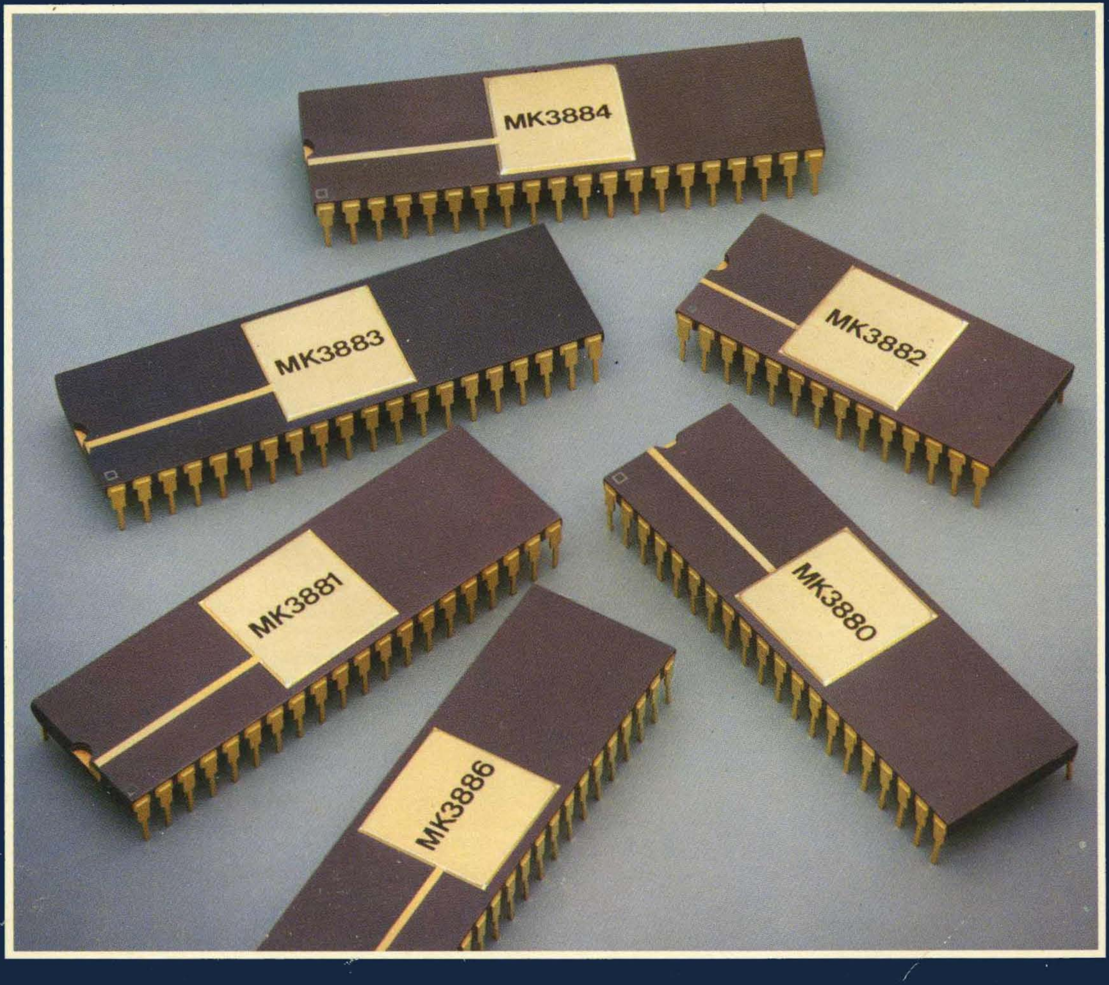
MOSTEK 1981

Z80 MICROCOMPUTER DATA BOOK

MOSTEK 1981 Z80 MICROCOMPUTER DATA BOOK



1981
Z80 MICROCOMPUTER
DATA BOOK

Mostek reserves the right to make changes in specifications at any time and without notice. The information furnished by Mostek in this publication is believed to be accurate and reliable. However, no responsibility is assumed by Mostek for its use; nor for any infringements of patents or other rights of third parties resulting from its use. No license is granted under any patents or patent rights of Mostek.

The "PRELIMINARY" designation on a Mostek data sheet indicates that the product is not characterized. The specifications are subject to change, are based on design goals or preliminary part evaluation, and are not guaranteed. Mostek Corporation or an authorized sales representative should be consulted for current information before using this product. No responsibility is assumed by Mostek for its use; nor for any infringements of patents and trademarks or other rights of third parties resulting from its use. No license is granted under any patents, patent rights, or trademarks of Mostek. Mostek reserves the right to make changes in specifications at any time and without notice.

PRINTED IN USA February 1981

1981 Z80 MICROCOMPUTER DATA BOOK

I TABLE OF CONTENTS

II GENERAL INFORMA TION

III Z80 FAMILY TECHNICAL MANUALS

IV Z80 FAMILY DATA SHEETS

V Z80 DEVELOP MENT EQUIPMENT

VI Z80 MICRO COMPUTER APPLICATION NOTES

VII Z80 MICRO COMPUTER PERIPHERALS

VIII Z80 MILITARY/ HI-REL

1981 Z80 MICROCOMPUTER DATA BOOK

1981 Z80 MICROCOMPUTER DATA BOOK

I Table of Contents

I TABLE OF CONTENTS

II General Information

II GENERAL INFORMA-TION

III Z80 Family Technical Manuals

III Z80 FAMILY TECHNICAL MANUALS

IV Z80 Family Data Sheets

IV Z80 FAMILY DATA SHEETS

V Z80 Development Equipment

V Z80 DEVELOP-MENT EQUIPMENT

VI Z80 Microcomputer Application Notes

VI Z80 MICRO-COMPUTER APPLICATION NOTES

VII Z80 Microcomputer Peripherals

VII Z80 MICRO-COMPUTER PERIPHERALS

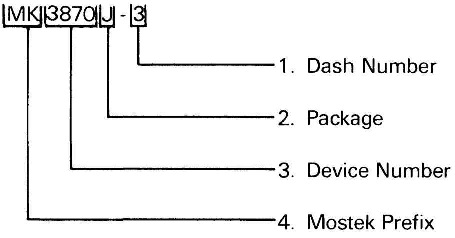
VIII Z80 Military/Hi-Rel

VIII Z80 MILITARY/HI-REL

ORDERING INFORMATION

Factory orders for parts described in this book should include a four-part number as explained below:

Example: MK 3870 J - 3



1. Dash Number

   One or two numerical characters defining specific device performance characteristic.

2. Package

P	-	Gold side-brazed ceramic DIP
J	-	CER-DIP
N	-	Epoxy DIP (Plastic)
R	-	P-PROM
K	-	Tin side-brazed ceramic DIP
T	-	Ceramic DIP with transparent lid
E	-	Ceramic leadless chip carrier

3. Device number

1XXX or 1XXXX	-	Shift Register, ROM
2XXX or 2XXXX	-	ROM, EPROM
3XXX or 3XXXX	-	ROM, EPROM
38XX	-	Microcomputer Components
4XXX or 4XXXX	-	RAM
5XXX or 5XXXX	-	Telecommunication and Industrial
7XXX or 7XXXX	-	Microcomputer Systems

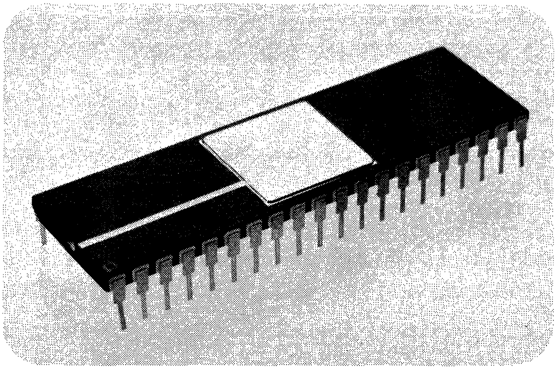
4. Mostek Prefix

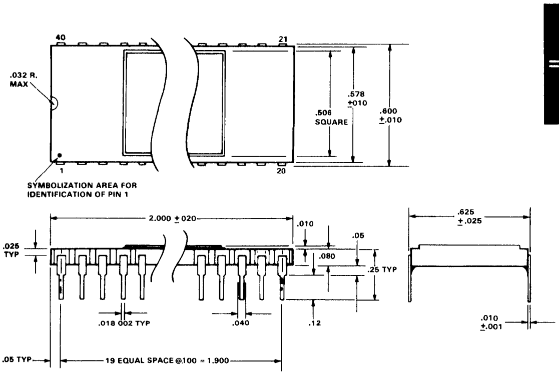
   MK-Standard Prefix

   MKB-100% 883B screening, with final electrical test at low, room and high-rated temperatures.

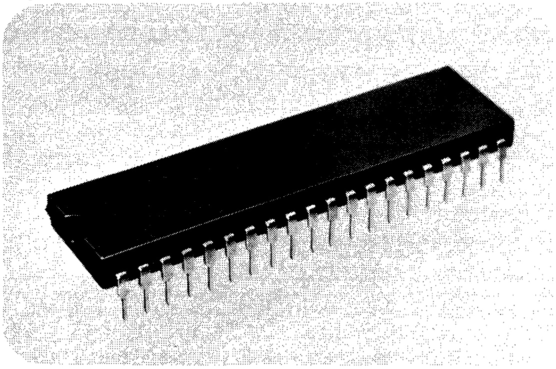
Package Descriptions

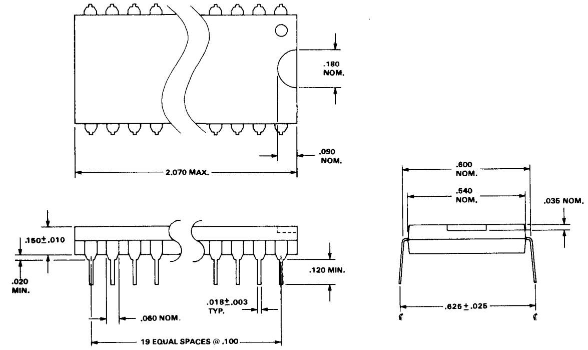
Ceramic Dual-In-Line Package (P) 40 Pin





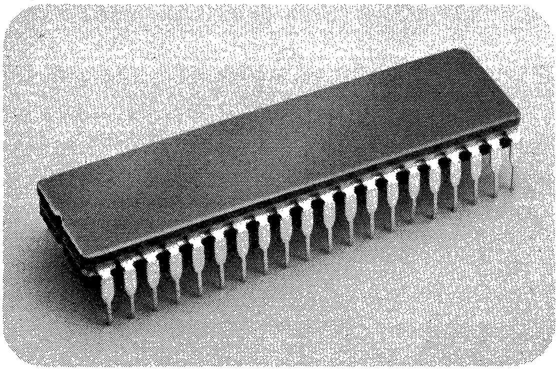
Plastic Dual-In-Line Package (N) 40 Pin

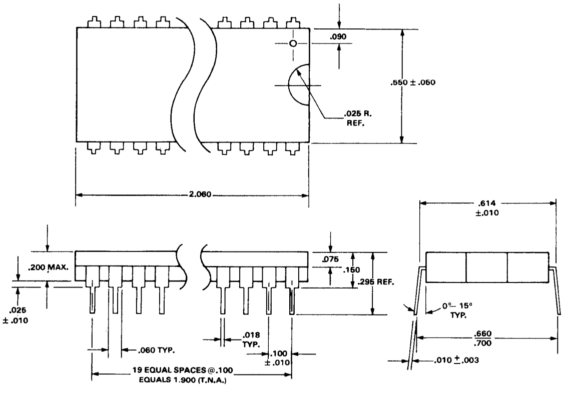




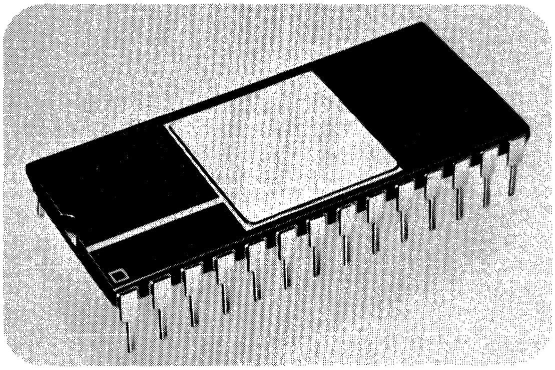
NOTE: Overall length includes .005 flash on either end of package.

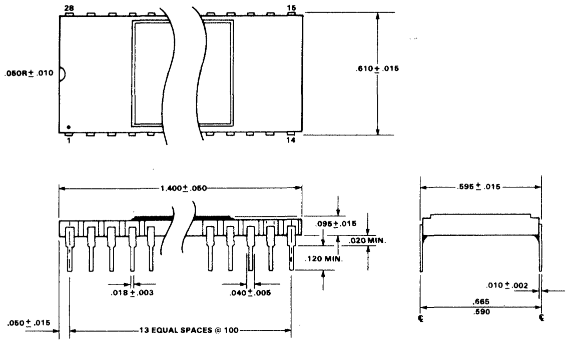
Cerdip Hermetic Packaging (J) 40 Pin



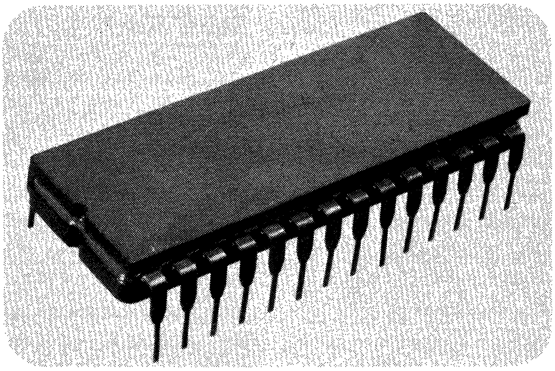


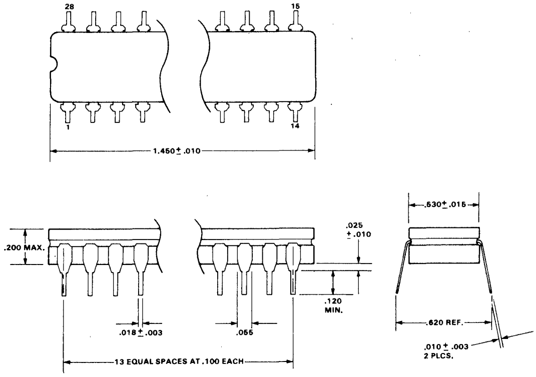
Ceramic Dual-In-Line Package (P)
28 Pin



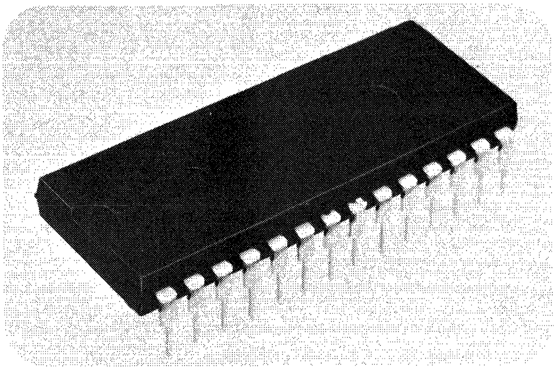


Cerdip Hermetic Packaging (J)
28 Pin

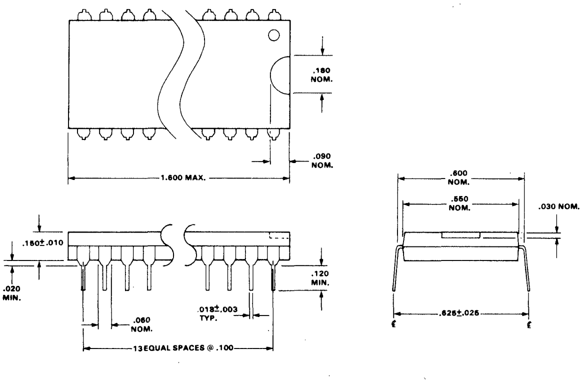




Plastic Dual-In-Line Package (N)
28 Pin

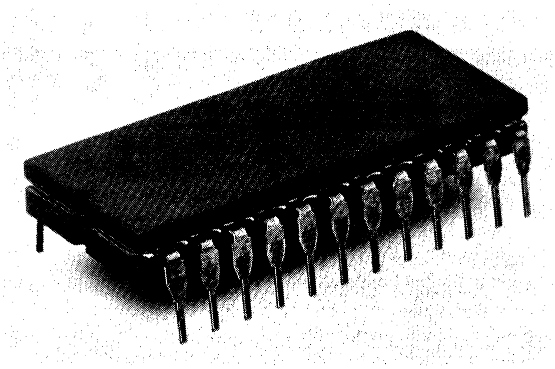


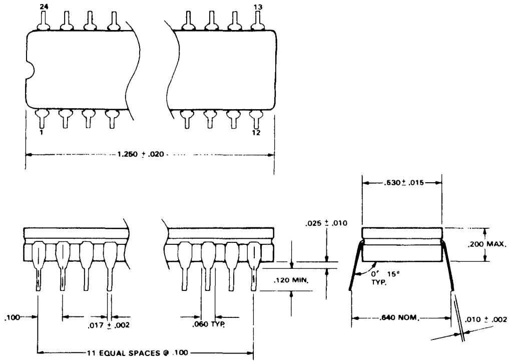
PLASTIC DUAL-IN LINE PACKAGING(N)
28 PIN



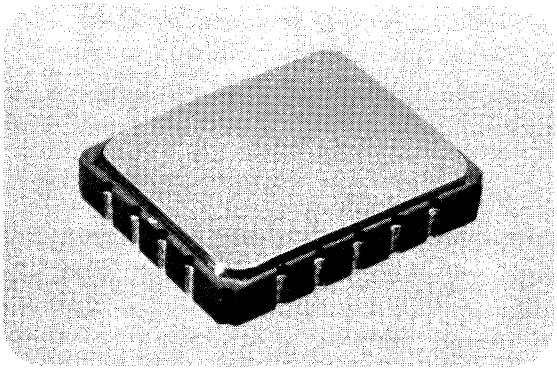
NOTE: Overall length includes .005 flash on either end of package

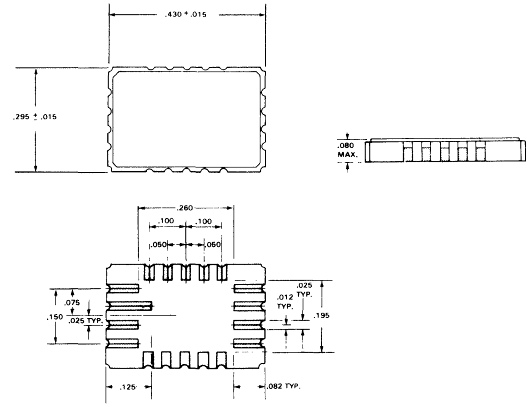
Cerdip Hermetic Package (J)
24 Pin



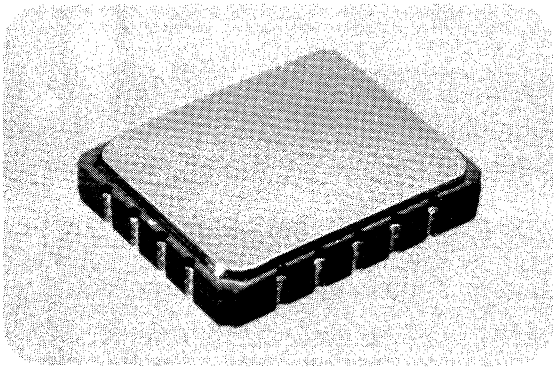


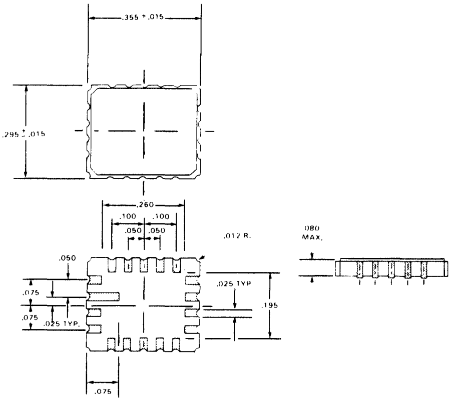
Leadless Hermetic Chip Carrier (E)
18 Pin



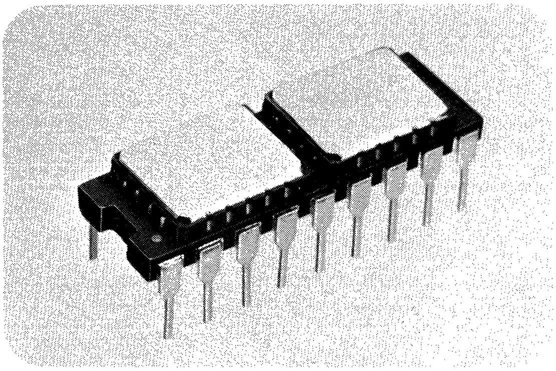


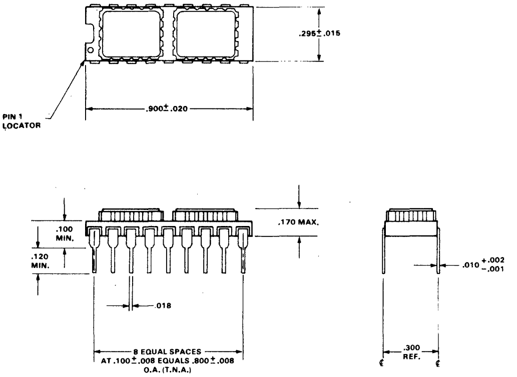
Leadless Hermetic Chip Carrier (E)
18 Pin



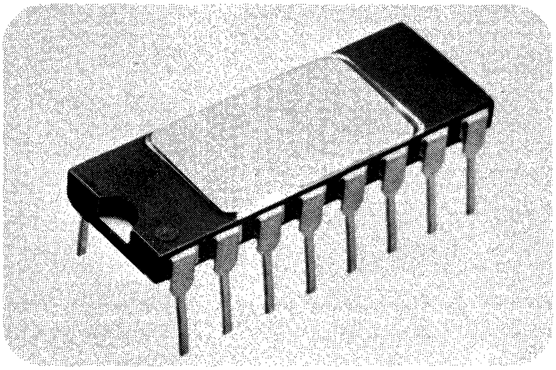


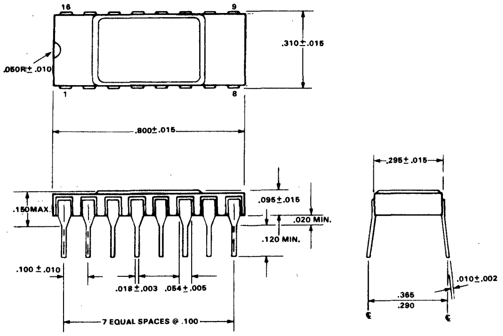
Dual-In-Line Double Density Ceramic Package (D)
18 Pin



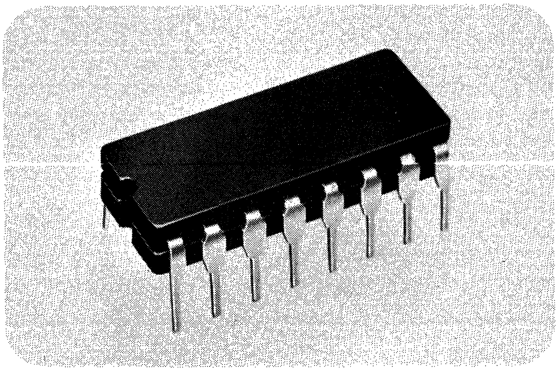


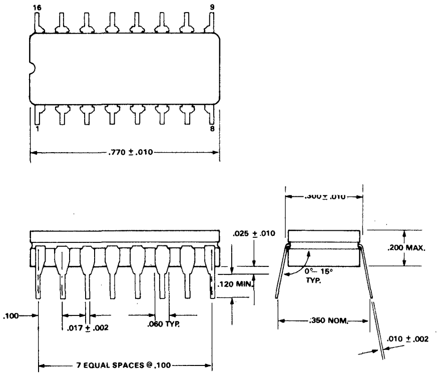
Ceramic Dual-In-Line Package (P)
16 Pin



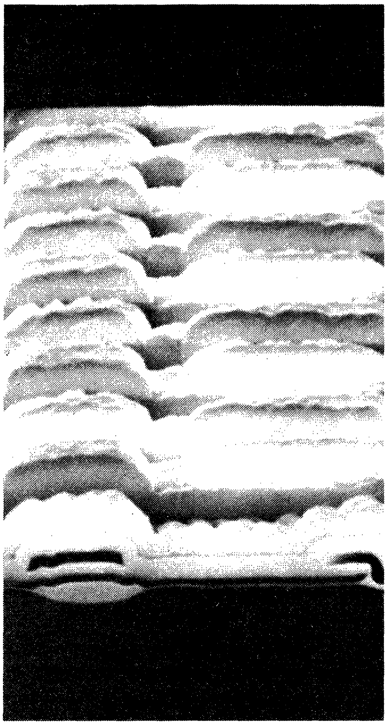


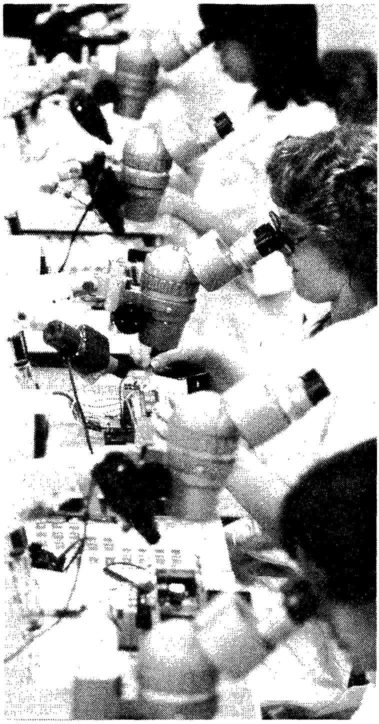
Cerdip Hermetic Package (J)
16 Pin





Mostek - Technology For Today And Tomorrow





TECHNOLOGY

From its beginning, Mostek has been an innovator. From the developments of the 1K dynamic RAM and the single-chip calculator in 1970 to the current 64K dynamic RAM, Mostek technological breakthroughs have proved the benefits and cost-effectiveness of metal oxide semiconductors. Today, Mostek represents one of the industry's most productive bases of MOS/LSI technology, including Direct-Step-on-Wafer processing and ion-implantation techniques.

The addition of the Microelectronics Research Center in Colorado Springs adds a new dimension to Mostek circuit design capabilities. Using the latest computer-aided design techniques, center engineers will be keeping ahead of the future with new technologies and processes.

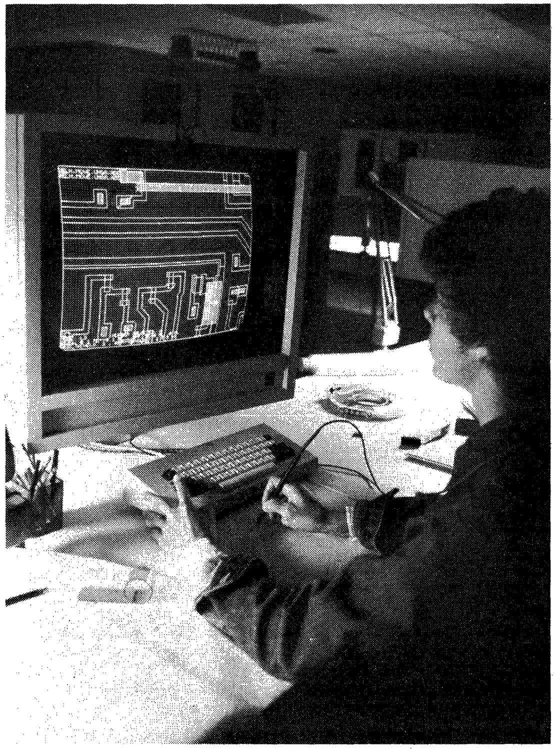
QUALITY

The worth of a product is measured by how well it is designed, manufactured and tested and by how well it works in your system.

In design, production and testing, the Mostek goal is meeting specifications the first time on every product. This goal requires strict discipline from the company and from its individual employees. Discipline, coupled with very personal pride, has enabled Mostek to build in quality at every level of production.

PRODUCTION CAPABILITY

The commitment to increasing production capability has made Mostek the world's largest manufacturer of dynamic RAMs. We entered the telecommunications market in 1974 with a tone dialer, and have shipped millions of telecom circuits since then. More than two million of our MK3870 single-chip microprocessors are in use throughout the world. To meet the demand, production capability is being constantly increased. Recent construction in Dallas, Ireland and Colorado Springs has added some 50 percent to the Mostek manufacturing capacity.



THE PRODUCTS

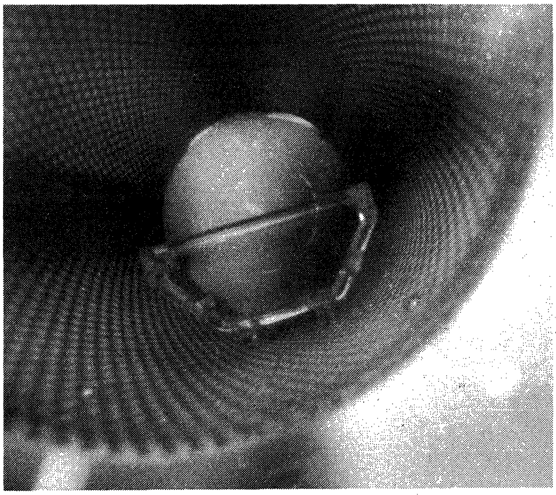
Telecommunications Products

Mostek is the leading supplier of tone dialers, pulse dialers, and CODEC devices. As each new generation of telecommunications systems emerges, Mostek is ready with new generation components, including PCM filters, tone receivers, repertory dialers, new integrated tone dialers, and pulse dialers.

These products, many of them using CMOS technology, represent the most modern advancements in telecommunications component design.

Industrial Products

Mostek's line of Industrial Products offers a high degree of versatility per device. This family of components includes various microprocessor-compatible A/D converters, a counter/time-base circuit for the division of clock signals, and combined counter/display decoders. As a result of the low parts count involved, an economical alternative to discrete logic systems is provided.



Memory Products

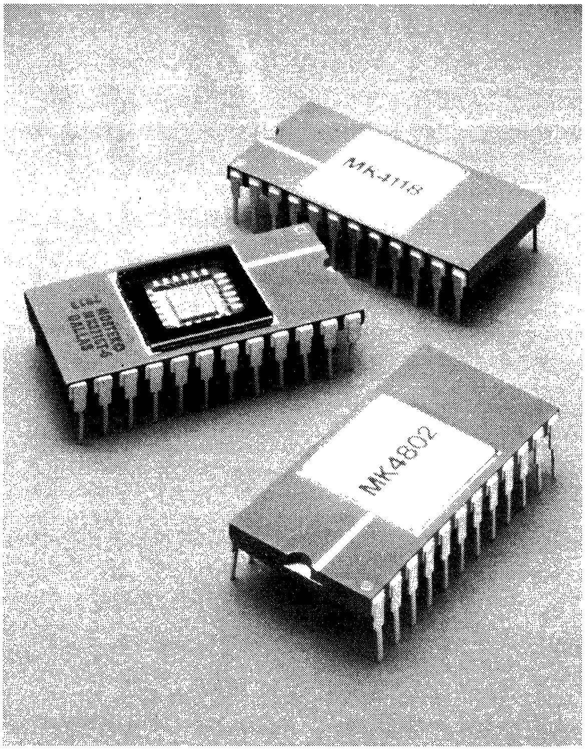
Through innovations in both circuit design, wafer processing and production, Mostek has become the industry's leading supplier of memory products.

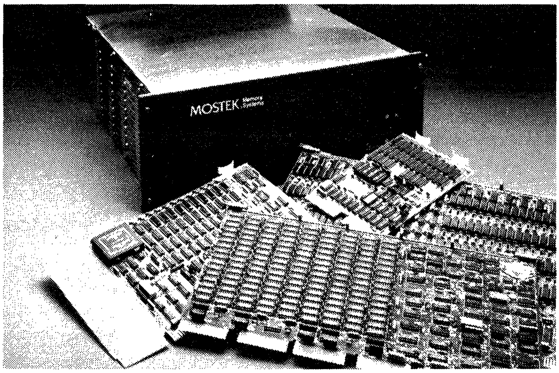
An example of Mostek leadership is our new BYTEWYDE™ family of static RAMs, ROMs, and EPROMs. All provide high performance, N words x 8-bit organization and common pin configurations to allow easy system upgrades in density and performance. Another important product area is fast static RAMs. With major advances in technology, Mostek static RAMs now feature access times as low as 55 nanoseconds. With high density ROMs and PROMs, static RAMs, dynamic RAMs and pseudostatic RAMs, Mostek now offers one of industry's broadest and most versatile memory product lines.

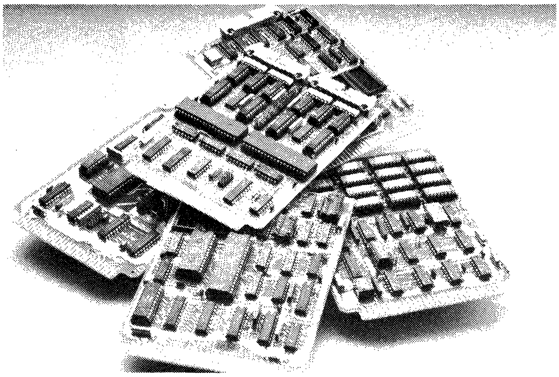
Microcomputer Components

Mostek's microcomputer components are designed for a wide range of applications.

Our Z80 family is today's industry standard 8-bit microcomputer. The MK3870 family is one of the industry's most popular 8-bit single-chip microcomputers, offering upgrade options in ROM, RAM and I/O, all in the same socket. The 38P7X EPROM versions support and prototype the entire family.







Microcomputer Systems

Complementing the component product line is the powerful MATRIX™ microcomputer development system, a Z80-based, dual floppy-disk system that is used to develop and debug software and hardware for all Mostek microcomputers.

A software operating system, FLP-80DOS, speeds and eases the design cycle with powerful commands. BASIC, FORTRAN, and PASCAL are also available for use on the MATRIX.

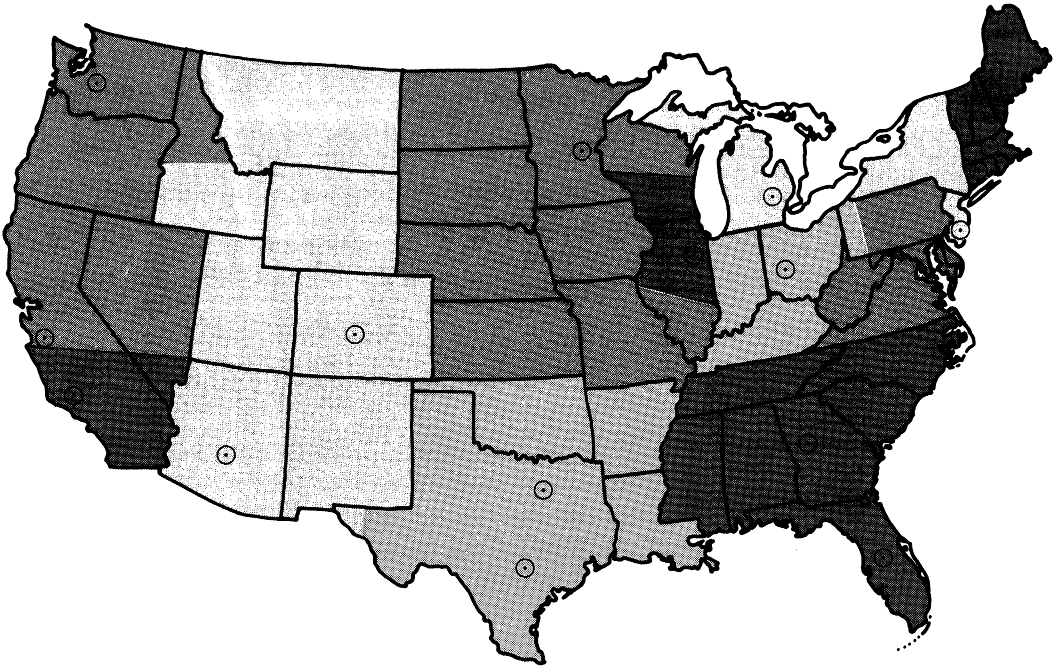
Mostek's MD Series™ features both stand-alone microcomputer boards and expandable microcomputer boards. The expandable boards are modularized by function, reducing system cost because the designer buys only the specific functional modules his system requires. All MDX boards are STD-Z80 BUS compatible.

Memory Systems

Taking full advantage of our leadership in memory components technology, Mostek Memory Systems offers a broad line of products, all with the performance and reliability to match our industry-standard circuits. Mostek Memory Systems offers add-in memory boards for popular DEC and Data General minicomputers.

Mostek also offers special purpose and custom memory boards for special applications.

U.S. AND CANADIAN SALES OFFICES



CORPORATE HEADQUARTERS

Mostek Corporation
1215 W. Crosby Rd.
P. O. Box 169
Carrollton, Texas 75006

REGIONAL OFFICES

Eastern U.S./Canada
Mostek
49 W. Putnam, 3rd Floor
Greenwich, Conn. 06830
203/622-0955
TWX 710-579-2928

Northeast U.S.
Mostek
29 Cummings Park, Suite #426
Woburn, Mass. 01801
617/935-0635
TWX 710-348-0459

Mid-Atlantic U.S.
Mostek
East Gate Business Center
125 Gaither Drive, Suite D
Mt. Laurel, New Jersey 08054
609/235-4112
TWX 710-897-0723

Southeast U.S.
Mostek
Exchange Bank Bldg.
1111 N. Westshore Blvd.
Suite 414
Tampa, Florida 33607
813/876-1304
TWX 810-876-4611

Atlanta Region
2 Exchange Place
2300 Peachford Rd. #2105
Atlanta, GA 30338
404/458-7922
TWX 810-757-4231

Upstate NY Region
Mostek
4651 Crossroads Park Dr., Suite 201
Liverpool, NY 13088
315/457-2160

Florida Region
Mostek
22521 Southwest 66th Ave.
Apt. A211
Boca Raton, FL 33433

Chicago Region
Mostek
701 E. Irving Park Road
Suite 206
Roselle, Ill. 60172
312/529-3993
TWX 910-291-1207

North Central U.S.
Mostek
6101 Green Valley Dr.
Bloomington, Mn. 55438
612/831-2322
TWX 910-576-2802

South Central U.S.
Mostek
3400 S. Dixie Ave.
Suite 101
Kettering, Ohio 45439
513/299-3405
TWX 810-459-1625

Michigan
Mostek
Livonia Pavillion East
29200 Vassar, Suite 520
Livonia, Mich. 48152
313/478-1470
TWX 810-242-2978

Central U.S.
Mostek
4100 McEwen Road
Suite 151
Dallas, Texas 75234
214/386-9340

Southwest Region
Mostek
4100 McEwen Road
Suite 237
Dallas, Texas 75234
214/386-9141
TWX 910-860-5437

Chevy Chase #4
7715 Chevy Chase Dr., #116
Austin, TX 78752
512/458-5226
TWX 910-874-2007

Western Region
Northern California
Mostek
1762 Technology Drive
Suite 126
San Jose, Calif. 95110

Seattle Region
Mostek
1107 North East 45th St.
Suite 411
Seattle, WA 98105
206/632-0245
TWX 910-444-4030

Southern California
Mostek
18004 Skypark Blvd.
Suite 140
Irvine, Calif. 92714
714/549-0397
TWX 910-595-2513

Arizona Region
Mostek
2150 East Highland Ave.
Suite 101
Phoenix, AZ 85016
602/954-6260
TWX 910-957-4581

Denver Region
3333 Quebec Street, #9090
Denver, CO 80207
303/321-6545
TWX 910-931-2583

U.S. AND CANADIAN REPRESENTATIVES

ALABAMA
Beacon Elect. Assoc., Inc.
11309 S. Memorial Pkwy.
Suite G
Huntsville, AL 35803
205/881-5031
TWX 810-726-2136

ARIZONA
Summit Sales
7825 E. Redfield Rd.
Scottsdale, AZ 85260
602/998-4850
TWX 910-950-1283

ARKANSAS
Beacon Elect. Assoc., Inc.
P.O. Box 5382, Brady Station
Little Rock, AK 72215
501/224-5449
TWX 910-722-7310

CALIFORNIA
Harvey King, Inc.
8124 Miramar Road
San Diego, CA 92126
714/566-5252
TWX 910-335-1231

COLORADO
Waugaman Associates
4800 Van Gordon
Wheat Ridge, CO 80033
303/423-1020
TWX 910-938-0750

CONNECTICUT
New England Technical Sales
240 Pomeroy Ave.
Meriden, CT 06450
203/237-8827
TWX 710-461-1126

FLORIDA
Conley & Associates, Inc.
P.O. Box 309
235 S. Central
Oviedo, FL 32765
305/365-3283
TWX 810-856-3520

Conley & Associates, Inc.
4021 W. Waters
Suite 2
Tampa, FL 33614
813/885-7658
TWX 810-876-9136

Conley & Associates, Inc.
P.O. Box 700
1612 N.W. 2nd Avenue
Boca Raton, FL 33432
305/395-6108
TWX 510-953-7548

GEORGIA
Conley & Associates, Inc.
3951 Pleasantdale Road
Suite 201
Doraville, GA 30340
404/447-6992
TWX 810-766-0488

ILLINOIS
Carlson Electronic Sales*
600 East Higgins Road
Elk Grove Village, IL 60007
312/956-8240
TWX 910-222-1819

INDIANA
Rich Electronic Marketing*
599 Industrial Drive
Carmel, IN 46032
317/844-8462
TWX 810-260-2631

Rich Electronic Marketing
3448 West Taylor St.
Fort Wayne, IN 46804
219/672-3329
TWX 810-332-1404

IOWA
Cahill, Schmitz & Cahill, Inc.
208 Collins Rd. N.E. Suite K
Cedar Rapids, IA 52402
319/377-8219
TWX 910-525-1363

Carlson Electronic Sales
204 Collins Rd. NE
Cedar Rapids, IA 52402
319/377-6341
TWX 910-222-1819

KANSAS
Rush & West Associates*
107 N. Chester Street
Olathe, KN 66061
913/764-2700
TWX 910-749-6404

KENTUCKY
Rich Electronic Marketing
5910 Bardstown Road
P. O. Box 91147
Louisville, KY 40291
502/239-2747
TWX 810-535-3757

MARYLAND
Arbotek Associates
3600 St. Johns Lane
Ellicott City, MD 21043
301/461-1323
TWX 710-862-1874

MASSACHUSETTS
New England Technical Sales*
135 Cambridge Street
Burlington, MA 01803
617/272-0434
TWX 710-332-0435

MICHIGAN
Action Components
19547 Coachwood Rd.
Riverview, MI 48192
313/479-1242

MINNESOTA
Cahill, Schmitz & Cahill, Inc.
315 N. Pierce
St. Paul, MN 55104
612/646-7217
TWX 910-563-3737

MISSOURI
Rush & West Associates
481 Melanie Meadows Lane
Ballwin, MO 63011
314/394-7271

NORTH CAROLINA
Conley & Associates, Inc.
3301 Womans Club Drive
Suite 130
Raleigh, NC 27616
919/787-8090
TWX 510-928-1829

NEW JERSEY
Tritek Sales, Inc.
21 E. Euclid Ave.
Haddonfield, NJ 08033
609/429-1551
215/627-0149 (Philadelphia Line)
TWX 710-896-0881

NEW MEXICO
Waugaman Associates
P.O. Box 14894
Albuquerque, NM 87111
or
9004 Menaul NE
Suite 7
Albuquerque, NM 87112
505/294-1437
505/294-1436 (Ans. Service)

NEW YORK
ERA Inc.
354 Veterans Memorial Highway
Commack, NY 11725
516/543-0510
TWX 510-226-1485
(New Jersey Phone #
800/645-5500, 5501)

Precision Sales Corp.
5 Arbustus Ln., MR-97
Binghamton, NY 13901
607/648-3686

Precision Sales Corp.*
1 Commerce Blvd.
Liverpool, NY 13088
315/451-3480
TWX 710-545-0250

Precision Sales Corp.
3594 Monroe Avenue
Pittsford, NY 14534
716/381-2820

Precision Sales Corp.
Drake Road
Pleasant Valley, NY 12569
914/635-3233

OHIO
Rich Electronic Marketing
7221 Taylorsville Road
Dayton, Ohio 45424
513/237-9422
TWX 810-459-1767

Rich Electronic Marketing
141 E. Aurora Road
Northfield, Ohio 44067
216/468-0583
TWX 810-427-9210

OREGON
Northwest Marketing Assoc.
9999 S.W. Wilshire St.
Suite 124
Portland OR 97225
503/297-2581
TELEX 36-0465 (AMAPORT PTL)

TEXAS
Southern States Marketing, Inc.
P.O. Box 8000
Addison, TX 75001
214/387-2489
TWX 910-860-5138

Southern States Marketing, Inc.
7745 Chevy Chase
Suite 219
Austin, TX 78752
512/452-9459

Southern States Marketing, Inc.
9730 Town Park Drive, Suite 104
Houston, Texas 77036
713/988-0991
TWX 910-881-1630

UTAH
Waugaman Associates
2520 S. State Street
#224
Salt Lake City, UT 84115
801/467-4263
TWX 910-925-4073

WASHINGTON
Northwest Marketing Assoc.
12835 Bellevue-Redmond Rd.
Suite 203E
Bellevue, WA 98005
206/455-5846
TWX 910-443-2445

WISCONSIN
Carlson Electronic Sales
Northbrook Executive Ctr.
10701 West North Ave.
Suite 209
Milwaukee, WI 53226
414/476-2790
TWX 910-222-1819

CANADA
Cantec Representatives Inc.*
1573 Laperriere Ave.
Ottawa, Ontario
Canada K1Z 7T3
613/725-3704
TWX 610-562-8967

Cantec Representatives Inc.
83 Galaxy Blvd., Unit 1A
(Rexdale)
Toronto, Canada M9W 5X6
416/675-2460
TWX 610-492-2655

Cantec Representatives Inc.
15737 rue Pierrefonds St.
Ste-Genevieve, P. Q.
(Montreal) H9H 1G3
514/620-6313
TWX 610-422-3985

*Home Office

U.S. AND CANADIAN DISTRIBUTORS

ARIZONA
Kierulff Electronics
4134 E. Wood St.
Phoenix, AZ 85040
602/243-4101
TWX 910/951-1550

Wyle Distribution Group
8155 North 24th Avenue
Phoenix, Arizona 85021
602/249-2232
TWX 910/951-4282

CALIFORNIA
Bell Industries
1161 N. Fair Oaks Avenue
Sunnyvale, CA 94086
408/734-8570
TWX 910/339-9378

Arrow Electronics
521 Weddell Dr.
Sunnyvale, CA 94086
408/745-6600
TWX 910/339-9371

Kierulff Electronics
2585 Commerce Way
Los Angeles, CA 90040
213/725-0325
TWX 910/580-3106

Kierulff Electronics
8797 Balboa Avenue
San Diego, CA 92123
714/278-2112
TWX 910/335-1182

Kierulff Electronics
14101 Franklin Avenue
Tustin CA 92680
714/731-5711
TWX 910/595-2599

Schweber Electronics
17811 Gillette Avenue
Irvine, CA 92714
714/556-3880
TWX 910/595-1720

Wyle Distribution Group
124 Maryland Street
El Segundo, CA 90245
213/322-8100
TWX 910/348-7111

Wyle Distribution Group
9525 Chesapeake Drive
San Diego, CA 92123
714/565-9171
TWX 910/335-1590

Wyle Distribution Group
17872 Cowan Ave.
Irvine, CA 92714
714/641-1600
TWX 910/348-7111

Wyle Distribution Group
3000 Bowers Ave.
Santa Clara, CA 95051
408/727-2500
TWX 910/338-0296

COLORADO
Kierulff Electronics
10890 E. 47th Avenue
Denver, CO 80239
303/371-6500
TWX 910/932-0169

Wyle Distribution Group
451 E. 124th Ave.
Thornton, CO 80241
303/457-9953
TWX 910/936-0770

CONNECTICUT
Arrow Electronics
12 Beaumont Rd.
Wallingford, CT 06492
203/265-7741
TWX 710/476-0162

Schweber Electronics
Finance Drive
Commerce Industrial Park
Danbury, CT 06810
203/792-3500
TWX 710/456-9405

FLORIDA
Arrow Electronics
1001 N.W. 62nd St.
Suite 108
Ft. Lauderdale, FL 33309
305/776-7790
TWX 510/955-9456

Arrow Electronics
115 Palm Bay Road, N.W.
Suite 10 Bldg. 200
Palm Bay, FL 32905
305/725-1480
TWX 510/959-6337

Diplomat Southland
2120 Calumet
Clearwater, FL 33515
813/443-4514
TWX 810/866-0436

Kierulff Electronics
3247 Tech Drive
St. Petersburg, FL 33702
813/576-1966
TWX 810/863-5625

GEORGIA
Arrow Electronics
2979 Pacific Ave.
Norcross, GA 30071
404/449-8252
TWX 810/766-0439

Schweber Electronics
4126 Pleasantdale Road
Atlanta, GA 30340
404/449-9170

ILLINOIS
Arrow Electronics
492 Lunt Avenue
P. O. Box 94248
Schaumburg, IL 60193
312/893-9420
TWX 910/291-3544

Bell Industries
3422 W. Touhy Avenue
Chicago, IL 60645
312/982-9210
TWX 910/223-4519

Kierulff Electronics
1536 Lanmeier
Elk Grove Village, IL 60007
312/640-0200
TWX 910/222-0351

INDIANA
Advent Electronics
8446 Moller
Indianapolis, IN 46268
317/297-4910
TWX 810/341-3228

Ft. Wayne Electronics
3606 E. Maumee
Ft. Wayne, IN 46803
219/423-3422
TWX 810/332-1562

Pioneer/Indiana
6408 Castleplace Drive
Indianapolis, IN 46250
317/849-7300
TWX 810/260-1794

IOWA
Advent Electronics
682 58th Avenue
Court South West
Cedar Rapids, IA 52404
319/363-0221
TWX 910/525-1337

MASSACHUSETTES
Kierulff Electronics
13 Fortune Drive
Billerica, MA 01821
617/935-5134
TWX 710/390-1449

Lionex Corporation
1 North Avenue
Burlington, MA 01803
617/272-9400
TWX 710/332-1387

Schweber Electronics
25 Wiggins Avenue
Bedford, MA 01730
617/275-5100
TWX 710/326-0268

Arrow Electronics
96D Commerce Way
Woburn, MA 01801
617/933-8130
TWX 710/393-6770

MARYLAND
Arrow Electronics
4801 Benson Avenue
Baltimore, MD 21227
301/247-5200
TWX 710/236-9005

Schweber Electronics
9218 Gaither Rd.
Gaithersburg, MD 20760
301/840-5900
TWX 710/828-9749

MICHIGAN
Arrow Electronics
3810 Varsity Drive
Ann Arbor, MI 48104
313/971-8220
TWX 810/223-6020

Schweber Electronics
33540 Schoolcraft Road
Livonia, MI 48150
313/525-8100
TWX 810/242-2983

MINNESOTA
Arrow Electronics
5251 W. 73rd Street
Edina, MN 55435
612/830-1800
TWX 910/576-3125

Industrial Components
5229 Edina Industrial Blvd.
Minneapolis, MN 55435
612/831-2666
TWX 910/576-3153

MISSOURI
Olive Electronics
9910 Page Blvd.
St. Louis, MO 63132
314/426-4500
TWX 910/763-0720

Semiconductor Spec
3805 N. Oak Trafficway
Kansas City, MO 64116
816/452-3900
TWX 910/771-2114

NEW HAMPSHIRE
Arrow Electronics
1 Perimeter Rd.
Manchester, NH 03103
603/668-6968
TWX 710/220-1684

NEW JERSEY
Arrow Electronics
Pleasant Valley Avenue
Morrestown, NJ 08057
609/235-1900
TWX 710/897-0829

Arrow Electronics
285 Midland Avenue
Saddlebrook, NJ 07662
201/797-5800
TWX 710/988-2206

Kierulff Electronics
3 Edison Place
Fairfield, NJ 07006
201/575-6750
TWX 710/734-4372

Schweber Electronics
18 Madison Road
Fairfield, NJ 07006
201/227-7880
TWX 710/734-4305

U.S. AND CANADIAN DISTRIBUTORS

NEW MEXICO

Bell Industries
11728 Linn N.E.
Albuquerque, NM 87123
505/292-2700
TWX 910/989-0625

Arrow Electronics
2460 Alamo Ave. S.E.
Albuquerque, NM 87106
505/243-4566
TWX 910/989-1679

NEW YORK

Arrow Electronics
900 Broad Hollow Rd.
Farmingdale, L.I., NY 11735
516/694-6800
TWX 510/224-6494

Arrow Electronics
7705 Maltlage Drive
P. O. Box 370
Liverpool, NY 13088
315/652-1000
TWX 710/545-0230

Arrow Electronics
3000 S. Winton Road
Rochester, NY 14623
716/275-0300
TWX 510/253-4766

Arrow Electronics
20 Oser Ave.
Hauppauge, NY 11787
516/231-1000
TWX 510/227-6623

Lionex Corporation
400 Oser Ave.
Hauppauge, NY 11787
516/273-1660
TWX 510/221-2196

Schweber Electronics
2 Twin Line Circle
Rochester, NY 14623
716/424-2222

Schweber Electronics
Jericho Turnpike
Westbury, NY 11590
516/334-7474
TWX 510/222-3660

NORTH CAROLINA

Arrow Electronics
938 Burke St.
Winston Salem, NC 27102
919/725-8711
TWX 510/931-3169

Hammond Electronics
2923 Pacific Avenue
Greensboro, NC 27406
919/275-6391
TWX 510/925-1094

OHIO

Arrow Electronics
7620 McEwen Road
Centerville, OH 45459
513/435-5563
TWX 810/459-1611

Arrow Electronics
10 Knoll Crest Drive
Reading, OH 45237
513/761-5432
TWX 810/461-2670

Arrow Electronics
6238 Cochran Road
Solon, OH 44139
216/248-3990
TWX 810/427-9409

Schweber Electronics
23880 Commerce Park Road
Beachwood, OH 44122
216/464-2970
TWX 810/427-9441

Pioneer/Cleveland
4800 East 131st Street
Cleveland, OH 44105
215/587-3600
TWX 810/422-2211

Pioneer/Dayton-Industrial
4433 Interpoint Blvd.
Dayton, OH 45424
513/236-9900
TWX 810/459-1622

OREGON

Kierulff Electronics
14273 NW Science Park
Portland, OR 97229
503/641-9150
TWX 910/467-8753

PENNSYLVANIA

Schweber Electronics
101 Rock Road
Horsham, PA 19044
215/441-0600

Arrow Electronics
650 Seco Rd.
Monroeville, PA 15146
412/856-7000

Pioneer/Pittsburgh
560 Alpha Drive
Pittsburgh, PA 15238
412/782-2300
TWX 710/795-3122

SOUTH CAROLINA

Hammond Electronics
1035 Lown Des Hill Rd.
Greenville, SC 29602
803/233-4121
TWX 810/281-2233

TEXAS

Arrow Electronics
13715 Gamma Road
P.O. Box 401068
Dallas, TX 75240
214/386-7500
TWX 910/860-5377

Quality Components
10201 McKalla
Suite D
Austin, TX 78758
512/835-0220
TWX 910/874-1377

Quality Components
4257 Kellway Circle
Addison, TX 75001
214/387-4949
TWX 910/860-5459

Quality Components
6126 Westline
Houston, TX 77036
713/772-7100

Schweber Electronics
7420 Harwin Drive
Houston, TX 77036
713/784-3600
TWX 910/881-1109

UTAH

Bell Industries
3639 W. 2150 South
Salt Lake City, UT 84120
801/972-6969
TWX 910/925-5686

Kierulff Electronics
2121 South 3600 West
Salt Lake City, UT 84104
801/973-6913

WASHINGTON

Kierulff Electronics
1005 Andover Park East
Tukwila, WA 98188
206/575-4420
TWX 910/444-2034

Wyle Distribution Group
1750 132nd Avenue N.E.
Bellevue, Washington 98005
206/453-8300
TWX 910/443-2526

WISCONSIN

Arrow Electronics
434 Rawson Avenue
Oak Creek, WI 53154
414/764-6600
TWX 910/262-1193

Kierulff Electronics
2212 E. Moreland Blvd.
Waukesha, WI 53186
414/784-8160
TWX 910/262-3653

CANADA

Prelco Electronics
2767 Thames Gate Drive
Mississauga, Ontario
Toronto L4T 1G5
416/678-0401
TWX 610/492-8974

Prelco Electronics
480 Port Royal St. W.
Montreal 357 P.Q. H3L 2B9
514/389-8051
TWX 610/421-3616

Prelco Electronics
1770 Woodward Drive
Ottowa, Ontario K2C 0P8
613/226-3491
Telex 05-34301

R.A.E. Industrial
3455 Gardner Court
Burnaby, B.C. V5G 4J7
604/291-8866
TWX 610/929-3065

Zentronics
141 Catherine Street
Ottawa, Ontario
K2P 1C3
613/238-6411
Telex 05-33636

Zentronics
1355 Meyerside Drive
Mississauga, Ontario
(Toronto) L5T 1C9
416/676-9000
Telex 06-983657

Zentronics
5010 Rue Pare
Montreal, Quebec
M4P 1P3
514/735-5361
Telex 05-827535

Zentronics
590 Berry Street
St. James, Manitoba
(Winnipeg) R2H OR4
204/775-8661

Zentronics
480A Dutton Drive
Waterloo, Ontario
N2L 4C6
519/884-5700

INTERNATIONAL MARKETING OFFICES

EUROPEAN HEAD OFFICE
Mostek International
Av de Tervuren 270-272
B-1150 Brussels/Belgium
02/762 18.80
Telex: 62011

FRANCE
Mostek France s.a.r.l.
30 Rue du Morvan
SILIC 505
F-94623 Rungis Cedex
(1) 687 34.14
Telex: 204049

GERMANY
PLZ 1-5
Mostek GmbH
FriedlandstraBe
D-2085 Quickborn
(4106) 2077/78
Telex: 213685

PLZ 6-7
Mostek GmbH
SchurwaldstraBe 15
D-7303 Neuhausen/Filder
7158/66.45
Telex: 72.38.86

PLZ 8
Mostek GmbH
Zaunkonigstr. 18
D-8021 Ottobrunn
089-609 1017
Telex: 5216516

ITALY
Mostek Italia SRL
Via G.D. Guerrazzi 27
I 20145 Milano
(02) 318.5337/349.2696
and 34.23.98
Telex: 333601

JAPAN
Mostek Japan KK
Sanyo Bldg 3F
1-2-7 Kita-Aoyama
Minato-Ku, Tokyo 107
(03) 404-7261
Telex: J23686

SWEDEN
Mostek Scandinavia AB
Magnusvagen 1/8 tr
S-1731 Jarfalla
0758-343 38/343 48
Telex: 12997

UNITED KINGDOM
Mostek U.K. Ltd.
Masons House,
1-3 Valley Drive
Kingsbury Road
London, N.W.9
01-204 9322
Telex: 25940

INTERNATIONAL SALES REPRESENTATIVES/DISTRIBUTORS

AUSTRIA
Transistor Vertriebsges, mbH
AuhofstraBe 41 A
A-1130 Vienna
(0222) 82 9451, 82 9404
Telex: 01-3738

BELGIUM
Sotronic
14 Rue Pere De Deken
B-1040 Brussels
02 736.10.07.
Telex: 25141

DENMARK
Semicap APS
Gammel Kongevej 148
DK-1850 Copenhagen
01-22.15.10
Telex: 15987

FINLAND
S.W. Instruments
Karstulantie 4B
SF-00550 Helsinki 55
8-0-73.82.65
Telex: 122411

FRANCE
Societe Copel
Rue Fourny, Z.I.
B.P. 22, F-78 530 BUC
(1)-735.33.20
Telex: 204 534

P.E.P.
4 Rue Barthelemy
F-92120 Montrouge
(1)-735.33.20
Telex: 204 534

SCAIB
80 Rue d'Arcueil
SILIC 137
F-94150 Rungis Cedex
(1) 687.23.12
Telex: 204674

Sorhodis
150-152 Rue A. France
F69100 Villeurbanne
(78) 850044
Telex: 380181

GERMANY
Dr Dohrenberg
Bayreuther StraBe 3
D-1000 Berlin 30
030-213.80.43
Telex: 0 184860

Neye Enatechnik GmbH
SchillerstraBe 14
D-2085 Quickborn
04106-612-1
Telex: 0 213.590

Branch offices in: Berlin, Hannover,
Dusseldorf, Darmstadt, Stuttgart,
Munchen.

Raffel-Electronic GmbH
LochnerstraBe 1
D-4030 Ratingen 1
0 2102-280.24
Telex: 8585180

Siegfried Ecker
Koenigsberger StraBe 2
D-6120 Michelstadt
0 6061-2233
Telex: 4191630

Matronic GmbH
Lichtenberger Weg 3
D-7400 Tubingen
07071-24331
Telex: 7262879

Dema-Electronic GmbH
BlutenstraBe 21
D-8000 Munchen 40
(089) 288018/19
Telex: 05-29345

ITALY
Comprel s.r.l.
V.le Romagna. 1
I-20092 Cinisello B. (MI)
(02) 61.20.641/2/3/4/5
Telex: 332484

Emesa S.P.A.
Via L. da Viadana, 9
I-20122 Milano
(02) 869.0616
Telex: 335066

Branch offices in
Bologna, Firenze,
Lavagna, Loreto,
Padova, Roma, Torino

THE NETHERLANDS
Nijkerk Elektronika BV
Drentestraat 7
1083 HK Amsterdam
(020) 428. 933
Telex: 11625

SWEDEN
Interelko AB
Strandbergsgatan, 47
S-12221 Enskede
081 132 160
Telex: 10 689

Lagercrantz Elektronik AB
Box M48 Kanalvagens
S-19421 Upplands Vasby
0760 861 20
Telex: 11275

SPAIN
Comelta S.A.
CiaElectronica Tecnicas Aplicadas
Diputacion, 79
Entlo 1-2
Barcelona-15
325 70 62
325 75 54
Telex: 519 34

Comelta S.A.
Emilio Munoz 41, ESC 1
Planta 1 Nave 2
Madrid-17
01-754 3001/3077
Telex: 42007

SWITZERLAND
Memotec AG
CH-4932 Lotzwil
063-28.11.22
Telex: 68636

NORWAY
Hefro Tekniska A/S
Postboks 6596
Rodelkka
Oslo 5
02-38.02.86
Telex: 16205

PORTUGAL
Digicontrole LDA
Rua Tenente Ferreira Durao 33 R/C
1300 Lisboa
19-688442/652613
Telex: 13639

UNITED KINGDOM
Celdis Limited
37-39 Loverock Road
Reading
Berks. RG 31 ED
0734-58.51.71
Telex: 848370

Lock Distribution Ltd.
Neville Street
Chadderton
Oldham
Lancashire
OL9 6LF
061-652.04.31
Telex: 669971

Pronto Electronic Systems Ltd,
466-478 Cranbrook Road
Gants Hill Illford
Essex 1G2 6LE
01-544 6222
Telex: 895 4213

VSI Electronics (UK) Ltd.
Roydondury Industrial Park
Horsecroft Rd.
Harlow
Essex CM19 5BY
(0279) 35477
Telex: 81387

YUGOSLAVIA
Chemcolor
Inozemma Zastupstva
Proleterskih brigada 37-a
41001 Zagreb
041-513.911
Telex: 21236

Branch office in Beograd

ISRAEL
Telsys Ltd.
12, Kehilat Venetsia St.
Tel Aviv. Israel
482126/7/8
Telex: 032392

For all other countries
MOSTEK INTERNATIONAL
Av de Tervuren 270-272
B-1150 Brussels/Belgium
02/762 18.80
Telex: 62011

or

MOSTEK CORPORATION
International Dept.
1215 West Crosby Road, Carrollton,
Texas 75006, USA
214/323.6000
Telex: 730423

1981 Z80 MICROCOMPUTER DATA BOOK

MK3880 CENTRAL PROCESSING UNIT

TABLE OF CONTENTS

Chapter Page

III
Z80 FAMILY
TECHNICAL
MANUALS

1.0 INTRODUCTION

The term "microcomputer" has been used to describe virtually every type of small computing device designed within the last few years. This term has been applied to everything from simple "microprogrammed" controllers constructed out of TTL MSI up to low end minicomputers with a portion of the CPU constructed out of TTL LSI "bit slices." However, the major impact of the LSI technology within the last few years has been with MOS LSI. With this technology, it is possible to fabricate complete and very powerful computer systems with only a few MOS LSI components.

The Mostek Z80 family of components is a significant advancement in the state-of-art of microcomputers. These components can be configured with any type of standard semiconductor memory to generate computer systems with an extremely wide range of capabilities. For example, as few as two LSI circuits and three standard TTL MSI packages can be combined to form a simple controller. With additional memory and I/O devices a computer can be constructed with capabilities that only a minicomputer could previously deliver. This wide range of computational power allows standard modules to be constructed by a user that can satisfy the requirements of an extremely wide range of applications.

The major reason for MOS LSI domination of the microcomputer market is the low cost of these few LSI components. For example, MOS LSI microcomputers have already replaced TTL logic in such applications as terminal controllers, peripheral device controllers, traffic signal controllers, point of sale terminals, intelligent terminals and test systems. In fact the MOS LSI microcomputer is finding its way into almost every product that now uses electronics and it is even replacing many mechanical systems such as weight scales and automobile controls.

The MOS LSI microcomputer market is already well established and new products using them are being developed at an extraordinary rate. The Mostek Z80 component set has been designed to fit into this market through the following factors:

1. The Z80 is fully software compatible with the popular 8080A CPU offered from several sources. Existing designs can be easily converted to include the Z80 as a superior alternative.

2. The Z80 component set is superior in both software and hardware capabilities to any other 8-bit microcomputer system on the market. These capabilities provide the user with significantly lower hardware and software development costs while also allowing him to offer additional features in his system.

3. A complete development and OEM system product line including full software support is available to enable the user to easily develop new products.

Microcomputer systems are extremely simple to construct using Z80 components. Any such system consists of three parts:

1. CPU (Central Processing Unit)

2. Memory

3. Interface circuits to peripheral devices

The CPU is the heart of the system. Its function is to obtain instructions from the memory and perform the desired operations. The memory is used to contain instructions and in most cases data that is to be processed. For example, a typical instruction sequence may be to read data from a specific peripheral device, store it in a location in memory, check the parity and write it out to another peripheral device. Note that the Mostek component set includes the CPU and various general purpose I/O device controllers, as well as a wide range of memory devices. Thus, all required components can be connected together in a very simple manner with virtually no other external logic. The user's effort then becomes primarily one of software development. That is, the user can concentrate on describing his problem and translating it into a series of instructions that can be loaded into the microcomputer memory. Mostek is dedicated to making this step of software generation as simple as possible. A good example of this is our assembly language in which a simple mnemonic is used to represent every instruction that the CPU can perform. This language is self documenting in such a way that from the mnemonic the user can understand exactly what the instruction is doing without constantly checking back to a complex cross listing.

2.0 Z80-CPU ARCHITECHURE

A block diagram of the internal architecture of the Z80-CPU is shown in Figure 2.0-1 The diagram shows all of the major elements in the CPU and it should be referred to throughout the following description.

Z80-CPU BLOCK DIAGRAM

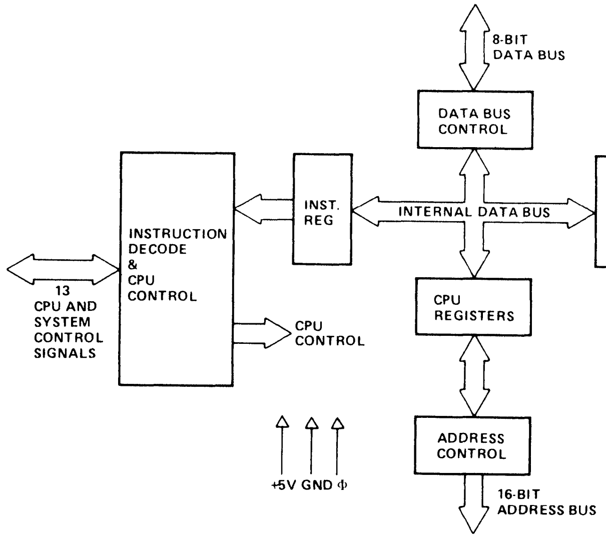


FIGURE 2.0-1

2.1 CPU REGISTERS

The Z80-CPU contains 208 bits of R/W memory that are accessible to the programmer. Figure 2.0-2 illustrates how this memory is configured into eighteen 8-bit registers and four 16-bit registers. All Z80 registers are implemented using static RAM. The registers include two sets of six general purpose registers that may be used individually as 8-bit registers or in pairs as 16-bit registers. There are also two sets of accumulator and flag registers.

Special Purpose Registers

1. **Program Counter (PC).** The program counter holds the 16-bit address of the current instruction being fetched from memory. The PC is automatically incremented after its contents have been transferred to the address lines. When a program jump occurs the new value is automatically placed in the PC, overriding the incrementer.

2. **Stack Pointer (SP).** The stack pointer holds the 16-bit address of the current top of a stack located anywhere in external system RAM memory. The external stack memory is organized as a last-in first-out (LIFO) file. Data can be pushed onto the stack from specific CPU registers or popped off of the stack into specific CPU registers through the execution of PUSH and POP instructions. The data popped from the stack is always the last data pushed onto it. The stack allows simple implementation of multiple level interrupts, unlimited subroutine nesting and simplification of many types of data manipulation.

Z80-CPU REGISTER CONFIGURATION

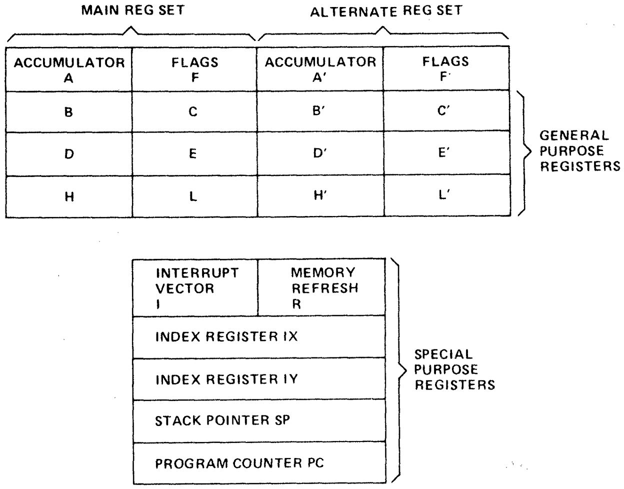


FIGURE 2.0-2

3. **Two Index Registers (IX & IY).** The two independent index registers hold a 16-bit base address that is used in indexed addressing modes. In this mode, an index register is used as a base to point to a region in memory from which data is to be stored or retrieved. An additional byte is included in indexed instructions to specify a displacement from this base. This displacement is specified as a two's complement signed integer. This mode of addressing greatly simplifies many types of programs, especially where tables of data are used.

4. **Interrupt Page Address Register (I).** The Z80-CPU can be operated in a mode where an indirect call to any memory location can be achieved in response to an interrupt. The I Register is used for this purpose to store the high order 8-bits of the indirect address while the interrupting device provides the lower 8-bits of the address. This feature allows interrupt routines to be dynamically located anywhere in memory with absolute minimal access time to the routine.

5. **Memory Refresh Register (R).** The Z80-CPU contains a memory refresh counter to enable dynamic memories to be used with the same ease as static memories. This 7-bit register is automatically incremented after each instruction fetch. The data in the refresh counter is sent out on the lower portion of the address bus along with a refresh control signal while the CPU is decoding and executing the fetched instruction. This mode of refresh is totally transparent to the programmer and does not slow down the CPU operation. The programmer can load the R register for testing purposes, but this register is normally not used by the programmer.

Accumulator and Flag Registers

The CPU includes two independent 8-bit accumulators and associated 8-bit flag registers. The accumulator holds the results of 8-bit arithmetic or logical operations while the flag register indicates specific conditions for 8 or 16-bit operations, such as indicating whether or not the result of an operation is equal to zero. The programmer selects the accumulator and flag pair that he wishes to work with with a single exchange instruction so that he may easily work with either pair.

General Purpose Registers

There are two matched sets of general purpose registers, each set containing six 8-bit registers that may be used individually as 8-bit registers or as 16-bit register pairs by the programmer. One set is called BC, DE, and HL while the complementary set is called BD', DE' and HL'. At any one time the programmer can select either set of registers to work with through a single exchange command for the entire set. In systems where fast interrupt response is required, one set of general purpose registers and an accumulator/flag register may be reserved for handling this very fast routine. Only a simple exchange command need be executed to go between the routines. This greatly reduces interrupt service time by eliminating the requirement for saving and retrieving register contents in the external stack during interrupt or subroutine processing. These general purpose registers are used for a wide range of applications by the programmer. They also simplify programming, especially in ROM based systems where little external read/write memory is available.

2.2 ARITHMETIC & LOGIC UNIT (ALU)

The 8-bit arithmetic and logical instructions of the CPU are executed in the ALU. Internally the ALU communicates with the registers and the external data bus on the internal data bus. The type of functions performed by the ALU include:

Add

Subtract

Logical AND

Logical OR

Logical Exclusive OR

Compare

Left or right shifts or rotates (arithmetic and logical)

Increment

Decrement

Set bit

Reset bit

Test bit

2.3 INSTRUCTION REGISTER AND CPU CONTROL

As each instruction is fetched from memory, it is placed in the instruction register and decoded. The control section performs this function and then generates and supplies all of the control signals necessary to read or write data from or to the registers, controls the ALU and provides all required external control signals.

3.0 Z80-CPU PIN DESCRIPTION

The Z80–CPU is packaged in an industry standard 40 pin Dual In-Line Package. The I/O pins are shown in Figure 3.0-1 and the function of each is described below.

Z80 PIN CONFIGURATION

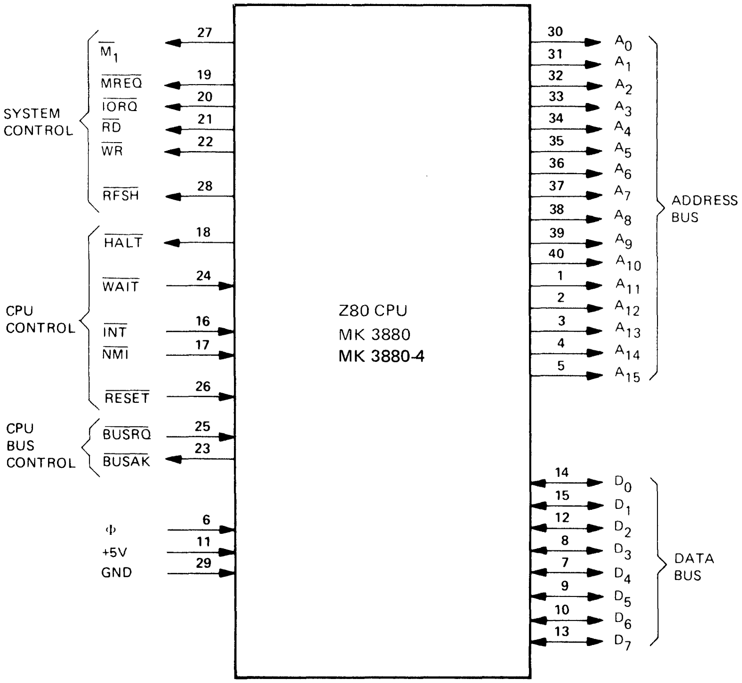


FIGURE 3.0-1

A_0-A_{15}
(Address Bus)

Tri-state output, active high. A_0-A_{15} constitute a 16-bit address bus. The address bus provides the address for memory (up to 64K bytes) data exchanges and for I/O device data exchanges. I/O addressing uses the 8 lower address bits to allow the user to directly select up to 256 input or 256 output ports. A_0 is the least significant address bit. During refresh time, the lower 7 bits contain a valid refresh address.

D_0-D_7
(Data Bus)

Tri-state input/output, active high. D_0-D_7 constitute an 8-bit bidirectional data bus. The data bus is used for data exchanges with memory and I/O devices.

$\overline{M_1}$
(Machine Cycle one)

Output, active low. $\overline{M_1}$ indicates that the current machine cycle is the OP code fetch cycle of an instruction execution. Note that during execution of 2-byte op-codes, $\overline{M_1}$ is generated as each op code byte is fetched. These two byte op-codes always begin with CBH, DDH, EDH, or FDH. $\overline{M_1}$ also occurs with $\overline{IORQ}$ to indicate an interrupt acknowledge cycle.

$\overline{MREQ}$
(Memory Request)

Tri-state output, active low. The memory request signal indicates that the address bus holds a valid address for a memory read or memory write operation.

$\overline{\text{IORQ}}$
(Input/Output Request)

Tri-state output, active low. The $\overline{\text{IORQ}}$ signal indicates that the lower half of the address bus holds a valid I/O address for a I/O read or write operation. An $\overline{\text{IORQ}}$ signal is also generated with an $\overline{M_1}$ signal when an interrupt is being acknowledged to indicate that an interrupt response vector can be placed on the data bus. Interrupt Acknowledge operations occur during M_1 time while I/O operations never occur during M_1 time.

$\overline{\text{RD}}$
(Memory Read)

Tri-state output, active low. $\overline{\text{RD}}$ indicates that the CPU wants to read data from memory or an I/O device. The addressed I/O device or memory should use this signal to gate data onto the CPU data bus.

$\overline{\text{WR}}$
(Memory Write)

Tri-state output, active low. $\overline{\text{WR}}$ indicates that the CPU data bus holds valid data to be stored in the addressed memory or I/O device.

$\overline{\text{RFSH}}$
(Refresh)

Output, active low. $\overline{\text{RFSH}}$ indicates that the lower 7 bits of the address bus contain a refresh address for dynamic memories and current $\overline{\text{MREQ}}$ signal should be used to do a refresh read to all dynamic memories. A_7 is a logic zero and the upper 8 bits of the Address Bus contains the I Register.

