realizację odbiornika modemu z czterowstęgową różnicową modulacją fazy za pomocą względnie wolnego mikroprocesora Intel 8080 wykonanego w technologii MOS.

Znaki mierzonych próbek, jednoznacznie definiujące początkową fazę sygnału liniowego, przechowywane są w pamięci ROM. Na rys.76 jest to obszar U. Format i zawartość obszaru U przedstawione są na rys.73. Obszar U składa się z czterech bajtów, ponieważ mamy do czynienia z modulacją czterowstęgową. Przyjęto, że zawartość czterech najstarszych pozycji bitowych w każdym z czterech bajtów z obszaru U odpowiada zawartości tab.16.

| | | |
|---|---|---|
| 1 1 1 1 X X X X | 00 E A | $P_0 = 0°$ |
| 0 0 1 1 X X X X | 00 E B | $P_0 = +90°$ |
| 0 0 0 0 X X X X | 00 E C | $P_0 = +180°$ |
| 1 1 0 0 X X X X | 00 E D | $P_0 = +270°$ |

Rys.73. Format i zawartość obszaru U

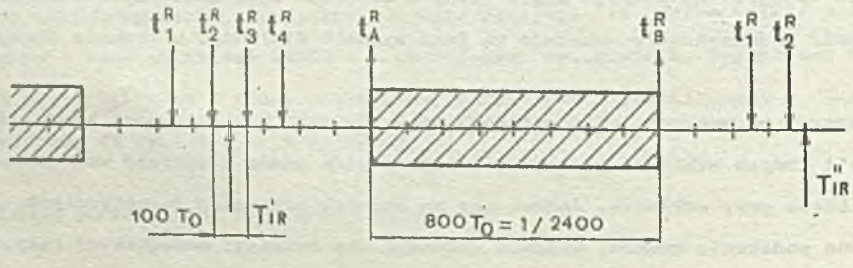
Bit 1 na rys.73 oznacza znak plus w tab.16, zaś bit 0 znak minus. Zawartość czterech najmłodszych pozycji bitowych jest nieistotna i oznaczona jako XXXX.

Przyjęto, że najstarszy bit w obszarze U odpowiada znakowi czwartego, tzn. ostatniego pomiaru sygnału liniowego w jednym przedziale dibitowym i dalej, kolejno. Tak więc znaki próbek, oznaczonych na rys.70 symbolem $S_1$ znajdują się w pozycjach bitowych $b_7$, znaki próbek $S_3$ w pozycjach bitowych $b_6$ itd.

Dla uporządkowanego zbioru czterech bitów stosuje się również termin nibl /Nibble/. Tak więc jeden bajt składa się z dwóch nibli.

Na wejściu do modemu znajduje się obecnie zwykły komparator. Załóżmy, że ten komparator na swoim wyjściu daje zero, gdy sygnał wejściowy jest ujemny, a jedność, gdy sygnał wejściowy jest dodatni. Za jego pośrednictwem w określonych momentach, pobierane są znaki sygnału liniowego i porównywane są ze znakami czterech zbiorów zapamiętanych wzorów. Jak już podkreślono, cztery koincydencje znaków jednoznacznie wskazują początkową fazę sygnału liniowego w poprzednim przedziale liniowym. W ten sposób realizowany jest odbiornik z binarnie dopasowanymi filtrami. Następujące dalej różnicowe dekodowanie nie sprawia trudności, ponieważ mikroprocesory są bogato wyposażone w rozkazy, za pomocą których realizuje się warunkowe skoki programowe.

Działanie odbiornika w funkcji czasu przedstawione jest na rys.74.




Rys.74. Działanie odbiornika w funkcji czasu

Zakreskowane części na rys.74 odnoszą się do przedziałów, w których dochodzi do nakładania się sąsiednich symboli kanałowych. Jeżeli podstawowa częstotliwość oscylatora kwarcowego wynosi 17, 28 MHz, wówczas przedziały te trwają dokładnie przez 800 okresów taktu podstawowego /800 $T_0$/. Momenty $T'_{IR}$ i $T''_{IR}$ na rys.74 oznaczają środki przedziałów, w których nie ma nakładania się sąsiednich symboli dibitowych, natomiast $t_1^R$, $t_2^R$, $t_3^R$ i $t_4^R$ oznaczają momenty, w których testuje się znak sygnału wejściowego. Oczywiście, testowanie znaku sygnału wejściowego musi być dokonywane w przedziale, w którym nie ma nakładania się sąsiednich symboli dibitowych. Dlatego momenty $t_1^R$, $t_2^R$, $t_3^R$ i $t_4^R$ skoncentrowane są na rys.74 w niezakreskowanym przedziale.

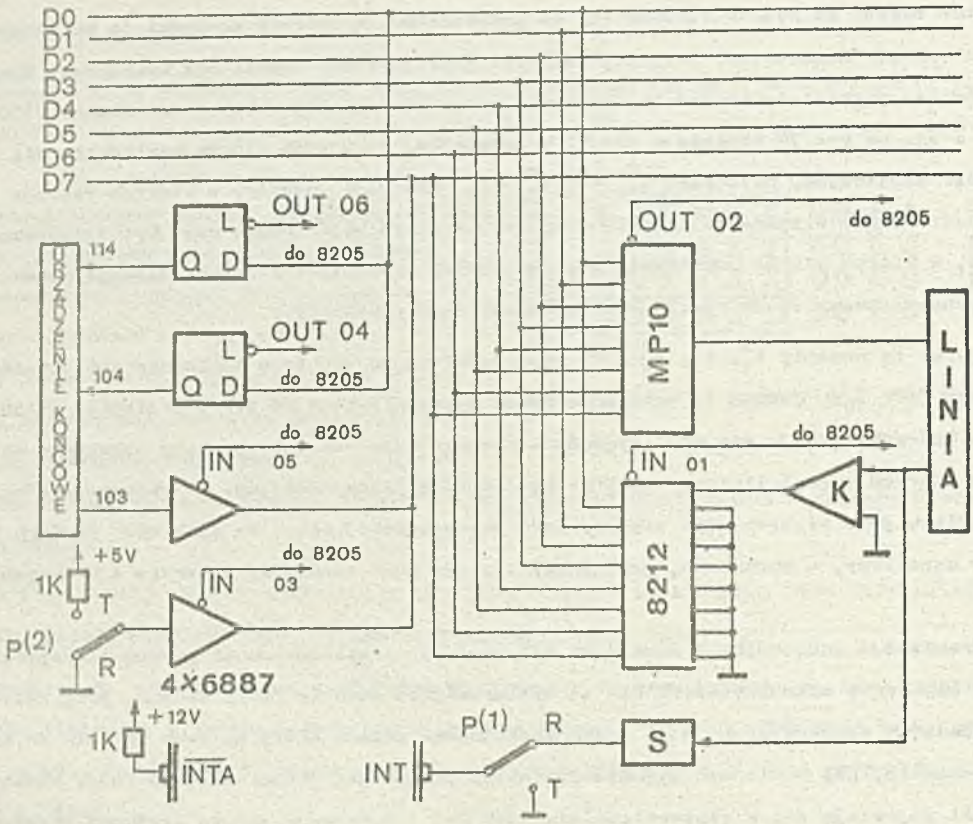
Przyjęto, że momenty $t_1^R$, $t_2^R$, $t_3^R$ i $t_4^R$ będą oddalone od siebie o dokładnie 100 okresów taktu podstawowego /100 $T_0$/. Jednak tę odległość można zoptymalizować /C 112/. Najlepiej byłoby, aby momenty, w których testuje się znak sygnału liniowego były tak dobrane, aby pokrywały się z momentami, w których sygnał liniowy ma przeciętnie największą amplitudę. W rzeczywistości, im amplituda pomiaru jest większa, tym większe jest prawdopodobieństwo, że znak tego pomiaru będzie prawidłowo określony, w warunkach, gdy sygnał liniowy jest zanurzony w szumie i w innych zakłóceniach.

Synchronizacja odbiornika i nadajnika nie może być zrealizowana za pomocą mikroprocesora MOS. Nawet bipolarne mikroprocesory nie są wystarczająco szybkie /C4/. Jednak, przy zastosowaniu kilku wzmacniaczy operacyjnych /741/ można zrealizować układ, który na swym wyjściu da dodatnie zbocze w momencie, gdy obwiednia sygnału liniowego przechodzi przez maksimum /B3/. Momenty maksimum obwiedni pojawiają się z częstotliwością 1200 Hz i pokrywają się ze środkami przedziałów, w których nie ma nakładania się sąsiednich symboli kanałowych. Widać to na rys.70 i 72. Rozpatrywany sygnał zawiera dokładną informację o częstotliwości i fazie przyjętego sygnału i może być wykorzystany do synchronizacji odbiornika i nadajnika. W tym sensie mikroprocesor po momencie $t_2^R$ na rys.74 zatrzymuje się, tzn. wchodzi w stan zatrzymania /Halt State/. Dokonuje się to programowo, tzn. istnieje rozkaz, po wykonaniu którego mikroprocesor zatrzymuje się. Jest to rozkaz HALT, o którym już mówiliśmy.

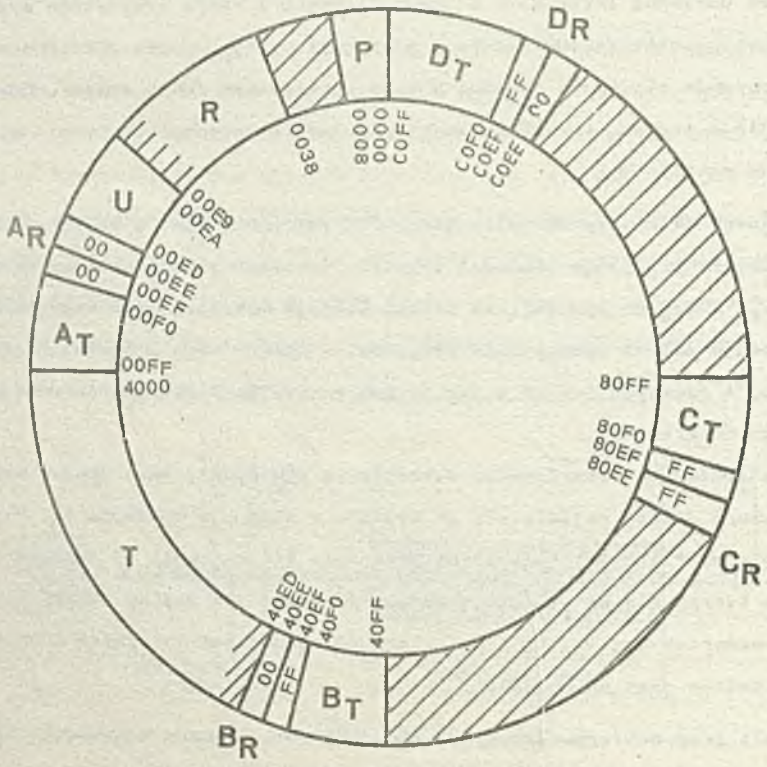
Wyjście wspomnianego układu synchronizującego /S/ połączone jest z wejściem INT układu 8080 /rys.75/. Dlatego w chwili przejścia obwiedni sygnału liniowego przez maksimum nadchodzi żądanie przerwania /momenty $T'_{IR}$ i $T''_{IR}$ na rys.74/, za pomocą którego mikroprocesor wyprowadza się ze stanu zatrzymania i powoduje dalsze wykonywanie programu. W ramach tego programu nieco później testuje się znak trzeciej i czwartej próbki z tab.16 lub z rys.70. W ten sposób odbiornik i nadajnik są absolutnie zsynchronizowane.

Odcinek programu służący do różnicowego dekodowania wykonywany jest między momentami $t_4^R$ i $t_A^R$. Drugi bit tego samego dibitu pojawia się na wyjściu z modemu w momencie $t_D^R$. Między tymi dwoma momentami mikroprocesor "liczy" i "czeka" na przejście 416 $\mu$s czyli 80 okresów taktu podstawowego. Zawartość, która tworzy się na wyjściu z modemu /przewód 104 według CCITT/ w momentach $t_A^R$ i $t_D^R$ znajduje się w obszarach $A_R$, $B_R$, $C_R$ i $D_R$. Wielkość tych obszarów wynosi 2 bajty. Zawartość tych obszarów przedstawiona jest na rys.76.

Na przykład, jeżeli przy odbiorze detektuje się dibit 01, wówczas w momencie $t_A^R$ "wyrzuci się" na szynę danych pierwszy bajt z obszaru $D_R$, tzn. bajt 00H, a w momencie $t_D^R$ drugi bajt z obszaru




Rys.75. Jednostki peryferyjne opisanego mikrokomputera pracującego jako modem




Rys.76. Wykorzystanie pamięci ROM

$D_R$, tzn. bajt FFH. Przy tym najmłodsze bity obu tych bajtów /odpowiednio 0 i 1/ pojawiają się na przewodzie 104, tzn. na wyjściu modemu. Przewód 104 jest pośrednio połączony z linią DO na szynie danych. Widać to na rys.75.

W naszym przykładzie czas między momentami $t_A^R$ i $t_B^R$ jest bezużytecznie stracony. Jednak gdyby zaistniała potrzeba mógłby on być wykorzystany do innych celów.

KONFIGURACJA MIKROKOMPUTERA

Schemat elektryczny mikrokomputera pracującego jako modem jest w zasadzie taki sam jak schemat z rys.9. Jedyna różnica polega na tym, że obecnie posiadamy inne jednostki peryferyjne wejścia-wyjścia oraz, że pamięci ROM przydzielono inny zbiór adresów.

Jednostki peryferyjne

Jednostki peryferyjne opisanego mikrokomputera, jak również układ synchronizujący /S/ przedstawione są na rys.75. Zamiast konwertera A/D na wejściu do modemu od strony liniowej znajduje się zwykły komparator /K/ i jeden układ 8212 /adres 01D/. Można je zastąpić komparatorem o trzech stanach /Motorola MC 1414/. Na wyjściu z modemu od strony linii znajduje się konwerter D/A, np. uprzednio opisany MP10 lub inny /adres 02D/. Na wyjściu z modemu od strony urządzenia końcowego znajduje się przerzutnik L /adres 04D/. Na wejściu do modemu od strony urządzenia końcowego znajduje się bufor o trzech stanach /adres 05D/. Sygnał dla synchronizacji urządzenia końcowego przechodzi przez przerzutnik L /adres 06D/.

Pamięć RAM

Pamięć RAM w naszym przykładzie nie jest potrzebna, ponieważ mikroprocesor Intel 8080 wyposażony jest w wystarczającą liczbę rejestrów. Tak więc, mikroprocesor Intel jest szczególnie użyteczny dla tego konkretnego zastosowania. Natomiast, np. mikroprocesor Motorola 6800 ma tylko dwa rejestry i dwa akumulatory pierwotne.

Przełącznik

Stan przełącznika P na rys.75 określa czy modem pracuje jako odbiornik /położenie R/ czy jako nadajnik /położenie T/. Przełącznik P ma dwa styki: $P^{/1/}$ i $P^{/2/}$.

Środkowy koniec styku $P^{/1/}$ jest połączony z wejściem INT w układzie 8080. Gdy modem pracuje jako odbiornik, wejście INT jest połączone z wyjściem układu synchronizującego S. Jeżeli modem pracuje jako nadajnik, wówczas nie obsługuje się żądania przerwania i wejście INT jest połączone z masą.

Styk $P^2$ bezpośrednio wpływa na wybór rodzaju pracy modemu. Jeżeli styk $P^2$ znajduje się w położeniu R, wówczas wejście bufora o trzech stanach, którego adresem jest 03D jest w stanie zera logicznego, Jeżeli styk $P^2$ znajduje się w położeniu T, wówczas wejście jest w stanie jedynki logicznej. Po uruchomieniu modemu najpierw dokonuje się testowania stanu przełącznika P.

Program testujący stan przełącznika P znajduje się w obszarze P na rys.76. Jeżeli przełącznik P znajduje się w położeniu R, wówczas następuje skok do początku programu realizującego nadawanie /obszar R na rys.76/.

Tak więc nasz modem nie może jednocześnie odbierać i nadawać,tzn. przeznaczony jest do pracy półdupleksowej.

Pamięć ROM

Pamięć ROM połączona jest z szyną adresową i szyną sterującą /rys.25/. Dlatego przydzielone są jej następujące adresy:

| | | |
|---|---|---|
| 0000H | do | 00FFH |
| 4000H | do | 40FFH |
| 8000H | do | 80FFH |
| C000H | do | C0FFH |

Wykorzystanie pamięci ROM /Memory Mapping/ przedstawiono na rys.76. Długość rejestru HL jest skończona i wynosi, jak już powiedzieliśmy, 16 bitów. Jeżeli zawartość tego rejestru jest równa FFFFH, wówczas po dodaniu liczby 0001H zawartość ta staje się równa 0000H, czyli pamięć ROM może być rozpatrywana jako pole, w którym obowiązuje dodawanie z modułem 65536D /Modulo 65,536 Addition/. Jest to powód, dla którego pamięć ROM jest przedstawiona na rys.76 w kształcie zamkniętego pierścienia.

Rekapitulując: w obszarze P znajduje się program testujący stan przełącznika P. W obszarach T i R znajdują się programy realizujące odpowiednio nadawanie i odbiór. W obszarach $A_T$, $B_T$, $C_T$ i $D_T$ znajdują się próbki symboli dibitowych. W obszarze U znajdują się wzory znaków sygnału liniowego, zaś w obszarach $A_R$, $B_R$, $C_R$ i $D_R$ znajdują się po dwa bajty, których najmłodsze bity odpowiadają detektowanym dibitom. Zakreskowane części pamięci ROM z rys.76 nie są używane. Kompletny program dla modemu QDPSK wymaga mniej niż 1 K miejsca w pamięci.

PROGRAM

Po włączeniu zasilania w pierwszej kolejności testuje się stan przełącznika P. Następnie, w zależności od stanu przełącznika P przechodzi się do programu odbiorczego lub nadawczego.

Testowanie stanu przełącznika P

W naszym przykładzie, po włączeniu zasilania wykonywany jest rozkaz, którego pierwszy /a może i jedyny/ bajt znajduje się w komórce pamięci z adresem 0000H. Dlatego początkowy adres programu testującego stan przełącznika P musi być 0000H. Program ten jest następujący:

```
        IN      03D
        JM      4000H
        LXI     HL,00EEH
        MVI     B,11000000B
A0:     EI
        HALT
```

Za pomocą rozkazu

IN 03D

stan logiczny przełącznika P wprowadza się do akumulatora. Jeżeli przełącznik P znajduje się w położeniu T, wówczas najstarszy bit w akumulatorze po wykonaniu tego rozkazu ma wartość 1. Rozkazem

JM 4000H

realizuje się skok do początkowego adresu programu T. Jeżeli przełącznik P znajduje się w położeniu R, wówczas najstarszy bit w akumulatorze otrzymuje wartość 0 i nie dochodzi do skoku programowego, lecz do wykonania czterech kolejnych rozkazów. Są to rozkazy typu przygotowawczego, ze względu na to, że następuje program, za pomocą którego realizuje się odbiór. O rozkazach

LXI H,00EEH

MVI B,11000000B

będziemy mówili później. Za pomocą rozkazu

EI

mikroprocesor doprowadza się do stanu, w którym bezwarunkowo przyjmuje pierwsze żądanie przerwania, które nadejdzie, zaś za pomocą rozkazu

HALT

mikroprocesor zatrzymuje wykonywanie programu.

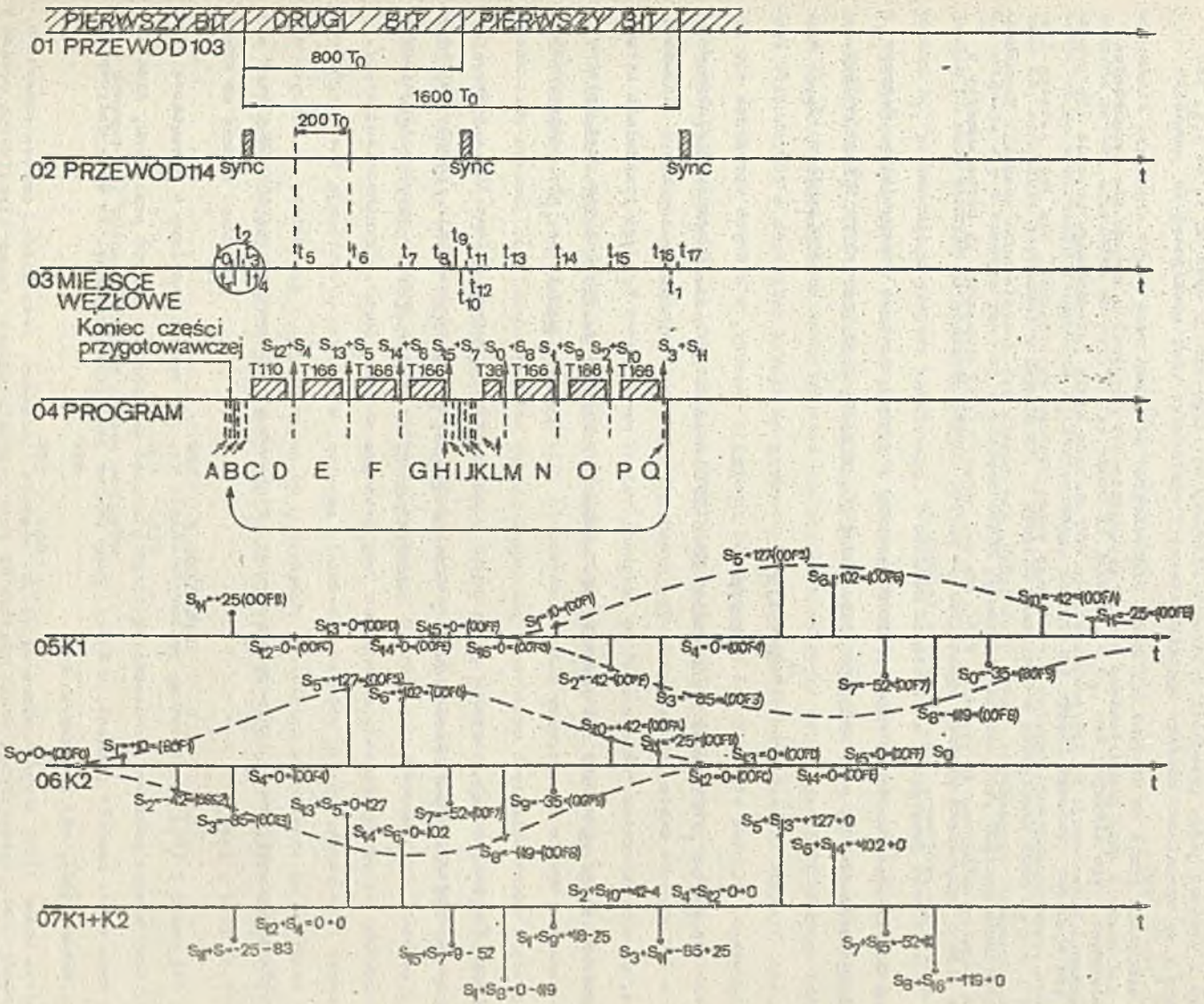
Nadawanie

Program, za pomocą którego realizuje się nadawanie:

| A | | PROGRAM | | B | T | V |
|---|---|---|---|---|---|---|
| A | | LXI | HL,00F3H | 3 | 10 | → $t_o$ |
| | | LXI | DE,00FBH | 3 | 10 | → $t_1$, $t_1^*$ |
| B | A0000: | IN | 05D | 2 | 10 | → $t_2$ |
| | | MOV | B,A | 1 | 5 | |
| C | | MVI | A,00000001B | 2 | 7 | |
| | | OUT | 06D | 2 | 10 | $t_3$ |
| | | CMA | | 1 | 4 | |
| | | OUT | 06D | 2 | 10 | → $t_4$ |
| D | | INX | HL | 1 | 5 | |
| | | INX | DE | 1 | 5 | |
| | | LDAX | DE | 1 | 7 | |
| | | ADD | M | 1 | 7 | |
| | | CALL | T110 | N1 | 110 | |
| | | OUT | 02D | 2 | 10 | → $t_5$ |
| E | | INX | HL | 1 | 5 | |
| | | INX | DE | 1 | 5 | |
| | | LDAX | DE | 1 | 7 | |
| | | ADD | M | 1 | 7 | |
| | | CALL | T166 | N2 | 166 | |
| | | OUT | 02D | 2 | 10 | → $t_6$ |
| F | | INX | HL | 1 | 5 | |
| | | INX | DE | 1 | 5 | |
| | | LDAX | DE | 1 | 7 | |
| | | ADD | M | 1 | 7 | |
| | | CALL | T166 | N2 | 166 | |
| | | OUT | 02D | 2 | 10 | → $t_7$ |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | | $t_7$ |
| G | | INX | HL | 1 | 5 | |
| | | INX | DE | 1 | 5 | |
| | | LDAX | DE | 1 | 7 | |
| | | ADD | M | 1 | 7 | |
| | | CALL | T166 | N2 | 166 | |
| | | OUT | 02D | 2 | 10 | $t_8$ |
| H | | JMP | A0001 | 3 | 10 | |
| I | A0001: | IN | 03D | 2 | 10 | $t_9$ |
| | | MOV | C,A | 1 | 5 | |
| J | | MVI | A,00000001B | 2 | 7 | |
| | | OUT | 06D | 2 | 10 | $t_{10}$ |
| | | CMA | | 1 | 4 | |
| | | OUT | 06D | 2 | 10 | $t_{11}$ |
| K | | MOV | D,H | 1 | 5 | |
| | | MOV | E,L | 1 | 5 | * |
| L | | MOV | A,C | 1 | 5 | * |
| | | XRA | B | 1 | 4 | * |
| | | JP | ADRXX | 3 | 10 | |
| | ADRXY: | MOV | A,B | 1 | 5 | |
| | | JP | ADR01 | 3 | 10 | |
| | ADR10: | LXI | BC,49144D | 3 | 10 | |
| | | DAD | BC | 1 | 10 | |
| | | JMP | A0002 | 3 | 10 | |
| | ADR01: | LXI | BC,16376D | 3 | 10 | |
| | | DAD | BC | 1 | 10 | |
| | | JMP | A0002 | 3 | 10 | |
| | ADRXX: | MOV | A,B | 1 | 5 | * |
| | | JP | ADR00 | 3 | 10 | * |
| | ADR11: | LXI | BC,32760D | 3 | 10 | |
| | | DAD | BC | 1 | 10 | |
| | | JMP | A0002 | 3 | 10 | |
| | ADR00: | LXI | BC,65528D | 3 | 10 | * |
| | | DAD | BC | 1 | 10 | * |
| | | JMP | A0002 | 3 | 10 | * |
| M | A0002: | INX | HL | 1 | 5 | |
| | | INX | DE | 1 | 5 | |
| | | LDAX | DE | 1 | 7 | |
| | | ADD | M | 1 | 7 | |
| | | CALL | T36 | N3 | 36 | |
| | | OUT | 02D | 2 | 10 | $t_{13}$ |
| N | | INX | HL | 1 | 5 | |
| | | INX | DE | 1 | 5 | |
| | | LDAX | DE | 1 | 7 | |
| | | ADD | M | 1 | 7 | |
| | | CALL | T166 | N2 | 166 | |
| | | OUT | 02D | 2 | 10 | $t_{14}$ |
| O | | INX | HL | 1 | 5 | |
| | | INX | DE | 1 | 5 | |
| | | LDAX | DE | 1 | 7 | |
| | | ADD | M | 1 | 7 | |
| | | CALL | T166 | N2 | 166 | |
| | | OUT | 02D | 2 | 10 | $t_{15}$ |
| P | | INX | HL | 1 | 5 | |
| | | INX | DE | 1 | 5 | |
| | | LDAX | DE | 1 | 7 | |
| | | ADD | M | 1 | 7 | |
| | | CALL | T166 | N2 | 166 | |
| | | OUT | 02D | 2 | 10 | $t_{16}$ |
| Q | | JMP | A0000 | 3 | 10 | $t_1$ |

Program jest dla lepszego zrozumienia, podzielony na odcinki oznaczone w kolumnie A dużymi literami od A do Q. Przy każdym rozkazie zaznaczono z ilu bajtów się składa /kolumna B/ i przez ile okresów taktu podstawowego trwa /kolumna T/. Ważne miejsca w programie oznaczono są w kolumnie V. Dotyczą dwóch sąsiednich rozkazów, rozdzielonych linią poziomą.