$\overline{\text{HALT}}$
(Halt state)

Output, active low. $\overline{\text{HALT}}$ indicates that the CPU has executed a HALT software instruction and is awaiting either a non maskable or a maskable interrupt (with the mask enabled) before operation can resume. While halted, the CPU executes NOP's to maintain memory refresh activity.

$\overline{\text{WAIT}}$*
(Wait)

Input, active low. $\overline{\text{WAIT}}$ indicates to the Z80-CPU that the addressed memory or I/O devices are not ready for a data transfer. The CPU continues to enter wait states for as long as this signal is active. This signal allows memory or I/O devices of any speed to be synchronized to the CPU.

$\overline{\text{INT}}$
(Interrupt Request)

Input, active low. The Interrupt Request signal is generated by I/O devices. A request will be honored at the end of the current instruction if the internal software controlled interrupt enable flip-flop (IFF) is enabled and if the $\overline{\text{BUSRQ}}$ signal is not active. When the CPU accepts the interrupt, an acknowledge signal ($\overline{\text{IORQ}}$ during M_1 time) is sent out at the beginning of the next instruction cycle. The CPU can respond to an interrupt in three different modes that are described in detail in section 8.

$\overline{\text{NMI}}$

Input, negative edge triggered. The non maskable interrupt request line has a higher priority than $\overline{\text{INT}}$ and is always recognized at the end of the current instruction, independent of the status of the interrupt enable flip-flop. $\overline{\text{NMI}}$ automatically forces the Z80-CPU to restart to location 0066_H. The program counter is automatically saved in the external stack so that the user can return to the program that was interrupted. Note that continuous WAIT cycles can prevent the current instruction from ending, and that a $\overline{\text{BUSRQ}}$ will override a $\overline{\text{NMI}}$.

$\overline{\text{RESET}}$

Input, active low. $\overline{\text{RESET}}$ forces the program counter to zero and initializes the CPU. The CPU initialization includes:

1) Disable the interrupt enable flip-flop
2) Set Register I = 00_H
3) Set Register R = 00_H
4) Set Interrupt Mode 0

During reset time, the address bus and data bus go to a high impedance state and all control output signals go to the inactive state. No refresh occurs.

$\overline{\text{BUSRQ}}$
(Bus Request)

Input, active low. The bus request signal is used to request the CPU address bus, data bus and tri-state output control signals to go to a high impedance state so that other devices can control these buses. When $\overline{\text{BUSRQ}}$ is activated, the CPU will set these buses to a high impedance state as soon as the current CPU machine cycle is terminated.

$\overline{\text{BUSAK}}$*
(Bus Acknowledge)

Output, active low. Bus acknowledge is used to indicate to the requesting device that the CPU address bus, data bus and tri-state control bus signals have been set to their high impedance state and the external device can now control these signals.

Φ

Single phase system clock.

*While the Z80-CPU is in either a $\overline{\text{WAIT}}$ state or a Bus Acknowledge condition, Dynamic Memory Refresh will not occur.

4.0 CPU TIMING

The Z80-CPU executes instructions by stepping through a very precise set of a few basic operations. These include:

Memory read or write

I/O device read or write

Interrupt acknowledge

All instructions are merely a series of these basic operations. Each of these basic operations can take from three to six clock periods to complete or they can be lengthened to synchronize the CPU to the speed of external devices. The basic clock periods are referred to as T states and the basic operations are referred to as M (for machine) cycles. Figure 4.0-0 illustrates how a typical instruction will be merely a series of specific M and T cycles. Notice that this instruction consists of three machine cycles (M1, M2 and M3). The first machine cycle of any instruction is a fetch cycle which is four, five or six T states long (unless lengthened by the wait signal which will be fully described in the next section). The fetch cycle (M1) is used to fetch the OP code of the next instruction to be executed. Subsequent machine cycles move data between the CPU and memory or I/O devices and they may have anywhere from three to five T cycles (again they may be lengthened by wait states to synchronize the external devices to the CPU). The following paragraphs describe the timing which occurs within any of the basic machine cycles. In section 7, the exact timing for each instruction is specified.

BASIC CPU TIMING EXAMPLE

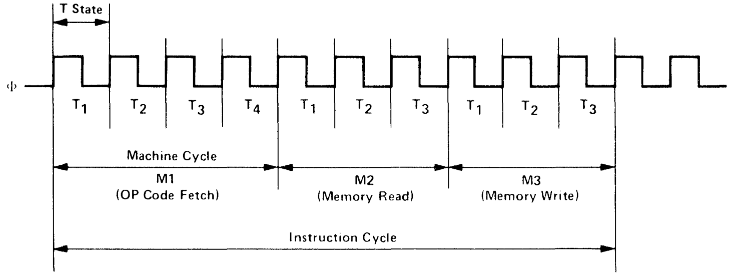


FIGURE 4.0-0

All CPU timing can be broken down into a few very simple timing diagrams as shown in Figure 4.0-1 through 4.0-7. These diagrams show the following basic operations with and without wait states (wait states are added to synchronize the CPU to slow memory or I/O devices).

4.0-1. Instruction OP code fetch (M1 cycle)

4.0-2. Memory data read or write cycles

4.0-3. I/O read or write cycles

4.0-4. Bus Request/Acknowledge Cycle

4.0-5. Interrupt Request/Acknowledge Cycle

4.0-6. Non maskable Interrupt Request/Acknowledge Cycle

4.0-7. Exit from a HALT instruction

INSTRUCTION FETCH

Figure 4.0-1 shows the timing during an M1 cycle (OP code fetch). Notice that the PC is placed on the address bus at the beginning of the M1 cycle. One half clock time later the MREQ signal goes active. At this time the address to the memory has had time to stabilize so that the falling edge of MREQ can be used directly as a chip enable clock to dynamic memories. The RD line also goes active to indicate that the memory read data should be enabled onto the CPU data bus. The CPU samples the data from the memory on the data bus with the rising edge of the clock of state T3 and this same edge is used by the CPU to turn off the RD and MREQ signals. Thus the data has already been sampled by the CPU before the RD signal becomes inactive. Clock state T3 and T4 of a fetch cycle are used to refresh dynamic memories. (The CPU uses this time to decode and execute the fetched instruction so that no other operation could be performed at this time). During T3 and T4 the lower 7 bits of the address bus contain a memory refresh address and the RFSH signal becomes active to indicate that a refresh read of all dynamic memories should be accomplished. Notice that a RD signal is not generated during refresh time to prevent data from different memory segments from being gated onto the data bus. The MREQ signal during refresh time should be used to perform a refresh read of all memory elements. The refresh signal can not be used by itself since the refresh address is only guaranteed to be stable during MREQ time.

INSTRUCTION OP CODE FETCH

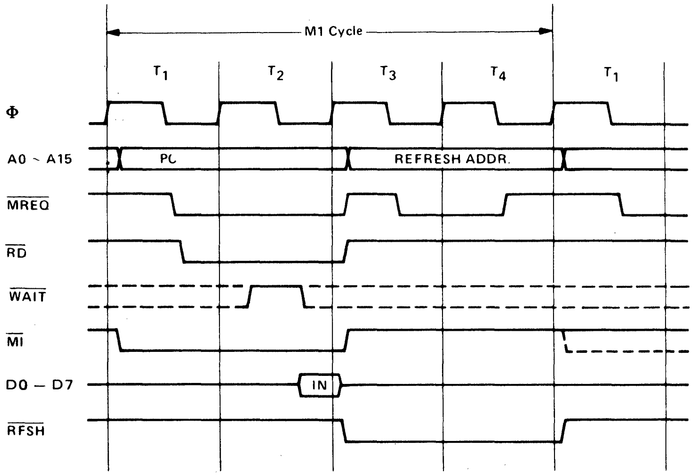


FIGURE 4.0-1

Figure 4.0-1A illustrates how the fetch cycle is delayed if the memory activates the WAIT line. During T2 and every subsequent Tw, the CPU samples the WAIT line with the falling edge of Φ. If the WAIT line is active at this time, another wait state will be entered during the following cycle. Using this technique the read cycle can be lengthened to match the access time of any type of memory device.

INSTRUCTION OP CODE FETCH WITH WAIT STATES

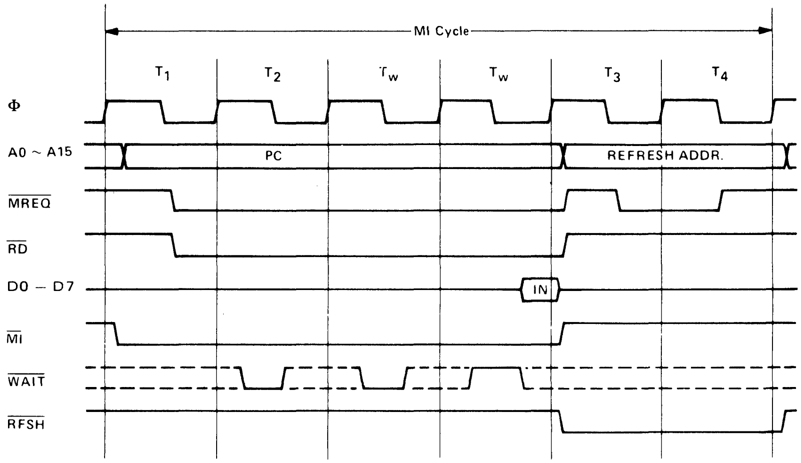


FIGURE 4.0-1A

MEMORY READ OR WRITE

Figure 4.0-2 illustrates the timing of memory read or write cycles other than an OP code fetch (M1 cycle). These cycles are generally three clock periods long unless wait states are requested by the memory via the $\overline{\text{WAIT}}$ signal. The $\overline{\text{MREQ}}$ signal and the $\overline{\text{RD}}$ signal are used the same as in the fetch cycle. In the case of a memory write cycle, the $\overline{\text{MREQ}}$ also becomes active when the address bus is stable so that it can be used directly as a chip enable for dynamic memories. The $\overline{\text{WR}}$ line is active when data on the data bus is stable so that it can be used directly as a R/W pulse to virtually any type of semiconductor memory. Furthermore the $\overline{\text{WR}}$ signal goes inactive one half T state before the address and data bus contents are changed so that the overlap requirements for virtually any type of semiconductor memory type will be met.

MEMORY READ OR WRITE CYCLES

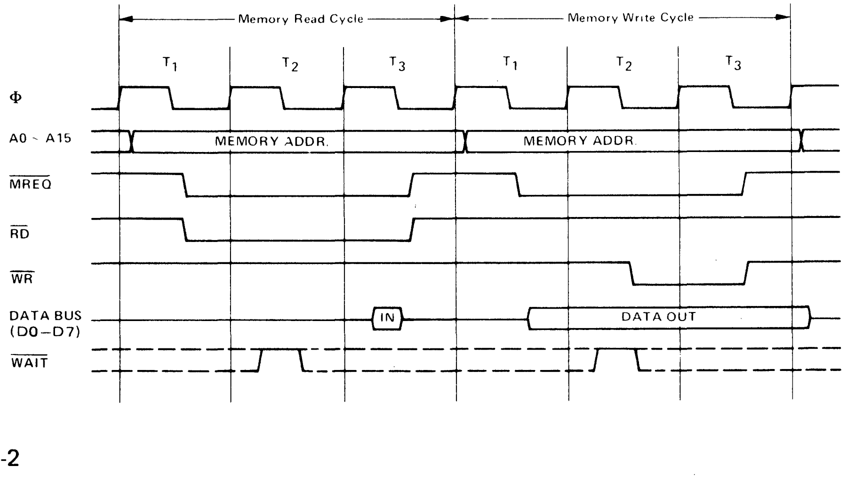


FIGURE 4.0-2

Figure 4.0-2A illustrates how a WAIT request signal will lengthen any memory read or write operation. This operation is identical to that previously described for a fetch cycle. Notice in this figure that a separate read and a separate write cycle are shown in the same figure although read and write cycles can never occur simultaneously.

MEMORY READ OR WRITE CYCLES WITH WAIT STATES

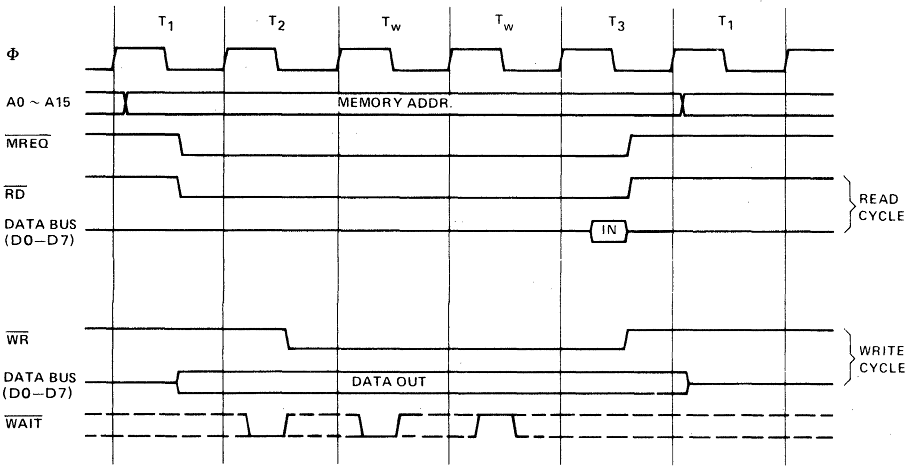


FIGURE 4.0-2A

INPUT OR OUTPUT CYCLES

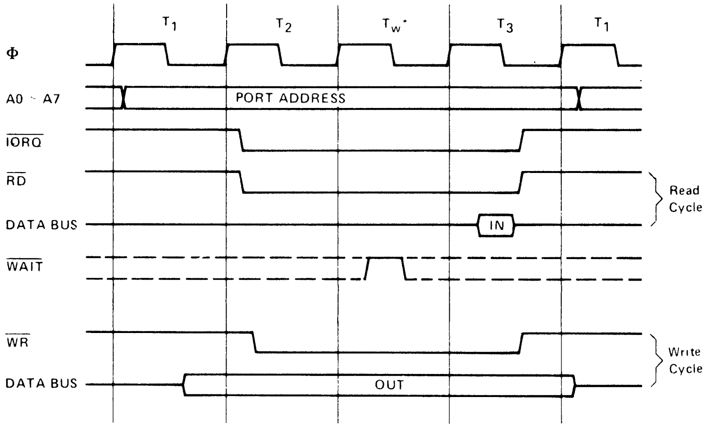
Figure 4.0-3 illustrates an I/O read or I/O write operation. Notice that during I/O operations a single wait state is automatically inserted. The reason for this is that during I/O operations, the time from when the IORQ signal goes active until the CPU must sample the WAIT line is very short and without this extra state sufficient time does not exist for an I/O port to decode its address and activate the WAIT line if a wait is required. Also, without this wait state it is difficult to design MOS I/O devices that can operate at full CPU speed. During this wait state time the WAIT request signal is sampled. During a read I/O operation, the RD line is used to enable the addressed port onto the data bus just as in the case of a memory read. For I/O write operations, the WR line is used as a clock to the I/O port, again with sufficient overlap timing automatically provided so that the rising edge may be used as a data clock.

Figure 4.0-3A illustrates how additional wait states may be added with the WAIT line. The operation is identical to that previously described.

BUS REQUEST/ACKNOWLEDGE CYCLE

Figure 4.0-4 illustrates the timing for a Bus Request/Acknowledge cycle. The BUSRQ signal is sampled by the CPU with the rising edge of the last clock period of any machine cycle. If the BUSRQ signal is active, the CPU will set its address, data and tri-state control signals to the high impedance state with the rising edge of the next clock pulse. At that time any external device can control the buses to transfer data between memory and I/O devices. (This is generally known as Direct Memory Access [DMA] using cycle stealing). The maximum time for the CPU to respond to a bus request is the length of a machine cycle and the external controller can maintain control of the bus for as many clock cycles as is desired. Note, however, that if very long DMA cycles are used, and dynamic memories are being used, the external controller must also perform the refresh function. This situation only occurs if very large blocks of data are transferred under DMA control. Also note that during a bus request cycle, the CPU cannot be interrupted by either a NMI or an INT signal.

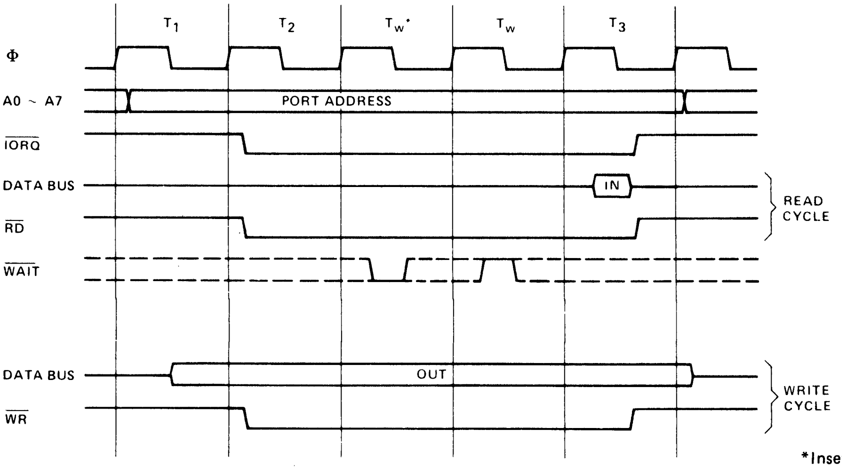
INPUT OR OUTPUT CYCLES



*Inserted by Z80 CPU

FIGURE 4.0-3

INPUT OR OUTPUT CYCLES WITH WAIT STATES



*Inserted by Z80 CPU

FIGURE 4.0-3A

BUS REQUEST/ACKNOWLEDGE CYCLE

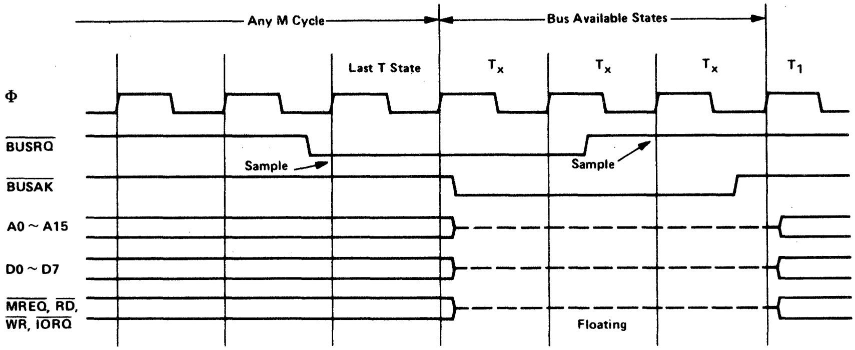
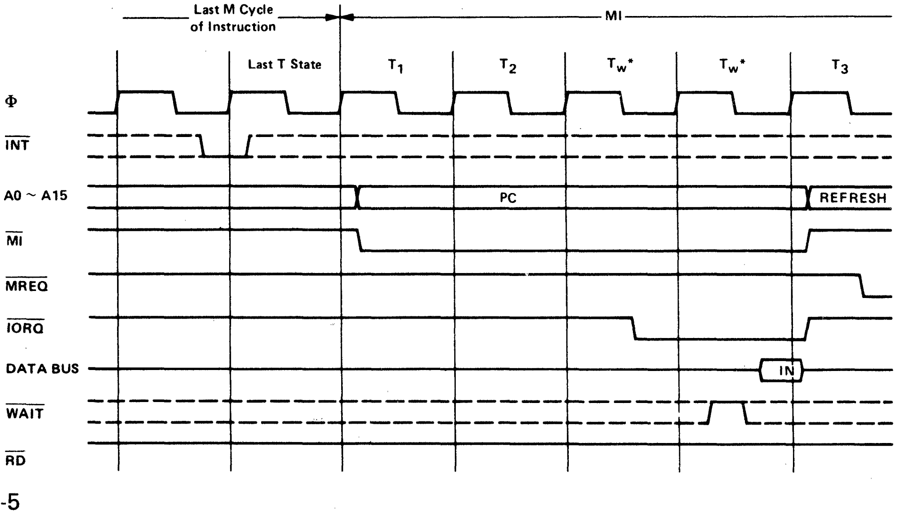


FIGURE 4.0-4

INTERRUPT REQUEST/ ACKNOWLEDGE CYCLE

Figure 4.0-5 illustrates the timing associated with an interrupt cycle. The interrupt signal (INT) is sampled by the CPU with the rising edge of the last clock at the end of any instruction. The signal will not be accepted if the internal CPU software controlled interrupt enable flip-flop is not set or if the BUSRQ signal is active. When the signal is accepted a special M1 cycle is generated. During this special M1 cycle the IORQ signal becomes active (instead of the normal MREQ) to indicate that the interrupting device can place an 8-bit vector on the data bus. Notice that two wait states are automatically added to this cycle. These states are added so that a ripple priority interrupt scheme can be easily implemented. The two wait states allow sufficient time for the ripple signals to stablilize and identify which I/O device must insert the response vector. Refer to section 8.0 for details on how the interrupt response vector is utilized by the CPU.

INTERRUPT REQUEST/ACKNOWLEDGE CYCLE

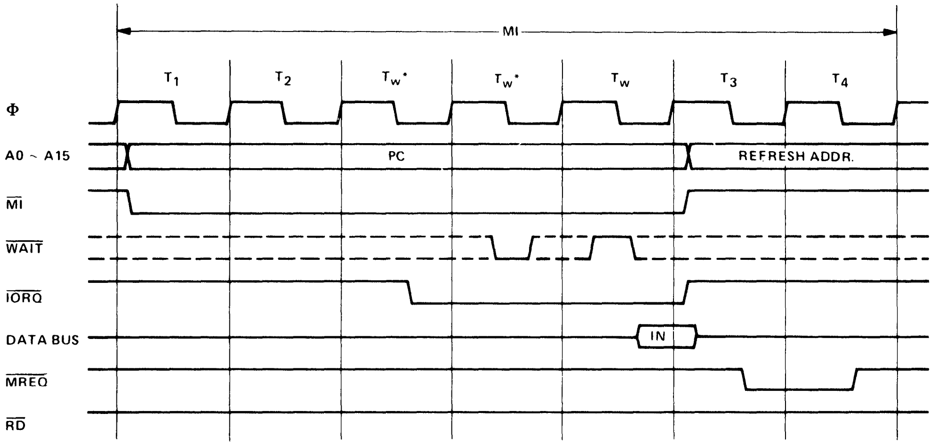


Mode 0 shown

FIGURE 4.0-5

Figure 4.0-5A illustrates how additional wait states can be added to the interrupt response cycle. Again the operation is identical to that previously described.

INTERRUPT REQUEST/ACKNOWLEDGE WITH WAIT STATES



Mode **0** shown

FIGURE 4.0-5A

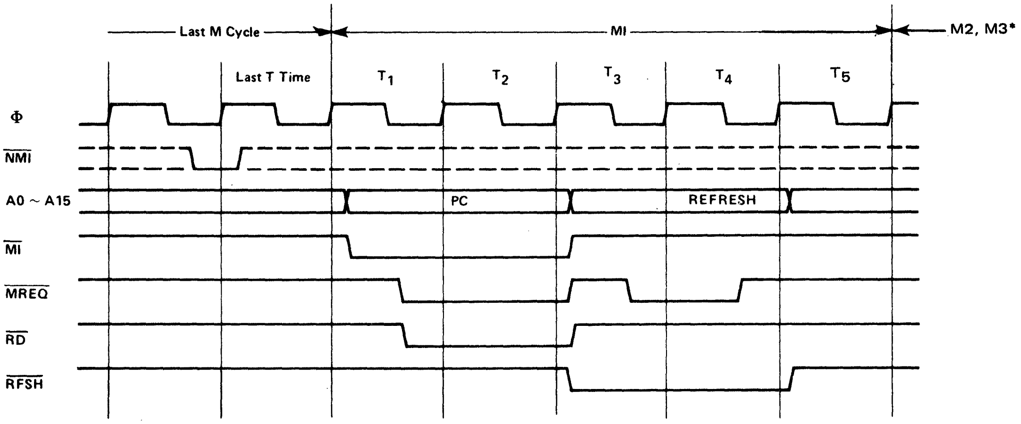
NON MASKABLE INTERRUPT RESPONSE

Figure 4.0-6 illustrates the request/acknowledge cycle for the non-maskable interrupt. A pulse on the $\overline{\text{NMI}}$ input sets an internal NMI latch which is tested by the CPU at the end of every instruction. This NMI latch is sampled at the same time as the interrupt line, but this line has priority over the normal interrupt and it can not be disabled under software control. Its usual function is to provide immediate response to important signals such as an impending power failure. The CPU response to a non maskable interrupt is similar to a normal memory read operation. The only difference being that the content of the data bus is ignored while the processor automatically stores the PC in the external stack and jumps to location 0066$_H$. The service routine for the non maskable interrupt must begin at this location if this interrupt is used.

HALT EXIT

Whenever a software halt instruction is executed the CPU begins executing NOP's until an interrupt is received (either a non-maskable or a maskable interrupt while the interrupt flip flop is enabled). The two interrupt lines are sampled with the rising clock edge during each T4 state as shown in Figure 4.0-7. If a non-maskable interrupt has been received or a maskable interrupt has been received and the interrupt enable flip-flop is set, then the halt state will be exited on the next rising clock edge. The following cycle will then be an interrupt acknowledge cycle corresponding to the type of interrupt that was received. If both are received at this time, then the non maskable one will be acknowledged since it was highest priority. The purpose of executing NOP instructions while in the halt state is to keep the memory refresh signals active. Each cycle in the halt state is a normal M1 (fetch) cycle except that the data received from the memory is ignored and a NOP instruction is forced internally to the CPU. The halt acknowledge signal is active during this time to indicate that the processor is in the halt state.

NON MASKABLE INTERRUPT REQUEST OPERATION



*M2 and M3 are stack write operations

FIGURE 4.0-6

HALT EXIT

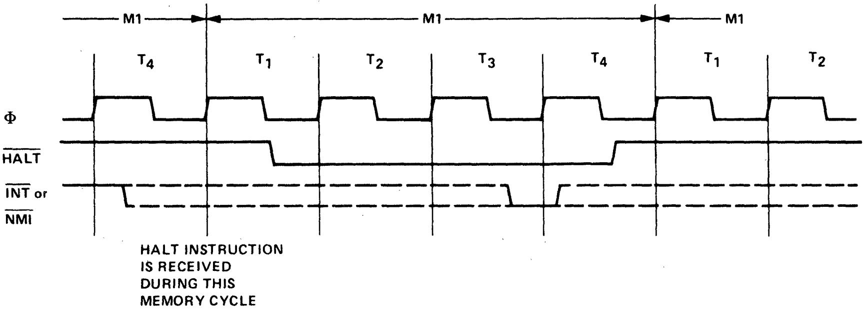


FIGURE 4.0-7

5.0 Z80-CPU INSTRUCTION SET

The Z80-CPU can execute 158 different instruction types including all 78 of the 8080A CPU. The instructions can be broken down into the following major groups:

- Load and Exchange
- Block Transfer and Search
- Arithmetic and Logical
- Rotate and Shift
- Bit Manipulation (set, reset, test)
- Jump, Call and Return
- Input/Output
- Basic CPU Control

5.1 INTRODUCTION TO INSTRUCTION TYPES

The load instructions move data internally between CPU registers or between CPU registers and external memory. All of these instructions must specify a source location from which the data is to be moved and a destination location. The source location is not altered by a load instruction. Examples of load group instructions include moves between any of the general purpose registers such as move the data to Register B from Register C. This group also includes load immediate to any CPU register or to any external memory location. Other types of load instructions allow transfer between CPU registers and memory locations. The exchange instructions can trade the contents of two registers.

A unique set of block transfer instructions is provided in the Z80. With a single instruction a block of memory of any size can be moved to any other location in memory. This set of block moves is extremely valuable when large strings of data must be processed. The Z80 block search instructions are also valuable for this type of processing. With a single instruction, a block of external memory of any desired length can be searched for any 8-bit character. Once the character is found the instruction automatically terminates. Both the block transfer and the block search instructions can be interrupted during their execution so as to not occupy the CPU for long periods of time.

The arithmetic and logical instructions operate on data stored in the accumulator and other general purpose CPU registers or external memory locations. The results of the operations are placed in the accumulator and the appropriate flags are set according to the result of the operation. An example of an arithmetic operation is adding the accumulator to the contents of an external memory location. The results of the addition are placed in the accumulator. This group also includes 16-bit addition and subtraction between 16-bit CPU registers.

The bit manipulation instructions allow any bit in the accumulator, any general purpose register or any external memory location to be set, reset or tested with a single instruction. For example, the most significant bit of register H can be reset. This group is especially useful in control applications and for controlling software flags in general purpose programming.

The jump, call and return instructions are used to transfer between various locations in the user's program. This group uses several different techniques for obtaining the new program counter address from specific external memory locations. A unique type of jump is the restart instruction. This instruction actually contains the new address as a part of the 8-bit OP code. This is possible since only 8 separate addresses located in page zero of the external memory may be specified. Program jumps may also be achieved by loading register HL, IX or IY directly into the PC, thus allowing the jump address to be a complex function of the routine being executed.

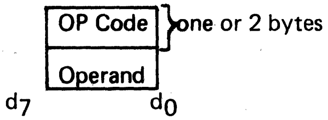
The input/output group of instructions in the Z80 allow for a wide range of transfers between external memory locations or the general purpose CPU registers, and the external I/O devices. In each case, the port number is provided on the lower 8 bits of the address bus during any I/O transaction. One instruction allows this port number to be specified by the second byte of the instruction while other Z80 instructions allow it to be specified as the content of the C register. One major advantage of using the C register as a pointer to the I/O device is that it allows different I/O ports to share common software driver routines. This is not possible when the address is part of the OP code if the routines are stored in ROM. Another feature of these input instructions is that they set the flag register automatically so that additional operations are not required to determine the state of the input data (for example its parity). The Z80-CPU includes single instructions that can move blocks or data (up to 256 bytes) automatically to or from any I/O port directly to any memory location. In conjunction with the dual set of general purpose registers, these instructions provide for fast I/O block transfer rates. The value of this I/O instruction set is demonstrated by the fact that the Z80-CPU can provide all required floppy disk formatting (i.e., the CPU provides the preamble, address, data and enables the CRC codes) on double density floppy disk drives on an interrupt driven basis.

Finally, the basic CPU control instructions allow various options and modes. This group includes instructions such as setting or resetting the interrupt enable flip flop or setting the mode of interrupt response.

5.2 ADDRESSING MODES

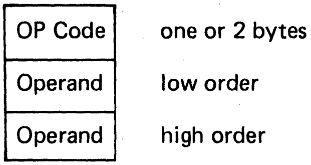
Most of the Z80 instructions operate on data stored in internal CPU registers, external memory or in the I/O ports. Addressing refers to how the address of this data is generated in each instruction. This section gives a brief summary of the types of addressing used in the Z80 while subsequent sections detail the type of addressing available for each instruction group.

Immediate. In this mode of addressing the byte following the OP code in memory contains the actual operand.



Examples of this type of instruction would be to load the accumulator with a constant, where the constant is the byte immediately following the OP code.

Immediate Extended. This mode is merely an extension of immediate addressing in that the two bytes following the op codes are the operand.



Examples of this type of instruction would be to load the HL register pair (16-bit register) with 16 bits (2 bytes) of data.

Modified Page Zero Addressing. The Z80 has a special single byte call instruction to any of 8 locations in page zero of memory. This instruction (which is referred to as a restart) sets the PC to an effective address in page zero. The value of this instruction is that it allows a single byte to specify a complete 16-bit address where commonly called subroutines are located, thus saving memory space.

OP Code	one byte
b_7 b_0	Effective address is $(00b_5b_4b_3000)$

Relative Addressing. Relative addressing uses one byte of data following the OP code to specify a displacement from the existing program to which a program jump can occur. This displacement is a signed two's complement number that is added to the address of the OP code of the following instruction.

OP Code	Jump relative (one byte OP code)
Operand	8-bit two's complement displacement added to Address (A+2)

The value of relative addressing is that it allows jumps to nearby locations while only requiring two bytes of memory space. For most programs, relative jumps are by far the most prevalent type of jump due to the proximity of related program segments. Thus, these instructions can significantly reduce memory space requirements. The signed displacement can range between +127 and -128 from A + 2. This allows for a total displacement of +129 to -126 from the jump relative OP code address. Another major advantage is that it allows for relocatable code.

Extended Addressing. Extended Addressing provides for two bytes (16 bits) of address to be included in the instruction. This data can be an address to which a program can jump or it can be an address where an operand is located.

OP Code	one or two bytes
Low Order Address or Low order operand	
High Order Address or High order operand	

Extended addressing is required for a program to jump from any location in memory to any other location, or load and store data in any memory location.

When extended addressing is used to specify the source or destination address of an operand, the notation (nn) will be used to indicate the content of memory at nn, where nn is the 16-bit address specified in the instruction. This means that the two bytes of address nn are used as a pointer to a memory location. The use of the parentheses always means that the value enclosed within them is used as a pointer to a memory location. For example, (1200) refers to the contents of memory at location 1200.

Indexed Addressing. In this type of addressing, the byte of data following the OP code contains a displacement which is added to one of the two index registers (the OP code specifies which index register is used) to form a pointer to memory. The contents of the index register are not altered by this operation.

OP Code	two byte OP code
OP Code	
Displacement	Operand added to index register to form a pointer to memory.

An example of an indexed instruction would be to load the contents of the memory location (Index Register + Displacement) into the accumulator. The displacement is a signed two's complement number. Indexed addressing greatly simplifies programs using tables of data since the index register can point to the start of any table. Two index registers are provided since very often operations require two or more tables. Indexed addressing also allows for relocatable code.

The two index registers in the Z80 are referred to as IX and IY. To indicate indexed addressing the notation:

(IX+d) or (IY+d)

is used. here d is the displacement specified after the OP code. The parentheses indicate that this value is used as a pointer to external memory.

Register Addressing. Many of the Z80 OP codes contain bits of information that specify which CPU register is to be used for an operation. An example of register addressing would be to load the data in register B into register C.

Implied Addressing. Implied addressing refers to operations where the OP code automatically implies one or more CPU registers as containing the operands. An example is the set of arithmetic operations where the accumulator is always implied to be the destination of the results.

Register Indirect Addressing. This type of addressing specifies a 16-bit CPU register pair (such as HL) to be used as a pointer to any location in memory. This type of instruction is very powerful and it is used in a wide range of applications.

OP Code
one or two bytes

An example of this type of instruction would be to load the accumulator with the data in the memory location pointed to by the HL register contents. Indexed addressing is actually a form of register indirect addressing except that a displacement is added with indexed addressing. Register indirect addressing allows for very powerful but simple to implement memory accesses. The block move and search commands in the Z80 are extensions of this type of addressing where automatic register incrementing, decrementing and comparing has been added. The notation for indicating register indirect addressing is to put parentheses around the name of the register that is to be used as the pointer. For example, the symbol

(HL)

specifies that the contents of the HL register are to be used as a pointer to a memory location. Often register indirect addressing is used to specify 16-bit operands. In this case, the register contents point to the lower order portion of the operand while the register contents are automatically incremented to obtain the upper portion of the operand.

Bit Addressing. The Z80 contains a large number of bit set, reset and test instructions. These instructions allow any memory location or CPU register to be specified for a bit operation through one of three previous addressing modes (register, register indirect and indexed) while three bits in the OP code specify which of the eight bits is to be manipulated.

ADDRESSING MODE COMBINATIONS

Many instructions include more than one operand (such as arithmetic instructions or loads). In these cases, two types of addressing may be employed. For example, load can use immediate addressing to specify the source and register indirect or indexed addressing to specify the source and register indirect or indexed addressing to specify the destination.

5.3 INSTRUCTION OP CODES

This section describes each of the Z80 instructions and provides tables listing the OP codes for every instruction. In each of these tables the shaded OP codes are identical to those offered in the 8080A CPU. Also shown is the assembly language mnemonic that is used for each instruction. All instruction OP codes are listed in hexadecimal notation. Single byte OP codes require two hex characters while double byte OP codes require four hex characters. The conversion from hex to binary is repeated here for convenience.

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

Z80 instruction mnemonics consist of an OP code and zero, one or two operands. Instructions in which the operand is implied have no operand. Instructions which have only one logical operand or those in which one operand is invariant (such as the Logical OR instruction) are represented by a one operand mnemonic. Instructions which may have two varying operands are represented by two operand mnemonics.

LOAD AND EXCHANGE

Table 5.3-1 defines the OP code for all of the 8-bit load instructions implemented in the Z80-CPU. Also shown in this table is the type of addressing used for each instruction. The source of the data is found on the top horizontal row while the destination is specified by the left hand column. For example, load register C from register B uses the OP code 48H. In all of the tables the OP code is specified in hexadecimal notation and the 48H (=0100 1000 binary) code is fetched by the CPU from the external memory during M1 time, decoded and then the register transfer is automatically performed by the CPU.

The assembly language mnemonic for this entire group is LD, followed by the destination followed by the source (LD DEST., SOURCE). Note that several combinations of addressing modes are possible. For example, the source may use register addressing and the destination may be register indirect, such as load the memory location pointed to by register HL with the contents of register D. The OP code for this operation would be 72. The mnemonic for this load instruction would be as follows: LD (HL), D

The parentheses around the HL means that the contents of HL are used as a pointer to a memory location. In all Z80 load instruction mnemonics the destination is always listed first, with the source following. The Z80 assembly language has been defined for ease of programming. Every instruction is self documenting and programs written in Z80 language are easy to maintain.

Note in Table 5.3-1 that some load OP codes that are available in the Z80 use two bytes. This is an efficient method of memory utilization since 8, 16, 24 or 32 bit instructions are implemented in the Z80. Thus often utilized instructions such as arithmetic or logical operations are only 8-bits which results in better memory utilization than is achieved with fixed instruction sizes such as 16-bits.

All load instructions using indexed addressing for either the source or destination location actually use three bytes of memory with the third byte being the displacement d. For example a load register E with the operand pointed to by IX with an offset of +8 would be written: LD E, (IX + 8)

The instruction sequence for this in memory would be:

Address A	DD	OP Code
A+1	5F	
A+2	08	Displacement operand

The two extended addressing instructions are also three byte instructions. For example the instruction to load the accumulator with the operand in memory location 6F32H would be written:

LD A, (6F 32H)

and its instruction sequence would be:

Address A	3A	OP Code
A+1	32	low order address
A+2	6F	high order address

Notice that the low order portion of the address is always the first operand.

The load immediate instructions for the general purpose 8-bit registers are two-byte instructions. The instruction load register H with the value 36H would be written:

LD H, 36H

and its sequence would be:

Address A	26	OP Code
A+1	36	Operand

Loading a memory location using indexed addressing for the destination and immediate addressing for the source requires four bytes. For example:

LD (IX - 15), 21H

would appear as:

Address A	DD	OP Code
A+1	36	
A+2	F1	displacement (-15 in signed two's complement)
A+3	21	operand to load

Notice that with any indexed addressing the displacement always follows directly after the OP code.

Table 5.3-2 specifies the 16-bit load operations. This table is very similar to the previous one. Notice that the extended addressing capability covers all register pairs. Also notice that register indirect operations specifying the stack pointer are the PUSH and POP instructions. The mnemonic for these instructions is "PUSH" and "POP". These differ from other 16-bit loads in that the stack pointer is automatically decremented and incremented as each byte is pushed onto or popped from the stack respectively. For example the instruction:

PUSH AF

is a single byte instruction with the OP code of F5H. When this instruction is executed the following sequence is generated:

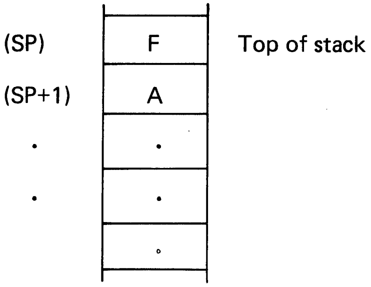
Decrement SP

LD (SP), A

Decrement SP

LD (SP), F

Thus the external stack now appears as follows:



8 BIT LOAD GROUP

			SOURCE															
			IMPLIED		REGISTER							REG INDIRECT			INDEXED		EXT. ADDR.	IMME.
			I	R	A	B	C	D	E	H	L	(HL)	(BC)	(DE)	(IX + d)	(IY + d)	(nn)	n
DESTINATION	REGISTER	A	ED 57	ED 5F	7F	78	79	7A	7B	7C	7D	7E	0A	1A	DD 7E d	FD 7E d	3A n n	3E n
		B			47	40	41	42	43	44	45	46			DD 46 d	FD 46 d		06 n
		C			4F	48	49	4A	4B	4C	4D	4E			DD 4E d	FD 4E d		0E n
		D			57	50	51	52	53	54	55	56			DD 56 d	FD 56 d		16 n
		E			5F	58	59	5A	5B	5C	5D	5E			DD 5E d	FD 5E d		1E n
		H			67	60	61	62	63	64	65	66			DD 66 d	FD 66 d		26 n
		L			6F	68	69	6A	6B	6C	6D	6E			DD 6E d	FD 6E d		2E n
	REG INDIRECT	(HL)			77	70	71	72	73	74	75							36 n
		(BC)			02													
		(DE)			12													
	INDEXED	(IX+d)			DD 77 d	DD 70 d	DD 71 d	DD 72 d	DD 73 d	DD 74 d	DD 75 d							DD 36 d n
		(IY+d)			FD 77 d	FD 70 d	FD 71 d	FD 72 d	FD 73 d	FD 74 d	FD 75 d							FD 36 d n
	EXT. ADDR	(nn)			32 n n													
	IMPLIED	I			ED 47													
		R			ED 4F													

TABLE 5.3-1

The POP instruction is the exact reverse of a PUSH. Notice that all PUSH and POP instructions utilize a 16-bit operand and the high order byte is always pushed first and popped last. That is a:

PUSH BC is PUSH B then C
PUSH DE is PUSH D then E
PUSH HL is PUSH H then L
POP HL is POP L then H

The instruction using extended immediate addressing for the source obviously requires 2 bytes of data following the OP code. For example:

LD DE, 0659H

will be:

Address A	11	OP Code
A+1	59	Low order operand to register E
A+2	06	High order operand to register D

In all extended immediate or extended addressing modes, the low order byte always appears first after the OP code.

Table 5.3-3 lists the 16-bit exchange instructions implemented in the Z80. OP code 08H allows the programmer to switch between the two pairs of accumulator flag registers while D9H allows the programmer to switch between the duplicate set of six general purpose registers. These OP codes are only one byte in length to absolutely minimize the time necessary to perform the exchange so that the duplicate banks can be used to effect very fast interrupt response times.

BLOCK TRANSFER AND SEARCH

Table 5.3-4 lists the extremely powerful block transfer instructions. All of these instructions operate with three registers.

HL points to the source location.
DE points to the destination location.
BC is a byte counter.

After the programmer has initialized these three registers, any of these four instructions may be used. The LDI (Load and Increment) instruction moves one byte from the location pointed to by HL to the location pointed to by DE. Register pairs HL and DE are then automatically incremented and are ready to point to the following locations. The byte counter (register pair BC) is also decremented at this time. This instruction is valuable when blocks of data must be moved but other types of processing are required between each move. The LDIR (Load, increment and repeat) instruction is an extension of the LDI instruction. The same load and increment operation is repeated until the byte counter reaches the count of zero. Thus, this single instruction can move any block of data from one location to any other.

Note that since 16-bit registers are used, the size of the block can be up to 64K bytes (1K = 1024) long and it can be moved from any location in memory to any other location. Furthermore the blocks can be overlapping since there are absolutely no constraints on the data that is used in the three register pair.

The LDD and LDDR instructions are very similar to the LDI and LDIR. The only difference is that register pairs HL and DE are decremented after every move so that a block transfer starts from the highest address of the designated block rather than the lowest.

16 BIT LOAD GROUP 'LD' 'PUSH' AND 'POP'

			SOURCE								
		REGISTER							IMM. EXT.	EXT. ADDR.	REG. INDIR.
		AF	BC	DE	HL	SP	IX	IY	nn	(nn)	(SP)
DESTINATION REGISTER	AF										F1
	BC								01 n n	ED 4B n n	C1
	DE								11 n n	ED 5B n n	D1
	HL								21 n n	2A n n	E1
	SP				F9		DD F9	FD F9	31 n n	ED 7B n n	
	IX								DD 21 n n	DD 2A n n	DD E1
	IY								FD 21 n n	FD 2A n n	FD E1
EXT. ADDR.	(nn)		ED 43 n n	ED 53 n n	22 n n	ED 73 n n	DD 22 n n	FD 22 n n			
PUSH INSTRUCTIONS → REG. IND.	(SP)	F5	C5	D5	E5		DD E5	FD E5			

↑ POP INSTRUCTIONS

NOTE: The Push & Pop Instructions adjust the SP after every execution

TABLE 5.3-2

EXCHANGES 'EX' AND 'EXX'

		IMPLIED ADDRESSING				
		AF'	BC', DE' & HL'	HL	IX	IY
IMPLIED	AF	08				
	BC, DE & HL		D9			
	DE			EB		
REG. INDIR.	(SP)			E3	DD E3	FD E3

TABLE 5.3-3

BLOCK TRANSFER GROUP

			SOURCE	
			REG. INDIR.	
			(HL)	
DESTINATION	REG. INDIR.	(DE)	ED A0	'LDI' – Load (DE)←(HL) Inc HL & DE, Dec BC
			ED A0	'LDIR,' – Load (DE)←(HL) Inc HL & DE, Dec BC, Repeat until BC = 0
			ED B0	'LDD' – Load (DE)←(HL) Dec HL & DE, Dec BC
			ED B8	'LDDR' – Load (DE)←(HL) Dec HL & DE, Dec BC, Repeat until BC = 0

Reg	HL	points to source
Reg	DE	points to destination
Reg	BC	is byte counter

Table 5.3-4

Table 5.3-5 specifies the OP codes for the four block search instructions. The first, CPI (compare and increment) compares the data in the accumulator, with the contents of the memory location pointed to by register HL. The result of the compare is stored in one of the flag bits (see section 6.0 for a detailed explanation of the flag operations) and the HL register pair is then incremented and the byte counter (register pair BC) is decremented.

The instruction CPIR is merely an extension of the CPI instruction in which the compare is repeated until either a match is found or the byte counter (register pair BC) becomes zero. Thus, this single instruction can search the entire memory for any 8-bit character.

The CPD (Compare and Decrement) and CPDR (Compare, Decrement and Repeat) are similar instructions, their only difference being that they decrement HL after every compare so that they search the memory in the opposite direction. (The search is started at the highest location in the memory block).

It should be emphasized again that these block transfer and compare instructions are extremely powerful in string manipulation applications.

ARITHMETIC AND LOGICAL

Table 5.3-6 lists all of the 8-bit arithmetic operations that can be performed with the accumulator, also listed are the increment (INC) and decrement (DEC) instructions. In all of these instructions, except INC and DEC, the specified 8-bit operation is performed between the data in the accumulator and the source data specified in the table. The result of the operation is placed in the accumulator with the exception of compare (CP) that leaves the accumulator unaffected. All of these operations affect the flag register as a result of the specified operation. (Section 6.0 provides all of the details on how the flags are affected by any instruction type). INC and DEC instructions specify a register or a memory location as both source and destination of the result. When the source operand is addressed using the index registers the displacement must follow directly. With immediate addressing the actual operand will follow directly. for example the instruction:

AND 07H

would appear as:

Address A	E6	OP Code
A+1	07	Operand

BLOCK SEARCH GROUP

SEARCH LOCATION	
REG. INDIR.	
(HL)	
ED A1	'CPI' Inc HL, Dec BC
ED B1	'CPIR', Inc HL, Dec BC repeat until BC = 0 or find match
ED A9	'CPD' Dec HL & BC
ED B9	'CPDR' Dec HL & BC Repeat until BC = 0 or find match

HL points to location in memory
to be compared with accumulator
contents
BC is byte counter

TABLE 5.3-5

Assuming that the accumulator contained the value F3H the result of 03H would be placed in the accumulator:

Acc before operation	1111 0011 = F3H
Operand	0000 0111 = 07H
Result to Acc	0000 0011 = 03H

The Add instruction (ADD) performs a binary add between the data in the source location and the data in the accumulator. The subtract (SUB) does a binary subtraction. When the add with carry is specified (ADC) or the subtract with carry (SBC), then the carry flag is also added or subtracted respectively. The flags and decimal adjust instruction (DAA) in the Z80 (fully described in section 6.0) allow arithmetic operations for:

multiprecision packed BCD numbers

multiprecision signed or unsigned binary numbers

multiprecision two's complement signed numbers

Other instructions in this group are logical and (AND), logical or (OR), exclusive or (XOR) and compare (CP).

There are five general purpose arithmetic instructions that operate on the accumulator or carry flag. These five are listed in Table 5.3-7. The decimal adjust instruction can adjust for subtraction as well as addition, thus making BCD arithmetic operations simple. Note that to allow for this operation the flag N is used. This flag is set if the last arithmetic operation was a subtract. The negate accumulator (NEG) instruction forms the two's complement of the number in the accumulator. Finally notice that a reset carry instruction is not included in the Z80 since this operation can be easily achieved through other instructions such as a logical AND of the accumulator with itself.

Table 5.3-8 lists all of the 16-bit arithmetic operations between 16-bit registers. There are five groups of instructions including add with carry and subtract with carry. ADC and SBC affect all of the flags. These two groups simplify address calculation operations or other 16-bit arithmetic operations.

8 BIT ARITHMETIC AND LOGIC

SOURCE

	REGISTER ADDRESSING							REG. INDIR.	INDEXED		IMMED.
	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)	n
'ADD'	87	80	81	82	83	84	85	86	DD 86 d	FD 86 d	C6 n
ADD w CARRY 'ADC'	8F	88	89	8A	8B	8C	8D	8E	DD 8E d	FD 8E d	CE n
SUBTRACT 'SUB'	97	90	91	92	93	94	95	96	DD 96 d	FD 96 d	D6 n
SUB w CARRY 'SBC'	9F	98	99	9A	9B	9C	9D	9E	DD 9E d	FD 9E d	DE n
'AND'	A7	A0	A1	A2	A3	A4	A5	A6	DD A6 d	FD A6 d	E6 n
'XOR'	AF	A8	A9	AA	AB	AC	AD	AE	DD AE d	FD AE d	EE n
'OR'	B7	B0	B1	B2	B3	B4	B5	B6	DD B6 d	FD B6 d	F6 n
COMPARE 'CP'	BF	B8	B9	BA	BB	BC	BD	BE	DD BE d	FD BE d	FE n
INCREMENT 'INC'	3C	04	0C	14	1C	24	2C	34	DD 34 d	FD 34 d	
DECREMENT 'DEC'	3D	05	0D	15	1D	25	2D	35	DD 35 d	FD 35 d	

TABLE 5.3-6

GENERAL PURPOSE AF OPERATIONS

Decimal Adjust Acc, 'DAA'	27
Complement Acc, 'CPL'	2F
Negate Acc, 'NEG' (2's complement)	ED 44
Complement Carry Flag, 'CCF'	3F
Set Carry Flag, 'SCF'	37

TABLE 5.3-7

16 BIT ARITHMETIC

		SOURCE					
DESTINATION		BC	DE	HL	SP	IX	IY
'ADD'	HL	09	19	29	39		
	IX	DD 09	DD 19		DD 39	DD 29	
	IY	FD 09	FD 19		FD 39		FD 29
ADD WITH CARRY AND SET FLAGS 'ADC'	HL	ED 4A	ED 5A	ED 6A	ED 7A		
SUB WITH CARRY AND SET FLAGS 'SBC'	HL	ED 42	ED 52	ED 62	ED 72		
INCREMENT 'INC.		03	13	23	33	DD 23	FD 23
DECREMENT 'DEC'		0B	1B	2B	3B	DD 2B	FD 2B

TABLE 5.3-8

ROTATE AND SHIFT

A major capability of the Z80 is its ability to rotate or shift data in the accumulator, any general purpose register, or any memory location. All of the rotate and shift OP codes are shown in Table 5.3-9. Also included in the Z80 are arithmetic and logical shift operations. These operations are useful in an extremely wide range of applications including integer multiplication and division. Two BCD digit rotate instructions (RRD and RLD) allow a digit in the accumulator to be rotated with the two digits in a memory location pointed to by register pair HL. (See Figure 5.3-9). These instructions allow for efficient BCD arithmetic.

BIT MANIPULATION

The ability to set, reset and test individual bits in a register or memory location is needed in almost every program. These bits may be flags in a general purpose software routine, indications of external control conditions or data packed into memory locations to make memory utilization more efficient.

The Z80 has the ability to set, reset or test any bit in the accumulator, any general purpose register or any memory location with a single instruction. Table 5.3-10 lists the 240 instructions that are available for this purpose. Register addressing can specify the accumulator or any general purpose register on which the operation is to be performed. Register indirect and indexed addressing are available to operate on external memory locations. Bit test operations set the zero flag (Z) if the tested bit is a zero. (Refer to section 6.0 for further explanation of flag operation).

JUMP, CALL AND RETURN

Figure 5.3-11 lists all of the jump, call and return instructions implemented in the Z80 CPU. A jump is a branch in a program where the program counter is loaded with the 16-bit value as specified by one of the three available addressing modes (Immediate Extended, Relative or Register Indirect). Notice that the jump group has several different conditions that can be specified to be met before the jump will be made. If these conditions are not met, the program merely continues with the next sequential instruction. The conditions are all dependent on the data in the flag register. (Refer to section 6.0 for details on the flag register). The immediate extended addressing is used to jump to any location in the memory. This instruction requires three bytes (two to specify the 16-bit address) with the low order address byte first followed by the high order address byte.

ROTATES AND SHIFTS

Source and Destination

TYPE OF ROTATE OR SHIFT	A	B	C	D	E	H	L	(HL)	(IX + d)	(IY + d)
'RLC'	CB 07	CB 00	CB 01	CB 02	CB 03	CB 04	CB 05	CB 06	DD CB d 06	FD CB d 06
'RRC'	CB 0F	CB 08	CB 09	CB 0A	CB 0B	CB 0C	CB 0D	CB 0E	DD CB d 0E	FD CB d 0E
'RL'	CB 17	CB 10	CB 11	CB 12	CB 13	CB 14	CB 15	CB 16	DD CB d 16	FD CB d 16
'RR'	CB 1F	CB 18	CB 19	CB 1A	CB 1B	CB 1C	CB 1D	CB 1E	DD CB d 1E	FD CB d 1E
'SLA'	CB 27	CB 20	CB 21	CB 22	CB 23	CB 24	CB 25	CB 26	DD CB d 26	FD CB d 26
'SRA'	CB 2F	CB 28	CB 29	CB 2A	CB 2B	CB 2C	CB 2D	CB 2E	DD CB d 2E	FD CB d 2E
'SRL'	CB 3F	CB 38	CB 39	CB 3A	CB 3B	CB 3C	CB 3D	CB 3E	DD CB d 3E	FD CB d 3E
'RLD'								ED 6F		
'RRD'								ED 67		

	A
RLCA	07
RRCA	0F
RLA	17
RRA	1F

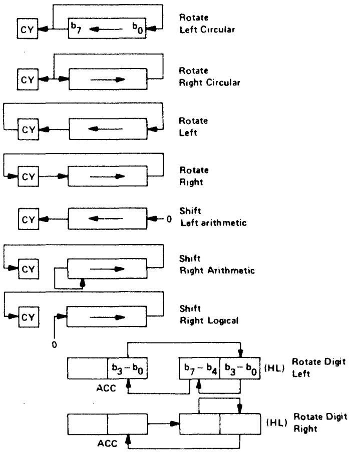


TABLE 5.3-9

For example an unconditional Jump to memory location 3E32H would be:

Address A	C3	OP Code
A+1	32	Low order address
A+2	3E	High order address

The relative jump instruction uses only two bytes, the second byte is a signed two's complement displacement from the existing PC. This displacement can be in the range of +129 to -126 and is measured from the address of the instruction OP code.

Three types of register indirect jumps are also included. These instructions are implemented by loading the register pair HL or one of the index registers IX or IY directly into the PC. This capability allows for program jumps to be a function of previous calculations.

A call is a special form of a jump where the address of the byte following the call instruction is pushed onto the stack before the jump is made. A return instruction is the reverse of a call because the data on the top of the stack is popped directly into the PC to form a jump address. The call and return instructions allow for simple subroutine and interrupt handling. Two special return instructions have been included in the Z80 family of components. The return from interrupt instruction (RETI) and the return from non-maskable interrupt (RETN) are treated in the CPU as an unconditional return identical to the OP code C9H. The difference is that (RETI) can be used at the end of an interrupt routine and all Z80 peripheral chips will recognize the execution of this instruction for proper control of nested priority interrupt handling. This instruction coupled with the Z80 peripheral devices implementation simplifies the normal return from nested interrupt. Without this feature the following software sequence would be necessary to inform the interrupting device that the interrupt routine is completed:

BIT MANIPULATION GROUP

		REGISTER ADDRESSING							REG. INDIR.	INDEXED	
	BIT	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)
TEST 'BIT'	0	CB 47	CB 40	Cd 41	CB 42	CB 43	CB 44	CB 45	CB 46	DD CB d 46	FD CB d 46
	1	CB 4F	CB 48	CB 49	CB 4A	CB 4B	CB 4C	CB 4D	CB 4E	DD CB d 4E	FD CB d 4E
	2	CB 57	CB 50	CB 51	CB 52	CB 53	CB 54	CB 55	CB 56	DD CB d 56	FD CB d 56
	3	CB 5F	CB 58	CB 59	CB 5A	CB 5B	CB 5C	CB 5D	CB 5E	DD CB d 5E	FD CB d 5E
	4	CB 67	CB 60	CB 61	CB 62	CB 63	CB 64	CB 65	CB 66	DD CB d 66	FD CB d 66
	5	CB 6F	CB 68	CB 69	CB 6A	CB 6B	CB 6C	CB 6D	CB 6E	DD CB d 6E	FD CB d 6E
	6	CB 77	CB 70	CB 71	CB 72	CB 73	CB 74	GB 75	CB 76	DD CB d 76	FD CB d 76
	7	CB 7F	CB 78	CB 79	CB 7A	CB 7B	CB 7C	CB 7D	CB 7E	DD CB d 7E	FD CB d 7E
RESET BIT 'RES'	0	CB 87	CB 80	CB 81	CB 82	CB 83	CB 84	CB 85	CB 86	DD CB d 86	FD CB d 86
	1	CB 8F	CB 88	CB 89	CB 8A	CB 8B	CB 8C	CB 8D	CB 8E	DD CB d 8E	FD CB d 8E
	2	CB 97	CB 90	CB 91	CB 92	CB 93	CB 94	CB 95	CB 96	DD CB d 96	FD CB d 96
	3	CB 9F	CB 98	CB 99	CB 9A	CB 9B	CB 9C	CB 9D	CB 9E	DD CB d 9E	FD CB d 9E
	4	CB A7	CB A0	CB A1	CB A2	CB A3	CB A4	CB A5	CB A6	DD CB d A6	FD CB d A6
	5	CB AF	CB A8	CB A9	CB AA	CB AB	CB AC	CB AD	CB AE	DD CB d AE	FD CB d AE
	6	CB B7	CB B0	CB B1	CB B2	CB B3	CB B4	CB B5	CB B6	DD CB d B6	FD CB d B6
	7	CB BF	CB B8	CB B9	CB BA	CB BB	CB BC	CB BD	CB BE	DD CB d BE	FD CB d BE
SET BIT 'SET'	0	CB C7	CB C0	CB C1	CB C2	CB C3	CB C4	CB C5	CB C6	DD CB d C6	FD CB d C6
	1	CB CF	CB C8	CB C9	CB CA	CB CB	CB CC	CB CD	CB CE	DD CB d CE	FD CB d CE
	2	CB D7	CB D0	CB D1	CB D2	CB D3	CB D4	CB D5	CB D6	DD CB d D6	FD CB d D6
	3	CB DF	CB D8	CB D9	CB DA	CB DB	CB DC	CB DD	CB DE	DD CB d DE	FD CB d DE
	4	CB E7	CB E0	CB E1	CB E2	CB E3	CB E4	CB E5	CB E6	DD CB d E6	FD CB d E6
	5	CB EF	CB E8	CB E9	CB EA	CB EB	CB EC	CB ED	CB EE	DD CB d EE	FD CB d EE
	6	CB F7	CB F0	CB F1	CB F2	CB F3	CB F4	CB F5	CB F6	DD CB d F6	FD CB d F6
	7	CB FF	CB F8	CB F9	CB FA	CB FB	CB FC	CB FD	CB FE	DD CB d FE	FD CB d FE

TABLE 5.3-10

Disable Interrupt	– prevent interrupt before routine is exited.
LD A, n OUT n, A	– notify peripheral that service routine is complete
Enable Interrupt	
Return	

This seven byte sequence can be replaced with the three byte EI RETI instruction sequence in the Z80. This is important since interrupt service time often must be minimized.

To facilitate program loop control the instruction DJNZ e can be used advantageously. This two byte, relative jump instruction decrements the B register and the jump occurs if the B register has not been decremented to zero. The relative displacement is expressed as a signed two's complement number. A simple example of its use might be:

Address	Instruction	Comments
N, N+1	LD B, 7	; set B register to count of 7
N + 2 to N + 9	(Perform a sequence of instructions)	; loop to be performed 7 times
N + 10, N + 11	DJNZ -10	; to jump from N + 12 to N + 2
N + 12	(Next Instruction)	

JUMP, CALL AND RETURN GROUP

			CONDITION									
			UN-COND.	CARRY	NON CARRY	ZERO	NON ZERO	PARITY EVEN	PARITY ODD	SIGN NEG	SIGN POS	REG B≠0
JUMP 'JP'	IMMED. EXT.	nn	C3 n n	DA n n	D2 n n	CA n n	C2 n n	EA n n	E2 n n	FA n n	F2 n n	
JUMP 'JR'	RELATIVE	PC+e	18 e-2	38 e-2	30 e-2	28 e-2	20 e-2					
JUMP 'JP'	REG. INDIR.	(HL)	E9									
JUMP 'JP'		(IX)	DD E9									
JUMP 'JP'		(IY)	FD E9									
'CALL'	IMMED. EXT.	nn	CD n n	DC n n	D4 n n	CC n n	C4 n n	EC n n	E4 n n	FC n n	F4 n n	
DECREMENT B, JUMP IF NON ZERO 'DJNZ'	RELATIVE	PC+e										10 e-2
RETURN 'RET'	REGISTER INDIR.	(SP) (SP+1)	C9	D8	D0	C8	C0	E8	E0	F8	F0	
RETURN FROM INT 'RETI'	REG. INDIR.	(SP) (SP+1)	ED 4D									
RETURN FROM NON MASKABLE INT 'RETN'	REG. INDIR.	(SP) (SP+1)	ED 45									

TABLE 5.3-11

NOTE–CERTAIN FLAGS HAVE MORE THAN ONE PURPOSE. REFER TO SECTION 6.0 FOR DETAILS

Table 5.3-12 lists the eight OP codes for the restart instruction. This instruction is a single byte call to any of the eight addresses listed. The simple mnemonic for these eight calls is also shown. The value of this instruction is that frequently used routines can be called with this instruction to minimize memory usage.

RESTART GROUP

		OP CODE	
CALL ADDRESS	0000_H	C7	'RST 0'
	0008_H	CF	'RST 8'
	0010_H	D7	'RST 16'
	0018_H	DF	'RST 24'
	0020_H	E7	'RST 32'
	0028_H	EF	'RST 40'
	0030_H	F7	'RST 48'
	0038_H	FF	'RST 56'

TABLE 5.3-12

INPUT/OUTPUT

The Z80 has an extensive set of Input and Output instructions as shown in table 5.3-13 and table 5.3-14. The addressing of the input or output device can be either absolute or register indirect, using the C register. Notice that in the register indirect addressing mode data can be transferred between the I/O devices and any of the internal registers. In addition eight block transfer instructions have been implemented. These instructions are similar to the memory block transfers except that they use register pair HL for a pointer to the memory source (output commands) or destination (input commands) while register B is used as a byte counter. Register C holds the address of the port for which the input or output command is desired. Since register B is eight bits in length, the I/O block transfer command handles up to 256 bytes.

In the instructions IN A, n and OUT n, A an I/O device address n appears in the lower half of the address bus (A_0-A_7) while the accumulator content is transferred in the upper half of the address bus. In all register indirect input output instructions, including block I/O transfers the content of register C is transferred to the lower half of the address bus (device address) while the content of register B is transferred to the upper half of the address bus.

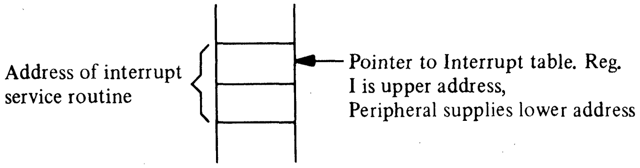
INPUT GROUP

				PORT ADDRESS		
				IMMED.	REG. INDIR.	
				n	(C)	
INPUT DESTINATION	INPUT 'IN'	REG ADDRESSING	A	DB	ED 78	
			B		ED 40	
			C		ED 48	
			D		ED 50	
			E		ED 58	
			H		ED 60	
			L		ED 68	
	'INI' – INPUT & Inc HL, Dec B	REG, INDIR	(HL)		ED A2	BLOCK INPUT COMMANDS
	'INIR'– INP, Inc HL, Dec B, REPEAT IF B≠0				ED B2	
	'IND'– INPUT & Dec HL, Dec B				ED AA	
	'INDR'– INPUT, Dec HL, Dec B, REPEAT IF B≠0				ED BA	

TABLE 5.3-13

CPU CONTROL GROUP

The final table, table 5.3-15 illustrates the six general purpose CPU control instructions. The NOP is a do-nothing instruction. The HALT instruction suspends CPU operation until a subsequent interrupt is received, while the DI and EI are used to lock out and enable interrupts. The three interrupt mode commands set the CPU into any of the three available interrupt response modes as follows. If mode zero is set the interrupting device can insert any instruction on the data bus and allow the CPU to execute it. Mode 1 is a simplified mode where the CPU automatically executes a restart (RST) to location 0038H so that no external hardware is required. (The old PC content is pushed onto the stack). Mode 2 is the most powerful in that it allows for an indirect call to any location in memory. With this mode the CPU forms a 16-bit memory address where the upper 8-bits are the content of register I and the lower 8-bits are supplied by the interrupting device. This address points to the first of two sequential bytes in a table where the address of the service routine is located. The CPU automatically obtains the starting address and performs a CALL to this address.


Address of interrupt service routine
Pointer to Interrupt table. Reg. I is upper address,
Peripheral supplies lower address

OUTPUT GROUP

SOURCE

			REGISTER							REG. IND.
			A	B	C	D	E	H	L	(HL)
'OUT'	IMMED.	n	D3 n							
	REG. IND.	(C)	ED 79	ED 41	ED 49	ED 51	ED 59	ED 61	ED 69	
'OUTI' – OUTPUT Inc HL, Dec b	REG. IND.	(C)								ED A3
'OTIR' – OUTPUT, Inc HL, Dec B, REPEAT IF B≠0	REG. IND.	(C)								ED B3
'OUTD' – OUTPUT Dec HL & B	REG. IND.	(C)								ED AB
'OTDR' – OUTPUT, Dec HL & B, REPEAT IF B≠0	REG. IND.	(C)								ED BB

BLOCK OUTPUT COMMANDS

PORT DESTINATION ADDRESS

TABLE 5.3-14

MISCELLANEOUS CPU CONTROL

'NOP'	00	
'HALT'	76	
DISABLE INT '(DI)'	F3	
ENABLE INT '(EI)'	FB	
SET INT MODE 0 'IM0'	ED 46	8080A MODE
SET INT MODE 1 'IM1'	ED 56	CALL TO LOCATION 0038_H
SET INT MODE 2 'IM2'	ED 5E	INDIRECT CALL USING REGISTER I AND 8 BITS FROM INTERRUPTING DEVICE AS A POINTER.

TABLE 5.3-15

6.0 FLAGS

Each of the two Z80-CPU Flag registers contains six bits of information which are set or reset by various CPU operations. Four of these bits are testable; that is, they are used as conditions for jump, call or return instructions. For example a jump may be desired only if a specific bit in the flag register is set. The four testable flag bits are:

1) Carry Flag (C) – This flag is the carry from the highest order bit of the accumulator. For example, the carry flag will be set during an add instruction where a carry from the highest bit of the accumulator is generated. This flag is also set if a borrow is generated during a subtraction instruction. The shift and rotate instructions also affect this bit.

2) Zero Flag (Z) – This flag is set if the result of the operation loaded a zero into the accumulator. Otherwise it is reset.

3) Sign Flag(S) – This flag is intended to be used with signed numbers and it is set if the result of the operation was negative. Since bit 7 (MSB) represents the sign of the number (A negative number has a 1 in bit 7), this flag stores the state of bit 7 in the accumulator.

4) Parity/Overflow Flag(P/V) – This dual purpose flag indicates the parity of the result in the accumulator when logical operations are performed (such as AND A, B) and it represents overflow when signed two's complement arithmetic operations are performed. The Z80 overflow flag indicates that the two's complement number in the accumulator is in error since it has exceeded the maximum possible (+127) or is less than the minimum possible (–128) number that can be represented two's complement notation. For example consider adding:

```
+120 =   0111 1000
+105 =   0110 1001
 C = 0   1110 0001 = -95 (wrong) Overflow has occurred;
```

Here the result is incorrect. Overflow has occurred and yet there is no carry to indicate an error. For this case the overflow flag would be set. Also consider the addition of two negative numbers:

```
 -5  =   1111 1011
-16  =   1111 0000
 C = 1   1110 1011 = -21 correct
```

Notice that the answer is correct but the carry is set so that this flag can not be used as an overflow indicator. In this case the overflow would not be set.

For logical operations (AND, OR, XOR) this flag is set if the parity of the result is even and it is reset if it is odd.

There are also two non-testable bits in the flag register. Both of these are used for BCD arithmetic. They are:

1) Half carry(H) – This is the BCD carry or borrow result from the least significant four bits of operation. When using the DAA (Decimal Adjust Instruction) this flag is used to correct the result of a previous packed decimal add or subtract.

2) Add/Subtract Flag (N) – Since the agorithim for correcting BCD operations is different for addition or subtraction, this flag is used to specify what type of instruction was executed last so that the DAA operation will be correct for either addition or subtraction.

The Flag register can be accessed by the programmer and its format is as follows:

D7							DØ
S	Z	X	H	X	P/V	N	C

X means flag is indeterminate.

Table 6.0-1 lists how each flag bit is affected by various CPU instructions. In this table a '•' indicates that the instruction does not change the flag, an 'X' means that the flag goes to an indeterminate state, an '0' means that it is reset, a '1' means that it is set and the symbol ↕ indicates that it is set or reset according to the previous discussion. Note that any instruction not appearing in this table does not affect any of the flags.

Table 6.0-1 includes a few special cases that must be described for clarity. Notice that the block search instruction sets the Z flag if the last compare operation indicated a match between the source and the accumulator data. Also, the parity flag is set if the byte counter (register pair BC) is not equal to zero. This same use of the parity flag is made with the block move instructions. Another special case is during block input or output instructions, here the Z flag is used to indicate the state of register B which is used as a byte counter. Notice that when the I/O block transfer is complete, the zero flag will be reset to a zero (i.e. B=0) while in the case of a block move command the parity flag is reset when the operation is complete. A final case is when the refresh or I register is loaded into the accumulator, the interrupt enable flip flop is loaded into the parity flag so that the complete state of the CPU can be saved at any time.

SUMMARY OF FLAG OPERATION

	D7							D0	
Instruction	S	Z		H		P/V	N	C	Comments
ADD A,s; ADC A,s	↕	↕	X	↕	X	V	0	↕	8-bit add or add with carry
SUB,s; SBCA,s; CP,s; NEG	↕	↕	X	↕	X	V	1	↕	8-bit subtract, subtract with carry, compare and negate accumulator
AND s	↕	↕	X	1	X	P	0	0	Logical operations
OR s; XOR s	↕	↕	X	0	X	P	0	0	
INC s	↕	↕	X	↕	X	V	0	•	8-bit increment
DEC s	↕	↕	X	↕	X	V	1	•	8-bit decrement
ADD DD, SS	•	•	X	X	X	•	0	↕	16-bit add
ADC HL, SS	↕	↕	X	X	X	V	0	↕	16-bit add with carry
SBC HL, SS	↕	↕	X	X	X	V	1	↕	16-bit subtract with carry
RLA; RLCA; RRA; RRCA	•	•	X	0	X	•	0	↕	Rotate accumulator
RL s; RLC s; RR s; RRC s; SLA s; SRA s; SRL s	↕	↕	X	0	X	P	0	↕	Rotate and shift locations
RLD; RRD	↕	↕	X	0	X	P	0	•	Rotate digit left and right
DAA	↕	↕	X	↕	X	P	•	↕	Decimal adjust accumulator
CPL	•	•	X	1	X	•	1	•	Complement accumulator
SCF	•	•	X	0	X	•	0	1	Set carry
CCF	•	•	X	X	X	•	0	↕	Complement carry
IN r, (C)	↕	↕	X	0	X	P	0	•	Input register indirect
INI; IND; OUTI; OUTD	X	↕	X	X	X	X	1	X	Block input and output
INIR; INDR; OTIR; OTDR	X	1	X	X	X	X	1	X	Z = 0 if B ≠ 0 otherwise Z = 1
LDI; LDD	X	X	X	0	X	↕	0	•	Block transfer instructions
LDIR; LDDR	X	X	X	0	X	0	0	•	P/V = 1 if BC ≠ 0, otherwise P/V = 0
CPI; CPIR; CPD; CPDR	↕	↕	X	↕	X	↕	1	•	Block search instructions Z = 1 if A = (HL), otherwise Z = 0 P/V = 1 if BC ≠ 0, otherwise P/V = 0
LD A, I; LD A, R	↕	↕	X	0	X	IFF	0	•	The content of the interrupt enable flip-flop (IFF) is copied into the P/V flag
BIT b, s	X	↕	X	1	X	X	0	•	The state of bit b of location s is copied into the Z flag

The following notation is used in this table:

SYMBOL	**OPERATION**
C	**Carry/link flag. C=1 if the operation produced a carry from the MSB of the operand or result.**
Z	**Zero flag. Z=1 if the result of the operation is zero.**
S	**Sign flag. S=1 if the MSB of the result is one.**
P/V	**Parity or overflow flag. Parity (P) and overflow (V) share the same flag. Logical operations affect this flag with the parity of the result while arithmetic operations affect this flag with the overflow of the result. If P/V holds parity, P/V=1 if the result of the operation is even, P/V=0 if result is odd. If P/V holds overflow, P/V=1 if the result of the operation produced an overflow.**
H	**Half-carry flag. H=1 if the add or subtract operation produced a carry into or borrow from bit 4 of the accumulator.**
N	**Add/Subtract flag. N=1 if the previous operation was a subtract.** **H and N flags are used in conjunction with the decimal adjust instruction (DAA) to properly correct the result into packed BCD format following addition or subtraction using operands with packed BCD format.** **The flag is affected according to the result of the operation.**
●	**The flag is unchanged by the operation.**
0	**The flag is reset by the operation.**
1	**The flag is set by the operation.**
X	**The flag is a "don't care".**
V	**P/V flag affected according to the overflow result of the operation.**
P	**P/V flag affected according to the parity result of the operation.**
r	**Any one of the CPU registers A, B, C, D, E, H, L.**
s	**Any 8-bit location for all the addressing modes allowed for the particular instruction.**
ss	**Any 16-bit location for all the addressing modes allowed for that instruction.**
ii	**Any one of the two index registers IX or IY.**
R	**Refresh counter.**
n	**8-bit value in range <0, 255>**
nn	**16-bit value in range <0, 65535>**

TABLE 6.0-1

7.0 SUMMARY OF OP CODES AND EXECUTION TIMES

The following section gives a summary of the Z80 instruction set. The instructions are logically arranged into groups as shown on Tables 7.0-1 through 7.0-11. Each table shows the assembly language mnemonic OP code, the actual OP code, the symbolic operation, the content of the flag register following the execution of each instruction, the number of bytes required for each instruction as well as the number of memory cycles and the total number of T states (external clock periods) required for the fetching and execution of each instruction. Care has been taken to make each table self-explanatory without requiring any cross reference with the text or other tables.

8-BIT LOAD GROUP

	Symbolic	Flags								Op-Code		No. of	No. of M	No. of T		
Mnemonic	Operation	S	Z		H		P/V	N	C	76 543 210	Hex	Bytes	Cycles	States	Comments	
LD r, s	r ← s	●	●	X	●	X	●	●	●	01 r s		1	1	4	r, s	Reg.
LD r, n	r ← n	●	●	X	●	X	●	●	●	00 r 110		2	2	7	000	B
										← n →					001	C
LD r, (HL)	r ← (HL)	●	●	X	●	X	●	●	●	01 r 110		1	2	7	010	D
LD r, (IX+d)	r ← (IX+d)	●	●	X	●	X	●	●	●	11 011 101	DD	3	5	19	011	E
										01 r 110					100	H
										← d →					101	L
LD r, (IY+d)	r ← (IY+d)	●	●	X	●	X	●	●	●	11 111 101	FD	3	5	19	111	A
										01 r 110						
										← d →						
LD (HL), r	(HL) ← r	●	●	X	●	X	●	●	●	01 110 r		1	2	7		
LD (IX+d), r	(IX+d) ← r	●	●	X	●	X	●	●	●	11 011 101	DD	3	5	19		
										01 110 r						
										← d →						
LD (IY+d), r	(IY+d) ← r	●	●	X	●	X	●	●	●	11 111 101	FD	3	5	19		
										01 110 r						
										← d →						
LD (HL), n	(HL) ← n	●	●	X	●	X	●	●	●	00 110 110	36	2	3	10		
										← n →						
LD (IX+d), n	(IX+d) ← n	●	●	X	●	X	●	●	●	11 011 101	DD	4	5	19		
										00 110 110	36					
										← d →						
										← n →						
LD (IY+d), n	(IY+d) ← n	●	●	X	●	X	●	●	●	11 111 101	FD	4	5	19		
										00 110 110	36					
										← d →						
										← n →						
LD A, (BC)	A ← (BC)	●	●	X	●	X	●	●	●	00 001 010	0A	1	2	7		
LD A, (DE)	A ← (DE)	●	●	X	●	X	●	●	●	00 011 010	1A	1	2	7		
LD A, (nn)	A ← (nn)	●	●	X	●	X	●	●	●	00 111 010	3A	3	4	13		
										← n →						
										← n →						
LD (BC), A	(BC) ← A	●	●	X	●	X	●	●	●	00 000 010	02	1	2	7		
LD (DE), A	(DE) ← A	●	●	X	●	X	●	●	●	00 010 010	12	1	2	7		
LD (nn), A	(nn) ← A	●	●	X	●	X	●	●	●	00 110 010	32	3	4	13		
										← n →						
										← n →						
LD A, I	A ← I	↕	↕	X	0	X	IFF	0	●	11 101 101	ED	2	2	9		
										01 010 111	57					
LD A, R	A ← R	↕	↕	X	0	X	IFF	0	●	11 101 101	ED	2	2	9		
										01 011 111	5F					
LD I, A	I ← A	●	●	X	●	X	●	●	●	11 101 101	ED	2	2	9		
										01 000 111	47					
LD R, A	R ← A	●	●	X	●	X	●	●	●	11 101 101	ED	2	2	9		
										01 001 111	4F					

Notes: r, s means any of the registers A, B, C, D, E, H, L
IFF the content of the interrupt enable flip-flop (IFF) is copied into the P/V flag

Flag Notation: ● = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation.

Table 7.0-1

16-BIT LOAD GROUP

	Symbolic	Flags								Op-Code		No. of	No. of M	No. of T	
Mnemonic	Operation	S	Z		H		P/V	N	C	76 543 210	Hex	Bytes	Cycles	States	Comments
LD dd, nn	dd ← nn	•	•	X	•	X	•	•	•	00 dd0 001		3	3	10	dd Pair
										← n →					00 BC
										← n →					01 DE
LD IX, nn	IX ← nn	•	•	X	•	X	•	•	•	11 011 101	DD	4	4	14	10 HL
										00 100 001	21				11 SP
										← n →					
										← n →					
LD IY, nn	IY ← nn	•	•	X	•	X	•	•	•	11 111 101	FD	4	4	14	
										00 100 001	21				
										← n →					
										← n →					
LD HL, (nn)	H ← (nn+1)	•	•	X	•	X	•	•	•	00 101 010	2A	3	5	16	
	L ← (nn)									← n →					
										← n →					
LD dd, (nn)	dd$_H$ ← (nn+1)	•	•	X	•	X	•	•	•	11 101 101	ED	4	6	20	
	dd$_L$ ← (nn)									01 dd1 011					
										← n →					
										← n →					
LD IX, (nn)	IX$_H$ ← (nn+1)	•	•	X	•	X	•	•	•	11 011 101	DD	4	6	20	
	IX$_L$ ← (nn)									00 101 010	2A				
										← n →					
										← n →					
LD IY, (nn)	IY$_H$ ← (nn+1)	•	•	X	•	X	•	•	•	11 111 101	FD	4	6	20	
	IY$_L$ ← (nn)									00 101 010	2A				
										← n →					
										← n →					
LD (nn), HL	(nn+1) ← H	•	•	X	•	X	•	•	•	00 100 010	22	3	5	16	
	(nn) ← L									← n →					
										← n →					
LD (nn), dd	(nn+1) ← dd$_H$	•	•	X	•	X	•	•	•	11 101 101	ED	4	6	20	
	(nn) ← dd$_L$									01 dd0 011					
										← n →					
										← n →					
LD (nn), IX	(nn+1) ← IX$_H$	•	•	X	•	X	•	•	•	11 011 101	DD	4	6	20	
	(nn) ← IX$_L$									00 100 010	22				
										← n →					
										← n →					
LD (nn), IY	(nn+1) ← IY$_H$	•	•	X	•	X	•	•	•	11 111 101	FD	4	6	20	
	(nn) ← IY$_L$									00 100 010	22				
										← n →					
										← n →					
LD SP, HL	SP ← HL	•	•	X	•	X	•	•	•	11 111 001	F9	1	1	6	
LD SP, IX	SP ← IX	•	•	X	•	X	•	•	•	11 011 101	DD	2	2	10	
										11 111 001	F9				
LD SP, IY	SP ← IY	•	•	X	•	X	•	•	•	11 111 101	FD	2	2	10	
										11 111 001	F9				qq Pair
PUSH qq	(SP-2) ← qq$_L$	•	•	X	•	X	•	•	•	11 qq0 101		1	3	11	00 BC
	(SP-1) ← qq$_H$														01 DE
PUSH IX	(SP-2) ← IX$_L$	•	•	X	•	X	•	•	•	11 011 101	DD	2	4	15	10 HL
	(SP-1) ← IX$_H$									11 100 101	E5				11 AF
PUSH IY	(SP-2) ← IY$_L$	•	•	X	•	X	•	•	•	11 111 101	FD	2	4	15	
	(SP-1) ← IY$_H$									11 100 101	E5				
POP qq	qq$_H$ ← (SP+1)	•	•	X	•	X	•	•	•	11 qq0 001		1	3	10	
	qq$_L$ ← (SP)														
POP IX	IX$_H$ ← (SP+1)	•	•	X	•	X	•	•	•	11 011 101	DD	2	4	14	
	IX$_L$ ← (SP)									11 100 001	E1				
POP IY	IY$_H$ ← (SP+1)	•	•	X	•	X	•	•	•	11 111 101	FD	2	4	14	
	IY$_L$ ← (SP)									11 100 001	E1				

Notes: dd is any of the register pairs BC, DE, HL, SP
qq is any of the register pairs AF, BC, DE, HL
(PAIR)$_H$, (PAIR)$_L$ refer to high order and low order eight bits of the register pair respectively.
e.g. BC$_L$ = C, AF$_H$ = A

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ flag is affected according to the result of the operation.

Table 7.0-2

EXCHANGE GROUP AND BLOCK TRANSFER AND SEARCH GROUP

	Symbolic	Flags								Op-Code		No. of	No. of M	No. of T	
Mnemonic	Operation	S	Z		H		P/V	N	C	76 543 210	Hex	Bytes	Cycles	States	Comments
EX DE, HL	DE↔HL	•	•	X	•	X	•	•	•	11 101 011	EB	1	1	4	
EX AF, AF'	AF↔AF'	•	•	X	•	X	•	•	•	00 001 000	08	1	1	4	
EXX	(BC↔BC'	•	•	X	•	X	•	•	•	11 011 001	D9	1	1	4	Register bank and
	DE↔DE'														auxiliary register
	HL↔HL')														bank exchange
EX (SP), HL	H ↔(SP+1)	•	•	X	•	X	•	•	•	11 100 011	E3	1	5	19	
	L ↔(SP)														
EX (SP), IX	IX_H ↔(SP+1)	•	•	X	•	X	•	•	•	11 011 101	DD	2	6	23	
	IX_L ↔(SP)									11 100 011	E3				
EX (SP), IY	IY_H ↔(SP+1)	•	•	X	•	X	•	•	•	11 111 101	FD	2	6	23	
	IY_L ↔(SP)									11 100 011	E3				
LDI	(DE)←(HL)	•	•	X	0	X	① ↕	0	•	11 101 101	ED	2	4	16	Load (HL) into
	DE ← DE+1									10 100 000	A0				(DE), increment the
	HL ← HL+1														pointers and
	BC ← BC-1														decrement the byte
															counter (BC)
LDIR	(DE)←(HL)	•	•	X	0	X	0	0	•	11 101 101	ED	2	5	21	If BC ≠ 0
	DE ← DE+1									10 110 000	B0	2	4	16	If BC = 0
	HL ← HL+1														
	BC ← BC-1														
	Repeat until														
	BC = 0														
LDD	(DE)←(HL)	•	•	X	0	X	① ↕	0	•	11 101 101	ED	2	4	16	
	DE ← DE-1									10 101 000	A8				
	HL ← HL-1														
	BC ← BC-1														
LDDR	(DE)←(HL)	•	•	X	0	X	0	0	•	11 101 101	ED	2	5	21	If BC ≠ 0
	DE ← DE-1									10 111 000	B8	2	4	16	If BC = 0
	HL ← HL-1														
	BC ← BC-1														
	Repeat until														
	BC = 0														
CPI	A – (HL)	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101	ED	2	4	16	
	HL ← HL+1									10 100 001	A1				
	BC ← BC-1														
CPIR	A – (HL)	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101	ED	2	5	21	If BC ≠ 0 and A ≠ (HL)
	HL ← HL+1									10 110 001	B1	2	4	16	If BC = 0 or A = (HL)
	BC ← BC-1														
	Repeat until														
	A = (HL) or														
	BC = 0														
CPD	A – (HL)	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101	ED	2	4	16	
	HL ← HL-1									10 101 001	A9				
	BC ← BC-1														
CPDR	A – (HL)	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101	ED	2	5	21	If BC ≠ 0 and A ≠ (HL)
	HL ← HL-1									10 111 001	B9	2	4	16	If BC = 0 or A = (HL)
	BC ← BC-1														
	Repeat until														
	A = (HL) or														
	BC = 0														

Notes: ① P/V flag is 0 if the result of BC-1 = 0, otherwise P/V = 1
② Z flag is 1 if A = (HL), otherwise Z = 0.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation.

Table 7.0-3

8-BIT ARITHMETIC AND LOGICAL GROUP

	Symbolic	Flags								Op-Code		No. of	No.of M	No.of T	
Mnemonic	Operation	S	Z		H		P/V	N	C	76 543 210	Hex	Bytes	Cycles	States	Comments
ADD A, r	A ← A + r	↕	↕	X	↕	X	V	0	↕	10 $\boxed{000}$ r		1	1	4	r Reg.
ADD A, n	A ← A + n	↕	↕	X	↕	X	V	0	↕	11 $\boxed{000}$ 110		2	2	7	000 B
										← n →					001 C
															010 D
ADD A, (HL)	A ← A+(HL)	↕	↕	X	↕	X	V	0	↕	10 $\boxed{000}$ 110		1	2	7	011 E
ADD A, (IX+d)	A←A+(IX+d)	↕	↕	X	↕	X	V	0	↕	11 011 101	DD	3	5	19	100 H
										10 $\boxed{000}$ 110					101 L
										← d →					111 A
ADD A, (IY+d)	A←A+(IY+d)	↕	↕	X	↕	X	V	0	↕	11 111 101	FD	3	5	19	
										10 $\boxed{000}$ 110					
										← d →					
ADC A, s	A←A+s+CY	↕	↕	X	↕	X	V	0	↕	$\boxed{001}$					s is any of r, n,
SUB s	A←A - s	↕	↕	X	↕	X	V	1	↕	$\boxed{010}$					(HL), (IX+d),
SBC A, s	A←A - s - CY	↕	↕	X	↕	X	V	1	↕	$\boxed{011}$					(IY+d) as shown for
AND s	A←A ∧ s	↕	↕	X	1	X	P	0	0	$\boxed{100}$					ADD instruction.
OR s	A←A ∨ s	↕	↕	X	0	X	P	0	0	$\boxed{110}$					The indicated bits
XOR s	A←A ⊕ s	↕	↕	X	0	X	P	0	0	$\boxed{101}$					replace the $\boxed{000}$ in
CP s	A - s	↕	↕	X	↕	X	V	1	↕	$\boxed{111}$					the ADD set above.
INC r	r ← r + 1	↕	↕	X	↕	X	V	0	•	00 r $\boxed{100}$		1	1	4	
INC (HL)	(HL)←(HL)+1	↕	↕	X	↕	X	V	0	•	00 110 $\boxed{100}$		1	3	11	
INC (IX+d)	(IX+d) ←	↕	↕	X	↕	X	V	0	•	11 011 101	DD	3	6	23	
	(IX+d)+1									00 110 $\boxed{100}$					
										← d →					
INC (IY+d)	(IY+d) ←	↕	↕	X	↕	X	V	0	•	11 111 101	FD	3	6	23	
	(IY+d)+1									00 110 $\boxed{100}$					
										← d →					
DEC s	s ← s - 1	↕	↕	X	↕	X	V	1	•	$\boxed{101}$					s is any of r, (HL),
															(IX+d), (IY+d) as
															shown for INC.
															DEC same format
															and states as INC.
															Replace $\boxed{100}$ with
															$\boxed{101}$ in OP Code.

Notes: The V symbol in the P/V flag column indicates that the P/V flag contains the overflow of the result of the operation. Similarly the P symbol indicates parity. V = 1 means overflow, V = 0 means not overflow, P = 1 means parity of the result is even, P = 0 means parity of the result is odd.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown.
↕ = flag is affected according to the result of the operation.

Table 7.0-4

GENERAL PURPOSE ARITHMETIC AND CPU CONTROL GROUPS

	Symbolic	Flags								Op-Code		No. of	No.of M	No.of T	
Mnemonic	Operation	S	Z		H		P/V	N	C	76 543 210	Hex	Bytes	Cycles	States	Comments
DAA	Converts acc. content into packed BCD following add or subtract with packed BCD operands	↕	↕	X	↕	X	P	•	↕	00 100 111	27	1	1	4	Decimal adjust accumulator
CPL	A ← $\overline{A}$	•	•	X	1	X	•	1	•	00 101 111	2F	1	1	4	Complement accumulator (One's complement)
NEG	A ← $\overline{A}$+1	↕	↕	X	↕	X	V	1	↕	11 101 101 01 000 100	ED 44	2	2	8	Negate acc. (two's complement)
CCF	CY ← $\overline{CY}$	•	•	X	X	X	•	0	↕	00 111 111	3F	1	1	4	Complement carry flag
SCF	CY ← 1	•	•	X	0	X	•	0	1	00 110 111	37	1	1	4	Set carry flag
NOP	No operation	•	•	X	•	X	•	•	•	00 000 000	00	1	1	4	
HALT	CPU halted	•	•	X	•	X	•	•	•	01 110 110	76	1	1	4	
DI*	IFF ← 0	•	•	X	•	X	•	•	•	11 110 011	F3	1	1	4	
EI*	IFF ← 1	•	•	X	•	X	•	•	•	11 111 011	FB	1	1	4	
IM 0	Set interrupt mode 0	•	•	X	•	X	•	•	•	11 101 101 01 000 110	ED 46	2	2	8	
IM 1	Set interrupt mode 1	•	•	X	•	X	•	•	•	11 101 101 01 010 110	ED 56	2	2	8	
IM 2	Set interrupt mode 2	•	•	X	•	X	•	•	•	11 101 101 01 011 110	ED 5E	2	2	8	

Notes: IFF indicates the interrupt enable flip-flop
CY indicates the carry flip-flop.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation.

*Interrupts are not sampled at the end of EI or DI

Table 7.0-5

16-BIT ARITHMETIC GROUP

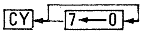
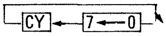
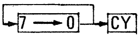
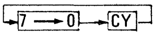
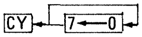
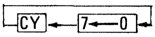
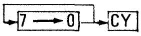
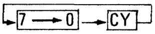
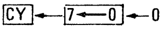
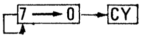
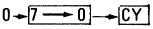
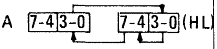
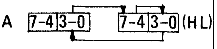
	Symbolic	Flags								Op-Code		No. of	No.of M	No.of T		
Mnemonic	Operation	S	Z		H		P/V	N	C	76 543 210	Hex	Bytes	Cycles	States	Comments	
ADD HL, ss	HL ← HL+ss	•	•	X	X	X	•	0	↕	00 ss1 001		1	3	11	ss	Reg.
															00	BC
ADC HL, ss	HL ← HL+ss+CY	↕	↕	X	X	X	V	0	↕	11 101 101	ED	2	4	15	01	DE
										01 ss1 010					10	HL
															11	SP
SBC HL, ss	HL ← HL-ss-CY	↕	↕	X	X	X	V	1	↕	11 101 101	ED	2	4	15		
										01 ss0 010						
ADD IX, pp	IX ← IX + pp	•	•	X	X	X	•	0	↕	11 011 101	DD	2	4	15	pp	Reg.
										00 pp1 001					00	BC
															01	DE
															10	IX
															11	SP
ADD IY, rr	IY ← IY + rr	•	•	X	X	X	•	0	↕	11 111 101	FD	2	4	15	rr	Reg.
										00 rr1 001					00	BC
															01	DE
															10	IY
															11	SP
INC ss	ss ← ss + 1	•	•	X	•	X	•	•	•	00 ss0 011		1	1	6		
INC IX	IX ← IX + 1	•	•	X	•	X	•	•	•	11 011 101	DD	2	2	10		
										00 100 011	23					
INC IY	IY ← IY + 1	•	•	X	•	X	•	•	•	11 111 101	FD	2	2	10		
										00 100 011	23					
DEC ss	ss ← ss - 1	•	•	X	•	X	•	•	•	00 ss1 011		1	1	6		
DEC IX	IX ← IX - 1	•	•	X	•	X	•	•	•	11 011 101	DD	2	2	10		
										00 101 011	2B					
DEC IY	IY ← IY - 1	•	•	X	•	X	•	•	•	11 111 101	FD	2	2	10		
										00 101 011	2B					

Notes: ss is any of the register pairs BC, DE, HL, SP
pp is any of the register pairs BC, DE, IX, SP
rr is any of the register pairs BC, DE, IY, SP.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown.
↕ = flag is affected according to the result of the operation.

Table 7.0-6

ROTATE AND SHIFT GROUP

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Op-Code 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
RLCA	 A	•	•	X	0	X	•	0	↕	00 000 111	07	1	1	4	Rotate left circular accumulator
RLA	 A	•	•	X	0	X	•	0	↕	00 010 111	17	1	1	4	Rotate left accumulator
RRCA	 A	•	•	X	0	X	•	0	↕	00 001 111	0F	1	1	4	Rotate right circular accumulator
RRA	 A	•	•	X	0	X	•	0	↕	00 011 111	1F	1	1	4	Rotate right accumulator
RLC r		↕	↕	X	0	X	P	0	↕	11 001 011 00 $\boxed{000}$ r	CB	2	2	8	Rotate left circular register r
RLC (HL)		↕	↕	X	0	X	P	0	↕	11 001 011 00 $\boxed{000}$ 110	CB	2	4	15	r Reg. 000 B 001 C
RLC (IX+d)	 r,(HL),(IX+d),(IY+d)	↕	↕	X	0	X	P	0	↕	11 011 101 11 001 011 ← d → 00 $\boxed{000}$ 110	DD CB	4	6	23	010 D 011 E 100 H 101 L 111 A
RLC (IY+d)		↕	↕	X	0	X	P	0	↕	11 111 101 11 001 011 ← d → 00 $\boxed{000}$ 110	FD CB	4	6	23	
RL s	 s ≡ r,(HL),(IX+d),(IY+d)	↕	↕	X	0	X	P	0	↕	$\boxed{010}$					Instruction format and states are as shown for RLC's. To form new Op-Code replace $\boxed{000}$ of RLC's with shown code
RRC s	 s ≡ r,(HL),(IX+d),(IY+d)	↕	↕	X	0	X	P	0	↕	$\boxed{001}$					
RR s	 s ≡ r,(HL),(IX+d),(IY+d)	↕	↕	X	0	X	P	0	↕	$\boxed{011}$					
SLA s	 s ≡ r,(HL),(IX+d),(IY+d)	↕	↕	X	0	X	P	0	↕	$\boxed{100}$					
SRA s	 s ≡ r,(HL),(IX+d),(IY+d)	↕	↕	X	0	X	P	0	↕	$\boxed{101}$					
SRL s	 s ≡ r,(HL),(IX+d),(IY+d)	↕	↕	X	0	X	P	0	↕	$\boxed{111}$					
RLD		↕	↕	X	0	X	P	0	•	11 101 101 01 101 111	ED 6F	2	5	18	Rotate digit left and right between the accumulator and location (HL). The content of the upper half of the accumulator is unaffected
RRD		↕	↕	X	0	X	P	0	•	11 101 101 01 100 111	ED 67	2	5	18	

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation.

Table 7.0-7

BIT SET, RESET AND TEST GROUP

	Symbolic	Flags								Op-Code		No. of	No.of M	No.of T		
Mnemonic	Operation	S	Z		H		P/V	N	C	76 543 210	Hex	Bytes	Cycles	States	Comments	
BIT b, r	Z ← $\overline{r}_b$	X	↕	X	1	X	X	0	•	11 001 011	CB	2	2	8	r	Reg.
										01 b r					000	B
BIT b, (HL)	Z ← $\overline{(HL)}_b$	X	↕	X	1	X	X	0	•	11 001 011	CB	2	3	12	001	C
										01 b 110					010	D
BIT b, (IX+d)$_b$	Z ← $\overline{(IX+d)}_b$	X	↕	X	1	X	X	0	•	11 011 101	DD	4	5	20	011	E
										11 001 011	CB				100	H
										← d →					101	L
										01 b 110					111	A
															b	Bit Tested
BIT b, (IY+d)$_b$	Z ← $\overline{(IY+d)}_b$	X	↕	X	1	X	X	0	•	11 111 101	FD	4	5	20	000	0
										11 001 011	CB				001	1
										← d →					010	2
										01 b 110					011	3
															100	4
															101	5
															110	6
															111	7
SET b, r	r_b ← 1	•	•	X	•	X	•	•	•	11 001 011	CB	2	2	8		
										$\boxed{11}$ b r						
SET b, (HL)	$(HL)_b$ ← 1	•	•	X	•	X	•	•	•	11 001 011	CB	2	4	15		
										$\boxed{11}$ b 110						
SET b, (IX+d)	$(IX+d)_b$ ← 1	•	•	X	•	X	•	•	•	11 011 101	DD	4	6	23		
										11 001 011	CB					
										← d →						
										$\boxed{11}$ b 110						
SET b, (IY+d)	$(IY+d)_b$ ← 1	•	•	X	•	X	•	•	•	11 111 101	FD	4	6	23		
										11 001 011	CB					
										← d →						
										$\boxed{11}$ b 110						
RES b, s	s_b ← 0 s≡r, (HL), (IX+d), (IY+d)	•	•	X	•	X	•	•	•	$\boxed{10}$					To form new Op-Code replace $\boxed{11}$ of SET b, s with $\boxed{10}$. Flags and time states for SET instruction	

Notes: The notation s_b indicates bit b (0 to 7) or location s.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation.

Table 7.0-8

JUMP GROUP

	Symbolic	Flags								Op-Code		No. of	No. of M	No. of T	
Mnemonic	Operation	S	Z		H		P/V	N	C	76 543 210	Hex	Bytes	Cycles	States	Comments
JP nn	PC ← nn	•	•	X	•	X	•	•	•	11 000 011	C3	3	3	10	
										← n →					
										← n →					cc Condition
JP cc, nn	If condition cc	•	•	X	•	X	•	•	•	11 cc 010		3	3	10	000 NZ non zero
	is true PC ← nn,									← n →					001 Z zero
	otherwise									← n →					010 NC non carry
	continue														011 C carry
															100 PO parity odd
															101 PE parity even
															110 P sign positive
JR e	PC ← PC + e	•	•	X	•	X	•	•	•	00 011 000	18	2	3	12	111 M sign negative
										← e-2 →					
JR C, e	If C = 0,	•	•	X	•	X	•	•	•	00 111 000	38	2	2	7	If condition not met
	continue									← e-2 →					
	If C = 1,											2	3	12	If condition is met
	PC ← PC+e														
JR NC, e	If C = 1,	•	•	X	•	X	•	•	•	00 110 000	30	2	2	7	If condition not met
	continue									← e-2 →					
	If C = 0,											2	3	12	If condition is met
	PC ← PC+e														
JR Z, e	If Z = 0	•	•	X	•	X	•	•	•	00 101 000	28	2	2	7	If condition not met
	continue									← e-2 →					
	If Z = 1,											2	3	12	If condition is met
	PC ← PC+e														
JR NZ, e	If Z = 1,	•	•	X	•	X	•	•	•	00 100 000	20	2	2	7	If condition not met
	continue									← e-2 →					
	If Z = 0,											2	3	12	If condition is met
	PC ← PC+e														
JP (HL)	PC ← HL	•	•	X	•	X	•	•	•	11 101 001	E9	1	1	4	
JP (IX)	PC ← IX	•	•	X	•	X	•	•	•	11 011 101	DD	2	2	8	
										11 101 001	E9				
JP (IY)	PC ← IY	•	•	X	•	X	•	•	•	11 111 101	FD	2	2	8	
										11 101 001	E9				
DJNZ, e	B ← B-1	•	•	X	•	X	•	•	•	00 010 000	10	2	2	8	If B = 0
	If B = 0,									← e-2 →					
	continue														
	If B ≠ 0,											2	3	13	If B ≠ 0
	PC ← PC+e														

Notes: e represents the extension in the relative addressing mode.

e is a signed two's complement number in the range <-126, 129>

e-2 in the op-code provides an effective address of pc+e as PC is incremented by 2 prior to the addition of e.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation.

Table 7.0-9

CALL AND RETURN GROUP

	Symbolic	Flags								Op-Code		No. of	No.of M	No.of T	
Mnemonic	**Operation**	**S**	**Z**		**H**		**P/V**	**N**	**C**	**76 543 210**	**Hex**	**Bytes**	**Cycles**	**States**	**Comments**
CALL nn	(SP-1) ← PC_H	•	•	X	•	X	•	•	•	11 001 101	CD	3	5	17	
	(SP-2) ← PC_L									← n →					
	PC ← nn									← n →					
CALL cc, nn	If condition	•	•	X	•	X	•	•	•	11 cc 100		3	3	10	If cc is false
	cc is false									← n →					
	continue,									← n →		3	5	17	If cc is true
	otherwise														
	same as														
	CALL nn														
RET	PC_L ← (SP)	•	•	X	•	X	•	•	•	11 001 001	C9	1	3	10	
	PC_H ← (SP+1)														
RET cc	If condition	•	•	X	•	X	•	•	•	11 cc 000		1	1	5	If cc is false
	cc is false														
	continue,											1	3	11	If cc is true
	otherwise														
	same as														
	RET														
RETI	Return from	•	•	X	•	X	•	•	•	11 101 101	ED	2	4	14	
	interrupt									01 001 101	4D				
RETN[1]	Return from	•	•	X	•	X	•	•	•	11 101 101	ED	2	4	14	
	non maskable									01 000 101	45				
	interrupt														
RST p	(SP-1) ← PC_H	•	•	X	•	X	•	•	•	11 t 111		1	3	11	
	(SP-2) ← PC_L														
	PC_H ← 0														
	PC_L ← p														

cc	Condition	
000	NZ	non zero
001	Z	zero
010	NC	non carry
011	C	carry
100	PO	parity odd
101	PE	parity even
110	P	sign positive
111	M	sign negative

t	p
000	00H
001	08H
010	10H
011	18H
100	20H
101	28H
110	30H
111	38H

[1] RETN loads IFF_2 ← IFF_1

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation.

Table 7.0-10

INPUT AND OUTPUT GROUP

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Op-Code 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
IN A, (n)	A ← (n)	•	•	X	•	X	•	•	•	11 011 011 ← n →	DB	2	3	11	n to $A_0 \sim A_7$ Acc to $A_8 \sim A_{15}$
IN r, (C)	r ← (C) if r = 110 only the flags will be affected	↕	↕	X	↕	X	P	0	•	11 101 101 01 r 000	ED	2	3	12	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INI	(HL) ← (C) B ← B - 1 HL ← HL + 1	X	① ↕	X	X	X	X	1	X	11 101 101 10 100 010	ED A2	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INIR	(HL) ← (C) B ← B - 1 HL ← HL + 1 Repeat until B = 0	X	1	X	X	X	X	1	X	11 101 101 10 110 010	ED B2	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
IND	(HL) ← (C) B ← B - 1 HL ← HL - 1	X	① ↕	X	X	X	X	1	X	11 101 101 10 101 010	ED AA	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INDR	(HL) ← (C) B ← B - 1 HL ← HL - 1 Repeat until B = 0	X	1	X	X	X	X	1	X	11 101 101 10 111 010	ED BA	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUT (n), A	(n) ← A	•	•	X	•	X	•	•	•	11 010 011	D3	2	3	11	n to $A_0 \sim A_7$ Acc to $A_8 \sim A_{15}$
OUT (C), r	(C) ← r	•	•	X	•	X	•	•	•	11 101 101 01 r 001	ED	2	3	12	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUTI	B ← B - 1 (C) ← (HL) HL ← HL + 1	X	① ↕	X	X	X	X	1	X	11 101 101 10 100 011	ED A3	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OTIR	B ← B - 1 (C) ← (HL) HL ← HL + 1 Repeat until B = 0	X	1	X	X	X	X	1	X	11 101 101 10 110 011	ED B3	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUTD	(C) ← (HL) B ← B - 1 HL ← HL - 1	X	① ↕	X	X	X	X	1	X	11 101 101 10 101 011	ED AB	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OTDR	(C) ← (HL) B ← B - 1 HL ← HL - 1 Repeat until B = 0	X	1	X	X	X	X	1	X	11 101 101 10 111 011	ED BB	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$

Notes: ① If the result of B - 1 is zero the Z flag is set, otherwise it is reset.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation.

Table 7.0-11

8.0 INTERRUPT RESPONSE

The prupose of an interrupt is to allow peripheral devices to suspend CPU operation in an orderly manner and force the CPU to start a peripheral service routine. Usually this service routine is involved with the exchange of data, or status and control information, between the CPU and the peripheral. Once the service routine is completed, the CPU returns to the operation from which it was interrupted.

INTERRUPT ENABLE – DISABLE

The Z80-CPU has two interrupt inputs, a software maskable interrupt and a non-maskable interrupt. The non-maskable interrupt ($\overline{\text{NMI}}$) can not be disabled by the programmer and it will be accepted whenever a peripheral device requests it. This interrupt is generally reserved for very important functions that must be serviced whenever they occur, such as an impending power failure. The maskable interrupt ($\overline{\text{INT}}$) can be selectively enabled or disabled by the programmer. This allows the programmer to disable the interrupt during periods where his program has timing constraints that do not allow it to be interrupted. In the Z80–CPU there is an enable flip flop (called IFF) that is set or reset by the programmer using the Enable Interrupt (EI) and Disable Interrupt (DI) instructions. When the IFF is reset, an interrupt can not be accepted by the CPU.

III
Z80 FAMILY
TECHNICAL
MANUALS

Actually, for purposes that will be subsequently explained, there are two enable flip flops, called IFF$_1$ and IFF$_2$.

$\boxed{\text{IFF}_1}$	$\boxed{\text{IFF}_2}$
Actually disables interrupts from being accepted.	Temporary storage location for IFF$_1$.

The state of IFF$_1$ is used to actually inhibit interrupts while IFF$_2$ is used as a temporary storage location for IFF$_1$. The purpose of storing the IFF$_1$ will be subsequently explained.

A reset to the CPU will force both IFF$_1$ and IFF$_2$ to the reset state so that interrupts are disabled. They can then be enabled by an EI instruction at any time by the programmer. When an EI instruction is executed, any pending interrupt request will not be accepted until after the instruction following EI has been executed. This single instruction delay is necessary for cases when the following instruction is a return instruction and interrupts must not be allowed until the return has been completed. The EI instructions sets both IFF$_1$ and IFF$_2$ to the enable state. When an interrupt is accepted by the CPU, both IFF$_1$ and IFF$_2$ are automatically reset, inhibiting further interrupts until the programmer wishes to issue a new EI instruction. Note that for all of the previous cases, IFF$_1$ and IFF$_2$ are always equal.

The purpose of IFF$_2$ is to save the status of IFF$_1$ when a non-maskable interrupt occurs. When a non-maskable interrupt is accepted, IFF$_1$ is reset to prevent further interrupts until reenabled by the programmer. Thus, after a non-maskable interrupt has been accepted maskable interrupts are disabled but the previous state of IFF$_1$ has been saved so that the complete state of the CPU just prior to the non-maskable interrupt can be restored at any time. When a Load Register A with Register I (LD A, I) instruction or a Load Register A with Register R (LD A, R) instruction is executed, the state of IFF$_2$ is copied into the parity flag where it can be tested or stored.

A second method of restoring the status of IFF$_1$ is thru the execution of a Return From Non-Maskable Interrupt (RETN) instruction. Since this instruction indicates that the non maskable interrupt service routine is complete, the contents of IFF$_2$ are now copied back into IFF$_1$, so that the status of IFF$_1$ just prior to the acceptance of the non-maskable interrupt will be restored automatically.

Figure 8.0-1 is a summary of the effect of different instructions on the two enable flip flops.

INTERRUPT ENABLE/DISABLE FLIP FLOPS

Action	IFF_1	IFF_2	
CPU Reset	0	0	
DI	0	0	
EI	1	1	
LD A, I	•	•	IFF_2 → Parity flag
LD A, R	•	•	IFF_2 → Parity flag
Accept NMI	0	•	
RETN	IFF_2	•	$IFF_2 \rightarrow IFF_1$
Accept INT	0	0	
RETI	•	•	

"•" indicates no change

FIGURE 8.0-1

CPU RESPONSE

Non-Maskable

A non-maskable interrupt will be accepted at all times by the CPU. When this occurs, the CPU ignores the next instruction that it fetches and instead does a restart to location 0066H. Thus, it behaves exactly as if it had received a restart instruction but, it is to a location that is not one of the 8 software restart locations. A restart is merely a call to a specific address in page 0 memory.

Maskable

The CPU can be programmed to respond to the maskable interrupt in any one of three possible modes.

Mode 0

This mode is identical to the 8080A interrupt response mode. With this mode, the interrupting device can place any instruction on the data bus and the CPU will execute it. Thus, the interrupting device provides the next instruction to be executed instead of the memory. Often this will be a restart instruction since the interrupting device only need supply a single byte instruction. Alternatively, any other instruction such as a 3 byte call to any location in memory could be executed by issuing a restart to the 3 byte op code.

The number of clock cycles necessary to execute this instruction is 2 more than the normal number for the instruction. This occurs since the CPU automatically adds 2 wait states to an interrupt response cycle to allow sufficient time to implement an external daisy chain for priority control. Section 4.0 illustrates the detailed timing for an interrupt response. After the application of RESET the CPU will automatically enter interrupt Mode 0.

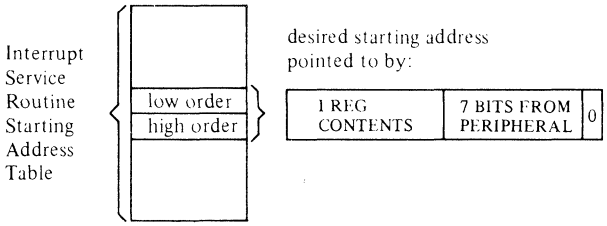
Mode 1

When this mode has been selected by the programmer, the CPU will respond to an interrupt by executing a restart to location 0038H. Thus the response is identical to that for a non maskable interrupt except that the call location is 0038H instead of 0066H. Another difference is that the number of cycles required to complete the restart instruction is 2 more than normal due to the two added wait states.

Mode 2

This mode is the most powerful interrupt response mode. With a single 8-bit byte from the user an indirect call can be made to any memory location.

With this mode the programmer maintains a table of 16 bit starting addresses for every interrupt service routine. This table may be located anywhere in memory. When an interrupt is accepted, a 16 bit pointer must be formed to obtain the desired interrupt service routine starting address from the table. The upper 8 bits of this pointer is formed from the contents of the I register. The I register must have been previously loaded with the desired value by the programmer, i.e. LD I, A. Note that a CPU reset clears the I register so that it is initialized to zero. The lower eight bits of the pointer must be supplied by the interrupting device. Actually, only 7 bits are required from the interrupting device as the least bit must be a zero. This is required since the pointer is used to get two adjacent bytes to from a complete 16 bit service routine starting address and the addresses must always start in even locations.



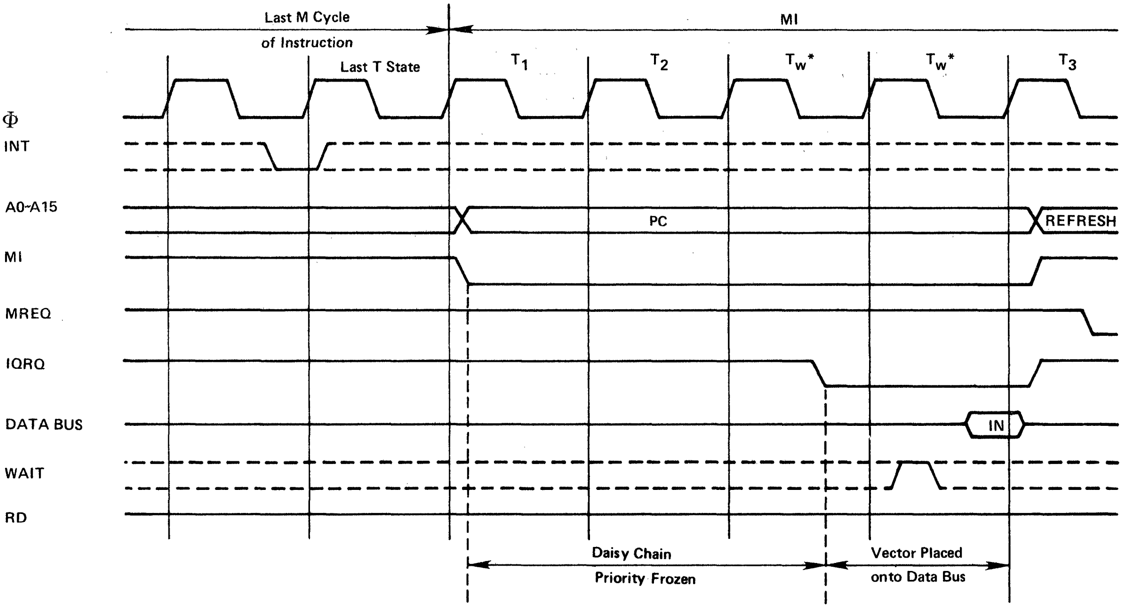
The first byte in the table is the least significant (low order) portion of the address. The programmer must obviously fill this table in with the desired addresses before any interrupts are to be accepted.

Note that this table can be changed at any time by the programmer (if it is stored in Read/ Write Memory) to allow different peripherals to be serviced by different service routines.

Once the interrupting device supplies the lower portion of the pointer, the CPU automat-cally pushes the program counter onto the stack, obtains the starting address from the table and does a jump to this address. This mode of response requires 19 clock periods to complete (7 to fetch the lower 8 bits from the interrupting device, 6 to save the program counter, and 6 to obtain the jump address.)

Note that the Z80 peripheral devices all include a daisy chain priority interrupt structure that automatically supplies the programmed vector to the CPU during interrupt acknowledge. Refer to the Z80-PIO, Z80-SIO and Z80-CTC manuals for details.

INTERRUPT REQUEST/ACKNOWLEDGE CYCLE



Z80 INTERRUPT ACKNOWLEDGE SUMMARY

1) PERIPHERAL DEVICE REQUESTS INTERRUPT. Any device requesting and interrupt can pull the wired-or line $\overline{\text{INT}}$ low.

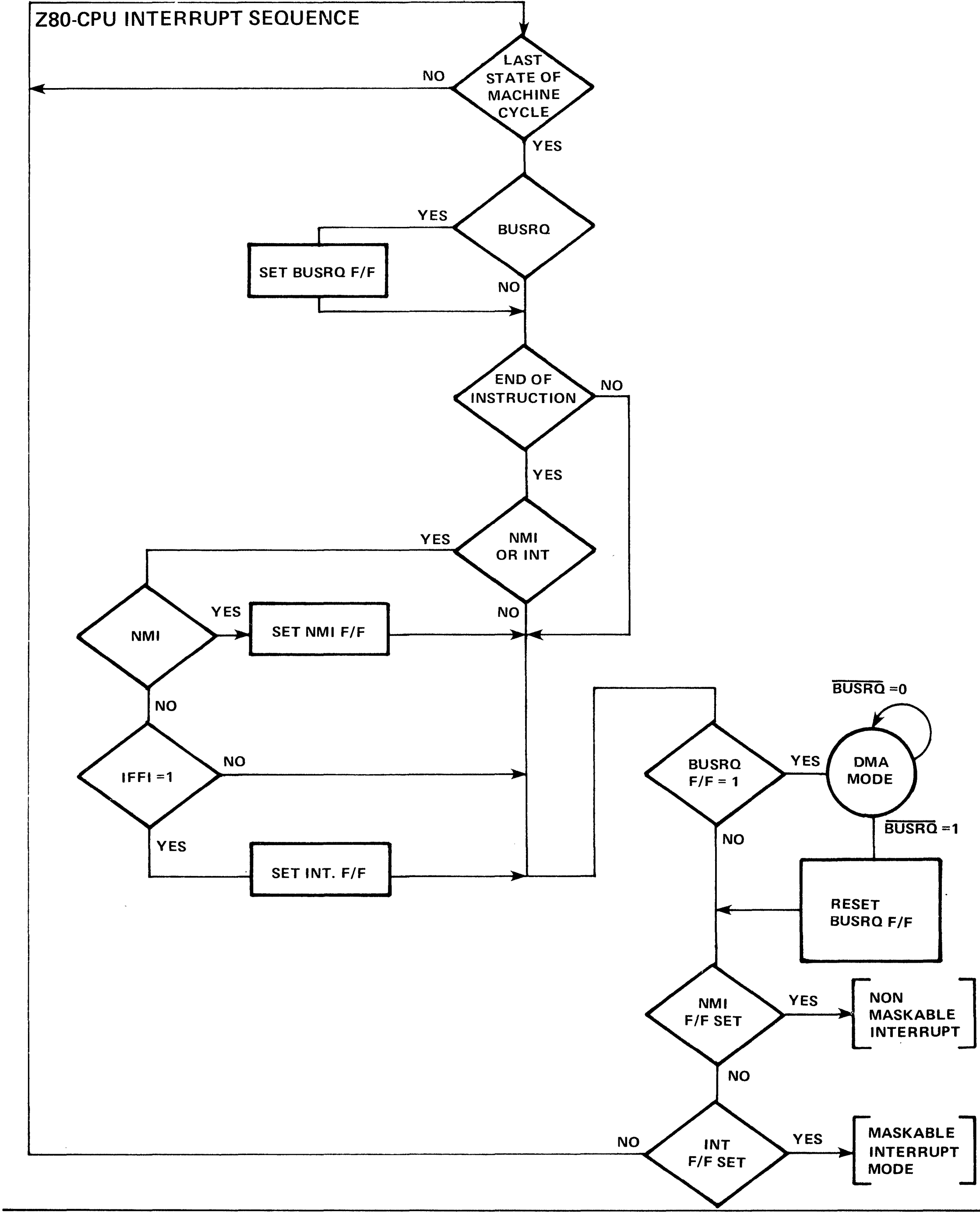
2) CPU ACKNOWLEDGES INTERRUPT. Priority status is frozen when $\overline{\text{M1}}$ goes low during the Interrupt Acknowledge sequence. Propagation delays down the IEI/IEO daisy chain must be settled out when $\overline{\text{IORQ}}$ goes low. If IEI is HIGH, an active Peripheral Device will place its Interrupt Vector on the Data Bus when $\overline{\text{IORQ}}$ goes low. That Peripheral then releases its hold on $\overline{\text{INT}}$ allowing interrupts from a higher priority device. Lower priority devices are inhibited from placing their Vector on the Data Bus or Interrupting because IEO is low on the active device.

3) INTERRUPT IS CLEARED. An active Peripheral device (IEI=1, IEO=0) monitors OP Code fetches for an RETI (ED 4D) instruction which tells the peripheral that its Interrupt Service Routine is over. The peripheral device then re-activates its internal Interrupt structure as well as raising its IEO line to enable lower priority devices.

INTERRELATIONSHIP OF $\overline{\text{INT}}$, $\overline{\text{NMI}}$, AND $\overline{\text{BUSRQ}}$

The following flow chart details the relationship of three control inputs to the Z80-CPU. Note the following from the flow chart.

1. $\overline{\text{INT}}$ and $\overline{\text{NMI}}$ are always acted on at the end of an instruction.
2. $\overline{\text{BUSRQ}}$ is acted on at the end of a machine cycle.
3. While the CPU is in the DMA MODE, it will not respond to active inputs on $\overline{\text{INT}}$ or $\overline{\text{NMI}}$.
4. These three inputs are acted on in the following order of priority: a)$\overline{\text{BUSRQ}}$ b)$\overline{\text{NMI}}$ c)$\overline{\text{INT}}$


Z80-CPU INTERRUPT SEQUENCE

9.0 HARDWARE IMPLEMENTATION EXAMPLES

This chapter is intended to serve as a basic introduction to implementing systems with the Z80-CPU.

MINIMUM SYSTEM

Figure 9.0-1 is a diagram of a very simple Z80 system. Any Z80 system must include the following five elements:

1) Five volt power supply
2) Oscillator
3) Memory devices
4) I/O circuits
5) CPU

MINIMUM Z80 COMPUTER SYSTEM

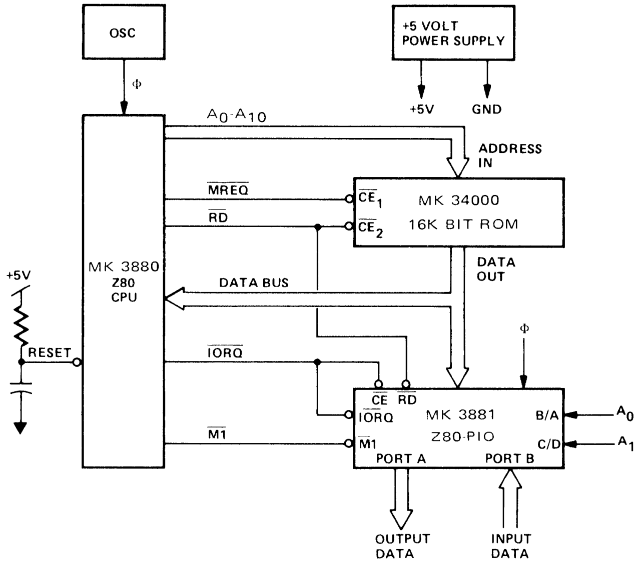


FIGURE 9.0-1

Since the Z80-CPU only requires a single 5 volt supply, most small systems can be implemented using only this single supply.

The oscillator can be very simple since the only requirement is that it be a 5 volt square wave. For systems not running at full speed, a simple RC oscillator can be used. When the CPU is operated near the highest possible frequency, a crystal oscillator is generally required because the system timing will not tolerate the drift or jitter that an RC network will generate. A crystal oscillator can be made from inverters and a few discrete components or monolithic circuits are widely available.

The external memory can be any mixture of standard RAM, ROM, or PROM. In this simple example we have shown a single 16K bit ROM (2K bytes) being utilized as the entire memory system. For this example we have assumed that the Z80 internal register configuration contains sufficient Read/Write storage so that external RAM memory is not required.

Every computer system requires I/O circuits to allow it to interface to the "real world." In this simple example it is assumed that the output is an 8 bit control vector and the input is an 8 bit status word. The input data could be gated onto the data bus using any standard tri-state driver while the output data could be latched with any type of standard TTL latch. For this example we have used a Z80-PIO for the I/O circuit. This single circuit attaches to the data bus as shown and provides the required 16 bits of TTL compatible I/O. (Refer to the Z80-PIO manual for details on the operation of this circuit.) Notice in this example that with only three LSI circuits, a simple oscillator and a single 5 volt power supply, a powerful computer has been implemented.

ADDING RAM

Most computer systems require some amount of external Read/Write memory for data storage and to implement a "stack". Figure 9.0-2 illustrates how 256 bytes of static memory can be added to the previous example. In this example the memory space is assumed to be organized as follows:

ROM & RAM IMPLEMENTATION EXAMPLE

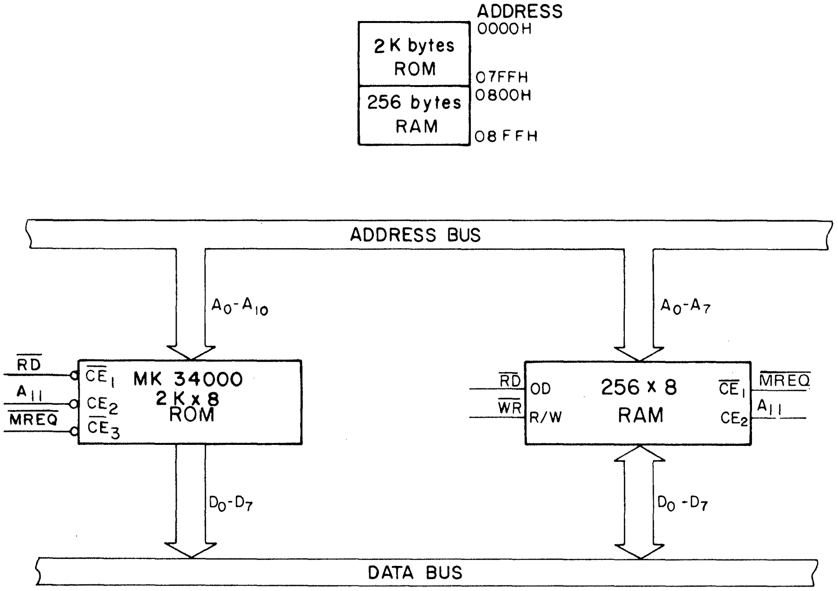


FIGURE 9.0-2

In this diagram the address space is described in hexidecimal notation. For this example, address bit A_{11} separates the ROM space from the RAM space so that it can be used for the chip select function. For larger amounts of external ROM or RAM, a simple TTL decoder will be required to form the chip selects.

MEMORY SPEED CONTROL

For many applications, it may be desirable to use slow memories to reduce costs. The $\overline{WAIT}$ line on the CPU allows the Z80 to operate with any speed memory. By referring back to section 4 you will notice that the memory access time requirements are most severe during the M1 cycle instruction fetch. All other memory accesses have an additional one half of a clock cycle to be completed. For this reason it may be desirable in some applications to add one wait state to the M1 cycle so that slower memories can be used. Figure 9.0-3 is an example of a simple circuit that will accomplish this task. This circuit can be changed to add a single wait state to any memory access as shown in Fıgure 9.0-4.

ADDING ONE WAIT STATE TO AN M1 CYCLE

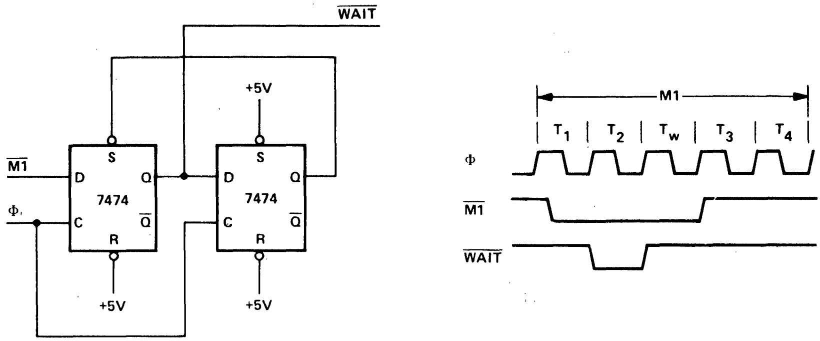


FIGURE 9.0-3

ADDING ONE WAIT STATE TO ANY MEMORY CYCLE

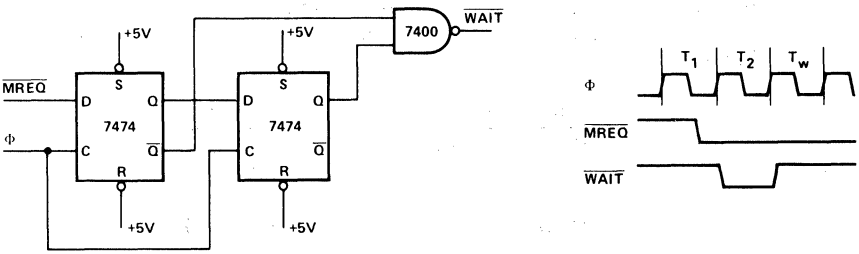


FIGURE 9.0-4

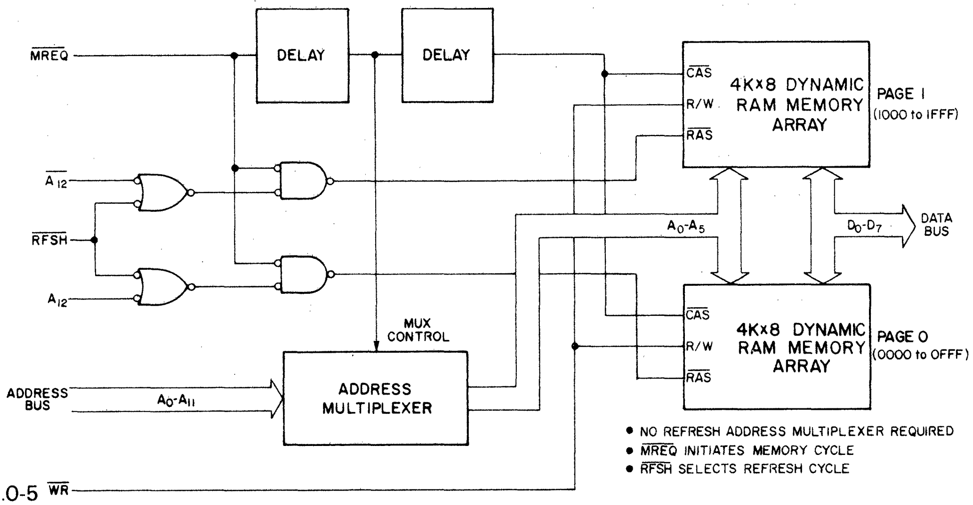
INTERFACING DYNAMIC MEMORIES

This section is intended only to serve as a brief introduction to interfacing dynamic memories. Each individual dynamic RAM has varying specifications that will require minor modifications to the description given here and no attempt will be made in this document to give details for any particular RAM.

Figure 9.0-5 illustrates the logic necessary to interface 8K bytes of dynamic RAM using 16-pin 4K dynamic memories. This Figure assumes that the RAM's are the only memory in the system so that A_{12} is used to select between the two pages of memory. During refresh time, all memories in the system must be read. The CPU provides the proper refresh address on lines A_0 through A_6. To add additional memory to the system it is necessary to only replace the two gates that operate on A_{12} with a decoder that operates on all required address bits. For larger systems, buffering for the address and data bus is also generally required.

An application note entitled "Z80 Interfacing Techniques for Dynamic RAM" is available from your MOSTEK representative which describes dynamic RAM design techniques.

INTERFACING DYNAMIC RAMS



- NO REFRESH ADDRESS MULTIPLEXER REQUIRED
- $\overline{\text{MREQ}}$ INITIATES MEMORY CYCLE
- $\overline{\text{RFSH}}$ SELECTS REFRESH CYCLE

FIGURE 9.0-5

Z80–CPU DESIGN CONSIDERATIONS: CLOCK CIRCUITRY

Proper Z80 clock circuitry design is of paramount importance when designing a Z80 system. Parameters such as clock rise and fall times, min./max. clock high and low times, and max clock over and under shoot should be closely adhered to. Violation of these specs will result in unreliable and unpredictable CPU/peripheral behavior. Several manufacturers offer a wide variety of combination oscillator/drivers housed in 14 pin DIP packages. The following is a suggested source of reliable oscillators/drivers currently available.

Vendor	Function	Part No.
Motorola	Oscillator/Driver	K1160 series
Motorola	Oscillator	K1114
MF Electronics	Oscillator	MF1114
Hybrid House	Driver	HH3006A

Figure 9.0-6 illustrates a schematic recommended for driving the Z80 CPU, as well as other Z80 peripherals. This configuration meets the 30 ns rise and fall time while driving up to a 150 pf. load. Note the divide by two input flip flop to provide a 50 percent duty cycle clock. This stage may be omitted if the oscillator is guaranteed to be within the specifications.

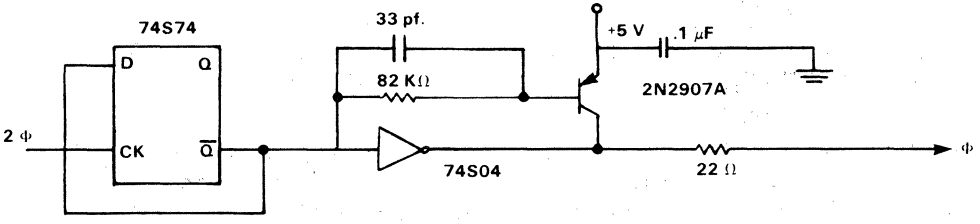


FIGURE 9.0-6

RESET CIRCUITRY

The Z80-CPU has the characteristic that if the $\overline{\text{RESET}}$ input goes low during T2 or T4 of a cycle that the MREQ signal will go to an indeterminate state for one T-State approximately 3 T-States later. If there are dynamic memories in the system this action could cause an aborted or short access of the dynamic RAM which could cause destruction of data within the RAM. If the contents of RAM are of no concern after RESET, then this characteristic is no problem as the CPU always resets properly. If RAM contents must be preserved, then the falling edge of the $\overline{\text{RESET}}$ input must be synchronized by the falling edge of $\overline{\text{M1}}$.

The circuitry of Figure 9.0-7 does this synchronization as well as providing a one-shot to limit the duration of the CPU $\overline{\text{RESET}}$ pulse. The CPU $\overline{\text{RESET}}$ signal must be a pulse even though the EXTERNAL RESET button is held closed to avoid suspending the CPU refresh of dynamic RAM for a time long enough to destroy data in the RAM.

MANUAL AND POWER–ON RESET CIRCUIT

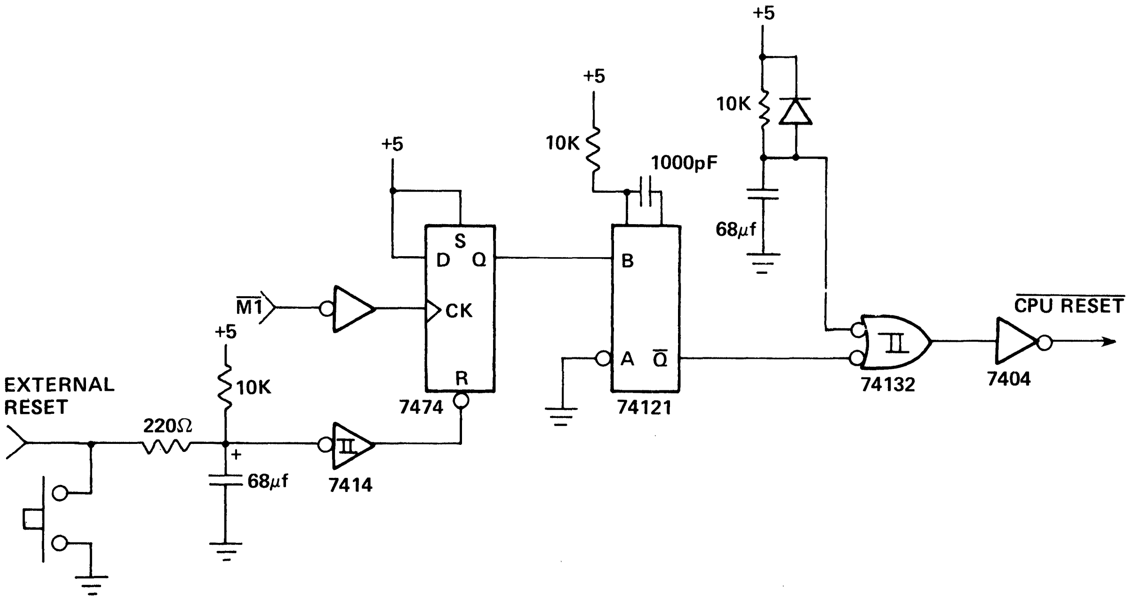


FIGURE 9.0-7

ADDRESS LATCHING

In order to guarantee proper operation of the Z80-CPU with dynamic RAMs the upper 4 bits of the address should be latched as shown in Figure 9.0-8. This action is required because the Z80-CPU does not guarantee that the Address Bus will hold valid before the rising edge of MREQ on an OP Code Fetch.

This action does not directly affect dynamic memories because they latch addresses internally. The problem comes from the address decoder which generates RAS. If the address lines which drive the decoder are allowed to change while $\overline{\text{MREQ}}$ is low, then a "glitch" can occur on the $\overline{\text{RAS}}$ line or lines, which may have the effect of destroying one row of data within the dynamic RAM.

ADDRESS LATCH

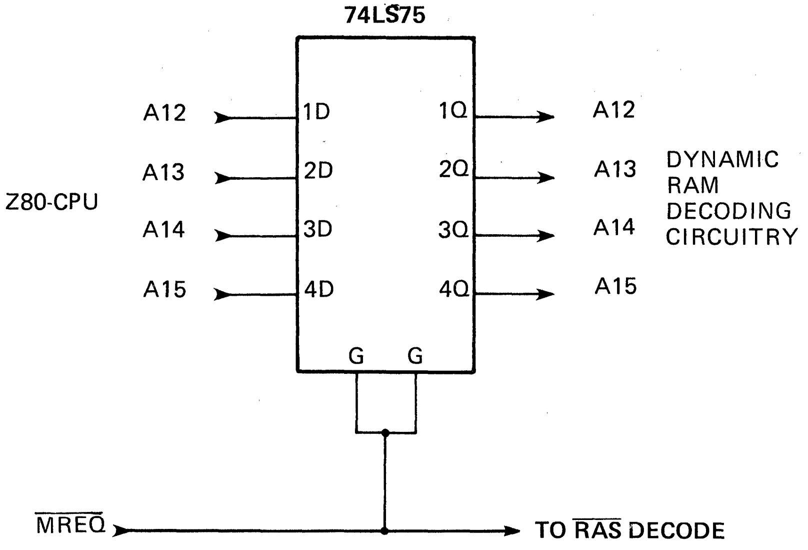


FIGURE 9.0-8

$\overline{\text{RAS}}$ TIMING WITH AND WITHOUT ADDRESS LATCH.

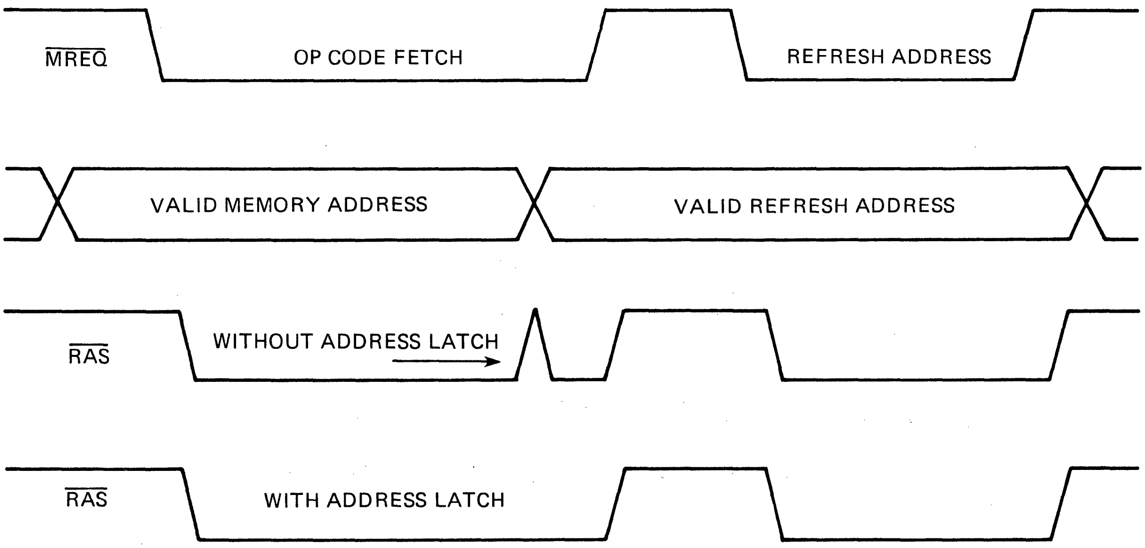


FIGURE 9.0-9

10.0 SOFTWARE IMPLEMENTATION EXAMPLES

10.1 Methods of Software Implementation

Several different approaches are possible in developing software for the Z80 (Figure 10.1) First of all, Assembly Language or a high level language may be used as the source language. These languages may then be translated into machine language on a commercial time sharing facility using a cross-assembler or cross-compiler or, in the case of assembly language, the translation can be accomplished on a Z80 Development System using a resident assembler. Finally, the resulting machine code can be debugged either on a time-sharing facility using a Z80 simulator or on a Z80 Development System which uses a Z80-CPU directly.

SOFTWARE GENERATION TECHNIQUES

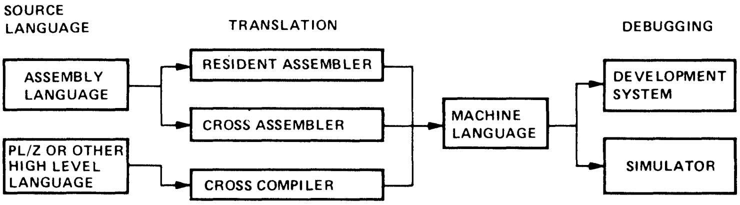


FIGURE 10.1

In selecting a source language, the primary factors to be considered are clarity and ease of programming vs. code efficiency. A high level language with its machine independent constraints is typically better for formulating and maintaining algorithms, but the resulting machine code is usually somewhat less efficient than what can be written directly in assembly language. These tradeoffs can often be balanced by combining high level language and assembly language routines, identifying those portions of a task which must be optimized and writing them as assembly language subroutines.

Deciding whether to use a resident or cross assembler is a matter of availability and short-term vs. long-term expense. While the initial expenditure for a development system is higher than that for a time-sharing terminal, the cost of an individual assembly using a resident assembler is negligible while the same operation on a time-sharing system is relatively expensive and in a short time this cost can equal the total cost of a development system.

Debugging on a development system vs. a simulator is also a matter of availability and expense combined with operational fidelity and flexibility. As with the assembly process, debugging is less expensive on a development system than on a simulator available through time-sharing. In addition, the fidelity of the operating environment is preserved through real-time execution on a Z80-CPU and by connecting the I/O and memory components which will actually be used in the production system. The only advantage to the use of a simulator is the range of criteria which may be selected for such debugging procedures as tracing and setting breakpoints. This flexibility exists because a software simulation can achieve any degree of complexity in its interpretation of machine instructions while development system procedures have hardware limitations such as the capacity of the real-time storage module, the number of breakpoint registers and the pin configuration of the CPU. Despite such hardware limitations, debugging on a development system is typically more productive than on a simulator because of the direct interaction that is possible between the programmer and the authentic execution of his program.

10.2 Software Features Offered by the Z80-CPU

The Z80 instruction set provides the user with a large and flexible repetoire of operations with which to formulate control of the Z80-CPU.

The primary, auxiliary and index registers can be used to hold the arguments of arithmetic and logical operations, or to form memory addresses, or as fast-access storage for frequently used data.

Information can be moved directly from register to register; from memory to memory; from memory to registers; or from registers to memory. In addition, register contents and register/memory contents can be exchanged without using temporary storage. In particular, the contents of primary and auxiliary registers can be completely exchanged by executing only two instructions. EX and EXX. This register exchange procedure can be used to separate the set of working registers between different logical procedures or to expand the set of available registers in a single procedure.

Storage and retrieval of data between pairs of registers and memory can be controlled on a last-in first-out basis through PUSH and POP instructions which utilize a special stack pointer register, SP. This stack register is available both to manipulate data and to automatically store and retrieve addresses for subroutine linkage. When a subroutine is called, for example, the address following the CALL instruction is placed on the top of the pushdown stack pointed to by SP. When a subroutine returns to the calling routine, the address on the top of the stack is used to set the program counter for the address of the next instruction. The stack pointer is adjusted automatically to reflect the current "top" stack position during PUSH, POP, CALL and RET instructions. This stack mechanism allows pushdown data stacks and subroutine calls to be nested to any practical depth because the stack area can potentially be as large as memory space.

The sequence of instruction execution can be controlled by six different flags (carry, zero, sign, parity/overflow, add-subtract, half-carry) which reflect the results of arithmetic, logical, shift and compare instructions. After the execution of an instruction which sets a flag, that flag can be used to control a conditional jump or return instruction. These instructions provide logical control following the manipulation of single bit, eight-bit byte (or) sixteen-bit data quantities.

A full set of logical operations, including AND, OR, XOR (exclusive –OR), CPL (NOR) and NEG (two's complement) are available for Boolean operations between the accumulator and 1) all other eight-bit registers, 2) memory locations or 3) immediate operands.

In addition, a full set of arithmetic and logical shifts in both directions are available which operate on the contents of all eight-bit primary registers or directly on any memory location. The carry flag can be included or simply set by these shift instructions to provide both the testing of shift results and to link register/register or register/memory shift operations.

10.3 Examples of Use of Special Z80 Instructions

A. Let us assume that a string of data in memory starting at location "DATA" is to be moved into another area of memory starting at location "BUFFER" and that the string length is 737 bytes. This operation can be accomplished as follows:

```
LD      HL, DATA      ;START ADDRESS OF DATA STRING
LD      DE, BUFFER    ;START ADDRESS OF TARGET BUFFER
LD      BC, 737       ;LENGTH OF DATA STRING
LDIR                  ;MOVE  STRING  –  TRANSFER  MEMORY
                      ;POINTED TO BY HL INTO MEMORY
                      ;LOCATION POINTED TO BY DE INCREMENT
                      ;HL  AND  DE,  DECREMENT  BC  PROCESS
                      ;UNTIL BC=0.
```

11 bytes are required for this operation and each byte of data is moved in 21 clock cycles.

B. Let's assume that a string in memory starting at location "DATA" is to be moved into another area of memory starting at location "BUFFER" until an ASCII $ character (used as string delimiter) is found. Let's also assume that the maximum string length is 132 characters. The operation can be performed as follows:

```
        LD      HL, DATA     ;STARTING ADDRESS OF DATA STRING
        LD      DE, BUFFER   ;STARTING ADDRESS OF TARGET BUFFER
        LD      BC, 132      ;MAXIMUM STRING LENGTH
        LD      A, '$'       ;STRING DELIMITER CODE
  LOOP: CP      (HL)         ;COMPARE MEMORY CONTENTS WITH DE-
                             ;LIMITER
        JR      Z, END-$     ;GO TO END IF CHARACTERS EQUAL
        LDI                  ;MOVE CHARACTER (HL) TO (DE)
                             ;INCREMENT HL AND DE, DECREMENT BC
        JP      PE,LOOP      ;GO TO "LOOP" IF MORE CHARACTERS
   END:                      ;OTHERWISE, FALL THROUGH
                             ;NOTE: P/V FLAG IS USED
                             ;TO INDICATE THAT REGISTER BC WAS
                             ;DECREMENTED TO ZERO.
```

19 bytes are required for this operation.

C. Let us assume that a 16-digit decimal number represented in packed BCD format (two BCD digits/byte) has to be shifted as shown in the Figure 10.2 in order to mechanize BCD multiplication or division. The operation can be accomplished as follows:

```
        LD      HL, DATA     ;ADDRESS OF FIRST BYTE
        LD      B, COUNT     ;SHIFT COUNT
        XOR     A            ;CLEAR ACCUMULATOR
ROTAT:  RLD                  ;ROTATE LEFT LOW ORDER DIGIT IN ACC
                             ;WITH DIGITS IN (HL)
        INC     HL           ;ADVANCE MEMORY POINTER
        DJNZ    ROTAT-$      ;DECREMENT B AND GO TO ROTAT IF
                             ;B IS NOT ZERO, OTHERWISE FALL THROUGH
```

BCD DATA SHIFTING

11 bytes are required for this operation.

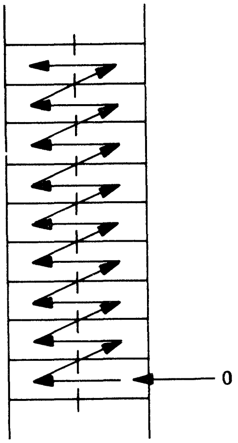


FIGURE 10.2

11 bytes are required for this operation.

D. Let us assume that one number is to be subtracted from another and a) that they are both in packed BCD format, b) that they are of equal but varying length, and c) that the result is to be stored in the location of the minuend. The operation can be accomplished as follows:

```
        LD      HL, ARG1        ;ADDRESS OF MINUEND
        LD      DE, ARG2        ;ADDRESS OF SUBTRAHEND
        LD      B, LENGTH       ;LENGTH OF TWO ARGUMENTS
        AND     A               ;CLEAR CARRY FLAG
SUBDEC: LD      A, (DE)         ;SUBTRAHEND TO ACC
        SBC     A, (HL)         ;SUBTRACT (HL) FROM ACC
        DAA                     ;ADJUST RESULT TO DECIMAL CODED VALUE
        LD      (HL), A         ;STORE RESULT
        INC     HL              ;ADVANCE MEMORY POINTERS
        INC     DE
        DJNZ    SUBDEC—$        ;DECREMENT B AND GO TO "SUBDEC" IF B
                                ;NOT ZERO, OTHERWISE FALL THROUGH
```

17 bytes are required for this operation.

10.4 Examples of Programming Tasks

A. The following program sorts an array of numbers each in the range <0,255> into ascending order using a standard exchange sorting algorithm.

```
01/22/76     11:14:37                  BUBBLE LISTING
LOC    OBJ CODE  STMT  SOURCE STATEMENT

                  1    ;  *** STANDARD EXCHANGE (BUBBLE) SORT ROUTINE***
                  2    ;
                  3    ;  AT ENTRY: HL CONTAINS ADDRESS OF DATA
                  4    .           C CONTAINS NUMBER OF ELEMENTS TO BE SORTED
                  5    ;           (1<C<256)
                  6    ;
                  7    ;  AT EXIT: DATA SORTED IN ASCENDING ORDER
                  8    ;
                  9    ;  USE OF REGISTERS
                 10    ;
                 11    ;  REGISTER    CONTENTS
                 12    ;
                 13    ;  A           TEMPORARY STORAGE FOR CALCULATIONS
                 14    ;  B           COUNTER FOR DATA ARRAY
                 15    ;  C           LENGTH OF DATA ARRAY
                 16    ;  D           FIRST ELEMENT IN COMPARISON
                 17    ;  E           SECOND ELEMENT IN COMPARISON
                 18    ;  H           FLAG TO INDICATE EXCHANGE
                 19    ;  L           UNUSED
                 20    ;  IX          POINTER INTO DATA ARRAY
                 21    ;  IY          UNUSED
                 22    ;
```

```
LOC   OBJ CODE  STMT SOURCE STATMENT
0000  222600    23   SORT:    LD    (DATA), HL     ;SAVE DATA ADDRESS
0003  CB84      24   LOOP:    RES   FLAG, H        ;INITIALIZE EXCHANGE FLAG
0005  41        25            LD    B,C            ;INITIALIZE LENGTH COUNTER
0006  05        26            DEC   B              ;ADJUST FOR TESTING
0007  DD2A2600  27            LD    IX, (DATA)     ;INITIALIZE ARRAY POINTER
000B  DD7E00    28   NEXT:    LD    A,(IX+0)       ;FIRST ELEMENT IN COMPARISON
000E  57        29            LD    D, A           ;TEMPORARY STORAGE FOR ELEMENT
000F  DD5E01    30            LD    E, (IX+1)      ;SECOND ELEMENT IN COMPARISON
0012  93        31            SUB   E              ;COMPARISON FIRST TO SECOND
0013  3008      32            JR    NC, NOEX-$     ;IF FIRST> SECOND, NO JUMP
0015  DD7300    33            LD    (IX), E        ;EXCHANGE ARRAY ELEMENTS
0018  DD7201    34            LD    (IX+1), D
001B  CBC4      35            SET   FLAG H         ;RECORD EXCHANGE OCCURRED
001D  DD23      36   NOEX:    INC   IX             ;POINT TO NEXT DATA ELEMENT
001F  10EA      37            DJNZ  NEXT-$         ;COUNT NUMBER OF COMPARISONS
                                                   ;REPEAT IF MORE DATA PAIRS
0021  CB44      39            BIT   FLAG, H        ;DETERMINE IF EXCHANGE OCCURRED
0023  20DE      40            JR    NZ, LOOP-$     ;CONTINUE IF DATA UNSORTED
0025  C9        41            RET                  ;OTHERWISE, EXIT
                42   ;
0026            43   FLAG:    EQU   0              ;DESIGNATION OF FLAG BIT
0026            44   DATA:    DEFS  2              ;STORAGE FOR DATA ADDRESS
                45            END
```

B. The following program multiplies two unsigned 16-bit integers and leaves the result in the HL register pair.

```
01/22/76     11:32:36              MULTIPLY LISTING
LOC    OBJ CODE    STMT   SOURCE STATEMENT

0000                 1    MULT:;    UNSIGNED SIXTEEN BIT INTEGER MULTIPLY.
                     2    ;         ON ENTRANCE: MULTIPLIER IN HL.
                     3    ;                      MULTIPLICAND IN DE.
                     4    ;
                     5    ;         ON EXIT: RESULT IN HL.
                     6    ;
                     7    ;         REGISTERS USES:
                     8    ;
                     9    ;
                    10    ;         H    HIGH ORDER PARTIAL RESULT
                    11    ;         L    LOW ORDER PARTIAL RESULT
                    12    ;         D    HIGH ORDER MULTIPLICAND
                    13    ;         E    LOW ORDER MULTIPLICAND
                    14    ;         B    COUNTER FOR NUMBER OF SHIFTS
                    15    ;         C    HIGH ORDER BITS OF MULTIPLIER
                    16    ;         A    LOW ORDER BITS OF MULTIPLIER
                    17    ;
0000   0610         18              LD   B, 16;          NUMBER OF BITS—INITIALIZE
0002   4A           19              LD   C,D;            MOVE MULTIPLIER
0003   7B           20              LD   A,E;
0004   EB           21              EX   DE,HL;          MOVE MULTIPLICAND
0005   210000       22              LD   HL,0;           CLEAR PARTIAL RESULT
0008   CB39         23    MLOOP:    SRL  C;              SHIFT MULTIPLIER RIGHT
000A   1F           24              RR   A;              LEAST SIGNIFICANT BIT IS
                                                         IN CARRY.
000B   3001         26              JR   NC, NOADD-$     IF NO CARRY' SKIP THE ADD.
```

III Z80 FAMILY TECHNICAL MANUALS

```
LOC    OBJ CODE   STMT  SOURCE STATMENT
000D    19         27           ADD HL, DE;           ELSE ADD MULTIPLICAND TO
                                                      PARTIAL RESULT.
000E    EB         29   NOADD:  EX   DE,HL;           SHIFT MULTIPLICANT LEFT
000F    29         30           ADD HL,HL;            BY MULTIPLYING IT BY TWO.
0010    EB         31           EX   DE,HL;
0011    10F5       32         DJNZ   MLOOP-$;         REPEAT UNTIL NO MORE BITS.
0013    C9         33           RET;
                   34           END;
```

11.0 ELECTRICAL SPECIFICATIONS
ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias	Specified Operating Range
Storage Temperature	–65° C to +150° C
Voltage on Any Pin with Respect to Ground	–0.3V to +7V
Power Dissipation	1.5W

D.C. CHARACTERISTICS

T_A = 0° C to 70° C, V_{CC} = 5V ± 5% unless otherwise specified

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNIT	TEST CONDITION
V_{ILC}	Clock Input Low Voltage	–0.3		0.8	V	
V_{IHC}	Clock Input High Voltage	Vcc-.6		Vcc+.3	V	
V_{IL}	Input Low Voltage	–0.3		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{CC}	V	
V_{OL}	Output Low Voltage			0.4	V	I_{OL} = 1.8mA
V_{OH}	Output High Voltage	2.4			V	I_{OH} = –250 μA
I_{CC}	Power Supply Current			150*	mA	
I_{LI}	Input Leakage Current			±10	μA	V_{IN} = 0 to V_{CC}
I_{LOH}	Tri-State Output Leakage Current in Float			10	μA	V_{OUT} = 2.4 to V_{CC}
I_{LOL}	Tri-State Output Leakage Current in Float			–10	μA	V_{OUT} = 0.4V
I_{LD}	Data Bus Leakage Current in Input Mode			±10	μA	$0 \leq V_{IN} \leq V_{CC}$

*200mA for -4, -10 or -20 devices

NOTE: All outputs are rated at one standard TTL load.