Rys.77. Zawiera 7 osi czasowych

Przebieg programu w czasie rzeczywistym przedstawiono na rys.77.Rys.77 zawiera 7 osi czasowych.

Na osi numer 1 (01) przedstawiono stan w przewodzie 103 /wejście do modemu od strony urządzenia końcowego/. Przedział czasowy między momentami $t_3$ i $t_{17}$ trwa przez 1600 okresów taktu podstawowego i stanowi jeden przedział dibitowy. Pierwszy bit tego dibitu trwa od $t_3$ do $t_{10}$, zaś drugi od $t_{10}$ do $t_{17}$.

Na osi numer 2 (02) przedstawiono stan w przewodzie 114 /wyjście z modemu dla synchronizacji urządzenia końcowego/. Na tym przewodzie co 800 okresów taktu podstawowego pojawia się przednie zbocze synchronizującego impulsu określonego współczynnika wypełnienia. Nie jest istotne ile wynosi współczynnik wypełnienia. Ważne jest tylko, aby impuls był dostatecznie szeroki. Zsynchronizowane impulsy oznaczono są symbolem SYNC. Na osi numer 3 (03) oznaczone są ważne momenty w trakcie przebiegu programu, tzn. momenty z kolumny V.

Na osi numer 4 (04) przedstawiono przebieg samego programu w czasie rzeczywistym. Pionowe strzałki nad osią czasową oznaczają momenty, w których na wejściu do konwertera D/A pojawiają się próbki sygnału liniowego. Przy strzałce znajduje się kolejny numer danego pomiaru: $S_i$ /i = = 0,...,15/. Pionowe strzałki nad osią czasową oddalone są od siebie dokładnie o 200 okresów taktu podstawowego. Pionowe strzałki pod osią czasową oznaczają momenty, w których wprowadza się do mikrokomputera stan z przewodu 103. Pionowe strzałki pod osią czasową są oddalone od siebie dokładnie o 800 okresów taktu podstawowego. Ten przedział odpowiada jednemu przedziałowi bitowemu. Przyjęto, że mikrokomputer "zwraca się" do przewodu 103 na samym końcu każdego przedziału bitowego /momenty $t_2$ i $t_9$/. Gdy bity informacyjne mają kształt prostokątny, wówczas nie jest istotne w jakim momencie będzie odczytywany stan w przewodzie 103. Istotne jest tylko, aby odczytywanie odbywało się w równo odległych momentach, odległość między którymi równa jest jednemu przedziałowi bitowemu. Nie ma żadnych przeszkód, aby moment odczytywania przesunąć w lewo lub w prawo. Miejsce momentu odczytywania zależy od odległości między momentami odczytywania /pionową strzałką pod osią czasową/ oraz momentów, w których występuje dodatnie zbocze impulsu synchronizującego /pod warunkiem, że synchronizacja dokonywana jest przednim zboczem impulsu synchronizującego/. Na osi numer 4(04) oznaczone są również przedziały czasowe, w których przebiegają poszczególne odcinki programu wg oznaczeń w kolumnie A dołączonej do programu jak i przedziały, w których mikroprocesor tylko "liczy" i "czeka" aż upłynie określony czas /części zakreskowane oznaczone symbolami T110, T166 i T36/.

Na osi numer 5 (05) oznaczone są pomiary w kanale K1, gdy symbol kanałowy z początkową fazą $P_0 = 0^0$. Przy każdym pomiarze oznaczone są: jego wielkość skwantowana na 256 poziomach, liczba kolejna oraz adres komórki pamięci, w której dany pomiar jest przechowywany. To samo dotyczy również osi numer 6(06), ale dla kanału K2.

Na osi numer 7(07) przedstawione są próbki sygnału liniowego. Przy każdej próbce oznaczono, w jaki sposób jest ona utworzona.

Obecnie przechodzimy do wyjaśnienia poszczególnych odcinków programu.

A

W rejestrze HL znajduje się adres komórki pamięci; w niej znajduje się próbka dla ostatniego

symbolu kanałowego, która po dodaniu do odpowiedniej próbki z przeciwnego kanału, jest wyprowadzana na linię. W rejestrze DE znajduje się adres komórki pamięci, w której przechowywana jest ta próbka poprzedniego symbolu kanałowego.

Załóżmy, że początkowe fazy ostatniego i przedostatniego symbolu kanałowego są w momencie włączenia zasilania równe zeru oraz, że bezpośrednio przed tym została na linię fikcyjnie wyprowadzona próbka oznaczona na rys.77 /oś numer 7/ jako $S_{11} + S_3$, zaś pojawiła się w momencie $t_0$. Rozpatrując w stosunku do momentu $t_0$ stwierdzamy, że ostatni symbol kanałowy znajduje się w kanale K2, natomiast poprzedni w kanale K1. W symbolu kanałowym z początkową fazą $P_0 = 0^\circ$, pomiar $S_3$ znajduje się w komórce pamięci z adresem 00F3H /obszar $A_1$ na rys.76/. Dlatego zawartość rejestru HL musi wynosić na początku 00F3H. Próbkę $S_3$ ostatniego symbolu kanałowego dodaje się do próbki $S_{11}$ poprzedniego symbolu kanałowego. W symbolu kanałowym z początkową fazą $P_0 = 0^\circ$ pomiar $S_{11}$ przechowywany jest w komórce pamięci z adresem 00FBH. Dlatego zawartość rejestru DE musi na początku wynosić 00FBH. Tworzenie początkowej zawartości rejestrów HL i DE dokonywane jest w ramach odcinka programu A:

```
LXI     HL,00F3H
LXI     DE,00FBH
```

Tworzenie początkowej zawartości rejestrów HL i DE kończy się w momencie $t_1$ na rys.77.

B

W ramach odcinka programu B wpierw wprowadza się pierwszy bit bieżącego dibitu do akumulatora w momencie $t_2$, za pomocą rozkazu

```
IN      05D
```

Ponieważ przewód 103 pośrednio połączony jest z linią D7 na szynie danych - po tym rozkazie omawiany bit znajduje się w najstarszej pozycji bitowej w akumulatorze. Następnie bit ten jest tymczasowo zapamiętywany w rejestrze B za pomocą rozkazu

```
MOV     B,A
```

C

W ramach odcinka programu C, w momencie $t_3$ tworzy się przednie zbocze impulsu synchronizującego za pomocą rozkazów:

```
MVI     A,00000001B
OUT     06D
```

zaś w momencie $t_4$ tylne zbocze za pomocą rozkazów

```
CMA
OUT     06D
```

W naszym przykładzie impuls synchronizujący trwa 14 okresów taktu podstawowego, tzn. około 7,3 µs ponieważ częstotliwość oscylatora kwarcowego wynosi 17,28 MHz.

D

Następna próbka sygnału liniowego tworzy się przez dodanie próbki $S_4$ ostatniego symbolu kanałowego do próbki $S_{12}$ przedostatniego symbolu kanałowego. Te próbki znajdują się w komórkach

pamięci z adresami odpowiednio 00F4H i 00FCH. Po pierwszych dwóch rozkazach z odcinka programu D

```
INX     HL
INX     DE
```

zawartość rejestru HL staje się równa 00F4H, zaś zawartość rejestru DE równa 00FCH. Następnie zawartość komórki pamięci, której adres znajduje się w rejestrze DE, tzn. próbkę $S_{12}$ z obszaru $A_T$ na rys.76 przenosi się do akumulatora za pomocą rozkazu

```
LDAX    DE
```

i dodaje do zawartości komórki pamięci, której adres jest równy zawartości rejestru HL, tzn. do próbki $S_4$ z obszaru $A_T$ za pomocą rozkazu

```
ADD     M
```

Tak została utworzona próbka, która powinna być "wyprowadzona" z mikrokomputera w momencie $t_5$. Jednak tworzenie tej próbki jest ukończone dużo wcześniej przed momentem $t_5$. Dlatego mikroprocesor wchodzi w skończoną pętlę, umownie oznaczoną jako

```
CALL    T110
```

i trwającą dokładnie 110 okresów taktu podstawowego. W rzeczywistości, w trakcie tej skończonej pętli mikroprocesor "liczy" i "czeka" aż minie przedział czasowy o długości T110 - zakreskowany na rys.77.

Taka skończona pętla jest realizowana za pomocą kilku rozkazów w programie głównym. Wówczas nie jest potrzebna pamięć RAM. Z drugiej strony rozkaz CALL ADRM zakłada istnienie podprogramu i obecność pamięci RAM. Aby jednak program był krótszy, zamiast całego odcinka programu, realizującego skończoną pętlę, podano tylko rozkaz CALL ADRM. Jednak nie ma się przy tym na myśli wywołania podprogramu, lecz realizację najzwyklejszej skończonej pętli w ramach programu głównego, co nie wymaga pamięci RAM. Dalej wyjaśnimy dlaczego skończona pętla T110 trwa właśnie przez 110 okresów taktu podstawowego.

Po tej skończonej pętli program przechodzi do rozkazu

```
OUT     02D
```

i próbka sygnału liniowego pojawia się na wejściu do konwertera D/A, dokładnie w momencie $t_5$.

E

Za pomocą czterech pierwszych rozkazów tego odcinka programu tworzy się następną próbkę sygnału liniowego, czyli odpowiednio dodaje zawartości komórek pamięci z adresami 00F5H i 00FDH. Znajdują się w nich próbki $S_5$ i $S_{13}$ z obszaru $A_T$. I tym razem utworzenie próbki jest zakończone znacznie wcześniej przed momentem $t_6$, w którym próbka powinna pojawić się na wejściu do konwertera D/A. Dlatego mikroprocesor ponownie wchodzi w skończoną pętlę, która tym razem umownie jest oznaczona jako

```
CALL    T166
```

Aby omawiana próbka pojawiła się dokładnie w momencie $t_6$ skończona pętla T166 musi trwać przez 166 okresów taktu podstawowego. Do tej wartości doszliśmy przez dodanie czasu trwania rozkazów, które wykonywane są między momentami $t_5$ i $t_6$. Suma powinna wynosić 200 okresów taktu podstawowego. Przypomnijmy, że czasy trwania rozkazów podane są w kolumnie T przy programie.

Po wykonaniu rozkazu

```
OUT     02D
```

również ta próbka pojawia się na wejściu do konwertera D/A.

F, G

Odcinków programu F i G dotyczy to samo,co i odcinka programu E. Nowe próbki sygnału liniowego pojawiają się w momentach $t_7$ i $t_8$.

H

Odcinek programu H ma jedno zadanie - wprowadzenie opóźnienia, wynoszącego 10 okresów taktu podstawowego. Potrzebne jest to dlatego, aby odległość między momentem $t_2$, w którym wzięto pierwszy bit z przewodu 103 a momentem $t_9$, w którym zostanie wzięty drugi bit z przewodu 103 wynosiła dokładnie 800 okresów taktu podstawowego.

I

W ramach tego odcinka programu wprowadza się drugi bit bieżącego dibitu do akumulatora i chwilowo zapamiętuje w rejestrze C. Tak jak i w odcinku programu B, stan w przewodzie 103 wprowadza się do akumulatora za pomocą rozkazu

IN 05D

którego wykonanie kończy się w momencie $t_9$. Z kolumny T można określić, że czas między momentami $t_2$ i $t_9$ wynosi dokładnie 800 okresów taktu podstawowego. Po wykonaniu rozkazu

MOV C,A

drugi bit bieżącego dibitu znajduje się w rejestrze C na najstarszej pozycji bitowej.

J

Odcinek programu J ma taką samą rolę jak odcinek programu C. Przednie zbocze impulsu synchronizującego pojawia się w momencie $t_{10}$, a tylne w momencie $t_{11}$.

K

Na rys.77 /oś numer 5/ widać, że ostatnia próbka symbolu kanałowego w kanale K1 jest wykorzystana w momencie $t_8$. Z drugiej strony, w momencie $t_9$ również drugi bit najnowszego dibitu jest wprowadzany do mikrokomputera. Załóżmy, że również ten dibit ma wartość 00. Według tab.13 dibit 00 powoduje w sygnale liniowym przesunięcie fazowe wynoszące $0^o$. Oznacza to, że nowy symbol w kanale K1, powinien mieć początkową fazę przesuniętą o $0^o$ w stosunku do początkowej fazy ostatniego symbolu kanałowego w kanale K2. Dlatego nowy symbol kanałowy w kanale K1, rozpoczynający się w momencie $t_{13}$ ma również fazę początkową $P_0 = 0^o$. W momencie $t_{13}$ pierwsza próbka $/S_o/$ najnowszego symbolu kanałowego w kanale K1 zostanie dodana do próbki $S_8$ symbolu kanałowego w kanale K2. Tak więc począwszy od momentu $t_{13}$ ostatni symbol dibitowy znajduje się w kanale K1, zaś przedostatni w kanale K2.

Jak już powiedzieliśmy, w rejestrze HL umieszczony jest adres komórki pamięci, w której znajduje się odpowiednia próbka ostatniego symbolu kanałowego, zaś w rejestrze DE adres komórki pamięci, w której znajduje się odpowiednia próbka przedostatniego symbolu kanałowego. Po momencie $t_8$, w którym "wyprowadzono" ostatnią próbkę sygnału liniowego zawartość rejestru HL wynosi 00F7H,

a w komórce pamięci z tym adresem znajduje się próbka odpowiadająca przedostatniemu, a nie ostatniemu symbolowi kanałowemu. Dlatego też zawartość rejestru HL należy przenieść do rejestru DE. W ten sposób rejestr HL pozostaje wolny, aby wprowadzić do niego adres komórki pamięci, w której znajduje się pierwsza próbka najnowszego symbolu dibitowego, zaś w rejestrze DE tworzy się nowy stan, tzn. adres odpowiedniej próbki symbolu kanałowego, który bezpośrednio przed tym był ostatni, a teraz jest przedostatni.

W naszym przykładzie najnowszy symbol kanałowy ma również początkową fazę $P_0 = 0^o$, adresem komórki pamięci, w której znajduje się jego pierwsza próbka jest 00F0H. Adres ten zostanie umieszczony w rejestrze HL, po czym dokona się przeniesienia zawartości rejestru HL do rejestru DE.

Przenoszenie zawartości rejestru HL do rejestru DE dokonuje się za pomocą następujących dwóch rozkazów:

```
MOV     D,H
MOV     E,L
```

L

W ramach odcinka programu L tworzy się nową zawartość rejestru HL w zależności od wartości najnowszego dibitu.

Po wykonaniu rozkazów

```
MOV     A,C
XRA     B
```

w akumulatorze znajduje się wynik operacji logicznej różnicy symetrycznej między zawartościami rejestrów B i C. Przypomnijmy, że pierwszy bit najnowszego dibitu znajduje się w rejestrze B, a drugi w rejestrze C. Jeżeli najnowszy dibit ma wartość 00 lub 11, wówczas po tej operacji wskaźnik S będzie miał wartość 0/S=0/. Jeżeli najnowszy dibit ma wartość 01 lub 10, wówczas po tej operacji wskaźnik S będzie miał wartość 1 /S=1/.

Następnym rozkazem jest

```
JP      ADRXX
```

Jeżeli S=0, wówczas po tym rozkazie program przechodzi do wykonania rozkazu, którego etykieta brzmi ADRXX. Jeżeli dojdzie do skoku programu pod adres ADRXX, wówczas oznaczać to będzie, że bity w dibicie są takie same. Jest to podkreślone przez oznaczenie XX. Jeżeli S = 1, wówczas po rozkazie JP ADRXX program przechodzi do wykonania następnego kolejnego rozkazu. Jego etykietą jest ADRXY. Jeżeli dojdzie do wykonania rozkazu z etykietą ADRXY, wówczas oznacza to, że bity w dibicie są różne. Podkreślone jest to przez oznaczenie XY.

Gdy bity w dibicie są takie same, następnymi wykonywanymi dwoma rozkazami są:

```
ADRXX:  MOV   A,B
        JP    ADR00
----------------------
ADR11:
```

Obecnie testuje się pierwszy bit w dibicie. Jeżeli jest on równy zeru, wówczas oznacza to, że mamy do czynienia z dibitem 00 i dokonuje się skoku w programie do rozkazu z etykietą ADR00. Jeżeli jest on równy jedności, wówczas oznacza to, że mamy do czynienia z dibitem 11 i program prze-

chodzi do następnego kolejnego rozkazu. Jego etykietą jest ADR11. Gdy bity w dibicie są różne, wówczas dwa następujące wykonywane rozkazy brzmią:

```
ADRXY:    MOV    A,B
          JP     ADR01
-------------------------
ADR10
```

Jeżeli pierwszy bit w dibicie równy jest zeru, oznacza to, że mamy do czynienia z dibitem 01 i wówczas dokonywany jest skok programu do rozkazu z etykietą ADR01. Jeżeli pierwszy bit w dibicie równy jest jedności, wówczas mamy do czynienia z dibitem 10, a program przechodzi do następnego kolejnego rozkazu. Jego etykieta brzmi: ADR10. Dopiero teraz tworzy się nową zawartość rejestru HL. Dokonuje się tego za pomocą rozkazów:

```
ETYKIETA:  LXI    BC,P16
           DAD    BC
```

Za pomocą rozkazu LXI BC,P16 do rejestru BC wprowadza się pewną liczbę 16-bitową. Jaka będzie ta liczba zależy od wartości najnowszego dibitu. Jeżeli wartość najnowszego dibitu wynosi 00, wówczas według tab.13 do sygnału liniowego należy wprowadzić przesunięcia o $0^{\circ}$. Dlatego zawartość rejestru HL należy zmniejszyć o 8D, aby utworzyła się w nim zawartość równa adresowi pierwszej próbki obszaru, w którym rejestr HL i przed tym się "poruszał". Zmniejszenie o 8D jest równoznaczne z powiększeniem o 65528D, ponieważ w naszej pamięci ROM obowiązuje dodawanie według modułu 65536D.

Tak więc w wypadku dibitu 00 wykonywane są dwa następujące rozkazy:

```
ADR00:    LXI    BC,65528D
          DAD    BC
```

Jeżeli wartość najnowszego dibitu wynosi 01, wówczas według tab.13 do sygnału liniowego należy wprowadzić przesunięcie fazowe wynoszące $+90^{\circ}$. Dlatego zawartość rejestru HL należy powiększyć o

$$16376 = 65536 / 4 - 8$$

aby zawartość rejestru HL stała się równa adresowi komórki pamięci, w której znajduje się pierwsza próbka z obszaru odpowiadającego symbolowi kanałowemu z początkową fazą większą o $+90^{\circ}$ od początkowej fazy symbolu kanałowego w obszarze, po którym rejestr HL przed tym się "poruszał". Tak więc w wypadku dibitu 01 wykonywane są następujące dwa rozkazy:

```
ADR01:    LXI    BC,16376D
          DAD    BC
```

Np., za pomocą tego rozkazu od próbki $S_8$ w obszarze $A_T$ przechodzi się do próbki $S_0$ w obszarze $D_T$.

Podobnie w wypadku dibitu 11 wykonywane są następujące dwa rozkazy:

```
ADR11     LXI    BC,32760D
          DAD    BC
```

W ten sposób np. od próbki $S_8$ w obszarze $A_T$, przechodzi się do próbki $S_0$ w obszarze $C_T$, ponieważ obowiązuje

$$32760 = 65536 /2 - 8$$

W końcu, w wypadku dibitu 10 wykonywane są następujące dwa rozkazy:

```
ADR10:    LXI    BC,49144D
          DAD    BC
```

W ten sposób np. od próbki $S_8$ w obszarze $A_T$ przechodzi się do próbki $S_o$ w obszarze $D_T$, ponieważ obowiązuje:

$$49144 = 3 \cdot 65536 / 4 - 8$$

W odcinku programu L pojawia się jeden nowy rozkaz. O rozkazie

DAD RP

dotychczas nie mówiliśmy. Za jego pomocą zawartość rejestru HL dodaje się do zawartości jednego z czterech rejestrów 16-bitowych /BC, DE, HL, SP/. Wynik pozostaje w rejestrze HL. Kodem tego rozkazu jest

00RR1001

W praktyce jest to rozkaz umożliwiający kodowanie różnicowe.

Wyjście z odcinka programu L realizuje się za pomocą rozkazu

JMP A0002

Bez względu na to o jaki dibit chodzi - odcinek programu L trwa zawsze przez taki sam czas, tzn. przez 64 okresy taktu podstawowego. W kolumnie V gwiazdkami oznaczono są te rozkazy z odcinka programu 1, które wykonywane są wtedy, gdy nadejdzie dibit 00.

M

W ramach odcinka programu M tworzy się nową próbkę sygnału liniowego w taki sam sposób, jak i poprzednio, przy odcinkach programów D,E,F i G.

Po wykonaniu pierwszego rozkazu tego odcinka programu zawartość rejestrów HL i DE wynosi odpowiednio 00F0H i 00F8H. Tak więc rejestr HL obecnie "wskazuje" pierwszą próbkę /$S_o$/ ostatniego symbolu dibitowego, natomiast rejestr DE obecnie "wskazuje" dziewiątą próbkę /$S_8$/ przedostatniego symbolu dibitowego.
Omawiany odcinek programu powinien zadziałać na wyjście mikrokomputera w momencie $t_{13}$. Aby odległość między momentami $t_8$ i $t_{13}$ wynosiła dokładnie 200 okresów taktu podstawowego, skończona pętla T36 musi trwać dokładnie przez 36 okresy taktu podstawowego.

N O P

Odcinki programu N,O i P są identyczne jak odcinki programu E,F i G. Nowe próbki sygnału liniowego pojawiają się w momentach $t_{14}$, $t_{15}$ i $t_{16}$.

Odcinek programu Q realizuje bezwarunkowy skok do początku programu. Przedział czasowy między momentami $t_{16}$ i $t_5$ musi również trwać przez 200 okresów taktu podstawowego. Należy przy tym mieć na uwadze, że sam bezwarunkowy skok, tzn. rozkaz

JMP A0000

trwa przez 10 okresów taktu podstawowego. Skok w programie kończy się w momencie $t_1^*$ na rys.77. Po dodaniu czasu trwania wszystkich rozkazów, wykonywanych między momentami $t_{16}$ i $t_5$ dochodzimy do wniosku, że skończona pętla T110, o której wcześniej mówiliśmy, musi trwać przez dokładnie 110 okresów taktu podstawowego.

Uwaga końcowa

Przedstawiony program jest napisany zbyt rozwlekle. Przy nieco większym doświadczeniu w tworzeniu programów, mógłby być on napisany tak, aby zajmował znacznie mniej przestrzeni w pamięci ROM. Jednak i w takiej formie może być on bez kłopotów umieszczony w obszarze T na rys.76, który jest dla niego przewidziany. Można to sprawdzić przez dodanie wszystkich elementów z kolumny B znajdującej się obok programu.