CAPACITANCE

T_A = 25° C, f = 1MHz unmeasured pins returned to ground

SYMBOL	PARAMETER	MAX.	UNIT
CΦ	Clock Capacitance	35	pF
C_{IN}	Input Capacitance	5	pF
C_{OUT}	Output Capacitance	10	pF

*Comment

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

A C CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = +5V ± 5%, Unless Otherwise Noted

SIGNAL	SYMBOL	PARAMETER	MIN.	MAX.	UNIT	TEST CONDITION
Φ	t_c	Clock Period	.4	[12]	µsec	
	$t_w(\Phi H)$	Clock Pulse Width, Clock High	180	(D)	nsec	
	$t_w(\Phi L)$	Clock Pulse Width, Clock Low	180	2000	nsec	
	$t_{r,f}$	Clock Rise and Fall Time		30	nsec	
$A_{0\text{-}15}$	$t_{D(AD)}$	Address Output Delay		145	nsec	C_L = 50pF
	$t_{F(AD)}$	Delay to Float		110	nsec	
	t_{acm}	Address Stable Prior to $\overline{\text{MREQ}}$ (Memory Cycle)	[1]		nsec	
	t_{aci}	Address Stable Prior to $\overline{\text{IORQ}}$, $\overline{\text{RD}}$ or $\overline{\text{WR}}$ (I/O Cycle)	[2]		nsec	
	t_{ca}	Address Stable From $\overline{\text{RD}}$, $\overline{\text{WR}}$, $\overline{\text{IORQ}}$ or $\overline{\text{MREQ}}$	[3]		nsec	Except T3-M1
	t_{caf}	Address Stable From $\overline{\text{RD}}$ or $\overline{\text{WR}}$ During Float	[4]		nsec	
$D_{0\text{-}7}$	$t_{D(D)}$	Data Output Delay		230	nsec	C_L = 50pF
	$t_{F(D)}$	Delay to Float During Write Cycle		90	nsec	
	$t_{S\Phi(D)}$	Data Setup Time to Rising Edge of Clock During M1 Cycle	50		nsec	
	$t_{S\overline{\Phi}(D)}$	Data Setup Time to Falling Edge at Clock During M2 to M5	60		nsec	
	t_{dcm}	Data Stable Prior to $\overline{\text{WR}}$ (Memory Cycle)	[5]		nsec	
	t_{dci}	Data Stable Prior to $\overline{\text{WR}}$ (I/O Cycle)	[6]		nsec	
	t_{cdf}	Data Stable From $\overline{\text{WR}}$	[7]		nsec	
	t_H	Input Hold Time	0		nsec	
$\overline{\text{MREQ}}$	$t_{DL\overline{\Phi}(MR)}$	$\overline{\text{MREQ}}$ Delay From Falling Edge of Clock, $\overline{\text{MREQ}}$ Low		100	nsec	C_L = 50 pF
	$t_{DH\Phi(MR)}$	$\overline{\text{MREQ}}$ Delay From Rising Edge of Clock, $\overline{\text{MREQ}}$ High		100	nsec	
	$t_{DH\overline{\Phi}(MR)}$	$\overline{\text{MREQ}}$ Delay From Falling Edge of Clock, $\overline{\text{MREQ}}$ High		100	nsec	
	$t_{w(\overline{MRL})}$	Pulse Width, $\overline{\text{MREQ}}$ Low	[8]		nsec	
	$t_{w(\overline{MRH})}$	Pulse Width, $\overline{\text{MREQ}}$ High	[9]		nsec	
$\overline{\text{IORQ}}$	$t_{DL\Phi(IR)}$	$\overline{\text{IORQ}}$ Delay From Rising Edge of Clock, $\overline{\text{IORQ}}$ Low		90	nsec	C_L = 50 pF
	$t_{DL\overline{\Phi}(IR)}$	$\overline{\text{IORQ}}$ Delay From Falling Edge of Clock, $\overline{\text{IORQ}}$ Low		110	nsec	
	$t_{DH\Phi(IR)}$	$\overline{\text{IORQ}}$ Delay From Rising Edge of Clock, $\overline{\text{IORQ}}$ High		100	nsec	
	$t_{DH\overline{\Phi}(IR)}$	$\overline{\text{IORQ}}$ Delay From Falling Edge of Clock, $\overline{\text{IORQ}}$ High		110	nsec	
$\overline{\text{RD}}$	$t_{DL\Phi(RD)}$	$\overline{\text{RD}}$ Delay From Rising Edge of Clock, $\overline{\text{RD}}$ Low		100	nsec	C_L = 50pF
	$t_{DL\overline{\Phi}(RD)}$	$\overline{\text{RD}}$ Delay From Falling Edge of Clock, $\overline{\text{RD}}$ Low		130	nsec	
	$t_{DH\Phi(RD)}$	$\overline{\text{RD}}$ Delay From Rising Edge of Clock, $\overline{\text{RD}}$ High		100	nsec	
	$t_{DH\overline{\Phi}(BD)}$	$\overline{\text{RD}}$ Delay From Falling Edge of Clock, $\overline{\text{RD}}$ High		110	nsec	
$\overline{\text{WR}}$	$t_{DL\Phi(WR)}$	$\overline{\text{WR}}$ Delay From Rising Edge of Clock, $\overline{\text{WR}}$ Low		80	nsec	C_L = 50pF
	$t_{DL\overline{\Phi}(WR)}$	$\overline{\text{WR}}$ Delay From Falling Edge of Clock $\overline{\text{WR}}$ Low		90	nsec	
	$t_{DH\Phi(WR)}$	$\overline{\text{WR}}$ Delay From Falling Edge of Clock, $\overline{\text{WR}}$ High		100	nsec	
	$t_{w(\overline{WRL})}$	Pulse Width, $\overline{\text{WR}}$ Low	[10]		nsec	

NOTES:

A Data should be enabled onto the CPU data bus when RD is active. During interrupt acknowledge data should be enabled when $\overline{\text{M1}}$ and $\overline{\text{IORQ}}$ are both active.

B The $\overline{\text{RESET}}$ signal must be active for a minimum of 3 clock cycles.

con't'd on page 81

SIGNAL	SYMBOL	PARAMETER	MIN.	MAX.	UNIT	TEST CONDITIONS
$\overline{\text{M1}}$	$t_{DL(M1)}$	$\overline{\text{M1}}$ Delay From Rising Edge of Clock $\overline{\text{M1}}$ Low		130	nsec	C_L = 50pF
	$t_{DH(M1)}$	$\overline{\text{M1}}$ Delay From Rising Edge of Clock $\overline{\text{M1}}$ High		130	nsec	
$\overline{\text{RFSH}}$	$t_{DL(RF)}$	$\overline{\text{RFSH}}$ Delay From Rising Edge of Clock, $\overline{\text{RFSH}}$ Low		180	nsec	C_L = 30pF
	$t_{DH(RF)}$	$\overline{\text{RFSH}}$ Delay From Rising Edge of Clock, $\overline{\text{RFSH}}$ High		150	nsec	
$\overline{\text{WAIT}}$	$t_{S(WT)}$	$\overline{\text{WAIT}}$ Setup Time to Falling Edge of Clock	70		nsec	
$\overline{\text{HALT}}$	$t_{D(HT)}$	$\overline{\text{HALT}}$ Delay Time From Falling Edge of Clock		300	nsec	C_L = 50pF
$\overline{\text{INT}}$	$t_{s(IT)}$	$\overline{\text{INT}}$ Setup Time to Rising Edge of Clock	80		nsec	
$\overline{\text{NMI}}$	$t_w(NML)$	Pulse Width, $\overline{\text{NMI}}$ Low	80		nsec	
$\overline{\text{BUSRQ}}$	$t_{s(BQ)}$	$\overline{\text{BUSRQ}}$ Setup Time to Rising Edge of Clock	80		nsec	
$\overline{\text{BUSAK}}$	$t_{DL(BA)}$	$\overline{\text{BUSAK}}$ Delay From Rising Edge of Clock, $\overline{\text{BUSAK}}$ Low		120	nsec	C_L = 50 pF
	$t_{DH(BA)}$	$\overline{\text{BUSAK}}$ Delay From Falling Edge of Clock, $\overline{\text{BUSAK}}$ High		110	nsec	
$\overline{\text{RESET}}$	$t_{s(RS)}$	$\overline{\text{RESET}}$ Setup Time to Rising Edge of Clock	90		nsec	
	$t_{F(C)}$	Delay to/from Float ($\overline{\text{MREQ}}$, $\overline{\text{IORQ}}$, $\overline{\text{RD}}$ and $\overline{\text{WR}}$)		100	nsec	
	t_{mr}	$\overline{\text{M1}}$ Stable Prior to IORQ (Interrupt Ack.)	[11]		nsec	

[1] $t_{acm} = t_w(\Phi H) + t_f - 75$

[2] $t_{aci} = t_c - 80$

[3] $t_{ca} = t_w(\Phi L) + t_r - 40$

[4] $t_{caf} = t_w(\Phi L) + t_r - 60$

[5] $t_{dcm} = t_c - 210$

[6] $t_{dci} = t_w(\Phi L) + t_r - 210$

[7] $t_{cdf} = t_w(\Phi L) + t_r - 80$

[8] $t_w(\overline{MRL}) = t_c - 40$

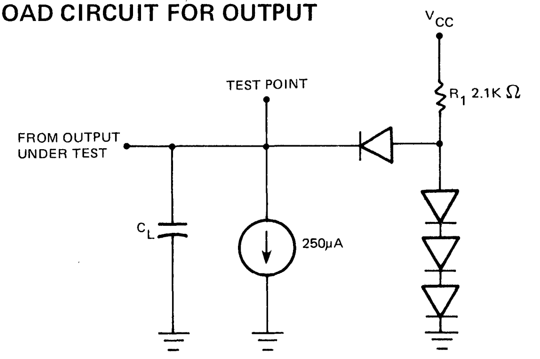
[9] $t_w(\overline{MRH}) = t_w(\Phi H) + t_f - 30$

[10] $t_w(\overline{WR}) = t_c - 40$

[11] $t_{mr} = 2t_c + t_w(\Phi H) + t_f - 80$

[12] $t_c = t_w(\Phi H) + t_w(\Phi L) + t_r + t_f$

LOAD CIRCUIT FOR OUTPUT



NOTES (Cont'd.)

C. Output Delay vs. Load Capacitance
T_A = 70°C V_{CC} = 5V±5%
Add 10 nsec delay for each 50pF increase in load up to a maximum of 200pF for the data bus and 100pF for address and control lines.

D. Although static by design, testing guarantees $t_w(\Phi H)$ of 200 μsec maximum.

A. C. CHARACTERISTICS T_A = 0°C to 70°C, Vcc = +5V ±5%, Unless Otherwise Noted

SIGNAL	SYMBOL	PARAMETER	MIN.	MAX.	UNIT	TEST CONDITIONS
Φ	t_c	Clock Period	.25	[12]	μsec	
	$t_{w(\Phi H)}$	Clock Pulse Width, Clock High	110	(D)	nsec	
	$t_{w(\Phi L)}$	Clock Pulse Width, Clock Low	110	2000	nsec	
	$t_{r,f}$	Clock Rise and Fall Time		30	nsec	
$A_{0\text{-}15}$	$t_{D(AD)}$	Address Output Delay		110	nsec	C_L = 50pF
	$t_{F(AD)}$	Delay to Float		90	nsec	
	t_{acm}	Address Stable Prior to $\overline{MREQ}$ (Memory Cycle)	[1]		nsec	
	t_{aci}	Address Stable Prior to $\overline{IORQ}$, $\overline{RD}$ or $\overline{WR}$ (I/O Cycle)	[2]		nsec	
	t_{ca}	Address Stable From $\overline{RD}$, $\overline{WR}$, $\overline{IORQ}$ or $\overline{MREQ}$	[3]		nsec	Except T3.M1
	t_{caf}	Address Stable From $\overline{RD}$ or $\overline{WR}$ During Float	[4]		nsec	
$D_{0\text{-}7}$	$t_{D(D)}$	Data Output Delay		150	nsec	C_L = 50pF
	$t_{F(D)}$	Delay to Float During Write Cycle		90	nsec	
	$t_{S\Phi(D)}$	Data Setup Time to Rising Edge of Clock During M1 Cycle	35		nsec	
	$t_{S\overline{\Phi}(D)}$	Data Setup Time to Falling Edge at Clock During M2 to M5	50		nsec	
	t_{dcm}	Data Stable Prior to $\overline{WR}$ (Memory Cycle)	[5]		nsec	
	t_{dci}	Data Stable Prior to $\overline{WR}$ (I/O Cycle)	[6]		nsec	
	t_{cdf}	Data Stable From $\overline{WR}$	[7]		nsec	
	t_H	Input Hold Time	0		nsec	
$\overline{MREQ}$	$t_{DL\overline{\Phi}(MR)}$	$\overline{MREQ}$ Delay From Falling Edge of Clock, $\overline{MREQ}$ Low	20	85	nsec	C_L = 50pF
	$t_{DH\Phi(MR)}$	$\overline{MREQ}$ Delay From Rising Edge of Clock, $\overline{MREQ}$ High		85	nsec	
	$t_{DH\overline{\Phi}(MR)}$	$\overline{MREQ}$ Delay From Falling Edge of Clock, $\overline{MREQ}$ High		85	nsec	
	$t_{w(\overline{MRL})}$	Pulse Width, $\overline{MREQ}$ Low	[8]		nsec	
	$t_{w(\overline{MRH})}$	Pulse Width, $\overline{MREQ}$ High	[9]		nsec	
$\overline{IORQ}$	$t_{DL\Phi(IR)}$	$\overline{IORQ}$ Delay From Rising Edge of Clock, $\overline{IORQ}$ Low		75	nsec	C_L = 50pF
	$t_{DL\overline{\Phi}(IR)}$	$\overline{IORQ}$ Delay From Falling Edge of Clock, $\overline{IORQ}$ Low		85	nsec	
	$t_{DH\Phi(IR)}$	$\overline{IORQ}$ Delay From Rising Edge of Clock, $\overline{IORQ}$ High		85	nsec	
	$t_{DH\overline{\Phi}(IR)}$	$\overline{IORQ}$ Delay From Falling Edge of Clock, $\overline{IORQ}$ High		85	nsec	
$\overline{RD}$	$t_{DL\Phi(RD)}$	$\overline{RD}$ Delay From Rising Edge of Clock, $\overline{RD}$ Low		85	nsec	C_L = 50pF
	$t_{DL\overline{\Phi}(RD)}$	$\overline{RD}$ Delay From Falling Edge of Clock, $\overline{RD}$ Low		95	nsec	
	$t_{DH\Phi(RD)}$	$\overline{RD}$ Delay From Rising Edge of Clock, $\overline{RD}$ High		85	nsec	
	$t_{DH\overline{\Phi}(RD)}$	$\overline{RD}$ Delay From Falling Edge of Clock, $\overline{RD}$ High		85	nsec	
$\overline{WR}$	$t_{DL\Phi(WR)}$	$\overline{WR}$ Delay From Rising Edge of Clock, $\overline{WR}$ Low		65	nsec	C_L = 50pF
	$t_{DL\overline{\Phi}(WR)}$	$\overline{WR}$ Delay From Falling Edge of Clock, $\overline{WR}$ Low		80	nsec	
	$t_{DH\Phi(WR)}$	$\overline{WR}$ Delay From Falling Edge of Clock, $\overline{WR}$ High		80	nsec	
	$t_{w(\overline{WR}L)}$	Pulse Width, $\overline{WR}$ Low	[10]		nsec	

NOTES:

A Data should be enabled onto the CPU data bus when $\overline{RD}$ is active. During interrupt acknowledge data should be enabled when M1 and $\overline{IORQ}$ are both active.

B The $\overline{RESET}$ signal must be active for a minimum of 3 clock cycles.

(Cont'd. on page 83)

SIGNAL	SYMBOL	PARAMETER	MIN.	MAX.	UNIT	TEST CONDITION
$\overline{M1}$	$t_{DL(M1)}$ $t_{DH(M1)}$	$\overline{M1}$ Delay From Rising Edge of Clock $\overline{M1}$ Low $\overline{M1}$ Delay From Rising Edge of Clock, $\overline{M1}$ High		100 100	nsec nsec	C_L = 50pF
$\overline{RFSH}$	$t_{DL(RF)}$ $t_{DH(RF)}$	$\overline{RFSH}$ Delay From Rising Edge of Clock, $\overline{RFSH}$ Low $\overline{RFSH}$ Delay From Rising Edge of Clock $\overline{RFSH}$ High		130 120	nsec nsec	C_L = 50pF
$\overline{WAIT}$	$t_{S(WT)}$	$\overline{WAIT}$ Setup Time to Falling Edge of Clock	70		nsec	
$\overline{HALT}$	$t_{D(HT)}$	$\overline{HALT}$ Delay Time From Falling Edge of Clock		300	nsec	C_L = 50pF
$\overline{INT}$	$t_{s(IT)}$	$\overline{INT}$ Setup Time to Rising Edge of Clock	80		nsec	
$\overline{NMI}$	$t_{w(\overline{NML})}$	Pulse Width, $\overline{NMI}$ Low	80		nsec	
$\overline{BUSRQ}$	$t_{s(BQ)}$	$\overline{BUSRQ}$ Setup Time to Rising Edge of Clock	50		nsec	
$\overline{BUSAK}$	$t_{DL(BA)}$ $t_{DH(BA)}$	$\overline{BUSAK}$ Delay From Rising Edge of Clock, $\overline{BUSAK}$ Low $\overline{BUSAK}$ Delay From Falling Edge of Clock, $\overline{BUSAK}$ High		100 100	nsec nsec	C_L = 50pF
$\overline{RESET}$	$t_{s(RS)}$	$\overline{RESET}$ Setup Time to Rising Edge of Clock	60		nsec	
	$t_{F(C)}$	Delay to/From Float ($\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$ and $\overline{WR}$)		80	nsec	
	t_{mr}	$\overline{M1}$ Stable Prior to $\overline{IORQ}$ (Interrupt Ack.)	[11]		nsec	

[1] $t_{acm} = t_w(\Phi H) + t_f - 65$

[2] $t_{aci} = t_c - 70$

[3] $t_{ca} = t_w(\Phi L) + t_r - 50$

[4] $t_{caf} = t_w(\Phi L) + t_r - 45$

[5] $t_{dcm} = t_c - 170$

[6] $t_{dci} = t_w(\Phi L) + t_r - 170$

[7] $t_{cdf} = t_w(\Phi L) + t_r - 70$

[8] $t_w(\overline{MRL}) = t_c - 30$

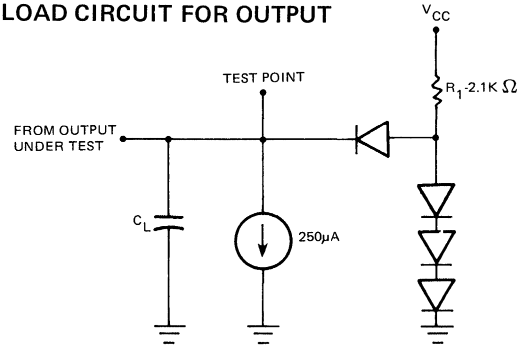
[9] $t_w(\overline{MRH}) = t_w(\Phi H) + t_f - 20$

[10] $t_w(\overline{WR}) = t_c - 30$

[11] $t_{mr} = 2t_c + t_w(\Phi H) + t_f - 65$

[12] $t_c = t_w(\Phi H) + t_w(\Phi L) + t_r + t_f$

LOAD CIRCUIT FOR OUTPUT



NOTES (Cont'd.)

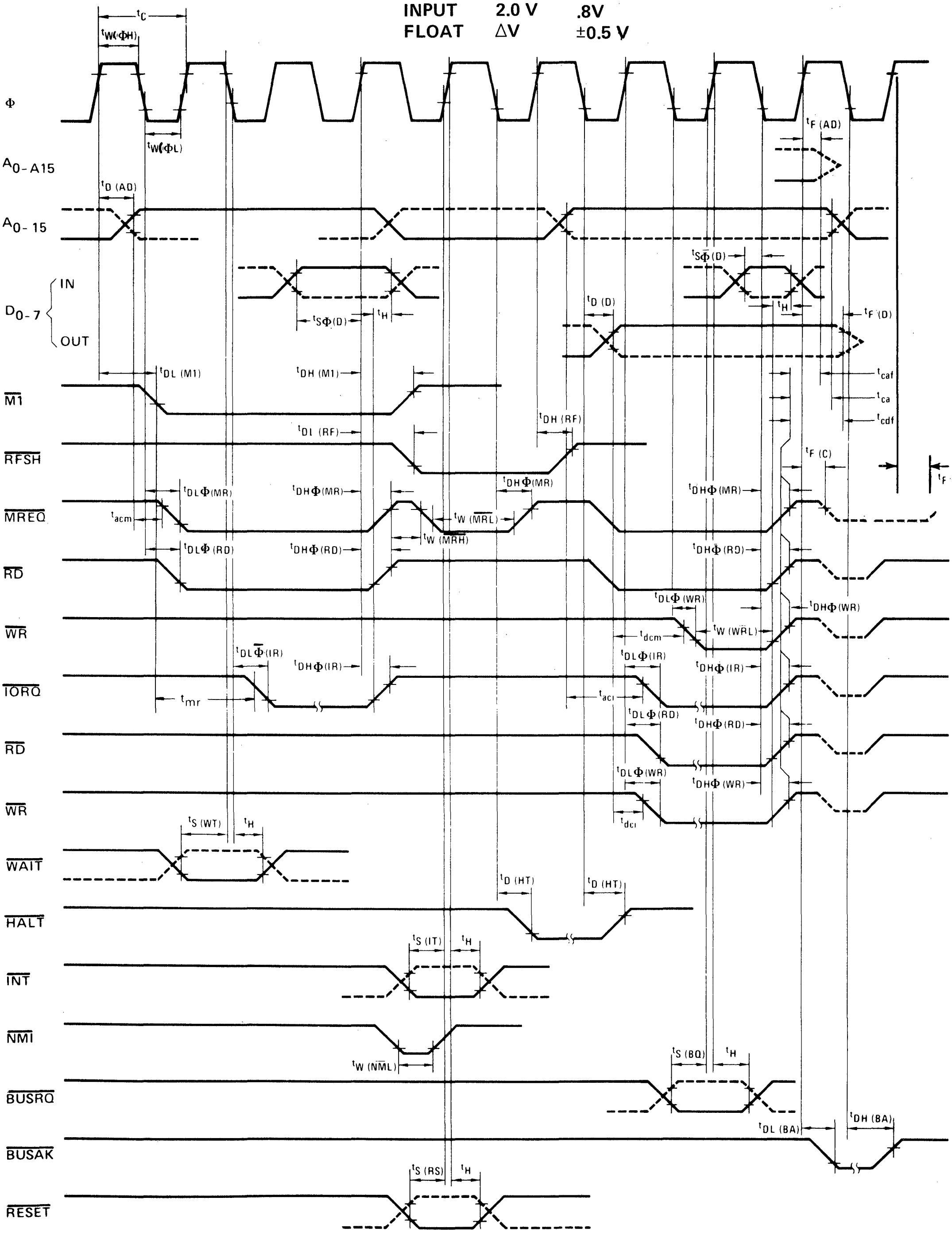
C. Output Delay vs. Load Capacitance
$T_A = 70^\circ C$ $V_{CC} = 5V \pm 5\%$
Add 10 nsec delay for each 50pF increase in load up to a maximum of 200pF for the data bus and 100pF for address and control lines

D. Although static by design, testing guarantees $t_W(\Phi H)$ of 200 μsec maximum.

A.C. TIMING DIAGRAM

Timing measurements are made at the following voltages, unless otherwise specified:

	"1"	"0"
CLOCK	$V_{CC}-.6$	.8V
OUTPUT	2.0 V	.8V
INPUT	2.0 V	.8V
FLOAT	ΔV	±0.5 V



12.0 Z80 INSTRUCTION BREAKDOWN BY MACHINE CYCLE

This section tabulates each Z80 instruction type and breaks each instruction down into its machine cycles and corresponding T States. The different standard machine cycles (OP Code Fetch, Memory Read, Port Read, etc.) are described in Section 4.0 of this manual. This chart will allow the system designer to predict what the Z80 will do on each clock cycle during the execution of a given instruction. The instruction types are listed together by functions and in the same order as the Tables in Section 7.

The best way to learn how to use these tables is to look at a few examples. The first example is to register exchange instructions (LD r, s) where r,s can be any of the following CPU Registers: B,C,D,E,H,L, or A. The instruction breakdown table shows this instruction to have one machine cycle (M1) four T-States long (number in parenthesis) which is an OP Code Fetch. Referring to Figure 4.0-1 one sees the standard form for an OP Code Fetch and the state of the CPU bus during these four T-States. Taking the next instruction shown (LD r, n) which loads one of the previous registers with data or immediate value "n" one finds the breakdown to be a four T-State OP Code Fetch followed by a three T-State Operand Data Read. An Operand Data Read takes the form of the Standard Memory Read shown in Figure 4.0-2.

After these two simple examples, a more complex one is in order. The LD r, (IX+d) is the first double byte OP Code shown and executes as follows: First there are two M1 cycles (and related memory refreshes) followed by an Operand Data Read of the displacement "d". Next M3 consists of a five T-State Internal Operation which is the calculation of the Indexed address (IX+d). The last machine cycle (M4) consists of a Memory Read of the data continued in address IX+d and the loading of register "r" with that data.

The LD dd, (nn) instruction loads an internal 16-bit register pair with the contents of the memory location specified in the Operand Bytes of the instruction. This instruction is four bytes long (two bytes of OP Code + two bytes of Operand Address). As shown, there are two M1 cycles to fetch the OP Code and then two Machine Cycles to read the Operand Addresses, low order byte first. Machine cycle 4 is a read of memory to obtain the data for the low order register (e.g., C of BC, E of DE and L of HL) followed by a read of the data for the high order register.

The first instruction to use the Stack Register is the PUSH qq instruction which executes as follows: Machine cycle 1 is extended by one cycle and the Stack Pointer is decremented in the extra T-State to point to an empty location on the Stack. Machine cycle 2 is a write of the high byte of the referenced register to the address contained in the Stack Pointer. The Stack Pointer is again decremented and a write of the low byte of the referenced register is made to the Stack in Machine Cycle 3. Note that the Stack Pointer is left pointing to the last data referenced on the Stack. The block transfer instructions such as LDI and LDIR are very similar. LDI is 16 T-States long and is composed of a double byte OP Code Fetch (two memory refreshes) followed by a memory read and a memory write. The memory write is 5 T-States long to allow updating of the block length counter —BC. The repetitive form of this instruction (LDIR) has an additional Machine Cycle (M4) of 5 T-States to allow decrementing of the Program Counter by two (PC-2) which results in refetching of the OP Code (LDIR). Each movement of data by this instruction is 21 T-States long (except the last) and the refetching of the OP Codes results in memory refresh occurring as well as the sampling of interrupts and $\overline{\text{BUSRQ}}$.

The $\overline{\text{NMI}}$ Interrupt sequence is 11 T-States long with the first M1 being a dummy OP Code Fetch of 5 T-States long. The Program Counter is not advanced, the OP Code on the data bus is ignored and an internal Restart is done to address 66H. The following two Machine Cycles are a write of the Program Counter to the Stack.

The $\overline{\text{INT}}$ Mode 0 is the 8080A mode and requires the user to place an instruction on the data bus for the CPU to execute. If a RST instruction is used, the CPU stacks the Program Counter and begins execution at the Restart Address. If a CALL instruction is used, the CALL Op Code is placed on the data bus during the INTA cycle (M1). M2 and M3 are normal Memory Read cycles (not INTA cycles) of the CALL addresses (low byte first). Program Counter is stacked in M4 and M5.

Mode 2 is used by the Z80 System Peripherals and operates as follows: During the INTA cycle (M1) a Vector is sent in from the highest priority interrupting device. M2 and M3 are used to Stack the Program Counter. The Vector (low byte) and an internal Interrupt Register (I) from a pointer to a table containing the addresses of Interrupt Service Routines. During M4 and M5 the Service Routines address is read from this table into the CPU. The next M1 cycle will fetch an OP Code from the address received is M4 and M5.

LEGEND

IO	— Internal CPU Operation
MR	— Memory Read
MRH	— Memory Read of High Byte
MRL	— Memory Read of Low Byte
MW	— Memory Write
MWH	— Memory Write of High Byte
MWL	— Memory Write of Low Byte
OCF	— Op Code Fetch
ODH	— Operand Data Read of High Byte
ODL	— Operand Data Read of Low Byte
PR	— Port Read
PW	— Port Write
SRH	— Stack Read of High Byte
SRL	— Stack Read of Low Byte
SWH	— Stack Write of High Byte
SWL	— Stack Write of Low Byte
( )	— Number of T-States in that Machine Cycle

Z80 INSTRUCTION BREAKDOWN BY MACHINE CODE

		MACHINE CYCLE				
INSTRUCTION TYPE	BYTES	M1	M2	M3	M4	M5
LD r, s	1	OCF (4)				
LD r, n	2	OCF (4)	OD (3)			
LD r, (HL)	1	OCF (4)	MR (3)			
LD (HL), r		OCF (4)	MW (3)			
LD r, (IX+d)	3	OCF (4)/OCF (4)	OD (3)	IO (5)	MR (3)	
LD (IX+d), r		OCF (4)/OCF (4)	OD (3)	IO (5)	MW (3)	
LD (HL), n	2	OCF (4)	OD (3)	MW (3)		
LD A, $(^{BC}_{DE})$	1	OCF (4)	MR (3)			
LD $(^{BC}_{DE})$, A		OCF (4)	MW (3)			
LD A, (nn)	3	OCF (4)	ODL (3)	ODH (3)	MR (3)	
LD (nn) , A		OCF (4)	ODL (3)	ODH (3)	MW (3)	
LD A, $^{I}_{R}$	2	OCF (4)/OCF(5)				
LD $^{I}_{R}$, A						
LD dd, nn	3	OCF (4)	ODL (3)	ODH (3)		
LD IX, nn	4	OCF (4)/OCF (4)	ODL (3)	ODH (3)		
LD HL, (nn)	3	OCF (4)	ODL (3)	ODH (3)	MRL (3)	MRH (3)
LD (nn), HL		OCF (4)	ODL (3)	ODH (3)	MWL (3)	MWH (3)
LD dd, (nn)	4	OCF (4)/OCF (4)	ODL (3)	ODH (3)	MRL (3)	MRH (3)
LD (nn), dd		OCF (4)/OCF (4)	ODL (3)	ODH (3)	MWL (3)	MWH (3)
LD IX, (nn)		OCF (4)/OCF (4)	ODL (3)	ODH (3)	MRL (3)	MRH (3)
LD (nn), IX		OCF (4)/OCF (4)	ODL (3)	ODH (3)	MWL (3)	MWH (3)
LD SP, HL	1	OCF (6)				
LD SP, IX	2	OCF (4)/OCF (6)				
PUSH qq	1	OCF (5) SP-1 →	SWH (3) SP-1 →	SWL (3)		
PUSH IX	2	OCF (4)/OCF (5) SP-1 →	SWH (3) SP-1 →	SWL (3)		
POP qq	1	OCF (4)	SRH (3) SP+1 →	SRL (3)	SP+1 →	
POP IX	2	OCF (4)/OCF (4)	SRH (3) SP+1 →	SRL (3)	SP+1 →	
EX DE, HL	1	OCF (4)				
EX AF, AF'	1	OCF (4)				

		MACHINE CYCLE				
INSTRUCTION TYPE	BYTES	M1	M2	M3	M4	M5
EXX	1	OCF (4)				
EX (SP), HL	1	OCF (4)	SRL (3) SP+1 →	SRH (4)	SWH (3) SP-1 →	SWL (5)
EX (SP), IX	2	OCF (4)/OCF (4)	SRL (3) SP+1 →	SRH (4)	SWH (3) SP-1 →	SWL (5)
LDI LDD CPI CPD	2	OCF (4)/OCF (4)	MR (3)	MW (5)		
LDIR LDDR CPIR CPDR	2	OCF (4)/OCF (4)	MR (3)	MW (5)	IO (5)* *only if BC ≠ 0	
ALU A, r ADD ADC SUB SBC AND OR XOR CP	1	OCF (4)				
ALU A, n	2	OCF (4)	OD (3)			
ALU A, (HL)	1	OCF (4)	MR (3)			
ALU A, (IX+d)	3	OCF (4)/OCF (4)	OD (3)	IO (5)	MR (3)	
DEC INC r	1	OCF (4)				
DEC INC (HL)	1	OCF (4)	MR (4)	MW (3)		
DEC INC (IX+D)	2	OCF (4)/OCF (4)	OD (3)	IO (5)	MR (4)	MW (3)
DAA CPL CCF SCF NOP HALT DI EI	1	OCF (4)				
NEG IMO IM1 IM2	2	OCF (4)/OCF (4)				

		MACHINE CYCLE				
INSTRUCTION TYPE	BYTES	M1	M2	M3	M4	M5
ADD HL, ss	1	OCF (4)	IO (4)	IO (3)		
ADC HL, ss	2	OCF (4)/OCF (4)	IO (4)	IO (3)		
SBC HL, ss						
ADD IX, pp						
INC ss	1	OCF (6)				
DEC ss						
DEC IX	2	OCF (4)/OCF (6)				
INC IX						
RLCA	1	OCF (4)				
RLA						
RRCA						
RRA						
RLC r	2	OCF (4)/OCF (4)				
RL						
RRC						
RR						
SLA						
SRA						
SRL						
RLC (HL)	2	OCF (4)/OCF (4)	MR (4)	MW (3)		
RL						
RRC						
RR						
SLA						
SRA						
SRL						
RLC (IX+d)	4	OCF (4)/OCF (4)	OD (3)	IO (5)	MR (4)	MW (3)
RL						
RRC						
RR						
SLA						
SRA						
SRL						
RLD	2	OCF (4)/OCF (4)	MR (3)	IO (4)	MW (3)	
RRD						
BIT b, r	2	OCF (4)/OCF (4)				
SET						
RES						

		MACHINE CYCLE				
INSTRUCTION TYPE	BYTES	M1	M2	M3	M4	M5
BIT b, (HL)	2	OCF (4)/OCF (4)	MR (4)			
SET b, (HL) RES	2	OCF (4)/OCF (4)	MR (4)	MW (3)		
BIT b, (IX+d)	4	OCF (4)/OCF (4)	OD (3)	IO (5)	MR (4)	
SET b, (IX+d) RES	4	OCF (4)/OCF (4)	OD (3)	IO (5)	MR (4)	MW (3)
JP nn JP cc, nn	3	OCF (4)	ODL (3)	ODH (3)		
JR e	2	OCF (4)	OD (3)	IO (5)		
JR C, e JR NC, e JR Z, e JR NZ, e	2	OCF (4)	OD (3)	IO (5)* * If condition is met		
JP (HL)	1	OCF (4)				
JP (IX)	2	OCF (4)/OCF (4)				
DJNZ, e	2	OCF (5)	OD (3)	IO (5)* * If B≠ 0		
CALL nn CALL cc, nn cc true	3	OCF (4)	ODL (3)	ODH (4) SP-1 →	SWH (3) SP-1 →	SWL (3)
CALL cc, nn cc false	3	OCF (4)	ODL (3)	ODH (3)		
RET	1	OCF (4)	SRL (3) SP+1 →	SRH (3) →	SP+1 →	
RET cc	1	OCF (5)	SRL (3)* * If cc is true SP+1 →	SRH (3)* →	SP+1 →	
RETI RETN	2	OCF (4)/OCF (4)	SRL (3) SP+1 →	SRH (3) →	SP+1 →	
RST p	1	OCF (5) SP-1 →	SWH (3) SP-1 →	SWL (3)		

		MACHINE CYCLE				
INSTRUCTION TYPE	BYTES	M1	M2	M3	M4	M5
IN A, (n)	2	OCF (4)	OD (3)	PR (4)		
IN r, (c)	2	OCF (4)/OCF (4)	PR (4)			
INI IND	2	OCF (4)/OCF (5)	PR (4)	MW (3)		
INIR INDR	2	OCF (4)/OCF (5)	PR (4)	MW (3)	IO (5)	
OUT (n) , A	2	OCF (4)	OD (3)	PW (4)		
OUT (C), r	2	OCF (4)/OCF (4)	PW (4)			
OUTI OUTD	2	OCF (4)/OCF (5)	MR (3)	PW (4)		
OTIR OTDR	2	OCF (4)/OCF (5)	MR (3)	PW (4)	IO (5)	
INTERRUPTS						
NMI	–	OCF (5) * SP-1 →	SWH (3) SP-1 →	SWL (3)	*Op Code Ignored	
INT						
MODE 0	–	INTA (6) (CALL INSERTED)	ODL (3)	ODH (4) SP-1 →	SWH (3) SP-1 →	SWL (3)
	–	INTA (6) (RST INSERTED) SP-1 →	SWH (3) SP-1 →	SWL (3)		
MODE 1	–	INTA (7) (RST 38H INTERNAL) SP-1 →	SWH (3) SP-1 →	SWL (3)		
MODE 2	–	INTA (7) (VECTOR SUPPLIED) SP-1 →	SWH (3) SP-1 →	SWL (3)	MRL (3)	MRH (3)

ORDERING INFORMATION

PART NO.	PACKAGE TYPE	MAX CLOCK FREQUENCY	TEMPERATURE RANGE
MK3880N Z80-CPU	Plastic	2.5 MHz	0° to +70°C
MK3880P Z80-CPU	Ceramic	2.5 MHz	
MK3880J Z80-CPU	Cerdip	2.5 MHz	
MK3880N-4 Z80-CPU	Plastic	4.0 MHz	
MK3880P-4 Z80-CPU	Ceramic	4.0 MHz	
MK3880J-4 Z80-CPU	Cerdip	4.0 MHz	
MK3880P-10 Z80-CPU	Ceramic	2.5 MHz	-40°C to +85°C

MOSTEK®

Z80 MICROCOMPUTER DEVICES

Technical Manual

III
Z80 FAMILY
TECHNICAL
MANUALS

MK3881
PARALLEL I/O
CONTROLLER

TABLE OF CONTENTS

SECTION PAGE

1.0 INTRODUCTION

The Z80 Parallel I/O Circuit is a programmable, two port device which provides a TTL compatible interface between peripheral devices and the Z80-CPU. The CPU can configure the Z80-PIO to interface with a wide range of peripheral devices with no other external logic required. Typical peripheral devices that are fully compatible with the Z80-PIO include most keyboards, paper tape readers and punches, printers, PROM programmers, etc. The Z80-PIO utilizes N channel silicon gate depletion load technology and is packaged in a 40 pin DIP. Major features of the Z80-PIO include:

- Two independent 8 bit bidirectional peripheral interface ports with 'handshake' data transfer control
- Interrupt driven 'handshake' for fast response
- Any one of four distinct modes of operation may be selected for a port including:

  Byte output
  Byte input
  Byte bidirectional bus (Available on Port A only)
  Bit control mode
  All with interrupt controlled handshake
- Daisy chain priority interrupt logic included to provide for automatic interrupt vectoring without external logic
- Eight outputs are capable of driving Darlington transistors
- All inputs and outputs fully TTL compatible
- Single 5 volt supply and single phase clock required.

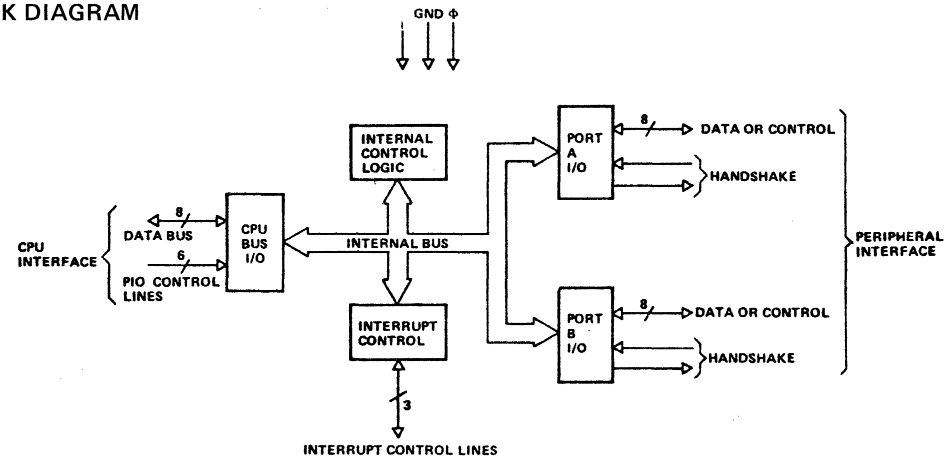
One of the unique features of the Z80-PIO that separates it from other interface controllers is that all data transfer between the peripheral device and the CPU is accomplished under total interrupt control. The interrupt logic of the PIO permits full usage of the efficient interrupt capabilities of the Z80-CPU during I/O transfers. All logic necessary to implement a fully nested interrupt structure is included in the PIO so that additional circuits are not required. Another unique feature of the PIO is that it can be programmed to interrupt the CPU on the occurrence of specified status conditions in the peripheral device. For example, the PIO can be programmed to interrupt if any specified peripheral alarm conditions should occur. This interrupt capability reduces the amount of time that the processor must spend in polling peripheral status.

III
Z80 FAMILY
TECHNICAL
MANUALS

2.0 PIO ARCHITECHTURE

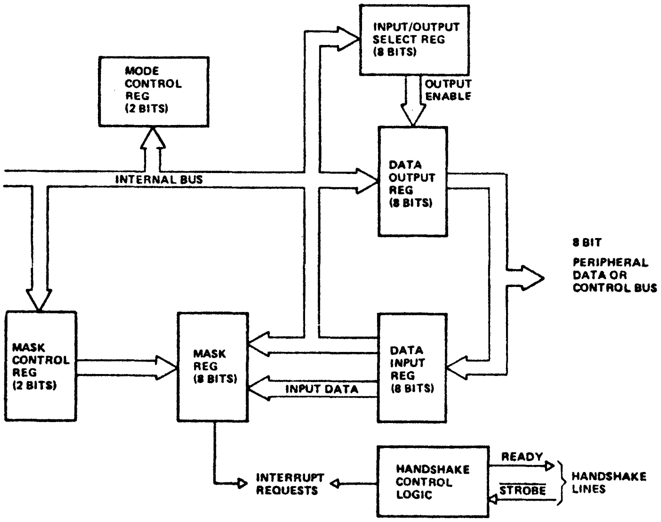
A block diagram of the Z80-PIO is shown in figure 2.0-1. The internal structure of the Z80-PIO consists of a Z80-CPU bus interface, internal control logic, Port A I/O logic, Port B I/O logic, and interrupt control logic. The CPU bus interface logic allows the PIO to interface directly to the Z80-CPU with no other external logic. However, address decoders and/or line buffers may be required for large systems. The internal control logic synchronizes the CPU data bus to the peripheral device interfaces (Port A and Port B). The two I/O ports (A and B) are virtually identical and are used to interface directly to peripheral devices.

PIO BLOCK DIAGRAM
Figure 2.0-1



The Port I/O logic is composed of 6 registers with "handshake" control logic as shown in figure 2.0-2. The registers include: an 8 bit data input register, an 8 bit data output register, a 2 bit mode control register, an 8 bit mask register, an 8 bit input/output select register, and a 2 bit mask control register.

PORT I/O BLOCK DIAGRAM
Figure 2.0-2



The 2-bit mode control register is loaded by the CPU to select the desired operating mode (byte output, byte input, byte bidirectional bus, or bit control mode). All data transfer between the peripheral device and the CPU is achieved through the data input and data output registers. Data may be written into the output register by the CPU or read back to the CPU from the input register at any time. The handshake lines associated with each port are used to control the data transfer between the PIO and the peripheral device.

The 8-bit mask register and the 8-bit input/output select register are used only in the bit control mode. In this mode any of the 8 peripheral data or control bus pins can be programmed to be an input or an output as specified by the select register. The mask register is used in this mode in conjunction with a special interrupt feature. This feature allows an interrupt to be generated when any or all of the unmasked pins reach a specified state (either high or low). The 2-bit mask control register specifies the active state desired (high or low) and if the interrupt should be generated when all unmasked pins are active (AND condition) or when any unmasked pin is active (OR condition). This feature reduces the requirement for CPU status checking of the peripheral by allowing an interrupt to be automatically generated on specific peripheral status conditions. For example, in a system with 3 alarm conditions, an interrupt may be generated if any one occurs or if all three occur.

The interrupt control logic section handles all CPU interrupt protocol for nested priority interrupt structures. The priority of any device is determined by its physical location in a daisy chain configuration. Two lines are provided in each PIO to form this daisy chain. The device closest to the CPU has the highest priority. Within a PIO, Port A interrupts have higher priority than those of Port B. In the byte input, byte output or bidirectional modes, an interrupt can be generated whenever a new byte transfer is requested by the peripheral. In the bit control mode an interrupt can be generated when the peripheral status matches a programmed value. The PIO provides for complete control of nested interrupts. That is, lower priority devices may not interrupt higher priority devices that have not had their interrupt service routine completed by the CPU. Higher priority devices may interrupt the servicing of lower priority devices.

When an interrupt is accepted by the CPU in mode 2, the interrupting device must provide an 8-bit interrupt vector for the CPU. This vector is used to form a pointer to a location in the computer memory where the address of the interrupt service routine is located. The 8-bit vector from the interrupting device forms the least significant 8 bits of the indirect pointer while the I Register in the CPU provides the most significant 8 bits of the pointer. Each port (A and B) has an independent interrupt vector. The least significant bit of the vector is automatically set to a 0 within the PIO since the pointer must point to two adjacent memory locations for a complete 16-bit address.

The PIO decodes the RETI (Return from interrupt) instruction directly from the CPU data bus so that each PIO in the system knows at all times whether it is being serviced by the CPU interrupt service routine without any other communication with the CPU.

3.0 PIN DESCRIPTION

A diagram of the Z80-PIO pin configuration is shown in figure 3.0-1. This section describes the function of each pin.

D_7-D_0

Z80-CPU Data Bus (bidirectional, tristate)
This bus is used to transfer all data and commands between the Z80-CPU and the Z80-PIO. D_0 is the least significant bit of the bus.

B/$\overline{A}$ Sel

Port B or A Select (input, active high)
This pin defines which port will be accessed during a data transfer between the Z80-CPU and the Z80-PIO. A low level on this pin selects Port A while a high level selects Port B. Often Address bit A_0 from the CPU will be used for this selection function.

C/$\overline{D}$ Sel

Control or Data Select (input, active high)
This pin defines the type of data transfer to be performed bwtween the CPU and the PIO. A high level on this pin during a CPU write to the PIO causes the Z80 data bus to be interpreted as a command for the port selected by the B/A Select line. A low level on this pin means that the Z80 data bus is being used to transfer data between the CPU and the PIO. Often Address bit A_1 from the CPU will be used for this function.

$\overline{CE}$

Chip Enable (input, active low)
A low level on this pin enables the PIO to accept command or data inputs from the CPU during a write cycle or to transmit data to the CPU during a read cycle. This signal is generally a decode of four I/O port numbers that encompass port A and B, data and control.

Φ

System Clock(input)
The Z80-PIO uses the standard Z80 system clock to synchronize certain signals internally. This is a single phase clock.

$\overline{M1}$

Machine Cycle One Signal from CPU (input, active low)
This signal from the CPU is used as a sync pulse to control several internal PIO operations. When $\overline{M1}$ is active and the $\overline{RD}$ signal is active, the Z80-CPU is fetching an instruction from memory. Conversely, when $\overline{M1}$ is active and $\overline{IORQ}$ is active, the CPU is acknowledging an interrupt. In addition, the $\overline{M1}$ signal has two other functions within the Z80-PIO.

1. $\overline{M1}$ synchronizes the PIO interrupt logic.

2. When $\overline{M1}$ occurs without an active $\overline{RD}$ or $\overline{IORQ}$ signal the PIO logic enters a reset state.

$\overline{IORQ}$

Input/Output Request from Z80-CPU (input, active low)
The $\overline{IORQ}$ signal is used in conjunction with the B/A Select, C/D Select, $\overline{CE}$, and $\overline{RD}$ signals to transfer commands and data between the Z80-CPU and the Z80-PIO. When $\overline{CE}$, $\overline{RD}$ and $\overline{IORQ}$ are active, the port addressed by B/A will transfer data to the CPU (a read operation). Conversely, when $\overline{CE}$ and $\overline{IORQ}$ are active but $\overline{RD}$ is not active, then the port addressed by B/A will be written into from the CPU with either data or control information as specified by the C/D Select signal. Also, if $\overline{IORQ}$ and $\overline{M1}$ are active simultaneously, the CPU is acknowledging an interrupt and the interrupting port will automatically place its interrupt vector on the CPU data bus if it is the highest device requesting an interrupt.

$\overline{\text{RD}}$

Read Cycle Status from the Z80-CPU (input, active low)
If $\overline{\text{RD}}$ is active a MEMORY READ or I/O READ operation is in progress. The $\overline{\text{RD}}$ signal is used with B/A Select, C/D Select, $\overline{\text{CE}}$ and $\overline{\text{IORQ}}$ signals to transfer data from the Z80-PIO to the Z80-CPU.

IEI

Interrupt Enable In (input, active high)
This signal is used to form a priority interrupt daisy chain when more than one interrupt driven device is being used. A high level on this pin indicates that no other devices of higher priority are being serviced by a CPU interrupt service routine.

IEO

Interrupt Enable Out (output, active high)
The IEO signal is the other signal required to form a daisy chain priority scheme. It is high only if IEI is high and the CPU is not servicing an interrupt from this PIO. Thus this signal blocks lower priority devices from interrupting while a higher priority device is being serviced by its CPU interrupt service routine.

$\overline{\text{INT}}$

Interrupt Request (output, open drain, active low)
When INT is active the Z80-PIO is requesting an interrupt from the Z80-CPU.

A_0-A_7

Port A Bus (bidirectional, tri-state)
This 8 bit bus is used to transfer data and/or status or control information between Port A of the Z80-PIO and a peripheral device. A_0 is the least significant bit of the Port A data bus.

$\overline{\text{A STB}}$

Port A Strobe Pulse from Peripheral Device (input, active low)
The meaning of this signal depends on the mode of operation selected for Port A as follows:

1) Output mode: The positive edge of this strobe is issued by the peripheral to acknowledge the receipt of data made available by the PIO.

2) Input mode: The strobe is issued by the peripheral to load data from the peripheral into the Port A input register. Data is loaded into the PIO when this signal is active.

3) Bidirectional mode: When this signal is active, data from the Port A output register is gated onto Port A bidirectional data bus. The positive edge of the strobe acknowledges the receipt of the data.

4) Control mode: The strobe is inhibited internally.

A RDY

Register A Ready (output, active high)
The meaning of this signal depends on the mode of operation selected for Port A as follows:

1) Output mode: This signal goes active to indicate that the Port A output register has been loaded and the peripheral data bus is stable and ready for transfer to the peripheral device.

2) Input mode: This signal is active when the Port A input register is empty and is ready to accept data from the peripheral device.

3) Bidirectional mode: This signal is active when data is available in Port A output register for transfer to the peripheral device. In this mode data is not placed on the Port A data bus unless $\overline{\text{A STB}}$ is active.

4) Control mode: This signal is disabled and forced to a low state.

B_0-B_7

Port B Bus (bidirectional, tristate)
This 8 bit bus is used to transfer data and/or status or control information between Port B of the PIO and a peripheral device. The Port B data bus is capable of supplying 1.5ma@ 1.5V to drive Darlington transistors. B_0 is the least significant bit of the bus.

$\overline{\text{B STB}}$

Port B Strobe Pulse from Peripheral Device (input, active low)
The meaning of this signal is similar to that of $\overline{\text{A STB}}$ with the following exception:

In the Port A bidirectional mode this signal strobes data from the peripheral device into the Port A input register.

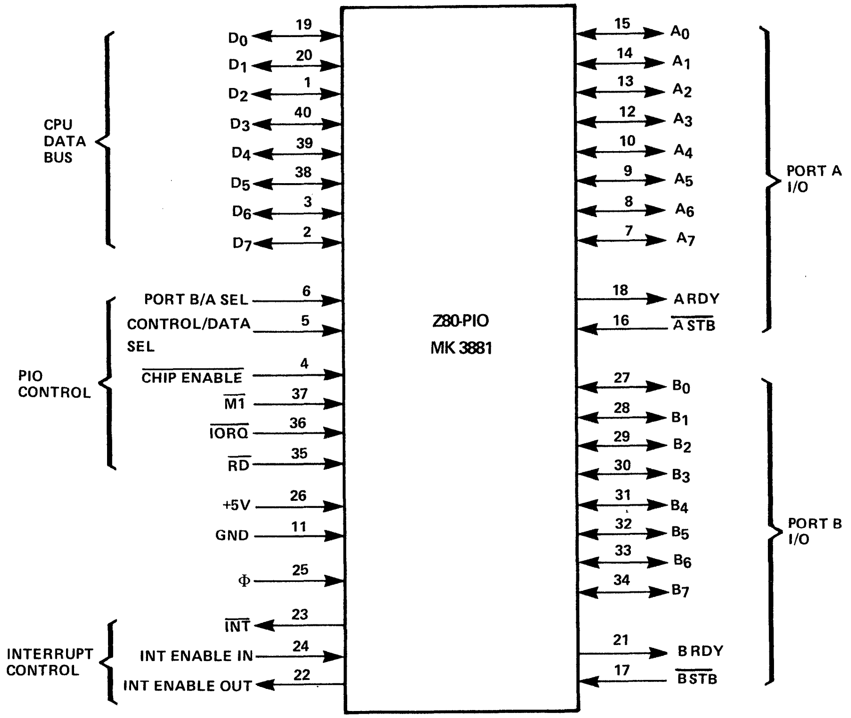
B RDY

Register B Ready (output, active high)
The meaning of this signal is similar to that of A Ready with the following exception:

In the Port A bidirectional mode this signal is high when the Port A input register is empty and ready to accept data from the peripheral device.

PIO PIN CONFIGURATION

Figure 3.0-1



4.0 PROGRAMMING THE PIO

4.1 RESET

The Z80-PIO automatically enters a reset state when power is applied. The reset state performs the following functions:

1) Both port mask registers are reset to inhibit all port data bits.

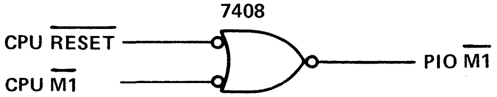
2) Port data bus lines are set to a high impedance state and the Ready "handshake" signals are inactive (low). Mode 1 is automatically selected.

3) The vector address registers are not reset.

4) Both port interrupt enable flip flops are reset.

5) Both port output registers are reset.

In addition to the automatic power on reset, the PIO can be reset by applying an $\overline{\text{M1}}$ signal without the presence of a $\overline{\text{RD}}$ or $\overline{\text{IORQ}}$ signal. If no $\overline{\text{RD}}$ or $\overline{\text{IORQ}}$ is detected during M1 the PIO will enter the reset state immediately after the $\overline{\text{M1}}$ signal goes inactive. The purpose of this reset is to allow a single external gate to generate a reset without a power down sequence. This approach was required due to the 40 pin packaging limitation. It is recommended that in breadboard systems and final systems with a "Reset" push button that a $\overline{\text{M1}}$ reset be implemented for the PIO.


7408
CPU $\overline{\text{RESET}}$
CPU $\overline{\text{M1}}$
PIO $\overline{\text{M1}}$

A software RESET is possible as described in Section 4.4, however, use of this method during early system debug may not be desirable because of non-functional system hardware (bus buffers or memory for example).

Once the PIO has entered the internal reset state it is held there until the PIO receives a control word from the CPU.

4.2 LOADING THE INTERRUPT VECTOR

The PIO has been designed to operate with the Z80-CPU using the mode 2 interrupt response. This mode requires that an interrupt vector be supplied by the interrupting device. This vector is used by the CPU to form the address for the interrupt service routine of that port. This vector is placed on the Z80 data bus during an interrupt acknowledge cycle by the highest priority device requesting service at that time. (Refer to the Z80-CPU Technical Manual for details on how an interrupt is serviced by the CPU). The desired interrupt vector is loaded into the PIO by writing a control word to the desired port of the PIO with the following format:

D7	D6	D5	D4	D3	D2	D1	D0
V7	V6	V5	V4	V3	V2	V1	0

signifies this control word is an interrupt vector

D0 is used in this case as a flag bit which when low causes V7 thru V1 to be loaded into the vector register. At interrupt acknowledge time, the vector of the interrupting port will appear on the Z80 data bus exactly as shown in the format above.

4.3 SELECTING AN OPERATING MODE

Port A of the PIO may be operated in any of four distinct modes: Mode 0 (output mode), Mode 1 (input mode), Mode 2 (bidirectional mode), and Mode 3 (control mode). Note that the mode numbers have been selected for mnemonic significance; i.e. 0=Out, 1=In, 2=Bidirectional. Port B can operate in any of these modes except Mode 2.

The mode of operation must be established by writing a control word to the PIO in the following format:

D7	D6	D5	D4	D3	D2	D1	D0
M1	M0	X	X	1	1	1	1

X=unused bit

mode word (D7–D6)

signifies mode word to be set (D3–D0)

Bits D7 and D6 from the binary code for the desired mode according to the following table:

D7	D6	MODE
0	0	0 (output)
0	1	1 (input)
1	0	2 (bidirectional)
1	1	3 (control)

Bits D5 and D4 are ignored. Bits D3-D0 must be set to 1111 to indicate "Set Mode".

Selecting Mode 0 enables any data written to the port output register by the CPU to be enabled onto the port data bus. The contents of the output register may be changed at any time by the CPU simply by writing a new data word to the port. Also the current contents of the output register may be read back to the Z80-CPU at any time through the execution of an input instruction.

With Mode 0 active, a data write from the CPU causes the Ready handshake line of that port to go high to notify the peripheral that data is available. This signal remains high until a strobe is received from the peripheral. The rising edge of the strobe generates an interrupt (if it has been enabled) and causes the Ready line to go inactive. This very simple handshake is similar to that used in many peripheral devices.

Selecting Mode 1 puts the port into the input mode. To start handshake operation, the CPU merely performs an input read operation from the port. This activates the Ready line to the peripheral to signify that data should be loaded into the empty input register. The peripheral device then strobes data into the port input register using the strobe line. Again, the rising edge of the strobe causes an interrupt request (if it has been enabled) and deactivates the Ready signal. Data may be strobed into the input register regardless of the state of the Ready signal if care is taken to prevent a data overrun condition.

Mode 2 is a bidirectional data transfer mode which uses all four handshake lines. Therefore only Port A may be used for Mode 2 operation. Mode 2 operation uses the Port A handshake signals for output control and the Port B handshake signals for input control. Thus, both A RDY and B RDY may be active simultaneously. The only operational difference between Mode 0 and the output portion of Mode 2 is that data from the Port A output register is allowed on to the port data bus only when $\overline{\text{A STB}}$ is active in order to achieve a bidirectional capability.

Mode 3 operation is intended for status and control applications and does not utilize the handshake signals. When Mode 3 is selected, the next control word sent to the PIO must define which of the port data bus lines are to be inputs and which are outputs. The format of the control word is shown below:

D7	D6	D5	D4	D3	D2	D1	D0
I/O$_7$	I/O$_6$	IO/$_5$	I/O$_4$	I/O$_3$	I/O$_2$	I/O$_1$	I/O$_0$

If any bit is set to a one, then the corresponding data bus line will be used as an input. Conversely, if the bit is reset, the line will be used as an output.

During Mode 3 operation the strobe signal is ignored and the Ready line is held low. Data may be written to a port or read from a port by the Z80-CPU at any time during Mode 3 operation. (An exception to this is when Port A is in Mode 2 and Port B is in Mode 3). When reading a port, the data returned to the CPU will be composed of input data from port data bus lines assigned as inputs plus port output register data from those lines assigned as outputs.

4.4 SETTING THE INTERRUPT CONTROL WORD

The interrupt control word for each port has the following format:

D7	D6	D5	D4	D3	D2	D1	D0
Enable Interrupt	AND/ OR	High/ Low	Masks follows	0	1	1	1
	used in Mode 3 only			signifies interrupt control word			

If bit D7=1 the interrupt enable flip flop of the port is set and the port may generate an interrupt. If bit D7=0 the enable flag is reset and interrupts may not be generated. If an interrupt occurs while D7=0, it will be latched internally by the PIO and passed onto the CPU when PIO Interrupts are Re-Enabled (D7=1). Bits D6, D5 and D4 are used mainly with Mode 3 operation, however, setting bit D4 of the interrupt control word during any mode of operation will cause a pending interrupt to be reset. These three bits are used to allow for interrupt operation in Mode 3 when any group of the I/O lines go to certain defined states. Bit D6 (AND/OR) defines the logical operation to be performed in port monitoring. If bit D6=1, and AND function is specified and if D6=0, an OR function is specified. For example, if the AND function is specified, all bits must go to a specified state before an interrupt will be generated while the OR function will generate an interrupt if any specified bit goes to the active state.

Bit D5 defines the active polarity of the port data bus line to be monitored. If bit D5=1 the port data lines are monitored for a high state while if D5=0 they will be monitored for a low state.

If bit D4=1 the next control word sent to the PIO must define a mask as follows:

D7	D6	D5	D4	D3	D2	D1	D0
MB_7	MB_6	MB_5	MB_4	MB_3	MB_2	MB_1	MB_0

Only those port lines whose mask bit is zero will be monitored for generating an interrupt.

The interrupt enable flip flop of a port may be set or reset without modifying the rest of the interrupt control word by using the following command:

Int Enable	X	X	X	0	0	1	1

If an external Asynchronous interrupt could occur while the processor is writing the disable word to the PIO (03H) then a system problem may occur. If interrupts are enabled in the processor it is possible that the Asynchronous interrupt will occur while the processor is writing the disable word to the PIO. The PIO will generate an INT and the CPU will acknowledge it, however, by this time, the PIO will have received the disable word and de-activated its interrupt structure. The result is that the PIO will not send in its interrupt vector during the interrupt acknowledge cycle because it is disabled and the CPU will fetch an erroneous vector resulting in a program fault. The cure for this problem is to disable interrupts within the CPU with the DI instruction just before the PIO is disabled and then re-enable interrupts with the EI instruction. This action causes the CPU to ignore any faulty interrupts produced by the PIO while it is being disabled. The code sequence would be:

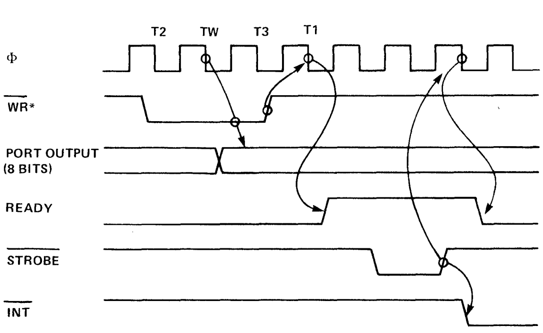
```
.
.
LD  A,03H
DI                ; DISABLE CPU
OUT (PIO),A       ; DISABLE PIO
EI                ; ENABLE CPU
.
.
```

5.0 TIMING

5.1 OUTPUT MODE (MODE 0)

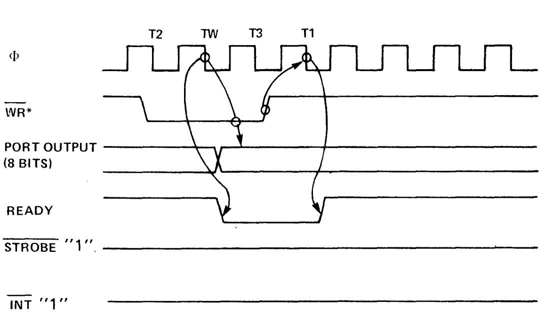
Figure 5.0-1a illustrates the timing associated with Mode 0 operation. An output cycle is always started by the execution of an output instruction by the CPU. A WR* pulse is generated by the PIO during a CPU I/O write operation and is used to latch the data from the CPU data bus into addressed port's (A or B) output register. The rising edge of the WR* pulse then raises the READY line after the next falling edge of Φ to indicate that data is available for the peripheral device. In most systems, the rising edge of the READY signal can be used as a latching signal in the peripheral device. The READY signal will remain active until a positive edge is received from the STROBE line indicating that the peripheral has taken the data shown in Figure 5.0-1a. If already active, READY will be forced low 1½ Φ cycles after the falling edge of IORQ if the port's output register is written into. READY will return high on the first falling edge of Φ after the rising edge of IORQ as shown in figure 5.0-1b. This action guarantees that READY is low while port data is changing and that a positive edge is generated on READY whenever an Output instruction is executed.

MODE 0 (OUTPUT)TIMING
Figure 5.0-1a



$\overline{WR^*} = RD \cdot \overline{CE} \cdot \overline{C/D} \cdot \overline{IORQ}$

MODE 0 (OUTPUT) TIMING
Figure 5.0-1b



$\overline{WR^*} = RD \cdot \overline{CE} \cdot \overline{C/D} \cdot \overline{IORQ}$

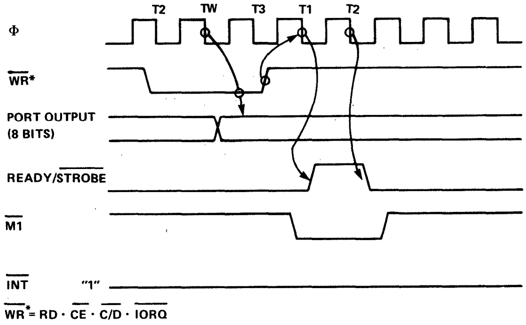
By connecting READY to STROBE a positive pulse with a duration of one clock period can be created as shown in Figure 5.0-1c. The positive edge of READY/STROBE will not generate an interrupt because the positive portion of STROBE is less than the width of M1 and as such will not generate an interrupt due to the internal logic configuration of the PIO.

If the PIO is not in a reset status (i.e. a control mode has been selected), the output register may be loaded before Mode 0 is selected. This allows port output lines to become active in a user defined state. For example, assume the outputs are desired to become active in a logic one state, the following would be the initialization sequence:

a) PIO RESET
b) Load Interrupt Vector
c) Select Mode 1 (input) (automatic due ro RESET)
d) Write FF to Data Port
e) Select Mode 0 (Outputs go to "1's")
f) Enable Interrupt if desired

MODE 0 (OUTPUT) TIMING - READY TIED TO STROBE

Figure 5.0-1c


T2
TW
T3
T1
T2
Φ
$\overline{\text{WR}}$*
PORT OUTPUT
(8 BITS)
READY/$\overline{\text{STROBE}}$
$\overline{\text{M1}}$
$\overline{\text{INT}}$ "1"
$\overline{\text{WR}}$* = RD · $\overline{\text{CE}}$ · C/$\overline{\text{D}}$ · $\overline{\text{IORQ}}$

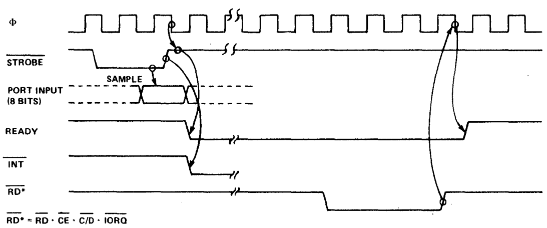
5.2 INPUT MODE (MODE 1)

Figure 5.0-2 illustrates the timing of an input cycle. The peripheral initiates this cycle using The $\overline{\text{STROBE}}$ line after the CPU has performed a data read. A low level on this line loads data into the port input register and the rising edge of the $\overline{\text{STROBE}}$ line activates the interrupt request line ($\overline{\text{INT}}$) if the interrupt enable is set and this is the highest priority requesting device. The next falling edge of the clock line (Φ) will then reset the READY line to an inactive state signifying that the input register is full and further loading must be inhibited until the CPU reads the data. The CPU will in the course of its interrupt service routine, read the data from the interrupting port. When this occurs, the positive edge from the CPU $\overline{\text{RD}}$ signal will raise the READY line with the next low going transition of Φ, indicating that new data can be loaded into the PIO.

Since RESET causes READY to go low a dummy Input instruction may be needed in some systems to cause READY to go high the first time in order to start "handshaking".

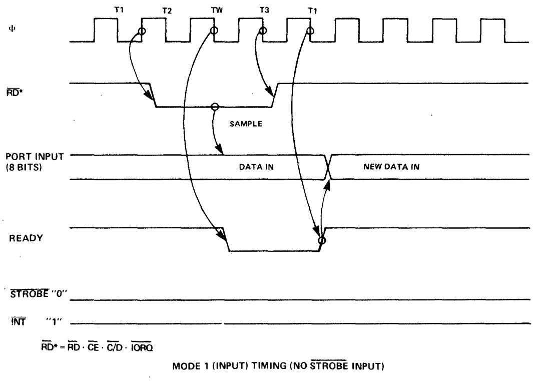
MODE 1 (INPUT) TIMING

Figure 5.0-2a


Φ
$\overline{\text{STROBE}}$
SAMPLE
PORT INPUT
(8 BITS)
READY
$\overline{\text{INT}}$
$\overline{\text{RD}}$*
$\overline{\text{RD}}$* = $\overline{\text{RD}}$ · $\overline{\text{CE}}$ · $\overline{\text{C/D}}$ · $\overline{\text{IORQ}}$

MODE 1 (INPUT) TIMING (NO STROBE INPUT)

Figure 5.0-2b


T1
T2
TW
T3
T1
Φ
$\overline{\text{RD}}$*
SAMPLE
PORT INPUT
(8 BITS)
DATA IN
NEW DATA IN
READY
$\overline{\text{STROBE}}$ "0"
$\overline{\text{INT}}$ "1"
$\overline{\text{RD}}$* = $\overline{\text{RD}}$ · $\overline{\text{CE}}$ · $\overline{\text{C/D}}$ · $\overline{\text{IORQ}}$
MODE 1 (INPUT) TIMING (NO $\overline{\text{STROBE}}$ INPUT)

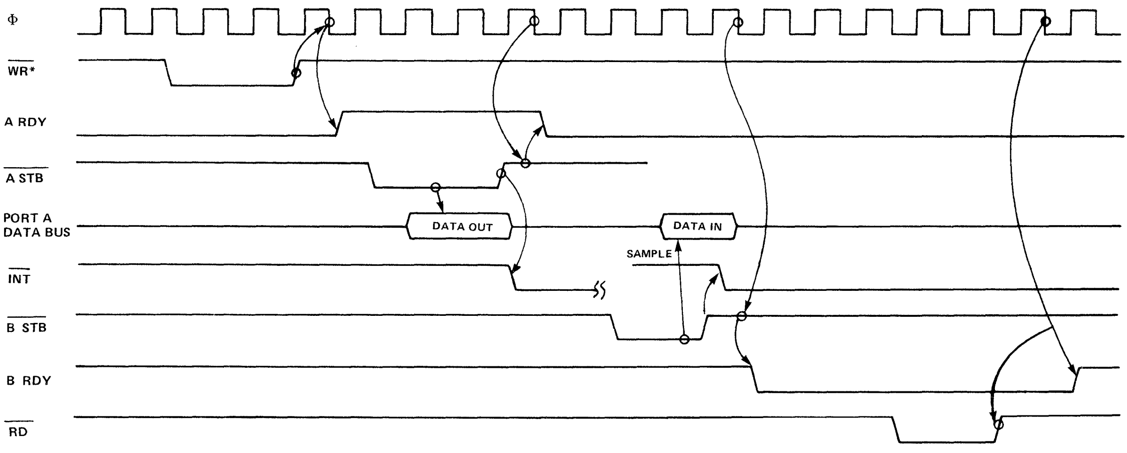
If already active, READY will be forced low one and one-half Φ periods following the falling edge of $\overline{\text{IORQ}}$ during a read of a PIO port as shown in Figure 5.0-2b. If the user strobes data into the PIO only when READY is high, the forced state of READY will prevent input register data from changing while the CPU is reading the PIO. Ready will go high again after the rising edge of the $\overline{\text{IORQ}}$ as previously described.

5.3 BIDIRECTIONAL MODE (MODE 2)

This mode is merely a combination of Mode 0 and Mode 1 using all four handshake lines. Since it requires all four lines, it is available only on Port A. When this mode is used on Port A, Port B must be set to the Bit Control Mode. The same interrupt vector will be returned for a Mode 3 interrupt on Port B and an input transfer interrupt during Mode 2 operation of Port A. Ambiguity is avoided if Port B is operated in a polled mode and the Port B mask register is set to inhibit all bits. Furthermore, interrupts from Port B (Mode 3) will not be generated when Port A is programmed for Mode 2, as $\overline{\text{BSTB}}$ would have to be active (low) in order to generate interrupts. ($\overline{\text{BSTB}}$ is normally high).

Figure 5.0-3 illustrates the timing for this mode. It is almost identical to that previously described for Mode 0 and Mode 1 with the Port A handshake lines used for output control and the Port B lines used for input control. The difference between the two modes is that, in Mode 2, data is allowed out onto the bus only when the A $\overline{\text{STROBE}}$ is low. The rising edge of this strobe can be used to latch the data into the peripheral since the data will remain stable until after this edge. The input portion of Mode 2 operates identically to Mode 1. Note that both Port A and Port B must have their interrupts enabled to achieve an interrupt driven bidirectional transfer.

PORT A, MODE 2 (BIDIRECTIONAL) TIMING
Figure 5.0-3



$\overline{\text{WR}}* = \text{RD} \cdot \overline{\text{CE}} \cdot \overline{\text{C/D}} \cdot \overline{\text{IORQ}}$
$\overline{\text{RD}}* = \overline{\text{RD}} \cdot \overline{\text{CE}} \cdot \overline{\text{C/D}} \cdot \overline{\text{IORQ}}$

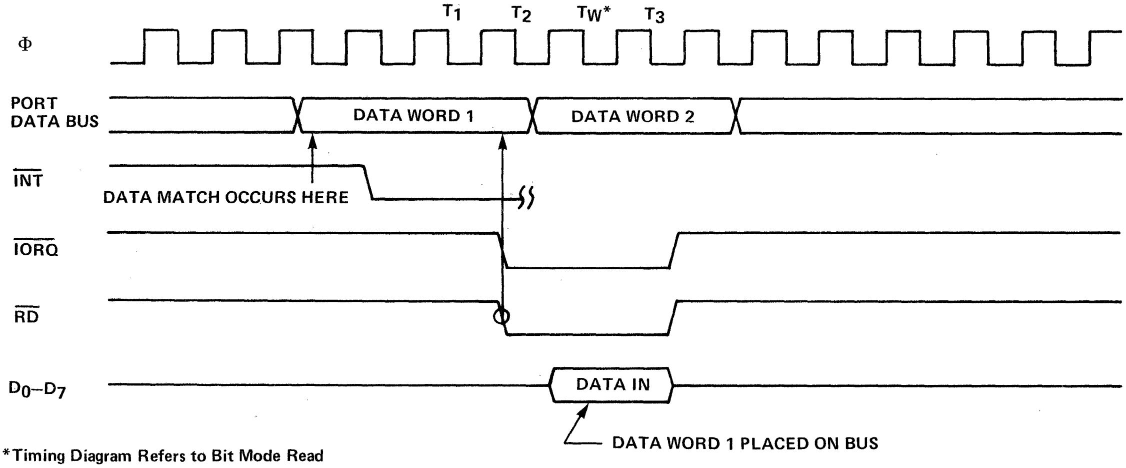
The peripheral must not gate data onto a port data bus while $\overline{\text{A STB}}$ is active. Bus contention is avoided if the peripheral uses $\overline{\text{B STB}}$ to gate input data onto the bus. The PIO uses the $\overline{\text{B STB}}$ low level to sample this data. The PIO has been designed with a zero hold time requirement for the data when latching in this mode so that this simple gating structure can be used by the peripheral. That is, the data can be disabled from the bus immediately after the strobe rising edge. Note that if $\overline{\text{A STB}}$ is low during a read operation of Port A (in response to a $\overline{\text{B STB}}$ interrupt) the data in the output register will be read by the CPU instead of the correct data in the data input register. The correct data is latched in the input register it just cannot be read by the CPU while $\overline{\text{A STB}}$ is low. If the $\overline{\text{A STB}}$ signal could go low during a CPU Read, it should be blocked from reaching the $\overline{\text{A STB}}$ input of the PIO while BRDY is low (the CPU read will occur while BRDY is low as the $\overline{\text{RD}}$ signal returns BRDY high).