Odbiór

Program realizujący odbiór:

| A | PROGRAM | | | B | T | V | |
|---|---|---|---|---|---|---|---|
| | | | | | | | $\rightarrow T_{IR}$ |
| Sprzętowe wytwarzanie i wykonywanie rozkazu RST 7 | | | | | | | |
| A | | CALL | T29 | N4 | 29 | | |
| | | IN | 01D | 2 | 10 | | $\rightarrow t_3^R$ |
| B | | ORA | B | 1 | 5 | | |
| | | RRC | | 1 | 4 | | |
| | | MOV | B,A | 1 | 5 | | |
| C | | CALL | T76 | N5 | 76 | | |
| | | IN | 01D | 2 | 10 | | $\rightarrow t_4^R$ |
| D | | ORA | B | 1 | 5 | | |
| | | RRC | | 1 | 4 | | |
| | | MOV | B,A | 1 | 5 | | |
| E | | LDA | 00EAH | 3 | 13 | * + - ● | |
| | | CMP | B | 1 | 4 | * + - ● | |
| | | JZ | A1 | 3 | 10 | * + - ● | |
| | | LDA | 00EBH | 3 | 13 | + - ● | |
| | | CMP | B | 1 | 4 | + - ● | |
| | | JZ | A2 | 3 | 10 | + - ● | |
| | | LDA | 00ECH | 3 | 13 | - ● | |
| | | CMP | B | 1 | 4 | - ● | |
| | | JZ | A3 | 3 | 10 | - ● | |
| | | LDA | 00EDH | 3 | 13 | ● | |
| | | CMP | B | 1 | 4 | ● | |
| | | JZ | A4 | 3 | 10 | ● | |
| | A1: | CALL | T162 | N6 | 162 | * | |
| | | LXI | DE,0000H | 3 | 10 | * | |
| | | JMP | A5 | 3 | 10 | * | |
| | A2: | CALL | T135 | N7 | 135 | + | |
| | | LXI | DE,4000H | 3 | 10 | + | |
| | | JMP | A5 | 3 | 10 | + | |
| | A3: | CALL | T108 | N8 | 108 | - | |
| | | LXI | DE,8000H | 3 | 10 | - | |
| | | JMP | A5 | 3 | 10 | - | |
| | A4: | CALL | T91 | N9 | 81 | ● | |
| | | LXI | DE,C000H | 3 | 10 | ● | |
| | | JMP | A5 | 3 | 10 | ● | |
| F | A5: | DAD | DE | 1 | 10 | | |
| | | MOV | A,M | 1 | 7 | | |
| | | OUT | 04D | 2 | 10 | | $\rightarrow t_A^R$ |
| G | | CALL | T773 | N10 | 773 | | |
| | | INX | HL | 1 | 5 | | |
| | | MOV | A,M | 1 | 7 | | |
| | | DCX | HL | 1 | 5 | | |
| | | OUT | 04D | 2 | 10 | | $\rightarrow t_B^R$ |
| H | | CALL | T240 | N11 | 240 | | |
| | | IN | 01D | 2 | 10 | | $\rightarrow t_1^R$ |
| I | | RRC | | 1 | 4 | | |
| | | MOV | B,A | 1 | 5 | | |

| | | | | | |
|---|---|---|---|---|---|
| J | CALL | T81 | N12 | 81 | |
| | IN | 01D | 2 | 10 | $\rightarrow t_2^R$ |
| K | ORA | B | 1 | 5 | |
| | RRC | | 1 | 4 | |
| | MOV | B,A | 1 | 5 | |
| L | JMP | A0 | 3 | 10 | |

Również ten program, dla łatwiejszego objaśnienia, podzielony jest na odcinki oznaczone dużymi literami od A do L w kolumnie A. Kolumn B, T i V dotyczy to samo, co i poprzednio. Przebieg tego programu w czasie rzeczywistym jest przedstawiony na rys.74.

Przyjęto, że znaki próbki bieżącego symbolu dibitowego przechowywane są w rejestrze B, zaś adres bitu przekazywanego przewodem 104 do urządzenia końcowego - w rejestrze HL. Przypomnijmy, że w obszarach $A_R$, $B_R$, $C_R$ i $D_R$ na rys.76 znajdują się po dwa bajty, których najmłodsze bity przedstawiają wszystkie cztery możliwe kombinacje dibitowe. Po detekcji różnicowej bity te przekazywane są do urządzenia końcowego przewodem 104.

Przy czterowstęgowej różnicowej modulacji fazy pierwszy dibit nie ma charakteru informacyjnego lecz referencyjny. Zakładamy, że jego wartości wynoszą 00. Dlatego początkowa zawartość rejestru HL musi wynosić 00EEH. 00EEH jest adresem pierwszego bajtu w obszarze $A_R$ z rys.76. Tworzenie początkowej zawartości rejestru HL dokonywane jest w ramach programu testującego stan przełącznika P za pomocą rozkazu

LXI HL,00EEH

Tworzenie początkowej zawartości rejestru B również dokonywane jest w ramach programu testującego stan przełącznika P. Ponieważ zakłada się, że dibit referencyjny ma wartość 00, to rozkaz, za pomocą którego tworzy się początkową zawartość rejestru B jest:

MVI B,11000000B

Do rejestru B wprowadza się bajt

11000000

ponieważ znaki pierwszych dwóch próbek symbolu dibitowego z początkową fazą $P_0 = 0^o$ są dodatnie /$S_2$ i $S_1$ z rys.70/.

Załóżmy, że znajdujemy się w pewnym miejscu w trakcie odbioru bloku danych informacyjnych oraz że objaśnianie programu, za pomocą którego realizuje się odbiór, rozpoczyna się w momencie, gdy mikroprocesor znajduje się w stanie zatrzymania, bezpośrednio przed nadejściem żądania obsługi przerwania. Przez to żądanie obsługi przerwania - przypomnijmy - realizowana jest synchronizacja dibitowa. Niech w tym momencie zawartość rejestru HL jest równa 00EEH, a zawartość rejestru B jest taka, jak przedstawiono na rys.78.

| $S_2$ | $S_1$ | 0 | 0 | 0 | 0 | 0 | 0 | B |
|---|---|---|---|---|---|---|---|---|

Rys.78. Zawartość rejestru B w trakcie przedziału czasowego między momentami $t_2^R$ i $T_{IR}$ z rys.74

Tak więc bezpośrednio przed momentem, w której rozpoczyna się objaśnianie programu zdetektowany został dibit 00, zaś znaki dwóch pierwszych próbek bieżącego symbolu dibitowego są już wprowadzone do mikrokomputera i znajdują się w rejestrze B.

W momencie $T'_{IR}$ na rys.74 z układu synchronizującego nadchodzi żądanie obsługi przerwania. Żądanie to będzie bezwarunkowo przyjęte przez CPU, ponieważ bezpośrednio przed rozkazem HALT w programie testującym stan przełącznika P został wykonany rozkaz EI. Ponieważ końcówki INT w układzie 8080 oraz $\overline{INTA}$ w układzie 8228 są połączone jak na rys.75, to po przyjęciu żądania przerwania dokonywane jest wytwarzanie sprzętowe i wykonywanie rozkazu

RST 7

Wykonywanie tego rozkazu trwa przez 11 okresów taktu podstawowego. Dopiero po tym rozpoczyna się wykonywanie programu, za pomocą którego realizowany jest odbiór. Oczywiście, tak się dzieje pod warunkiem, że pierwszy bajt programu jest umieszczony w komórce pamięci z adresem 0038H /rys.76/.

Obecnie przechodzimy do objaśniania poszczególnych odcinków programu.

A

W momencie $t_3^R$ na rys.74 z linii pobierany jest znak trzeciej próbki w bieżącym przedziale dibitowym. Dokonuje się tego w ramach odcinka programu A:

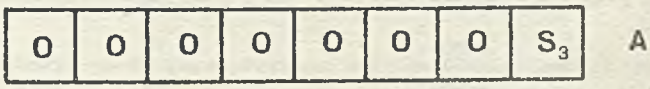
```
CALL    T29
IN      01D
```

Skończona pętla T29 realizuje opóźnienie o 29 okresów taktu podstawowego, ponieważ między momentem $T'_{IR}$, w którym nadchodzi żądanie obsługi przerwania i momentem $t_3^R$ musi upłynąć dokładnie 50 okresów taktu podstawowego.

Po wykonaniu rozkazu

```
IN      01D
```

zawartość akumulatora jest taka jak na rys.79.




Rys.79. Zawartość akumulatora po wykonaniu rozkazu IN 01D

Siedem najstarszych bitów w akumulatorze jest teraz równych zeru. Powodem tego jest fakt, że końcówki wejściowe w układzie 8212 z rys.75, odpowiadające tym bitom są połączone z masą. Nie byłoby celowe stosowanie w tym miejscu układu 8212, ale jego obecność znacznie ułatwia napisanie programu.

B

W ramach odcinka programu B, znak trzeciej próbki sygnału liniowego w bieżącym przedziale dibitowym umieszcza się na odpowiednim miejscu w rejestrze B.

Zawartość akumulatora po wykonaniu rozkazu

```
ORA     B
```

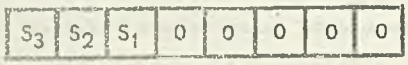
przedstawiona jest na rys.80.

Zawartość akumulatora po wykonaniu rozkazu

```
RRC
```

przedstawiona jest na rys.81.


Rys.80. Zawartość akumulatora po wykonaniu rozkazu ORA B



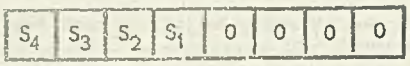

Rys.81. Zawartość akumulatora po wykonaniu rozkazu RRC

Zawartość rejestru B po wykonaniu rozkazu

```
MOV     B,A
```

przedstawiona jest na rys.82.


Rys.82. Zawartość rejestru B po wykonaniu rozkazu MOV B,A




Rys.83. Zawartość rejestru B po wykonaniu odcinka programu D

C

W ramach odcinka programu C

```
CALL    T 76
IN      01D
```

do akumulatora wprowadza się znak czwartej próbki sygnału liniowego w bieżącym przedziale dibitowym. Dokonuje się tego w momencie $t_4^R$ z rys.74. Aby odległość między momentami $t_3^R$ i $t_4^R$ wynosiła dokładnie 100 okresów taktu podstawowego, skończona pętla T76 musi trwać przez 76 okresów taktu podstawowego. Można to sprawdzić w kolumnie T znajdującej się przy programie.

D

Ten odcinek programu ma taką samą postać jak odcinek programu B. W jego ramach znak czwartej próbki sygnału liniowego w bieżącym przedziale dibitowym umieszcza się na odpowiednim miejscu w rejestrze B. Zawartość rejestru B po wykonaniu odcinka programu D jest przedstawiona na rys.83.

E

W ramach odcinka programu E zawartość rejestru B porównuje się z zawartością wszystkich czterech bajtów z obszaru U na rys.73. Przypomnijmy, że w obszarze U znajdują się znaki próbek symboli dibitowych z rys.70.

Gdy działanie szumu i innych zakłóceń jest niewystarczające do wywołania błędu przy detekcji znaku pomiaru, w trakcie omawianego porównania powinno dojść do zgodności zawartości rejestru B z zawartością jednego z czterech bajtów z obszaru U. Za pomocą trzech rozkazów

```
LDA     00EAH
CMP     B
JZ      A1
```

dokonuje się porównania zawartości rejestru B z zawartością komórki pamięci o okresie 00EAH. Jeżeli dojdzie do koincydencji zawartości, wówczas dokonywany jest skok do rozkazu z etykietą A1. W przeciwnym razie przechodzi się do następnych trzech kolejnych rozkazów.

Rozkaz CMP B nie występował do tej pory. Jego ogólna postać jest następująca:

```
CMP     R
```

Podczas wykonywania tego rozkazu zawartość jednego z siedmiu rejestrów odejmowana jest od zawartości akumulatora. Zawartość akumulatora przy tym się nie zmienia, ale w rejestrze stanu ustala się nowy stan. Kodem tego rozkazu jest

```
10111RRR
```

Do tego rozkazu podobny jest rozkaz

```
CMP     M
```

Teraz zawartość komórki pamięci, której adres jest równy zawartości rejestru HL odejmowana jest od zawartości akumulatora. Zawartość akumulatora również nie zmienia się, a nowo utworzony stan

w rejestrze stanu stanowi podstawę do realizacji warunkowego skoku programowego.

Następnymi trzema kolejnymi rozkazami są

```
LDA     00EBH
CMP     B
JZ      A2
```

Za ich pomocą dokonuje się porównania zawartości rejestru B z zawartością komórki pamięci, której adresem jest 00EBH. Jeżeli dojdzie do koincydencji zawartości, wówczas dokonuje się skoku programowego do rozkazu z etykietą A2. W przeciwnym razie przechodzi się do nowych, kolejnych trzech rozkazów, itd. W każdym wypadku program musi dojść do jednego z czterech rozkazów z etykietami A1, A2, A3 lub A4.

Załóżmy, że detekcji podlega symbol dibitowy z początkową fazą $P_0 = 0^o$. W tej sytuacji program dochodzi do rozkazu z etykietą A1 i wykonywane są następująco trzy rozkazy:

```
A1:   CALL    T162
      LXI     DE, 0000H
      JMP     A5
```

Tak więc najpierw realizowana jest skończona pętla T162, trwająca przez 162 okresy taktu podstawowego. Poniżej wyjaśnimy dlaczego ta skończona pętla musi trwać właśnie przez 162 okresy taktu podstawowego. Następnie tworzy się nowa zawartość rejestru DE. Nowa zawartość rejestru DE zależy od tego, jaki symbol dibitowy został zdetektowany. Ponieważ założyliśmy, że zdetektowany został symbol dibitowy z początkową fazą $P_0 = 0^o$ - w rejestrze umieszcza się liczbę 0000H. Zależności między początkową fazą detektowanego symbolu dibitowego a nową zawartością rejestru DE przedstawione są w tab.17.

| Dibit | Nowa zawartość rejestru DE |
|---|---|
| 00 | 0000H |
| 01 | 4000H |
| 11 | 8000H |
| 10 | C000H |

Tab.17

Później, w ramach odcinka programu F zawartość rejestru DE będzie dodana do zawartości rejestru HL. W ten sposób dokonuje się dekodowania różnicującego. Przypomnijmy, że w rejestrze HL umieszczony jest adres bajtu, w którym znajduje się bit przekazywany do urządzenia końcowego.

Wyjście z odcinka programu E realizuje się za pomocą rozkazu

```
JMP     A5
```

Przy detektowaniu symbolu dibitowego z początkową fazą $P_0 = +90^o$ po pierwszych sześciu rozkazach w odcinku programu E zostałyby wykonane następujące 3 rozkazy:

```
A2:   CALL    T135
      LXI     DE,4000H
      JMP     A5
```

Jeżeli symbol dibitowy z początkową fazą $P_0 = +180^o$, to po pierwszych dziewięciu rozkazach w odcinku programu E zostałyby wykonane następujące trzy rozkazy:

```
A3:     CALL    T108
        LXI     DE, 8000H
        JMP     A5
```

Jeżeli symbol z początkową fazą $P_0 = +270^0$, to po pierwszych 12 rozkazach w odcinku programu E zostałyby wykonane następujące trzy rozkazy:

```
A4:     CALL    T81
        LXI     DE, 0000H
        JMP     A5
```

Poniżej wyjaśnimy jak określono czas trwania skończonych pętli T135, T108 i T81.

Dotychczas zakładano, że nigdy nie dochodzi do błędu przy określaniu znaku próbki sygnału liniowego. Jednak w rzeczywistym kanale zawsze występują szumy i inne zakłócenia, więc takie błędy są nie do uniknięcia.

Gdy doszłoby do błędu - przebieg programu byłby zakłócony. Po wykonaniu wszystkich rozkazów oznaczonych kropkami nie doszłoby do koincydencji zawartości rejestru B i bajtów z obszaru U. Wówczas program przeszedłby do wykonania następnego kolejnego rozkazu. Jest to rozkaz z etykietą A1 i na wejściu do urządzenia końcowego pojawiłby się dibit 00, którego przednie zbocze opóźnione jest o 31 okresów taktu podstawowego. Aby temu zapobiec, przed rozkazem A1 wstawia się odcinek programu, który w wypadku zaistnienia błędu wyprowadza w momentach $t_A^R$ i $t_B^R$ na przewód 104 dowolny, ale zawsze ten sam dibit. Następnie przechodzi się do detekcji nowego dibitu tak, jakby nic się nie stało. Nie nastąpi propagacja błędu, ponieważ dekodowanie różnicowe ma taką właściwość, że przy błędnej detekcji jednego symbolu dibitowego następuje błędna interpretacja tylko dwóch dibitów. W modemach czasem istnieje świetlne wskazanie detekcji błędu. W tym celu w ramach wstawionego odcinka programu można spowodować, że jedna dioda świetlna LED zaświeci się, gdy tylko dojdzie do detekcji błędu.

Sprzężenie diody LED i mikroprocesora może być wykonane w sposób jak na rys.84.

Dla programisty dioda LED przedstawia obecnie tylko jedną wyjściową komórkę peryferyjną z określonym adresem. Dioda LED zapali się, jeżeli do tej komórki peryferyjnej wpisze się określoną zawartość, np.

```
11111111
```

Suma oporności diody LED i wstawionego opornika R musi odpowiadać wartości wyspecyfikowanej przez producenta bufora z rys.84.

F

Dekodowanie różnicowe w rzeczywistości dokonywane jest w odcinku programu F.

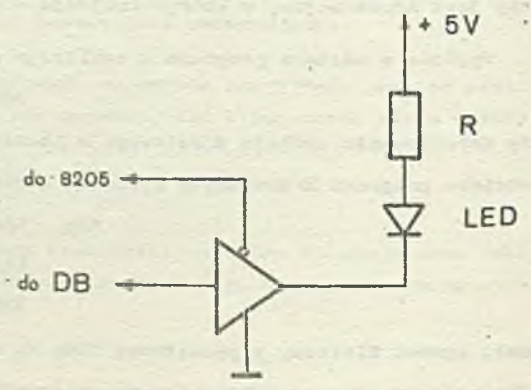
Za pomocą rozkazu

```
A5:     DAD     DE
```

tworzy się nową zawartość rejestru HL.

Za pomocą rozkazu

```
        MOV     A,M
```




Rys.84. Połączenie mikroprocesora i diody LED

przenosi się do akumulatora zawartość komórki pamięci, której adres jest równy zawartości rejestru HL. W rzeczywistości, w ten sposób pierwszy bajt z jednego z obszarów $A_R$, $B_R$, $C_R$ lub $D_R$ przenosi się do akumulatora.

Za pomocą rozkazu

OUT 04D

zawartość najmłodszej pozycji bitowej w akumulatorze przekazuje się do urządzenia końcowego po przewodzie 104.

Załóżmy, że bezpośrednio przed momentem, od którego rozpoczęto objaśnienie programu został zdetektowany dibit 00. Dlatego zastana zawartość rejestru HL jest równa 00EEH. Poza tym załóżmy, że pierwszy detektowany dibit przy rozpoczęciu objaśnienia programu ma również wartość 00. Dlatego po wykonaniu rozkazu

A5: DAD DE

zawartość rejestru HL pozostaje równa 00EEH, zaś po wykonaniu rozkazu

MOV A,M

zawartość akumulatora ma postać jak na rys.85.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | A |

Rys.85. Zawartość akumulatora po wykonaniu rozkazu MOV, A,M

Za pomocą rozkazu OUT 04D

zawartość najmłodszej pozycji bitowej w akumulatorze przekazywana jest do urządzenia końcowego przewodem 104. Przewód 104 za pośrednictwem przerzutnika L z adresem 04D jest połączony z linią D0 na szynie danych. Widać to na rys.75. Pierwszy bit nowego dibitu musi pojawić się w przewodzie 104 dokładnie w momencie $t_A^R$. Z rys.74 widać, że przedział czasowy między momentami $t_4^R$ i $t_A^R$ trwa przez dokładnie 250 okresów taktu podstawowego. Dlatego odcinki programu D,E i F muszą razem trwać przez 250 okresów taktu podstawowego.

Bez względu na to, który symbol dibitowy jest detektowany - wszystkie rozkazy odcinków programu D i F bezwzględnie są wykonywane. Jednak, które rozkazy z odcinka programu E będą wykonane, zależy od konkretnej wartości detektowanego symbolu dibitowego. W kolumnie V gwiazdkami oznaczono rozkazy wykonywane przy detekcji symbolu dibitowego z początkową fazą $P_0 = 0^o$. Plusami oznaczono rozkazy wykonywane przy detekcji symbolu dibitowego z początkową fazą $P_0 = +90^o$. Minusami oznaczono rozkazy wykonywane przy detektowaniu symbolu dibitowego z początkową fazą $P_0 = +180^o$. Kropkami oznaczono rozkazy wykonywane przy detektowaniu symbolu dibitowego z początkową fazą $P_0 = +270^o$.

Bez względu na to, który symbol dibitowy jest detektowany - odcinek programu E musi zawsze trwać tak samo długo, tzn. dokładnie 209 okresów taktu podstawowego. Aby zapewnić trwanie odcinka programu E dokładnie przez 209 okresów taktu podstawowego, skończone pętle T162, T135, T108 i T91 muszą trwać odpowiednio 162, 135, 108 i 81 okresów taktu podstawowego.

G

W ramach tego odcinka programu drugi bajt ostatniego detektowanego dibitu jest przygotowany i "wyprowadzany" na przewód 104 w momencie $t_B^R$, oddalonej od momentu $t_A^R$ o 800 okresów taktu podstawowego, tzn. dokładnie o jeden przedział bitowy. Aby ten bit pojawił się w przewodzie 104 dokładnie w momencie $t_B^R$ najpierw realizuje się skończoną pętlę T773 trwającą przez 733 okresy taktu podstawowego. Oznaczona jest ona rozkazem

CALL T773

Następnie zawartość rejestru HL zwiększa się o 1 za pomocą rozkazu

INX HL

W ten sposób program "skacze" do drugiego bajtu z obszaru /$A_R$, $B_R$, $C_R$ lub $D_R$/, w którym już się znajdował. Przypomnijmy, że najmłodszy bit tego bajtu odpowiada drugiemu bitowi ostatniego detektowanego dibitu.

Za pomocą rozkazu

MOV A,M

omawiany bajt przenoszony jest do akumulatora.

Za pomocą rozkazu

DCX HL

program wraca do pierwszego bajtu z obszaru, w którym się znajduje.

Po wykonaniu rozkazu

OUT 04D

również drugi bit ostatniego detektowanego dibitu pojawia się w przewodzie 104 i to dokładnie w momencie $t_B^R$.

H

W ramach tego odcinka programu mikroprocesor "czeka" aż upłynie 240 okresów taktu podstawowego, zaś w momencie $t_1^R$ testuje znak sygnału liniowego. Jest to znak pierwszej próbki następnego symbolu dibitowego.

I

W ramach tego odcinka programu znak pierwszej próbki następnego symbolu dibitowego umieszcza się w odpowiednim miejscu rejestru B.

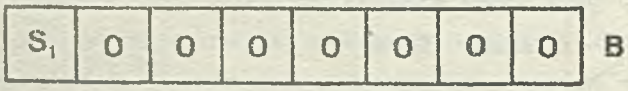
Po wykonaniu rozkazów

RRC

J

MOV B,A

zawartość rejestru B jest taka jak na rys.86.




Rys.86. Zawartość rejestru B po wykonaniu odcinka programu I

J

W ramach tego odcinka programu, podobnie jak odcinka programu H, mikroprocesor "czeka" aż upłynie 81 okresów taktu podstawowego, zaś w momencie $t_2^R$ ponownie testuje znak sygnału liniowego. Jest to znak drugiej próbki jednego z symboli dibitowych z rys.80

K

Ten odcinek programu ma taką samą postać jak odcinki programu B i D. Po jego wykonaniu zawartość rejestru B jest taka sama jak na rys.78.

L

W ramach tego odcinka programu dokonuje się bezwarunkowego skoku do rozkazu z etykietą A0 w programie testującym stan przełącznika P. Bezpośrednio po tym mikroprocesor zatrzymuje się i "rusza" dopiero po nadejściu z układu synchronizującego nowego żądania obsługi przerwania. Następnie wszystko się powtarza.

Uwaga końcowa

Również i ten program został napisany w sposób bardzo rozwlekły. Jednak może on być bez kłopotów umieszczony w obszarze R z rys.76, który jest dla niego przewidziany.

W modemie nie zrealizowano przewodu 113 według CCITT(synchronizacja nadawania), służącego do zewnętrznej synchronizacji nadajnika modemowego z nadajnikiem urządzenia końcowego. Aby to zrealizować należy wprowadzić jeszcze jeden poziom przerwania. Po utworzeniu czterech próbek sygnału liniowego mikroprocesor zatrzymałby się, zaś takt nadawczy co 416 μs zgłaszałby nowe żądanie przerwania. Jest to więc podobne do synchronizacji przy odbiorze. Tę problematykę szerzej omówiono w artykule: R.W.Stroh:An Experimental Mikroprocessor-Implemented 4800 bs PSK MODEM, IEEE Transaction on Communication 1978 t.26 nr 5.

WNIOSEK

Przedstawiony modem jest względnie czuły na szum i inne zakłócenia w kanale. Powodem tego jest stosunkowo uproszczona demodulacja. Dlatego może być on tylko stosowany w wydzielonych łączach z dopasowaniem /Leased Conditioned Lines/. Z drugiej strony modem jest wyjątkowo tani. Według załącznika umieszczonego na końcu tej książki, cena wszystkich zastosowanych elementów wynosi około 150 dolarów. Cały modem, wraz z zestawem synchronizującym, może być umieszczony na jednej płytce drukowanej o standardowej wielkości.

Tak więc jakość poświęcono dla ceny. Jednak przy projektowaniu urządzeń handlowych, przeznaczonych dla szerokiego kręgu odbiorców jest to dozwolone, a czasami nawet wymagane. Przypomnijmy sobie tylko modulację delta /DM/ i jej porównanie z impulsową modulacją kodową /PCM/.

Za pomocą mikrokomputera można zrealizować również modem wyższej jakości. Rozpatruje się wówczas modulator i demodulator jako dwa filtry cyfrowe, których algorytmy zawierają określoną liczbę dodawań i mnożeń. Aby te algorytmy mogły być zrealizowano w czasie rzeczywistym, należy zastosować mikroprocesor bipolarny. Niestety, mnożenie zawsze stwarza kłopoty. Dlatego niezależ-

nie od wykorzystywania mikrokomputerów, opracowanych jest wiele metod, za pomocą których liczbę mnożeń w określonym algorytmie można zredukować. Oczywiście metody te mogą być również wykorzystane przy projektowaniu modemu z wykorzystaniem mikrokomputera. Problem mnożeń może być rozwiązany także przez zastosowanie scalonych układów mnożących. W naszym przykładzie uniknięto całkowicie mnożenia i sprowadzono je do stosowania tablic funkcjonalnych /Look-Up Tables/.