5.4 CONTROL MODE (MODE 3)

The control mode does not utilize the handshake signals and a normal port write or port read can be executed at any time. When writing, the data will be latched into output registers with the same timing as Mode 0. A RDY will be forced low whenever Port A is operated in Mode 3. B RDY will be held low whenever Port B is operated in Mode 3 unless Port A is in Mode 2. In the latter case, the state of B RDY will not be affected.

When reading the PIO, the data returned to the CPU will be composed of output register data from those port data lines assigned as outputs and input register data from those port data lines assigned as inputs. The input register will contain data which was present immediately prior to the falling edge of $\overline{\text{RD}}$. See Figure 5.0-4.

MODE 3 TIMING
Figure 5.0-4a



*Timing Diagram Refers to Bit Mode Read

An interrupt will be generated if interrupts from the port are enabled and the data on the port data lines satisfies the logical equation defined by the 8-bit mask control registers. Another interrupt will not be generated until a change occurs in the status of the logical equation. A Mode 3 interrupt will be generated only if the result of a Mode 3 logical operation changes from false to true. For example, assume that the Mode 3 logical equation is an "OR" function. An unmasked port data line becomes active and an interrupt is requested. If a second unmasked port data line becomes active concurrently with the first, a new interrupt will not be requested since a change in the result of the Mode 3 logical operation has not occurred. Note that port pins defined as outputs can contribute to the logical equation if their bit positions are unmasked.

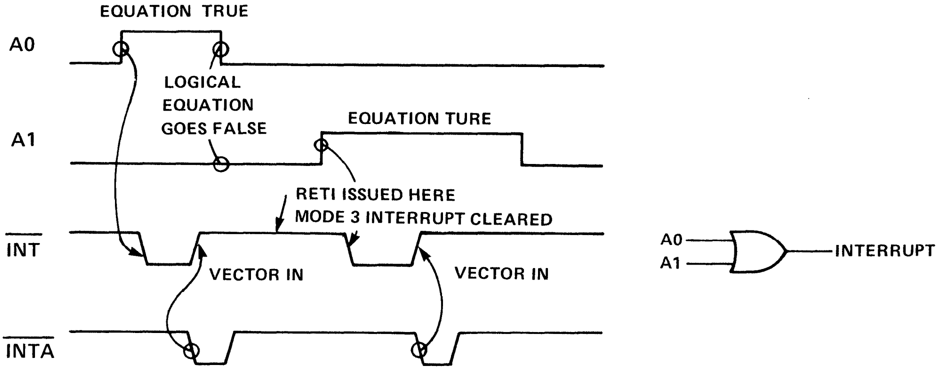
If the result of a logical operation becomes true immediately prior to or during $\overline{\text{M1}}$, an interrupt will be requested after the trailing edge of $\overline{\text{M1}}$, provided the logical equation remains true after $\overline{\text{M1}}$ returns high.

Figure 5.0-4b is an example of Mode 3 interrupts. The port has been placed in Mode 3 and OR logic selected and signals are defined to be high. All but bits A0 and A1 are masked out and are not monitored thereby creating a two input positive logic OR gate. In the timing diagram A0 is shown going high and creating an interrupt ($\overline{\text{INT}}$ goes low) and the CPU responds with an Interrupt Acknowledge cycle ($\overline{\text{INTA}}$). The PIO port with its interrupt pending sends in its Vector and the CPU goes off into the Interrupt Service Routine. A0 is shown going inactive either by itself or perhaps as a result of action taken in the Interrupt Service Routine (making the logical equation false). An arrow is shown at the point in time where the Service Routine issues the RETI instruction which clears the PIO interrupt structure. A1 is next shown going high making the logical equation-true and generating another interrupt. Two important points need to be made from this example:

1) A1 must not go high before A0 goes low or else the logical equation will not go false – a requirement fo' A1 to be able to generate an interrupt.

2) In order for A1 to generate an interrupt it must be high after the RETI issued by A0's Service Routine clears the PIO's Interrupt structure. In other words, if A1 were a positive pulse that occurred after A0 went low (to make the equation false) and went low before the RETI had cleared the Interrupt Structure it would have been missed. The logic equation must become false after the INTA for A0's service and then must be true or go true after RETI clears the previous interrupt for another interrupt to occur.

MODE 3 EXAMPLE
Figure 5.0-4b



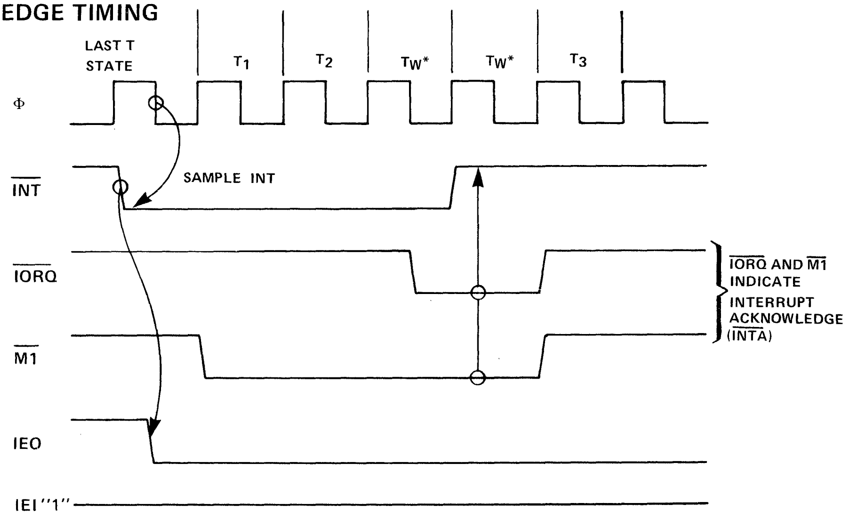
6.0 INTERRUPT SERVICING

Some time after an interrupt is requested by the PIO, the CPU will send out an interrupt acknowledge ($\overline{M1}$ and $\overline{IORQ}$). During this time the interrupt logic of the PIO will determine the highest priority port which is requesting an interrupt. (This is simply the device with its Interrupt Enable Input high and its Interrupt Enable Output low). To insure that the daisy chain enable lines stabilize, devices are inhibited from changing their interrupt request status when $\overline{M1}$ is active. The highest priority device places the contents of its interrupt vector register onto the Z80 data bus during interrupt acknowledge.

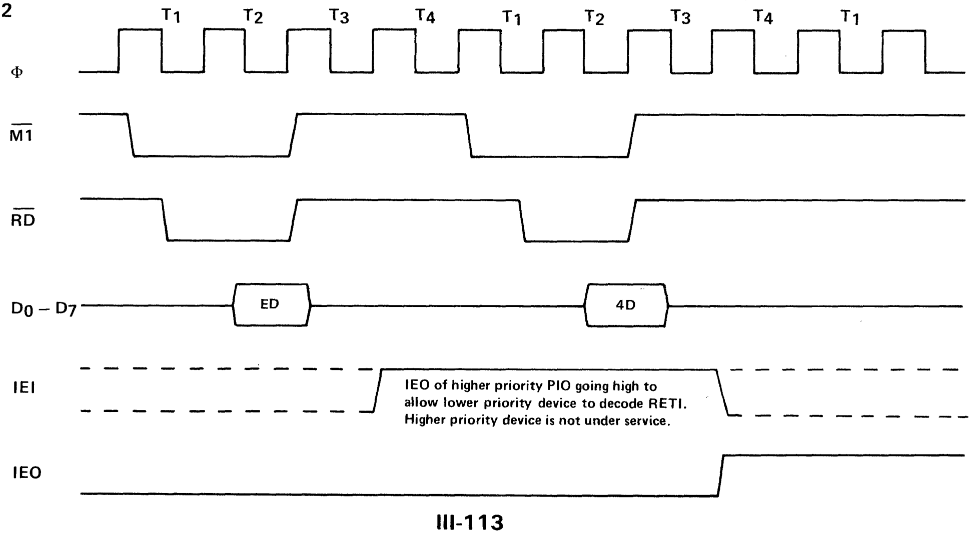
Figure 6.0-1 illustrates the timing associated with interrupt requests. During $\overline{M1}$ time, no new interrupt requests can be generated. This gives time for the Int Enable signals to ripple through up to four PIO circuits. The PIO with IEI high and IEO low during $\overline{INTA}$ will place the 8-bit interrupt vector of the appropriate port on the data bus at this time.

If an interrupt requested by the PIO is acknowledged, the requesting port is 'under service'. IEO of this port will remain low until a return from interrupt instruction (RETI) is executed while IEI of the port is high. If an interrupt request is not acknowledged, IEO will be forced high for one $\overline{M1}$ cycle after the PIO decodes the opcode 'ED'. This action guarantees that the two byte RETI instruction is decoded by the proper PIO port. See Figure 6.0-2.

INTERRUPT ACKNOWLEDGE TIMING
Figure 6.0-1

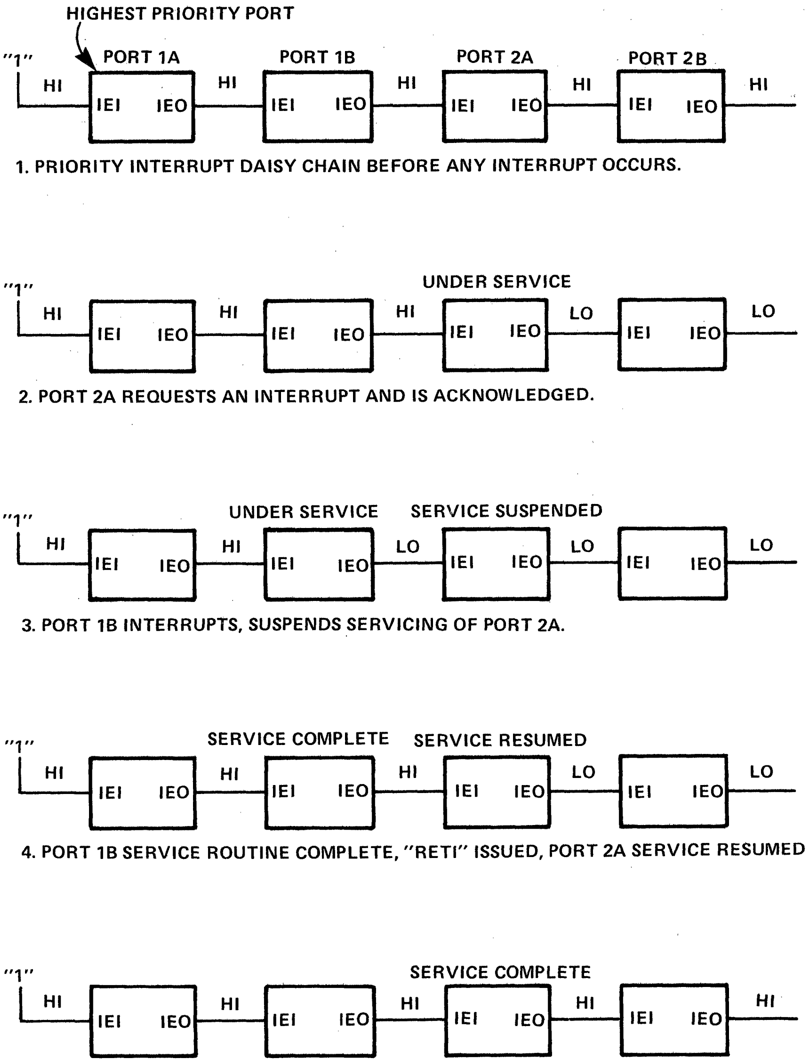


RETURN FROM INTERRUPT CYCLE
Figure 6.0-2


T1
T2
T3
T4
T1
T2
T3
T4
T1
Φ
M1
RD
D0 – D7
ED
4D
IEI
IEO of higher priority PIO going high to allow lower priority device to decode RETI. Higher priority device is not under service.
IEO

DAISY CHAIN INTERRUPT SERVICING

Figure 6.0-3



1. PRIORITY INTERRUPT DAISY CHAIN BEFORE ANY INTERRUPT OCCURS.

2. PORT 2A REQUESTS AN INTERRUPT AND IS ACKNOWLEDGED.

3. PORT 1B INTERRUPTS, SUSPENDS SERVICING OF PORT 2A.

4. PORT 1B SERVICE ROUTINE COMPLETE, "RETI" ISSUED, PORT 2A SERVICE RESUMED.

5. SECOND "RETI" INSTRUCTION ISSUED ON COMPLETION OF PORT 2A SERVICE ROUTINE.

Figure 6.0-3 illustrates a typical nested interrupt sequence that could occur with four ports connected in the daisy chain. In this sequence Port 2A requests and is granted an interrupt. While this port is being serviced, a higher priority port (1B) requests and is granted an interrupt. The service routine for the higher priority port is completed and a RETI instruction is executed to indicate to the port that its routine is complete. At this time the service routine of the lower priority port is completed.

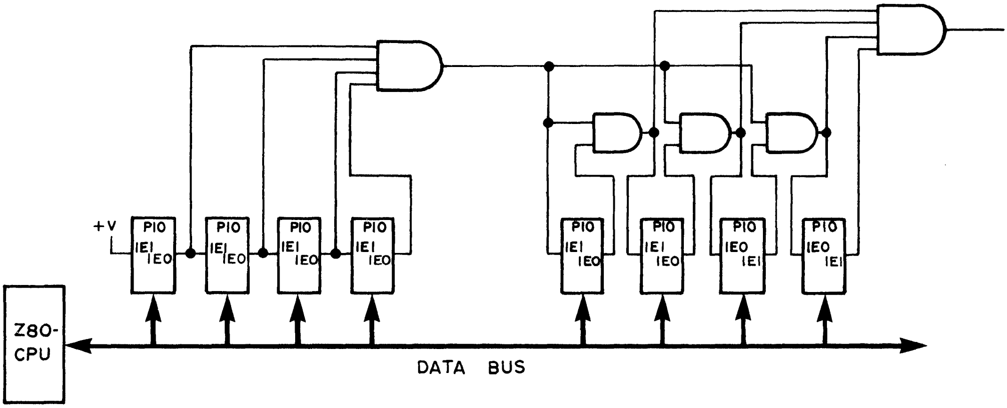
7.0 APPLICATIONS

7.1 EXTENDING THE INTERRUPT DAISY CHAIN

Without any external logic, a maximum of four Z80-PIO devices may be daisy chained into a priority interrupt structure. This limitation is required so that the interrupt enable status (IEO) ripples through the entire chain between the beginning of $\overline{M1}$, and the beginning of $\overline{IORQ}$ during an interrupt acknowledge cycle. Since the interrupt enable status cannot change during $\overline{M1}$, the vector address returned to the CPU is assured to be from the highest priority device which requested an interrupt.

If more than four PIO devices must be accommodated, a "look-ahead" structure may be used as shown in figure 7.0-1. With this technique more than thirty PIO's may be chained together using standard TTL logic.

A METHOD OF EXTENDING THE INTERRUPT PRIORITY DAISY CHAIN
Figure 7.0-1



7.2 I/O DEVICE INTERFACE

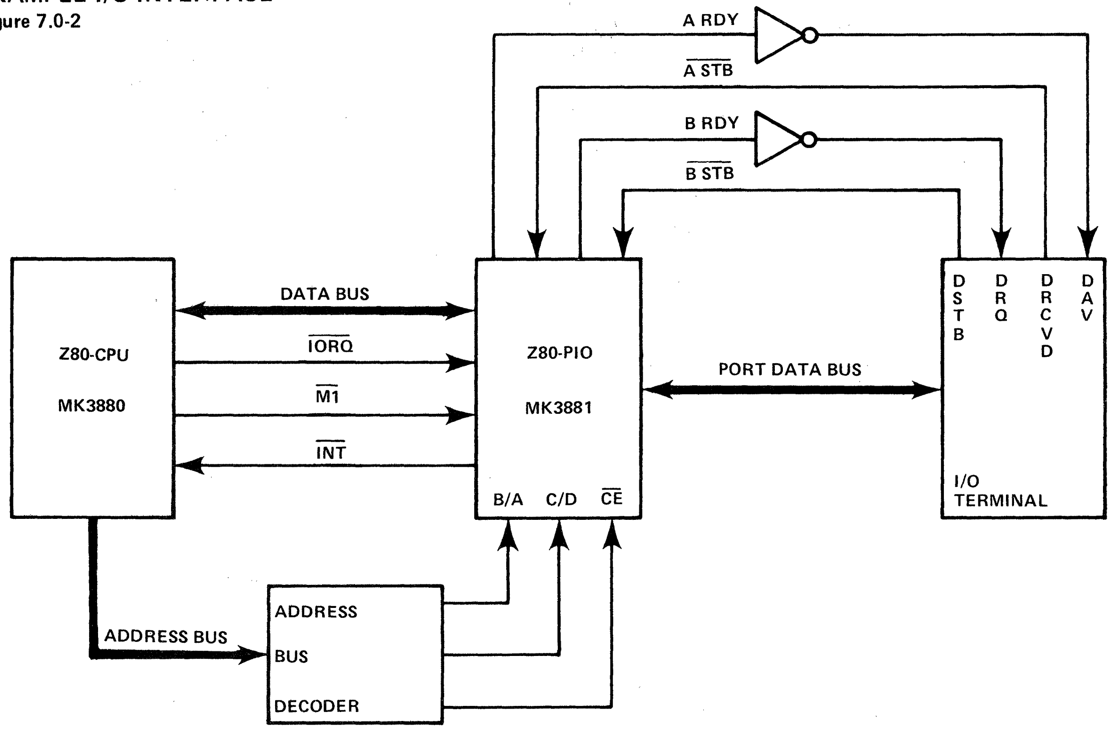
In this example, the Z80-PIO is connected to an I/O terminal device which communicates over an 8 bit parallel bidirectional data bus as illustrated in figure 7.0-2. Mode 2 operation (bidirectional) is selected by sending the following control word to Port A:

EXAMPLE I/O INTERFACE
Figure 7.0-2

D7	D6	D5	D4	D3	D2	D1	D0
1	0	X	X	1	1	1	1

$\underbrace{\text{D3 D2 D1 D0}}_{\text{MODE CONTROL}}$

EXAMPLE I/O INTERFACE
Figure 7.0-2



Next, the proper interrupt vector is loaded (refer to CPU Manual for details on the operation of the interrupt).

V7	V6	V5	V4	V3	V2	V1	0

Interrupts are then enabled by the rising edge of the first $\overline{\text{M1}}$ after the interrupt mode word is set unless that $\overline{\text{M1}}$ defines an interrupt acknowledge cycle. If a mask follows the interrupt mode word, interrupts are enabled by the rising edge of the first M1 following the setting of the mask.

Data can now be transferred between the peripheral and the CPU. The timing for this transfer is as described in Section 5.0.

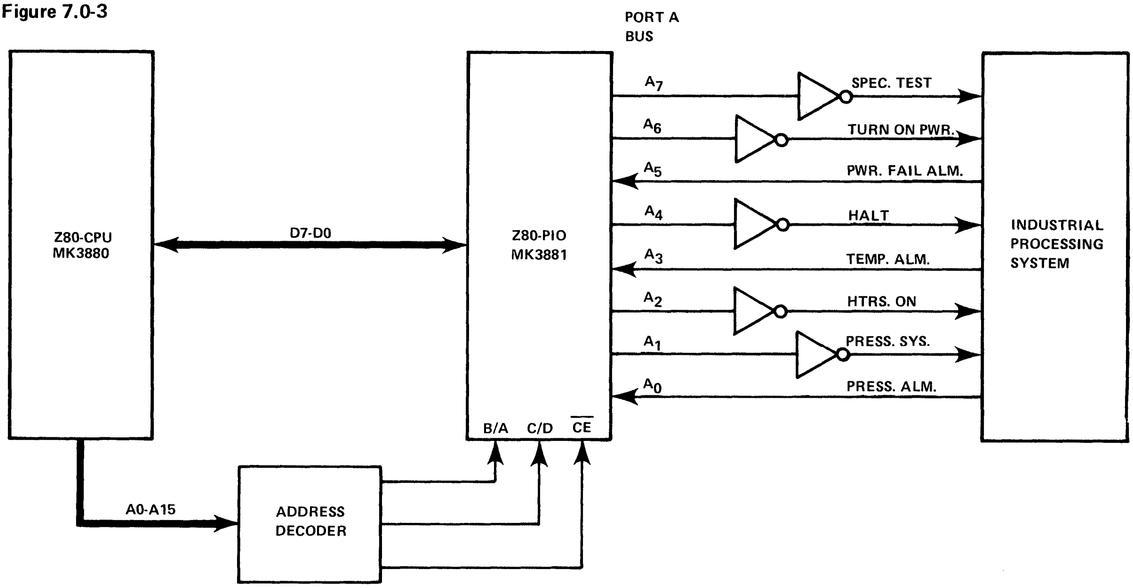
7.3 CONTROL INTERFACE

A typical control mode application is illustrated in figure 7.0-3. Suppose an industrial process is to be monitored. The occurrence of any abnormal operating condition is to be reported to a Z80-CPU based control system. The process control and status word has the following format:

D7	D6	D5	D4	D3	D2	D1	D0
Special Test	Turn On Power	Power Failure Alarm	Halt Processing	Temp. Alarm	Temp Heaters On	Pressurize System	Pressure Alarm

CONTROL MODE APPLICATION

Figure 7.0-3



The PIO may be used as follows. First Port A is set for Mode 3 operation by writing the following control word to Port A.

D7	D6	D5	D4	D3	D2	D1	D0
1	1	X	X	1	1	1	1

Whenever Mode 3 is selected, the next control word sent to the port must be an I/O select word. In this example we wish to select port data lines A5, A3, and A0 as inputs and so the following control word is written:

D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	0	1	0	0	1

Next the desired interrupt vector must be loaded (refer to the CPU manual for details);

D7	D6	D5	D4	D3	D2	D1	D0
V7	V6	V5	V4	V3	V2	V1	V0

An interrupt control word is next sent to the port:

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	1	0	1	1	1
Enable Interrupts	OR Logic	Active High	Mask Follows	Interrupt Control			

The mask word following the interrupt mode word is:

D7	D6	D5	D4	D3	D2	D1	D0
1	1	0	1	0	1	1	0

Selects A5, A3 and A0 to be monitored

Now, if a sensor puts a high level on line A5, A3, or A0, an interrupt request will be generated. The mask word may select any combination of inputs or outputs to cause an interrupt. For example, if the mask word above had been:

D7	D6	D5	D4	D3	D2	D1	D0
0	1	0	1	0	1	1	0

then an interrupt request would also occur if bit A7 (special Test) of the output register was set.

Assume that the following port assignments are to be used:

E0$_H$= Port A Data
E1$_H$= Port B Data
E2$_H$= Port A Control
E3$_H$= Port B Control

All port numbers are in hexadecimal notation. This particular assignment of port numbers is convenient since A_0 of the address bus can be used as the Port B/A Select and A_1 of the address bus can be used as the Control/Data Select. The Chip Enable would be the decode of CPU address bits A_7 thru A_2 (111000). Note that if only a few peripheral devices are being used, a Chip Enable decode may not be required since a higher order address bit could be used directly.

8.0 PROGRAMMING SUMMARY

8.1 LOAD INTERRUPT VECTOR

V7	V6	V5	V4	V3	V2	V1	0

8.2 SET MODE

M1	M0	X	X	1	1	1	1

MODE NUMBER	M_1	M_0	MODE
0	0	0	Output
1	0	1	Input
2	1	0	Bidirectional
3	1	1	Bit Control

When selecting Mode 3, the next word to the PIO must set the I/O Register:

I/O_7	I/O_6	I/O_5	I/O_4	I/O_3	I/O_2	I/O_1	I/O_0

I/O = 1 Sets bit to Input
I/O = 0 Sets bit to Output

8.3 SET INTERRUPT CONTROL

Int Enable	AND/ OR	High/ Low	Mask Follows	0	1	1	1

USED IN MODE 3 ONLY

If the "mask follows" bit is high, the next control word written to the PIO must be the mask:

MB_7	MB_6	MB_5	MB_4	MB_3	MB_2	MB_1	MB_0

MB = 0, Monitor bit
MB = 1, Mask bit from being monitored

Also, the interrupt enable flip flop of a port may be set or reset without modifying the rest of the interrupt control word by using the following command:

Int Enable	X	X	X	0	0	1	1

9.0 ELECTRICAL SPECIFICATIONS

9.1 ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias	Specified operating range.
Storage Temperature	−65°C to +150°C
Voltage On Any Pin With Respect To Ground	−0.3V to +7V
Power Dissipation	.6W

9.2 D. C. CHARACTERISTICS

Table 9.2-1

T_A = 0°C to 70°C, V_{CC} = 5 V ± 5% unless otherwise specified

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3	0.80	V	
V_{IHC}	Clock Input High Voltage	V_{CC}-.6	V_{CC}+.3	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 2.0mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -250μA
I_{CC}	Power Supply Current		70*	mA	
I_{LI}	Input Leakage Current		± 10	μA	V_{IN} = 0 to V_{CC}
I_{LOH}	Tri-State Output Leakage Current in Float		10	μA	V_{OUT}=2.4 to V_{CC}
I_{LOL}	Tri-State Output Leakage Current in Float		-10	μA	V_{OUT} = 0.4 V
I_{LD}	Data Bus Leakage Current in Input Mode		±10	μA	$0 \leq V_{IN} \leq V_{CC}$
I_{OHD}	Darlington Drive Current	-1.5		mA	V_{OH}=1.5V Port B Only

* 150mA for -4, -10, and -20 devices.

9.3 CAPACITANCE

Table 9.3-1

T_A = 25°C, f = 1 MHz

Symbol	Parameter	Max	Unit	Test Condition
C_Φ	Clock Capacitance	10	pF	Unmeasured Pins Returned to Ground
C_{IN}	Input Capacitance	5	pF	
C_{OUT}	Output Capacitance	10	pF	

*Comment

Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

9.4A A.C. CHARACTERISTICS MK3881, MK3881-10, MK3881-20, Z80-PIO

Table 9.4-1A T_A = 0°C to 70°C, V_{CC} = +5V ± 5%, unless otherwise noted

SIGNAL	SYMBOL	PARAMETER	MIN	MAX	UNIT	COMMENTS
Φ	t_c	Clock Period	400	[1]	nsec	
	$t_W(\Phi H)$	Clock Pulse Width, Clock High	170	2000	nsec	
	$t_W(\Phi L)$	Clock Pulse Width, Clock Low	170	2000	nsec	
	t_r, t_f	Clock Rise and Fall Times		30	nsec	
	t_h	Any Hold Time for Specified Set-Up Time	0		nsec	
C/D SEL $\overline{\text{CE}}$ ETC.	$t_S \Phi(CS)$	Control Signal Set-up Time to Rising Edge of Φ During Read or Write Cycle	280		nsec	
D_0-D_7	$t_{DR}(D)$	Data Output Delay from Falling Edge of RD		430	nsec	[2]
	$t_{S\Phi}(D)$	Data Set-Up Time to Rising Edge of Φ During Write or $\overline{\text{M1}}$ Cycle	50		nsec	C_L = 50pF
	$t_{DI}(D)$	Data Output Delay from Falling Edge of IORQ During $\overline{\text{INTA}}$ Cycle		340	nsec	[3]
	$t_F(D)$	Delay to Floating Bus (Output Buffer Disable Time)		160	nsec	
IEI	$t_S(IEI)$	IEI Set-Up Time to Falling Edge of $\overline{\text{IORQ}}$ During $\overline{\text{INTA}}$ Cycle	140		nsec	
IEO	$t_{DH}(IO)$	IEO Delay Time from Rising Edge of IEI		210	nsec	[5]
	$t_{DL}(IO)$	IEO Delay Time from Falling Edge of IEI		190	nsec	[5] C_L = 50pF
	$t_{DM}(IO)$	IEO Delay from Falling Edge of M1 (Interrupt Occurring Just Prior to M1) See Note A.		300	nsec	[5]
$\overline{\text{IORQ}}$	$t_S \Phi(IR)$	$\overline{\text{IORQ}}$ Set-Up Time to Rising Edge of Φ During Read or Write Cycle	250		nsec	
$\overline{\text{M1}}$	$t_S \Phi(M1)$	$\overline{\text{M1}}$ Set-Up Time to Rising Edge of Φ During $\overline{\text{INTA}}$ or $\overline{\text{M1}}$ Cycle. See Note B.	210		nsec	
$\overline{\text{RD}}$	$t_S \Phi(RD)$	$\overline{\text{RD}}$ Set-Up Time to Rising Edge of Φ During Read or $\overline{\text{M1}}$ Cycle	240		nsec	
A_0-A_7 B_0-B_7	$t_S(PD)$	Port Data Set-Up Time to Rising Edge of $\overline{\text{STROBE}}$ (Mode 1)	260		nsec	
	$t_{DS}(PD)$	Port Data Output Delay from Falling Edge of $\overline{\text{STROBE}}$ (Mode 2)		230	nsec	[5]
	$t_F(PD)$	Delay to Floating Port Data Bus from Rising Edge of $\overline{\text{STROBE}}$ (Mode 2)		200	nsec	C_L = 50pF
	$t_{DI}(PD)$	Port Data Stable from Rising Edge of $\overline{\text{IORQ}}$ During $\overline{\text{WR}}$ Cycle (Mode 0)		200	nsec	[5]
$\overline{\text{ASTB}}$ $\overline{\text{BSTB}}$	$t_W(ST)$	Pulse Width, $\overline{\text{STROBE}}$	150 [4]		nsec nsec	
$\overline{\text{INT}}$	$t_D(IT)$	$\overline{\text{INT}}$ Delay Time from Rising Edge of $\overline{\text{STROBE}}$		490	nsec	
	$t_D(IT3)$	$\overline{\text{INT}}$ Delay Time from Data Match During Mode 3 Operation		420	nsec	
ARDY BRDY	$t_{DH}(RY)$	Ready Response Time from Rising Edge of $\overline{\text{IORQ}}$		t_c+ 460	nsec	[5] C_L = 50pF
	$t_{DL}(RY)$	Ready Response Time from Rising Edge of $\overline{\text{STROBE}}$		t_c+ 400	nsec	[5]

A. $2.5t_c > (N-2)t_{DL(IO)} + t_{DM\,(IO)} + t_{S(IEI)}$ + TTL Buffer Delay, if any

B. $\overline{\text{M1}}$ must be active for a minimum of 2 clock periods to reset the PIO.

[1] $t_c = t_W(\Phi H) + t_W(\Phi L) + t_r + t_f$

[2] Increase $t_{DR(D)}$ by 10 nsec for each 50pF increase in loading up to 200pF max.

[3] Increase $t_{DI\,(D)}$ by 10 nsec for each 50pF increase in loading up to 200pF max.

[4] For Mode 2: $t_{W\,(ST)} > t_{S(PD)}$

[5] Increase these values by 2 nsec for each 10pF increase in loading up to 100pF max.

9.4B A.C. CHARACTERISTICS MK3881-4, Z80A-PIO

Table 9.4-1B T_A=0°C to 70°C, V_{CC} = +5V ± 5%, unless otherwise noted

SIGNAL	SYMBOL	PARAMETER	MIN	MAX	UNIT	COMMENTS
Φ	t_c	Clock Period	250	[1]	nsec	
	$t_W(\Phi H)$	Clock Pulse Width, Clock High	105	2000	nsec	
	$t_W(\Phi L)$	Clock Pulse Width, Clock Low	105	2000	nsec	
	t_r, t_f	Clock Rise and Fall Times		30	nsec	
	t_h	Any Hold Time for Specified Set-Up Time	0		nsec	
C/D SEL CE ETC.	$t_S\Phi(CS)$	Control Signal Set-Up Time to Rising Edge of Φ During Read or Write Cycle	145		nsec	
D_0-D_7	$t_{DR}(D)$	Data Output Delay From Falling Edge of $\overline{RD}$		380	nsec	[2]
	$t_S\Phi(D)$	Data Set-Up Time to Rising Edge of Φ During Write or $\overline{M1}$ Cycle	50		nsec	C_L = 50pF
	$t_{DI}(D)$	Data Output Delay from Falling edge of $\overline{IORQ}$ During $\overline{INTA}$ Cycle		250	nsec	[3]
	$t_F(D)$	Delay to Floating Bus (Output Buffer Disable Time)		110	nsec	
IEI	$t_S(IEI)$	IEI Set-Up Time to Falling edge of $\overline{IORQ}$ during $\overline{INTA}$ Cycle	140		nsec	
IEO	$t_{DH}(IO)$	IEO Delay Time from Rising Edge of IEI		160	nsec	[5]
	$t_{DL}(IO)$	IEO Delay Time from Falling Edge of IEI		130	nsec	[5] C_L = 50pF
	$t_{DM}(IO)$	IEO Delay from Falling Edge of $\overline{M1}$ (Interrupt Occurring Just Prior to $\overline{M1}$) See Note A.		190	nsec	[5]
$\overline{IORQ}$	$t_S\Phi(IR)$	$\overline{IORQ}$ Set-Up Time to Rising Edge of Φ During Read or Write Cycle	115		nsec	
$\overline{M1}$	$t_S\Phi(M1)$	$\overline{M1}$ Set-Up Time to Rising Edge of Φ During $\overline{INTA}$ or $\overline{M1}$ Cycle. See Note B.	90		nsec	
$\overline{RD}$	$t_S\Phi(RD)$	$\overline{RD}$ Set-Up Time to Rising Edge of Φ During Read or $\overline{M1}$ Cycle	115		nsec	
A_0-A_7, B_0-B_7	$t_S(PD)$	Port Data Set-Up Time to Rising Edge of $\overline{STROBE}$ (MODE 1)	230		nsec	
	$t_{DS}(PD)$	Port Data Output Delay from Falling Edge of STROBE (Mode 2)		210	nsec	[5]
	$t_F(PD)$	Delay to Floating Port Data Bus from Rising Edge of STROBE (Mode 2)		180	nsec	C_L = 50pF
	$t_{DI}(PD)$	Port Data Stable from Rising Edge of $\overline{IORQ}$ During $\overline{WR}$ Cycle (Mode 0)		180	nsec	[5]
$\overline{ASTB}$ $\overline{BTSB}$	$t_W(ST)$	Pulse Width, $\overline{STROBE}$	150 [4]		nsec nsec	
$\overline{INT}$	$t_D(IT)$	$\overline{INT}$ Delay Time from Rising Edge of $\overline{STROBE}$		440	nsec	
	$t_D(IT3)$	$\overline{INT}$ Delay Time from Data Match During Mode 3 Operation		380	nsec	
ARBY, BRDY	$t_{DH}(RY)$	Ready Response Time from Rising Edge of $\overline{IORQ}$		t_c + 410	nsec	[5] C_L = 50pF
	$t_{DL}(RY)$	Ready Response Time from Rising Edge of $\overline{STROBE}$		t_c+ 360	nsec	[5]

A. $2.5t_c > (N-2)t_{DL}(IO) + t_{DM}(IO) + t_S(IEI)$ + TTL Buffer Delay, if any

B. M1 must be active for a minimum of 2 clock periods to reset the PIO.

[1] $t_c = t_W(\Phi H) + t_W(\Phi L) + t_r + t_f$

[2] Increase $t_{DR(D)}$ by 10 nsec for each 50pF increase in loading up to 200pF max.

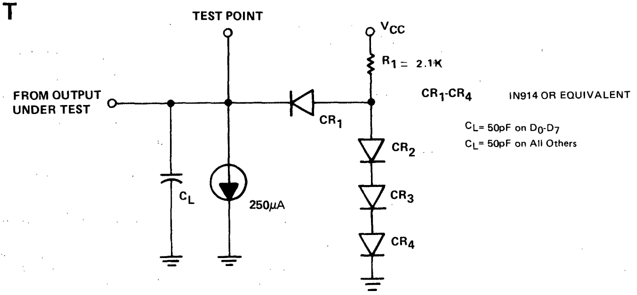
[3] Increase $t_{DI}(D)$ by 10 nsec for each 50pF increase in loading up to 200pF max.

[4] For Mode 2: $t_W(ST) > t_S(PD)$

[5] Increase these values by 2 nsec for each 10pF increase in loading up to 100pF max.

OUTPUT LOAD CIRCUIT

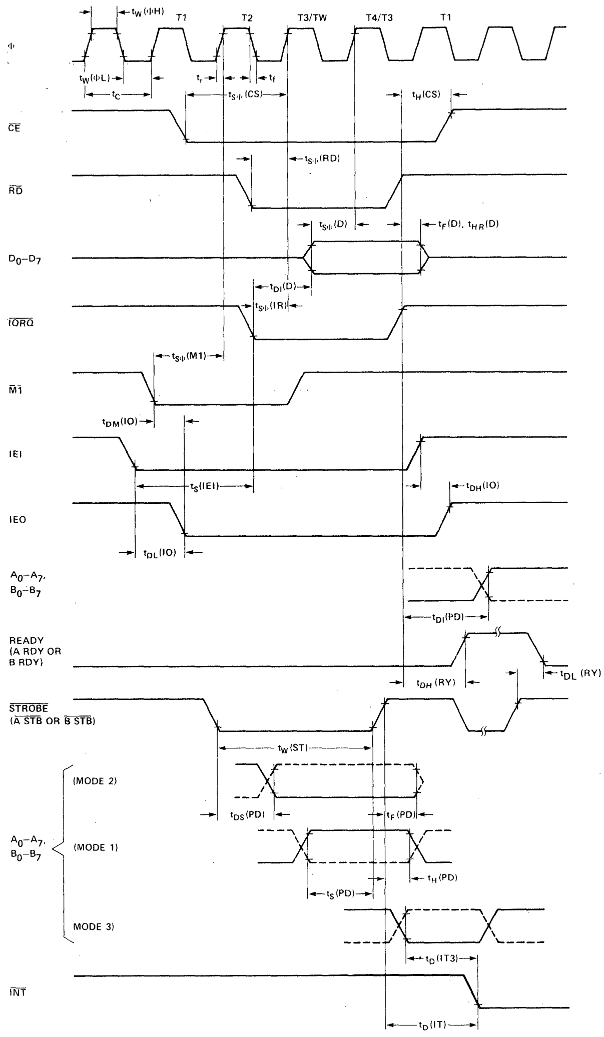
Figure 9.4-1



9.5 TIMING DIAGRAM

Timing measurements are made at the following voltages, unless otherwise specified:

	"1"	"0"
CLOCK	4.2V	0.8V
OUTPUT	2.0V	0.8V
INPUT	2.0V	0.8V
FLOAT	ΔV	= +0.5V



10.0 ORDERING INFORMATION

PART NO.	DESIGNATOR	PACKAGE TYPE	MAX CLOCK FREQUENCY	TEMPERATURE RANGE
MK3881N	Z80-PIO	Plastic	2.5 MHz	0° to 70°C
MK3881P	Z80-PIO	Ceramic	2.5 MHz	
MK3881J	Z80-PIO	Cerdip	2.5 MHz	
MK3881N-4	Z80A-PIO	Plastic	4.0 MHz	
MK3881P-4	Z80A-PIO	Ceramic	4.0 MHz	
MK3881J-4	Z80A-PIO	Cerdip	4.0 MHz	
MK3881P-10	Z80-PIO	Ceramic	4.0 MHz	-40° to +85°C

MOSTEK®

Z80 MICROCOMPUTER DEVICES

Technical Manual

MK3882 COUNTER TIMER CIRCUIT

TABLE OF CONTENTS

1.0 INTRODUCTION

The Z80-Counter Timer Circuit (CTC) is a programmable component with four independent channels that provide counting and timing functions for microcomputer systems based on the Z80-CPU. The CPU can configure the CTC channels to operate under various modes and conditions as required to interface with a wide range of devices. In most applications, little or no external logic is required. The Z80-CTC utilizes N-channel silicon gate depletion load technology and is packaged in a 28-pin DIP. The Z80-CTC requires only a single 5 volt supply and a one-phase 5 volt clock. Major features of the Z80-CTC include:

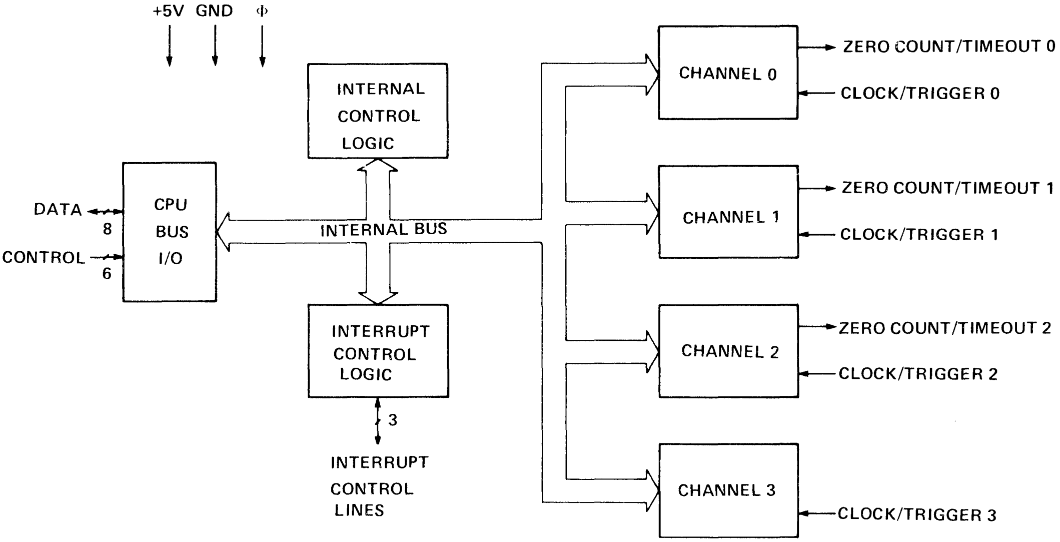
- All inputs and outputs fully TTL compatible.
- Each channel may be selected to operate in either Counter Mode or Timer Mode.
- Used in either mode, a CPU-readable Down Counter indicates number of counts-to-go until zero.
- A Time Constant Register can automatically reload the Down Counter at Count Zero in Counter and Timer Mode.
- Selectable positive or negative trigger initiates time operation in Timer Mode. The same input is monitored for event counts in Counter Mode.
- Three channels have Zero Count/Timeout outputs capable of driving Darlington transistors.
- Interrupts may be programmed to occur on the zero count condition in any channel.
- Daisy chain priority interrupt logic included to provide for automatic interrupt vectoring without external logic.

2.0 CTC ARCHITECTURE

2.1 OVERVIEW

A block diagram of the Z80-CTC is shown in Figure 2.0-1. The internal structure of the Z80-CTC consists of a Z80-CPU bus interface, Internal Control Logic, four sets of Counter/Timer Channel Logic, and Interrupt Control Logic. The four independent counter/timer channels are identified by sequential numbers from 0 to 3. The CTC has the capability of generating a unique interrupt vector for each separate channel (for automatic vectoring to an interrupt service routine). The 4 channels can be connected into four contiguous slots in the standard Z80 priority chain with channel number 0 having the highest priority.The CPU bus interface logic allows the CTC device to interface directly to the CPU with no other external logic. However, port address decoders and/or line buffers may be required for large systems.

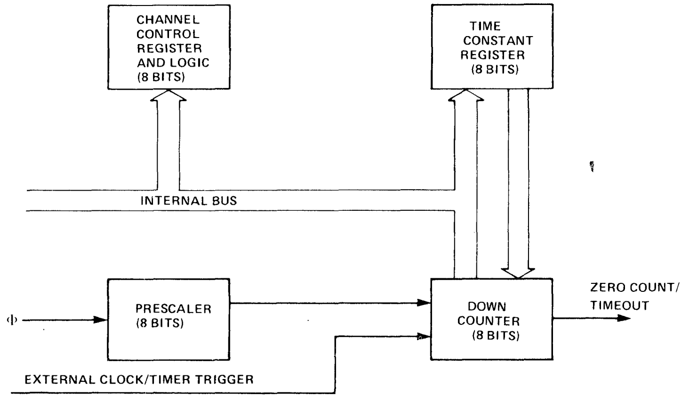
Z80-CTC BLOCK DIAGRAM
Figure 2.0-1



2.2 STRUCTURE OF CHANNEL LOGIC

The structure of one of the four sets of Counter/Timer Channel Logic is shown in Figure 2.0-2. This logic is composed of 2 registers, 2 counters and control logic. The registers are an 8-bit Time Constant Register and an 8-bit Channel Control Register. The counters are an 8-bit CPU-readable Down Counter and an 8-bit Prescaler.

CHANNEL BLOCK DIAGRAM
Figure 2.0-2



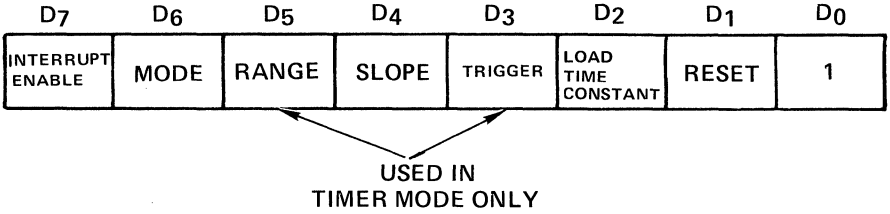
2.2.1 THE CHANNEL CONTROL REGISTER AND LOGIC

The Channel Control Register (8-bit) and Logic is written to by the CPU to select the modes and parameters of the channel. Within the entire CTC device there are four such registers, corresponding to the four Counter/Timer Channels. Which of the four is being written to depends on the encoding of two channel select input pins: CS0 and CS1 (usually attached to A0 and A1 of the CPU address bus). This is illustrated in the truth table below:

	CS1	CS0
Ch0	0	0
Ch1	0	1
Ch2	1	0
Ch3	1	1

In the control word written to program each Channel Control Register, bit 0 is always set, and the other 7 bits are programmed to select alternatives on the channel's operating modes and parameters, as shown in the diagram below. (For a more complete discussion see section 4.0: "CTC Operating Modes" and section 5.0: "CTC Programming.")

CHANNEL CONTROL REGISTER



D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
INTERRUPT ENABLE	MODE	RANGE	SLOPE	TRIGGER	LOAD TIME CONSTANT	RESET	1

2.2.2 THE PRESCALER

Used in the Timer Mode only, the Prescaler is an 8-bit device which can be programmed by the CPU via the Channel Control Register to divide its input, the System Clock (Φ), by 16 or 256. The output of the Prescaler is then fed as an input to clock the Down Counter, which initially, and every time it clocks down to zero, is reloaded automatically with the contents of the Time Constant Register. In effect this again divides the System Clock by an additional factor of the time constant. Every time the Down Counter counts down to zero, its output, Zero Count/Timeout (ZC/TO), is pulsed high.

2.2.3 THE TIME CONSTANT REGISTER

The Time Constant Register is an 8-bit register, used in both Counter Mode and Timer Mode, programmed by the CPU just after the Channel Control Word with an integer time constant value of 1 through 256. This register loads the programmed value into the Down Counter when the CTC is first initialized and reloads the same value into the Down Counter automatically whenever it counts down thereafter to zero. If a new time constant is loaded into the Time Constant Register while a channel is counting or timing, the present down count will be completed before the new time constant is loaded into the Down Counter. (For details of how a time constant is written to a CTC channel, see section 5.0: "CTC Programming.")

2.2.4 THE DOWN COUNTER

The Down Counter is an 8-bit register used in both Counter Mode and Timer Mode loaded initially, and later when it counts down to zero, by the Time Constant Register. The Down Counter is decremented by each external clock edge in the Counter Mode, or in the Timer Mode, by the clock output of the Prescaler. At any time, by performing a simple I/O Read at the port address assigned to the selected CTC channel, the CPU can access the contents of this register and obtain the number of counts-to-zero. Any CTC channel may be programmed to generate an interrupt request sequence each time the zero count is reached.

In channels 0, 1, and 2, when the zero count condition is reached, a signal pulse appears at the corresponding ZC/TO pin. Due to package pin limitations, however, channel 3 does not have this pin and so may be used only in applications where this output pulse is not required.

2.3 INTERRUPT CONTROL LOGIC

The Interrupt Control Logic insures that the CTC acts in accordance with Z80 system interrupt protocol for nested priority interrupting and return from interrupt. The priority of any system device is determined by its physical location in a daisy chain configuration. Two signal lines (IEI and IEO) are provided in CTC devices to form this system daisy chain. The device closest to the CPU has the highest priority; within the CTC, interrupt priority is predetermined by channel number, with channel 0 having highest priority down to channel 3 which has the lowest priority. The purpose of a CTC-generated interrupt, as with any other peripheral device, is to force the CPU to execute an interrupt service routine. According to Z80 system interrupt protocol, lower priority devices or channels may not interrupt higher priority devices or channels that have already interrupted and have not had their interrupt service routines completed. However, high priority devices or channels may interrupt the servicing of lower priority devices or channels.

A CTC channel may be programmed to request an interrupt every time its Down Counter reaches a count of zero. (To utilize this feature requires that the CPU be programmed for interrupt mode 2.) Some time after the interrupt request, the CPU will send out an interrupt acknowledge, and the CTC's Interrupt Control Logic will determine the highest-priority channel which is requesting an interrupt within the CTC device. Then if the CTC's IEI input is active, indicating that it has priority within the system daisy chain, it will place an 8-bit Interrupt Vector on the system data bus. The high-order 5 bits of this vector will have been written to the CTC earlier as part of the CTC initial programming process; the next two bits will be provided by the CTC's Interrupt Control Logic as a binary code corresponding to the highest-priority channel requesting an interrupt; finally the low-order bit of the vector will always be zero according to a convention described below.

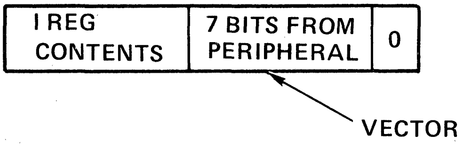
INTERRUPT VECTOR

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
V_7	V_6	V_5	V_4	V_3	X	X	0

0	0	CHANNEL 0
0	1	CHANNEL 1
1	0	CHANNEL 2
1	1	CHANNEL 3

This interrupt vector is used to form a pointer to a location in memory where the address of the interrupt service routine is stored in a table. The vector represents the least significant 8 bits, while the CPU reads the contents of the I register to provide the most significant 8-bits of the 16-bit pointer. The address in memory pointed to will contain the low-order byte, and the next highest address will contain the high-order byte of an address which in turn contains the first opcode of the interrupt service routine. Thus in mode 2, a single 8-bit vector stored in an interrupting CTC can result in an indirect call to any memory location.

Z80 16-BIT POINTER (INTERRUPT STARTING ADDRESS)



2.3 INTERRUPT CONTROL LOGIC (Cont'd)

There is a Z80 system convention that all addresses in the interrupt service routine table should have their low-order byte in an even location in memory, and their high-order byte in the next highest location in memory, which will always be odd so that the least significant bit of any interrupt vector will always be even. Hence the least significant bit of any interrupt vector will always be zero.

The RETI instruction is used at the end of any interrupt service routine to initialize the daisy chain enable line IEO for proper control of nested priority interrupt handing. The CTC monitors the system data bus and decodes this instruction when it occurs. Thus the CTC channel control logic will know when the CPU has completed servicing an interrupt, without any further communication with the CPU being necessary.

3.0 CTC PIN DESCRIPTION

A diagram of the Z80-CTC pin configuration is shown in Figure 3.0-1. This section describes the function of each pin.

D7 - D0
Z80-CPU Data Bus (bi-directional, tri-state)

This bus is used to transfer all data and command words between the Z80-CPU and the Z80-CTC. There are 8 bits on this bus, of which D0 is the least significant.

CS1 - CS0
Channel Select (input, active high)

These pins form a 2-bit binary address code for selecting one of the four independent CTC channels for an I/O Write or Read (See truth table below.)

	CS1	CS0
Ch0	0	0
Ch1	0	1
Ch2	1	0
Ch3	1	1

$\overline{\text{CE}}$
Chip Enable (input, active low)

A low level on this pin enables the CTC to accept control words, Interrupt Vectors, or time constant data words from the Z80 Data Bus during an I/O Write cycle, or to transmit the contents of the Down Counter to the CPU during an I/O Read cycle. In most applications this signal is decoded from the 8 least significant bits of the address bus for any of the four I/O port addresses that are mapped to the four Counter/Timer Channels.

Clock (Φ)
System Clock (input)

This single-phase clock is used by the CTC to synchronize certain signals internally.

$\overline{\text{M1}}$
Machine Cycle One Signal from CPU (input, active low)

When $\overline{\text{M1}}$ is active and the $\overline{\text{RD}}$ signal is active, the CPU is fetching an instruction from memory. When $\overline{\text{M1}}$ is active and the $\overline{\text{IORQ}}$ signal is active, the CPU is acknowledging an interrupt, alerting the CTC to place an Interrupt Vector on the Z80 Data Bus if it has daisy chain priority and one of its channels has requested an interrupt.

$\overline{\text{IORQ}}$
Input/Output Request from CPU (input, active low)

The $\overline{\text{IORQ}}$ signal is used in conjunction with the $\overline{\text{CE}}$ and $\overline{\text{RD}}$ signals to transfer data and Channel Control Words between the Z80-CPU and the CTC. During a CTC Write Cycle, $\overline{\text{IORQ}}$ and $\overline{\text{CE}}$ must be true and $\overline{\text{RD}}$ false. The CTC does not receive a specific write signal, instead generating its own internally from the inverse of a valid $\overline{\text{RD}}$ signal. In a CTC Read Cycle, $\overline{\text{IORQ}}$, $\overline{\text{CE}}$ and $\overline{\text{RD}}$ must be active to place the contents of the Down Counter on the Z80 Data Bus. If $\overline{\text{IORQ}}$ and $\overline{\text{M1}}$ are both true, the CPU is acknowledging an interrupt request, and the highest-priority interrupting channel will place its Interrupt Vector on the Z80 Data Bus.

3.0 CTC PIN DESCRIPTION (CONT'D)

$\overline{\text{RD}}$

Read Cycle Status from the CPU (input, active low)

The $\overline{\text{RD}}$ signal is used in conjunction with the $\overline{\text{IORQ}}$ and $\overline{\text{CE}}$ signals to transfer data and Channel Control Words between the Z80-CPU and the CTC. During a CTC Write Cycle, $\overline{\text{IORQ}}$ and $\overline{\text{CE}}$ must be true and $\overline{\text{RD}}$ false. The CTC does not receive a specific write signal, instead generating its own internally from the inverse of a valid $\overline{\text{RD}}$ signal. In a CTC Read Cycle, $\overline{\text{IORQ}}$, $\overline{\text{CE}}$ and $\overline{\text{RD}}$ must be active to place the contents of the Down Counter on the Z80 Data Bus.

IEI

Interrupt Enable In (input, active high)

This signal is used to help form a system-wide interrupt daisy chain which establishes priorities when more than one peripheral device in the system has interrupting capability. A high level on this pin indicates that no other interrupting devices of higher priority in the daisy chain are being serviced by the Z80-CPU.

IEO

Interrupt Enable Out (output, active high)

The IEO signal, in conjunction with IEI, is used to form a system-wide interrupt priority daisy chain. IEO is high only if IEI is high and the CPU is not servicing an interrupt from any CTC channel. Thus this signal blocks lower priority devices from interrupting while a higher priority interrupting device is being serviced by the CPU.

$\overline{\text{INT}}$

Interrupt Request (output, open drain, active low)

This signal goes true when any CTC channel which has been programmed to enable interrupts has a zero-count condition in its Down Counter.

$\overline{\text{RESET}}$

Reset (input, active low)

This signal stops all channels from counting and resets channel interrupt enable bits in all control registers, thereby disabling CTC-generated interrupts. The ZC/TO and $\overline{\text{INT}}$ outputs go to their inactive states, IEO reflects IEI, and the CTC's data bus output drivers go to the high impedance state.

CLK/TRG3–CLK/TRG0

External Clock/Timer Trigger (input, user-selectable active high or low)

There are four CLK/TRG pins, corresponding to the four independent CTC channels. In the Counter Mode, every active edge on this pin decrements the Down Counter. In the Timer Mode, an active edge on this pin initiates the timing function. The user may select the active edge to be either rising or falling.

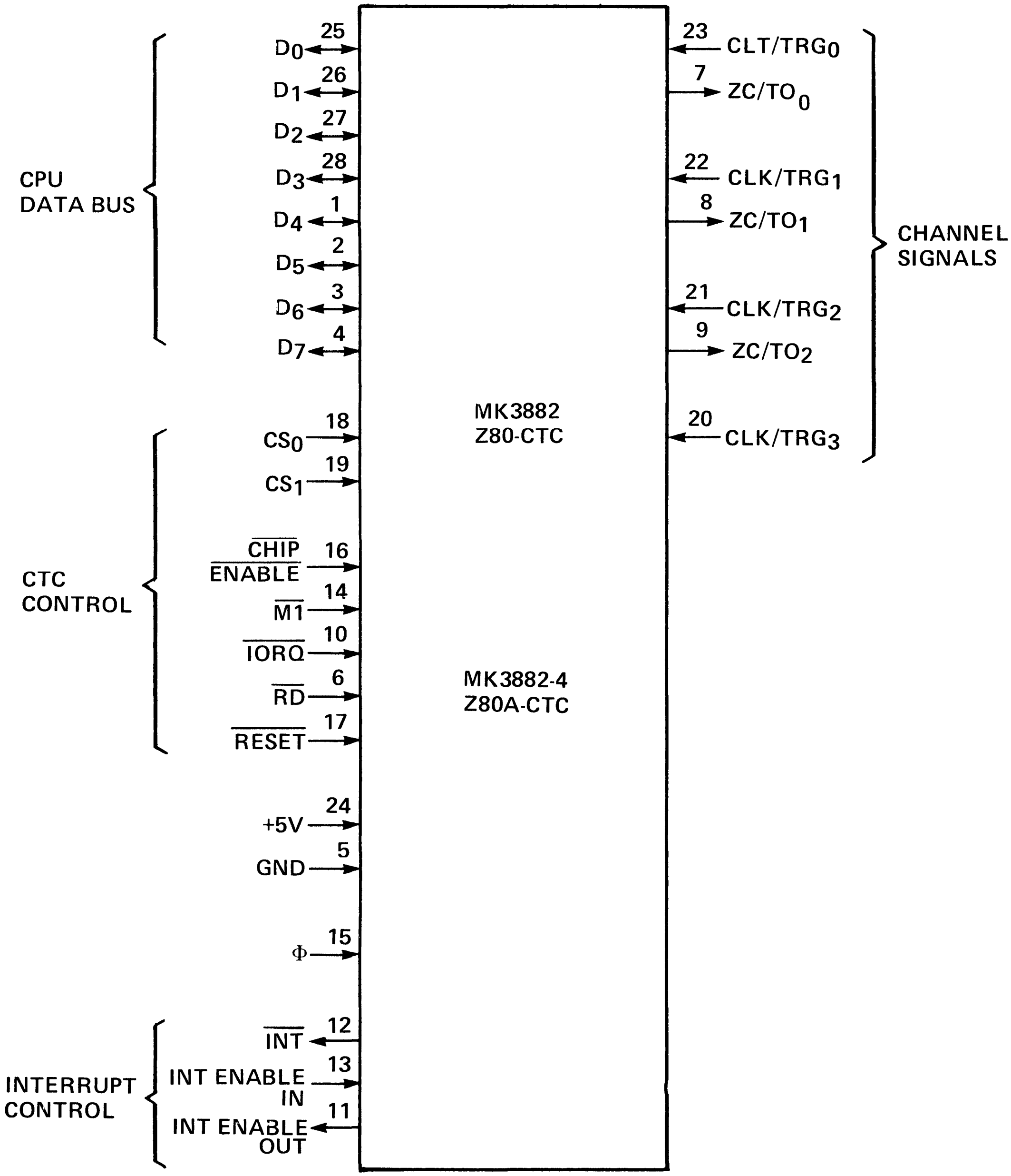
ZC/TO2–ZC/TO0

Zero Count/Timeout (output, active high)

There are three ZC/TO pins, corresponding to CTC channels 2 through 0. (Due to package pin limitations channel 3 has no ZC/TO pin.) In either Counter Mode or Timer Mode, when the Down Counter decrements to zero an active high going pulse appears at this pin.

Z80-CTC PIN CONFIGURATION

Figure 3.0-1



4.0 CTC OPERATING MODES

At power-on, the Z80-CTC state is undefined. Asserting $\overline{\text{RESET}}$ puts the CTC in a known state. Before any channel can begin counting or timing, a Channel Control Word and a time constant data word must be written to the appropriate registers of that channel. Further, if any channel has been programmed to enable interrupts, an Interrupt Vector word must be written to the CTC's Interrupt Control Logic. (For further details, refer to section 5.0: "CTC Programming.") When the CPU has written all of these words to the CTC, all active channels will be programmed for immediate operation in either the Counter Mode or the Timer Mode.

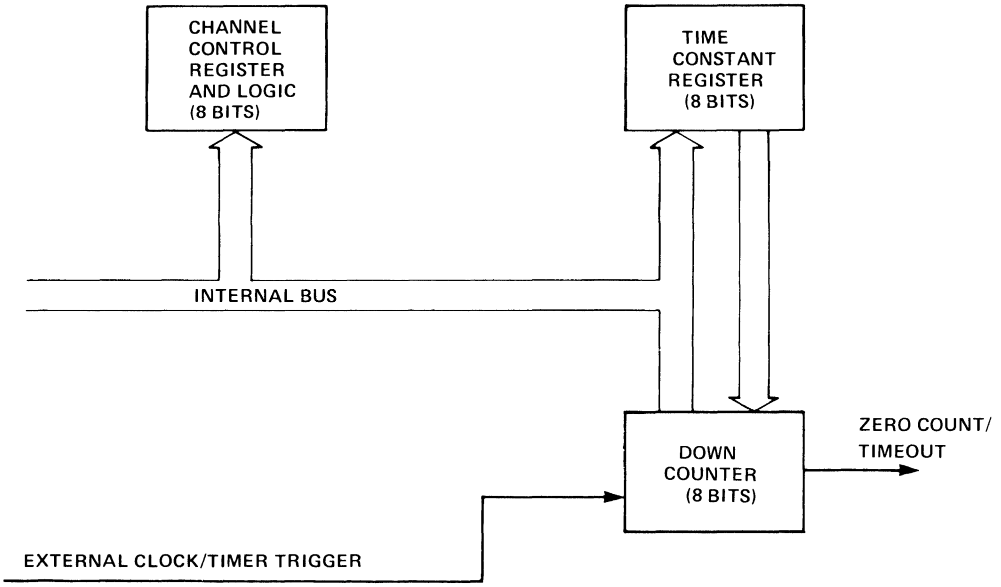
4.1 CTC COUNTER MODE

In this mode the CTC counts edges of the CLK/TRG input. The Counter Mode is programmed for a channel when its Channel Control Word is written with bit 6 set. The Channel's External Clock (CLK/TRG) input is monitored for a series of triggering edges; after each, in synchronization with the next rising edge of Φ (the System Clock), the Down Counter (which was initialized with the time constant data word at the start of any sequence of down-counting) is decremented. Although there is no set-up time requirement between the triggering edge of the External Clock and the rising edge of Φ, (Clock), the Down Counter will not be decremented until the following Φ pulse. (See the parameter ts(CK) in section 8.3: "A.C. Characteristics.") A channels's External Clock input is pre-programmed by bit 4 of the Channel Control Word to trigger the decrementing sequence with either a high or a low going edge.

In any of Channels 0, 1, or 2, when the Down Counter is successively decremented from the original time constant until finally it reaches zero, the Zero Count (ZC/TO) output pin for that channel will be pulsed active (high). (However, due to package pin limitations, channel 3 does not have this pin and so may only be used in applications where this output pulse is not required.) Further, if the channel has been so pre-programmed by bit 7 of the Channel Control Word, an interrupt request sequence will be generated. (For more details, see section 7.0: "CTC Interrupt Servicing.")

As the above sequence is proceeding, the zero count condition also results in the automatic reload of the Down Counter with the original time constant data word in the Time Constant Register. There is no interruption in the sequence of continued down-counting. If the Time Constant Register is written to with a new time constant data word while the Down Counter is decrementing, the present count will be completed before the new time constant will be loaded into the Down Counter.

CHANNEL - COUNTER MODE
Figure 4.1-0


CHANNEL CONTROL REGISTER AND LOGIC (8 BITS)
TIME CONSTANT REGISTER (8 BITS)
INTERNAL BUS
DOWN COUNTER (8 BITS)
ZERO COUNT/ TIMEOUT
EXTERNAL CLOCK/TIMER TRIGGER

4.2 CTC TIMER MODE

In this mode the CTC generates timing intervals that are an integer value of the system clock period. The Timer Mode is programmed for a channel when its Channel Control Word is written with bit 6 reset. The channel then may be used to measure intervals of time based on the System Clock period. The System Clock is fed through two successive counters, the Prescaler and the Down Counter. Depending on the pre-programmed bit 5 in the Channel Control Word, the Prescaler divides the System Clock by a factor of either 16 or 256. The output of the Prescaler is then used as a clock to decrement the Down Counter, which may be pre-programmed with any time constant integer between 1 and 256. As in the Counter Mode, the time constant is automatically reloaded into the Down Counter at each zero-count condition, and counting continues. Also at zero-count, the channel's Time Out (ZC/TO) output (which is the output of the Down Counter) is pulsed, resulting in a uniform pulse train of precise period given by the product.

$$t_C * P * TC$$

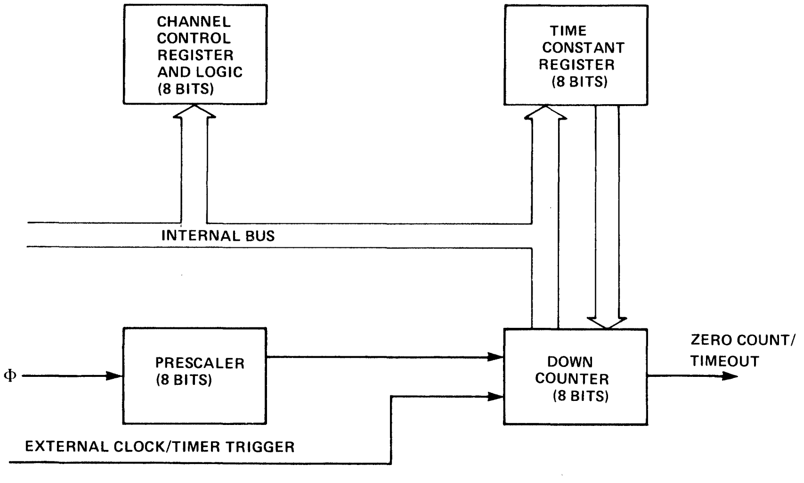
where tc is the System Clock period, P is the Prescaler factor of 16 or 256 and TC is the pre-programmed time constant.

Bit 3 of the Channel Control Word is pre-programmed to select whether timing will be automatically initiated, or whether it will be initiated with a triggering edge at the channel's Timer Trigger (CLK/TRG) input. If bit 3 is reset the timer automatically begins operation at the start of the CPU cycle following the I/O Write machine cycle that loads the time constant data word to the channel. If bit 3 is set the timer begins operation on the second succeeding rising edge of Φ after the Timer Trigger edge following the loading of the time constant data word. If no time constant data word is to follow then the timer begins operation on the second succeeding rising edge of Φ after the Timer Trigger edge following the control word write cycle. Bit 4 of the Channel Control Word is pre-programmed to select whether the Timer Trigger will be sensitive to a rising or falling edge. Although there is no set-up requirement between the active edge of the Timer Trigger and the next rising edge of Φ. If the Timer Trigger edge occurs closer than a specified minimum set-up time to the rising edge of Φ, the Down Counter will not begin decrementing until the following rising edge of Φ. (See parameter ts(TR) in section 8.3: "A.C. Characteristics".)

If bit 7 in the Channel Control Word is set, the zero-count condition in the Down Counter, besides causing a pulse at the channel's Time Out pin, will be used to initiate an interrupt request sequence, (For more details, see section 7.0: "CTC Interrupt Servicing.")

CHANNEL - TIMER MODE

Figure 4.2-0


CHANNEL CONTROL REGISTER AND LOGIC (8 BITS)
TIME CONSTANT REGISTER (8 BITS)
INTERNAL BUS
Φ
PRESCALER (8 BITS)
DOWN COUNTER (8 BITS)
ZERO COUNT/ TIMEOUT
EXTERNAL CLOCK/TIMER TRIGGER

5.0 CTC PROGRAMMING

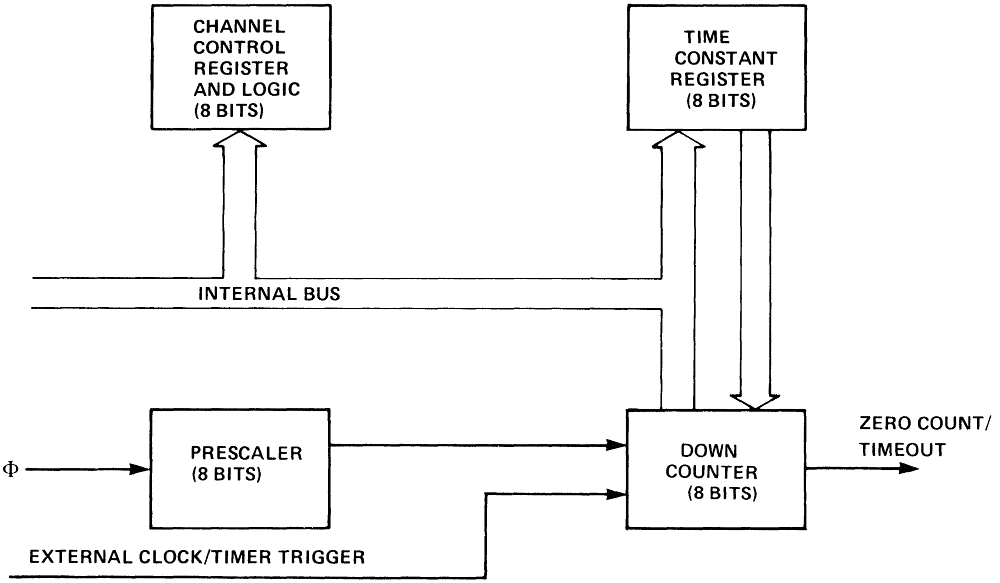
Before a Z80-CTC channel can begin counting or timing operations, a Channel Control Word and a Time Constant data word must be written to it by the CPU. These words will be stored in the Channel Control Register and the Time Constant Register of that channel. In addition, if any of the four channels have been programmed with bit 7 of their Channel Control Words to enable interrupts, an Interrupt Vector must be written to the appropriate register in the CTC. Due to automatic features in the Interrupt Control Logic, one pre-programmed Interrupt Vector suffices for all four channels.

5.1 LOADING THE CHANNEL CONTROL REGISTER

To load a Channel Control Word, the CPU performs a normal I/O Write sequence to the port address corresponding to the desired CTC channel. Two CTC input pins, namely CS0 and CS1, are used to form a 2-bit binary address to select one of four channels within the device. (For a truth table, see section 2.2.1: "The Channel Control Register and Logic".) In many system architectures, these two input pins are connected to Address Bus lines A0 and A1, respectively, so that the four channels in a CTC device will occupy contiguous I/O port addresses. A word written to a CTC channel will be interpreted as a Channel Control Word, and loaded into the Channel Control Register, its bit 0 is a logic 1. The other seven bits of this word select operating modes and conditions as indicated in the diagram below. Following the diagram the meaning of each bit will be discussed in detail.

CHANNEL BLOCK DIAGRAM

Figure 5.1-0


CHANNEL CONTROL REGISTER AND LOGIC (8 BITS)
TIME CONSTANT REGISTER (8 BITS)
INTERNAL BUS
Φ
PRESCALER (8 BITS)
DOWN COUNTER (8 BITS)
ZERO COUNT/ TIMEOUT
EXTERNAL CLOCK/TIMER TRIGGER

CHANNEL CONTROL REGISTER

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
INTERRUPT ENABLE	MODE	RANGE	SLOPE	TRIGGER	LOAD TIME CONSTANT	RESET	1
		USED IN TIMER MODE ONLY		USED IN TIMER MODE ONLY			

5.1 LOADING THE CHANNEL CONTROL REGISTER (CONT'D)

Bit 7 = 1

The channel is enabled to generate an interrupt request sequence every time the Down Counter reaches a zero-count condition. To set this bit to 1 in any of the four Channel Control Registers necessitates that an Interrupt Vector also be written to the CTC before operation begins. Channel interrupts may be programmed in either Counter Mode or Timer Mode. If an updated Channel Control Word is written to a channel already in operation, with bit 7 set, the interrupt enable selection will not be retroactive to a preceding zero-count condition.

Bit 7 = 0

Channel interrupts disabled. Any pending interrupt by that channel will be cleared.

Bit 6 = 1

Counter Mode selected. The Down Counter is decremented by each triggering edge of the External Clock (CLK/TRG) input. The Prescaler is not used.

Bit 6 = 0

Timer Mode selected. The Prescaler is clocked by the System Clock Φ, and the output of the Prescaler in turn clocks the Down Counter. The output of the Down Counter (the channel's ZC/TO output) is a uniform pulse train of period given by the product.

$$t_C * P * TC$$

where t_C is the period of System Clock Φ, P is the Prescaler factor of 16 or 256, and TC is the time constant data word.

Bit 5 = 1

(Defined for Timer Mode only.) Prescaler factor is 256.

Bit 5 = 0

(Defined for Timer Mode only.) Prescaler factor is 16.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
INTERRUPT ENABLE	MODE	RANGE	SLOPE	TRIGGER	LOAD TIME CONSTANT	RESET	1
		USED IN TIMER MODE ONLY		USED IN TIMER MODE ONLY			

Bit 4 = 1

TIMER MODE - positive edge trigger starts timer operation.
COUNTER MODE - positive edge decrements the down counter.

Bit 4 = 0

TIMER MODE - negative edge trigger starts timer operation.
COUNTER MODE - negative edge decrements the down counter.

5.1 LOADING THE CHANNEL CONTROL REGISTER (CONT'D)

Bit 3 = 1

Timer Mode Only - External trigger is valid for starting timer operation after rising edge of T_2 of the machine cycle following the one that loads the time constant. The Prescaler is decremented 2 clock cycles later if the setup time is met, otherwise 3 clock cycles. Once timer has been started it will free run at the rate determined by the Time Constant register.

Bit 3 = 0

Timer Mode Only - Timer begins operation on the rising edge of T_2 of the machine cycle following the one that loads the time constant.

Bit 2 = 1

The time constant data word for the Time Constant Register will be the next word written to this channel. If an updated Channel Control Word and time constant data word are written to a channel while it is already in operation, the Down Counter will continue decrementing to zero before the new time constant is loaded into it.

Bit 2 = 0

No time constant data word for the Time Constant Register should be expected to follow. To program bit 2 to this state implies that this Channel Control Word is intended to update the status of a channel already in operation, since a channel will not operate without a correctly programmed data word in the Time Constant Register, and a set bit 2 in this Channel Control Word provides the only way of writing to the Time Constant Register.

Bit 1 = 1

Reset channel. Channel stops counting or timing. This is not a stored condition. Upon writing into this bit a reset pulse discontinues current channel operation, however, none of the bits in the channel control register are changed. If both bit 2 = 1 and bit 1 = 1 the channel will resume operation upon loading a time constant.

Bit 1 = 0

Channel continues current operation.

5.2 DISABLING THE CTC'S INTERRUPT STRUCTURE

If an external Asynchronous interrupt could occur while the processor is writing the disable word to the CTC (01H); a system problem may occur. If interrupts are enabled in the processor it is possible that the Asynchronous interrupt will occur while the processor is writing the disable word to the CTC. The CTC will generate an INT and the CPU will acknowledge it, however, by this time, the CTC will have received the disable word and de-activated its interrupt structure. The result is that the CTC will not send in its interrupt vector during the interrupt acknowledge cycle because it is disabled and the CPU will fetch an erroneous vector resulting in a program fault. The cure for this problem is to disable interrupts within the CPU with the DI instruction just before the CTC is disabled and then re-enable interrupts with the EI instruction. This action causes the CPU to ignore any interrupts produced by the CTC while it is being disabled. The code sequence would be:

```
LD  A, 01H
DI                ; DISABLE CPU
OUT (CTC), A      ; DISABLE CTC
EI                ; ENABLE CPU
—
—
```

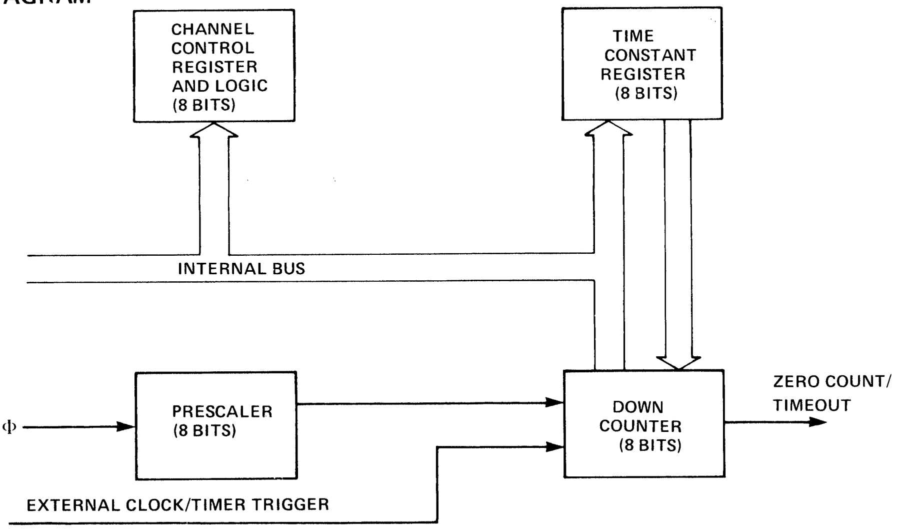
5.3 LOADING THE TIME CONSTANT REGISTER

A channel may not begin operation in either Timer Mode or Counter Mode unless a time constant data word is written into the Time Constant Register by the CPU. This data word will be expected on the next I/O Write to this channel following the I/O Write of the Channel Control Word, provided that bit 2 of the Channel Control Word is set. The time constant data word may be an integer value in the range 1-256. If all eight bits in this word are zero, it is interpreted as 256. If a time constant data word is loaded to a channel already in operation, the Down Counter will continue decrementing to zero before the new time constant is loaded from the Time Constant Register to the Down Counter.