LITERATURA

A2, A3, B1, B2, B3, B5, B6, C4, C5, C6, C55, C56, C98, C112, D1, D6, F1, F2, G3, G4, K1, K2, K3.

# ELEMENTY PROCESORA KOMUNIKACYJNEGO

Analizując dostępną literaturę można wyciągnąć wniosek, że najwięcej opublikowanych prac z dziedziny zastosowań mikrokomputerów w telekomunikacji traktuje o zastosowaniu mikrokomputerów przy projektowaniu procesorów komunikacyjnych. Pod pojęciem procesora komunikacyjnego, w najszerszym znaczeniu tego słowa, rozumiane są zarówno procesory centralne, jak i terminale peryferyjne oraz inne urządzenia kontrolujące przesyłanie danych w sieci komputerowej.

Przesyłanie danych w sieci komputerowej może mieć charakter synchroniczny lub asynchroniczny.

Przy przesyłaniu synchronicznym, dane informacyjne pakowane są w bloki danych. Przed i za każdym blokiem danych przekazuje się określoną liczbę danych kontrolnych. Za pomocą danych kontrolnych realizuje się protokół komunikacyjny /Handshaking/.

## PROTOKÓŁ KOMUNIKACYJNY

Standardowy protokół komunikacyjny nie istnieje, chociaż prace w tym zakresie są bardzo zaawansowane. Każdy producent urządzeń sieci komputerowej posiada swój własny "znormalizowany" protokół. Tego rodzaju protokołem jest SDLC /Synchronous Data Link Control/, opracowany w firmie IBM. Protokół SDLC jest jednym z najczęściej stosowanych protokołów, dlatego poniższy rozdział traktuje właśnie o nim.

Na podstawie protokołu SDLC, po uzyskaniu połączenia, w pierwszej kolejności wysyła się dwa znaki synchronizujące

SYN /SYNchronous idle/

Siedmiobitowy kod ASCII dla znaku SYN jest:

0010110

Kontrola parzystości /Parity Check/ realizowana jest przez dodanie jednego bitu kontrolnego. Po umieszczeniu bitu kontrolnego w najstarszej pozycji bitowej kod znaku SYN jest:

10010110

Dwa znaki SYN mają za zadanie zrealizować synchronizację bitową przesyłania.

Następnym wysyłanym znakiem jest

ENQ /ENQuiry/

Wysyłając ten znak nadawczy procesor komunikacyjny "pyta", czy odbiorczy procesor komunikacyjny jest "gotowy" do przyjęcia polecenia. Kod ASCII dla znaku ENQ jest:

00000101

Po tym nadawczy procesor komunikacyjny "zatrzymuje się" i "czeka" na odpowiedź.

Jeżeli odbiorczy procesor komunikacyjny jest "gotowy", wówczas wkrótce nadchodzi znak

DIE /Data Link Escape/

Kod ASCII dla tego znaku jest:

10010000

Po przyjęciu znaku DIE nadawczy procesor komunikacyjny wysyła znak

STX /Start of TeXt/

W ten sposób nadawczy procesor komunikacyjny "zgłasza" rozpoczęcie wysyłania bloku danych. Kod

ASCII dla tego znaku jest:

10000010

Blok danych następuje zaraz za znakiem STX. Polecenie może składać się z kilku bloków tej samej długości. Między blokami oraz za ostatnim blokiem, wysyłane są również określone znaki kontrolne.

Program

Załóżmy, że nadawczy procesor komunikacyjny został zrealizowany za pomocą mikroprocesora z rezonatorem kryształicznym, którego częstotliwość wynosi 17,28 MHz tzn., że przesyłanie danych następuje z szybkością 2400 b/s. Ponieważ znaki ASCII składają się z 8 bitów - oznacza to, że do modemu przekazuje się jeden znak co 6400 okresów taktu podstawowego. Tak więc mikroprocesor MOS jest dostatecznie szybki do wykonywania wymagań protokołu komunikacyjnego, ponieważ w tym wypadku między przekazaniem dwóch następujących po sobie znaków może być wykonany odcinek programu, składający się z kilkuset rozkazów asemblera. Tak samo jest gdy procesor komunikacyjny jest sprzężony z kilkudziesięcioma terminalami.

Jednak w praktyce częste są sytuacje, w których procesor komunikacyjny sprzężony jest z kilkuset urządzeniami końcowymi. Wówczas jeden mikroprocesor nie jest dostatecznie szybki aby wykonać wszystkie postawione przed nim zadania. Jednym ze sposobów zwiększenia efektywnej szybkości mikrokomputera jest zastosowanie układu wielomikroprocesorowego. Wówczas mikrokomputer zawiera kilka mikroprocesorów. Każdy mikroprocesor wykonuje określoną funkcję, lub wszystkie mikroprocesory wykonują wszystkie funkcje, lecz każdy w trakcie określonego przedziału czasowego; który ze sposobów będzie wybrany, założeć to będzie od organizacji systemu wielomikroprocesorowego.

Wielomikroprocesorowanie znalazło szerokie zastosowanie przy projektowaniu procesorów komunikacyjnych, w najszerszym znaczeniu tego słowa.

My rozpatrujemy przykład, w którym procesor komunikacyjny jest sprzężony tylko z jednym modemem. Zakłada się, że mikroprocesor jest połączony z modemem przez jednostkę peryferyjną wejścia-wyjścia z adresem 00H oraz, że blok składający się z 200 danych 8-bitowych umieszczony jest w pamięci RAM, począwszy od adresu 0400H.

Program realizujący w pierwszej kolejności uprzednio opisany protokół, a następnie emitujący blok składający się z 200 danych ma następującą postać:

```
    MVI   A,10010110B       ;
    OUT   00H               ; WYSYŁA SIĘ SYN
    CALL  K6390
    OUT   00H                 WYSYŁA SIĘ SYN
-------------------------------------------------
    MVI   A,00000101B
    CALL  K6383
    OUT   00H               ; WYSYŁA SIĘ ENQ
-------------------------------------------------
A1: MVI   B,10010000B       ; CZEKA
    IN    00H               ; SIĘ
    CMP   B                 ; NA
    JNZ   A1                ; DLE
-------------------------------------------------
```

```
        MVI     B,10010000B
        OUT     00H             ; WYSYŁA SIĘ STX
        ---------------------------------------------------
        CALL    K6366
        LXI     HL,0400H
        MVI     B,0200D
        ---------------------------------------------------
A2:     MOV     A,M             ; WYSYŁA
        OUT     00H             ; SIĘ
        INX     HL              ; BLOK
        DCR     B               ; SKŁADAJĄCY SIĘ Z
        JZ      A3              ; 200
        CALL    K6353           ; INFORMACYJNYCH
        JMP     A2              ; DANYCH
        ---------------------------------------------------
A3:                             ; PROTOKOŁ JEST KONTYNUOWANY
        ---------------------------------------------------
```

Jako KNNNN oznaczony jest podprogram realizujący opóźnienie o NNNN okresów taktu podstawowego. Np. podprogram K6353 wprowadza opóźnienie trwające 6353 okresy taktu podstawowego. Takie opóźnienie umożliwia wysyłanie znaków kontrolnych oraz danych w równo odległych momentach.

Szkieletem podprogramu realizującego opóźnienie jest poprzednio opisany program liczący. W naszym przykładzie jako licznik może posłużyć również komórka pamięci. Wykorzystanie komórki pamięci jest uwarunkowane zastosowaniem adresowania pośredniego. Powiększenie zawartości komórki pamięci dokonywane jest za pomocą rozkazu

INR M

Odwrotna operacja dokonywana jest za pomocą rozkazu

DCR M

Oczywiście, równoległość momentów, w których wysyła się znaki kontrolne oraz dane może być zrealizowana również za pomocą systemu przerwania wg wcześniej opisanego sposobu.

UZYSKIWANIE POŁĄCZENIA

Protokół komunikacyjny może być wykonywany dopiero po uzyskaniu połączenia między dwoma procesorami komunikacyjnymi czyli między dwoma dołączonymi z nimi modemami. Nawiązywanie połączenia dokonywane jest za pomocą przewodów $\overline{\text{DTR}}$, $\overline{\text{DSR}}$, $\overline{\text{RTS}}$ i $\overline{\text{CTS}}$. Przewody te znajdują się zarówno w procesorze, jak i w modemie.

Po pierwsze, nadawczy procesor komunikacyjny tworzy niski poziom sygnału w przewodzie

$\overline{\text{DTR}}$ /Data Terminal Ready/

W ten sposób nadawczy procesor komunikacyjny "daje do zrozumienia" modemowi, że chce się komunikować. Jeżeli modem /Data Set/ jest "gotów", wówczas tworzy się niski poziom sygnału w przewodzie

$\overline{\text{DSR}}$ /Data Set Ready/

a następnie nadawczy procesor komunikacyjny za pośrednictwem swojego modemu wysyła do modemu odbiorczego procesora komunikacyjnego żądanie nawiązania połączenia. Osiąga się to przez stworzenie niskiego poziomu sygnału w przewodzie

$\overline{RTS}$ /Ready To Send/

W systemie pracy półdupleksowej w przewodzie

$\overline{CTS}$ /Clear To Send/

po 2 ms automatycznie tworzy się niski poziom sygnału. W systemie pracy dupleksowej w przewodzie $\overline{CTS}$ pojawi się niski poziom sygnału dopiero po wyzerowaniu przez odbiorczy procesor komunikacyjny sygnału $\overline{RTS}$.

Program

| $b_7$ | $b_6$ | $\overline{RTS}$ | $b_4$ | $b_3$ | $b_2$ | $\overline{DTR}$ | $b_0$ |
|---|---|---|---|---|---|---|---|

| $\overline{DSR}$ | $b_6$ | $b_5$ | $b_4$ | $b_3$ | $b_2$ | $b_1$ | $b_0$ |
|---|---|---|---|---|---|---|---|

Rys.87. Komórki peryferyjne, przez które uzyskuje się połączenie między dwoma procesorami komunikacyjnymi

Załóżymy, że nawiązywanie połączenia dokonywane jest przez dwie ośmiobitowe komórki peryferyjne z adresami 01H. Jedna jest komórką wyjściową, a druga wejściową. W wejściowej komórce peryferyjnej pozycja bitowa $b_1$ związana jest z przewodem $\overline{DTR}$, natomiast pozycja bitowa $b_5$ związana jest z przewodem $\overline{RST}$. W wyjściowej komórce peryferyjnej pozycja bitowa $b_7$ związana jest z przewodem $\overline{DSR}$. Te dwie komórki peryferyjne przedstawione są na rys.87.

Uwzględniając wszystko, co powyżej przedstawiono stwierdzamy, że program służący do nawiązania połączenia ma następującą postać:

```
        MVI     A,11111101B     ; ZERUJE SIĘ
        OUT     01H             ; DTR
-------------------------------------------------
A0:     IN      01H             ; CZEKA SIĘ
        ANI     10000000B       ; NA
        JNZ     A0              ; WYZEROWANIE DSR
-------------------------------------------------
        MVI     A,11011101B     ; ZERUJE SIĘ
        OUT     01H             ; RTS
-------------------------------------------------
        CALL    K2              ; CZEKA SIĘ 2 ms
-------------------------------------------------
```

K2 jest nazwą podprogramu realizującego opóźnienie trwające przez 2 milisekundy.

Za pomocą rozkazu

ANI 10000000B

wykonywana jest operacja logiczna I między zawartością akumulatora i zawartością operandu. Ogólną postacią tego rozkazu jest

ANI P8

Przez P8 oznaczona jest jedna dana 8-bitowa. Rozkaz ten składa się z dwóch bajtów. Pierwszy z nich definiuje operację a drugi operand. Kodem tego rozkazu jest

11100110

10000000

Za pomocą rozkazu ANI 10000000B określa się wartość najstarszego bitu w bajcie. Określanie naj-

starszego bitu w bajcie oparte jest na następującym równaniu logicznym:

XXXXXXXX • 10000000 = X0000000

Do rozkazu ANI P8 podobne są następujące rozkazy:

| | | |
|---|---|---|
| ORI | P8 | LUB |
| XRI | P8 | Logiczna różnica symetryczna /wyłączne LUB/ |
| CPI | P8 | Porównanie |
| ADI | P8 | Dodawanie |
| ACI | P8 | Dodawanie z przeniesieniem |
| SUI | P8 | Odejmowanie |
| SBI | P8 | Odejmowanie z "pożyczką" |

We wszystkich tych rozkazach stosuje się pewien specyficzny sposób adresowania. Chodzi o adresowanie chwilowe /Immediate Addressing/, w którym dana przetwarzana w trakcie wykonywania rozkazu znajduje się w samym rozkazie. Dana ta jest identyczna z drugim bajtem rozkazu.

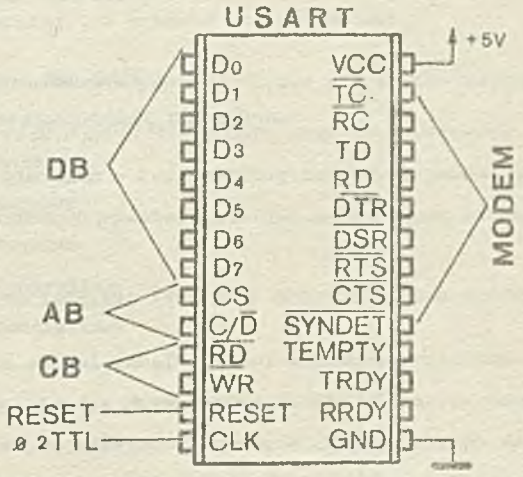
USART

Bezpośrednie sprzężenie mikrokomputera z modemem jest niemożliwe. Mikrokomputer traktuje jeden bajt jako 8 równoległych bitów, natomiast modem traktuje jako jeden kontrolny lub informacyjny znak 8 kolejnych bitów. Oznacza to, że między mikrokomputerem a modemem musi znajdować się równoległo-szeregowy lub szeregowo-równoległy konwerter.

Konwersja równoległo-szeregowa, jak i inne operacje kontrolne, wykonywana jest za pomocą specjalnej jednostki peryferyjnej w układzie o nazwie USART /Universal Synchronous Asynchronous Receiver Transmitter/.

Układ ten był już wymieniony, oznaczony jest kodem 8251 i przedstawiony na rys.88. Na rys.88 oznaczono również połączenie układu USART z innymi elementami mikrokomputerowymi oraz z samym modemem.

Dla programisty USART stanowi zbiór składający się z jednej komórki peryferyjnej wejściowo-wyjściowej oraz z trzech komórek sterujących, z których już dwie przedstawiono na rys.87. Wybieranie tych komórek peryferyjnych dokonywane jest za pomocą sygnałów $\overline{CS}$, $\overline{C/D}$, $\overline{RD}$ i $\overline{WR}$. Końcówki $\overline{CS}$ /Chip Select/ oraz C/D /Control Data/ łączy się z szyną adresową, zaś końcówki $\overline{RD}$ oraz $\overline{WR}$ z szyną sterującą. Końcówki RESET oraz CLK /Clock/ łączy się z końcówkami RESET oraz Ø 2TTL w układzie 8224. W stosunku do modemu USART zachowuje się tak jak pokazano na rys.89, tzn. jak układ z 5 wejściami i 3 wyjściami.



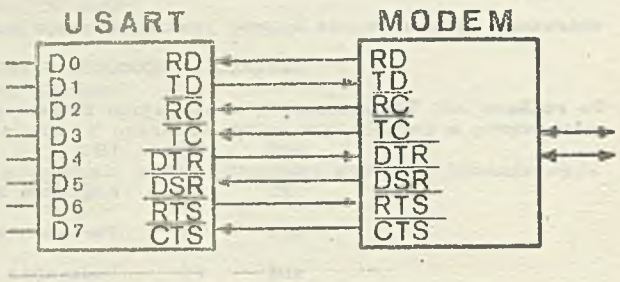

Rys.88. USART

Osiem równoległych bitów nadchodzących z szyny danych podlega konwersji i wyprowadzane jest przez zacisk $\overline{TD}$ /Transmit Data/ jako osiem kolejnych bitów / pierwszy najstarszy bit/. Kolejne dane z mo-

domu nadchodzą przez przewód $\overline{RD}$ /Receive Data/. Przez przewody $\overline{TC}$ /Transmit Clock/ oraz $\overline{RC}$ /Receive Clock/ modem podaje przebieg taktu zegara, z którym dokonuje się przesyłanie danych między modemem a procesorem komunikacyjnym.

Jedna z komórek sterujących w układzie USART nazywa się rejestrem instrukcji sterujących /Command Instruction Register/. Właśnie górna komórka peryferyjna z rys.87 jest takim rejestrem. Druga z trzech komórek kontrolnych nazywana jest rejestrem stanu. Na rys.87 jest to dolna komórka peryferyjna.

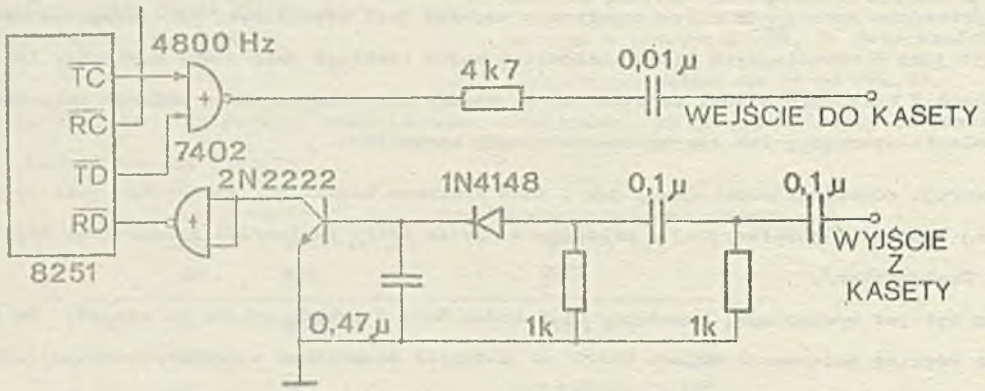
Sygnały $\overline{SYNDET}$, TEMPTY, TRDY oraz RRDY mają charakter kontrolny. Nie będziemy zagłębiać się w ich wyjaśnianie.




Rys.89. Połączenie między modemem a układem USART

Ogólnie mówiąc, możliwości układu USART są bardzo szerokie. Umożliwia on zarówno przekazywanie synchroniczne jak i asynchroniczne. Parametry przekazu mogą być zmienne. Dokonuje się tego przez wpisanie określonych zawartości do trzech komórek sterujących.

USART może być również używany do przesyłania danych między mikrokomputerem a taśmową pamięcią kasetową. W tym celu można stosować układ, którego schemat elektryczny przedstawiony jest na rys.90.




Rys.90. Zestaw służący do sprzężenia pamięci kasetowej z układem USART

Konkretne wartości z rys.90 dotyczą transmisji danych z szybkością 300 b/s.

Przesyłanie danych między mikrokomputerem a dyskami elastycznymi jest najprościej dokonywane za pomocą jednostki peryferyjnej w układzie 8271 /Programmable Floppy Disk Controller/. Ta jednostka peryferyjna dopiero niedawno pojawiła się na rynku.

## NAJNOWSZE UKŁADY SCALONE JEDNOSTEK PERYFERYJNYCH

Moc mikrokomputera zależy między innymi od możliwości jednostek peryferyjnych. Dlatego producenci elementów mikrokomputerowych starają się wprowadzić na rynek jak najwięcej różnych jednostek peryferyjnych. W ostatnim okresie pojawiło się więc wiele nowych układów scalonych zawierających jednostki peryferyjne, które są kompatybilne z mikroprocesorem Intel 8080. Ich skrócony opis jest tematem poniższej części pracy.

Układ 8279 / Programmable Keyboard/Display Interface / przeznaczony jest do projektowania urządzeń końcowych z klawiaturą oraz monitorem ekranowym opartym na mikrokomputerze. Jednak mo-

nitor ekranowy często wymaga, aby przedstawiane na nim dane były wyrażone w kodzie BCD. W tym celu mikroprocesor Intel 8080 za pomocą rozkazu DAA /Decimal Adjust Accumulator/ wykonuje bezpośrednią konwersję kodu naturalnego na kod BCD.

Układ 8275 /Programmable CRT Controller/ przeznaczony jest do projektowania urządzeń końcowych CRT /Cathode Ray Tube/ za pomocą mikrokomputera. W urządzeniu końcowym CRT ekran służy do komunikacji między maszyną a człowiekiem.

Układ 8273 /SDLC Protocol Controller/ przeznaczony jest do projektowania procesora komunikacyjnego z protokołem SDLC za pomocą mikrokomputera.

Układ 8253 /Programmable Interval Timer/ umożliwia programowe tworzenie sygnału cyfrowego o dowolnej postaci lub programową realizację dowolnego opóźnienia.

W tab.18 wymieniono są wszystkie jednostki peryferyjne dotychczas wspomniane, jak również wiele innych, o których jeszcze nie mówiliśmy. Układy te, oprócz firmy Intel, produkuje również firma National.

Tab. 18

| | | |
|---|---|---|
| 1671 | National | ASTRO Communications Interface |
| 2910 | Intel | PCM CODEC μ-Law |
| 2911 | Intel | PCM CODEC A-Law |
| 8202 | Intel | Dynamic RAM Controller |
| 8202 | National | Tri-State Octal Buffer |
| 8203 | National | Inverting Tri-State Octal Buffer |
| 8205 | Intel, Nat. | 1-of-8 Decoder |
| 8206 | National | Octal Latch |
| 8208 | National | Octal Bus Driver |
| 8212 | Intel, Nat. | Octal I/O Port |
| 8213 | National | Octal Bidirectional I/O Port |
| 8214 | Intel | Picu |
| 8216 | Intel | 4-Bit Bidirectional Bus Driver |
| 8226 | Intel | Inverting 4-Bit Bidirectional Bus Driver |
| 8244 | National | 90-Key Keyboard Encoder |
| 8245 | National | 16-Key Keyboard Encoder |
| 8246 | National | 20-Key Keyboard Encoder |
| 8247 | National | 4 Digit Display Controller |
| 8248 | National | 6 Digit Display Controller |
| 8250 | National | ACE |
| 8251 | Intel, Nat. | USART |
| 8252 | National | Advanced USART |
| 8253 | Intel, Nat. | Programmable Interval Timer |
| 8254 | National | 16-Bit Programmable Peripheral Interface |
| 8255 | Intel, Nat. | PPI |
| 8257 | Intel, Nat. | Programmable DMA Controller |
| 8259 | Intel, Nat. | PIC |

| | | |
|---|---|---|
| 8261 | National | Programmable Communications Subsystem |
| 8271 | Intel | Programmable Floppy Disk Controller |
| 8272 | National | Floppy Disk Formatter Controller |
| 8273 | Intel | Programmable Protocol Controller |
| 8274 | National | Multiprotocol Controller |
| 8275 | Intel | Programmable CRT Controller |
| 8276 | National | Programmable CRT Controller |
| 8278 | Intel | Programmable Keyboard Interface |
| 8279 | Intel | Programmable Keyboard Display Interface |
| 8283 | National | Advanced Protocol Controller |
| 8285 | National | Character Generator for CRT |
| 8292 | Intel | 8-Bit A/D |
| 8294 | Intel | Data Encryption Unit |
| 8294 | National | 3 3/4-Digit DVM |
| 8296 | National | Basic Interpretter |
| 8298 | National | Basic Debugger |
| 8692/3/4 | DS | Seiko Printer Interface |

Również inne firmy intensywnie pracują nad projektowaniem układów scalonych z jednostkami peryferyjnymi, np. Motorola skonstruowała nadajnik z modemem 2400 b/s z czterowstęgową różnicową modulacją fazową /M6862/, jak również odbiornik-nadajnik dla modemu od 200 do 600 b/s z binarną modulacją częstotliwości /M6860L/. Firma Collins Radio proponuje układ z jednostką peryferyjną do detekcji częstotliwości tonu na linii /Dual-Tone Multifrequency Detector/. Oznaczeniem tego układu jest CRC 8030; może on znaleźć zastosowanie w telefonii.

Aby informacja była kompletna w tab.19 wyliczono układy pamięci kompatybilne z mikroprocesorem Intel 8080. Również te układy, oprócz firmy Intel, produkuje firma National.

Tab.19

| | |
|---|---|
| 8356 | 2048x8 ROM, 128x8 RAM 1/O |
| 8154 | 128x8 Static RAM with 16-Bit 1/O |
| 8364/E | 8192x8 MOS Mask ROM |
| 8364E | 8192x8 MOS Mask ROM /2708 Compatibile/ |
| 8316A/E | 2048x8 MOS Mask ROM |
| 8332E | 4096x8 MOS Mask ROM /2708 Compatibile/ |
| 1702A | 256x8 EPROM |
| 8704 | 512x8 EPROM |
| 2708/8708 | 1024x8 EPROM |
| 8101A-4 | 256x4 Static PAM with Separate 1/O |
| 8111A-4 | 256x4 Stabic RAM with Common 1/O |
| 8102A | 1024x1 Static RAM |
| 74C920 | 256x4 CMOS Static RAM with Separate 1/O |
| 74C921 | 256x4 CMOS Static RAM with Common 1/O |
| 74C949 | 1024x1 CMOS Static RAM |
| 2114 | 1024x4 Static RAM |
| MM257 | 4096x1 Static RAM |
| DM875296 | 512x8 Bipolar PROM |
| 5290 | 16K Dynamic RAM |

| | |
|---|---|
| 8316A/E | Mask ROM /2708 Compatibile/ |
| DM74S472 | 512x8 Bipolar PROM/20-Pin DIP |
| MM5204 | 512x8 EPROM |

Pozostaje jeszcze wymienić dwa ostatnie rozkazy ze zbioru rozkazów mikroprocesora Intel 8080. Są to:

STC /Set Carry/

CMC /COMplement Carry/

Za pomocą pierwszego rozkazu zeruje się wskaźnik C, zaś za pomocą drugiego dokonuje się inwersji stanu tego wskaźnika.

LITERATURA

A4, C7, C8, C9, C11, C13, C14, C15, C85, D1, D6, D8, E5, E7, E8, F1, F2, F3, J1, J2, K1, K2, K3, K4.

## CZ.IV. ZAŁĄCZNIKI

# ROZWÓJ URZĄDZEŃ OPARTYCH NA MIKROKOMPUTERZE

Dwa podstawowe problemy należy rozwiązać w zakresie prac rozwojowych nad urządzeniem opartym na mikrokomputerze. Po pierwsze, należy napisać dokładny program. Po drugie, należy dokładnie połączyć wzajemnie układy, stanowiące elementy mikrokomputera.

## USUWANIE BŁĘDÓW

Tylko długie programy mają praktyczną wartość. Gdy program jest krótki, wówczas jest więcej niż pewne, że dany problem mógł być rozwiązany również w inny, tańszy sposób niż przez zastosowanie mikrokomputera.

Liczba instrukcji w programie w języku asemblera może wzrosnąć aż do kilku tysięcy. Dlatego jest bardzo prawdopodobne, że przy pisaniu programu popełni się określoną liczbę błędów. Błędy te należy usunąć, tzn. program musi być skorygowany /Software Debugging/.

Samo połączenie układów nie stanowi problemu. Dokładnie wiadomo jak połączyć układy, ale może się zdarzyć, że technik nie dokona jakiegoś połączenia, lub że niektóre lutowania będą zimne. Istnieje także możliwość, że sam inżynier-projektant popełni błąd wskutek niedostatecznej wiedzy o elementach, których używa. Dlatego należy również dokonać testowania sprzętem /Hardware Debugging/.

## ŚRODKI ROZWOJOWE

Cały splot wypadków może spowodować, że poszczególne błędy będą "niewidoczne". /Dotyczy to zarówno błędów programowych, jak i sprzętowych/. Błędy te napewno wynikną przy eksploatacji mikrokomputera. Dlatego powstaje zapotrzebowanie na specjalne środki, które maksymalnie ułatwią proces testowania i usuwania błędów.

Jednak usuwanie błędów stanowi tylko jedną fazę w rozwoju urządzenia opartego na mikrokomputerze. Proces rozwojowy obejmuje również inne fazy. Są to: pisanie programu źródłowego, tłumaczenie programu źródłowego na program wynikowy, wprowadzenie programu wynikowego do pamięci ROM, łączenie układów itd.

Rozwój można osiągnąć za pomocą specjalnego komputera, w pamięci którego znajduje się cały szereg specjalnych programów. Programy te umożliwiają tłumaczenie programu źródłowego na program wynikowy, wprowadzenie programu wynikowego do pamięci ROM, korektę, jak i wiele innych funkcji. Wspólną nazwą dla wszystkich tych programów jest wspomaganie programowe /Software Support/.

Komputor, o którym mowa, może być również minikomputerem opartym na mikrokomputerze. Taki komputer nazywany jest systemem wspomagania lub systemem rozwojowym /Development System/. Systemy wspomagania nie mają zazwyczaj zbyt wielkiej pojemności pamięci, ponieważ programy wspomagania programowego przechowywane są w jakimś nośniku pamięci, poza systemem rozwojowym. Do systemu rozwojowego wprowadzone są w zależności od potrzeb i to najczęściej tylko jeden po drugim. Jednak proces rozwoju może być prowadzony również na jakimkolwiek innym minikomputerze lub makrokomputerze /Framework/. Komputer ten może znajdować się w samym laboratorium lub w jakimś oddalonym miejscu. W tym drugim wypadku w laboratorium znajduje się tylko urządzenie końcowe, a łączność

z komputerem prowadzona jest przez sieć służącą do przekazywania danych /Time-Share Network/.

W Europie istnieją trzy sieci dostępne użytkownikom mikroprocesora Intel 8080. Są to: Timeshare, Timesharing LTD i Honeywell.

Proces rozwoju urządzenia opartego na mikrokomputerze jest bardzo ciężką i skomplikowaną pracą, nawet przy zastosowaniu najnowocześniejszych pomocy. Proces ten wymaga nie tylko doświadczenia, ale i inwencji, ale zasługuje na poświęcenie mu daleko większej uwagi niż uczyniono to w niniejszym opracowaniu.

Zastosowanie systemu wspomagania

Istnieje wiele systemów wspomagania,które mogą być stosowane przy pracy z mikroprocesorem Intel 8080. Różnią się one swoimi możliwościami. Najnowocześniejszym z nich jest Intellec MDS--800*.

System wspomagania jest najczęściej połączony z dalekopisem /Teletype/, używanym do wprowadzania programu i danych do pamięci systemu wspomagania, za pomocą jednego z programów pomocniczych, o czym dalej będzie mowa.

Dalekopis

Każdy dalekopis /teletajp/ wyposażony jest w drukarkę /Printer/ i klawiaturę /Keyboard/, której kształt jest podobny do klawiatury maszyny do pisania, lecz funkcjonalnie różni się w wielu szczegółach. W zasadzie drukarka może być zwykła lub wierszowa, jednak drugi rodzaj nie jest stosowany do pracy z dalekopisem. Kiedy program jest wprowadzany do nośnika pamięci systemu wspomagania, wówczas na wyjściu drukarki wyprowadza się jego kopia.

Dalekopis może być połączony z określonym nośnikiem pamięci. W praktyce za pomocą dalekopisu program źródłowy jest kodowany i doprowadzany do postaci odpowiedniej do wprowadzenia dla danego nośnika pamięci.

Najczęściej tym nośnikiem pamięci jest taśma papierowa. Dalekopis jest wówczas połączony z dziurkarką oraz czytnikiem taśmy papierowej /Paper Tape Puncher/Reader/. Jednak taśma papierowa jest, jak już to podkreślono, bardzo prymitywnym nośnikiem pamięci, i dlatego coraz częściej stosuje się jako nośnik pamięci taśmę magnetyczną lub dysk elastyczny. Dalekopis wtedy za pośrednictwem systemu wspomagania łączy się z urządzeniem zapewniającym komunikowanie się z taśmą magnetyczną lub z dyskiem.

Najistotniejszymi parametrami nośnika pamięci są: pojemność nośnika pamięci oraz szybkość wpisywania i odczytywania danych, tzn. szybkość, z jaką dokonuje się przesyłania danych przez nośnik pamięci.

Przeciętna szybkość przesyłania danych w wypadku urządzenia końcowego z perforatorem wynosi jeden bajt na 100 ms. Przeciętna szybkość przesyłania przez urządzenie końcowe z kasetą wynosi

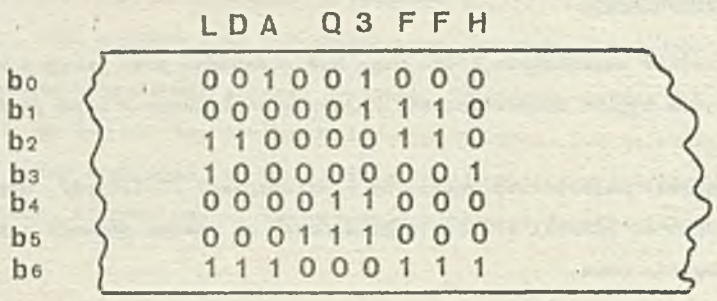
---

* Znane są już nowsze opracowania systemów wspomagania firmy Intel, mianowicie Intellec MDS-Model 210, 220, 230 i 240 /dop.red./.

1 bajt na 250 μs. Przeciętna szybkość przesyłania danych przez urządzenie końcowe z dyskiem elastycznym wynosi 1 bajt na 20 μs. Pojemność kasety wynosi ok. 200.000 bajtów, natomiast pojemność dysku elastycznego wynosi ok. 500.000 bajtów.

Przy wprowadzaniu programu źródłowego do nośnika pamięci koduje się każdy znak alfanumeryczny oraz pustą przestrzeń wewnątrz poszczególnych instrukcji[*]. Najczęściej stosuje się kod ASCII.

Jeżeli taśma papierowa jest wykorzystywana jako nośnik pamięci, wówczas kod jest równoległy /rys.91/. Wszędzie tam, gdzie na rys.91 znajduje się jedynka - na taśmie papierowej znajduje się okrągła dziurka.




Rys.91. Odcinek taśmy perforowanej z rozkazem LDA 03FFH /znaki alfanumeryczne kodowane są w kodzie ASCII/

Jeżeli jako nośnik pamięci stosuje się taśmę magnetyczną lub dysk elastyczny, wówczas kod jest szeregowy.

Edytor

Program wspomagania programowego, umożliwiający użytkownikowi wyprowadzenie swojego programu do pamięci systemu wspomagania, nazywa się edytorem. Przy wprowadzaniu programu łatwo można popełnić błąd. Dlatego często zachodzi potrzeba dodania lub usunięcia jednego lub kilku znaków alfanumerycznych. Również może zaistnieć potrzeba dodania lub usunięcia jednej lub kilku instrukcji. Edytor umożliwia takie czynności.

Asembler

Jeden z programów wspomagania programowego nazywa się asembler /Assembler/. Nie należy mylić programu assembler /Assembler/ z językiem programowym asembler /Assembly Language/. Jednak dla obu pojęć stosuje się ten sam termin.

Program asembler dokonuje tłumaczenia programu źródłowego na program wynikowy. On właśnie rozkaz

LDA 003FH

przetwarza do postaci

00111010
00111111
00000000

[*] Ta pusta przestrzeń odpowiada znakom spacji przy zapisie instrukcji w języku asemblera /dop.red/

Tłumaczenie poprzedzone jest wprowadzeniem zawartości taśmy papierowej lub magnetycznej, albo dysku do pamięci RAM systemu wspomagania. Dopiero po zakończeniu tej operacji wywoływany jest translator programowy asembler.

Proces tłumaczenia programu źródłowego na program wynikowy dokonywany jest w dwóch przebiegach asemblera /Assembly Pass/.
Przypomnijmy, że jeden rozkaz programu źródłowego składa się z czterech pól. W rozkazach, w których dokonywany jest skok programowy, drugi i trzeci bajt specyfikują etykietę rozkazu, do którego się "skacze". Skok może być dokonywany do poprzedzającego rozkazu lub do rozkazu, który nastąpi.

W pierwszym przebiegu asemblera wszystkim etykietom przypisuje się konkretne wartości adresów komórek pamięci. Jednocześnie kodowane są wszystkie rozkazy, oprócz rozkazów, za pomocą których dokonuje się skoku do jakiegoś rozkazu,z jakim asembler dopiero ma się zetknąć. W drugim przebiegu asemblera kodowane są również i te rozkazy. Asembler zgłosi błąd, jeżeli przy pisaniu jakiejś instrukcji nie są spełnione wszystkie wymagania formalne /odległość między polami, brak mnemonika itd./. Tak więc pierwsze błędy poprawiane są jeszcze w trakcie tłumaczenia /translacji/ Często trzeba wydrukować przetłumaczony program. Drukowanie realizowane jest w trakcie trzeciego przebiegu asemblera*.

Deasembler

Tłumaczenie w przeciwnym kierunku, tzn. tłumaczenie programu wynikowego na program źródłowy dokonywane jest za pomocą translatora programowego o nazwie desasembler /Disassembler/.

Dyrektywy asemblera

Przy tłumaczeniu i drukowaniu asembler przestrzega określonych poleceń wyspecyfikowanych przez programistę. Polecenia te noszą nazwę dyrektyw asemblera /Assembler Directives/. Np. programista musi wyspecyfikować dla asemblera początkowy adres programu /Origine Directive/. Musi się również oznaczyć ostatnią tłumaczoną instrukcję /End Directive/. Za pomocą dyrektyw asemblera mogą być specyfikowane konkretne wartości poszczególnych zmiennych w programie /Equate Directive/. Za pomocą dyrektyw asemblera można także wyspecyfikować typ zmiennej /Define Address Directive, Define Data Directive/. Cały szereg dyrektyw asemblera określa sposób drukowania danych wyjściowych /Output Format Request Directive, Printed Listing Control Directive/.

Niektóre asemblery pozwalają na stosowanie makroinstrukcji. Są to makroasemblery. Specjalne dyrektywy makroasemblera specyfikują pierwszą i ostatnią instrukcję danej makroinstrukcji /Start of Macro Directive, End of Macro Directive/.

Debager

Jeden z programów wspomagania programowego nosi nazwę debagera /Debugger/. Umożliwia on zatrzymanie programu po wykonaniu każdego rozkazu /Single Step Execution/. Umożliwia też po każdym

---

* Tak dzieje się zwykle przy wykorzystywaniu wyżej wspomnianego dalekopisu zarówno do wyprowadzania kodu wynikowego, jak i drukowania przetłumaczonego programu /listingu/ /dop.red./.

rozkazie wydrukowanie lub przedstawienie na ekranie zawartości dowolnego rejestru bądź zawartości dowolnej komórki pamięci lub peryferyjnej. Przez zastosowanie debagera znacznie upraszcza się proces testowania i korekty. Debager umożliwia zatrzymanie programu po przebiegu całego odcinka programu, co do którego jesteśmy pewni, że nie zawiera żadnego błędu. Program ten ma i inne możliwości, ale jest bezsilny w wypadku testowania przerwania i bezpośredniego dostępu do pamięci, tzn. ustalania błędów przy działaniu w czasie rzeczywistym.

Emulator

Program emulator jest podobny do programu debagera, ma jednak znacznie większe możliwości. Umożliwia testowanie przerwania oraz bezpośredni dostęp do pamięci w czasie rzeczywistym. Spotyka się go również pod nazwą Symulator.

Przy pracy z mikroprocesorem Intel 8080 stosuje się emulator oznaczony jako ICE-80 /In-Circuit/Emulator/. Za jego pomocą może dokonywać testowania i usuwania błędów zarówno programowych jak i sprzętowych.

Korekty sprzętowej można uniknąć, jeżeli zakupi się gotową, połączoną i wypróbowaną płytkę drukowaną z przylutowanymi elementami /Prewired Printed Circuit Card/. Płytki te wykonują nie tylko producenci mikroprocesorów; zajmują się tym często inne, mniejsze firmy.

Korekty programowej częściej stosowanych programów, można uniknąć przez członkowstwo w bibliotece programowej /User`s Library/ producentów elementów mikrokomputerowych. Roczna składka członkowska w bibliotece programowej Intel /8080/ wynosi 100 dolarów USA.

Program ładujący

Programy wspomagania programowego mogą, ale nie muszą, być stale umieszczone w pamięci systemu wspomagania. Jeżeli nie są stale umieszczone w pamięci, wówczas przechowywane są na taśmie papierowej, magnetycznej lub na dysku. Do pamięci systemu wspomagania wprowadzane są tuż przed rozpoczęciem pracy. Ich wprowadzenie do pamięci systemu wspomagania dokonywane jest za pomocą programu ładującego /Loader/. Program ten znajduje się stale w pamięci systemu wspomagania.

PL/M

Asembler jest "niskim" językiem programowania. Programowanie operacji matematycznych w niższych językach programowych jest bardzo skomplikowane. Jednej operacji matematycznej odpowiada czasem kilkadziesiąt instrukcji asemblera.

Fortran należy do wyższych języków programowania. W wyższym języku programowania jednej operacji matematycznej odpowiada tylko jeden rozkaz. Tak więc wyższe języki programowe są odpowiednie, gdy program musi wykonać wiele operacji matematycznych.

PL/M-80 jest wyższym językiem programowania, specjalnie ukierunkowanym na mikroprocesor Intel 8080. Program wspomagania programowego dokonujący tłumaczenia programu źródłowego, pisanego w wyższym języku programowania, na program wynikowy nosi nazwę kompilatora /Compiler/.

Jednak wyższe języki programowania nie są odpowiednie wówczas, gdy program musi wykonać wiele operacji logicznych, sterujących i wejściowo-wyjściowych. Jednocześnie niższe języki pro-

gramowania umożliwiają lepsze wykorzystanie przestrzeni pamięciowej w pamięci ROM. Dlatego korzystnie jest, gdy rozkazy niższego i wyższego języka programowania mogą być kombinowane.

Jednym słowem, jeżeli mikroprocesor służy jako cyfrowy element logiczny, wówczas bardziej opłaci się zastosowanie asemblera. Jeżeli mikroprocesor wykorzystany jest do projektowania komputera, wówczas bardziej opłaci się zastosowanie PL/M. Jeżeli zalety PL/M i asemblera ocenia się z punktu widzenia liczby wyprodukowanych urządzeń, wówczas zastosowanie asemblera opłaci się tym bardziej, im dłuższy jest program. Program napisany w języku PL/M wymaga większej pamięci ROM, a przy długich programach cena każdego zbytecznego rozkazu ma szczególne znaczenie.

Testowanie i korekta na niekompatybilnym komputerze

Różne komputery mają różnorodne rozkazy i różne kody wynikowe. Jeżeli prace rozwojowe mikrokomputera prowadzone są na niekompatybilnym komputerze, wówczas tłumaczenie programu źródłowego na program wynikowy prowadzone jest za pomocą asemblera skrośnego /Cross-Assembler/.

Asembler skrośny jest to program asemblera napisany dla komputera z jedną listą rozkazów posiadających jeden wynikowy kod, dokonujący tłumaczenia programu źródłowego, napisanego w języku opartym na innym zbiorze rozkazów i posiadającego inny kod wynikowy. W odróżnieniu od asemblera skrośnego uprzednio omawiany asembler nosi nazwę asemblera rezydentnego /Resident Assembler/. Tak samo istnieją kompilator własny i skrośny, debager rezydentny i skrośny, emulator rezydentny i skrośny itd.

Programy te kupuje się oddzielnie, w formie pakietów programowych. Zapisywane są na taśmie papierowej lub magnetycznej, albo na dysku.

Zastosowanie komputera niekompatybilnego jest celowe tylko wtedy, gdy znajduje się on w laboratorium w momencie, gdy rozpoczęto prace z mikroprocesorem.

MIKROKOMPUTER Z PRZEZNACZENIEM DYDAKTYCZNYM

Poszczególne firmy produkują specjalne płytki drukowane, na których znajduje się mikrokomputer, prosta klawiatura kalkulatorowa do wprowadzania programu wynikowego, jak również wskaźnik segmentowy monitora ekranowego, na którym można zobaczyć zawartość dowolnego rejestru, bądź dowolnej komórki pamięci lub peryferyjnej. Znajduje się tu również pamięć ROM wraz z programem monitora, umożliwiającym pewien rodzaj prymitywnego poprawiania programu. Wyroby te /np. MCS-80 Prototype Kit with 8080 CPU, SDK-80 System Design Kit/ służą do nauczania i poprawiania prostszych programów. Elementy mikrokomputera nie zawsze są połączone. Często pozostawia się użytkownikowi, aby sam wykonał połączenie /Ready-to-Assemble Kit/.

Produkcja i sprzedaż takich i podobnych wyrobów jest na zachodzie intratna. Najczęstszymi odbiorcami są hobbyści.

SEMINARIA DYDAKTYCZNE

Organizowanie seminariów dydaktycznych jest również na zachodzie przedmiotem zainteresowania. Istnieją firmy, które między innymi zajmują się organizacją takich seminariów /SYBEX, ICS/.

Są one przeznaczone zarówno dla początkujących jak i dla doświadczonych inżynierów-projektantów, pragnących uzupełnić dwoją wiedzę. Tematy takich seminariów są najrozmaitsze /np. mikroprocesory w telekomunikacji, nowe jednostki peryferyjne, praca z dyskiem elastycznym itd./. Seminaria stoją zazwyczaj na wysokim poziomie i są kosztowne.

SYMPOZJA NAUKOWE

Poza seminariami dydaktycznymi coraz częściej organizowane są sympozja naukowe. W Europie sympozja naukowe na najwyższym poziomie organizuje Europejskie Stowarzyszenie Mikroprocesorów i Mikroprogramowania, utworzone w 1973 r. Bardziej znane jest pod skrótem EUROMICRO.

LITERATURA

D1, D2, D5, D6, E1, E2, G1, G2, I1, J1, J2, K1, K2.

# ZABEZPIECZENIE MIKROKOMPUTERA

Technologia MOS jest czuła na ładunki elektrostatyczne. Dotyczy to również techniki ECL w technologii bipolarnej. Mikrokomputery są również czułe na zakłócenia impulsowe. Szczególnie jest to wyraźne w zastosowaniach telekomunikacyjnych. Zakłócenia impulsowe mogą przeniknąć do mikrokomputera z linii lub przez zasilanie. Zarówno ładunki elektrostatyczne jak i zakłócenia impulsowe mają działanie destruktywne.

## ŁADUNKI ELEKTROSTATYCZNE

Na człowieku może nagromadzić się elektryczności statycznej około kilku kilowoltów. Szczególnie niebezpieczne są podeszwy plastykowe w pomieszczeniach z wełnianymi dywanami. Na elektryczność statyczną szczególnie czułe są końcówki wejściowe układów wykonanych w technologii MOS.

Jeżeli użytkownik jest naelektryzowany i dotknie ręką końcówki wejściowej, wówczas zachodzi niebezpieczeństwo zniszczenia układu. Dlatego należy wprowadzić określone środki ostrożności. Podłoga w laboratorium nie powinna być pokryta. Dobrze, jeżeli krzesło robocze wykonane będzie z metalu. Biurko musi być pokryte płytą aluminiową. Płytę tę przed rozpoczęciem pracy należy potrzeć rękami. W ten sposób dokonuje się rozelektryzowania.

Niektórzy idą nawet tak daleko, że na każdy palec obu rąk zakładają pierścień metalowy, połączony przewodem z uziemionym punktem. Mikrokomputer jest w ten sposób całkowicie zabezpieczony, ale nie użytkownik.
Układy wykonane w technologii MOS sprzedawane są ze zwarciowo połączonymi końcówkami. Zwieracz zdejmuje się dopiero po wciśnięciu układu w podstawę bądź po zalutowaniu na płytce drukowanej.

Istnieją zastosowania, w których nie wykorzystuje się wszystkich końcówek w mikrokomputerze. Jak już podkreślono, niewykorzystane końcówki wyjściowe pozostawione są i "wiszą". Na niewykorzystanych końcówkach wejściowych poziom napięcia musi być stale niski lub stale wysoki. W pierwszym wypadku końcówki łączy się z masą, w drugim łączy się przez inwerter z masą lub przez opornik o oporności wynoszącej od 1K - 10K do +5 V. Pierwszy sposób jest lepszy, drugi natomiast tańszy.

## SZUM MOCY

Moc impulsów napięciowych, przenikających z linii lub przez zasilanie, może być dostatecznie wielka aby zniszczyć określoną liczbę układów w mikrokomputerze.

Dla zabezpieczenia wejściowych i wyjściowych jednostek peryferyjnych używa się m.in. warystorów Transzorb. Te warystory działają jak diody Zenera o wielkiej szybkości i mocy. Produkowane są dla różnych napięć, między innymi dla 5,12 i 15 voltów. Łączy się je z zasilaniem oraz z zewnętrznymi końcówkami jednostek peryferyjnych.

Dla mikrokomputera opartego na mikroprocesorze Intel 8080 przewidziane są warystory MPTE-5, MPTE-12 oraz MPTE-15. Ich producentem jest General Semiconductor Industries, Inc.

## LITERATURA

F8, F9, J1, J2, K1, K2.

## CENY NIEKTÓRYCH ELEMENTÓW ŚRODKÓW POMOCNICZYCH

W każdym projektowaniu należy liczyć się z czynnikiem ekonomicznym. Z tego powodu podajemy ceny niektórych składników i środków pomocniczych wymienionych w tej pracy. Podane ceny obowiązywały w końcu 1977 r. na rynku USA. Wyrażone są w dolarach amerykańskich. Oczywiście należy je brać warunkowo. W sprawie rzeczywistych cen należy zwrócić się do najbliższego producenta lub jego przedstawiciela.