TIME CONSTANT REGISTER

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
TC_7	TC_6	TC_5	TC_4	TC_3	TC_2	TC_1	TC_0
MSB							LSB

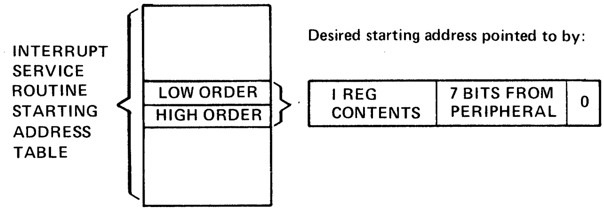
CHANNEL BLOCK DIAGRAM
Figure 5.3-0


CHANNEL CONTROL REGISTER AND LOGIC (8 BITS)
TIME CONSTANT REGISTER (8 BITS)
INTERNAL BUS
Φ
PRESCALER (8 BITS)
DOWN COUNTER (8 BITS)
ZERO COUNT/ TIMEOUT
EXTERNAL CLOCK/TIMER TRIGGER

5.4 LOADING THE INTERRUPT VECTOR REGISTER

The Z80-CTC has been designed to operate with the Z80-CPU programmed for mode 2 interrupt response. Under the requirements of this mode, when a CTC channel requests an interrupt and is acknowledged, a 16-bit pointer must be formed to obtain a corresponding interrupt service routine starting address from a table in memory. The upper 8 bits of this pointer are provided by the CPU's I register, and the lower 8 bits of the pointer are provided by the CTC in the form of an Interrupt Vector unique to the particular channel that requested the interrupt. (For further details, see section 7.0: "CTC Interrupt Servicing".)

MODE 2 INTERRUPT OPERATION


INTERRUPT SERVICE ROUTINE STARTING ADDRESS TABLE
LOW ORDER
HIGH ORDER
Desired starting address pointed to by:
I REG CONTENTS
7 BITS FROM PERIPHERAL
0

5.4 LOADING THE INTERRUPT VECTOR REGISTER (Cont'd)

The high order 5 bits of this Interrupt Vector must be written to the CTC in advance as part of the initial programming sequence. To do so, the CPU must write to the I/O port address corresponding to the CTC channel 0, just as it would if a Channel Control Word were being written to that channel, except that bit 0 of the word being written must contain a 0. (As explained above in section 5.1, if bit 0 of a word written to a channel were set to 1, the word would be interpreted as a Channel Control Word, so a 0 in bit 0 signals the CTC to load the incoming word into the Interrupt Vector Register.) Bits 1 and 2, however are not used when loading this vector. At the time when the interrupting channel must place the Interrupt Vector on the Z80 Data Bus, the Interrupt Control Logic of the CTC automatically supplies a binary code in bits 1 and 2 indentifying which of the four CTC channels is to be serviced.

INTERRUPT VECTOR REGISTER

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
V_7	V_6	V_5	V_4	V_3	X	X	0

SUPPLIED BY USER

D_2	D_1		
0	0	CHANNEL 0	(Highest Priority)
0	1	CHANNEL 1	
1	0	CHANNEL 2	
1	1	CHANNEL 3	(Lowest Priority)

AUTOMATICALLY INSERTED BY Z80-CTC

6.0 CTC TIMING

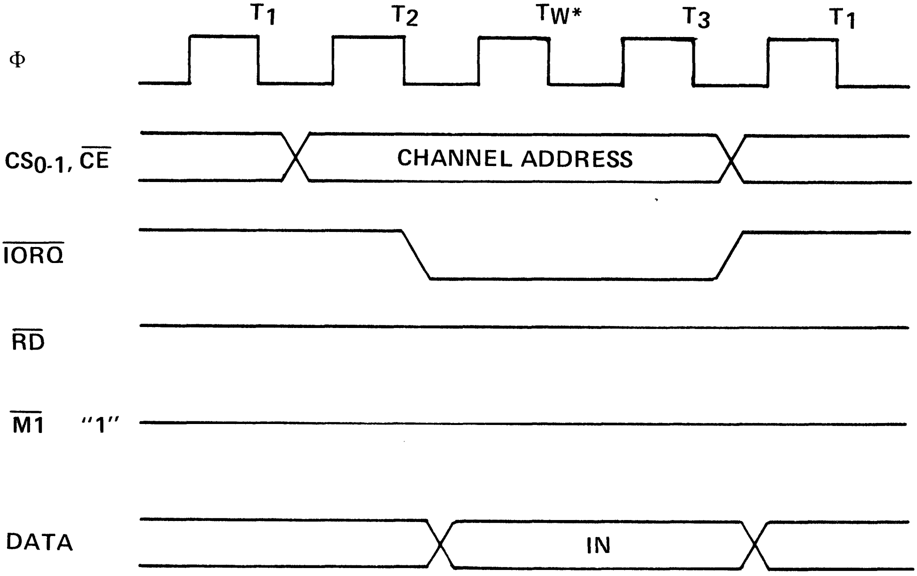
This section illustrates the timing relationships of the relevant CTC pins for the following types of operation: writing a word to the CTC, reading a word from the CTC, counting, and timing. Elsewhere in this manual may be found timing diagrams relating to interrupt servicing (section 7.0) and an A.C. Timing Diagram which quantitatively specifies the timing relationships (section 8.4).

6.1 CTC WRITE CYCLE

Figure 6.1-0 illustrates the timing associated with the CTC Write Cycle. This sequence is applicable to loading either a Channel Control Word, an Interrupt Vector, or a time constant data word.

In the sequence shown, during clock cycle T_1, the Z80-CPU prepares for the Write Cycle with a false (high) signal at CTC input pin $\overline{RD}$ (Read). Since the CTC has no separate Write signal input, it generates its own internally form the false $\overline{RD}$ input. Later, during clock cycle T_2, the Z80-CPU initiates the Write Cycle with true (low) signals at CTC input pins $\overline{IORQ}$ (I/O Request) and $\overline{CE}$ (Chip Enable). (Note: $\overline{M1}$ must be false to distinguish the cycle form an interrupt acknowledge.) Also at this time a 2-bit binary code appears at CTC inputs CS1 and CS0 (Channel Select 1 and 0), specifying which of the four CTC channels is being written to, and the word being written appears on the Z80 Data Bus. Now everything is ready for the word to be latched into the appropriate CTC internal register in synchronization with the rising edge beginning clock cycle T_3. No additional wait states are allowed.

CTC WRITE CYCLE
Figure 6.1-0



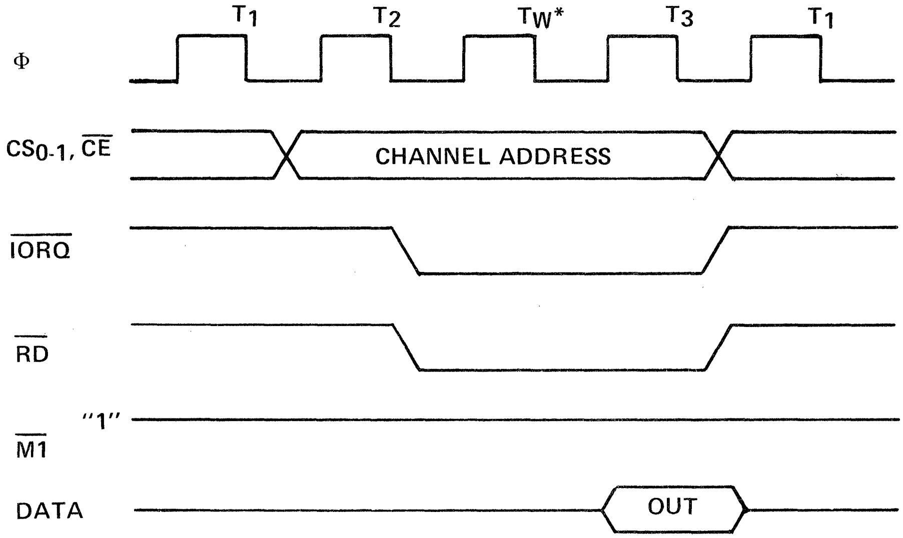
*AUTOMATICALLY INSERTED BY Z80-CPU

6.2 CTC READ CYCLE

Figure 6.2-0 illustrates the timing associated with the CTC Read Cycle. This sequence is used any time the CPU reads the current contents of the Down Counter. During clock cycle T_2, the Z80-CPU initiates the Read Cycle with true signals at input pins $\overline{RD}$ (Read), $\overline{IORQ}$ (I/O Request), and $\overline{CE}$ (Chip Enable). also at this time a 2-bit binary code appears at CTC inputs CS1 and CS0 (Channel Select 1 and 0), specifying which of the four CTC channels is being read from. (Note: $\overline{M1}$ must be false to distinguish the cycle form an interrupt acknowledge.) On the rising edge of the cycle T_3 the valid contents of the Down Counter as of the rising edge of cycle T_2 will be available on the Z80 Data Bus. No additional wait states are allowed.

CTC READ CYCLE

Figure 6.2-0



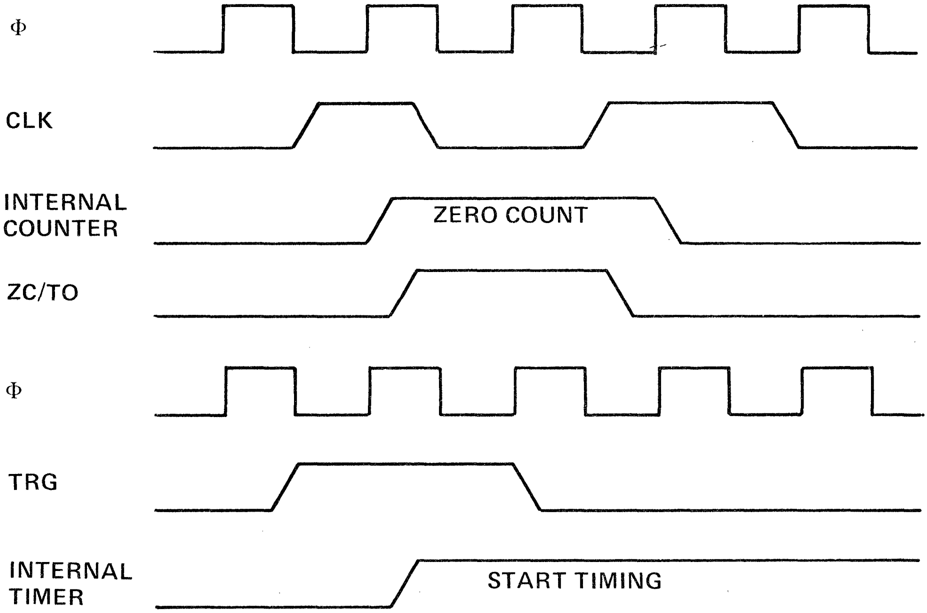
*AUTOMATICALLY INSERTED BY Z80-CPU

6.3 CTC COUNTING AND TIMING

Figure 6.3-0 illustrates the timing diagram for the CTC Counting and Timing Modes.

CTC COUNTING AND TIMING

Figure 6.3-0



In the Counter Mode, the edge (rising edge is active in this example) form the external hardware connected to pin CLK/TRG decrements the Down Counter in synchronization with the System Clock Φ. As specified in the A.C. Characteristics (Section 9.1) this CLK/TRG pulse must have a minimum width and the minimum period must not be less than twice the system clock period. Although there is no set-up requirement between the active edge of the CLK/TRG and the rising edge of Φ if the CLK/TRG edge occurs closer than a specified minimum time, the decrement of the Down Counter will be delayed one cycle of Φ. Immediately after the decrement of the Down Counter, 1 to 0, the ZC/TO output is pulsed true.

In the Timer Mode, a pulse trigger (user-selectable as either active high or active low) at the CLK/TRG pin enables timing function on the second succeeding rising edge of Φ. As in the Counter Mode, the triggering pulse is detected asynchronously and must have a minimum width. The timing function is initiated in syncronization with Φ, and a minimum set-up time is required between the active edge of the CLK/TRG and the next rising edge of Φ. If the CLK/TRG edge occurs closer than this, the initiation of the timer function will be delayed one cycle of Φ.

7.0 CTC INTERRUPT SERVICING

Each CTC channel may be individually programmed to request an interrupt every time its Down Counter reaches a count of zero. The purpose of a CTC-generated interrupt, as for any other peripheral device, is to force the CPU to execute an interrupt service routine. To utilize this feature the Z80-CPU must be programmed for mode 2 interrupt response. Under the requirements of this mode, when a CTC channel requests an interrupt and is acknowledged, a 16-bit pointer must be formed to obtain a corresponding interrupt service routine starting address from a table in memory. The lower 8 bits of the pointer are provided by the CTC in the form of an Interrupt Vector unique to the particular channel that requested the interrupt. (For further details, refer to Chapter 8.0 of the Z80-CPU Technical Manual.)

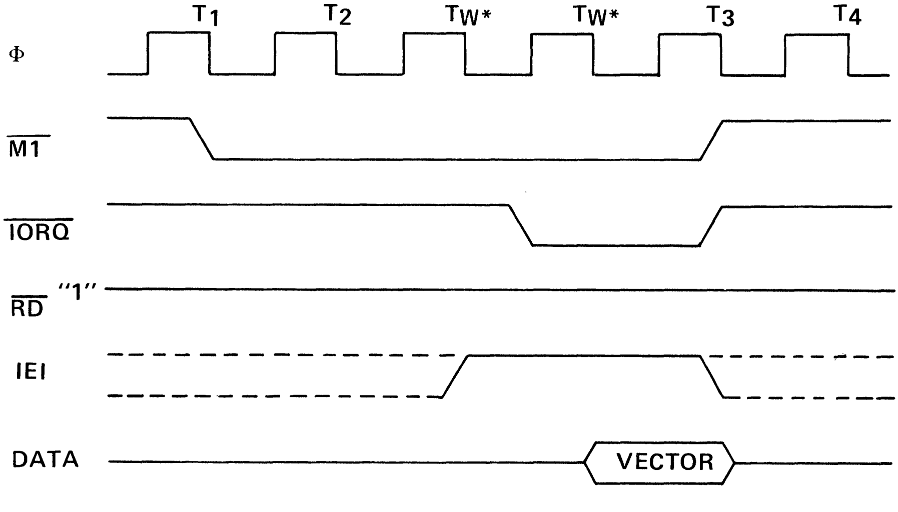
The CTC's Interrupt Control Logic insures that it acts in accordance with Z80 system interrupt protocol for nested priority interrupt and proper return from interrupt. The priority of any system device is determined by its physical location in a daisy chain configuration. Two signal lines (IEI and IEO) are provided in the CTC and all Z80 peripheral devices to form the system daisy chain. The device closest to the CPU has the highest priority; within the CTC, interrupt priority is predetermined by channel number, with channel 0 having highest priority. According to Z80 system interrupt protocol, low priority devices or channels may not interrupt higher priority devices or channels that have already interrupted and not had their interrupt service routines completed. However, high priority devices or channels may interrupt the servicing of lower priority devices or channels. (For further details, see section 2.3: "Interrupt Control Logic".)

Sections 7.1 and 7.2 below describe the nominal timing relationships of the relevant CTC pins for the Interrupt Acknowledge Cycle and the Return form Interrupt Cycle. Section 7.3 below discusses a typical example of daisy chain interrupt servicing.

7.1 INTERRUPT ACKNOWLEDGE CYCLE

Figure 7.1-0 illustrates the timing associated with the Interrupt Acknowledge Cycle. Some time after an interrupt is requested by the CTC, the CPU will send out an interrupt acknowledge ($\overline{M1}$ and $\overline{IORQ}$). To insure that the daisy chain enable lines stabilize, channels are inhibited from changing their interrupt request status when $\overline{M1}$ is active. $\overline{M1}$ is active about two clock cycles earlier than $\overline{IORQ}$, and $\overline{RD}$ is false to distinguish the cycle from an instruction fetch. During this time the interrupt logic of the CTC will determine the highest priority interrupting channel within the CTC places its Interrupt Vector onto the Data Bus when $\overline{IORQ}$ goes active. Two wait states (T_W*) are automatically inserted at this time to allow the daisy chain to stablize. Additional wait states may be added.

INTERRUPT ACKNOWLEDGE CYCLE
Figure 7.1-0

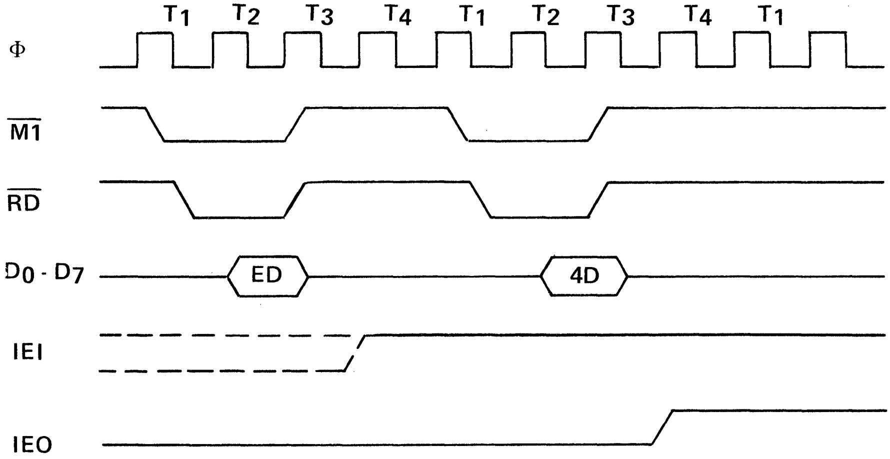


7.2 RETURN FROM INTERRUPT CYCLE

Figure 7.2-0 illustrates the timing associated with the RETI Instruction. This instruction is used at the end of an interrupt service routine to initialize the daisy chain enable lines for proper control of nested priority interrupt handling. The CTC decodes the two-byte RETI code internally and determines whether it is intended for a channel being serviced.

When several Z80 peripheral chips are in the daisy chain IEI will become active on the chip currently under service when an EDH opcode is decoded. If the following opcode is 4DH, the peripheral being serviced will be re-initialized and its IEO will become active. Additional wait states are allowed.

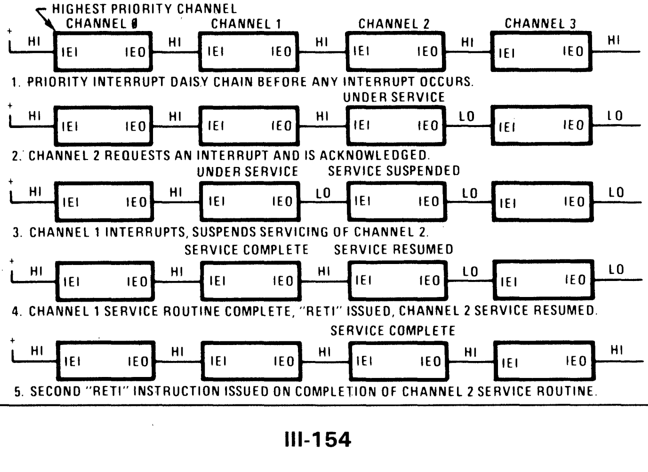
RETURN FROM INTERRUPT CYCLE
Figure 7.2-0



7.3 DAISY CHAIN INTERRUPT SERVICING

Figure 7.3-0 illustrates a typical nested interrupt sequence which may occur in the CTC. In this example, channel 2 interrupts and is granted service. While this channel is being serviced, higher priority channel 1 interrupts and is granted service. The service routine for the higher priority channel is completed, and a RETI instruction (see section 7.2 for further details) is executed to signal the channel that its routine is complete. At this time, the service routine of the lower priority channel 2 is resumed and completed.

DAISY CHAIN INTERRUPT SERVICING
Figure 7.3-0



1. PRIORITY INTERRUPT DAISY CHAIN BEFORE ANY INTERRUPT OCCURS.

2. CHANNEL 2 REQUESTS AN INTERRUPT AND IS ACKNOWLEDGED.

3. CHANNEL 1 INTERRUPTS, SUSPENDS SERVICING OF CHANNEL 2.

4. CHANNEL 1 SERVICE ROUTINE COMPLETE, "RETI" ISSUED, CHANNEL 2 SERVICE RESUMED.

5. SECOND "RETI" INSTRUCTION ISSUED ON COMPLETION OF CHANNEL 2 SERVICE ROUTINE.

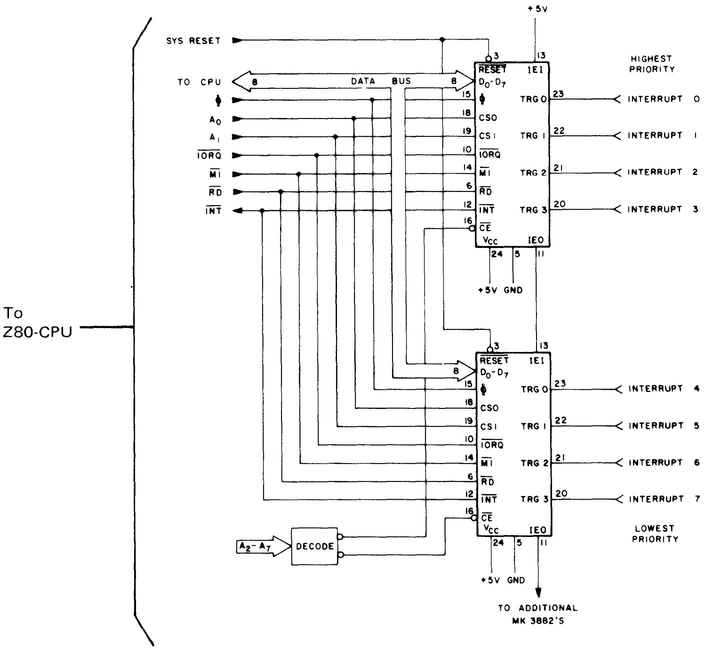
7.4 USING THE CTC AS AN INTERRUPT CONTROLLER

All of the Z80 family parts contain circuitry 'for prioritizing interrupts and supplying the vector to the CPU. However, in many Z80 based systems interrupts must be processed from devices which do not contain this interrupt circuitry. To handle this requirement the MK3882 CTC can be used, providing prioritized, independently vectored, maskable, edge selectable, count programmable external interrupt inputs. The MK3882 parts may be cascaded, expanding the system to as many as 256 interrupt inputs.

Each MK3882 contains 4 channels with counter inputs able to interrupt upon one or more (up to 256) edge transitions. The active transition may be programmed to be positive or negative. Each of the 4 channels has a programmable vector which is used in powerful Z80 mode 2 interrupt processing. When an interrupt is processed the vector is combined with the CPU I register to determine where the interrupt service routine start address is located. Additionally, priority resolution is handled within the MK3882 when more than one interrupt request is made simultaneously. When more than one MK3882 is used, the prioritizing is done, with the IEI/IEO chain resolving inter-chip priorities. Each channel can be independently "masked" by disabling that channel's local interrupt.

When programming the MK3882 to handle an input as a general purpose interrupt line, the channel is put in the counter mode, with the count set to 1, the active edge specified and the vector is loaded. When the programmed edge occurs a mode 2 interrupt will be generated by the CTC and the Z80-CPU can vector directly to the service routine for the non-Z80 peripheral device. Note that after the interrupt, the CTC down counter is automatically reloaded with a count of one and the CTC channel begins looking for another active edge after the RETI of the interrupt routine. Therefore, once a particular channel is under service, no active edges will be recognized by that channel until execution of the RETI instruction of the corresponding interrupt routine. Of course, other channels of the CTC can generate interrupts and/or pending interrupts asynchronously, depending on their priority.

CTC AS AN INTERRUPT CONTROLLER
Figure 7.4-0



8.0 ABSOLUTE MAXIMUM RATINGS

Temperature Under Bias	Specified Operating Range
Storage Temperature	-65°C to +150°C
Voltage on Any Pin with Respect to Ground	-0.3V to +7V
Power Dissipation	0.8V

8.1 D. C. CHARACTERISTICS

TA = 0°C to 70°C, Vcc = 5V ± 5% unless otherwise specified

SYMBOL	PARAMETER	MIN	MAX	UNIT	TEST CONDITION
V_{ILC}	Clock Input Low Voltage	–0.3	0.80	V	
V_{IHC}	Clock Input High Voltage (1)	V_{CC}–.6	V_{CC}+.3	V	
V_{IL}	Input Low Voltage	–0.3	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 2 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = –250 μA
I_{CC}	Power Supply Current		120	mA	T_C = 400 nsec**
I_{LI}	Input Leakage Current		±10	μA	V_{IN} = 0 to V_{CC}
I_{LOH}	Tri-State Output Leakage Current In Float		10	μA	V_{OUT} = 2.4 to V_{CC}
I_{LOL}	Tri-State Output Leakage Current In Float		–10	μA	V_{OUT} = 0.4V
I_{OHD}	Darlington Drive Current	–1.5		mA	V_{OH} = 1.5V

**T_C = 250 nsec for MK 3882-4

8.2 CAPACITANCE

TA = 25°C, f = 1 MHz

SYMBOL	PARAMETER	MAX	UNIT	TEST CONDITION
C_Φ	Clock Capacitance	20	pF	Unmeasured Pins
C_{IN}	Input Capacitance	5	pF	Returned to Ground
C_{OUT}	Output Capacitance	10	pF	

*COMMENT

Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

8.3 A.C. CHARACTERISTICS MK 3882, MK 3882-10, Z80-CTC

TA = 0° C to 70° C, Vcc = +5 V ± 5%, unless otherwise noted

Signal	Symbol	Parameter	Min	Max	Unit	Comments
Φ	t_C	Clock Period	400	(1)	ns	
	$t_W(\Phi H)$	Clock Pulse Width, Clock High	170	2000	ns	
	$t_W(\Phi L)$	Clock Pulse Width, Clock Low	170	2000	ns	
	tr, tf	Clock Rise and Fall Times		30	ns	
	t_H	Any Hold Time for Specified Setup Time	0		ns	
CS, $\overline{CE}$, etc.	$t_{S\Phi}(CS)$	Control Signal Setup Time to Rising Edge of Φ During Read or Write Cycle	160		ns	
D_0-D_7	$t_{DR}(D)$	Data Output Delay from Rising Edge of $\overline{RD}$ During Read Cycle		480	ns	(2)
	$t_{S\Phi}(D)$	Data Setup Time to Rising Edge of Φ During Write or M1 Cycle	60		ns	
	$t_{DI}(D)$	Data Output Delay from Falling Edge of IORQ During INTA Cycle		340	ns	(2)
	$t_F(D)$	Delay to Floating Bus (Output Buffer Disable Time)		230	ns	
IEI	$t_S(IEI)$	IEI Setup Time to Falling Edge of $\overline{IORQ}$ During INTA Cycle	200		ns	
IEO	$t_{DH}(IO)$	IEO Delay Time from Rising Edge of IEI		220	ns	(3)
	$t_{DL}(IO)$	IEO Delay Time from Falling Edge of IEI		190	ns	(3)
	$t_{DM}(IO)$	IEO Delay from Falling Edge of $\overline{M1}$ (Interrupt Occurring just Prior to $\overline{M1}$)		300	ns	(3)
$\overline{IORQ}$	$t_{S\Phi}(IR)$	$\overline{IORQ}$ Setup Time to Rising Edge of Φ During Read or Write Cycle	250		ns	
$\overline{M1}$	$t_{S\Phi}(M1)$	$\overline{M1}$ Setup Time to Rising Edge of Φ During INTA or M1 Cycle	210		ns	
$\overline{RD}$	$t_{S\Phi}(RD)$	$\overline{RD}$ Setup Time to Rising Edge of Φ During Read or M1 Cycle	240		ns	
$\overline{INT}$	$t_{DCK}(IT)$	$\overline{INT}$ Delay Time from Rising Edge of CLK/TRG		$2t_C(\Phi) + 200$		Counter Mode
	$t_{D\Phi}(IT)$	$\overline{INT}$ Delay Time from Rising Edge of Φ		$t_C(\Phi) + 200$		Timer Mode
CLK/TRG$_{0-3}$	$t_C(CK)$	Clock Period	$2t_C(\Phi)$			Counter Mode
	t_r, t_f	Clock and Trigger Rise and Fall Times		50	ns	
	$t_S(CK)$	Clock Setup Time to Rising Edge of Φ for Immediate Count	210		ns	Counter Mode
	$t_S(TR)$	Trigger Setup Time to Rising Edge of Φ for Enabling of Prescaler on Following Rising Edge of Φ	210		ns	Timer Mode
	$t_W(CTH)$	Clock and Trigger High Pulse Width	200		ns	Counter and Timer Modes
	$t_W(CTL)$	Clock and Trigger Low Pulse Width	200		ns	Counter and Timer Modes
ZC/TO$_{0-2}$	$t_{DH}(ZC)$	ZC/TO Delay Time from Rising Edge of Φ, ZC/TO High		190	ns	Counter and Timer Modes
	$t_{DL}(ZC)$	ZC/TO Delay Time from Falling Edge of Φ , ZC/TO Low		190	ns	Counter and Timer Modes

NOTES:

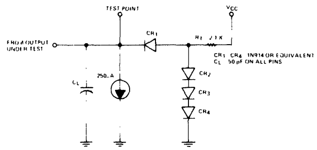
(1) $t_C = t_W(\Phi H) + t_W(\Phi L) + t_r + t_f$.

(2) Increase delay by 10 nsec for each 50 pF increase in loading 200pF maximum for data lines and 100pF for control lines.

(3) Increase delay by 2nsec for each 10pF increase in loading, 100pF maximum.

(4) $\overline{RESET}$ must be active for a minimum of 3 clock cycles.

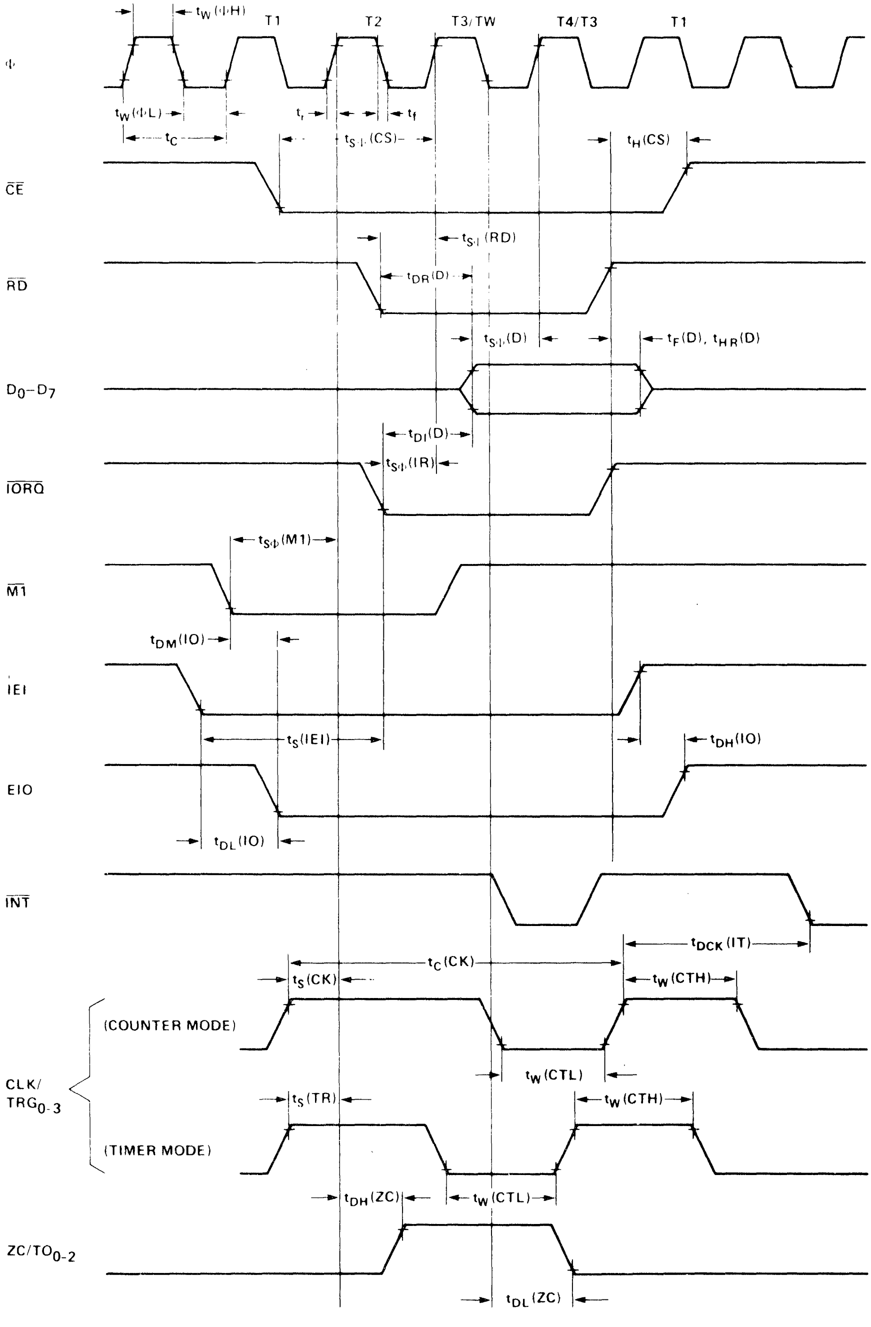
OUTPUT LOAD CIRCUIT



8.4 A. C. TIMING DIAGRAM

Timing measurements are made at the following voltages, unless otherwise specified

	"1"	"0"
CLOCK	V_{CC} - .6V	45V
OUTPUT	2.0V	.8V
INPUT	2.0V	.8V
FLOAT	ΔV	+0.5V



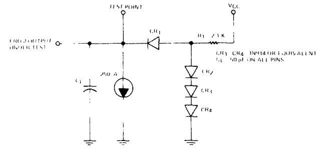
8.5 A.C. CHARACTERISTICS MK 3882-4,Z80A-CTC

TA = 0° C to 70° C, Vcc = +5 V ± 5%, unless otherwise noted

Signal	Symbol	Parameter	Min	Max	Unit	Comments
Φ	t_C	Clock Period	250	(1)	ns	
	$t_W(\Phi H)$	Clock Pulse Width, Clock High	105	2000	ns	
	$t_W(\Phi L)$	Clock Pulse Width, Clock Low	105	2000	ns	
	t_r, t_f	Clock Rise and Fall Times		30	ns	
	t_H	Any Hold Time for Specified Setup Time	0		ns	
CS, $\overline{CE}$, etc	$t_S\Phi(CS)$	Control Signal Setup Time to Rising Edge of Φ During Read or Write Cycle	145		ns	
D_0-D_7	$t_{DR}(D)$	Data Output Delay from Falling Edge of $\overline{RD}$ During Read Cycle		380	ns	(2)
	$t_S\Phi(D)$	Data Setup Time to Rising Edge of Φ During Write or M1 Cycle	50		ns	
	$t_{DI}(D)$	Data Output Delay form Falling Edge of IORQ During INTA Cycle		160	ns	(2)
	$t_F(D)$	Delay to Floating Bus (Output Buffer Disable Time)		110	ns	
IEI	$t_S(IEI)$	IEI Setup Time to Falling Edge of $\overline{IORQ}$ During INTA Cycle	140		ns	
IEO	$t_{DH}(IO)$	IEO Delay Time from Rising Edge of IEI		160	ns	(3)
	$t_{DL}(10)$	IEO Delay Time from Falling Edge of IEI		130	ns	(3)
	$t_{DM}(10)$	IEO Delay from Falling Edge of $\overline{M1}$ (Interrupt Occurring just Prior to $\overline{M1}$)		190	ns	(3)
$\overline{IORQ}$	$t_S\Phi(IR)$	$\overline{IORQ}$ Setup Time to Rising Edge of Φ During Read or Write Cycle	115		ns	
$\overline{M1}$	$t_S\Phi(M1)$	$\overline{M1}$ Setup Time to Rising Edge of Φ During INTA or M1 Cycle	90		ns	
$\overline{RD}$	$t_S\Phi(RD)$	$\overline{RD}$ Setup Time to Rising Edge of Φ During Read or M1 Cycle	115		ns	
$\overline{INT}$	$t_{DCK}(IT)$	$\overline{INT}$ Delay Time from Rising Edge of CLK/TRG		$2t_C(\Phi) + 140$		Counter Mode
	$t_D\Phi(IT)$	$\overline{INT}$ Delay Time from Rising Edge of Φ		$t_C(\Phi) + 140$		Timer Mode
CLK/TRG_{0-3}	$t_C(CK)$	Clock Period	$2t_C(\Phi)$		ns	Counter Mode
	t_r, t_f	Clock and Trigger Rise and Fall Times		30		
	$t_S(CK)$	Clock Setup Time to Rising Edge of Φ for Immediate Count	130		ns	Counter Mode
	$t_S(TR)$	Trigger Setup Time to Rising Edge of Φ for enabling of Prescaler on Following Rising Edge of Φ	130		ns	Timer Mode
	$t_W(CTH)$	Clock and Trigger High Pulse Width	120		ns	Counter and Timer Modes
	$t_W(CTL)$	Clock and Trigger Low Pulse Width	120		ns	Counter and Timer Modes
ZC/TO0-2	$t_{DH}(ZC)$	ZC/TO Delay Time from Rising Edge of Φ, ZC/TO High		120	ns	Counter and Timer Modes
ZC/TO_{0-2}	$t_{DL}(ZC)$	ZC/TO Delay Time from Falling Edge of Φ, ZC/TO Low		120	ns	Counter and Timer Modes

NOTES:
(1.) $t_C = t_W(\Phi H) + t_W(\Phi L) + t_r + t_f$.
(2.) Increase delay by 10 nsec for each 50 pF increase in loading, 200pF maximum for data lines and 100pF for control lines.
(3.) Increase delay by 2nsec for each 10pF increase in loading, 100pF maximum.
(4.) $\overline{RESET}$ must be active for a minimum of 3 clock cycles.

OUTPUT LOAD CIRCUIT



PART NO.	DESIGNATOR	PACKAGE TYPE	MAX CLOCK FREQUENCY	TEMPERATURE RANGE
MK3882N	Z80-PIO	Plastic	2.5 MHz	0° to 70°C
MK3882P	Z80-PIO	Ceramic	2.5 MHz	
MK3882-J	Z80-PIO	Cerdip	2.5 MHz	
MK3882N-4	Z80A-PIO	Plastic	4.0 MHz	
MK3882P-4	Z80A-PIO	Ceramic	4.0 MHz	
MK3882J-4	Z80A-PIO	Cerdip	4.0 MHz	
MK3882P-10	Z80-PIO	Ceramic	2.5 MHz	-40° to +85°C

MOSTEK®

Z80 MICROCOMPUTER COMPONENTS

Technical Manual

MK3884/5/7 SERIAL INPUT/OUTPUT

TABLE OF CONTENTS

SECTION **PAGE**

1.0 GENERAL INFORMATION

1.1 INTRODUCTION

The Mostek Z80-SIO (Serial Input/Output) is a dual-channel, multi-function peripheral component designed to satisfy a wide variety of serial data communications requirements in microcomputer systems. Its basic function is a serial-to-parallel, parallel-to-serial converter/controller, but, within that role, it is configurable by systems software so its "personality" can be optimized for a given serial data communications application.

The Z80-SIO is capable of handling asynchronous and synchronous byte-oriented protocols, such as IBM Bisync, and synchronous bit-oriented protocols, such as HDLC and IBM SDLC. This versatile device can also be used to support virtually any other serial protocol for applications other than data communications (cassette or floppy disk interface, for example).

The Z80-SIO can generate and check CRC codes in any synchronous mode and can be programmed to check data integrity in various modes. The device also has facilities for modem controls in both channels. In applications where these controls are not needed, the modem controls can be used for general-purpose I/O.

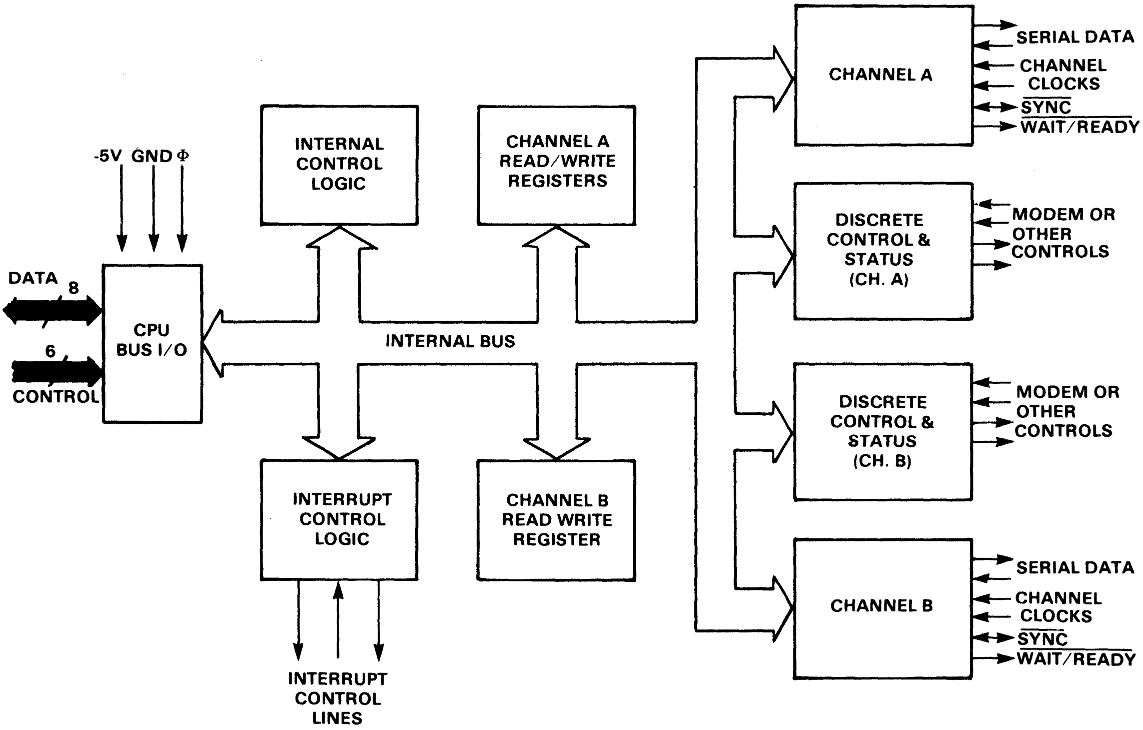
1.2 STRUCTURE

- N-channel silicon-gate depletion-load technology
- 40-pin DIP
- Single 5V power supply
- Single-phase 5V clock
- All inputs and outputs TTL compatible

1.3 FEATURES

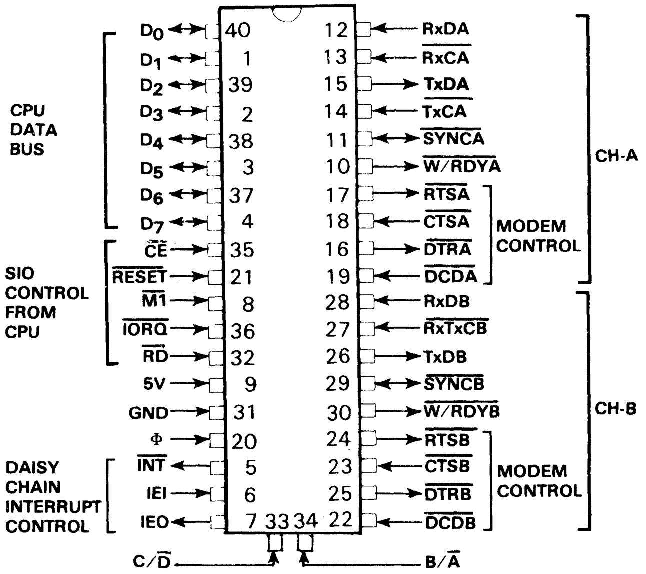
- Two independent full-duplex channels
- Data rates in synchronous or isosynchronous modes:
  0-550K bits/second with 2.5 MHz system clock rate
  0-800K bits/second with 4.0 MHz system clock rate
- Receiver data registers quadruply buffered; transmitter doubly buffered.
- Asynchronous features:
  - 5, 6, 7 or 8 bits/character
  - 1, 1 1/2 or 2 stop bits
  - Even, odd or no parity
  - x1, x16, x32 and x64 clock modes
  - Break generation and detection
  - Parity, overrun and framing error detection
  - Binary synchronous features:
    - Internal or external character synchronization
    - One or two sync characters in separate registers
    - Automatic sync character insertion
    - CRC generation and checking
  - HDLC and IBM SDLC-features:
    - Abort sequence generation and detection
    - Automatic zero insertion and deletion
    - Automatic flag insertion between messages
    - Address field recognition
    - I-field residue handling
    - Valid receive messages protected from overrun
    - CRC generation and checking
  - Separate modem control inputs and outputs for both channels
  - CRC-16 or CRC-CCITT block check
  - Daisy-Chain Priority interrupt logic provides automatic interrupt vectoring without external logic
  - Modem status can be monitored

Z80-SIO BLOCK DIAGRAM
Figure 1.0



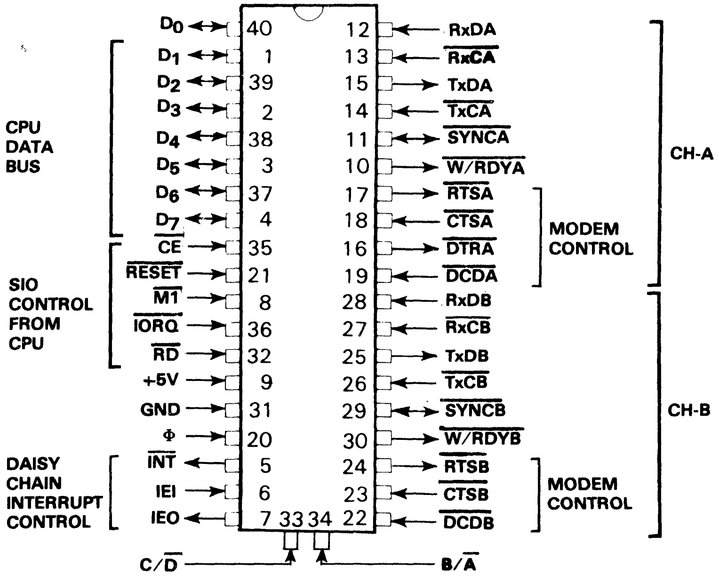
MK3884 PIN CONFIGURATION

Figure 1.1



MK3885 PIN CONFIGURATION

Figure 1.2



1.4 PIN DESCRIPTION

D_0-D_7. System Data Bus (bidirectional, 3-state). The system data bus transfers data and commands between the CPU and the Z80-SIO. D_0 is the least significant bit.

B/$\overline{A}$. Channel A Or B Select (input, High selects Channel B). This input defines which channel is accessed during a data transfer between the CPU and the Z80-SIO. Address bit A_0 from the CPU is often used for the selection function.

$C/\overline{D}$. Control Or Data Select (input, High selects Control). This input defines the type of information transfer performed between the CPU and the Z80-SIO. A High at this input, during a CPU write to the Z80-SIO, causes the information on the data bus to be interpreted as a command for the channel selected by $B/\overline{A}$. A Low at $C/\overline{D}$ means that the information on the data bus is data. Address bit A_1 is often used for this function.

$\overline{CE}$. Chip Enable (input, active Low). A Low level at this input enables the Z80-SIO to accept command or data inputs from the CPU during a write cycle, or to transmit data to the CPU during a read cycle.

Φ. System Clock (input). The Z80-SIO uses the standard Z80A System Clock to synchronize internal signals. This is a single-phase clock.

$\overline{M1}$. Machine Cycle One (input from Z80-CPU, active Low). When $\overline{M1}$ is active and $\overline{RD}$ is also active, the Z80-CPU is fetching an instruction from memory. When $\overline{M1}$ is active, while $\overline{IORQ}$ is active, the Z80-SIO accepts $\overline{M1}$ and $\overline{IORQ}$ as an interrupt acknowledge if the Z80-SIO is the highest priority device that has interrupted the Z80-CPU.

$\overline{IORQ}$. Input/Output Request (input from CPU, active Low). $\overline{IORQ}$ is used in conjunction with $B/\overline{A}$, $C/\overline{D}$, $\overline{CE}$ and $\overline{RD}$ to transfer commands and data between the CPU and the Z80-SIO. When $\overline{CE}$, $\overline{RD}$ and $\overline{IORQ}$ are all active, the channel selected by $B/\overline{A}$ transfers data to the CPU (a read operation). When $\overline{CE}$ and $\overline{IORQ}$ are active, but $\overline{RD}$ is inactive, the channel selected by $B/\overline{A}$ is written to by the CPU with either data or control information, as specified by $C/\overline{D}$. As mentioned previously, if $\overline{IORQ}$ and $\overline{M1}$ are active simultaneously, the CPU is acknowledging an interrupt and the Z80-SIO automatically places its interrupt vector on the CPU data bus, if it is the highest priority device requesting an interrupt.

$\overline{RD}$. Read Cycle Status. (input from CPU, active Low). If $\overline{RD}$ is active, a memory or I/O read operation is in progress. $\overline{RD}$ is used with $B/\overline{A}$, $\overline{CE}$ and $\overline{IORQ}$ to transfer data from the Z80-SIO to the CPU.

$\overline{RESET}$. Reset (input, active Low). A Low $\overline{RESET}$ disables both receivers and transmitters, forces TxDA and TxDB marking, forces the modem controls High and disables all interrupts. The control registers must be rewritten after the Z80-SIO is reset and before data is transmitted or received.

IEI. Interrupt Enable In (input, active High). This signal is used with IEO to form a priority daisy chain when there is more than one interrupt-driven device. A High on this line indicates that no other device of higher priority is being serviced by a CPU interrupt service routine.

IEO. Interrupt Enable Out (output, active High). IEO is High only if IEI is High and the CPU is not servicing an interrupt from this Z80-SIO. Thus, this signal blocks lower priority devices from interrupting while a higher priority device is being serviced by its CPU interrupt service routine.

$\overline{INT}$. Interrupt Request (output, open drain, active Low). When the Z80-SIO is requesting an interrupt, it pulls $\overline{INT}$ Low.

$\overline{W/RDYA}$, $\overline{W/RDYB}$. Wait/Ready A, Wait/Ready B (outputs, open drain when programmed for Wait function, driven High and Low when programmed for Ready function). These dual-purpose outputs may be programmed as Ready lines for a DMA controller or as Wait lines that synchronize the CPU to the Z80-SIO data rate. The reset state is open drain.

$\overline{CTSA}$, $\overline{CTSB}$. Clear To Send (inputs, active Low). When programmed as Auto Enables, a Low on these inputs enables the respective transmitter. If not programmed as Auto Enables, these inputs may be programmed as general-purpose inputs. Both inputs are Schmitt-trigger buffered to accommodate slow-risetime inputs. The Z80-SIO detects pulses on these inputs and interrupts the CPU on both logic level transitions. The Schmitt-trigger inputs do not guarantee a specified noise-level margin.

$\overline{DCDA}$, $\overline{DCDB}$. Data Carrier Detect (inputs, active Low). These signals are similar to the CTS inputs, except they can be used as receiver enables.

RxDA, RxDB. Receive Data (inputs, active High).

TxDA, TxDB. Transmit Data (outputs, active High).

$\overline{\mathrm{RxCA}}$, $\overline{\mathrm{RxCB}}$*. Receiver Clocks (inputs). See the following section on bonding options. The Receive Clocks may be 1, 16, 32 or 64 times the data rate in asynchronous modes. Receive data is sampled on the rising edge of $\overline{\mathrm{RxC}}$.

$\overline{\mathrm{TxCA}}$, $\overline{\mathrm{TxCB}}$*. Transmitter Clocks (inputs). See section on bonding options. In asynchronous modes, the Transmitter clocks may be 1, 16, 32 or 64 times the data rate. The multiplier for the transmitter and the receiver must be the same. Both the $\overline{\mathrm{TxC}}$ and $\overline{\mathrm{RxC}}$ inputs are Schmitt-trigger buffered for relaxed rise-and fall-time requirements (no noise margin is specified). TxD changes on the falling edge of $\overline{\mathrm{TxC}}$.

$\overline{\mathrm{RTSA}}$, $\overline{\mathrm{RTSB}}$. Request To Send (outputs, active Low). When the RTS bit is set, the RTS output goes low. When the RTS bit is reset in the Asynchronous mode, the output goes High after the transmitter is empty. In Synchronous modes, the $\overline{\mathrm{RTS}}$ pin strictly follows the state of the RTS bit. Both pins can be used as general purpose outputs.

$\overline{\mathrm{DTRA}}$, $\overline{\mathrm{DTRB}}$. Data Terminal Ready (outputs, active Low). See note on bonding options. These outputs follow the state programmed into the DTR bit. They can also be programmed as general-purpose outputs.

$\overline{\mathrm{SYNC\ A}}$, $\overline{\mathrm{SYNC\ B}}$. Synchronization (inputs/outputs, active Low). These pins can act either as inputs or outputs. In the Asynchronous Receive mode, they are inputs similar to $\overline{\mathrm{CTS}}$ and $\overline{\mathrm{DCD}}$. In this mode, the transitions on these lines affect the state of the Sync/Hunt status bits in RRO. In the External Sync mode, these lines also act as inputs. When external synchronization is achieved, $\overline{\mathrm{SYNC}}$ must be driven Low on the second rising edge of $\overline{\mathrm{RxC}}$ after that rising edge of $\overline{\mathrm{RxC}}$, on which the last bit of the sync character was received. In other words, after the sync pattern is detected, the external logic must wait for two full Receive Clock cycles to activate the $\overline{\mathrm{SYNC}}$ input. Once $\overline{\mathrm{SYNC}}$ is forced Low, it is wise to keep it Low until the CPU informs the external sync logic that synchronization has been lost or a new message is about to start. Character assembly begins on the rising edge of $\overline{\mathrm{RxC}}$ that immediately precedes the falling edge of $\overline{\mathrm{SYNC}}$ in the External Sync mode.

In the Internal Synchonrization mode (Monosync and Bisync), these pins act as outputs that are active during the part of the receive clock ($\overline{\mathrm{RxC}}$) cycle in which sync characters are recognized. The sync condition is not latched, so these outputs are active each time a sync pattern is recognized, regardless of character boundaries.

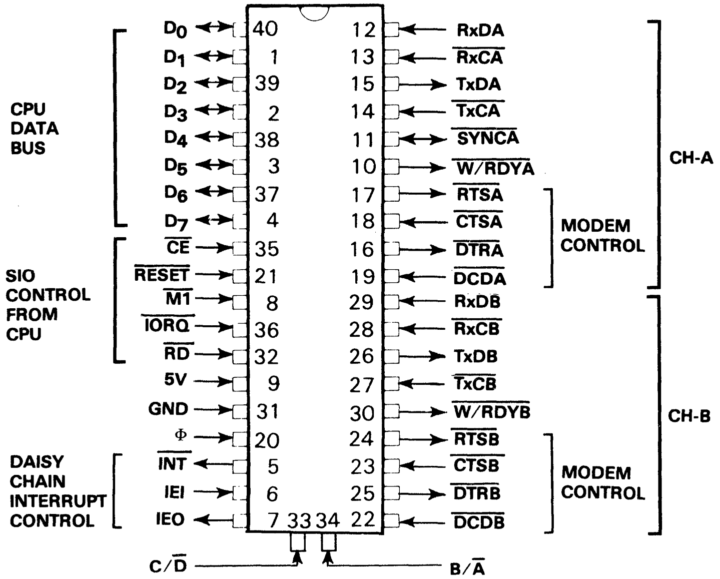
1.5 BONDING OPTIONS

The constraints of a 40-pin package make it impossible to bring out the Receive Clock, Transmit Clock, Data Terminal Ready and Sync signals for both channels. Therefore, Channel B must sacrifice a signal or have two signals bonded together. Since user requirements vary, three bondings options are offered:
MK3884 Z80-SIO has all four signals, but $\overline{\mathrm{TxCB}}$ and $\overline{\mathrm{RxCB}}$ are bonded together (Fig. 1.1).
MK3885 Z80-SIO sacrifices $\overline{\mathrm{DTRB}}$ and keeps $\overline{\mathrm{TxCB}}$, $\overline{\mathrm{RxCB}}$ and $\overline{\mathrm{SYNCB}}$ (Fig. 1.2).
MK3887 Z80-SIO sacrifices $\overline{\mathrm{SYNCB}}$ and keeps $\overline{\mathrm{TxCB}}$, $\overline{\mathrm{RxCB}}$ and $\overline{\mathrm{DTRB}}$ (Fig. 1.3).

*These clocks may be directly driven by the Z80-CTC (Counter Timer Circuit) for fully programmable baud rate generation.

MK3887 PIN CONFIGURATION

Figure 1.3



2.0 ARCHITECTURE

2.1 INTRODUCTION

The device internal structure includes a Z80-CPU interface, internal control and interrupt logic and two full-duplex channels. Associated with each channel are read and write registers and discrete control and status logic that provide the interface to modems or other external devices.

The read and write register group includes five 8-bit control registers, two sync-character registers and two status registers. The interrupt vector is written into an additional 8-bit register (Write Register 2) in Channel B that may be read through Read Register 2 in Channel B. The registers for both channels are designated in the text as follows:
WR0-WR7 -- Write Registers 0 through 7
RR0-RR2 -- Read Registers 0 through 2

The bit assignment and functional grouping of each register is configured to simplify and organize the programming process. Table 2.1 illustrates the functions assigned to each read or write register.

FUNCTIONAL ASSIGNMENTS OF READ AND WRITE REGISTERS
Table 2.1

WR0	Register pointers, CRC initialize, initialization commands for the various modes, etc.
WR1	Transmit/Receive interrupt and data transfer mode definition
WR2	Interrupt vector (Channel B only)
WR3	Receive parameters and controls
WR4	Transmit/Receive miscellaneous parameters and modes
WR5	Transmit parameters and controls
WR6	Sync character of SDLC address field
WR7	Sync character of SDLC flag
	(a) Write Register Functions
RR0	Transmit/Receive buffer status, interrupt status and external status
RR1	Special Receive Condition status
RR2	Modified interrupt vector (Channel B only)
	(b) Read Register Functions

The logic for both channels provides formats, synchronization and validation for data transferred to and from the channel interface. The modem control inputs, Clear to Send ($\overline{CTS}$) and Data Carrier Detect ($\overline{DCD}$), are monitored by the discrete control logic under program control. All the modem control signals are general purpose in nature and can be used for functions other than modem control.

For automatic interrupt vectoring, the interrupt control logic determines which channel and which device within the channel has the highest priority. Priority is fixed with Channel A assigned a higher priority than Channel B; Receive, Transmit and External/Status interrupts are prioritized in that order within each channel.

2.2 DATA PATH

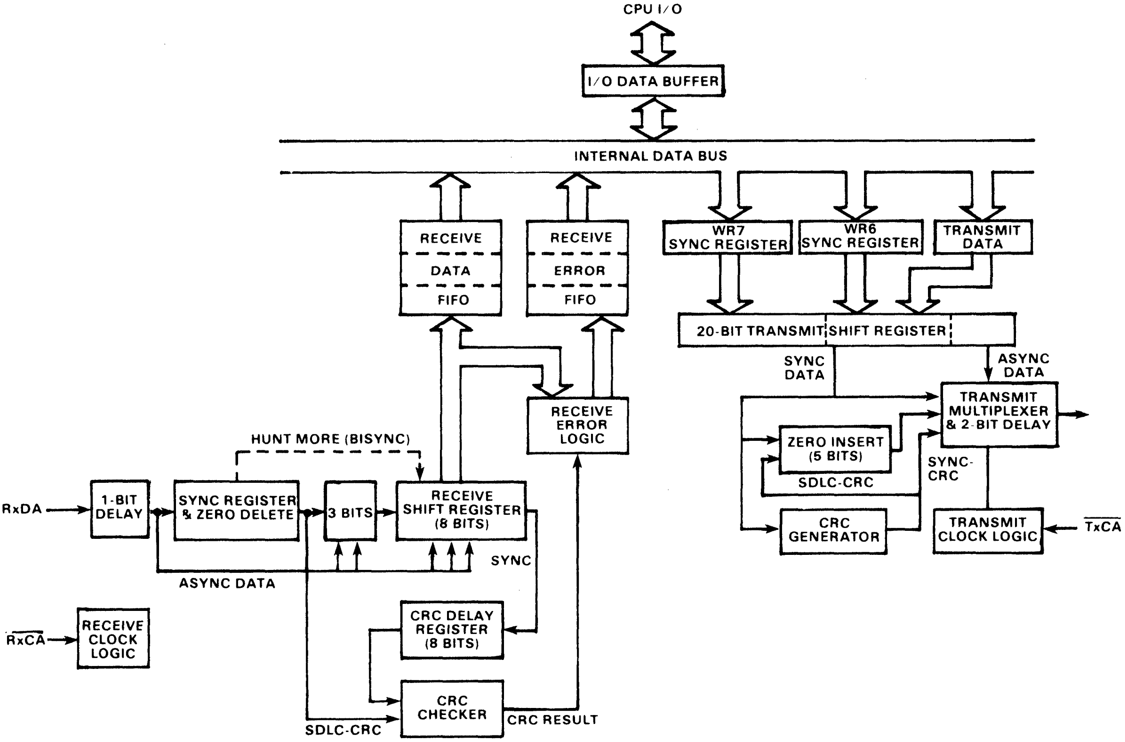
The transmit and receive data paths for each channel are shown in Figure 2.1. The receiver has three 8-bit buffer registers in a FIFO arrangement (to provide a 3-byte delay) in addition to the 8-bit receive shift register. This arrangement creates additional time for the CPU to service an interrupt at the beginning of a block of high-speed data. The receive error FIFO stores parity and framing errors and other types of status information for each of the three bytes in the receive data FIFO.

Incoming data is routed through one of several paths depending on the mode and character length. In the Asynchronous mode, serial data is entered into the 3-bit buffer if it has a character length of seven or eight bits, or is entered into the 8-bit receive shift register if it has a length of five or six bits.

In the Synchronous mode, however, the data path is determined by the phase of the receive process currently in operation. A Synchronous Receive operation begins with the receiver in the Hunt phase, during which the receiver searches the incoming data stream for a bit pattern that matches the

TRANSMIT AND RECEIVE DATA PATH

Figure 2.1



preprogrammed sync characters (or flags in the SDLC mode). If the device is programmed for Monosync Hunt, a match is made with a single sync character stored in WR7. In Bisync Hunt, a match is made with dual sync characters stored in WR6 and WR7.

In either case, the incoming data passes through the receive sync registers and is compared against the programmed sync character in WR6 or WR7. In the Monosync mode, a match between the sync character programmed into WR7 and the character assembled in the receive sync register establishes synchronization.

In the Bisync mode, however, incoming data is shifted to the receive shift register while the next eight bits of the message are assembled in the receive sync register. The match between the assembled character in the receive sync registers with the programmed sync character in WR6 and WR7 establishes synchronization. Once synchronization is established, incoming data bypasses the receive sync register and directly enters the 3-bit buffer.

In the SDLC mode, incoming data first passes through the receive sync register, which continuously monitors the receive data stream and performs zero deletion when indicated. Upon receiving five contiguous I's, the sixth bit is inspected. If the sixth bit is a 0, it is deleted from the data stream. If the sixth bit is a 1, the seventh bit is inspected. If that bit is a 0, a Flag sequence has been received; if it is a 1, an Abort sequence has been received.

The reformatted data enters the 3-bit buffer and is transferred to the receive shift register. Note that the SDLC receive operation also begins in the Hunt phase, during which the Z80-SIO tries to match the assembled character in the receive shift register with the flag pattern in WR7. Once the first flag character is recognized, all subsequent data is routed through the same path, regardless of character length.

Although the same CRC checker is used for both SDLC and synchronous data, the data path taken for each mode is different. In Bisync protocol, a byte-oriented operation requires that the CPU decide to include the data character in CRC. To allow the CPU ample time to make this decision, the Z80-SIO provides an 8-bit delay for sychronous data. In the SDLC mode, no delay is provided since the Z80-SIO contains logic that determines the bytes on which CRC is calculated.

The transmitter has an 8-bit transmit data register that is loaded from the internal data bus and a 20-bit transmit shift register that can be loaded from WR6, WR7 and the transmit data register. WR6 and WR7 contain sync characters in the Monosync or Bisync modes or address field (one character long) and flag, respectively, in the SDLC mode. During Synchronous modes, information contained in WR6 and WR7 is loaded into the transmit shift register at the beginning of the message and, as a time filler, in the middle of the message if a Transmit Underrun condition occurs. In the SDLC mode, the flags are loaded into the transmit shift register at the beginning and end of message.

Asynchronous data in the transmit shift register is formatted with start and stop bits and is shifted out to the transmit multiplexer at the selected clock rate.

Synchronous (Monosync or Bisync) data is shifted out to the transmit multiplexer and also to the CRC generator at the x 1 clock rate.

III
Z80 FAMILY
TECHNICAL
MANUALS

SDLC/HDLC data is shifted out through the zero insertion logic, which is disabled while the flags are being sent. For all other fields (address, control and frame check) a 0 is inserted following five contiguous 1's in the data stream. The CRC generator result for SDLC data is also routed through the zero insertion logic.

2.3 FUNCTIONAL DESCRIPTION

The functional capabilities of the Z80-SIO can be described from two different points of view: as a data communications device, it transmits and receives serial data, and meets the requirements of various data communications protocols; as a Z80 family peripheral, it interacts with the Z80-CPU and other Z80 peripheral circuits, and shares their data, address and control busses, as well as being a part of the Z80 interrupt structure. As a peripheral to other microprocessors, the Z80-SIO offers valuable features such as non-vectored interrupts, polling, and simple handshake capabilities.

The first part of the following functional description describes the interaction between the CPU and Z80-SIO; the second part introduces its data communications capabilities.

2.3.1 I/O CAPABILITIES

The Z80-SIO offers the choice of Polling, Interrupt (vectored or non-vectored) and Block Transfer modes to transfer data, status, and control information to and from the CPU. The Block Transfer mode can be implemented under CPU or DMA control.

Polling. The Polled mode avoids interrupts. Status registers RR0 and RR1 are updated at appropriate times for each function being performed (for example, CRC Error status valid at the end of the message). All the interrupt modes of the Z80-SIO must be disabled to operate the device in a polled environment.

While in its Polling sequence, the CPU examines the status contained in RR0 for each channel; the RR0 status bits serve as an acknowledge to the Poll inquiry. The two RR0 status bits D_0 and D_2 indicate that a receive or transmit data transfer is needed. The status also indicates Error or other special status conditions (see "Z80-SIO Programming"). The Special Receive Condition status continued in RR1 does not have to be read in a Polling sequence because the status bits in RR1 are accompanied by a Receive Character Available status in RR0.

Interrupts. The Z80-SIO offers an elaborate interrupt scheme to provide fast interrupt response in real-time applications. As mentioned earlier, Channel B registers WR2 and RR2 contain the interrupt vector that points to an interrupt service routine in the memory. To service operations in both channels and to eliminate the necessity of writing a status analysis routine, the Z80-SIO can modify the interrupt vector in RR2 so it points directly to one of eight interrupt service routines. This is done under program control by setting a program bit (WR1, D_2) in Channel B called "Status Affects Vector." When this bit is set, the interrupt vector in WR2 is modified according to the assigned priority of the various interrupting conditions. The table in the Write Register 1 description (Z80-SIO Programming section) shows the modification details.

Transmit interrupts, Receive interrupts and External/Status interrupts are the main sources of interrupts (Figure 5). Each interrupt source is enabled under program control with Channel A having a higher priority than Channel B, and with Receiver, Transmit and External/Status interrupts prioritized in that order within each channel. When the Transmit interrupt is enabled, the CPU is interrupted by the transmit buffer becoming empty. (This implies that the transmitter must have had a data character written into it so it can become empty.) When enabled, the receiver can interrupt the CPU in one of three ways:

Interrupt on first receive character
Interrupt on all receive characters
Interrupt on a Special Receive condition

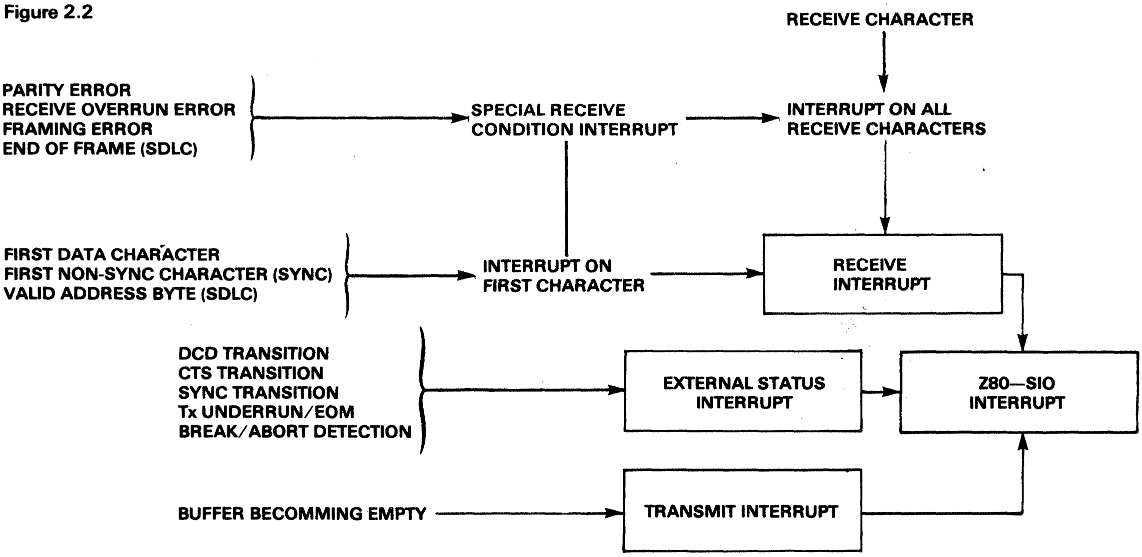
Interrupt On First Character is typically used with the Block Transfer mode.

Interrupt On All Receive Characters has the option of modifying the interrupt vector in the event of a parity error. The Special Receive Condition interrupt can occur on a character or message basis (End Of Frame interrupt in SDLC, for example). The Special Receive condition can cause an interrupt only if the Interrupt On First Receive Character or Interrupt On All Receive Characters mode is selected. In Interrupt On First Receive Character, an interrupt can occur from Special Receive conditions (except parity Error) after the first receive character interrupt (example: Receive Overrun interrupt).

The main function of the External/Status interrupt is to monitor the signal transitions of the $\overline{\text{CTS}}$, $\overline{\text{DCD}}$ and $\overline{\text{SYNC}}$ pins; however, an External/Status interrupt is also caused by a Transmit Underrun condition or by the detection of a Break (Asynchronous mode) or Abort (SDLC mode) sequence in the data stream. The interrupt caused by the Break/Abort sequence has a special feature that allows the Z80-SIO to interrupt when the Break/Abort sequence is detected or terminated. The feature facilitates the proper termination of the current message, correct initialization of the next message, and the accurate timing of the Break/Abort conditon in external logic.

INTERRUPT STRUCTURE

Figure 2.2



CPU/DMA Block Transfer. The Z80-SIO provides a Block Transfer mode to accommodate block transfer functions and DMA controllers (Z80-DMA or other designs). The Block Transfer mode uses the $\overline{\text{WAIT/READY}}$ output in conjunction with the Wait/Ready bits of Write Register 1. The $\overline{\text{WAIT/READY}}$ output can be defined under software control as a WAIT line in the CPU Block Transfer mode or as a READY line in the DMA Block Transfer mode.

To a DMA controller, the Z80-SIO READY output indicates that the Z80-SIO is ready to transfer data to or from memory. To the CPU, the WAIT output indicates that the Z80-SIO is not ready to transfer data, thereby requesting the CPU to extend the I/O cycle. The programming of bits 5, 6 and 7 of Write Register 1 and the logic states of the $\overline{\text{WAIT/READY}}$ line are defined in the Write Register 1 description (Z80-SIO Programming section.)

2.3.2 DATA COMMUNICATIONS CAPABILITIES

In addition to the I/O capabilities previously discussed, the Z80-SIO provides two independent full-duplex channels as well as Asynchronous, Synchronous and SDLC (HDLC) operational modes. These modes facilitate the implementation of commonly used data communications protocols.

The specific features of these modes are described in the following sections. To preserve the independence and completeness of each section, some information common to all modes is repeated.

3.0 ASYNCHRONOUS OPERATION

3.1 INTRODUCTION

To receive or transmit data in the Asynchronous mode, the Z80-SIO must be initialized with the following parameters: character length, clock rate, number of stop bits, even or odd parity, interrupt mode, and receiver or transmitter enable. The parameters are loaded into the appropriate write registers by the system program. WR4 parameters must be issued before WR1, WR3, and WR5 parameters or commands.

If the data is transmitted over a modem or RS232C interface, the $\overline{\text{REQUEST TO SEND}}$ ($\overline{\text{RTS}}$) and $\overline{\text{DATA TERMINAL READY}}$ ($\overline{\text{DTR}}$) outputs must be set along with the Transmit Enable bit. Transmission cannot begin until the Transmit Enable bit is set.

The Auto Enables feature allows the programmer to send the first data character of the message to the Z80-SIO without waiting for $\overline{\text{CTS}}$. If the Auto Enables bit is set, the Z80-SIO will wait for the $\overline{\text{CTS}}$ pin to go Low before it begins data transmission. $\overline{\text{CTS}}$, $\overline{\text{DCD}}$, and $\overline{\text{SYNC}}$ are general-purpose I/O lines that may be used for functions other than their labeled purposes. If $\overline{\text{CTS}}$ is used for another purpose, the Auto Enables Bit must be programmed to 0.

Figure 3.1 illustrates asynchronous message formats. Table 3.1 shows WR3, WR4, and WR5 with bits set to indicate the applicable modes, parameters, and commands in asynchronous modes. WR2 (Channel B only) stores the interrupt vector and WR1 defines the interrupt modes and data transfer modes. WR6 and WR7 are not used in asynchronous modes. Table 3.2 shows the typical program steps that implement a full-duplex receive/transmit operation in either channel.

3.2 ASYNCHRONOUS TRANSMIT

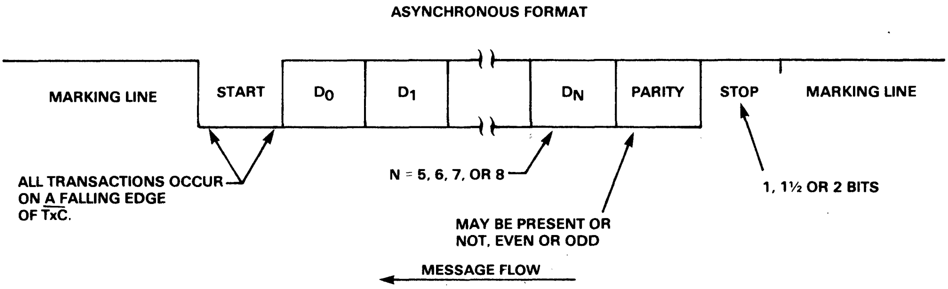
The Transmit Data output (TxD) is held marking (High) when the transmitter has no data to send. Under program control, the Send Break (WR5, D_4) command can be issued to hold TxD spacing (Low) until the command is cleared.

The Z80-SIO automatically adds the start bit, the programmed parity bit (odd, even or no parity) and the programmed number of stop bits to the data character to be transmitted. When the character length is six or seven bits, the unused bits are automatically ignored by the Z80-SIO. If the character length is five bits or less, refer to the table in the Write Register 5 description (Z80-SIO Programming section) for the data format.

Serial data is shifted from TxD at a rate equal to 1, 1/16th, 1/32nd, or 1/64th of the clock rate supplied to the Transmit Clock input ($\overline{\text{TxC}}$). Serial data is shifted out on the falling edge of ($\overline{\text{TxC}}$).

If set, the External/Status Interrupt mode monitors the status of $\overline{\text{DCD}}$, $\overline{\text{CTS}}$ and $\overline{\text{SYNC}}$ throughout the transmission of the message. If these inputs change for a period of time greater than the minimum specified pulse width, the interrupt is generated. In a transmit operation, this feature is used to monitor the modem control signal $\overline{\text{CTS}}$.

ASYNCHRONOUS MESSAGE FORMAT
Figure 3.1



3.3 ASYNCHRONOUS RECEIVE

Asynchronous Receive operation begins when the Receive Enable bit is set. If the Auto Enables option is selected, $\overline{\text{DCD}}$ must be Low as well. A Low (spacing) condition on the Receive Data input (RxD) indicates a start bit. If this Low persists for at least one-half of a bit time, the start bit is assumed to be valid and the data input is then sampled at mid-bit time until the entire character is assembled. This method of detecting a start bit improves error rejection when noise spikes exist on an otherwise marking line.

If the x1 clock mode is selected, bit synchronization must be accomplished externally. Receive data is sampled on the rising edge of RxC. The receiver inserts 1's when a character length of other than eight bits is used. If parity is enabled, the parity bit is not stripped from the assembled character for character lengths other than eight bits. For lengths other than eight bits, the receiver assembles a character length of the required number of data bits, plus a parity bit and 1's for any unused bits. For example, the receiver assembles a 5-bit character with the following format: 11 P D_4 D_3 D_2 D_1 D_0.

Since the receiver is buffered by three 8-bit registers in addition to the receive shift register, the CPU has enough time to service an interrupt and to accept the data character assembled by the Z80-SIO. The receiver also has three buffers that store error flags for each data character in the receive buffer. These error flags are loaded at the same time as the data characters.

After a character is received, it is checked for the following error conditions:

When parity is enabled, the Parity Error bit (RR1, D_4) is set whenever the parity bit of the character does not match with the programmed parity. Once this bit is set, it remains set until the Error Reset Command (WR0) is given.

CONTENTS OF WRITE REGISTERS 3, 4 and 5 in ASYNCHRONOUS MODES
Table 3.1

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
WR3	00 = Rx 5 BITS/CHAR 10 = Rx 6 BITS/CHAR 01 = Rx 7 BITS/CHAR 11 = Rx 8 BITS/CHAR		AUTO ENABLES	0	0	0	0	Rx ENABLE
WR 4	00 = x1 CLOCK MODE 01 = X16 CLOCK MODE 10 = x32 CLOCK MODE 11 = x64 CLOCK MODE		0	0	00 = NOT USED 01 = 1 STOP BIT/CHAR 10 = 1½ STOP BITS/CHAR 11 = 2 STOP BITS/CHAR		EVEN-$\overline{\text{ODD}}$ PARITY	PARITY ENABLE
WR5	DTR	00 = Tx 5 BITS (OR LESS) CHAR 10 = Tx 6 BITS/CHAR 01 = Tx 7 BITS/CHAR 11 = Tx 8 BITS/CHAR		SEND BREAK	Tx ENABLE	0	RTS	0

The Framing Error bit (RR1, D_6) is set if the character is assembled without any stop bits (that is a Low level detected for a stop bit). Unlike the Parity Error bit, this bit is set (and not latched) only for the character on which it occurred. Detection of framing error adds an additional one-half of a bit time to the character time so the framing error is not interpreted as a new start bit.