ELEMENTY

Symbol C dotyczy opakowania ceramicznego. Symbol P - opakowania plastykowego. Symbol M - oznacza, że element pracuje w zakresie temperatur wyznaczonych normami wojskowymi /MIL-STD-8839/. W pierwszej kolumnie znajduje się cena jednostkowa dla zamówień poniżej 25 sztuk, natomiast w drugiej kolumnie znajduje się cena dla zamówień przekraczających 100 sztuk. Gwiazdka oznacza, że cena elementu ma tendencję spadkową.

| Element | Cena jednostkowa /do 25 sztuk/ | Cena jednostkowa /powyżej 100 sztuk/ |
|---|---|---|
| C 8080A | 19.00* | 14.30* |
| P 8080A | 15.00* | 11.25* |
| M 8080A | 53.20 | 35.95 |
| C 8085 | 34.00 | 24.00 |
| P 8085 | 27.00 | 19.00 |
| M 8101A | 12.50 | 9.85 |
| C 8102A-4 | 6.15 | 4.10 |
| P 8102A-4 | 4.20 | 2.80 |
| M 8102A-4 | 10.30* | 7.25* |
| C 8205 | 6.20 | 4.20 |
| P 8205 | 4.35 | 2.80 |
| C 8212 | 8.70 | 5.30 |
| P 8212 | 4.60 | 2.90 |
| M 8212 | 11.80 | 9.60 |
| C 8214 | 9.20 | 5.75 |
| P 8214 | 7.40 | 4.65 |
| M 8214 | 16.80 | 10.90 |
| C 8216 | 6.40 | 3.90 |
| P 8216 | 3.90 | 2.75 |
| M 8216 | 10.70 | 8.00 |
| C 8224 | 8.10 | 4.80 |
| P 8224 | 6.50 | 4.30 |
| M 8224 | 17.75 | 11.40 |
| C 8226 | 6.40 | 3.90 |
| P 8226 | 3.90 | 2.75 |
| M 8226 | 10.70 | 8.00 |
| C 8228 | 10.00 | 7.10 |
| P 8228 | 8.20 | 6.40 |
| M 8228 | 21.40 | 15.85 |

| | | |
|---|---|---|
| C 8243 | 9.20 | 6.20 |
| P 8243 | 8.00 | 5.00 |
| C 8251 | 13.50 | 10.85 |
| P 8251 | 12.50 | 10.00 |
| M 8251 | 28.15 | 24.30 |
| C 8253 | 24.25 | 17.55 |
| C 8255A | 13.50 | 10.85 |
| P 8255A | 11.10 | 7.40 |
| M 8255 | 38.10 | 28.50 |
| C 8257 | 28.75 | 20.00 |
| P 8257 | brak danych | |
| C 8259 | 23.65 | 17.20 |
| P 8259 | brak danych | |
| C 8275 | 100.00 | 62.50 |
| C 8279 | 20.00 | 14.10 |
| C 8702A | 18.30 | 11.70 |
| M 8702A | 33.80 | 22.50 |
| C 8708 | 38.00[x] | 25.65[x] |
| M 8708 | 75.35[x] | 51.55[x] |
| MM 57558 | 100.00 | 65.00 |
| MP10 | 40.00 | brak danych |
| MP20 | 140.00 | - " - |
| SN 7404 | 0.60 | 0.40 |

SPRZĘTOWE ŚRODKI WSPOMAGANIA

| | | |
|---|---|---|
| MCS-80 | Prototypo Kit with 8080 CPU | 125.00 |
| SDK-80 | 8080 System Design Kit | 250.00 |
| MDS-800 | Development System | 3950.00 |
| MDS-PLM | PL/M Compiler | 975.00 |
| MDS-016 | 16K RAM Module /2107/ | 975.00 |
| MDS-416 | 16K ROM Module /8708/ | 295.00 |
| MDS-PTR | High-Spoed Paper Tape Reader | 1150.00 |
| MDS-2DS | Diskette Operating System | 3350.00 |
| MDS-CRT | Alphanumeric Keyboard Cathode Ray Tube Display Concole | 2240.00 |
| MDS-PRN | Buffered Printer | 3200.00 |
| MDS-80-ICE | 8080 In-Circuit Emulator | 1750.00 |
| UPP-878 | 8708/8704/2708/2704 Personality Card | 325.00 |

WSPOMAGANIE PROGRAMOWE

| | | |
|---|---|---|
| CRONIS | Micro Assembler | 1250.00 |
| MAC 80 | Macro Assembler | 1250.00 |
| PL/M-80 | PL/M Compiler | 1250.00 |
| INTERP/80 | 8080 Simulator | 750.00 |

WNIOSEK

Z podanych cen wynika, że elementy są względnie tanie, natomiast środki wspomagania wyjątkowo kosztowne. Często można usłyszeć opinię, że producenci środków wspomagania nie osiągają na nich żadnego zysku, lecz produkują je tylko dlatego, aby łatwiej sprzedawać elementy. Ciekawe w jakim stopniu jest to prawdą, ponieważ istnieją firmy zajmujące się wyłącznie rozwojem i produkcją nowych środków przeznaczonych do rozwoju urządzeń opartych na mikrokomputerach.

LITERATURA

I1, K3, K4.

## KOD ASCII

Kod ASCII jest najczęściej stosowany w praktyce. Niżej przedstawiono tablice z siedmio- i ośmiobitowym kodem.

KOD SIEDMIOBITOWY /1968/

Przedstawiony jest za pomocą cyfr binarnych, przy czym $a = b_0$, $b = b_1$, $c = b_2$, $d = b_3$, $e = b_4$, $f = b_5$, $g = b_6$, $h = b_7$.

| gfe / dcba | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|---|---|---|---|---|---|---|---|---|
| 0000 | NUL | DLE | SP | 0 | @ | P | ` | p |
| 0001 | SOH | DC1 | ! | 1 | A | Q | a | q |
| 0010 | STX | DC2 | " | 2 | B | R | b | r |
| 0011 | ETX | DC3 | # | 3 | C | S | c | s |
| 1100 | EOT | DC4 | $ | 4 | D | T | d | t |
| 1101 | ENQ | NAK | % | 5 | E | U | e | u |
| 1110 | ACK | SYN | & | 6 | F | V | f | v |
| 1111 | BEL | ETB | ' | 7 | G | W | g | w |
| 0000 | BS | CAN | ( | 8 | H | X | h | x |
| 0001 | HT | EM | ) | 9 | I | Y | i | y |
| 0010 | LF | SUB | * | : | J | Z | j | z |
| 0011 | VT | ESC | + | ; | K | [ | k | { |
| 1100 | FF | FS | , | < | L | \ | l | \| |
| 1101 | CR | GS | - | = | M | ] | m | } |
| 1110 | SO | RS | . | > | N | ↑ | n | ~ |
| 1111 | SI | US | / | ? | O | ← | o | DEL |

KOD OŚMIOBITOWY

Przedstawiony jest za pomocą cyfr heksadecymalnych

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 20 | blanko | 30 | 0 | | | 50 | P | | | 70 | p |
| 21 | ! | 31 | 1 | 41 | A | 51 | Q | 61 | a | 71 | q |
| 22 | " | 32 | 2 | 42 | B | 52 | R | 62 | b | 72 | r |
| 23 | # | 33 | 3 | 43 | C | 53 | S | 63 | c | 73 | s |
| 24 | $ | 34 | 4 | 44 | D | 54 | T | 64 | d | 74 | t |
| 25 | % | 35 | 5 | 45 | E | 55 | U | 65 | e | 75 | u |
| 26 | & | 36 | 6 | 46 | F | 56 | V | 66 | f | 76 | v |
| 27 | ' | 37 | 7 | 47 | G | 57 | W | 67 | g | 77 | w |
| 28 | ( | 38 | 8 | 48 | H | 58 | X | 68 | h | 78 | x |
| 29 | ) | 39 | 9 | 49 | I | 59 | Y | 69 | i | 79 | y |
| 2A | * | 3A | : | 4A | J | 5A | Z | 6A | j | 7A | z |
| 2B | + | 3B | ; | 4B | K | | | 6B | k | | |
| 2C | , | 3C | < | 4C | L | | | 6C | l | | |
| 2D | - | 3D | = | 4D | M | | | 6D | m | | |
| 2E | . | 3E | > | 4E | N | | | 6E | n | | |
| 2F | / | 3F | ? | 4F | O | | | 6F | o | | |

LITERATURA

D1, D6.

## LITERATURA


KSIĄŻKI Z DZIEDZINY TELEKOMUNIKACJI

A1. Gold, Rader: Digital Processing of Signals; McGraw-Hill, Inc.; 1969.

A2. Lucky, Salts, Weldon: Principles of Data Communication; McGraw-Hill, Inc.; 1968.

A3. Helstrom: Statistical Theory of Signal Detection; Pergamon Press, Inc.; 1968.

A4. Martin: Teleprocessing Network Organization; Prentice-Hall, Inc.;1970.

A5. Petrovič: Peredača diskretnoj informacii v kanalah s fazovoj manipulacioj; Svjaz, Moskva, 1965.

ARTYKUŁY Z DZIEDZINY TELEKOMUNIKACJI

B1. Baker: Phase-Modulation Data Sets for Serial Transmission at 2,000 and 2,400 Bits per Second; AIEE Transactions; July 1962.

B2. Glanco: Power Spectra of Multilevel Digital Phase-Modulated Signals; BSTJ; Nov. 1971.

B3. Jovičić, Milutinović, Nedić, Zisić: Modem PP-2400; XI jugoslovenski simpozij o telekomunikacijah; Ljubljana; Okt. 1977.

B4. Lucky: Automatic Equalisation for Digital Communication; BSTJ; Apr. 1965.

B5. Nedić: Uobličavanje spektra fazno modulisanog signala; XI jugoslovenski simpozij o telekomunikacijah; Ljubljana; Okt. 1977.

B6. Rosenbaum: Error Performance of Multiphase DPSK with Noise and Interferences, IEEE Trans. on Commun. Techn.; June 1970.

ARTYKUŁY Z ZAKRESU ZASTOSOWANIA MIKROPROCESORÓW W TELEKOMUNIKACJI

C1. Lemack, Morris, Savell: Automatic Tactical Military Switchboard; Military Electronics Defence Conference Proceedings; Oct. 1976; Wiesbaden.

C2. Benton: Microprocessors in Telecommunications;UK Post Office Research Centre Report; Sep. 1976.

C3. Rees: Programming a Microprocessor to Act as a Digital Filter; Wireless World; Oct. 1976.

C4. Van Gerwen et al.: Microprocessor Implementation of High-Speed Data Modems; IEEE Transaction on Communications; Feb. 1977.

C5. Harris: Microcomputer Implemented CCITT V.27 Switched Network 4,800 bps Modem; Communications International; Nov. 1976.

C6. Boume /Intel/: Trends in Intelligent Peripheral Chip Design; 13th IEEE Computer Society International Conference; Washington, D.C., USA; Sep. 1976.

C7. Carter: Microprocessors in Data Communications; Communications International; Mar. 1975.

C8. Older: A Front-End Preprocessor for 16-Bit Minicomputer Hosts; Communications International; Jan. 1975.

C9. Parasurman: LSI Microprocessors in Telecommunications; Communications International; Mar. 1975.

C10. Butler, Cantoni: Nonitorativo Automatic Equalization; IEEE Trans. on Communications; Jun. 1975.

C11. Sahanian: Computer/Cassette Interface takes Tone from Clock; Electronics; 6 Jan. 1977.

C12. Hobbs, Spencer: The Role of Minicomputers and Microprocessors in Distributed Systems; Infotech. Internat.; Maidenhead, Berks., England, 1976.

C13. Dromard, Ebelle, Gauthier: Implementation of a Data Communication Line Control Procedure on a Microprocessor; Microarchitecture of Computer Systems - Euromicro Simposium; Nice; June 1975.

C14. Rhodes: Designing MOS-LSI Circuit for Telecommunications Applications; Electronic Engineering Jan. 1976.

C15. Spademan: Microprocessors and Data Terminals; IEEE Transaction on Industrial Electronics and Control Instrumentation; Aug. 1975.

C16. Zaugg: Mikroprozessoren als Ersatz Klassischer Hardware in kleingn Teilnehmeranlagen; Bull. Sev.; Vol.67; 1976.

C17. Stutz: Mikroprozessorgesteuerter Konzentrator; Bull. Sev.; Vol.67; 1976.

C18. Ziedler, Urlich: Neue Technologien Beeinflussen die Nachrichten-technischen Einrichtungen; Elektr. Nach. ITT; Vol.51; 1976.

C19. Geitz, Hoerbst, Sauer: Software-Entwicklungssystem fuer Mikroprozessoren; Elektronik; Vol. 25; 1976.

C20. Fullagar, Bradshaw, Evans: Interfacing Data Converters and Microprocessors; Electronics; Vol. 49; 1976.

C21. Carter, Milgo: Microprocessors in Data Communications; Communications International; Vol.3; 1976.

C22. Smith: USART - A Universal up Interface for Serial Data Communication; EDN; Vol.21; 1976.

C23. Mohar: Koncepti Sodobnih PAEK Sistemov; X jugoslovenski simpozij o telekomunikacijah, Ljubljana; Oktobar 1976.

C24. Dezman, Osolnik: Koncept EPARK 100; X jugoslovenski simpozij o telekomunikacijah, Ljubljana; Oktobar 1976.

C25. Cheron: The Connection of an HP 65 Pocket Calculator as a Microprocessor in a Control Loop; Autom. and Ind.; Apr. 1976.

C26. Wiebe, Stevens: Digital Integrator Enhances Echo Sounder's Accuracy; Electronics; Apr. 1976.

C27. Gerke /Siemens AG, München, Germany/: Interaction Between Circuit Technology and System Concepts in the Field of Digital TDM Switching; International Conference on Digital Communications; Zürich, Switz; Mar. 1976.

C28. Skytte: Incorporate a Calculator Chip; Electron. Design; Jan. 1976.

C29. O'Grady: Architectures for Real-Time Digital Channel Simulators; International Telemering Conference; Silver Spring, USA; Oct. 1975.

C30. Hakimoglu: Microprocessor Controlled PCM Telemetry System; International Telemering Conference; Silver Spring, USA; Oct. 1975.

C31. Livon: A Programmable Logic Array /PLA/; Electron and Microelectron Ind.; Feb. 1976.

C32. Poter, Machen, Naivar, Elkins, Simmonds: CAMAC Based Computer-Computer Communications via Microprocessor Data Links; IEEE Trans. on Nucl. Sci.; Feb. 1976.

C33. Kirk: MPU Display Driver; New Electron. /GB/; 6. April 1976.

C34. Lavelle, Da Cunha: Data Compression Through Microprocessors for Data Transmission; Computer Science Conference; Washington, DC, USA; Feb. 1975.

C35. Deeforth, Halsall, Woollons: Further Work on Data Communication Systems at the University of Sussex; International Meeting on Mini-Computers and Data Communications; Liege; Jan. 1975.

C36. Hussband: The Application of Microprocessors to Low Data Rate Information Distribution Systems; Computer Science Conference; Washington, DC, USA; Feb. 1975.

C37. Watanabe: Distributed Processing, Signal Multiplexing and Computer-Generated CRT Displays; ZEE /Japan/; Nov.-Dec. 1975.

C38. Buedel, Wolff: EAX Microprocessor-Controlled Universal Message System; Autom. Electr. Tech. J. /USA/; Jan. 1976.

C39. Hendrickson, Wallace: Design of a Microprocessor-Based Data System for Chromatography; Am. Lab. /USA/; Sep. 1975.

C40. Moore: MPU-Controled Filter; New Electron /GB/; Feb. 1976.

C41. Torrero: The Multifaces of $I^2L$; IEEE Spectrum; June 1977.

C42. Petritz: The Pervasive Microprocessor: Trends and Prospects; IEEE Spectrum; July 1977.

C43. Brainard, Sosnowski /Bell Labs/: A Microprocessor-Based Visual Communications Terminal; 1976 IECI Anual Congress on Industrial Applications of Microprocessors; Philadelphia, USA; Mar. 1976.

C44. Wallingford, Cavill: Microprocessor Matched Filtering of Emergency Chirp Signals; IEEE Tr. on Aerospace and Electronic Systems; July 1977.

C45. Stone: $I^2L$; Solid State Technology; June 1977.

C46. Prommer / Siemens AG, München, Germany/: Microprocessors: A new Dimension For Logic Circuitry; Mess. and Pruef.; Mar. 1976.

C47. Allan: ICS: From Sensors to Displays; IEEE Spectrum; Apr. 1976.

C48. Gisson: LSI in Communications: A Tool For Effecting Product Leadership; International Conference on Communications; Philadelphia, USA; June 1976.

C49. Huber: A Monitoring System Fault and Time Information on Distribution Network Breakdowns; Elektro-Anz.; Apr. 1976.

C50. Castellani, Del Corso, Giordana: Communication Interface For High-Speed Synchronous Telephone Line Connection of a Master Computer with Several Interactive Video Terminals; Euromicro Newsl.; Jan. 1976.

C51. Hollom: Microprocessors in Communications: International Conference on Communications Equipment and Systems; Brighton, England; June 1976.

C52. Forney, Vander May: 8-Bit Microprocessors Can Control Data Networks; Electronics; 24-June 1976.

C53. Hussein: MPU Data Multiplexer; New Electron.; 4.May 1976.

C54. Kitai, Renyi, Vajda: Using a Microprocessor in a Walsh-Furior Spectral Analyser; Computer; Apr. 1976.

C55. Logan: MOS/LSI Multiple Configuration 9600 b/s Data Modem; International Conference on Communications; Philadelphia, USA; June 1976.

C56. Nash: MOS Implementation of Low Speed Modems; International Conference on Communications; Philadelphia, USA; June 1976.

C57. Forney: Application of LSI Microprocessors in Data Network Hardware; International Conference on Communications; Philadelphia, USA; June 1976.

C58. Walter: Programmable Communications Terminal Optimization Using Systematic Digital Signal Compression Procedures; IEEE International Conference on Acoustics, Speech and Signal Processing; Philadelphia, USA; April 1976.

C59. Peled: A Digital Signal Processing System; IEEE International Conference on Acoustics, Speech and Signal Processing; Philadelphia, USA; April 1976.

C60. Graham: Some Aspects of Driving L.E.D. Displays; New Electron.; 5 Oct. 1976.

C61. Fleming: Electronic Switching: No Universal Design; Teleph. Eng. and Manage.; 15 June 1976.

C62. Haas: Peripheral Processors in the Integrated Communication System IFS-1; Bull. Assoc. Suisse Electr.; 18 Sep. 1976.

C63. Ahmed, Jayapalan: On Digital Filter Implementation via Microprocessors; IEEE Trans. on Ind. Electron. and Control Instrum.; Aug. 1976.

C64. Stutz: Microprocessor-Controlled Concentrator; Bull. Assoc. Suisse Electr.; 18 Sep. 1976.

C65. Lippman, Russo, Marcantonio: A Microprocessor Controlled Store-and-Forward Communications System; IEEE International Symposium on Circuits and Systems; Newton, USA; 21 Apr. 1975.

C66. Ramirez: A Multiprocessor as a Data Concentrator; Rev. Inf. and Autom.; Vol.9, No.28; 1976.

C67. Titus: Data-Acquisition System Built Modularly Around Intel 8080; Electronics; 14. Oct.1976.

C68. Kožuh, Starc: Mikroprocesor v telefonski centrali; XI jugoslovenski simpozij o telekomunikacijah; Ljubljana, Okt. 1977.

C69. Pipan, Gučtin: Realizacija funkcijskih modelov z mikroprocesorjem; XI jugoslovenski simpozij o telekomunikacijah; Ljubljana; Okt. 1977.

C70. Milutinović, Nedić: Predlog za realizaciju predajnika za modem 2400 b/s pomoću MOS mikroprocesora, XI jugoslovenski simpozij o telekomunikacijah; Ljubljana, Okt. 1977.

C71. Schuckert, Polco: Applied Ideas - Microprocessor Drives Segment Display Direct; Electronic Engineering; Aug. 1977.

C72. Stumer: A Floppy Disc Interface for 8080 System; Electronic Engineering; Aug. 1977.

C73. Higuchi, Saito, Kanomata: A Microprocessor-Based Digital Filter Programmed in a Block Diagram Language; IEEE Transaction on Industrial Electronics and Control Instrumentation; Aug. 1977.

C74. Brofferio, Soire: Impiego del Microcalcolotore per la Gestione della Segnalazione Telefonica d'Utente; Alta Frequenza; Aug. 1977.

C75. Bouwman, De Jong, Lindelov: Philips Teleprinters PACT 200 and 500; Philips Telecommun. Rev.; June 1976.

C76. Hillman: Trends in Telephone Test Equipment; International Conference on Communications; Philadelphia, USA; June 1976.

C77. Trosch: Trends in Instrumentation to Meet International Transmission Test Requirements for Local Distribution Networks; International Conference on Communications; Philadelphia, USA; June 1976.

C78. Dommelmeier, Kolmegies, Weiss: A Modular Microprogrammable Pipeline Signal Processor in ECL Technology; III Euromicro Simposium; Amsterdam; October 1977.

C79. Serain, Signor: Use of Microprocessors as Communication Front-End to Satelite Processors in a Processor Network; III Euromicro Simposium; Amsterdam; October 1977.

C80. Layec, L'Hostis: Microprocessors and Packet Switching Network; III Euromicro Simposium; Amsterdam; October 1977.

C81. Dromard, Lafage: A Multi-Transmission Procedure Microprogrammed Controller; III Euromicro Simposium; Amsterdam; October 1977.

C82. Altman: The New Concept for Memory and Imaging: Charge Coupling; Weber /editor/: Large and Medium Scale Integration, Devices and Applications ; McGraw-Hill, Inc.; 1974.

C83. Hoener, Roehder: Microprocessor-Controlled Switching Networks, 11 simpozijum Informatika 76; Bled; Oktobar 1976.

C84. Doutiller: Un Concentrateur d'Envoi de Messages a Microprocesseur, Electronique; 1er October 1977.

C85. - : The 8080A Diagram Intel Coudn't Run; Electronics; September 1, 1977.

C86. Boutiller: 8-by-8 Multiplier Dissipates 1 Watt; Electronics; August 4, 1977.

C87. Bansod, Kitai: Microprocessor Application in Walsh Furier Spectral Conversion; International Simposium on Electromagnetic Compatibility; Washington, D.C., U.S.A.; July 1976.

C88. Lind: Efficient Generation Techniques for Polynomial Codesets; Electronic Letters; October 1976.

C89. Allen: Special Purpose Computer Architecture for Speech Processing; EASCON 76 Record; Washington, D.C.; September 1976.

C90. Scott, Erich: Microprocessor-Based Correlator Reference System; 13th IEEE Computer Society International Conference; Washington, D.C.; September 1976.

C91. Bjerede: Algorithms for a Microprocessor-Based TACAN Signal Processor; Bicentennial National Aerospace Symposium on new Frontiers in Aerospace Navigation; Warminster, Pa. U.S.; April 1976.

C92. Belter, Williams, Bass: Microprocessor-Controlled Communications in an Advanced Instrument Landing System; Proceedings of the 1976 IEEE International Symposium on Circuits and Systems; München, Germany; April 1976.

C93. Runyon: Focus on Logic and UP ≠ Analysers, Electronic Design; Vol. 25; 1977.

C94. Brofferio, Soire: Electronic Telephone Switching System; III Euromicro Simposium; Amsterdam; October 1977.

C95. van Spronse, Bearzatto: Data Communication Protocol Processor; III Euromicro Simposium; Amsterdam; October 1977.

C96. MacMillan: A Method of Intermodule Communication in a Multimicrocomputer Network; III Euromicro Simposium; Amsterdam; October 1977.

C97. Pedenon, Monchaud, Lomaire: Microprogrammed Biprocessor System for Real Time Processing; III Euromicro Simposium, Amsterdam; October 1977.

C98. Siewiorek, Barbacci, Spencer: Modularity and Multiprocessor Structures - Some Open Problems in the Construction and Utilization of Mini- and Microprocessor Networks; Infotech. Internat.; Maidenhead., Berks., England; 1976.

C99. Fahrenschon: Mikroprozessoren zur Gebührenerfassung in Fernsprech-Nebenstellean-Agen; NTZ; Heft 9; 1977.

C100. Winter Rufnummern - Identifizierung mit Mikroprozessoren; NTZ; Heft 9; 1977.

C101. McLanghlin: GTD-120 Hotel/Motel Digital PABX; Autom. Electr. Tech.; October 1976.

C102. Bjerede: A Unified Signal Processor For TACAN Navigation Sets; Navigation; Summer 1976.

C103. Padgett, Cobb: Microprocessor-Controlled Spread-Spectrum Demodulator; EASCON 76 Record; Washington, D.C., U.S.A.; September 1976.

C104. Waggener: Designer's Guide to Digital Synchronization Circuits; EDN; August 1976.

C105. Bliss: Stopping the 8080; Electron. Ind.; November 1976.

C106. Smith: USART - A Universal MUP Interface for Serial Data Communications; EDN; September 1976.

C107. Microprocessor Opens New Possibilities for the Worldwide Telecommunication Network; NTZ; November 1976.

C108. Dogliotti, Marrei, Mengali: Design and Performance of an All-Digital Receiver for Multilevel AM-SSB Data Transmission; Proceeding of the 1976 IEEE International Symposium on Circuits and Systems; München, Germany, April 1976.

C109. Smith: Multifrequency Tone Detection by Zero-Interval Analysis; Electronic Letters; January 1977.

C110. Seymour, Catward: A Microprocessor Approach to Speech Recognition; Acustica; February 1977.

C111. Feldmah: Microprocessors in ECM Applications; Microwave, September 1976.

C112. Milutinović: Prijemnik za QDPSK modem na bazi MOS mikroprocesora; Simpozij o telekomunikacijah; Ljubljana, Yu.; Oktobar 1978.

C113. Conan, Oliver: Hardware Implementation of the Viterbi Decoding Algorithm for Convolutional Codes; Proceeding of the International Symposium on Mini and Micro Computers; Toronto, Canada; 8-11 Nov. 1976.

C114. Vender Hoy, Forney, Stearns: LSIs in Data Networks; Data Processing /GB/, April 1977.

C115. Mayr-Stein: Impact of Microprocessors on the Development Tasks in Communications Engineering; Nachrichtentechnik; June 1977.

C116. Pitroda, Fechalos, Stelman: Digital Systems Microprocessor Controlled; Telecommunications; January 1977.

C117. Fox: Digital Technology in Spectrum Analysers; Commun. Int. /GB/; March 1977.

C118. Runyon: Focus on Logic and MUP Analyzers; Electron. Des.; February, 1 1977.

C119. Higuchi, Saito, Kanomata: A Microprocessor-Based Digital Filter Programmed in a Block Diagram Language; IEEE Trans. on Ind. Electron. and Control Instrum.; Aug. 1977.

C120. Coit: Programmable Multiline Communications Processor Provides Front-End Flexibility; Comput. Des.; May 1977.

C121. Field, Titus, Rony, Larsen: A Software UART; Elektroniker /Switz./; Vol. 16, No.3; 1977.

C122. Sturmer: A Floppy Disc Interface for 8080 Systems; Electron. Eng.; August 1977.

C123. Young: Modern Message Switching Systems; Electron. Eng. /GB/; August 1977.

C124. Vivinn, Von Cavallar, Dodds: Micro-Processor Based System Design in Broadcast Engineering Applications; Int. Broadcast Eng. /GB/; Feb. 1977.

C125. Cooper, Copeland, Krambeck, Stanzione, Thomas: A CMOS Microprocessor for Telecommunications Applications; 1977 International Solid-State Circuits Conference; Philadelphia, Pa., USA; 16-18 February 1977.

C126. Fulghum: Communications Systems Control Utilizing Micro-Processor; Proceedings of the IEEE 1976 National Aerospace and Electronics Conference; Dayton, Ohio, USA; 18-20 May 1976.

C127. Tillotson, Jusko: Microprocessors: Voice Processor Application; Proceedings of the IEEE 1976 National Aerospace and Electronics Conference; Dayton, Ohio, USA; 18-20 May 1976.

C128. Winsborrow, Belcher, Cox: A General Purpose Communications Processor; EUROMICRO Newsletter, Vol. 3; 1977.

C129. Choquet, Nussbauer: Microcoded Modem Transmitters; IBM J. Res. Develop.; July 1974.

C130. Nordling, Walsh: Programming a Modem; Proceedings Nat. Tel. Conference; 1976.

C131. Tritton, Cullen: A Prototype Digital Private Telephone Exchange; Sci. and Technol. /GB/; Vol. 43, No.2; 1976

C132. Aranguren, Langseth: A Microprocessor Controlled Interface for Digital Satellite Systems; International Conference on Communications; Chicago, III., USA; 5 June 1977.

C133. Di Tria, Montagna, Perlino: Algorithms for Signal Treatment and Simulation for 3,200 bit/s Modem; Rapp. Tec./Italy/; Vol.5, No.2; April 1977.

C134. Gandini, Giandonato, Impallomeni: Voice Band Modem and DCE for New Data Network Integrated in Microprocessor Unit; Rapp. Tec. /Italy/; Vol.5, No.2; April 1977.

C135. Paus: Micro-processor Used in Computer Communication Equipment; Electro /Norway/, Vol.90, No. 14; 25 Aug. 1977.

C136. Sharpless, Penna, Turner: An Advanced Home Terminal for Interactive Data Communication; International Conference on Communications; Chicago, III., USA; 5 June 1977.

C137. Boddi: Speech Recognition for a Personal Computer System; Byte /USA/; July 1977.

C138. Allen, Grant: Digital Device Technology for Signal Processing; International Specialist Seminar on the Impact of new Technologies in Signal Processing; Aviemore, Scotland, Sep. 1976.

C139. Ferjančič - Štiglic: Realizacija in uporaba Walshovih funkcij v mikroračunalniku, ETAN, Yu.; Zadar; June 1978.

C140. Lah, Bernard: Problem programske opreme mikroračunalniškega teleinformacijskega sistema TI-30; ETAN, Yu; Zadar; Juni 1978.

C141. Jagodić: Vpliv mikroelektronskih tehnologija na načrtovanje telekomunikacijskih sistemov; ETAN, Yu.; Zadar; Juni 1978.

C142. Borojević, Lazić, Milošević, Temerinac: Primena mikroprocesora u elektronskoj realizaciji uredjaja za prikupljanje podataka o intenzitetu saobračaja u TF centrali; ETAN, Yu.; Zadar, Juni 1978.

C143. Kobolt: Nachrichtenmesstechnik mit Mikroprozessorsteuerung; Techn. mit PTT; Vol.55, 1977.

C144. Adolphs: NT 2000 Cointelephone Using Microprocessor Techniques Electrical Communication; Vol. 52; 1977.

C145. Matera: Data Converters Latch onto Microprocessors; Electronics; 1 Sept. 1977.

C146. Rahko, Tossavainen: A Microprocessor Controlled Integrated Digital Grid Switching System - IDGSS; International Conference on Distributed Computer Control; Birmingham, Eng.; Sept. 1977.

C147. Deal: TDMA Open-Loop Acquisition and Synchronization; Joint Automatic Control Conference; San Francisco, USA; June 1977.

C148. Allan: Logic Analysers; IEEE Spectrum; Aug. 1977.

C149. Fahrenschon: Microprocessors for Call Charge Registration in PABXs; Nachrichtentechn. Z.; Sept. 1977.

C150. Winter: Directory Number Identification with the aid of Microprocessors /in Telephone Switching Centres with Decentralized Control/; Nachrichtentech, Z.; Sept. 1977.

C151. Teixeira: Tactical Switchboard; International Conference on Communications; Chicago, USA; June 1977.

C152. Dice: Message Handling; Western Electric Eng. /USA/; July 1977.

C153. Fisher, Fitch, Kuhar: A Microprocessor Implementation for Measurement of Data Signal Distortion; International Conference on Communications; Chicago, USA; June 1977.

C154. Svetska: Seven Segment to BCD Converter for Calculator Chips, Electron. Eng. /GB/; Sept. 1977.

C155. Thurber, Masson: Recent Advances in Microprocessor Technology and their Impact on Interconnection Design in Computer Systems; International Conference on Communications; Chicago, USA; June 1977.

C156. Rattner /Intel Corp./: Microprocessor Architecture - where to go from here; 14th IEEE Computer Society International Conference; San Francisco, USA; Feb.-Mar. 1977.

C157. Pierce, Moore: Network Operating System Functions and Microprocessor Front-Ends; 14th IEEE Computer Society International Conference; San Francisco, USA; Feb./-Mar.1977.

C158. Higuchi, Saitto: Programmable Digital Filter Based on Microprocessor; Trans. of Inst. for Electron. and Commun. Eng. /Jpn./; June 1977.

C159. Huse: Magnetic-Bubble Memory for Microprocessor Systems; Elektronik /Germany/; Oct. 1977.

C160. Lesea, Urkumyan: Multiplexer System Reduces Cost of Terminal Interfacing; Computer Design; Aug. 1977.

C161. Bastiani, Lisee: Use of Microprocessor Interface for High Level Protocol Processing in Computer Systems; International Conference on Communications; Chicago, USA; June 1977.

C162. Eberle, Goodchild: Development of an Encryption System to Secure Data Transmission and Data Storage; International Conference on Communications; Chicago, USA; June 1977.

C163. - : The IMP-16 in Communication Applications; Compute /USA/, April 1977.

C164. Blume, Goodman, Philips, Sample: A Universal Peripheral Interface Chip for Microprocessor Systems; 14th IEEE Computer Society International Conference; San Francisco, USA; Feb.-Mar. 1977.

C165. Grimes: Distributed Processing Concepts Using Microprocessors; 14th IEEE Computer Society International Conference; San Francisco, USA; Feb.-Mar. 1977.

C166. Toong: MICROSTAR - A Microprocessor Controlled Distributed Minicomputer Network; 14th IEEE Computer Society International Conference, San Francisco, USA; Feb.-Mar. 1977.

C167. Bishop: A Multi-Processor Simulator for a Network of Cooperating Microprocessors; International Conference on Distributed Computer Control; Birmingham, Eng.; Sept. 1977.

C168. Blackwell: Modem 24 LSI; Seminar o prenosu podataka; Institut Mihailo Pupin, Beograd, Yu.; 1974.

C169. Queyssac /Motorola/: Communicating with Microprocessors /Modems/; Commun. Int. /GB/; July-Aug. 1976.

C170. Macarthur: Design of the Seaset - A Radar Altimeter; Simposium OCEANS 1976; Washington, D.C., USA; September 1976.

C171. Antanaitis, Lombaer, Pagel: Tie-Break and Allegiance Network; IBM Tech. Disclosure Bull. /USA/; Nov. 1976.

C172. Cheminaud: A Microprogrammed Distributed IOCS; Conference on Informatics; Paris, France; Sep. 1976.

C173. Belter, Williams, Bas: Microprocessor - Controlled Communications in an Advanced Instrument Landing System; IEEE Trans. Aerosp. and Electron. Syst.; Nov. 1976.

C174. Garen, Lazar: Microprocessors in Telecommunications; Telecommunications /USA/; April 1976.

C175. Gaon /Rockwell/: Hand Held Calculator Technology Applied to an Advanced Low Cost Omega Receiver; Conference on Medium Accuracy Low Cost Navigation; Sandefjord, Norway, Sep. 1975.

C176. Dupuy: Message Switching and a Specialised Microprocessor; Conference on Informatics; Paris, France; Sep. 1976.

C177. McAuliffe, Hagstrom: Microprocessor Applications in Communications Switching; National Telecommunications Conference; Dallas, Tex., USA; Nov.-Dec. 1976.

C178. Hammer: A Microprocessor-Based Bit Error Performance Monitor; National Telecommunications Conference; Dallas, Tex. USA; Nov.-Dec. 1976.

C179. Cobb, Osborne: Hardware Implementation of a Spread-Spectrum Modulator/Demodulator for the Space Shuttle Orbiter; National Telecommunications Conference; Dallas, Tex.; USA; Nov.-Dec. 1976.

C180. - : Intelligent Data Storage Device Functions as Programmable Communication System; Comput. Des. /USA/; Dec. 1976.

C181. Hellwig: The Control of Concentrators in a Data Network; Nachrichtentech. Electron. /Germany/; Vol.27, No.1; 1977.

C182. Elliot, Demange: The Microprocessor Technology Role in Suport of Telecommunications System Growth; National Telecommunications Conference; Dallas, Tex., USA; Nov.-Dec. 1976.

C183. Falk: Microprocessors Talk it up; IEEE Spectrum /USA/; Mar. 1977.

C184. Sherman, Verma: System Discription of a Programmable Multiple-Dataset; Nat. Telecommun. Conf.; New Orleans, LA; Dec. 1975.

C185. Salazar, Sherman, Verma, Werner: Implementation of Voiceband Modems on a Digital Signal Processor; Nat. Telecommun. Conf.; New Orleans, LA; Dec. 1975.

C186. Snijders, Verhoeckx, van Essen, van Gerwen: Digital Generation of Linearly Modulated Data Waveforms; IEEE Trans. on Commun.; Nov. 1975.

C187. - : Microprocessors - A Selection of Bits /Communication Applications/; Commun. Int. /GB/; July 1977.

C188. Winfield. MAC-8: A Microprocessor for Telecommunication Application; Western Electric Eng. /USA/; July 1977.

C189. Nash: Microprocessor Software Programms Bit-Rate Generator; EDN /USA/; 20 Aug. 1977.

C190. Hofstetter, Tierney, Wheeler: Microprocessor Realization of a Linear Predictive Vocoder; IEEE Trans. Acoust. Speech and Signal Process.; Oct. 1977.

C191. Gundlach: Fiber Optics, LSI Expand System Capacities /Telecommunications/; Electronics /USA/; 27. Oct. 1977.

C192. Coates: Digital Processing Techniques and Equipment; Conference on New Devices, Techniques and Systems in Radar; the Hague, Netherlands; June 1976.

C193. Waser, Peterson: Real-Time Processing Gains Ground with Fast Digital Multiplier; Electronics /USA/; 29 Sept. 1977.

C194. Gibson: Microprocessor Message Switching; Commun. and Broadcast /GB/; Spring 1977.

C195. - : Signetics Develops an $I^2L$ Codec; Electronics; 27 April 1978.

C196. Travis: Communications Chip Interfaces with most Microprocessors; Electronics; 16 March 1978.

C197. Gandini, Giandonato, Impallomeni: Voice Band Modem and DCE for New Data Network Integrated in Microprocessor Unit; Rapporti Technici /Italy/; April 1977

C198. Melvin: Microcomputer Applications in Telephony; Proc. of the IEEE; Feb. 1978.

C199. Stanzione: Microprocessors in Telecommunication Systems; Proc. of the IEEE; Feb. 1977.

C200. Pitroda, Fechalos, Stehman: The Microprocessor Controlled 580 Digital Switching System; Int. Switching Sym.; Kyoto, Japan; Oct. 1976.

C201. Stiles: The Impact of Microprocessors on Communication Equipment; Conf. Proc. ELECTRO 76; May 1976.

C202. Cooper, Copeland, Krambeck, Stanzione, Thomas: A CMOS Microprocessor for Telecommunication Applications; Dig. of Tech. Papers 1977 IEEE Int. Solid-State Circuits Conf.; Feb. 1977.

C203. Holsinger: Where are Modems Going?; Telecommunications; June 1977.

KSIĄŻKI O MIKROPROCESORACH

D1. Osborne: An Introduction to Microcomputers - Basic Concepts; Sybex; 1976.

D2. Osborne: An Introduction to Microcomputers - Some Real Products; Sybex; 1976.

D3. Osborne: 8080 Programming for Logic Design; Sybex; 1976.

D4. Guzeman: Microcomputer Applications Handbook; Iasis, Inc.; 1976.

D5. Souček: Microprocessors and Microcomputers; John Wiley and Sons, Inc.; 1976.

D6. McGlynn: Microprocessors; John Wiley and Sons, Inc.; 1976.

D7. Martinez: A Look of Trends in Microprocessor/Microcomputer Software Systems; Computer Design June 1975.

D8. Peatman: Microcomputer-Based Design; McGraw-Hill. Inc.; 1977.

D9. Barna, Porat: Introduction to Microcomputers & Microprocessors; Wiley-Interscience; 1976.

D10. Osborne: Einführung in die Microcomputer Technik; Sybex; 1977.

D11. Osborne: 8080A and 8085 Assembly Language Programming; Sybex; 1978.

D12. Hilburn, Julich: Microcomputers/Microprocessors; Prentice-Hall, Englewood Cliffs; 1976.

D13. Libion: De Microprocesseur au Micro-Ordinateur; La Bibliotheque des Editions Radio; Paris; 1976.

D14. Alan Salisbury: Microprogrammable Computer Architecture; Elsever Computer Science Library; New York, Oxford, Amsterdam; 1976.

D15. Katzan, Jr.: Microprogramming Primer; McGraw-Hill; 1977.

D16. Sippl: Microcomputer Handbook; Mason/Charter Publishers, Inc.; 1977.

D17. Klingman: Microprocessor Systems Design, Prentice-Hall, Inc.; 1977.

D18. Healey: Minicomputers and Microprocessors; Hodder and Stoughton Educational; 1976.

D19. Diehl: Mikroprozessoren und Mikrocomputer; Vogel-Verlag; 1977.

D20. Zaks: Microprocessors: From Chips to Systems; Sybex; 1977.

D21. Lesea, Zaks: Microprocessor Interfacing Techniques; Sybex, 1977.

D22. Leahy: Microprocessor Architecture and Programming; John Wiley and Sons Inc.; 1977

D23. Chu: Computer Organization and Microprogramming; Prentice Hall, Inc.; 1972.

D24. - : Microprocessor Encyclopedia; Sybex; 1977.

D25. Leo: Microprocessor Software Laboratory; Sybex; 1977.

D26. Šuhel: Mikroračunalnik; zal. Fakulteta za elektrotehniko Univerze v Ljubljani; Ljubljana 1977; Učbenik dopolnilnega izobraževanja ob delu.

D27. Souček: Mikroračunala; Tehnička knjiga; Zagreb; 1978.

D28. Boulaye: Microprogramming; Dunod Inc.; 1971.

D29. Husson: Microprogramming; Prentice-Hall, Inc.; 1970.

D30. Souček, Carlson: Computers in Neurobiology and Behavior; Wiley-Interscience; 1976.

D31. Bibbero: Microprocessors in Instruments and Control, John Wiley and Sons, Inc.; 1977.

D32. Rodnay Zaks: An Introduction to Personal and Business Computing; Sybex; 1978.

D33. Rodnay Zaks: Microprogrammed APL Implementation; Sybex; 1978.

D34. Robert Findley: 8080 Cookbook; Sybex; 1978.

D35. Robert Findley: 6800 Cookbook; Sybex; 1978.

D36. Nat Wadsworth: Understanding Microcomputers; Sybex; 1978.

D37. Scelbi: 8080 Standard Editor; Sybex; 1978.

D38. Raymond Edwards: 8080 Standard Assembler; Sybex; 1978.

D39. Scelbi: 8080 Standard Monitor; Sybex; 1978.

D40. Adam Osborne: An Introduction to Microcomputers, Volume 0; Sybex; 1978.

D41. Adam Osborne: 6800 Programming for Logic Design; Sybex; 1976.

D42. Adam Osborne: Z80 Programming for Logic Design; Sybex; 1978.

D43. Rodnay Zaks: Les Microprocesseurs; Sybex; 1978.

D44. Rodnay Zaks: Techniques d'Interface aux Microprocesseurs; Sybex; 1978.

D45. Adam Osborne: Introduction aux Microordinateurs.

D46. Adam Osborne: Accounts Payable and Accounts Receivable; Sybex; 1978.

D47. Adam Osborne: General Ledger; Sybex; 1978.

D48. Adam Osborne: 6800 Assembly Language Programming; Sybex; 1978.

D49. Larsen Titus: Bugobook Series /I, II, IIA, III, V, VI/; Sybex; 1978.

D50. Rodnay Zaks: Self-Study Courses with Cassettes; Sybex; 1977.

D51. Rodnay Zaks: Programmation du 6502; Sybex; 1978.

D52. P.R. Rony, D.G. Larsen and J.A.Titus: The 8080A bugbook: Microcomputer Interfacing and programming; Howard W.Sams, Indiana, USA; 1977

D53. Peter Šuhel, Jernej Virant: Mikroračunalnik; Univerzum, Ljubljana, Yu.: 1978.

D54. Guthikonda V.Rao: Microprocessors and Microcomputer Systems; Van Nostrand Reinhold Company, New York; 1978.

D55. Alvin W. Moore at al. /Motorola Inc. Engineering Staff/: Microprocessor Application Manual; Mc Graw-Hill Book Company, New York; 1975.

D56. Pelka: Cos'e un microprocessore?; Franco Murzio & c. editore; Padova; 1978.

D57. Herbert Bernstein: Hochintegrierte Digitalschaltungen und Mikroprocessoren; Richard Pflaum Verlag KG, München; 1978.

D58. Horst Polka: Von der Schaltalgebra zum Mikroprocessor; Franzis; 1977.

D59. Granino Korn: Microprocessors & Small Digital Computer Systems for Engineers Scientists; McGraw-Hill Book Company; 1977.

D60. D.Aspinall, L. Daglas: Introduction to Microprocessors; Academic Pres, Inc.; 1977.

D61. P.Blomeyer-H.Bartenstein: Micro Computer Technik; ing. W.Hofacker Verlag GMBH, Holzkirchen; 1976.

D62. H.Bartenstein: Mikroprozessor; Grundlagen Eigenschaften und Aufbau /I; II/; ing. W.Hofacker Verlag GMBH, Holzkirchen; 1978.

D63. C.Lorenz: Mikrocomputer Programmierung; Software Handbuch; ing. W.Hofacker Verlag GMBH, Holzkirchen; 1978.

D64. M.W. Mc. Murran: Programming Microprocessors; Tab Books, Inc.; 1977.

D65. Charles Adams: A Beginner's Guide to Computers & Microprocessors - with projects; Tab Books, Inc., 1978.

D66. Neill Graham: Microprocessor Programming for Computer Hobbyists; Tab Books, Inc.; 1978.

D67. Brice Ward: Microprocessor/Microprogramming Handbook; Tab Books, Inc.; 1975.

D68. -: MOS Microcomputers - Programmier Handbuch; MOS Technology, Inc.; 1977.

D69. Nat Wadsworth: Understanding Microcomputers and Small Computer Systems: Scelbi Computer Consulting, Inc.; 1977.

D70. Freiberger and Chew: Consumer's Guide to Personal Computing and Microcomputers; Scelbi Computer Consulting, Inc.; 1978.

D71. Leventhal: The 6800 Microprocessor: A Self-Study Course with Applications; Scelbi Computer Consulting, Inc.; 1978.

D72. C.Lorenz: Microprocessor Software Handbuch MSH; ing. Hofacker Verlag GMBH, Holzkirchen; 1978.

D73. C.Lorentz: Mikrocomputer Datenbuch; ing. Hofacker Verlag GMBH.

D74. C.Lorenz: Aktivtraining-Mikrocomputer /Sammelordner/; ing. Hofacker Verlag GMBH, Holzkirchen; 1978.

D75. C.Lorentz: 57 Programme in BASIC; ing. Hofacker Verlag GMBH, Holzkirchen; 1978.

D76. J.Hatzenbicher: Mikrocomputer Programmierbeispiele für 2650; ing. Hofacker Verlag GMBH, Holzkirchen; 1978.

D77. K.L. Bowels: Microcomputer - problem solving using PASCAL; Springer-Verlag, Berlin, FDR; 1977.

D78. - : Einführung in die Mikroprozessortechnik; Texas Instruments; Freising; 1978.

D79. - : Software Design for Microprocessors; Texas Instruments; Freising; 1978.

D80. - : The Bipolar Microcomputer Components Data Book; Texas Instruments; Freising; 1978.

D81. Weller: Microprocessors; Prentice-Hall, Inc.; 1978.

D82. William Barden, Jr.: How to Program Microcomputers; Howard W. Sams & Co. Inc., 1977.

D83. Andrew Veronis: Microprocessors: Design & Applications; Reston Publishing Company, Inc. A Prentice-Hall Company, Reston, Virginia, 1978.

ZBIORY ARTYKUŁÓW O MIKROPROCESORACH

E1. Lee /editor/: Microprocessors; Proceedings IEEE; June 1976.

E2. Queyssac /editor/: Understanding Microprocessors; Motorola, Inc.; 1976.

E3. Torrero /editor/: Microprocessors: New Directions for Designers; Electronic Design; 1976.

E4. Altman /editor/: Microprocessors; Electronics; 1976.

E5. Hartenstein, Zaks /editor/: Euromicro-Workshop; I Euromicro Simposium; Nice, Juni 1975; Microarchitecture of Computer Systems; North-Holland.

E6. Lin /editor/: Microprocessors: Fundamentals and Applications; IEEE Pres; 1977.

E7. Nicoud, Wilmink, Zaks /editori/: The Third Euromicro Simposium; Amsterdam, October 1977; North-Holland.

E8. Zaks, Sami, Wilmink /editori/: Second Euromicro Simposium; Venice; October 1976; North-Holland, Inc.; 1976.

E9. - : Electronics - A Special Issue on Microprocessors; April 19, 1976.

E10. - : Proceedings of the IEEE; Aug. 1977.

E11. - : Proceedings of the IEEE; Feb. 1978.

E12. Altman, Scrupski /editori/: Applying Microprocessors; McGraw-Hill, Inc.; 1976.

E13. Nat Wadsworth, Carl Helmers /editori/; The Scelbi/Byte Prymer; Scelby Computer Consulting, Inc.; 1978.

E14. Samuel Lee /editor/: Microcomputer Design and Applications; Academic Press, Inc.; 1977.

E15. Michael S. Elphick /editor/: Microprocessor Basics; selected from Electronic Design; Hayden Book Company, Inc.; Rochelle Park, New Jersy, USA; 1977.

E16. C.Whitby-Strevens /editor/: Microcomputer Systems /I, II/; Infotech International Limited, England; 1978.

E17. C.Whitby - Strevens /editor/: Microprocessors /I, II/; Infotech International Limited, England; 1978.

KATALOGI TECHNICZNE

F1. Intel: 8080 Assembly Language Programming Manual; 1975.

F2. Intel: 8080 Microcomputer Systems Manual; 1975.

F3. Intel: 8085 Microcomputer System Preliminary Handbook; Oct. 1977.

F4. Intel: Polysilicon Fuse Bipolar PROMs Reliability Report; Oct. 1975.

F5. Intel: 2708 Erasable PROM Reliability Report; Sep. 1976.

F6. Burr & Brown: Analog Input Microperipheral; 1976.

F7. Burr & Brown: Analog Output Microperipheral; 1976.

F8. General Semiconductor Industries, Inc.: Transzorb Microprocessor Transient Protector; 1976.

F9. Philips: MOS Circuits Reliability and Protection Report;

F10. Texas Instruments: Integrated Circuits Catalog 1974.

F11. National: Integrated Circuits Catalog 1974.

F12. Motorola: Integrated Circuits Catalog 1974.

F13. Motorola: Microprocessor Applications Manual; 1975.

KATALOGI REKLAMOWE

G1. Intellec: Microcomputer Development System; 1976.
G2. Intel: PL/M; 1976.
G3. Motorola: Microcomputer Bypolar Interface; 1976.
G4. Motorola: New Microcomputer System Components; 1975.
G5. Intel: SBC-20; 1977.
G6. Intel: Data Catalog; 1978.
G7. COSSOR: General Purpose Multi-Tone /Radio / Modem; CGT 1070.
G8. Sylvania: Mark IV Stored program HF Modem;
G9. Sylvania: Multi-Rate Microprocessor Modem
G10. Advanced Products Division: Digital Data /Radio / Modem; MX-513.
G11. Paradyne: Micro-processor 48; CCITTV.27 bis/ter Modem.

REFERATY

H1. Sippl, Kidd: Microcomputer Dictionary and Guide; Matrix Publishers, Inc.; 1976.
H2. Zaks: 10 Language Microdictionary; Sybex; 1976.
H3. Sippl, Sippl: Computer Dictionary and Handbook; Howard W. Sams and Co., Inc./The Dobbs-Merrill Co., Inc.
H4. - : Microprocessor Lexicon; Sybex; 1978.
H5. C. Lorenz: Lexikon und Wörterbuch für Elektronik und Mikroprozessortechnik; Ing. W.Hofacker GMBH; 1978.
H6. - : Loxique Microprocesseurs; Sybex; 1978.
H7. - : Dictionnaire international des microprocesseurs; Sybex; 1978.

MATERIAŁY SEMINARYJNE

I1. Zaks: Sybex
I2. Collins: ICS
I3. Rijeka: ETAN - Opći seminar o mikroprocesorskim sistemima i upotrebi mikroprocesora; Rijeka, Siječanj 1978.
I4. Ljubljana: Isemoc
I5. C. Whitby - Strevens; Infotech.

CZASOPISMA SPECJALISTYCZNE

J1. Microprocessors
J2. Euromicro Journal
J3. Byte
J4. Mini-Micro Systems
J5. Micomp /niem./
J6. Interface Age
J7. Elcomp
J8. Elcomp /niem./
J9. Chip /niem./
J10. Microprocessors and Microsystems
J11. Compute

POZOSTAŁE ŹRÓDŁA

K1. Źródła mówione /dyskusje panelowe, wykłady .../
K2. Źródła własne oraz wnioski
K3. Oferty producentów
K4. Intel OEM Price List; September 6, 1977.
K5. Iskra - Indok informacije; 1977; 1978.

DODATEK

C204. Wahlstrom: Telegraph, Telex, Data - Three Rapidly Changing Services; TELE /Sweden/; Vol.83, No.2-3; 1977.

C205. Abel, Jacobs, Gilhousen: Built-in Test for Microprocessor Based UHF Satellite Modem; Proc. of the IEEE 1977 National Aerospace and Electronics Conference; Daytone, Ohio, USA; 17-19 May 1977.

C206. Gabriele: Multiprocessing Can Marry a Radar That Detects Only Moving Targets to a Large Air-Traffic Control Computer Inexpensively and with High Reliability; Electron. Des.; 2 Aug. 1977.