ASYNCHRONOUS MODE

Table 3.2

FUNCTION	TYPICAL PROGRAM STEPS		COMMENTS
	REGISTER:	**INFORMATION LOADED:**	
	WR0	CHANNEL RESET	Reset SIO
	WR0	POINTER 2	
	WR2	INTERRUPT VECTOR	Channel B only
	WR0	POINTER 4, RESET EXTERNAL/STATUS INTERRUPT	
	WR4	ASYNCHRONOUS MODE, PARITY INFORMATION, STOP BITS INFORMATION, CLOCK RATE INFORMATION	Issue Parameters
INITIALIZE	WR0	POINTER 3	
	WR3	RECEIVE ENABLE, AUTO ENABLES, RECEIVE CHARACTER LENGTH	
	WR0	POINTER 5	
	WR5	REQUEST TO SEND, TRANSMIT ENABLE, TRANSMIT CHARACTER LENGTH, DATA TERMINAL READY	Receive and Transmit both fully initialized. Auto Enables will enable Transmitter if $\overline{CTS}$ is active and Receiver if $\overline{DCD}$ is active.
	WR0	POINTER 1, RESET EXTERNAL/STATUS INTERRUPT	
	WR1	TRANSMIT INTERRUPT ENABLE, STATUS AFFECTS VECTOR, INTERRUPT ON ALL RECEIVE CHARACTERS, DISABLE WAIT/READY FUNCTION, EXTERNAL INTERRUPT ENABLE	Transmit/Receive interrupt mode selected. External Interrupt monitors the status $\overline{CTS}$. $\overline{DCD}$ and $\overline{SYNC}$ inputs and detects the Break sequence. Status Affects Vector in Channel B only.
	TRANSFER FIRST DATA BYTE TO SIO		This data byte must be transferred or no transmit interrupts will occur.
IDLE MODE		EXECUTE HALT INSTRUCTION OR SOME OTHER PROGRAM	Program is waiting for an interrupt from the SIO.
		Z80 INTERRUPT ACKNOWLEDGE CYCLE TRANSFERS RR2 TO CPU	When the interrupt occurs, the interrupt vector is modified by: 1. Receive Character Available; 2. Transmit Buffer Empty; 3. External/Status change; and 4. Special Receive condition.

		IF A CHARACTER IS RECEIVED:
		TRANSFER DATA CHARACTER TO CPU
		UPDATE POINTERS AND PARAMETERS RETURN FROM INTERRUPT	

DATA TRANSFER AND ERROR MONITORING		IF TRANSMITTER BUFFER IS EMPTY:
		TRANSFER DATA CHARACTER TO SIO
		UPDATE POINTERS AND PARAMETERS RETURN FROM INTERRUPT
			Program control is transferred to one of the eight interrupt service routines.

		IF EXTERNAL STATUS CHANGES:
		TRANSFER RR0 TO CPU
		PERFORM ERROR ROUTINES (INCLUDE BREAK DETECTION)
		RETURN FROM INTERRUPT
			If used with processors other than the Z80, the modified interrupt vector (RR2) should be returned to the CPU in the Interrupt Acknowledge sequence.

ASYNCHRONOUS MODE

Table 3.2

FUNCTION	TYPICAL PROGRAM STEPS	COMMENTS
REGISTER:	INFORMATION LOADED:	

	IF SPECIAL RECEIVE CONDITION OCCURS:
	TRANSFER RR1 to CPU
	DO SPECIAL ERROR (E.G. FRAMING ERROR)
	RETURN FROM INTERRUPT	

	REDEFINE RECEIVE/TRANSMIT INTERRUPT MODES	When transmit or receive data transfer is complete.
TERMINATION	DISABLE TRANSMIT/RECEIVE MODES	
	UPDATE MODEM CONTROL OUTPUTS (E.G. RTS OFF)	In transmit the All Sent Status bit indicates transmission is complete.

If the CPU fails to read a data character while more than three characters have been received, the Receive Overrun bit (RR1, D_5) is set. When this occurs, the fourth character assembled replaces the third character in the receive buffers. With this arrangement, only the character that has been written over is flagged with the Receive Overrun Error bit. Like Parity Error, this bit can only be reset by the Error Reset command from the CPU. Both the Framing Error and Receive Overrun Error cause an interrupt with the interrupt vector indicating a Special Receive condition (if Status Affects Vector is selected).

Since the Parity Error and Receive Overrun Error flags are latched, the error status that is read reflects an error in the current word in the receive buffer plus any Parity or Overrun Errors received since the last Error Reset command. To keep correspondence between the state of the error buffers and the contents of the receive data buffers, the error status register must be read before the data. This is easily accomplished if vectored interrupts are used, because a special interrupt vector is generated for these conditions.

While the External/Status interrupt is enabled, break detection causes an interrupt and the Break Detected status bit (RR0, D_7), is set. The Break Detected interrupt should be handled by issuing the Reset External/Status Interrupt command to the Z80-SIO in response to the first Break Detected interrupt that has a Break satus of 1 (RR0, D_7). The Z80-SIO monitors the Receive Data input and waits for the Break sequence to terminate, at which point the Z80-SIO interrupts the CPU with the Break status set to 0. The CPU must again issue the Reset External/Status Interrupt command in its interrupt service routine to reinitialize the break detection logic.

The External/Status interrupt also monitors the status of DCD. If the DCD pin becomes inactive for a period greater than the minimum specified pulse width, an interrupt is generated with the DCD status bit (RR0, D_3) set to 1. Note that the DCD input is inverted in the RR0 status register.

If the status is read after the data, the error data for the next word is also included if it has been stacked in the buffer. If operations are performed rapidly enough so the next character is not yet received, the status register remains valid. An exception occurs when the Interrupt On First Character Only mode is selected. A special interrupt in this Mode holds the error data and the character itself (even if read from the buffer) until the Error Reset command is issued. This prevents further data from becoming available in the receiver until the Reset command is issued, and allows CPU intervention on the character with the error even if DMA or block transfer techniques are being used.

If Interrupt On Every Character is selected, the interrupt vector is different if there is an error status in RR1. If a Receiver Overrun occurs, the most recent character received is loaded into the buffer; the character preceding it is lost. When the character that has been written over is read, the Receive Overrun bit is set and the Special Receive Condition vector is returned if Status Affects Vector is enabled.

In a polled environment, the Receive Character Available bit (RR0, D_0) must be monitored so the Z80-CPU can know when to read a character. This bit is automatically reset when the receive buffers are read. To prevent overwriting data in polled operations, the transmit buffer status must be checked before writing into the transmitter. The Transmit Buffer Empty bit is set to 1 whenever the transmit buffer becomes empty.

4.0 SYNCHRONOUS OPERATION

4.1 INTRODUCTION

Before describing synchronous transmission and reception, the three types of character synchronization-Monosync, Bisync, and External Sync-require some explanation. These modes use the x1 clock for both Transmit and Receive operations. Data is sampled on the rising edge of the Receive Clock input ($\overline{\text{RxC}}$). Transmitter data transitions occur on the falling edge of the Transmit Clock input ($\overline{\text{TxC}}$).

The differences between Monosync, Bisync, and External Sync are in the manner in which initial character synchronization is achieved. The mode of operation must be selected before sync characters are loaded because the registers are used differently in the various modes. Figure 4.1 shows the formats for all three of these synchronous modes.

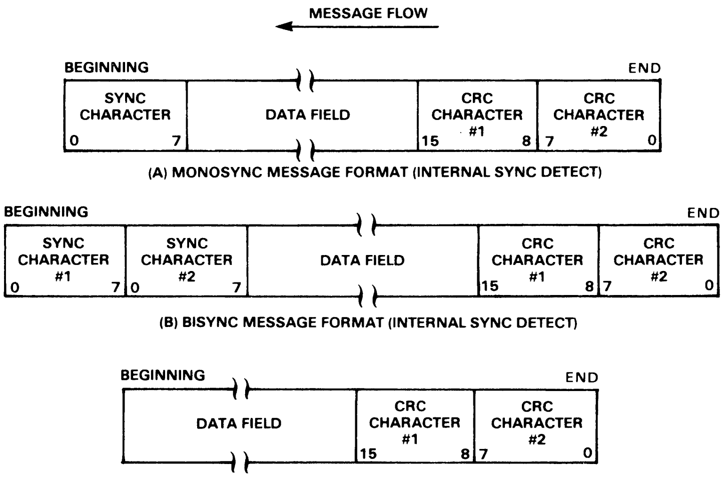
Monosync. In a Receive operation, matching a single sync character (8-bit sync mode) with the programmed sync character stored in WR7 implies character synchronization and enables data transfer.

Bisync. Matching two contiguous sync characters (16-bit sync mode) with the programmed sync characters stored in WR6 and WR7 implies character synchronization. In both the Monosync and Bisync modes, $\overline{\text{SYNC}}$ is used as an output and is active for the part of the receive clock that detects the sync character.

External Sync. In this mode, character synchronization is established externally; $\overline{\text{SYNC}}$ is an input that indicates external character synchronization has been achieved. After the sync pattern is detected, the external logic must wait for two full Receive Clock Cycles to activate the $\overline{\text{SYNC}}$ input. The $\overline{\text{SYNC}}$ input must be held low until character synchronization is lost. Character assembly begins on the rising edge of $\overline{\text{RxC}}$ that precedes the falling edge of $\overline{\text{SYNC}}$.

In all cases after a reset, the receiver is in the Hunt phase, during which the Z80-SIO looks for character synchronization. The hunt can begin only when the receiver is enabled, and data transfer can begin only when character synchronization has been achieved. If character synchronization is lost, the Hunt phase can be re-entered by writing a control word with the Enter Hunt Phase bit set (WR3, D_4). In the Transmit mode, the transmitter always sends the programmed number of sync bits (8 or 16). In the Monosync mode, the transmitter transmits for WR6; the receiver compares against WR7.

SYNCHRONOUS FORMATS
Figure 4.1



(A) MONOSYNC MESSAGE FORMAT (INTERNAL SYNC DETECT)

(B) BISYNC MESSAGE FORMAT (INTERNAL SYNC DETECT)

(C) EXTERNAL SYNC DETECT FORMAT

In the Monosync, Bisync,and External Sync modes, assembly of received data continues until the Z80-SIO is reset, or until the receiver is disabled (by command or $\overline{\text{DCD}}$ in the Auto Enables mode), or until the CPU sets the Enter Hunt Phase bit.

After initial synchronization has been achieved, the operation of the Monosync, Bisync, and External Sync modes is quite similar. Any differences are specified in the following text.

Table 4.1 shows how WR3, WR4,and WR5 are used in synchronous receive and transmit operations. WR0 points to other registers and issues various commands, WR1 defines the interrupt modes, WR2 stores the interrupt vector and WR6 and WR7 store sync characters. Table 4.2 illustrates the typical program steps that implement a half-duplex Bisync transmit operation.

CONTENTS OF WRITE REGISTERS 3, 4, AND 5 IN SYNCHRONOUS MODES
Table 4.1

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
WR3	00=Rx 5 BITS/CHAR 10=Rx 6 BITS/CHAR 01=Rx 7 BITS/CHAR 11=Rx 8 BITS/CHAR		AUTO ENABLES	ENTER HUNT MODE	Rx CRC ENABLE	0	SYNC CHAR LOAD INHIBIT	RX ENABLE
WR4	0	0	00=8-BIT SYNC CHAR 01=16-BIT SYNC CHAR 10=SDLC MODE 11=EXT SYNC MODE		0 SELECTS SYNC MODES	0	EVEN/$\overline{\text{ODD}}$ PARITY	PARITY ENABLE
WR5	DTR	00=Tx 5 BITS (OR LESS)/CHAR 10=Tx 6 BITS/CHAR 01=Tx 7 BITS/CHAR 11=Tx 8 BITS/CHAR		SEND BREAK	Tx ENABLE	1 SELECTS CRC-16	RTS	Tx CRC ENABLE

4.2 SYNCHRONOUS TRANSMIT

4.2.1 INITIALIZATION

The system program must initialize the transmitter with the following parameters: odd or even parity, x1 clock mode, 8- or 16-bit sync character(s), CRC polynomial, Transmitter Enables, Request To Send, Data Terminal Ready, interrupt modes,and transmit character length. WR4 parameters must be issued before WR1, WR3, WR5, WR6, and WR7 paramters or commands.

One of two polynomials CRC-16 ($x^{16} + X^{15} + X^2 + 1$) or SDLC ($X^{16} + X^{12} + X^5 + 1$) may be used with synchronous modes. In either case (SDLC mode not selected), the CRC generator and checker are reset to all 0's. In the transmit initialization process, the CRC generator is initialized by setting the Reset Transmit CRC Generator command bits (WR0). Both the transmitter and the receiver use the same polynomial.

Transmit Interrupt Enable or Wait/Ready Enable can be selected to transfer the data. The External/Status interrupt mode is used to monitor the status of the $\overline{\text{CLEAR TO SEND}}$ input as well as the Transmit Underun/EOM latch. Optionally, the Auto Enables feature can be used to enable the transmitter when $\overline{\text{CTS}}$ is active monitored so the Z80-CPU can know when to read a character. This bit is automatically reset when the receive buffers are read. To prevent overwriting data in polled operations, the transmit buffer status must be checked before writing into the transmitter. The Transmit Buffer Empty bit is set to 1 whenever the transmit buffer becomes empty.

The first data transfer to the Z80-SIO can begin when the External/Status interrupt occurs (CTS status bit set) or immediately following the Transmit Enable command (if the Auto Enables modes are set).

Transmit data is held marking after reset or if the transmitter is not enabled. Break may be programmed to generate a spacing line that begins as soon as the Send Break bit is set. With the transmitter fully initialized and enabled, the default condition is continuous transmission of the 8- or 16-bit sync character.

4.2.2 DATA TRANSFER AND STATUS MONITORING

In this phase, there are several combinations of interrupts and Wait/Ready.

Data Transfer Using Interrupts. If the Transmit Interrupt Enable bit (WR1, D_1) is set, an interrupt is generated each time the transmit buffer becomes empty. The interrupt can be satisfied either by writing another character into the transmitter or by resetting the Transmitter Interrupt Pending latch with a Reset Transmitter Pending command (WRO, CMD_5). If the interrupt is satisfied with this command and nothing more is written into the transmitter, there can be no further Transmit Buffer Empty interrupts, because it is the process of the buffer becoming empty that causes the interrupts. This situation does cause a Transmit Underrun condition, which is explained in the "Bisync Transmit Underrun" section.

Data Transfer Using WAIT/READY. To the CPU, the activation of $\overline{\text{WAIT}}$ indicates that the Z80-SIO is not ready to accept data and that the CPU must extend the output cycle. To a DMA controller, READY indicates that the transmit buffer is empty and that the Z80-SIO is ready to accept the next data character. If the data character is not loaded into the Z80-SIO by the time the transmit shift register is empty, the Z80-SIO enters the Transmit Underrun condition.

Bisync Transmit Underrun. In Bisync protocol, filler characters are inserted to maintain synchronization when the transmitter has no data to send (Transmit Underrun condition). The Z80-SIO has two programmable options for solving this situation: it can insert sync characters or it can send the CRC characters generated so far, followed by sync characters.

These options are under the control of the Reset Transmit Underrun/EOM command in WRO. Following a chip or channel reset, the Transmit Underrun/EOM status bit (RRO, D_6) is in a set condition and allows the insertion of sync characters when there is no data to send. CRC is not calculated on the automatically inserted sync characters. When the CPU detects the end of message, a Reset Transmit Underrun/EOM command can be issued. This allows CRC to be sent when the transmitter has no data. In this case, the Z80-SIO sends CRC, followed by sync characters, to terminate the message.

There is no restriction as to when in the message the Transmit Underrun/EOM bit can be reset. If Reset is issued after the first data character has been loaded, the 16-bit CRC is sent and followed by sync characters the first time the transmitter has no data to send. Because of the Transmit Underrun condition, an External/Status interrupt is generated whenever the Transmit Underrun/EOM bit becomes set.

In the case of sync insertion, an interrupt is generated only after the first automatically inserted sync character has been loaded. The status indicates the Transmit Underrun/EOM bit and the Transmit Buffer Empty bit are set.

In the case of CRC insertion, the Transmit Underrun/EOM bit is set and the Transmit Buffer Empty bit is reset while CRC is being sent. When CRC has been completely sent, the Transmit Buffer Empty status bit is set and an interrupt is generated to indicate to the CPU that another message can begin (this interrupt occurs because CRC has been sent and sync has been loaded). If no more messages are to be sent, the program can terminate transmission by resetting RTS and disabling the transmitter (WRS, D_3).

Pad characters may be sent by setting the Z80-SIO to 8-bits/transmit character and writing FF to the transmitter while CRC is being sent. Alternatively, the sync characters can be redefined as pad characters during this time. The following example is included to clarify this point.

The Z80-SIO interrupts with the Transmit Buffer Empty bit set.

The CPU recognizes that the last character (ETX) of the message has already been sent to the Z80-SIO by examining the internal program status.

To force the Z80-SIO to send CRC, the CPU issues the Reset Transmit Underrun/EOM Latch command (WRO) and satisfies the interrupt with the Reset Transmit Interrupt Pending command. (This command prevents the Z80-SIO from requesting more data.) Because of the transmit underrun caused by this command, the Z80-SIO starts sending CRC. The Z80-SIO also causes an External/Status interrupt with the Transmit Underrun/EOM latch set.

The CPU satisfies this interrupt by loading pad characters into the transmit buffer and issuing the Reset External/Status Interrupt command.

With this sequence, CRC is followed by a pad character instead of a sync character. Note that the Z80-SIO will interrupt with a Transmit Buffer Empty interrupt when CRC is completely sent and that the pad character is loaded into the transmit shift register.

From this point on, the CPU can send more pad characters or sync characters.

Bisync CRC Generation. Setting the Transmit CRC enable bit (WR5, D_O) initiates CRC accumulation when the program sends the first data character to the Z80-SIO. Although the Z80-SIO automatically transmits up to two sync characters (16-bit sync), it is wise to send a few more sync characters ahead of the message (before enabling Transmit CRC) to ensure synchronization at the receiving end.

The transmit CRC Enable bit can be changed on the fly any time in the message to include or exclude a particular data character from CRC accumulation. The Transmit CRC Enable bit should be in the desired state when the data character is loaded from the transmit data buffer into the transmit shift register. To ensure this bit is in the proper state, the Transmit CRC Enable bit must be issued before sending the data character to the Z80-SIO.

Transmit Transparent Mode. Transparent mode (Bisync protocol) operation is made possible by the ability to change Transmit CRC Enable on the fly and by the additional capability of inserting 16-bit sync characters. Exclusion of the DLE characters from CRC calculation can be acheived by disabling CRC calculation immediately preceding the DLE character transfer to the Z80-SIO.

In the case of a Transmit Underrun condition in the Transparent mode, a pair of DLE-SYN characters are sent. The Z80-SIO can be programmed to send the DLE-SYN sequence by loading a DLE character into WR6 and a sync character into WR7.

Transmit Termination. The Z80-SIO is equipped with a special termination that maintains data integrity and validity. If the transmitter is disabled while a data or sync character is being sent, that character is sent as usual, but is followed by a marking line rather than CRC or sync characters. When the transmitter is disabled, a character in the buffer remains in the buffer. If the transmitter is disabled while CRC is being sent, the 16-bit transmission is completed, but sync is sent instead of CRC.

A programmed break is effective as soon as it is written into the control register; characters in the transmit buffer and shift register are lost.

In all modes, characters are sent with the least significant bits first. This requires right-hand justification of transmitted data if the word length is less than eight bits. If the word length is five bits or less, the special technique described in the Write Register 5 discussion (Z80-SIO Programming section) must be used for the data format. The states of any unused bits in a data character are irrelevant, except when in the Five Bits or Less mode.

If the External/Status Interrupt Enable bit is set, transmitter conditions such as "starting to send CRC characters" "starting to send sync characters," and $\overline{CTS}$ changing state cause interrupts that have a unique vector if Status Affects Vector is set. This interrupt mode may be used during block transfers.

All interrupts may be disabled for operation in a Polled mode or to avoid interrupts at inappropriate times during the execution of a program.

4.3 SYNCHRONOUS RECEIVE

4.3.1 INITIALIZATION

The system program initiates the Synchronous Receive operation with the following parameters: odd or even parity, 8- or 16-bit sync characters, x1 clock mode, CRC polynomial, receive character length, etc. Sync characters must be loaded into registers WR6 and WR7. The receivers can be enabled only after all receive parameters are set. WR4 parameters must be issued before WR1, WR3, WR5, WR6 and WR7 parameters or commands.

After this is done, the receiver is in the Hunt phase. It remains in this phase until character synchronization is achieved. Note that, under program control, all the leading sync characters of the message can be inhibited from loading the receive buffers by setting the Sync Character Load Inhibit bit in WR3.

4.3.2 DATA TRANSFER AND STATUS MONITORING

After character synchronization is achieved, the assembled characters are transferred to the receive data FIFO. The following four interrupt modes are available to transfer the data and its associated status to the CPU.

No Interrupts Enabled. This mode is used for a purely polled operation or for off-line conditions.

Interrupt On First Character Only. This mode is normally used to start a polling loop or a Block Transfer instruction using $\overline{WAIT/READY}$ to synchronize the CPU or the DMA device to the incoming data rate. In this mode, the Z80-SIO interrupts on the first character and thereafter interrupts only if Special Receive conditions are detected. The mode is reinitialized with the Enable Interrupt on Next Receive Character command to allow the next character received to generate an interrupt. Parity errors do not cause interrupts in this mode, but End Of Frame (SDLC mode) and Receive Overrun do.

If External/Status interrupts are enabled, they may interrupt any time $\overline{DCD}$ changes state.

Interrupt On Every Character. Whenever a character enters the receive buffer, an interrupt is generated. Error and Special Receive conditions generate a special vector if Status Affects Vector is selected. Optionally, a Parity Error may be directed not to generate the special interrupt vector.

Special Receive Condition Interrupts. The Special Receive Condition interrupt can occur only if either the Receive Interrupt On First Character Only or Interrupt On Every Receive Character modes are also set. The Special Receive Condition interrupt is caused by the Receive Overrun error condition. Since the Receive Overrun and Parity error status bits are latched, the error status-when read-reflects an error in the current word in the receive buffer in addition to any Parity or Overrun errors received since the last Error Reset command. These status bits can only be reset by the Error reset command.

CRC Error Checking and Termination. A CRC error check on the receive message can be performed on a per character basis under program control. The Receive CRC Enable bit (WR3, D_3) must be set/reset by the program before the next character is transferred from the receive shift register into the receive buffer register. This ensures proper inclusion or exclusion of data characters in the CRC check.

In the Monosync, Bisync, and External Sync modes, the CRC/Framing Error bit (RR1, D_6) contains the comparison result of the CRC checker 16 bit times (eight bits delay and eight shifts for CRC) after the character has been transferred from the receive shift register to the buffer. The result should be zero, indicating an error-free transmission. (Note that the result is valid only at the end of CRC calculation. If the result is examined before this time, it usually indicates an error.) The comparison is made with each transfer and is valid only as long as the character remains in the receive FIFO.

Following is an example of the CRC checking operation when four characters (A,B,C, and D) are received in that order.

Character A loaded into buffer
Character B loaded into buffer

If CRC is disabled before C is in the buffer, CRC is not calculated on B.

Character C loaded into buffer

After C is loaded, the CRC/Framing Error bit shows the result of the comparison through character A.

Character D loaded into buffer

After D is in the buffer, the CRC Error bit shows the result of the comparison through character B whether or not B was included in the CRC calculations.

Due to the serial nature of CRC calculation, the Receive Clock (RxC) must cycle 16 times (8-bit delay plus 8-bit CRC shift) after the second CRC character has been loaded into the receive buffer, or 20 times (the previous 16 plus 3-bit buffer delay and 1-bit input delay) after the last bit is at the RxD input, before CRC calculation is complete. A faster external clock can be gated into the Receive Clock input to supply the required 16 cycles. The Transmit and Receive Data Path diagram (Figure 4) illustrates the various points of delay in the CRC path.

The typical program steps that implement a half-duplex Bisync Receive mode are illustrated in Table 6. The complete set of command and status bit definitions are explained under "Z80-SIO Programming."

BISYNC TRANSMIT CODE
Table 4.2

FUNCTION	TYPICAL PROGRAM STEPS		COMMENTS
	REGISTER:	**INFORMATION LOADED:**	
	WR0	CHANNEL RESET, RESET TRANSMIT CRC GENERATOR	Reset SIO, initialize CRC generator
	WR0	POINTER 2	
	WR2	INTERRUPT VECTOR	Channel B only
	WR0	POINTER 3	
	WR3	AUTO ENABLES	Transmission begins only after CTS is detected.
	WR0	POINTER 4	
	WR4	PARITY INFORMATION, SYNC MODES INFORMATION, x1 CLOCK MODE	Issue transmit parameters.
	WR0	POINTER 6	
	WR6	SYNC CHARACTER 1	
	WR0	POINTER 7, RESET EXTERNAL/STATUS INTERRUPTS	

FUNCTION		TYPICAL PROGRAM STEPS	COMMENTS
INITIALIZE	WR7	SYNC CHARACTER 2	
	WR0	POINTER 1, RESET EXTERNAL/STATUS INTERRUPTS	
	WR1	STATUS AFFECTS VECTOR, EXTERNAL INTERRUPT ENABLE, TRANSMIT INTERRUPT ENABLE OR WAIT/READY ENABLE	External Interrupt mode Monitors the status of $\overline{CTS}$ and $\overline{DCD}$ input pins as well as the status of Tx Underrun/EOM latch. Transmit Interrupt Enable interrupts when the Transmit buffer becomes empty; the Wait/Ready mode can be used to transfer data using DMA or CPU Block Transfer.
	WR0	POINTER 5	Status Affects Vector (Channel B only)
	WR5	REQUEST TO SEND, TRANSMIT ENABLE, BISYNC CRC, TRANSMIT CHARACTER LENGTH	Transmit CRC Enable should be set when first non-sync data is sent to Z80-SIO.
		FIRST SYNC BYTE TO SIO	Need several sync characters in the beginning of message. Transmitter is fully initialized.
IDLE MODE		EXECUTE HALT INSTRUCTION OR SOME OTHER PROGRAM	Waiting for interrupt or Wait/Ready output to transfer data.
DATA TRANSFER AND STATUS MONITORING		**WHEN INTERRUPT (WAIT/READY) OCCURS:** INCLUDE/EXCLUDE DATA BYTE FROM CRC ACCUMULATION (IN SIO). TRANSFER DATA BYTE FROM CPU (OR MEMORY) TO SIO. DETECT AND SET APPROPRIATE FLAGS FOR CONTROL CHARACTERS (IN CPU). RESET Tx UNDERRUN/EOM LATCH (WR0) IF LAST CHARACTER OF MESSAGE IS DETECTED. UPDATE POINTERS AND PARAMETERS (CPU). RETURN FROM INTERRUPT.	Interrupt occurs (Wait/Ready becomes active) when first data byte is being sent. Wait mode allows CPU block tranfer from memory to SIO; Ready mode allows DMA block transfer from memory to SIO. The DMA chip can be programmed to capture special control characters (by examining only the bits that specify ASCII or EBCDIC control characters) and interrupt CPU.

		IF ERROR CONDITION OR STATUS CHANGE OCCURS:
		TRANSFER RR0 TO CPU
		EXECUTE ERROR ROUTINE
		RETURN FROM INTERRUPT
			Tx Underrun/EOM indicates either transmit underrun (sync character being) sent to end of message (CRC-16 being sent)

		REDEFINE INTERRUPT MODES.	
TERMINATION		UPDATE MODEM CONTROL OUTPUTS (E.G., TURN OFF RTS).	Program should gracefully terminate message.
		DISABLE TRANSMIT MODE	

BISYNC RECEIVE MODE

Table 4.3

FUNCTION		TYPICAL PROGRAM STEPS	COMMENTS
	REGISTER:	**INFORMATION LOADED**	
	WR0	CHANNEL RESET, RESET RECEIVE CRC CHECKER	Reset SIO, initialize Receive CRC checker
	WR0	POINTER 2	
	WR2	INTERRUPT VECTOR	Channel B only
	WR0	POINTER 4	
	WR4	PARITY INFORMATION, SYNC MODES INFRMATION, x1 CLOCK MODE	Issue receive parameters.
	WR0	POINTER 5, RESET EXTERNAL STATUS INTERRUPT	
	WR5	BISYNC CRC-16 DATA TERMINAL READY	
	WR0	POINTER 3	
INITIALIZE	WR3	SYNC CHARACTER LOAD INHIBIT, RECEIVE CRC ENABLE, ENTER HUNT MODE, AUTO ENABLES, RECEIVE CHARACTER LENGTH	Sync character load inhibit strips all the loading sync characters at the beginning of the message. Auto Enables enables the receiver to accept data only after the $\overline{\text{DCD}}$ input is active
	WR0	POINTER 6	
	WR6	SYNC CHARACTER 1	
	WR0	POINTER 7	
	WR7	SYNC CHARACTER 2	
	WR0	POINTER 1, RESET EXTERNAL/STATUS INTERRUPT	
	WR1	STATUS AFFECTS VECTOR, EXTERNAL INTERRUPT ENABLE, RECEIVE INTERRUPT ON FIRST CHARACTER ONLY	In this interrupt mode, only the first non-sync data character is transferred to the CPU. All subsequent data is transferred on a DMA basis; however, Special Receive Condition interrupts will interrupt the CPU. Status Affects Vector used in Channel B only.
	WR0	POINTER 3, ENABLE INTERRUPT ON NEXT RECEIVE CHARACTER	Resetting this interrupt mode provides simple program loopback entry for the next transaction.
	WR3	RECEIVE ENABLE SYNC CHARACTER LOAD INHIBIT, ENTER HUNT MODE, AUTO ENABLE, RECEIVE WORD LENGTH	WR3 is reissued to enable receiver. Receive CRC Enable must be set after receiving SOH or STX character.
IDLE MODE		EXECUTE HALT INSTRUCTION OR SOME OTHER PROGRAM	Receive mode is fully initialized and the system is waiting for interrupt on first character.

FUNCTION	TYPICAL PROGRAM STEPS	COMMENTS
	WHEN INTERRUPT ON FIRST CHARACTER OCCURS, THE CPU DOES THE FOLLOWING: • TRANSFERS DATA BYTE TO CPU • DETECTS AND SETS APPROPRIATE FLAGS FOR CONTROL CHARACTERS (IN CPU) • INCLUDES/EXCLUDES DATA BYTE IN CRC CHECKER • UPDATES POINTERS AND OTHER PARAMETERS • ENABLES WAIT/READY FOR DMA OPERATION • ENABLES DMA CONTROLLER • RETURNS FOR INTERRUPT	During the Hunt mode, the SIO detects two contiguous characters to establish synchronization. The CPU establishes the DMA mode and all subsequent data characters are transferred by the DMA controller. The controller is also programmed to capture special characters (by examining only the bits that specify ASCII or EBCDIC control characters) and interrupt the CPU upon detection. In response, the CPU examines the status or control characters and takes appropriate action (e.g., CRC Enable Update)
DATA TRANSFER AND STATUS MONITORING	**WHEN WAIT/READY BECOMES ACTIVE, THE DMA CONTROLLER DOES THE FOLLOWING:** • TRANSFERS DATA BYTE TO MEMORY • INTERRUPTS CPU IF A SPECIAL CHARACTER IS CAPTURED BY THE DMA CONTROLLER • INTERRUPTS THE CPU IF THE LAST CHARACTER OF THE MESSAGE IS DETECTED	
	FOR MESSAGE TERMINATION, THE CPU DOES THE FOLLOWING: • TRANSFERS RR1 TO THE CPU • SETS ACK/NAK REPLY FLAG BASED ON CRC RESULT • UPDATES POINTERS AND PARAMETERS • RETURNS FROM INTERRUPT	The SIO interrupts the CPU for error condition and the error routine aborts the present message, clears the error condition and repeats the operation.
TERMINATION	REDEFINE INTERRUPT MODES AND SYNC MODES UPDATE MODEM CONTROLS DISABLES RECEIVE MODE	

5.0 SDLC (HDLC OPERATION)

5.1 INTRODUCTION

The Z80-SIO is capable of handling both High-level Synchronous Data Link Control (HDLC) and IBM Synchronous Data Link Control (SDLC) protocols. In the following discussion, only SDLC is referred to because of the high degree of similarity between SDLC and HDLC.

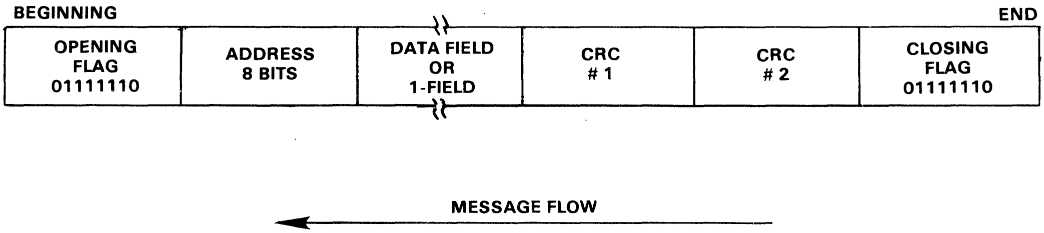
The SDLC mode is considerably different than Synchronous Bisync protocol because it is bit oriented rather than character oriented and, therefore, can naturally handle transparent operation. Bit orientation makes SDLC a flexible protocol in terms of message length and bit patterns. The Z80-SIO has several built-in features to handle variable message length. Detailed information concerning SDLC protocol can be found in literature published on this subject, such as IBM document GA27-3093.

The SDLC message, called the frame (Figure 8), is opened and closed by flags that are similar to the sync characters in Bisync protocol. The Z80-SIO handles the transmission and recognition of the flag characters that mark the beginning and end of the frame. Note that the Z80-SIO can receive shared-zero flags, but cannot transmit them. The 8-bit address field of a SDLC frame contains the secondary station address. The Z80-SIO has an Address Search mode that recognizes the secondary station so that it can accept or reject the frame.

Since the control field of the SDLC frame is transparent to the Z80-SIO, it is simply transferred to the CPU. The Z80-SIO handles the Frame Check sequence in a manner that simplifies the program by incorporating features such as initializing the CRC generator to all 1's, resetting the CRC checker when the opening flag is detected in the Receive mode,and sending the Frame Check/Flag sequence in the Transmit mode. Controller hardware is simplified by automatic zero insertion and deletion logic contained in the Z80-SIO.

Table 5.1 shows the contents of WR3,WR4,and WR5 during SDLC Receive and Transmit modes. WR0 points to other registers and issues various commands. WR1 defines the interrupt modes and WR2 stores the interrupt vector. WR7 stores the flag character and WR6 stores the secondary address.

TRANSMIT/RECEIVE SDLC/HDLCMESSAGE FORMAT
Figure 5.1



OPENING FLAG 01111110	ADDRESS 8 BITS	DATA FIELD OR 1-FIELD	CRC # 1	CRC # 2	CLOSING FLAG 01111110

5.2 SDLC TRANSMIT

5.2.1 INITIALIZATION

Like Synchronous operation, the SDLC Transmit mode must be initialized with the following parameters: SDLC mode, SDLC polynomial, Request to Send, Data Terminal Ready, transmit character length, transmit interrupt modes (or Wait/Ready function), Transmit Enable, Auto Enables and External/Status interrupt.

Selecting the SDLC mode and the SDLC polynomial enables the Z80-SIO to initialize the CRC Generator to all 1's. This is accomplished by issuing the Reset Transmit CRC Generator command (WR0). Refer to the Synchronous Operation section for more details on the interrupt modes.

After reset, or when the transmitter is not enabled, the Transmit Data output is held marking. Break may be programmed to generate a spacing line. With the transmitter fully initialized and enabled, continuous flags are transmitted on the Transmit Data output.

An abort sequence may be sent by issuing the Send Abort command (WR0, CMD_1). This causes at least eight, but less than fourteen, 1's to be sent before the line reverts to continuous flags. It is possible that the Abort sequence (eight 1's) could follow up to five continuous 1 bits (allowed by the zero insertion logic) and, thus, cause up to thirteen 1's to be sent. Any data being transmitted and any data in the transmit buffer is lost when an abort is issued.

When required, an extra 0 is automatically inserted when there are five contiguous 1's in the data stream. This does not apply to flags or aborts.

5.2.2 DATA TRANSFER AND STATUS MONITORING

There are several combinations of interrupts and the Wait/Ready function in the SLDC mode.

Data Transfer Using Interrupts. If the Transmit Interrupt Enable bit is set, an interrupt is generated each time the buffer becomes empty. The interrupt may be satisfied either by writing another character into the transmitter or by resetting the Transmit Interrupt Pending latch with a Reset Transmitter Pending command (WR0, CMD_5). If the interrupt is satisfied with this command and nothing more is written into the transmitter, there are no further transmitter interrupts. The result is a Transmit Underrun condition. When another character is written and sent out, the transmitter can again become empty and interrupt the CPU. Following the flags in an SDLC operation, the 8-bit address field, control field and information field may be sent to the Z80-SIO using the Transmit Interrupt mode. The Z80-SIO transmits the Frame Check sequence using the Transmit Underrun feature.

When the transmitter is first enabled, it is already empty and obviously cannot then become empty. Therefore, no Transmit Buffer Empty interrupts can occur until after the first data character is written.

Data Transfer Using Wait/Ready. If the Wait/Ready function has been selected, WAIT indicates to the CPU that the Z80-SIO is not ready to accept the data and the CPU must extend the I/O cycle. To a DMA controller, READY indicates that the transmitter buffer is empty and that the Z80-SIO is ready to accept the next character. If the data character is not loaded into the Z80-SIO by the time the transmit shift register is empty, the Z80-SIO enters the Transmit Underrun condition. Address,control,and information fields may be transferred to the Z80-SIO with this mode using the Wait/Ready function. The Z80-SIO transmits the Frame Check sequence using the Transmit Underrun feature.

SDLC Transmit Underrun/End of Message. SDLC-like protocols do not have provisions for fill characters within a message. The Z80-SIO therefore automatically terminates an SDLC frame when the transmit data buffer and output shift register have no more bits to send. It does this by first sending the two bytes of CRC and following these with one or more flags. This technique allows very high-speed transmissions under DMA or CPU block I/O control without requiring the CPU to respond quickly to the end of message situation.

The action that the Z80-SIO takes in the underrun situation depends on the state of the Transmit Underrun/EOM command. Following a reset, the Transmit Underrun/EOM status bit is in the set state and prevents the insertion of CRC characters during the time there is no data to send. Consequently, flag characters are sent. The Z80-SIO begins to send the frame as data is written into the transmit buffer. Between the time the first data byte is written and the end of the message, the Reset Transmit Underrun/EOM command must be issued. Thus the Transmit Underrun/EOM status bit is in the reset state at the end of the message (when underrun occurs), which automatically sends the CRC characters. The sending of the CRC again sets the Transmit/Underrun/EOM status bit.

Although there is no restriction as to when the Transmit Underrun/EOM bit can be reset within a message, it is usually reset after the first data character (secondary address) is sent to

CONTENTS OF WRITE REGISTERS, 3, 4 AND 5 IN SDLC MODES

Table 5.1

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
WR3	00=Rx 5 BITS/CHAR 10=Rx 6 BITS/CHAR 01=Rx 7 BITS/CHAR 11=Rx 8 BITS/CHAR		AUTO ENABLES	ENTER HUNT MODE (IF INCOMING DATA NOT NEEDED)	Rx CRC ENABLE	ADDRESS SEARCH MODE	0	Rx ENABLE
WR4	0	0	1 SELECTS SDLC MODE	0	0	0	0	0
WR5	DTR	00=Tx 5 BITS (OR LESS)/CHAR 10=Tx 6 BITS/CHAR 01=Tx 7 BITS/CHAR 11=Tx 8 BITS/CHAR		0	Tx ENABLE	0 SELECTS SDLC CRC	RTS	Tx CRC ENABLE

the Z80-SIO. Resetting this bit allows CRC and flags to be sent when there is no data to send which gives additional time to the CPU for recognizing the fault and responding with an abort command. By resetting it early in the message, the entire message has the maximum amount of CPU response time in an unintentional transmit underrun situation.

When the External/Status interrupt is set and while CRC is being sent, the Transmit Underrun/EOM bit is set and the Transmit Buffer Empty bit is reset to indicate that the transmit register is full of CRC data. When CRC has been completely sent, the Transmit Buffer Empty status bit it set and an interrupt is generated to indicate to the CPU that another message can begin. This interrupt occurs because CRC has been sent and the flag has been loaded. If no more messages are to be sent, the program can terminate transmission by resetting RTS, and disabling the transmitter.

In the SDLC mode, it is good practice to reset the Transmit Underrun/EOM status bit immediately after the first character is sent to the Z80-SIO. When the Transmit Underrun is detected, this ensures that the transmission time is filled by CRC characters, giving the CPU enough time to issue the Send Abort command. This also stops the flags from going on the line prematurely and eliminates the possibility of the receiver accepting the frame as valid data. The situation can happen because it is possible that—at the receiving end)—the data pattern immediately preceding the automatic flag insertion could match the CRC checker, giving a false CRC check result. The External/Status interrupt is generated whenever the Transmit Underrun/EOM bit is set because of the Transmit Underrun condition.

The transmit underrun logic provides additional protection against premature flag insertion if the proper response is given to the Z80-SIO by the CPU interrupt service routine. The following example is given to clarify this point:

The Z80-SIO raises an interrupt with the Transmit Buffer Empty status bit set.

The CPU does not respond in time and causes a Transmit Underrun condition.

The Z80-SIO starts sending CRC characters (two bytes).

The CPU eventually satisfies the Transmit Buffer Empty interrupt with a data character that follows the CRC character being transmitted.

The Z80-SIO sets the External/Status interrupt with the Transmit Underrun/EOM status bit set.

The CPU recognizes the Transmit Underrun/EOM status and determines from its internal program status that the interrupt is not for "end of message".

The CPU immediately issues a Send Abort Command (WRO) to the Z80-SIO.

The Z80-SIO sends the Abort sequence by destroying whatever data (CRC, data or flag) is being sent.

This sequence illustrates that the CPU has a protection of 22 minimum and 30 maximum transmit clock cycles.

SDLC CRC Generation. The CRC generator must be reset to all 1's at the beginning of each frame before CRC accumulation can begin. Actual accumulation begins when the program sends the address field (eight bits) to the Z80-SIO. Although the Z80-SIO automatically transmits one flag character following the Transmit Enable, it may be wise to send a few more flag characters ahead of the message to ensure character synchronization at the receiving end. This can be done by externally timing out after enabling the transmitter and before loading the first character.

The Transmit CRC Enable (WR5, D_0) should be enabled prior to sending the address field. In the SDLC mode all the characters between the opening and closing flags are included in CRC accumulation, and CRC generated in the Z80-SIO transmitter is inverted before it is sent on the line.

Transmit Termination. If the transmitter is disabled while a character is being sent, that character (data or flag) is sent in the normal fashion, but is followed by a marking line rather than CRC or flag characters.

A character in the buffer when the transmitter is disabled remains in the buffer; however, a programmed Abort sequence is effective as soon as it is written into the control register. Characters being transmitted, if any, are lost. In the case of CRC, the 16-bit transmission is completed if the transmitter is disabled; however, flags are sent in place of CRC.

In all modes, characters are sent with the least-significant bits first. This requires right-hand justification of data to be transmitted if the word length is less than eight bits. If the word length is five bits or less, the special technique described in the Write Register 5 section ("Z80-SIO Programming" chapter; "Write Registers" section) must be used.

Since the number of bits/character can be changed on the fly, the data field can be filled with any number of bits. When used in conjunction with the Receiver Residue codes, the Z80-SIO can receive a message that has a variable I-field and retransmit it exactly as received with no previous information about the character structure of the I-field (if any). A change in the number of bits does not affect the character in the process of being shifted out. Characters are sent with the number of bits programmed at the time that the character is loaded from the transmit buffer to the transmitter.

If the External/Status Interrupt Enable is set, transmitter conditions such as "starting to send CRC characters," "starting to send flag characters," and $\overline{\text{CTS}}$ changing state cause interrupts that have a unique vector if Status Affects Vector is set. All interrupts can be disabled for operation in a polled mode.

Table 5.2 shows the typical program steps that implement the half-duplex SDLC Transmit mode.

SDLC TRANSMIT MODE
Table 5.2

FUNCTION		TYPICAL PROGRAM STEPS	COMMENTS
	REGISTER:	**INFORMATION LOADED:**	
	WRO	CHANNEL RESET	Reset SIO
	WRO	POINTER 2	
	WR2	INTERRUPT VECTOR	Channel B only
	WRO	POINTER 3	
	WR3	AUTO ENABLES	Transmitter sends data only after $\overline{\text{CTS}}$ is detected
	WRO	POINTER 4, RESET EXTERNAL STATUS INTERRUPTS	
	WR4	PARITY INFORMATION, SDLC MODE, x1 CLOCK MODE	
	WRO	POINTER 1, RESET EXTERNAL/STATUS INTERRUPTS	
INITIALIZE	WR1	EXTERNAL INTERRUPT ENABLE, STATUS AFFECTS VECTOR, TRANSMIT INTERRUPT ENABLE OR WAIT/READY MODE ENABLE	The External Interrupt Mode monitors the status of the $\overline{\text{CTS}}$ and $\overline{\text{DCD}}$ inputs, as well as the status of Tx Underrun/EOM latch. Transmit Interrupt interrupts when the Transmit buffer becomes empty; the Wait/Ready mode can be used to transfer data on a DMA or Block Transfer basis. The first interrupt occurs when $\overline{\text{CTS}}$ becomes active, at which point flags are transmitted by the Z80-SIO. The first data byte (address field) can be loaded into the Z80-SIO after this interrupt. Flags cannot be sent to the Z80-SIO as data. Status Affects Vector used in Channel B only.
	WRO	POINTER 5	
	WR5	TRANSMIT CRC ENABLE, REQUEST TO SEND, SDLC-CRC, TRANSMIT ENABLE, TRANSMIT WORD LENGTH, DATA TERMINAL READY	Sync mode must be defined before initializing transmit CRC generator.
	WRO	RESET TRANSMIT CRC GENERATOR	Initialize CRC generator to all 1's.
IDLE MODE		EXECUTE HALT INSTRUCTION OR SOME OTHER PROGRAM	Waiting Interrupt or Wait/Ready output to transfer data.
		WHEN INTERRUPT (WAIT/READY) OCCURS, THE CPU DOES THE FOLLOWING: • CHANGES TRANSMIT WORD LENGTH (IF NECESSARY) • TRANSFERS DATA BYTE FROM CPU (MEMORY) TO SIO • RESETS Tx UNDERRUN EOM LATCH WRO	Flags are transmitted by the SIO as soon as Transmit Enable is set and $\overline{\text{CTS}}$ becomes active. The $\overline{\text{CTS}}$ status change is the first interrupt that occurs and is followed by transmit buffer empty for subsequent transfers.

FUNCTION	TYPICAL PROGRAM STEPS	COMMENTS
	IF LAST CHARACTER OF THE I-FIELD IS SENT, THE SIO DOES THE FOLLOWING: • SENDS CRC • SENDS CLOSING FLAG • INTERRUPTS CPU WITH BUFFER EMPTY STATUS	Word length can be changed on the fly for variable I-field length. The data byte can contain address, control or I-field information (never a flag). It is good practice to reset Tx Underrun/EOM latch in the beginning of the message to avoid a false end-of-frame detection at the receiving end; This ensures that when underrun occurs, CRC is transmitted and underrun interrupt (Tx Underrun/EOM latch active) occurs. Note that "Send Abort" can be issued to the SIO in response to any interrupting continuing to abort the transmission.
DATA TRANSFER AND STATUS MONITORING	CPU DOES THE FOLLOWING: • ISSUES RESET Tx INTERRUPT PENDING COMMAND TO THE Z80-SIO • UPDATES NS COUNT • REPEATS THE PROCESS FOR NEXT MESSAGE, ETC.	
	IF VECTOR INDICATES AN ERROR, THE CPU DOES THE FOLLOWING • SENDS ABORT • EXECUTES ERROR ROUTINE • UPDATES PARAMETERS, MODES, ETC. • RETURNS FROM INTERRUPT	
TERMINATION	REDEFINE INTERRUPT MODES UPDATE MODEM CONTROL OUTPUTS DISABLE TRANSMIT MODE	Terminate gracefully

5.3 SDLC RECEIVE

5.3.1 INITIALIZATION

The SDLC Receive mode is initialized by the system with the following parameters: SDLC mode, x1 clock mode, SDLC polynomial, receive word length, etc. The flag characters must also be loaded in WR7 and the secondary address field loaded in WR6. The receiver is enabled only after all the receive parameters have been set. After all this has been done, the receiver is in the Hunt phase and remains in this phase until the first flag is received. While in the SDLC mode, the receiver never re-enters the Hunt phase, unless specifically instructed to do so by the program. The WR4 parameters must be issued prior to the WR1, WR3, WR5, WR6 and WR7 parameters.

Under program control, the receiver can enter the Address Search mode. If the Address Search bit (WR3, D_2) is set, a character following the flag (first non-flag character) is compared against the programmed address in WR6 and the hardwired global address (11111111). If the SDLC frame address field matches either address, data transfer begins.

Since the Z80-SIO is capable of matching only one address character, extended address field recognition must be done by the CPU. In this case, the Z80-SIO simply transfers the additional address bytes to the CPU as if they were data characters. If the CPU determines that the frame does not have the correct address field, it can set the Hunt bit, and the Z80-SIO suspends reception and searches for a new message headed by a flag. Since the control field of the frame is transparent to the Z80-SIO, it is transferred to the CPU as a data character. Extra zeros inserted in the data stream are automatically deleted; flags are not transferred to the CPU.

5.3.2 DATA TRANSFER AND STATUS MONITORING

After receipt of a valid flag, the assembled characters are transferred to the receive data FIFO. The following four interrupt modes are available to transfer this data and its associated status.

No Interrupts Enabled. This mode is used for purely polled operations or for off-line conditions.

Interrupt On First Character Only. This mode is normally used to start a software polling loop or a Block Transfer instruction using $\overline{\text{WAIT/READY}}$ to synchronize the CPU or DMA device to the incoming data rate. In this mode, the Z80-SIO interrupts on the first character and thereafter only interrupts if Special Receive conditions are detected. The mode is reinitialized with the Enable Interrupt On Next Receive Character Command.

The first character received after this command is issued causes an interrupt. If External/Status interrupts are enabled, they may interrupt any time the $\overline{\text{DCD}}$ input changes state. Special Receive conditions such as End Of Frame and Receiver Overrun also cause interrupts. The End of Frame interrupt can be used to exit the Block Transfer mode.

Interrupt On Every Character. An interrupt is generated whenever the receive FIFO contains a character. Error and Special Receive conditions generate a special vector if Status Affects Vector is selected.

Special Receive Condition Interrupts. The Special Receive Condition interrupt is not, as such, a separate interrupt mode. Before the Special Receive condition can cause an interrupt, either Interrupt On First Receive Character Only or Interrupt On Every Character must be selected. The Special Receive Condition interrupt is caused by a Receive Overrun or End of Frame detection. Since the Receive Overrun status bit is latched, the error status read reflects an error in the current word in the receive buffer in addition to any errors received since the last Error Reset command. The Receive Overrun status bit can only be reset by the Error Reset command. The End Of Frame status bit indicates that a valid ending flag has been received and that the CRC Error and Residue codes are also valid.

Character length may be changed on the fly. If the address and control bytes are processed as 8-bit characters, the receiver may be switched to a shorter character length during the time that the first information character is being assembled. This change must be made fast enough so it is effective before the number of bits specified for the character length have been assembled. For example, if the change is to be from the 8-bit control field to a 7-bit information field, the change must be made before the first seven bits of the I-field are assembled.

SDLC Receive CRC Checking. Control of the receive CRC checker is automatic. It is reset by the leading flag and CRC is calculated up to the final flag. The byte that has the End Of Frame bit set is the byte that contains the result of the CRC check. If the CRC/Framing Error bit is not set, the CRC indicates a valid message. A special check sequence is used for the SDLC check because the transmitted CRC check is inverted. The final check must be 0001110100001111. The 2-byte CRC check characters must be read by the CPU and discarded because the Z80-SIO, while using them for CRC checking, treats them as ordinary data.

SDLC Receive Termination. If enabled, a special vector is generated when the closing flag is received. This signals that the byte with the End Of Frame bit set has been received. In addition to the results of the CRC check, RR1 has three bits of Residue code valid at this time. For those cases in which the number of bits in the I-field is not an integral multiple of the character length used, these bits indicate the boundary between the CRC check bits and the I-field bits. For a detailed description of the meaning of these bits, see the description of the residue codes in RR1 under "Z80-SIO Programming".

Any frame can be prematurely aborted by an Abort sequence. Aborts are detected if seven or more 1's occur and cause an External/Status interrupt (if enabled) with the Break/Abort bit in RR0 set. After the Reset External/Status interrupts command has been issued a second interrupt occurs when the continuous 1's condition has been cleared. This can be used to distinguish between the Abort and Idle line conditions.

Unlike the synchronous mode, CRC calculation in SDLC does not have an 8-bit delay since all the characters are included in CRC calculation. When the second CRC character is loaded into the receive buffer, CRC calculation is complete.

Table 5.2 shows the typical steps required to implement a half-duplex SDLC receive mode. The complete set of command and status bit definitions is found in the next section.

SDLC RECEIVE MODE

Table 5.3

FUNCTION		TYPICAL PROGRAM STEPS	COMMENTS
	REGISTER:	**INFORMATION LOADED**	
	WR0	CHANNEL 2	Reset SIO
	WR0	POINTER 2	
	WR2	INTERRUPT VECTOR	Channel B only
	WR0	POINTER 4	
	WR4	PARITY INFORMATION, SYNC MODE, SDLC MODE, x1 CLOCK MODE	
	WR0	POINTER 5, RESET EXTERNAL/STATUS INTERRUPTS	
	WR5	SDLC-CRC, DATA TERMINAL READY	
	WR0	POINTER 3	
	WR3	RECEIVE CRC ENABLE, ENTER HUNT MODE, AUTO ENABLES RECEIVE CHARACTER LENGTH, ADDRESS SEARCH MODE	"Auto Enables" enables the receiver to accept data only after $\overline{DCD}$ becomes active. Address Search Mode enables SIO to match the message address with the programmed address or the global address.
INITIALIZE	WR0	POINTER 6	
	WR6	SECONDARY ADDRESS FIELD	This address is matched against the message address in an SDLC poll operation.
	WR0	POINTER 7	
	WR7	SDLC FLAG 01111110	This flag detects the start and end of frame in an SDLC operation.
	WR0	POINTER 1, RESET EXTERNAL/STATUS INTERRUPTS	In this interrupt mode, only the Address Field (1 character only) is transferred to CPU. All subsequent fields (Control, Information, etc.) are transferred on a DMA basis. Status Affects Vector in Channel B only.
	WR1	STATUS AFFECTS VECTOR, EXTERNAL INTERRUPT ENABLE, RECEIVE INTERRUPT ON FIRST CHARACTER ONLY.	

FUNCTION	TYPICAL PROGRAM STEPS	COMMENTS
WR0	POINTER 3, ENABLE INTERRUPT ON NEXT RECEIVE CHARACTER	Used to provide simple loop-back entry point for next transaction. WR3 reissued to enable receiver.
WR3	RECEIVE ENABLE, RECEIVE CRC ENABLE, ENTER HUNT MODE, AUTO ENABLE, RECEIVER CHARACTER LENGTH, ADDRESS SEARCH MODE	
IDLE MODE	EXECUTE HALT INSTRUCTION OR SOME OTHER PROGRAM	SDLC Receive Mode is fully initialized and SIO is waiting for the opening flag followed by a matching address field to interrupt the CPU.
	WHEN INTERRUPT ON FIRST CHARACTER OCCURS, THE CPU DOES THE FOLLOWING: • TRANSFERS DATA BYTE (ADDRESS BYTE) TO CPU • DETECTS AND SETS APPROPRIATE FLAG FOR EXTENDED ADDRESS FIELD • UPDATES POINTER AND PARAMETERS • ENABLES DMA CONTROLLER • ENABLES WAIT/READY FUNCTION IN SIO • RETURNS FROM INTERRUPT	During the Hunt phase, the SIO interrupts when the programmed address matches the message address. The CPU establishes the DMA mode and all subsequent data characters are transferred by the DMA controller to memory.
	WHEN THE READY OUTPUT BECOMES ACTIVE, THE DMA CONTROLLER DOES THE FOLLOWING: TRANSFERS THE DATA BYTE TO MEMORY UPDATES THE POINTERS	During the DMA operation, the SIO monitors the $\overline{\text{DCD}}$ input and the Abort sequence in the data stream to interrupt the CPU with External Status error. The Special Receive condition interrupt is caused by Receive Overrun error.
DATA TRANSFER AND STATUS MONITORING	WHEN END OF FRAME INTERRUPT OCCURS, THE CPU DOES THE FOLLOWING: • EXITS DMA MODE (DISABLES WAIT/READY) • TRANSFERS RR1 TO THE CPU • CHECKS THE CRC ERROR BIT STATUS AND RESIDUE CODES • UPDATES NR COUNT • ISSUES "ERROR RESET" COMMAND TO SIO	Detection of End of Frame (Flag) causes interrupt and deactiviates the Wait/Ready function. Residue codes indicate the bit structure of the last two bytes of the message, which were transferred to memory under DMA. "Error Reset" is issued to clear the special condition.
	WHEN ABORT SEQUENCE DETECTED INTERRUPT OCCURS, THE CPU DOES THE FOLLOWING: • TRANSFERS RR0 TO THE CPU • EXITS DMA MODE • ISSUES THE RESET EXTERNAL STATUS INTERRUPT COMMAND TO THE SIO • ENTERS THE IDLE MODE	Abort sequence is detected when seven or more 1's are found in the data stream. CPU is waiting for Abort Sequence to terminate. Termination clears the Break/Abort status bit and causes interrupt.

FUNCTION	TYPICAL PROGRAM STEPS	COMMENTS
	WHEN THE SECOND ABORT SEQUENCE INTERRUPT OCCURS, THE CPU DOES THE FOLLOWING: • ISSUES THE RESET EXTERNAL STATUS INTERRUPT COMMAND TO THE SIO.	At this point, the program proceeds to terminate this message.
TERMINATION	REDFINE INTERRUPT MODES, SYNC MODE AND SDLC MODES DISABLE RECEIVE MODE	

6.0 Z80-SIO PROGRAMMING

6.1 INTRODUCTION

To program the Z80-SIO, the system program first issues a series of commands that initialize the basic mode of operation and then other commands that qualify conditions within the selected mode. For example, the Asynchronous mode, character length, clock rate, number of stop bits, even or odd parity are first set, then the interrupt mode, and finally, receiver or transmitter enable. The WR4 parameters must be issued before any other parameters are issued in the initialization routine.

Both channels contain command registers that must be programmed via the system program prior to operation. The Channel Select input ($B/\overline{A}$) and the Control/Data input ($C/\overline{D}$) are the command structure addressing controls, and are normally controlled by the CPU address bus. Figures 8.1 - 8.4 illustrate the timing relationships for programming the write registers, and transferring data and status.

$C/\overline{D}$	$B/\overline{A}$	Function
0	0	Channel A Data
0	1	Channel B Data
1	0	Channel A Commands/Status
1	1	Channel B Commands/Status

WRITE REGISTERS

The Z80-SIO contains eight registers (WR0-WR7) in each channel that are programmed separately by the system program to configure the functional personality of the channels. With the exception of WR0, programming the write registers requires two bytes. The first byte contains three bits (D_0-D_2) that point to the selected register; the second byte is the actual control word that is written into the register to configure the Z80-SIO.

Note that the programmer has complete freedom, after pointing to the selected register, of either reading to test the read register or writing to initialize the write register. By designing software to initialize the Z80-SIO in a modular and structured fashion, the programmer can use powerful block I/O instructions.

WR0 is a special case in that all the basic commands (CMD_0-CMD_2) can be accessed with a single byte. Reset (internal or external) initializes the pointer bits (D_0-D_2) to point to WR0.

The basic commands (CMD_0-CMD_2) and the CRC controls (CRC_0, CRC_1) are contained in the first byte of any write register access. This maintains maximum flexibility and system control. Each channel contains the following control registers. These registers are addressed as commands (not data).

6.2 WRITE REGISTER 0

WR0 is the command register; however, it is also used for CRC reset codes and to point to the other registers.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CRC Reset Code 1	CRC Reset Code 0	CMD 2	CMD 1	CMD 0	PTR 2	PTR 1	PTR 0

Pointer Bits (D_0-D_2). Bits D_0-D_2 are pointer bits that determine which other write register the next byte is to be written into or which read register the next byte is to be read from. The first byte written into each channel after a reset (either by a Reset command or by the external reset input) goes into WR0. Following a read or write to any register (except WR0), the pointer will point to WR0.

Command Bits (D_3-D_5). Three bits, D_3-D_5, are encoded to issue the seven basic Z80-SIO commands.

COMMAND	CMD_2	CMD_1	CMD_0	
0	0	0	0	Null Command (no effect)
1	0	0	1	Send Abort (SDLC Mode)
2	0	1	0	Reset External/Status Interrupts
3	0	1	1	Channel Reset
4	1	0	0	Enable Interrupt on next Rx Character
5	1	0	1	Reset Transmitter Interrupt Pending
6	1	1	0	Error Reset (latches)
7	1	1	1	Return from Interrupt (Channel A)

Command 0 (Null). The Null command has no effect. Its normal use is to cause the Z80-SIO to do nothing while the pointers are set for the following byte.

Command 1 (Send Abort). This command is used only with the SDLC mode to generate a sequence of eight to thirteen 1's.

Command 2 (Reset External/Status Interrupts). After an External/Status interrupt (a change on a modem line or a break condition, for example), the status bits of RR0 are latched. This command re-enables them and allows interrupts to occur again. Latching the status bits captures short pulses until the CPU has time to read the change.

Command 3 (Channel Reset). This command performs the same function as an External Reset, but only on a single channel. Channel A Reset also resets the interrupt prioritization logic. All control registers for the channel must be rewritten after a Channel Reset command.

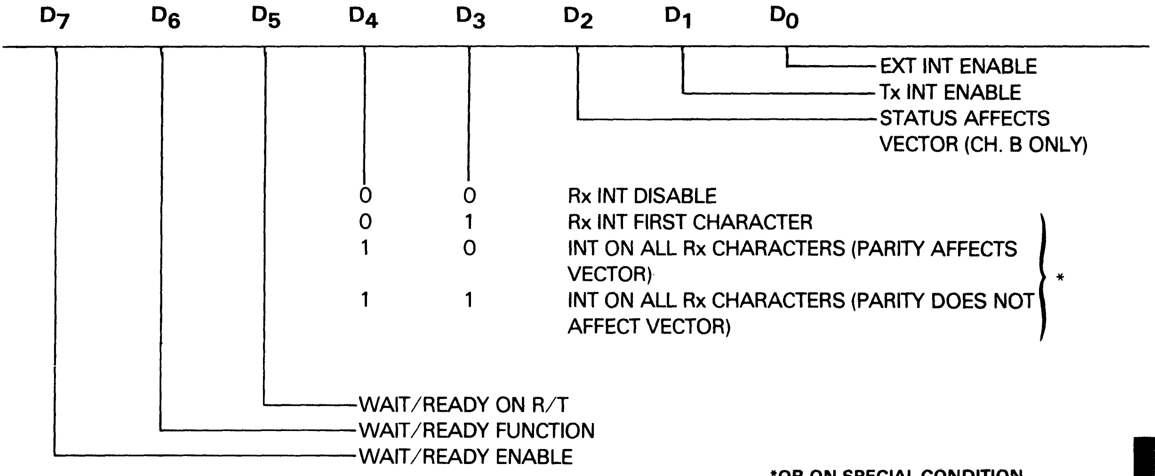
WRITE REGISTER BIT FUNCTIONS

Figure 6.1

WRITE REGISTER 0

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
					0	0	0	REGISTER 0
					0	0	1	REGISTER 1
					0	1	0	REGISTER 2
					0	1	1	REGISTER 3
					1	0	0	REGISTER 4
					1	0	1	REGISTER 5
					1	1	0	REGISTER 6
					1	1	1	REGISTER 7
		0	0	0	NULL CODE			
		0	0	1	SEND ABORT (SDLC)			
		0	1	0	RESET EXT/STATUS INTERRUPTS			
		0	1	1	CHANNEL RESET			
		1	0	0	ENABLE INT ON NEXT Rx CHARACTER			
		1	0	1	RESET Tx INT PENDING			
		1	1	0	ERROR RESET			
		1	1	1	RETURN FROM INT (CH-A ONLY)			
0	0	NULL CODE						
0	1	RESET Rx CRC CHECKER						
1	0	RESET Tx CRC GENERATOR						
1	1	RESET Tx UNDERRUN/EOM LATCH						

WRITE REGISTER 1



D7 D6 D5 D4 D3 D2 D1 D0

D0 — EXT INT ENABLE

D1 — Tx INT ENABLE

D2 — STATUS AFFECTS VECTOR (CH. B ONLY)

D4	D3	
0	0	Rx INT DISABLE
0	1	Rx INT FIRST CHARACTER *
1	0	INT ON ALL Rx CHARACTERS (PARITY AFFECTS VECTOR) *
1	1	INT ON ALL Rx CHARACTERS (PARITY DOES NOT AFFECT VECTOR) *

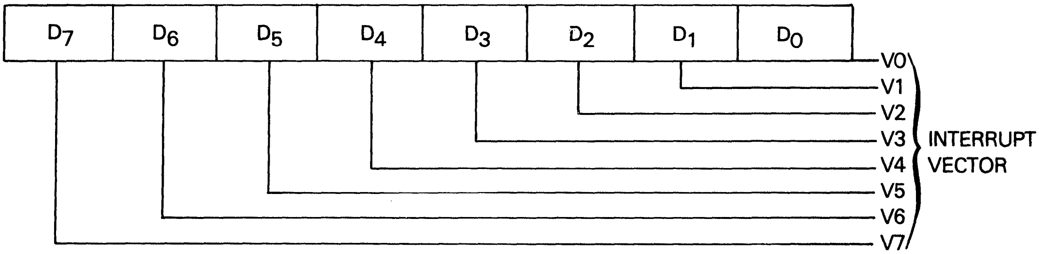
D5 — WAIT/READY ON R/T

D6 — WAIT/READY FUNCTION

D7 — WAIT/READY ENABLE

*OR ON SPECIAL CONDITION

WRITE REGISTER 2 (CHANNEL B ONLY)



D7 D6 D5 D4 D3 D2 D1 D0

D0 — V0

D1 — V1

D2 — V2

D3 — V3

D4 — V4

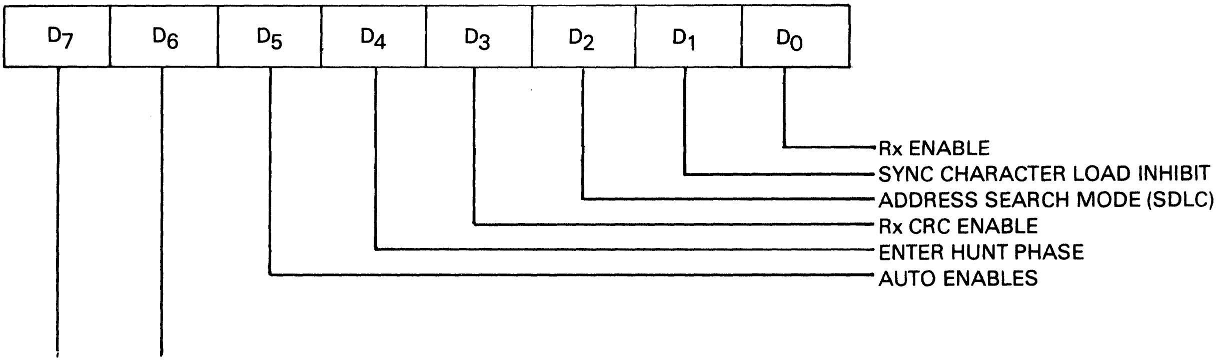
D5 — V5

D6 — V6

D7 — V7

INTERRUPT VECTOR

WRITE REGISTER 3



D7 D6 D5 D4 D3 D2 D1 D0

D0 — Rx ENABLE

D1 — SYNC CHARACTER LOAD INHIBIT

D2 — ADDRESS SEARCH MODE (SDLC)

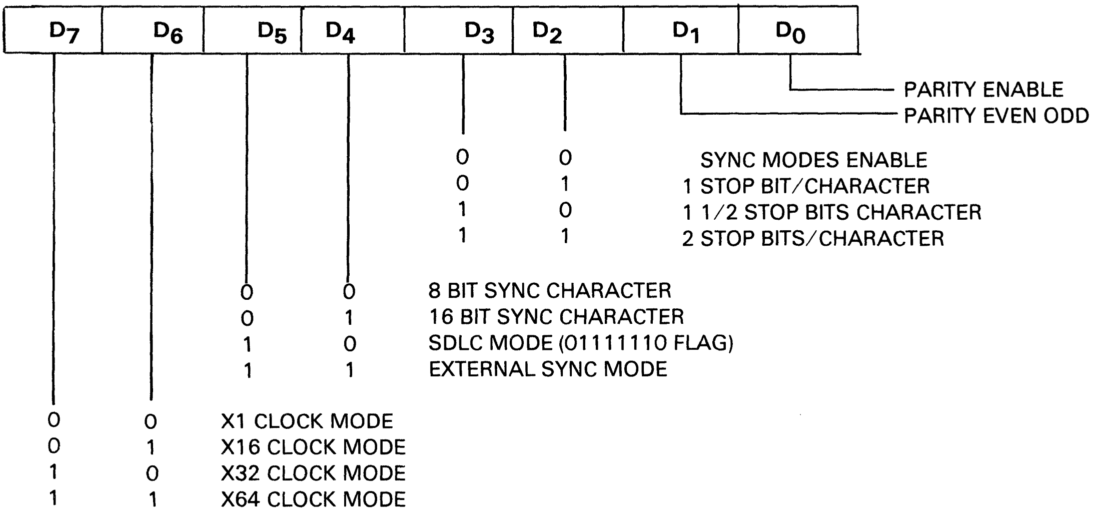
D3 — Rx CRC ENABLE

D4 — ENTER HUNT PHASE

D5 — AUTO ENABLES

D7	D6	
0	0	Rx 5 BITS/CHARACTER
0	1	Rx 7 BITS/CHARACTER
1	0	Rx 6 BITS/CHARACTER
1	1	Rx 8 BITS/CHARACTER

WRITE REGISTER 4

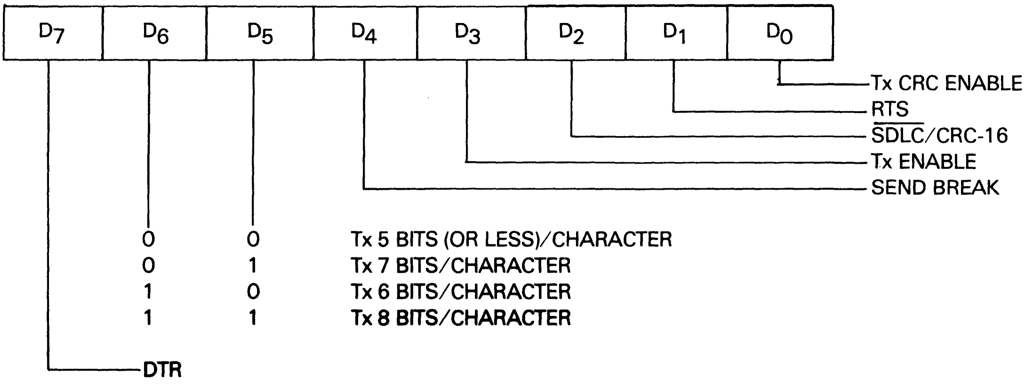


D3	D2	
0	0	SYNC MODES ENABLE
0	1	1 STOP BIT/CHARACTER
1	0	1 1/2 STOP BITS CHARACTER
1	1	2 STOP BITS/CHARACTER

D5	D4	
0	0	8 BIT SYNC CHARACTER
0	1	16 BIT SYNC CHARACTER
1	0	SDLC MODE (01111110 FLAG)
1	1	EXTERNAL SYNC MODE

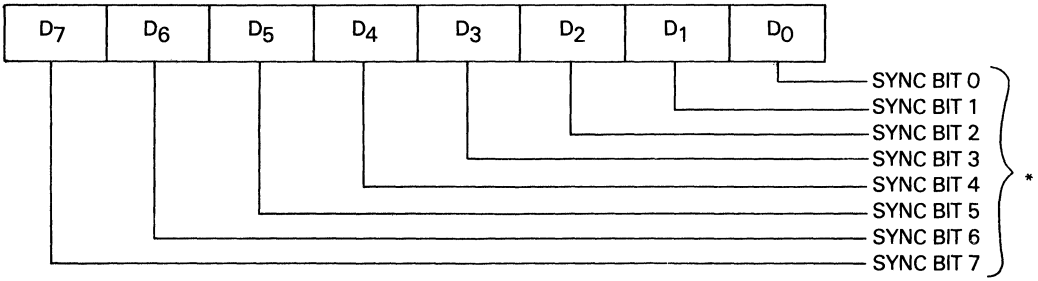
D7	D6	
0	0	X1 CLOCK MODE
0	1	X16 CLOCK MODE
1	0	X32 CLOCK MODE
1	1	X64 CLOCK MODE

WRITE REGISTER 5



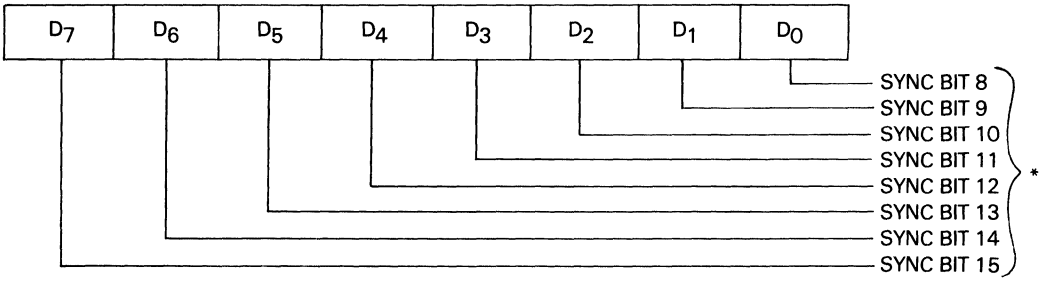
D6	D5	
0	0	Tx 5 BITS (OR LESS)/CHARACTER
0	1	Tx 7 BITS/CHARACTER
1	0	Tx 6 BITS/CHARACTER
1	1	Tx 8 BITS/CHARACTER

WRITE REGISTER 6



*ALSO SDLC ADDRESS FIELD

WRITE REGISTER 7



*FOR SDLC, IT MUST BE PROGRAMMED TO "01111110" FOR FLAG RECOGNITION

After a Channel Reset, four extra system clock cycles should be allowed for Z80-SIO reset time before any additional commands or controls are written into that channel. This can normally be the time used by the CPU to fetch the next op code.

Command 4 (Enable Interrupt On Next Character). If the Interrupt On First Receive Character mode is selected, this command reactivates that mode after each complete message is received to prepare the Z80-SIO for the next message.

Command 5 (Reset Transmitter Interrupt Pending). The transmitter interrupts when the transmit buffer becomes empty if the Transmit Interrupt Enable mode is selected. In those cases where there are no more characters to be sent (at the end of message, for example), issuing this command prevents further transmitter interrupts until after the next character has been loaded into the transmit buffer or until CRC has been completely sent.

Command 6 (Error Reset). This command resets the error latches. Parity and Overrun errors are latched in RR1 until they are reset with this command. With this scheme, parity errors occurring in block transfers can be examined at the end of the block.

Command 7 (Return From Interrupt). This command must be issued in Channel A and is interpreted by the Z80-SIO in exactly the same way it would interpret a RETI command on the data bus. It resets the interrupt-under-service latch of the highest-priority internal device under service and thus allows lower priority devices to interrupt via the daisy chain. This command allows use of the internal daisy chain even in systems with no external daisy chain or RETI command.

CRC Reset Codes 0 and 1 (D_6 and D_7). Together, these bits select one of the three following reset commands:

CRC Reset Code 1	CRC Reset Code 0	
0	0	Null Code (no effect)
0	1	Reset Receive CRC Checker
1	0	Reset Transmit CRC Generator
1	1	Reset Tx Underrun/End Of Message Latch

The Reset Transmit CRC Generator command normally initializes the CRC generator to all 0's. If the SDLC mode is selected, this command initializes the CRC generator to all 1's. The Receive CRC checker is also initialized to all 1's for the SDLC mode.

6.3 WRITE REGISTER 1

WR1 contains the control bits for the various interrupt and Wait/Ready modes.

D_7 Wait/Ready Enable	D_6 Wait Or Ready Function	D_5 Wait/Ready On Receive/Transmit	D_4 Receive Interrupt Mode 1
D_3 Receive Interrupt Mode 0	D_2 Status Affects Vector	D_1 Transmit Interrupt Enable	D_0 External Interrupts Enable

External/Status Interrupt Enable (D_0). The External/Status Interrupt Enable allows interrupts to occur as a result of transitions on the $\overline{DCD}$, $\overline{CTS}$ or $\overline{SYNC}$ inputs, as a result of a Break/Abort detection and termination, or at the beginning of CRC or sync character transmission when the Transmit Underrun/EOM latch becomes set.

Transmitter Interrupt Enable (D_1). If enabled, the interrupts occur whenever the transmitter buffer becomes empty.