C207. Terrell, Davenport, Ross: The Emerging World of Digital Filters; Electron. Eng. /GB/; Oct. 1977.

C208. Snijders: Microprocessor Implementation of Data Modems; Tijdschr. Ned. Electron. - and Radiogenoot. /Netherlands/; Vol. 42, No.4; 1977.

C209. Dixon, Silverman, Winograd: Small-Vocabulary Discrette-Word Recognizer Using a Microprocessor and Winograd-Fourier Transform; IBM Tech. Disclosure Bull.; Vol.19, No.1; 1977.

C210. Gordon: Microprocessor in Communications; Commun. Int. /GB/; Oct. 1977.

C211. Gebler: Microprocessor Based Waveform Generation; Electron. Eng. /GB/; Nov. 1977.

C212. - : Mikroprozessoren - Neve Bauelemente der Nachrichten-technik; Ingen der DBP; Vol.26; 1977.

C213. - : Microprocessor Application for Numerical ECG Encoding and Transmission; Proceedings IEEE; Vol.65.; 1977.

C214. Cimander, Winter: Mikroprozessoren in der Fernsprechvermittlungs-technik; Fernmeldetechnik; Vol.18; 1978.

C215. Smith, Park: Pathfinderoan Experimental Telephone Exchange Using-Stored Program Control; POEEJ; Vol.70; 1977.

C216. E.Harris, T.Saliga, D.Walsh: An All Digital 9,600 bps LSI Modem; Proceedings dell NTC 74; December 1974.

C217. K.Murano, S.Unagami, T.Tsunda: LSI Processor for Digital Processing and its Application to 4,800 bit/s Modem; IEEE Trans. on Comm.; Vol. COM-26, May 1978.

C218. K.Watanabe, K.Inone, Y.Sato: A 4800 bit/s Microprocessor Data Modem; IEEE Trans. on Comm.; Vol COM-26, May 1978.

C219. L.Bella, S.Benedotto, D.Del Corso, M.Palmeri: Impiego di microprocessori nei modem ad alta velocita:

Realizzarione di un trasmettitore a 9600 b/s Riporto di Instituto di Elettronica e Telecomunicazioni, Politecnica di Torino; C. so Duca Abruzzi 24-10129, Torino, Italia.

C220. Naffah.: N. Use of Microprocessors for Interconnection of Computers in an information Network;
Convention Informatique 1977 /Information Convention 1977/:, France 20-23 Sept.1977.

C221. Blats, J.R., Marquis, L., Houle, J.L. Hamza, M.H.: L. Canada A Microprocessor Application in Network Automation;
Proceedings of the International Symposium on Mini and Micro Computers: Montreal, Canada 11-18 Nov. 1977.

C222. Shum, M.: Modem Design Using Microprocessors;
Electron. Eng. /GB/, Vol.50, No.603, P.53, 55 /March 1978/.

C223. Smith, K.W., Carter, C.R. Hamza, M.H.: Engng., "Microprocessor Controlled Synchroniser for an SDMA/SS-IDMA Communications Satellite Earth Station",
Proceedings of the International Symposium on Mini and Micro Computers:, Montreal, Canada 11-18 Nov. 1977.

C224. Schmidt, J.E.: Applications of Microprocessors in Telephone Exchanges; Teloktronikk /Norway/, No 4, P.340-6 /1977/.

C225. Montemurro, R.: Distributed Microprocessor Stored Program Control in a Switching Network; EUROCON * 77 Proceedings on Communications:, Venice, Italy 3-7 May 1977.

C226. Williams, P.E.: An Optimum Local Network Derived from an Existing Telephone System;
The International Symposium on Subscriber Loops and Services: Atlanta, Ga, USA 20-24 March 1978.

C227. Anderson, R.V., Rotton, P.J., Wurst, W.: TCS-3 Central Office and DM32s Subscriber Carrier Integration the International Symposium on Subscriber Loops and Services: Atlanta, Ga, USA 20-24, March 1978.

C228. Microprocessor Controls Receiver for Communications Systems; Nachr. Elektron. /Germany/, Vol. 32, No. 2, P.65-6 /Feb. 1978/.

C229. Holmes, J.N., Judd, M.W., Walesby, D.H.: A High-Quality All-Digital Sound Spectrograph Developed for Speech Signal Analysis ;
Proceedings of the 1978 IEEE International Conference on Acoustics, Speech and Signal Processing;, Tulsa, Ok, USA 10-12 April 1978.

C230. Bourgonjon, L.R.: Developments in Commercial Radio Receivers; Ingenieur /Netherlands/, Vol. 90, No 19, P.397-9 /11 May 1978/.

C231. Gillespie, C.J.: MUP-Based Data Communications Involves More Than Mere I/O; EDN /USA/, Vol. 23, No.6, P.87-90 /20 March 1978/.

C232. -: Microprocessors Decentralize Datacom Networks; Digital Des. /USA/, Vol.8, No.3, P.42, 44, 50, 54-5 /May 1978/.

C233. Neumann, A.: Communications Measuring Equipment with Microprocessors; Eletron. Anz. /Germany/, Vol.10, No.2, /Feb. 1978/.

C234. Stroh, R.W.: An Experimental Microprocessor-Implemented 4800 bit/s Limited Distance Voice Band PSK Modem; IEEE Trans. Commun. /USA/, Vol. COM-26, No.5, P.507-12 /May 1978/.

C235. Aranguron, W.L., Langseth, R.E.: A Microprocessor Controlled Digital Interface for Digital, Satellite, Systems, IEEE Trans. Commun. /USA/, Vol.COM-26, No.5, P.631-7 /May 1978/.

C236. Dubnowski, J.J.: A Microprocessor LOG PCM/ADPCM, Code, Converter /Speech Processing/; IEEE Trans. Commun. /USA/, Vol. COM-26, No.5, P.660-4 /May 1978/.

C237. Austin, G.E.: Computer Controlled Solid State Audio Switching, System; Twenty-Eighth IEEE Vehicular Technology Conference, Denver, Co, USA 22-2 4 March 1978.

C238. Kotisaari, M., Kurru, K., Warsta, M.: A Real Time Operating System for Microprocessors and its Application into the DX 200 Statistical Unit;
Proceedings of the Third International Conference on Software Engineering for Telecommunication Switching, Systems, Helsinki, Finland 27-29, June 1978.

C239. -: Mikroprozessor Erfasst Telephonges´raechsdaten; Techn.Rundsch.Vol.70 Leto 78. St.32.

C240. Kral W.A.: Die Mikroprozessoren Sind Da - Auch Fuer Die Nachrichtentechnik; Nachr. Elektronik Vol. 32 Leto 78 St. 4.

C241. White, C.E.: Bits of Voice; Telecommunications /USA/, Vol.12, No.4, P.46, 48 /April 1978/.

C242. Bram, M.: Adapting the M6800 Processor for Automatic Telephone, Dialing; Electronics/USA/, Vol.51, No.14, P.128-9 /6 July 1978/.

C243. Anderson, E.W., Newhall, E.E., Venetsanopoulos, A.N.: A Microprocessor-Based Controller for a Loop Switching System; 1978 International Communications:, Toronto, Canada 4-7 June 1978.

C244. Callow, M.S., Crist, S.C., Meyer, R.A.: Microprocessor Utilization in Performance Monitoring of Communication, Networks; International Conference on Communications:, Toronto, Canada 4-7 June 1978.

C245. Cesaratto, C., Wood, R.G.: DNS-200 Control Structure; 1978 International Conference on Communications:, Toronto, Canada 4-7 June 1978.

C246. Smith, K.W., Carter, C.R.: Synchronizing to the Switched Satellite Using a Microprocessor Based System; 1978 International Conference on Communications:, Toronto, Canada 4-7 June 1978.

C247. Yamasaki, D.T., Morganstein, S.J.: Functional Partitioning of Telephonic Activities in a Distributed Load Sharing Environment 1978 International Conference on Communications:, Toronto, Canada 4-7 June 1978.

C248. Fenyves, A.: Application of Commercial Microprocessors for Implementing Trunk Signaling Adapters; 4-7 June 1978.

C249. Belt, R.A., Keele, R.V., Murray, G.G.: Digital TV Microprocessor System; NTC 77 Conference Record, Los Angeles, Ca, USA 5-7 Dec. 1977.

C250. Kobori, T., Nishiyama, T., Fukushima, M., Suzuki, T.
Wide-Area Centralized Supervision Systems Utilizing Telephone Lines; Natl. Tech.Rep. /Japan/., Vol.23, No.5, P.726-34 /Dec. 1977/.

C251. Young O.H., LIU, M.T.: Interface Design for Distributed Control Loop Networks; NTC 77 Conference Record: Los Angeles, Ca, USA 5-7 Dec. 1977.

C252. Puroy, T.: Computer Peripherals for Vehicles; Can. Electron. Eng. /Canada/, Vol.21, No.12, P.43-4 /Dec. 1977/.

C253. Fox, S.: Microwave Spectrum Analysis; New Electron. /GB/, Vol.11, No.3, P.30, 35 /7 Feb. 1978/.

C254. Sekamoto, M., Kinoshita, K.: Mobile Unit for 800 MHz Band Land Mobile Telephone System; Rev. Electr. Commun. Lab. /Japan/, Vol. 25, No.11-12, P.1231-44 /Nov.-Dec. 1977/.

C255. Jonckheer, F.: A Family of Digital PABXs; International Conference on Private Electronis Switching System:, London, England 10-12, April 1978.

C256. Travis, S.: Communications Chip Interfaces with Most Microprocessors; Electronics /USA/, Vol.51, No.6, P.123-8 /16 March 1976/.

C257. Shrivastava, A.K., Banerjee, D., Srinivasan, S.D., Ramanadhan, P.: An Interface for Using, Hindustan Teleprinter With Microprocessors; J.Inst.Electron. and Telecommun. Eng. /India/, Vol.23, No.10, P.607-10 /Dec. 1977/.

C258. Freund, W.R.: Data Transmission and Processing with Data Multiplexer RUO04/PUO40; News Rohde and Schwarz /Germany/, Vol.17, No.79, P.21, 24-5 /1977/.

C259. Horvath, S.Jr.: A Simple Adaptive Recursive Equalizer for High Speed Data Transmission, Mitt. Agen /Switzerland/, No. 24, P.31-8 /Dec. 1977/.

C260. Elmer, C.: The Mitel SX-200 Superswitch; Can. Electron. Eng. /Canada/, Vol.21, No.12, P.17-18 /DEC. 1977/.

C261. Rahko, K., Leppanen, R., Tossavainen, M.: A Small Microprocessor Controlled Switching System with Electronic Crosspoints and PCM-Switching Network; International Conference on Private Electronic Switching Systems:, London, England 10-12, April 1978.

C262. Barr, A.J., Brown, J.A.: An Application of Microprocessors to a Private Automatic Branch Exchange; International Conference on Private Electronic Switching Systems:, London, England 10-12 April 1978.

C263. Peyser, A., Hills, M.I.: TELEMAX-A Traffic Controller for PABX Networks; International Conference on Private Electronic Switching Systems: London, England 10-12 April 1978.

C264. Chen, F.F.: Development of a Microprocessor Controlled EPABX: International Conference on Private Electronic Switching Systems:, London, England 10-12 April 1978.

C265. -: Mikroprocessoren - Neue Bauelemente der Nachrichtentechnik; Ingen der DBP, Vol.26 Leto 77 St.3.

C266. Cimander W. Winter W.: Mikroprozessoren in der Fernsprechvermittlungstechnik; Fernmeldetechnik, Vol.18, Leto 78, St.1.

C267. Smith C.S.A., Park I.D.C.: Patheinderoan Experimental Telephone Exchange Using Stored - Program Control; POPEJ, Vol.70, Leto 77 St.2.

C268. Stimec B., Matosic M., Jagodić M.: Nekateri Nacini Realizacije Digitalnega Komunikacijskega Sistema s Posebnim Poudarkom na uporabi mikroprocesorja; ETAN 77, Banja Luka; Jugoslavija.

C269. -: Digital Techniques in the Design of Radio Equipment; Commun. Inter. Vol.5, Leto 78 St.4.

C270. Goodman, R.M.F., Green, A.D.: Microprocessor-Controlled Soft-Decision Decoding of ERROR-Correcting Block Codes;
Digital Processing of Signals in Communications:, Loughborough, England 5-9 Sept. 1977.

C271. Boutiller, R.: A Microprocessor Concentrator for Message Switching; Electron. and. Appl. Ind. /France/, No.241, P.43-6 /1 Oct. 1977/.

C272. Tanaka, Y., Vagasawa, K., Ohzond, Y., Kikuta, M.: New Bank Data Terminal System; Jpn. Telecommun. Rev. /Japan/, Vol. 20, No.1 P.57-52 /Jan. 1978/.

C273. Das, S.K., Silverman, H.F., Tappert, C.C.: A Microprocessor Controlled Speech Communication Link to a Time-Shared Computer; Digital Processing of Signals in Communications:, Loughborough, England 6-9 Sept. 1977.

C274. Shaw, J.: The Growth of Microprocessors in Data Communications; Can. Electron. Eng. /Canada/, Vol.22, No.1. P.33-4 /Jan. 1978/.

C275. Saettone, R. Danthine, A.: MITS Microprocessor Implementation of a Transport Station; Proceedings of the Computer Network Protocols Symposium: Liege, Belgium 13-15 Feb. 1978.

C276. Naffah, N. Danthine, A.: Tipac: An Approach to the Intelligent Terminal Interconnection to Packet Switched Networks; Proceedings of the Computer Network Protocols Symposium;, Liege, Belgium 13-15 Feb. 1978.

C277. Stanziono, D.C.: Microprocessors in Telecommunication Systems Proc. IEEE /USA/, Vol.66, No.2, P.192-9 /Feb. 1978/.

C278. Wang, R.T.: Programming a Microcomputer for a D-A Conversion; Electronics /USA/, Vol.50, N° 25, P.101-2 /8 Dec. 1977/.

C279. Lindemulder, C.R., Nahabedian, C.E., Trimble, D.C.: Horizon Communication System Innovation in System Design and Development; Bell Lab. Rec. /USA/, Vol.55, No.10, P.276--81 /Nov. 1977/.

C280. Stier L.: Cose Effective Switching System Design; Telecommunications /USA/, Vol.11, No.8, P.29-30, 34 /Aug. 1977/.

C281. Newport, C.B. Kaul, P.: Communications Processors for TELENET'S Third Generation Packet Switching Net work; EASON-77:, Arlington, Va, USA 26-28 Sept. 1977.

C282. Libby P.: New Concepts in Data Communications Telecommunications /USA/, Vol. 11, No.7, P.25/1-28/6 /July 1977/.

C283. Stehman, C.J.: Distributed Control of the 580 Digital Switching System; EASCON-77, Arlington, Va, USA 26-28 Sept. 1977.

C284. -: Programmed Microcomputer Controls Colour Television Receiver, Funk-Tech. /Germany/, Vol. 32. No.16, P.265-7 /Aug. 1977/.

C285. Fukuzumi, K., Nakatani, S., Kokan, J.: Design of Supervisory System for Land Mobile Radio Telephone System; Electr. Commun. Lab. Tech. J. /Japan/, Vol.26, No.7. P.1889-1919 /1977/.

C286. Dickopp, G.: Microprocessor Controls Station Selection /Radio Receivers/; Radio Mentor Electron. /Germany/, Vol.43, No.11, P.480-1 /Nov. 1977/.

C287. Marazas, G.: Microprocessors Need the Call to Supervise I/O; Electronics /USA/, Vol.51, No.3, P.104-8 /2 Feb. 1978/.

C288. Fronheiser, K.: Implementing a Microprocessor System; Electron. Equip. News /GB/. /July-Aug. 1977/.

C289. Powers, I.E.: Remote Data Terminals for Military Use; Syst. Technol. /GB/, No.26, P.22.7 /June 1977/.

C290. Okada K., Kajinara, M.: Functionally Distributed Telephone Exchange Control System Using Microprocessors; Euromicro Newsl. /France/, Vol.3, No.4, P.79-82 /Oct. 1977/.

C291. Allen, J. Steingart R.J.: A Special Purpose Processor for Speech Synthesis; 1977 Electro Conference Record: New York, USA 19-20 April 1977.

C292. Beauchamp J.N.: Microprocessor Decoding of Convolutional Codes; Proceedings of the IEEE 1977 National Aerospace and Electronics Conference, N AECON '77, Dayton, Ohid USA 17-19 May 1977.

C293. Simpson, R.C., Peterson, J.B.: Microprocessor Realization of the Bus Interface Unit for a Distributed Avionic Computer System
Proceedings of the IEEE 1977 National Aerospace and Electronics Conference, NAECON '77:, Dayton, Ohio, USA 17-19 May 1977.

C294. Franke, H.: Microprocessor System for the Control of Indicating Boards; Energie /Germany/, Vol.29, No.12 P.405-7 /Dec. 1977/.

C295. Holsinger, J.L.: Where are Modems Going; IEEE Commun. Soc. Mag. /USA/, Vol.15, No.5, P.3-5, 10 /Sept. 1977/.

C296. Baum, W.: Colour-TV Set with Microprocessor Control; Funkschau /Germany/, Vol.49, No.17, P.763-8 /12 Aug. 1977/.

C297. -: In Spite of Changed Speed: The Same Tone Pitch /Speech Reproduction/. Funkschau /Germany/, Vol.49, No 18, P.847-51 /26 Aug. 1977/.

C298. Layec, H., L'Hostis, J.C. Nicoud, J., Wilmink, J.Zaks, R.: Microprocessors and Packet Switching Networks; Microprocessing and Microprogramming: Amsterdam, Netherlands 3-6 Oct. 1977.

C299. Shearin, E.A., King. L.L.: Microprocessor Implementation of CCITT Recommendation X.25 /Levels 1 and 2 / Protocol: Proceedings of Computer Networking Symposium. Gaithersburg, Md.USA 15 Dec. 1977.

C300. Deforeii, C.: A Double Microprocessor for Message Switching; Electron. and Appl. Ind. /France/, No.240, P.93-5 /15 Sept. 1977/.

C301. Brofferio, S., Scire, G.Nicoud, J.Wilmink, J.Zaks, R.: Electronic Telephone Switching System: A Distributed Control Approach; Microprocessing and Microprogramming:, Amsterdam, Netherlands 3-6, Oct. 1977. /North-Holland, 1977/.

C302. Jonson W., Engebretsen J.E.: Microcomputers as Modular Data Network Compononts; submitted for EUROMICRO '79; Göteborg, Sweden; August 28-30, 1979.

C303. Milutinović V.: Detekcija na bazi multiple opservacije sa ponderisanom većinskom logikom; rad prijavljen za ETAN '79; Maribor, Yugoslavia; Juni 1979.

C304. Stuttard, E.B.: Application of Large Scale Integration to Data Modems; Racal-Milgo Report; Berkshire, England; 1975.

C305. E.D.Gibson: Application of Advanced Microelectronics to Modems /Rockwell International/; 1976, Int. Conf. on Com. /ICC 76/; June 14-16; Philadelphia, Pa, U.S.A.

C306. Frizzo R.A.: Some Processor Techniques for Telephone Subscribers & Call Metering; Australian Telecomm. Res. Vol. 12 Leto 78 str.1.

C307. Bishop A.: Microprocessors in Telecommunications; Commun. Inter. Vol.5 Leto 78 str.7.

C308. Spetcu S.G., Doyle J.: A Low-Cost Real-Time Service Digital Signal Processor; IEEE Trans. Commun. Vol. 26, Leto 78 str.5.

C309. Bloch, A.: Some Aspects of a Processor-Controlled PABX; TECI * 78 Annual Conference Proceedings on Industrial Applications of Micropro Philadelphia, PA, USA 20-22 March 1978.

C310. England, D.: Teleprocessing Monitor/Logger; Microprocessors /GB/, Vol.2, No.4, P.225-30 /Aug. 1978/.

C311. Davis, D.C.: A Microprocessor-Controlled Communications Terminal; Proceedings of the IEEE 1978. National Aerospace and Electronics Conference NA ECON. 78, Dayton, OH., USA 16-18 May 1978.

C312. Jennings, F.D.G.: Automatic Service Monitor for Telephone Switching Systems;
IECI * 78 Annual Conference Proceedings on Industrial Applications of Microprocessors;, Philadelphia, PA, USA 20-22 March 1978.

C313. O Haver, I.: Audio Processing With a Microprocessor; Byte /USA/, Vol.3, No.6, P.166-73 /June 1978/.

C314. Tierney, J., Blankenship, P.: Microprocessor Applications in Narrowband Speech Devices; Proceedings of the 1978. IEEE International Symposium on Circuits and Systems: New York, USA 17-19 May 1978.

C315. Troughton, P.: Improvement of Operator Efficiency by Use of Microprocessors-Automatic Call Recording Equipment /ACRE/:
Communications 78. Conference on Communications Equipment and Systems: Birmingham, England 4-7 April 1978.

C316. Trueman, P.: The Plessy K1; Communications 78. Conference on Communications Equipment and Systems:, Birmingham, England 4-7 April 1978.

C317. Bol'shakov, I.A., Rakoshits, V.S.: Application of Orthogonal Systems of Discrete Functions to Microprocessor Processing of Signals; Tekh. Kibern. /USSR/, Eng. Cybern. /USA/, Vol.15, No.5, P.143-57/Sept.Oct. 1977/.

C318. Carter, P.: Future Trends in Data Communications; Commun. Int. /GB/; Vol.5, No.6, P.30, 34 /June 1978/.

C319. Stuttard, E.B.: Microprocessors in Data Communications; Communications 78. Conference on Communications Equipment and Systems:, Birmingham, England 4-7 April 1978.

C320. Colby, K.M., Christinaz, D., Graham, S.: A Computer Driven, Personal, Portable, and Intelligent Speech Prosthesis; Comput, and Biomed. Res. /USA/; Vol.11, No.4, /Aug.1978/.

C321. -: The Microprocessor Learns to Talk Elektronikschau /Austria/; Vol.54, No.1 /1978/.

C322. Barnes, DAA., Koller, F.R.: GTF Autom. Electr. J. /USA/; Vol.16, No.4, July 1978.

C323. Parasuraman, B., Williams, F.H.: Central Office Monitoring System; Telecommunications /USA/: Vol.12, No.7/July 1978/.

C324. Arif Motiwala, M.: Harris Adaptive ARQ /HAARO/; Communications 78. Conference on Communications Equipment and Systems: Birmingham, England 4-7 April 1978.

C325. Robson, P.G., Maycock, A.G.: Software Development-the Ptarmigan Approach; Communications 78. Conference on Communications Equipment and Systems: Birmingham, England 4-7 April 1978.

C326. Yonton, P.C., West, N.: The Ptarmigan Switch; Communications 78. Conference on Communications Equipment and Systems:, Birmingham, England 4-7 April 1978.

C327. Taneki, T.R.: 11-Byte Program Generates 2 Billion Pseudorandom Bits; Electronics /USA/, Vol.51, No.21, P.148, / 12 Oct. 1978/.

C328. Brafman, J.P. /Szczupak, J./ Mitra, S.K.: An Approach to the Implementation of Digital Filters Using Microprocessors; IEEE Trans. Acoust., Speech and Signal Process. /USA/ Vol. ASSP-26, No.5, P.442-6, /Oct. 1978/.

C329. Knuth, K.: Colour Television Receivers with Simple Pre-Programming; Funkschau /Germany/, Vol.50, No.9, P.405-7, /17 April 1978/.

C330. Smith, G.K.L.: PR. 2250-A New HF Receiver for Local/Remote Systems; Conference on Communications Equipment and Systems, Birmingham, England 4-7 April 1978.

C331. Bishop, A.: Microprocessors in Telecommunications; Commun. Int. /GB/, Vol.5, No.7, P.43, 46-7 /July 1978/.

C332. Thomasset, D. /Retif, J.M./ Masson, J.P./Reynaud, R.: Stochastic Coding:; Mes. Regul. Autom. /France/, Vol.43, No.6-7, P.77-80 /June-July 1978/.

C333. Carpenter, P.J./Sokol, J., JR./ Rosenthal, P.: A Microprocessor-Based Local Network Node; Proceedings of Compcon Fall * 78, Computer Communications Networks, Washington DC. USA 5-8 Sept. 1978.

C334. Kunzke, H.J.: Microprocessors and Programmable Calculators as Aids in Relay Design; Proceedings of the 26 th Relay Conference, Stillwater, OK, USA 25-26, April 1978.

C335. Fang, G.S.: Reliability of a Microprocessor-Based Protection Switching System; Bell Syst. Tech. J. /U.S.A/, Vol.57, No.7, pt.2, P.2633-61 /Sept. 1978/.

C336. Matoba, T. /Yamauchi, I/ Matsumoto, H./ Ikeda, T./ Nakajima, K.: A Data way for Distributed Control Systems;

WARUNKI PRENUMERATY

Prenumeratę na kraj przyjmują Oddziały RSW "Prasa-Książka-Ruch" oraz urzędy pocztowe i doręczyciele w terminie do dnia 25 listopada na rok następny.

Cena prenumeraty rocznej zł 840.

Jednostki gospodarki uspołecznionej, instytucje, organizacje i wszelkiego rodzaju zakłady pracy zamawiają prenumeratę w miejscowych Oddziałach RSW "Prasa-Książka-Ruch", w miejscowościach zaś, w których nie ma Oddziałów RSW - w urzędach pocztowych.

Czytelnicy indywidualni opłacają prenumeratę wyłącznie w urzędach pocztowych i u doręczycieli.

Prenumeratę ze zleceniem wysyłki za granicę przyjmuje RSW "Prasa-Książka-Ruch", Centrala Kolportażu Prasy i Wydawnictw, ul. Towarowa 28, 00-958 Warszawa, konto PKO Nr 1153-201045.

Prenumerata ze zleceniem wysyłki za granicę jest droższa od prenumeraty krajowej o 50% dla zleceniodawców indywidualnych i o 100% dla zlecających instytucji i zakładów pracy.

INDEKS 35362