Status Affects Vector (D_2). This bit is active in Channel B only. If this bit is not set, the fixed vector programmed in WR2 is returned from an interrupt acknowledge sequence. If this bit is set, the vector returned from an interrupt acknowledge is variable according to the following interrupt conditions:

	V_3	V_2	V_1	
Ch B	0	0	0	Ch B Transmit Buffer Empty
	0	0	1	Ch B External/Status Change
	0	1	0	Ch B Receive Character Available
	0	1	1	Ch B Special Receive Condition*
Ch A	1	0	0	Ch A Transmit Buffer Empty
	1	0	1	Ch A External/Status Change
	1	1	0	Ch A Receive Character Available
	1	1	1	Ch A Special Receive Condition*

*Special Receive Conditions: Parity Error, Rx Overrun Error, Framing Error, End Of Frame (SDLC).

Receive Interrupt Modes 0 and 1 (D_3 and D_4). Together, these two bits specify the various character-available conditions. In Receive Interrupt modes 1, 2 and 3, a Special Receive Condition can cause an interrupt and modify the interrupt vector.

D_4 Receive Interrupt Mode 1	D_3 Receive Interrupt Mode 0	
0	0	0. Receive Interrupts Disabled
0	1	1. Receive Interrupt On First Character Only
1	0	2. Interrupt On All Receive Characters—parity error is a Special Receive condition
1	1	3. Interrupt On All Receive Characters—parity error is not a Special Receive condition

Wait/Ready Function Selection (D_5-D_7). The Wait and Ready functions are selected by controlling D_5, D_6 and D_7. Wait/Ready function is enabled by setting Wait/Ready Enable (WR1, D_7) to 1. The Ready Function is selected by setting D_6 (Wait/Ready function) to 1. If this bit is 1, the $\overline{\text{WAIT/READY}}$ output switches from High to Low when the Z80-SIO is ready to transfer data. The Wait function is selected by setting D_6 to 0. If this bit is 0, the $\overline{\text{WAIT/READY}}$ output is in the open-drain state and goes Low when active.

Both the Wait and Ready functions can be used in either the Transmit or Receive modes, but not both simultaneously. If D_5 (Wait/Ready or Receive/Transmit) is set to 1, the Wait/Ready function responds to the condition of the receive buffer (empty or full). If D_5 is set to 0, the Wait/Ready function responds to the condition of the transmit buffer (empty or full).

The logic states of the $\overline{\text{WAIT/READY}}$ output when active or inactive depend on the combination of modes selected. Following is a summary of these combinations:

If $D_7 = 0$	
And $D_6 = 1$	And $D_6 = 0$
$\overline{\text{READY}}$ is High	$\overline{\text{WAIT}}$ is floating

If $D_7 = 1$			
	And $D_5 = 0$		And $D_5 = 1$
$\overline{\text{READY}}$	Is High when transmit buffer is full.	$\overline{\text{READY}}$	Is High when receive buffer is empty.
$\overline{\text{WAIT}}$	Is Low when transmit buffer is full and an SIO data port is selected.	$\overline{\text{WAIT}}$	Is Low when receive buffer is empty and an SIO data port is selected.
$\overline{\text{READY}}$	Is Low when transmit buffer is empty.	$\overline{\text{READY}}$	Is Low when receive buffer is full.
$\overline{\text{WAIT}}$	Is floating when transmit buffer is empty.	$\overline{\text{WAIT}}$	Is Floating when receive buffer is full.

The $\overline{\text{WAIT}}$ output High-to-Low transition occurs when the delay time t_DIC(WR) after the I/O request. The Low-to-High transition occurs with the delay t_DHΦ(WR) from the falling edge of Φ. The $\overline{\text{READY}}$ output High-to-Low transition occurs with the delay t_DLΦ(WR) from the rising edge of Φ. The $\overline{\text{READY}}$ output Low-to-High transition occurs with the delay t_DIC(WR) after $\overline{\text{IORQ}}$ falls.

The Ready function can occur any time the Z80-SIO is not selected. When the $\overline{\text{READY}}$ output becomes active (Low), the DMA controller issues $\overline{\text{IORQ}}$ and the corresponding B/$\overline{\text{A}}$ and C/$\overline{\text{D}}$ inputs to the Z80-SIO to transfer data. The $\overline{\text{READY}}$ output becomes inactive as soon as $\overline{\text{IORQ}}$ and $\overline{\text{CS}}$ become active. Since the Ready function can occur internally in the Z80-SIO whether it is addressed or not, the $\overline{\text{READY}}$ output becomes inactive when any CPU data or command transfer takes place. This does not cause problems because the DMA controller is not enabled when the CPU transfer takes place.

The Wait function—on the other hand—is active only if the CPU attempts to read Z80-SIO data that has not yet been received, which occurs frequently when block transfer instructions are used. The Wait function can also become active (under program control) if the CPU tries to write data while the transmit buffer is still full. The fact that the $\overline{\text{WAIT}}$ output for either channel can become active when the opposite channel is addressed (because the Z80-SIO is addressed) does not affect operation of software loops or block move instructions.

6.4 WRITE REGISTER 2

WR2 is the interrupt vector register; it exists in Channel B only. V_4-V_7 and V_0 are always returned exactly as written; V_1-V_3 are returned as written if the Status Affects Vector (WR1, D_2) control bit is 0. If this bit is 1, they are modified as explained in the previous section.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
V_7	V_6	V_5	V_4	V_3	V_2	V_1	V_0

6.5 WRITE REGISTER 3

WR3 contains receiver logic control bits and parameters.

D_7	D_6	D_5	D_4
Receiver Bits/ Char 1	Receiver Bits/ Char 0	Auto Enables	Enter Hunt Phase

D_3	D_2	D_1	D_0
Receiver CRC Enable	Address Search Mode	Sync Char Load Inhibit	Receiver Enable

Receiver Enable (D_0). A 1 programmed into this bit allows receive operations to begin. This bit should be set only after all other receive parameters are set and receiver is completely initialized.

Sync Character Load Inhibit (D_1). Sync characters preceding the message (leading sync characters) are not loaded into the receive buffers if this option is selected. Because CRC calculations are not stopped by sync character stripping, this feature should be enabled only at the beginning of the message.

Address Search Mode (D_2). If SDLC is selected, setting this mode causes messages with addresses not matching the programmed address in WR6 or the global (11111111) address to be rejected. In other words, no receive interrupts can occur in the Address Search mode unless there is an address match.

Receiver CRC Enable (D_3). If this bit is set, CRC calculation starts (or restarts) at the beginning of the last character transferred from the receive shift register to the buffer stack, regardless of the number of characters in the stack. See "SDLC Receive CRC Checking" (SDLC Receive section) and "CRC Error Checking" (Synchronous Receive section) for details regarding when this bit should be set.

Enter Hunt Phase (D_4). The Z80-SIO automatically enters the Hunt phase after a reset; however, it can be re-entered if character synchronization is lost for any reason (Synchronous mode) or if the contents of an incoming message are not needed (SDLC mode). The Hunt phase is re-entered by writing a 1 into bit D_4. This sets the Sync/Hunt bit (D_4) in RR0.

Auto Enables (D_5). If this mode is selected, $\overline{DCD}$ and $\overline{CTS}$ become the receiver and transmitter enables, respectively. If this bit is not set, $\overline{DCD}$ and $\overline{CTS}$ are simply inputs to their corresponding status bits in RR0.

Receiver Bits/Character 1 and 0 (D_7 and D_6). Together, these bits determine the number of serial receive bits assembled to form a character. Both bits may be changed during the time that a character is being assembled, but they must be changed before the number of bits currently programmed is reached.

D_7	D_6	Bits/Character
0	0	5
0	1	7
1	0	6
1	1	8

6.6 WRITE REGISTER 4

WR4 contains the control bits that affect both the receiver and transmitter. In the transmit and receive initialization routine, these bits should be set before issuing WR1, WR3, WR5, WR6, and WR7.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Clock Rate	Clock Rate	Sync Modes	Sync Modes	Stop Bits	Stop Bits	Parity Even/$\overline{\text{Odd}}$	Parity
1	0	1	0	1	0		

Parity (D_0). If this bit is set, an additional bit position (in addition to those specified in the bits/character control) is added to transmitted data and is expected in receive data. In the Receive mode, the parity bit received is transferred to the CPU as part of the character, unless 8 bits/character is selected.

Parity Even $\overline{\text{Odd}}$ (D_1). If parity is specified, this bit determines whether it is sent and checked as even or odd (1=even).

Stop Bits 0 and 1 (D_2 and D_3). These bits determine the number of stop bits added to each asynchronous character sent. The receiver always checks for one stop bit. A special mode (00) signifies that a synchronous mode is to be selected.

D_3 Stop Bits 1	D_2 Stop Bits 0	
0	0	Sync modes
0	1	1 stop bit per character
1	0	1½ stop bits per character
1	1	2 stop bits per character

Sync Modes 0 and 1 (D_4 and D_5). These bits select the various options for character synchronization.

Sync Mode 1	Sync Mode 0	
0	0	8-bit programmed sync
0	1	16-bit programmed sync
1	0	SDLC mode (01111110 flag pattern)
1	1	External Sync mode

Clock Rate 0 and 1 (D_6 and D_7). These bits specify the multiplier between the clock ($\overline{\text{TxC}}$ and $\overline{\text{RxC}}$) and data rates. For synchronous modes, the x1 clock rate must be specified. Any rate may be specified for asynchronous modes; however, the same rate must be used for both the receiver and transmitter. The system clock in all modes must be at least 5 times the data rate. If the x1 clock rate is selected, bit synchronization must be accomplished externally.

Clock Rate 1	Clock Rate 0	
0	0	Data Rate x1=Clock Rate
0	1	Data Rate x16=Clock Rate
1	0	Data Rate x32=Clock Rate
1	1	Data Rate x64=Clock Rate

6.7 WRITE REGISTER 5

WR5 contains control bits that affect the operation of transmitter, with the exception of D2, which affects the transmitter and receiver.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
DTR	Tx Bits/ Char 1	Tx Bits/ Char 0	Send Break	Tx Enable	CRC-16/ SDLC	RTS	Tx CRC Enable

Transmit CRC Enable (D_0). This bit determines if CRC is calculated on a particular transmit character. If it is set at the time the character is loaded from the transmit buffer into the transmit shift register, CRC is calculated on the character. CRC is not automatically sent unless this bit is set when the Transmit Underrun condition exists.

Request To Send (D_1). This is the control bit for the $\overline{RTS}$ pin. When the $\overline{RTS}$ bit is set, the $\overline{RTS}$ pin goes Low; when reset, $\overline{RTS}$ goes High. In the Asynchronous mode, $\overline{RTS}$ goes High only after all the bits of the character are transmitted and the transmitter buffer is empty. In Synchronous modes, the pin directly follows the state of the bit.

CRC-16/$\overline{\text{SDLC}}$ (D_2). This bit selects the CRC polynomial used by both the transmitter and receiver. When set, the CRC-16 polynomial ($X^{16} + X^{15} + X^2 + 1$) is used; when reset, the SDLC polynomial ($X^{16} + X^{12} + X^5 + 1$) is used. If the SDLC mode is selected, the CRC generator and checker are preset to all 1's and a special check sequence is used. The SDLC CRC polynomial must be selected when the SDLC mode is selected. If the SDLC mode is not selected, the CRC generator and checker are present to all 0's (for both polynomials).

Transmit Enable (D_3). Data is not transmitted until this bit is set and the Transmit Data output is held marking. Data or sync characters in the process of being transmitted are completely sent if this bit is reset after transmission has started. If the transmitter is disabled during the transmission of a CRC character, sync or flag characters are sent instead of CRC.

Send Break (D_4). When set, this bit immediately forces the Transmit Data output to the spacing condition, regardless of any data being transmitted. When reset, TxD returns to marking.

Transmit Bits/Character 0 and 1 (D_5 and D_6). Together, D_6 and D_5 control the number of bits in each byte transferred to the transmit buffer.

D_6 Transmit Bits/ Character 1	D_5 Transmit Bits/ Character 0	Bits/Character
0	0	Five or less
0	1	7
1	0	6
1	1	8

Bits to be sent must be right justified, least-significant bits first. The Five Or Less mode allows transmission of one to five bits per character; however, the CPU should format the data character as shown in the following table.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
1	1	1	1	0	0	0	D	Sends one data bit
1	1	1	0	0	0	D	D	Sends two data bits
1	1	0	0	0	D	D	D	Sends three data bits
1	0	0	0	D	D	D	D	Sends four data bits
0	0	0	D	D	D	D	D	Sends five data bits

Data Terminal Ready (D_7). This is the control bit for the $\overline{DTR}$ pin. When set, $\overline{DTR}$ is active (Low); when reset, $\overline{DTR}$ is inactive (High).

6.8 WRITE REGISTER 6

This register is programmed to contain the transmit sync character in the Monosync mode, the first eight bits of a 16-bit sync character in the Bisync mode or a transmit sync character in the External Sync mode. In the SDLC mode, it is programmed to contain the secondary address field used to compare against the address field of the SDLC frame.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Sync 7	Sync 6	Sync 5	Sync 4	Sync 3	Sync 2	Sync 1	Sync 0

6.9 WRITE REGISTER 7

This register is programmed to contain the receive sync character in the Monosync mode, a second byte (last eight bits) of a 16-bit sync character in the Bisync mode and a flag character (01111110) in the SDLC mode. WR7 is not used in the External Sync mode.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Sync 15	Sync 14	Sync 13	Sync 12	Sync 11	Sync 10	Sync 9	Sync 8

7.0 READ REGISTERS

7.1 INTRODUCTION

The Z80-SIO contains three registers, RR0-RR2 (Figure 7.1), that can be read to obtain the status information for each channel (except for RR2-Channel B only). The status information includes error conditions, interrupt vector and standard communications-interface signals.

To read the contents of a selected read register other than RR0, the system program must first write the pointer byte to WR0 in exactly the same way as a write register operation. Then, by executing an input instruction, the contents of the addressed read register can be read by the CPU.

The status bits of RR0 and RR1 are carefully grouped to simplify status monitoring. For example, when the interrupt vector indicates that a Special Receive Condition interrupt has occurred, all the appropriate error bits can be read from a single register (RR1).

7.2 READ REGISTER 0

This register contains the status of the receive and transmit buffers, the $\overline{\text{DCD}}$, $\overline{\text{CTS}}$ and $\overline{\text{SYNC}}$ inputs, the Transmit Underrun/EOM latch; and the Break/Abort latch.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Break	Transmit	$\overline{\text{CTS}}$	Sync/	DCD	Transmit	Interrupt	Receive
Abort	Underrun/		Hunt		Buffer Empty	Pending	Character
	EOM					(Ch. A only)	Available

Receive Character Available (D_0). This bit is set when at least one character is available in the receive buffer; it is reset when the receive FIFO is completely empty.

Interrupt Pending (D_1). Any interrupting condition in the Z80-SIO causes this bit to be set; however, it is readable only in Channel A. This bit is mainly used in applications that do not have vectored interrupts available. During the interrupt service routine in these applications, this bit indicates if any interrupt conditions are present in all Z80-SIO. This eliminates the need for analyzing all the bits of RR0 in both Channels A and B. Bit D_1 is reset when all the interrupting conditions are satisfied. This bit is always 0 in Channel B.

Transmit Buffer Empty (D_2). This bit is set whenever the transmit buffer becomes empty, except when a CRC character is being sent in a synchronous or SDLC mode. The bit is reset when a character is loaded into the transmit buffer. This bit is in the set condition after a reset.

Data Carrier Detect (D_3). The DCD bit shows the inverted state of the $\overline{\text{DCD}}$ input at the time of the last change of any of the five External/Status bits (DCD, $\overline{\text{CTS}}$, Sync/Hunt, Break/Abort or Transmit Underrun/EOM). Any transition of the $\overline{\text{DCD}}$ input causes the DCD bit to be latched and causes an External/Status interrupt. To read the current state of the DCD bit, this bit must be read immediately following a Reset External/Status Interrupt command.

Sync/Hunt (D_4). Since this bit is controlled differently in the Asynchronous, Synchronous and SDLC modes, its operation is somewhat more complex than that of the other bits and, therefore, requires more explanation.

In Asynchronous modes, the operation of this bit is similar to the DCD status bit, except that Sync/Hunt shows the state of the $\overline{\text{SYNC}}$ input. Any High-to-Low transition on the $\overline{\text{SYNC}}$ pin sets this bit and causes an External/Status interrupt (if enabled). The Reset External/Status Interrupt command is issued to clear the interrupt. A Low-to-High transition clears this bit and sets the External/Status interrupt. When the External/Status interrupt is set by the change in state of any other input or condition, this bit shows the inverted state of $\overline{\text{SYNC}}$ pin at the time of the change. This bit must be read immediately following a Reset External/Status Interrupt command to read the current state of the $\overline{\text{SYNC}}$ input.

In the External Sync mode, the Sync/Hunt bit operates in a fashion similar to the Asynchronous mode, except the Enter Hunt Mode control bit enables the external sync detection logic. When the External

Sync Mode and Enter Hunt Mode bits are set (for example, when the receiver is enabled following a reset), the $\overline{\text{SYNC}}$ input must be held High by the external logic until external character synchronization is achieved. A High at the $\overline{\text{SYNC}}$ input holds the Sync/Hunt status bit in the reset condition.

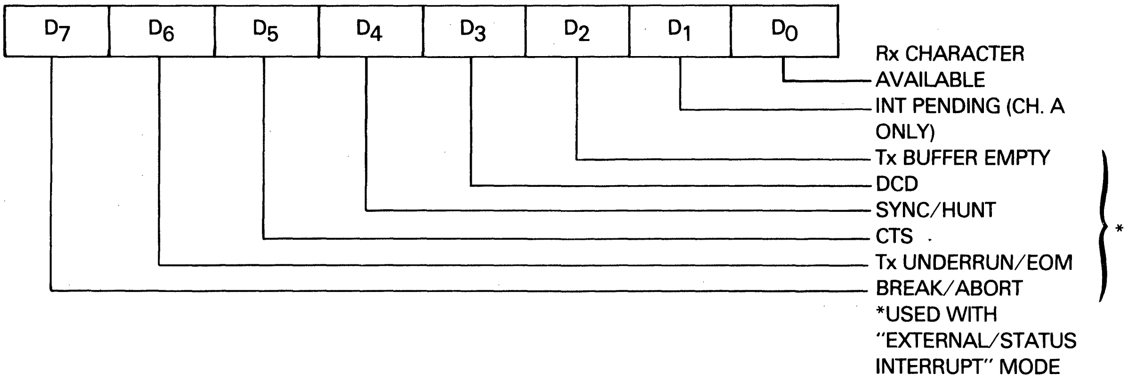
When external synchronization is achieved, $\overline{\text{SYNC}}$ must be driven Low on the second rising edge or $\overline{\text{RxC}}$ on which the last bit of the sync character was received. In other words, after the sync pattern is detected, the external logic must wait for two full Receive clock cycles to activate the $\overline{\text{SYNC}}$ input. Once $\overline{\text{SYNC}}$ is forced Low, it is a good practice to keep it Low until the CPU informs the external sync logic that synchronization has been lost or a new message is about to start. Refer to Figure 18 for timing details. The High-to-Low transition of the $\overline{\text{SYNC}}$ input sets the Sync/Hunt bit, which—in turn—sets the External/Status interrupt. The CPU must clear the interrupt by issuing the Reset External/Status Interrupt command.

When the $\overline{\text{SYNC}}$ input goes High again, another External/Status interrupt is generated that must also be cleared. The Enter Hunt Mode control bit is set whenever character synchronization is lost or the end of message is detected. In this case, the Z80-SIO again looks for a High-to-Low transition on the $\overline{\text{SYNC}}$ input and the operation repeats as explained previously. This implies the CPU should also inform the external logic that character synchronization has been lost and that the Z80-SIO is waiting for $\overline{\text{SYNC}}$ to become active.

READ REGISTER BIT FUNCTIONS

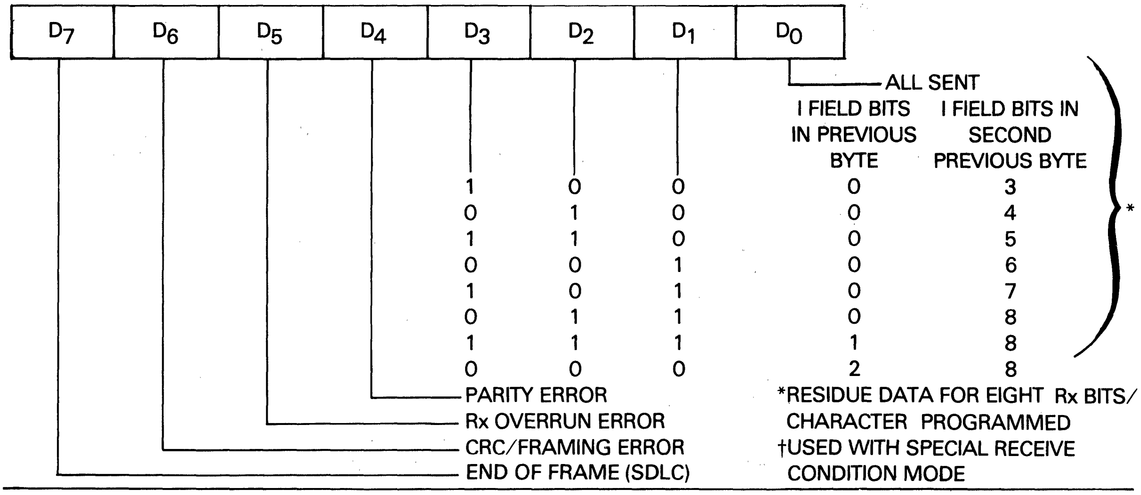
Figure 7.1

READ REGISTER 0



*USED WITH "EXTERNAL/STATUS INTERRUPT" MODE

READ REGISTER 1†

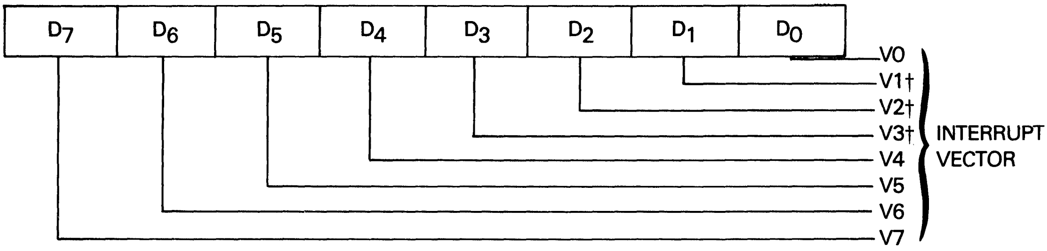


D3	D2	D1	I FIELD BITS IN PREVIOUS BYTE	I FIELD BITS IN SECOND PREVIOUS BYTE
1	0	0	0	3
0	1	0	0	4
1	1	0	0	5
0	0	1	0	6
1	0	1	0	7
0	1	1	0	8
1	1	1	1	8
0	0	0	2	8

*RESIDUE DATA FOR EIGHT Rx BITS/CHARACTER PROGRAMMED

†USED WITH SPECIAL RECEIVE CONDITION MODE

READ REGISTER 2



†VARIABLE IF "STATUS AFFECTS VECTOR" IS PROGRAMMED

In the Monosync and Bisync Receive modes, the Sync/Hunt status bit is initially set to 1 by the Enter Hunt Mode bit. The Sync/Hunt bit is reset when the Z80-SIO establishes character synchronization. The High-to-Low transition of the Sync/Hunt bit causes an External/Status interrupt that must be cleared by the CPU issuing the Reset External/Status Interrupt command. This enables the Z80-SIO to detect the next transition of other External/Status bits.

When the CPU detects the end of message of that character synchronization is lost, it sets the Enter Hunt Mode control bit, which—in turn—sets the Sync/Hunt bit to 1. The Low-to-High transition of the Sync/Hunt bit sets the External/Status interrupt, which must also be cleared by the Reset External/Status Interrupt command. Note that the $\overline{\text{SYNC}}$ pin acts as an output in this mode and goes Low every time a sync pattern is detected in the data stream.

In the SDLC mode, the Sync/Hunt bit is initially set by the Enter Hunt mode bit or when the receiver is disabled. In any case, it is reset to 0 when the opening flag of the first frame is detected by the Z80-SIO. The External/Status interrupt is also generated and should be handled as discussed previously.

Unlike the Monosync and Bisync modes, once the Sync/Hunt bit is reset in the SDLC mode, it does not need to be set when the end of message is detected. The Z80-SIO automatically maintains synchronization. The only way the Sync/Hunt bit can be set again is by the Enter Hunt Mode bit or by disabling the receiver.

Clear to Send (D_5). This bit is similar to the DCD bit, except that it shows the inverted state of the $\overline{\text{CTS}}$ pin.

Transmit Underrun/End of Message (D_6). This bit is in a set condition following a reset (internal or external). The only command that can reset this bit is the Reset Transmit Underrun/EOM Latch command (WR0, D_6 and D_7). When the Transmit Underrun condition occurs, this bit is set; its becoming set causes the External/Status interrupt, which must be reset by issuing the Reset External/Status Interrupt command bits (WR0). This status bit plays an important role in conjunction with other control bits in controlling a transmit operation. Refer to "Bisync Transmit Underrun" and "SDLC Transmit Underrun" for additional details.

Break/Abort (D_7). In the Asynchronous Receive mode, this bit is set when a Break sequence (null character plus framing error) is detected in the data stream. The External/Status interrupt, if enabled, is set when Break is detected. The interrupt service routine must issue the Reset External/Status Interrupt command (WR0, CMD_2) to the break detection logic so the Break sequence termination can be recognized.

The Break/Abort bit is reset when the termination of the Break sequence is detected in the incoming data stream. The termination of the Break sequence also causes the External/Status interrupt to be set. The Reset External/Status Interrupt command must be issued to enable the break detection logic to look for the next Break sequence. A single extraneous null character is present in the receiver after the termination of a break; it should be read and discarded.

In the SDLC Receive mode, this status bit is set by the detection of an Abort sequence (seven or more 1's). The External/Status Interrupt is handled the same way as in the case of a Break. The Break/Abort bit is not used in the Synchronous Receive mode.

7.3 READ REGISTER 1

This register contains the Special Receive condition status bits and Residue codes for the I-field in the SDLC Receive Mode.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
End of Frame (SDLC)	CRC/ Framing Error	Receiver Overrun Error	Parity Error	Residue Code 2	Residue Code 1	Residue Code 0	All Sent

All Sent (D_0). In Asynchronous modes, this bit is set when all the characters have completely cleared the transmitter. Transitions of this bit do not cause interrupts. It is always set in Synchronous modes.

Residue Codes 0, 1 ,and 2 (D_1-D_3). In those cases of the SDLC receive mode where the I-field is not an integral multiple of the character length, these three bits indicate the length of the I-field. These codes are meaningful only for the transfer in which the End Of Frame bit is set (SDLC). For a receive character length of eight bits per character, the codes signify the following:

Residue Code 2	Residue Code 1	Residue Code 0	I-Field Bits In Previous Byte	I-Field Bits In Second Previous Byte
1	0	0	0	3
0	1	0	0	4
1	1	0	0	5
0	0	1	0	6
1	0	1	0	7
0	1	1	0	8
1	1	1	1	8
0	0	0	2	8

I-Field bits are right-justified in all cases

If a receive character length different from eight bits is used for the I-field, a table similar to the previous one may be constructed for each different character length. For no residue (that is, the last character boundary coincides with the boundary of the I-field and CRC field), the Residue codes are:

Bits per Character	Residue Code 2	Residue Code 1	Residue Code 0
8 Bits per Character	0	1	1
7 Bits per Character	0	0	0
6 Bits per Character	0	1	0
5 Bits per Character	0	0	1

Parity Error (D_4). When parity is enabled, this bit is set for those characters whose parity does not match the programmed sense (even/odd). The bit is latched, so once an error occurs, it remains set until the Error Reset command (WR0) is given.

Receive Overrun Error (D_5). This bit indicates that more than three characters have been received without a read from the CPU. Only the character that has been written over is flagged with this error, but when this character is read, the error condition is latched until reset by the Error Reset command. If Status Affects Vector is enabled, the character that has been overrun interrupts with a Special Receive Condition vector.

CRC/Framing Error (D_6). If a Framing Error occurs (asynchronous modes), this bit is set (and not latched) for the receive character in which the Framing error occurred. Detection of a Framing Error adds an additional one-half of a bit time to the character time so the Framing Error is not interpreted as a new start bit. In Synchronous and SDLC modes, this bit indicates the result of comparing the CRC checker to the appropriate check value. This bit is reset by issuing an Error Reset command. The bit is not latched, so it is always updated when the next character is received. When used for CRC error and status in Synchronous modes, it is usually set since most bit combinations result in a non-zero CRC, except for a correctly completed message.

End of Frame (D_7). This bit is used only with the SDLC mode and indicates that a valid ending flag has been received and that the CRC Error and Residue codes are also valid. This bit can be reset by issuing the Error Reset command. It is also updated by the first character of the following frame.

7.4 READ REGISTER 2 (Ch. B Only)

This register contains the interrupt vector written into WR2 if the Status Affects Vector control bit is not set. If the control bit is set, it contains the modified vector shown in the Status Affects Vector paragraph of the Write Register 1 section. When this register is read, the vector returned is modified by the highest priority interrupting condition at the time of the read. If no interrupts are pending, the vector is modified with V_3=0, V_2=1, and V_1=1. This register may be read only through Channel B.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
V_7	V_6	V_5	V_4	V_3	V_2	V_1	V_0

$\underbrace{V_3\ V_2\ V_1}_{}$

Variable if Status
Affects Vector is
enabled

7.5 APPLICATIONS

The flexibility and versatility of the Z80-SIO make it useful for numerous applications, a few of which are included here. These examples show several applications that combine the Z80-SIO with other members of the Z80 family.

Figure 7.2 shows the simple processor-to-processor communication over a direct line. Both remote processors in this system can communicate to the Z80-CPU with different protocols and data rates. Depending on the complexity of the application, other Z80 peripheral circuits (Z80-CTC, for example) may be required. The unused channel of the Z80-SIO can be used to control other peripherals, or they can be connected to other remote processors.

Figure 7.3 illustrates how both channels of a single Z80-SIO are used with modems that have primary and secondary or reverse channel options. Alternatively, two modems without these options can be connected to the Z80-SIO. A suitable baud-rate generator (Z80-CTC) must be used for Asynchronous modems.

Figure 7.4 shows the Z80-SIO in a data concentrator, a relatively complex application that uses two Z80-SIOs to perform a variety of functions. The data concentrator can be used to collect data from many terminals over low-speed lines and transmit it over a single high-speed line after editing and reformatting.

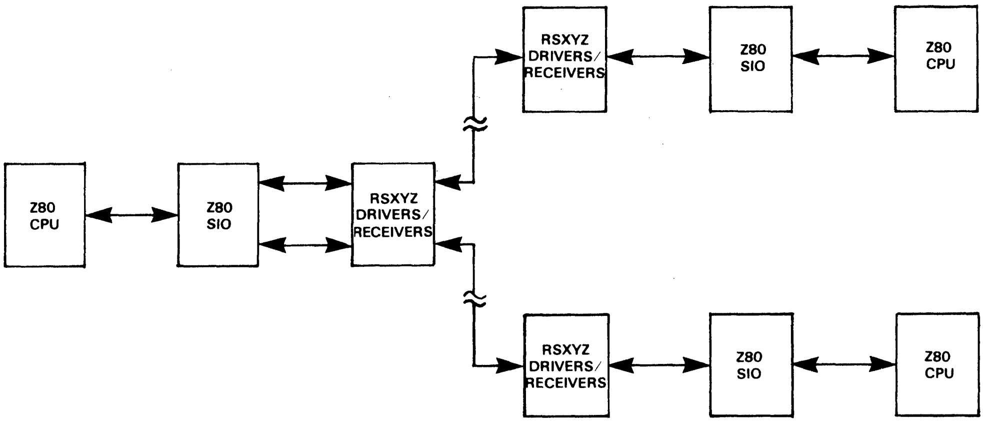
The Z80-DMA controller circuit is used with Z80-SIO #2 to transmit the reformatted data at high speed with the required protocol. The high-speed modem provides the transmit clock for this channel. The Z80-CTC counter-timer circuit supplies the transmit and receive clocks for the low-speed lines and is also used as a time-out counter for various functions.

The Z80-SIO #1 controls local or remote terminals. A single intelligent terminal is shown within the dashed lines. The terminal employs a Z80-SIO to communicate to the data concentrator on one channel while providing the interface to a line printer over its second channel. The intelligent terminal shown could be designed to operate interactively with the operator.

Depending on the software and hardware capabilities built into this system, the data concentrator can employ store-and-forward or hold-and-forward methods for regulating information traffic between slow terminals and the high-speed remote processor. If the high-speed channel is provided with a dial-out option, the channel can be connected to a number of remote processors over a switched line.

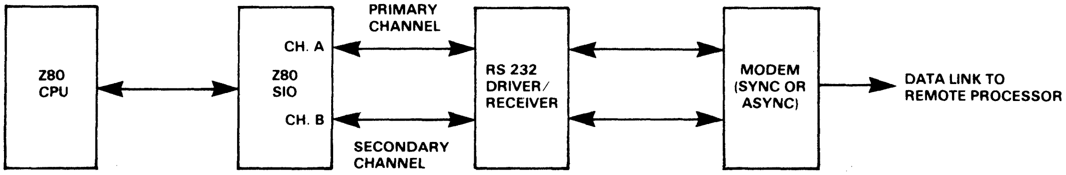
SYNCHRONOUS/ASYNCHRONOUS PROCESSOR-TO-PROCESSOR COMMUNICATION (USING TELEPHONE LINE)

Figure 7.2



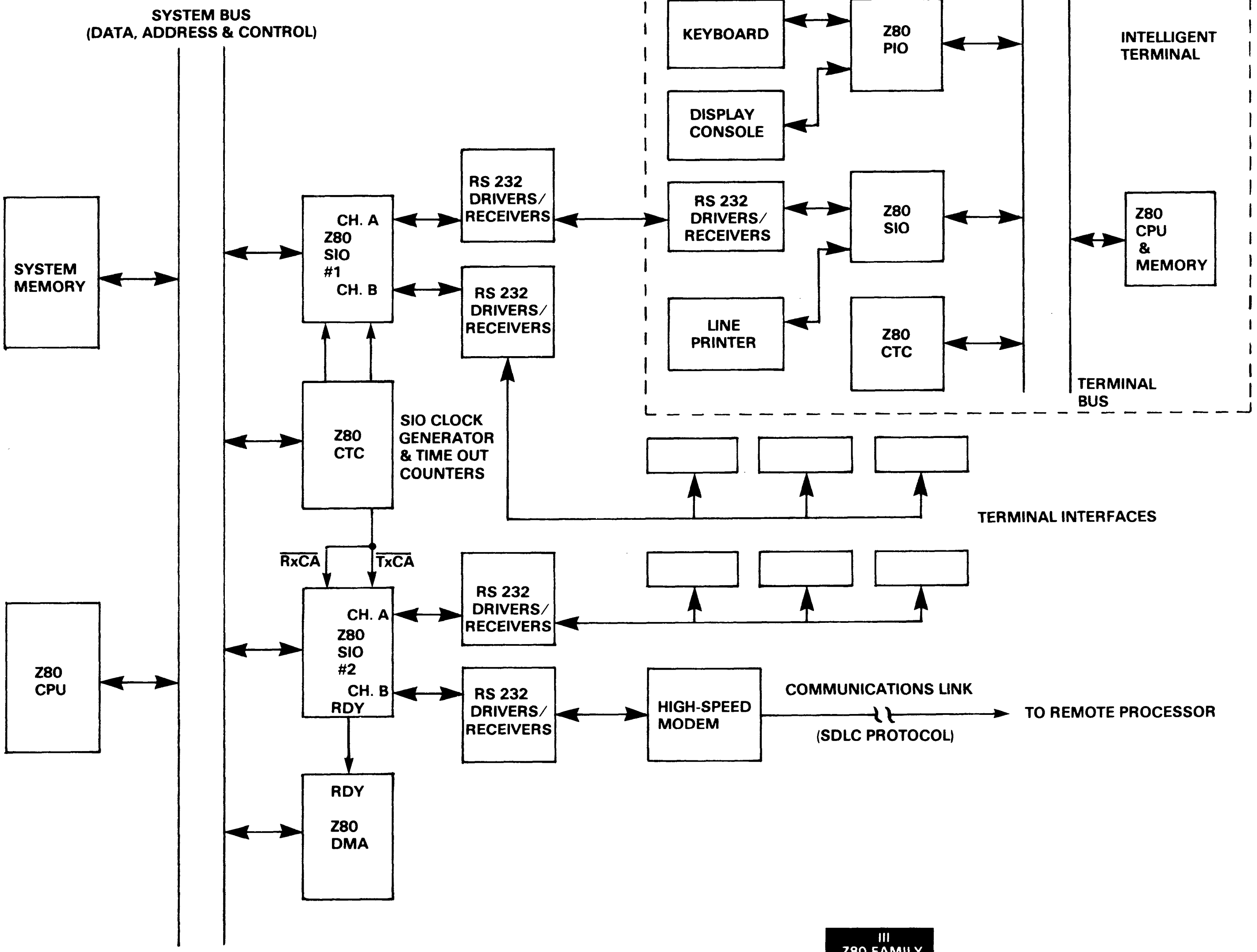
BOTH CHANNELS OF A SINGLE Z80-SIO

Figure 7.3



DATA CONCENTRATOR

Figure 7.4

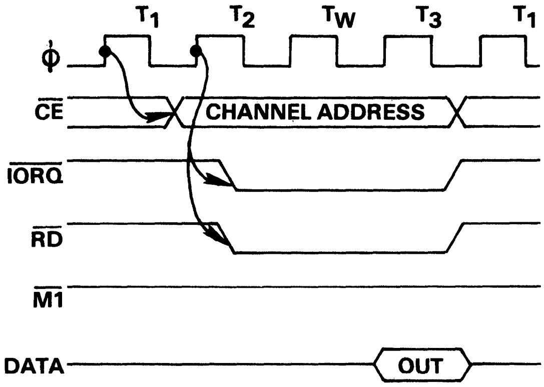


8.0 TIMING

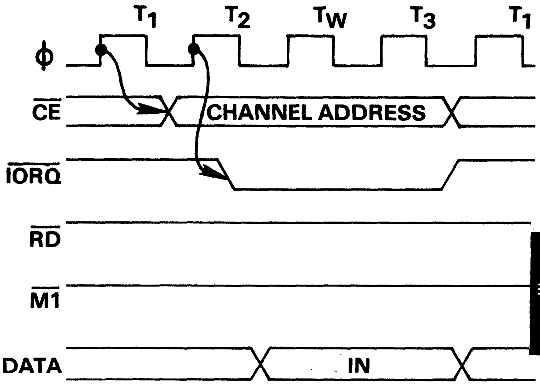
8.1 READ CYCLE

The timing signals generated by a Z80-CPU input instruction to read a Data or Status byte from the Z80-SIO are illustrated in Figure 8.1.

READ CYCLE
Figure 8.1



WRITE CYCLE
Figure 8.2



8.2 INTERRUPT ACKNOWLEDGE CYCLE

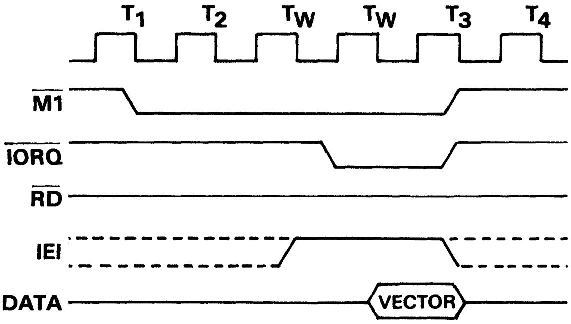
After receiving an Interrupt Request signal ($\overline{\text{INT}}$ pulled Low,) the Z80-CPU sends an Interrupt Acknowledge signal ($\overline{\text{M1}}$ and $\overline{\text{IORQ}}$ both Low). The daisy-chained interrupt circuits determine the highest priority interrupt requestor. The IEI of the highest priority peripheral is terminated High. For any peripheral that has no interrupt pending or under service, IEO=IEI. Any peripheral that does have an interrupt pending or under service forces its IEO Low.

To insure stable conditions in the daisy chain, all-interrupt status signals are prevented from changing while $\overline{\text{M1}}$ is Low. When $\overline{\text{IORQ}}$ is Low, the highest priority interrupt requestor (the one with IEI High) places its interrupt vector on the data bus and sets its internal interrupt-under-service latch.

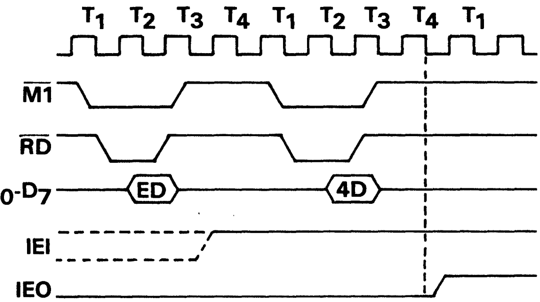
8.3 WRITE CYCLE

Figure 8.2 illustrates the timing and data signals generated by a Z80-CPU output instruction to write a Data or Control byte into the Z80-SIO.

ACKNOWLEDGE CYCLE
Figure 8.3



RETURN FROM INTERRUPT CYCLE



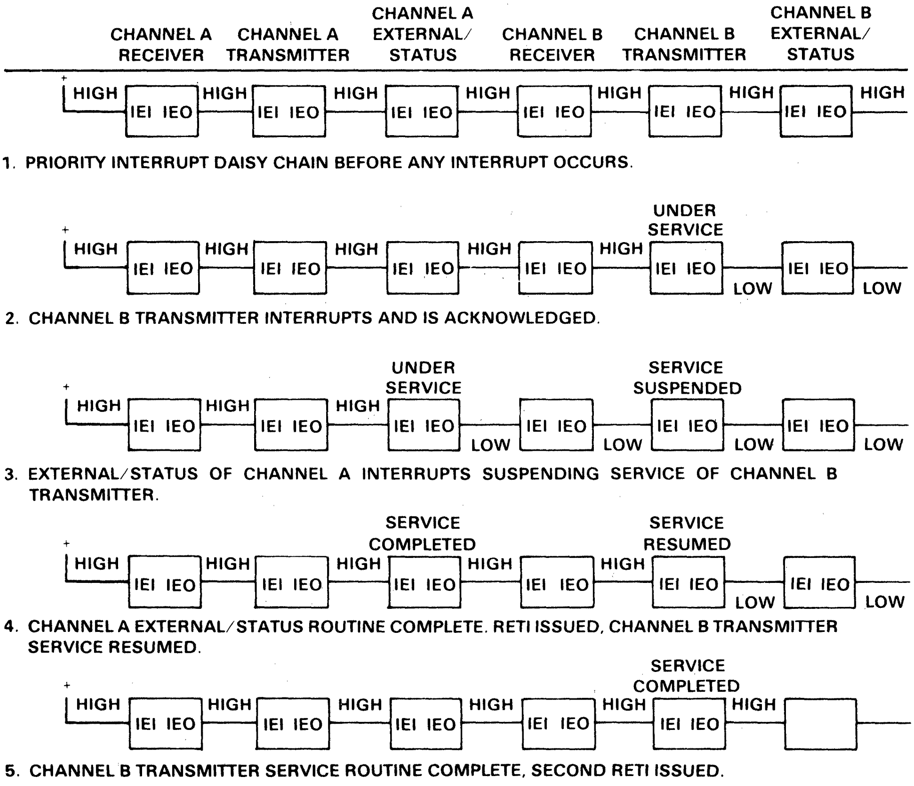
8.4 RETURN FROM INTERRUPT CYCLE

Normally, the Z80-CPU issues a RETI (Return from interrupt) instruction at the end of an interrupt service routine. RETI is a 2-byte opcode (ED-4D) that resets the interrupt-under-service latch to terminate the interrupt that has just been processed. This is accomplished by manipulating the daisy chain in the following way.

The normal daisy chain operation can be used to detect a pending interrupt; however, it cannot distinguish between an interrupt under service and a pending unacknowledged interrupt of a higher priority. Whenever "ED" is decoded, the daisy chain is modified by forcing High the IEO of any interrupt that has not yet been acknowledged. Thus, the daisy chain identifies the device presently under service as the only one with an IEI High and an IEO Low. If the next opcode byte is "4D", the interrupt-under-service latch is reset.

The ripple time of the interrupt daisy chain (both the High-to-Low and the Low-to-High transitions) limits the number of devices that can be placed in the daisy chain. Ripple time can be improved with carry-look-read, or by extending the interrupt aknowledge cycle. For further information about techniques for increasing the number of daisy-chained devices, refer to Mostek Application Note on extending the Z80 Interrupt Daisy Chain.

TYPICAL INTERRUPT SEQUENCE
Figure 8.4



1. PRIORITY INTERRUPT DAISY CHAIN BEFORE ANY INTERRUPT OCCURS.

2. CHANNEL B TRANSMITTER INTERRUPTS AND IS ACKNOWLEDGED.

3. EXTERNAL/STATUS OF CHANNEL A INTERRUPTS SUSPENDING SERVICE OF CHANNEL B TRANSMITTER.

4. CHANNEL A EXTERNAL/STATUS ROUTINE COMPLETE. RETI ISSUED, CHANNEL B TRANSMITTER SERVICE RESUMED.

5. CHANNEL B TRANSMITTER SERVICE ROUTINE COMPLETE, SECOND RETI ISSUED.

8.5 DAISY CHAIN INTERRUPT NESTING

Figure 8.4 illustrates the daisy chain configuration of interrupt circuits and their behavior with nested interrupts (an interrupt that is interrupted by another with a higher priority).

Each box in the illustration could be a separate external Z80 peripheral circuit with a user-defined order of interrupt priorities. However, a similar daisy chain structure also exists inside the Z80-SIO, which has six interrupt levels with a fixed order of priorities.

The case illustrated occurs when the transmitter of Channel B interrupts and is granted service. While this interrupt is being serviced, it is interrupted by a higher priority interrupt from Channel A. The second interrupt is serviced and—upon completion—a RETI instruction is executed or a RETI command is written into the Z80-SIO, resetting the interrupt-under-service latch of the Channel A interrupt. At this time, the service routine for Channel B is resumed. When it is completed, another RETI instruction is executed to complete the interrupt service.

ABSOLUTE MAXIMUM RATINGS

Voltages on all inputs and outputs with respect to GND .... –0.3V to +7.0V
Operating Ambient Temperature .... As Specified in Ordering Information
Storage Temperature .... –65°C to +150°C

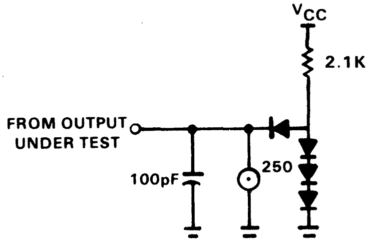
Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

STANDARD TEST CONDITIONS

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin. Standard conditions are as follows:

- $+4.75V \leq V_{CC} \leq +5.25V$
- GND = 0V
- T_A as specified in Ordering Information

All ac parameters assume a load capacitance of 100 pF max. Timing references between two output signals assume a load difference of 50 pF max.



DC CHARACTERISTICS

SYM	PARAMETER	MIN	MAX	UNIT	TEST CONDITION
V_{ILC}	Clock Input Low Voltage	-0.3	+0.80	V	
V_{IHC}	Clock Input High Voltage	V_{CC} -0.6	+5.5	V	
V_{IL}	Input Low Voltage	-0.3	+0.8	V	
V_{IH}	Input High Voltage	+2.0	+5.5	V	
V_{OL}	Output Low Voltage		+0.4	V	I_{OL} = 2.0mA
V_{OH}	Output High Voltage	+2.4		V	I_{OH} = -250 μA
I_{LI}	Input Leakage Current	-10	±10	μA	$0 < V_{IN} < V_{CC}$
I_Z	3-State Output/Data Bus Input Leakage Current	-10	+10	μA	$0 < V_{IN} < V_{CC}$
$I_{L(SY)}$	SYNC Pin Leakage Current	-40	+10	μA	$0 < V_{IN} < V_{CC}$
I_{CC}	Power Supply Current		100	mA	

Overall specified temperature and voltage range.

CAPACITANCE

SYM	PARAMETER	MIN	MAX	UNIT	TEST CONDITION
C	Clock Capacitance		40	pF	Unmeasured
C_{IN}	Input Capacitance		10	pF	pins returned
C_{OUT}	Output Capacitance		10	pF	to ground

Over specified temperature range; f = 1MHz

AC ELECTRICAL CHARACTERISTICS

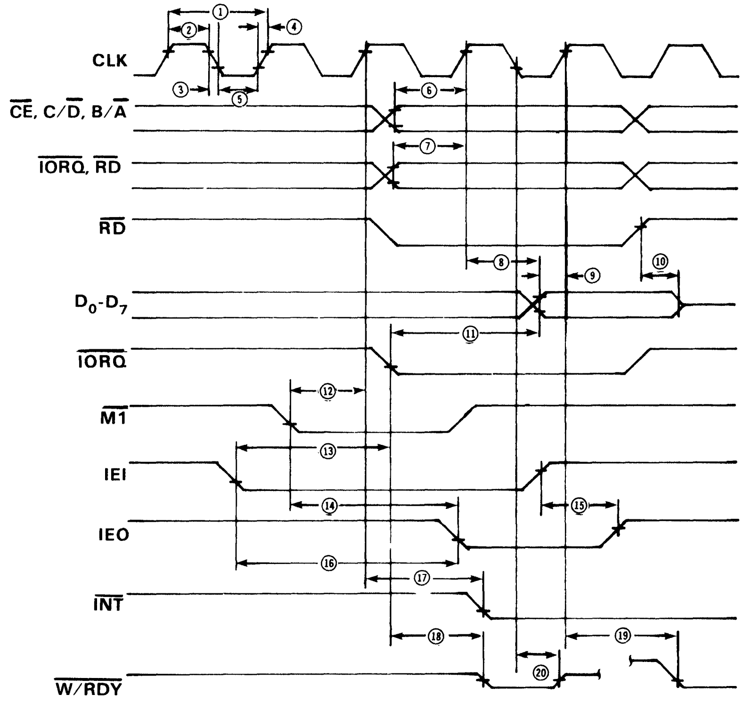
NUMBER	SYM	PARAMETER	MK3884		MK3884-4		UNIT
			MIN	MAX	MIN	MAX	
1	TcC	Clock Cycle Time	400	4000	250	4000	ns
2	TwCh	Clock Width (High)	170	2000	105	2000	ns
3	TfC	Clock Fall Time		30		30	ns
4	TrC	Clock Rise Time		30		30	ns
5	TwC1	Clock Width (Low)	170	2000	105	2000	ns
6	TsAD(C)	$\overline{CE}$, C/$\overline{D}$, B/$\overline{A}$ to Clock ↑ Setup Time	160		145		ns
7	TsCS(C)	$\overline{IORQ}$, $\overline{RD}$ to Clock ↑ Setup Time	240		115		ns
8	TdC(DO)	Clock ↑ to Data Out Delay		240		220	ns
9	TsDI(C)	Data In to Clock ↑ Setup (Write or $\overline{M1}$ Cycle)	50		50		ns
10	TdRD(DOz)	RD ↑ to Data Out Float Delay		230		110	ns
11	TdIO(DOI)	IORQ ↓ to Data Out Delay (INTA Cycle)		340		160	ns
12	TsM1(C)	$\overline{M1}$ to Clock ↑ Setup Time	210		90		ns
13	TsIEI(IO)	IEI to $\overline{IORQ}$ ↓ Setup Time (INTA Cycle)	200		140		ns
14	TdM1(IEO)	$\overline{M1}$ ↓ to IEO ↓ Delay (interrupt before $\overline{M1}$)		300		190	ns
15	TdIEI(IEOr)	IEI ↑ to IEO ↑ Delay (after ED decode)		150		100	ns
16	TdIEI(IEOf)	IEI ↓ to IEO ↓ Delay		150		100	ns
17	TdC(INT)	Clock ↑ to $\overline{INT}$ ↓ Delay		200		200	ns
18	TdIO (W/RWf)	$\overline{IORQ}$ ↓ or $\overline{CE}$ ↓ to $\overline{W/RDY}$ ↓ Delay Wait Mode		300		210	ns
19	TdC (W/RR)	Clock ↑ to $\overline{W/RDY}$ ↓ Delay (Ready Mode)		120		120	ns
20	TdC (W/RWz)	Clock ↓ to $\overline{W/RDY}$ Float Delay (Wait Mode)		150		130	ns
21	Th	Any unspecified Hold when Setup is specified	0		0		ns

AC ELECTRICAL CHARACTERISTICS (continued)

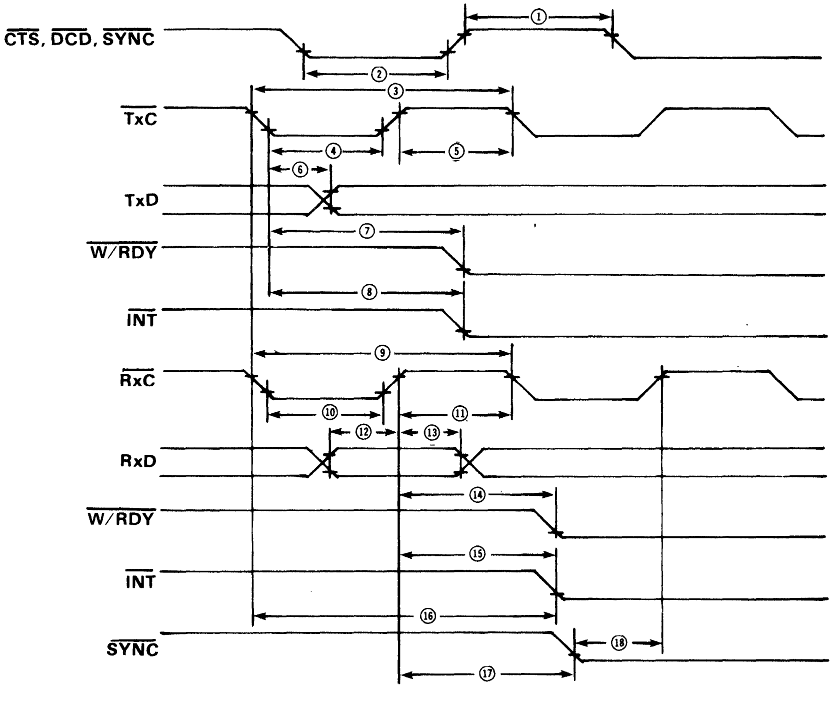
NUMBER	SYM	PARAMETER	MK3884		MK3884-4		UNIT
			MIN	MAX	MIN	MAX	
1	TwPh	Pulse Width (High)	200		200		ns
2	TwPl	Pulse Width (Low)	200		200		ns
3	TcTxC	$\overline{\mathrm{TxC}}$ Cycle Time	400	∞	400	∞	ns
4	TwTxCl	$\overline{\mathrm{TxC}}$ Width (Low)	180	∞	180	∞	ns
5	TwTxCh	$\overline{\mathrm{TxC}}$ Width (High)	180	∞	180	∞	ns
6	TdTxC(TxD)	$\overline{\mathrm{TxC}}$ ↓ to TxD Delay (x1 Mode)		400		300	ns
7	TdTxC (W/RRf)	$\overline{\mathrm{TxC}}$ ↓ to W/RDY ↓ Delay (Ready Mode)	5	9	5	9	Clk Periods*
8	TdTxC(INT)	$\overline{\mathrm{TxC}}$ ↓ to INT ↓ Delay	5	9	5	9	Clk Periods*
9	TcRxC	$\overline{\mathrm{RxC}}$ Cycle Time	400	∞	400	∞	ns
10	TwRxCl	$\overline{\mathrm{RxC}}$ Width (Low)	180	∞	180	∞	ns
11	TwRxCh	$\overline{\mathrm{RxC}}$ Width (High)	180	∞	180	∞	ns
12	TsRxD(RxC)	RxD to $\overline{\mathrm{RxC}}$ ↑ Setup Time (x1 Mode)	0		0		ns
13	ThRxD(RxC)	$\overline{\mathrm{RxC}}$ ↑ to RxD Hold time (x1 Mode)	140		140		ns
14	TdRxC (W/RRf)	$\overline{\mathrm{RxC}}$ ↑ to $\overline{\mathrm{W/RDY}}$ ↓ Delay (Ready Mode)	10	13	10	13	Clk Periods*
15	TdRxC(INT)	$\overline{\mathrm{RxC}}$ ↑ to $\overline{\mathrm{INT}}$ ↓ Delay	10	13	10	13	Clk Periods*
16	TdTxC(INT)	$\overline{\mathrm{TxC}}$ ↓ to $\overline{\mathrm{INT}}$ ↓ Delay	5	9	5	9	Clk Periods*
17	TdRxC (SYNC)	$\overline{\mathrm{RxC}}$ ↑ to $\overline{\mathrm{SYNC}}$ ↓ Delay (Output Modes)	4	7	4	7	Clk Periods*
18	TsSYNC (RxC)	$\overline{\mathrm{SYNC}}$ ↓ to $\overline{\mathrm{RxC}}$ ↑ Setup (External Sync Modes)	-100		-100		ns

In all modes, the System Clock rate must be at least five times the maximum data rate.
RESET must be active a minimum of one complete Clock Cycle.
*System Clock

AC ELECTRICAL CHARACTERISTICS

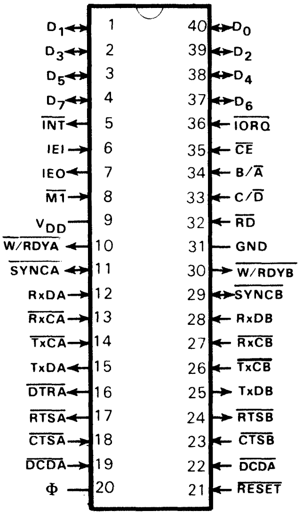


AC ELECTRICAL CHARACTERISTICS

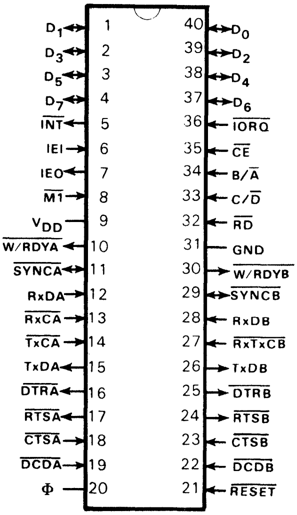


PIN ASSIGNMENTS

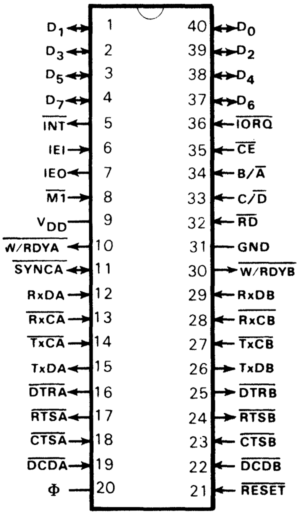
MK3885
Z80-SIO/1



MK3884
Z80-SIO/0



MK3887
Z80-SIO/2



ORDERING INFORMATION

PART NO.		PACKAGE TYPE	MAX CLOCK FREQUENCY	TEMPERATURE RANGE
MK3884N	Z80-SIO/0	Plastic	2.5MHz	0°C to + 70°C
MK3884P	Z80-SIO/0	Ceramic	2.5MHz	0°C to + 70°C
MK3884J	Z80-SIO/0	CERDIP	2.5MHz	0°C to + 70°C
MK3884N-10	Z80-SIO/0	Plastic	2.5MHz	-40°C to + 85°C
MK3884P-10	Z80-SIO/0	Ceramic	2.5MHz	-40°C to + 85°C
MK3884J-10	Z80-SIO/0	CERDIP	2.5MHz	-40°C to + 85°C
MK3884N-4	Z80A-SIO/0	Plastic	4MHz	0°C to + 70°C
MK3884P-4	Z80A-SIO/0	Ceramic	4MHz	0°C to + 70°C
MK3884J-4	Z80A-SIO/0	CERDIP	4MHz	0°C to + 70°C
MK3885N	Z80-SIO/1	Plastic	2.5MHz	0°C to + 70°C
MK3885P	Z80-SIO/1	Ceramic	2.5MHz	0°C to + 70°C
MK3885J	Z80-SIO/1	CERDIP	2.5MHz	0°C to + 70°C
MK3885N-10	Z80-SIO/1	Plastic	2.5MHz	-40°C to + 85°C
MK3885P-10	Z80-SIO/1	Ceramic	2.5MHz	-40°C to + 85°C
MK3885J-10	Z80-SIO/1	CERDIP	2.5MHz	-40°C to + 85°C
MK3885N-4	Z80A-SIO/1	Plastic	4MHz	0°C to + 70°C
MK3885P-4	Z80A-SIO/1	Ceramic	4MHz	0°C to + 70°C
MK3885J-4	Z80A-SIO/1	CERDIP	4MHz	0°C to + 70°C
MK3887N	Z80-SIO/2	Plastic	2.5MHz	0°C to + 70°C
MK3887P	Z80-SIO/2	Ceramic	2.5MHz	0°C to + 70°C
MK3887J	Z80-SIO/2	CERDIP	2.5MHz	0°C to + 70°C
MK3887N-10	Z80-SIO/2	Plastic	2.5MHz	-40°C to + 85°C
MK3887P-10	Z80-SIO/2	Ceramic	2.5MHz	-40°C to + 85°C
MK3887J-10	Z80-SIO/2	CERDIP	2.5MHz	-40°C to + 85°C
MK3887N-4	Z80A-SIO/2	Plastic	4MHz	0°C to + 70°C
MK3887P-4	Z80A-SIO/2	Ceramic	4MHz	0°C to + 70°C
MK3887J-4	Z80A-SIO/2	CERDIP	4MHz	0°C to + 70°C

NOTE: Refer to section on Pin Description for explanation of the differences between the MK3884, MK3885, and MK3887.

MK3886
COMBO CHIP

TABLE OF CONTENTS

1.0 INTRODUCTION

1.1 The MK3886 Combo chip is a Z80 microprocessor peripheral containing a combination of features that enables a user to have great flexibility with a single component. The chip contains 256 bytes of RAM, two timers, a serial Input/Output port and three external interrupt inputs. Additionally, the interrupt vector and priority circuitry can be software programmable to configure the chip for special user requirements. The Combo chip utilizes N-channel silicon gate depletion load technology and is packaged in a 40-pin DIP. Major features of the MK3886 are:

□ 256 x 8 static RAM - 64 bytes of which can operate in write protected or low power standby mode

□ Two programmable timers which operate from an independent clock source

□ Three external interrupt channels with programmable vector for each channel

□ Serial I/O port - synchronous or asynchronous operation with end of word interrupt

□ Z80 compatible daisy chain interrupt structure

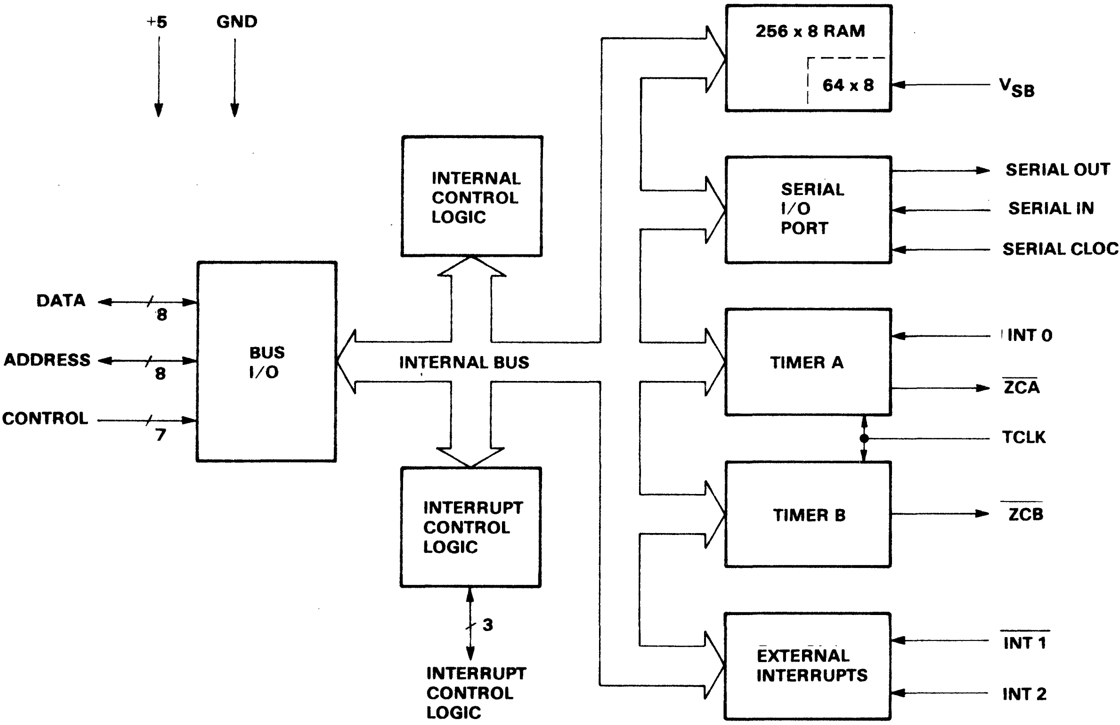
□ Single 5 volt supply (±5%)

2.0 MK3886 ARCHITECTURE

2.1 A block diagram of the MK3886 Combo chip is shown in Figure 2.0-1. The internal structure of the Combo chip consists of a latched CPU bus interface, internal control logic, 256 bytes of random access memory, serial I/O port, two timers, and three external interrupt channels. This part is not restricted for use only in Z80 CPU systems but can easily be adapted to other bus oriented systems.

MK3886 BLOCK DIAGRAM

Figure 2.0-1



The internal control logic of the Combo Chip receives and decodes the control sequences to be performed from the CPU bus. Three types of access cycles may occur: memory read/write, I/O read/write, and interrupt acknowledge. None of these operations are dependent upon the timer clock, TCLK, but rather on timing conditions present on the control input lines. The TCLK input is only used to drive the two timers. The control logic and I/O ports can be addressed via ten registers. These internal registers serve to configure the chip for proper operation and provide a means for exchange of control and data information between the CPU and Combo chip.

Four individual microcomputer component features can be identified within the Combo Chip. These features are Read/Write memory, counting and timing channels, serial input/output, and external interrupt inputs. These combinations enable the MK3886 to function as a key element in a minimum component Z80 system.

The memory consists of 256 bytes of static RAM. The lower 64 bytes have two special features: Write protection and standby power. The write protection protects this memory area from undesired write operations. The low power standby RAM provides a method to preserve important data during a loss of system power.

Two versatile software programmable timers are provided. The Timer clock (TCLK) input is used by both timers to provide an accurate time base. In addition to the zero count output on both timers, Timer A has an external interrupt input linked directly to its control circuitry which provides two additional timer modes.

The serial port allows input and output of serial data in either asynchronous or synchronous modes. The port is basically a 16 bit shift register that can be read from or written to while the data is being shifted at a rate determined by the external serial clock. This port can be used to provide serial data communications or to interface to external serial logic such as shift registers or serial memories (CCD).

Three external interrupt lines are provided in order to allow prioritized, vectored, maskable, edge triggered external interrupt inputs. All interrupt lines are TTL compatible with Schmitt trigger buffered input circuits.

The interrupt control logic section handles all CPU interrupt protocol for nested priority interrupt structures. Priority is determined in two ways. First, an internal priority has been assigned for the 7 channels capable of generating an interrupt. This priority is listed in section 7. Secondly, priority of any component device is determined by its physical location in a daisy chain configuration. Two lines are provided in each Combo chip to form this daisy chain with the device closest to the CPU having the highest priority.

The MK3886 requires only a single +5 volt supply. However a second input, V_{SB}, is provided to supply power to the low power standby RAM. This supply input can be tied either to a battery back up supply or simply to V_{CC} if low power standby is not required.

3.0 PIN DESCRIPTION

A diagram of the MK3886 Combo chip is shown in Figure 3.0-1. This section describes the function of each pin.

D_7-D_0. CPU Data Bus (bi-directional, tri-state). This eight bit bus is used to transfer data and control information between the MK3886 and CPU. DO is the least significant bit.

A_7-A_0. Address Input Bus (input, active high). The eight address input lines are connected to the CPU address bus and are used to select either the RAM memory address or an I/O port.

$\overline{\text{M1}}$. Machine Cycle One from CPU (input, active low). This signal from the CPU is used as a sync pulse during an interrupt acknowledge cycle. When $\overline{\text{M1}}$ is active and $\overline{\text{IORQ}}$ is active, the CPU is acknowledging an interrupt.

$\overline{\text{IORQ}}$. Input/Output Request from Z80-CPU (input, active low). The $\overline{\text{IORQ}}$ signal is used in conjunction with the $\overline{\text{CS}_{\text{I/O}}}$, $\overline{\text{RD}}$, and $\overline{\text{WR}}$ signals to transfer commands and data between the CPU and the Combo Chip. When $\overline{\text{CS}_{\text{I/O}}}$, $\overline{\text{RD}}$, and $\overline{\text{IORQ}}$ are active, the contents of the port addressed from address inputs A3-A0 will be placed on the data bus (a read operation). When $\overline{\text{CS}_{\text{I/O}}}$, $\overline{\text{WR}}$, and $\overline{\text{IORQ}}$ are active, the port addressed by address inputs A3-A0 will be loaded with the contents of the data bus. Also, if $\overline{\text{IORQ}}$ and $\overline{\text{M1}}$ are active simultaneously, the CPU is acknowledging an interrupt. During this time, the interrupting device within the Combo Chip will place its interrupt vector on the CPU data bus if it is the highest priority device requesting an interrupt.

$\overline{\text{MREQ}}$. Memory Request from CPU (input, active low). The $\overline{\text{MREQ}}$ signal is used in conjunction with the $\overline{\text{CS}_\text{M}}$, $\overline{\text{RD}}$, and $\overline{\text{WR}}$ signals to either read or write to the RAM location addressed from A_7-A_0.

$\overline{\text{RD}}$. Read Cycle Status from the CPU (input, active low). An active $\overline{\text{RD}}$ in conjunction with $\overline{\text{CS}_\text{M}}$ and $\overline{\text{MREQ}}$ signals that a memory read is in progress. The MK3886 responds by placing the contents of the addressed RAM location on the data bus. If $\overline{\text{CS}_{\text{I/O}}}$ and $\overline{\text{IORQ}}$ are active simultaneously with $\overline{\text{RD}}$, the MK3886 responds by placing the I/O port contents on the data bus.

$\overline{\text{WR}}$. Write Cycle Status (Input, active low). An active $\overline{\text{WR}}$, $\overline{\text{CS}_\text{M}}$, and $\overline{\text{MREQ}}$ indicates that a memory write is in progress to the RAM. If an active $\overline{\text{WR}}$, $\overline{\text{CS}_{\text{I/O}}}$, and $\overline{\text{IORQ}}$ occur, it indicates an I/O write is in progress.

$\overline{\text{CS}_\text{M}}$. Chip Select RAM (input, active low). $\overline{\text{CS}_\text{M}}$ selects the 256 x 8 RAM for a Memory Read or Write cycle. $\overline{\text{CS}_\text{M}}$ is usually decoded from the CPU Address Bus

$\overline{\text{CS}_{\text{I/O}}}$. Chip Select I/O (input, active low). $\overline{\text{CS}_{\text{I/O}}}$ selects the MK3886 for an I/O Read or Write cycle. $\overline{\text{CS}_{\text{I/O}}}$ is usually decoded from the CPU Address Bus.

$\overline{\text{INT}}$. Interrupt Request (output, open drain, active low). This signal is active low to signal the CPU whenever an interrupt is pending from one of the seven interrupt channels within the MK3886.

IEI. Interrupt Enable In (input, active high). This signal is used to help form an interrupt priority daisy chain when more than one peripheral device in the system has interrupting capability. A high level on this pin indicates that no other interrupting devices of higher priority in the daisy chain are requesting interrupt service from the CPU or being serviced by an interrupt subroutine.

IEO. Interrupt Enable Out (output, active high). The IEO signal, in conjunction with IEI, is used to form an interrupt priority daisy chain. IEO is high only if IEI is high and an interrupt is not being requested from any Combo channel.

$\overline{\text{INT1}}$, INT2. External Interrupt lines. These two edge triggered interrupt lines are uncommitted to system activity and can be used to flag the CPU as desired by external activity. $\overline{\text{INT1}}$ generates an interrupt on a negative edge whereas INT2 generates an interrupt on a positive edge transition.

TCLK. Timer Clock (input). This single phase clock is used as an input to the two timers on the MK3886. The TCLK input may be asynchronous with respect to the Read and Write control signals.

SRIN. Serial Data Input (input). This signal is the input data line to the Combo chip's serial port.

SROUT. Serial Data Output (output). This signal is the output data line from the serial port.

SRCLK. Serial Port Clock (input). This is the clock used to derive the shift register clock in order to shift data into or out of the serial port.

INTO. External Interrupt O (input, programmed as active high or low). INTO is used in conjunction with Timer A to operate in the Pulse Width Measurement Mode or Event Counter Mode. It can also be used as an External Interrupt input and can be programmed to generate an interrupt on either a positive or negative edge transition. Because this interrupt is referenced from TCLK, TCLK must be present to allow interrupts.

$\overline{\text{ZCA}}$. Zero Count Out (out, active low). This signal is a zero count output which pulses low for an integral multiple of Timer clock cycles. The number of clock cycles is a function of the selected prescale value.

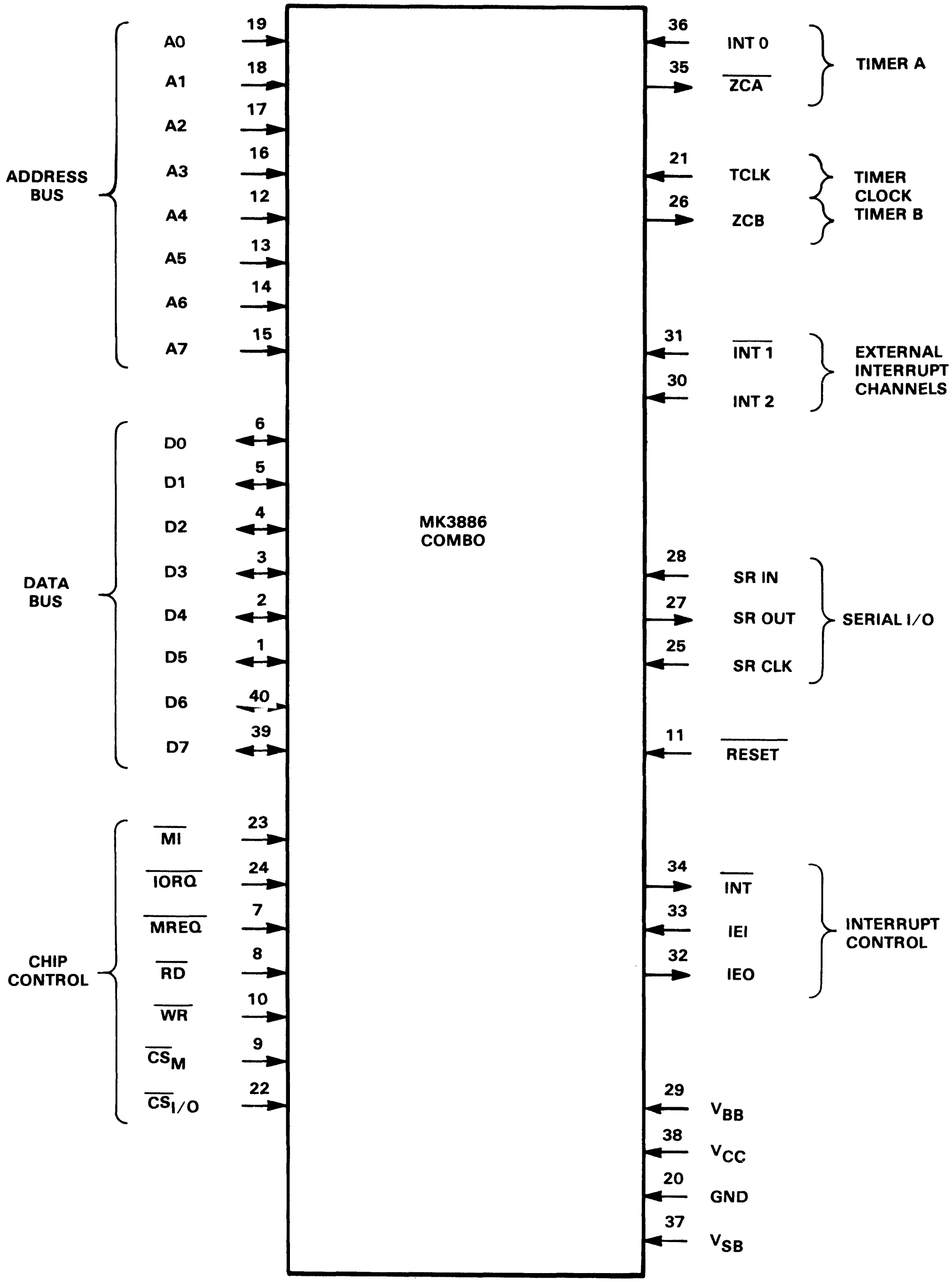
ZCB. Zero Count Out (output, active high). The Zero Count output pin is toggled every time Timer B counts to zero. This produces a square wave which is half the time out frequency. This output can be connected to SRCLK to drive the serial port. This pin can also be programmed via timer B to provide asymetrical output wave forms.

$\overline{\text{RESET/RAMPRT}}$. Reset (input, active low). $\overline{\text{RESET}}$ disables all interrupts, masks all interrupts, stops both timers, and prevents any memory access from occuring. It also forces SROUT and ZCA to a logical 1 output condition. ZCB is forced to a logical zero condition. The $\overline{\text{RAMPRT}}$ is the RAM protect control signal. When it is brought low, the RAM is disabled and therefore protected against any alternations.

V_{BB}. Substrate Decoupling. This pin allows a .01 microfared capacitor to be tied to the substrate to allow additional substrate decoupling when powering V_{CC} up and down.

V_{SB}. Low Power Standby Supply. This supply voltage is used to power the 64 bytes of low power RAM. The voltage can vary from 3.2 to 5.6 volts D.C. and is independent of V_{CC}.

MK3886 PIN CONFIGURATION
Figure 3.0-1



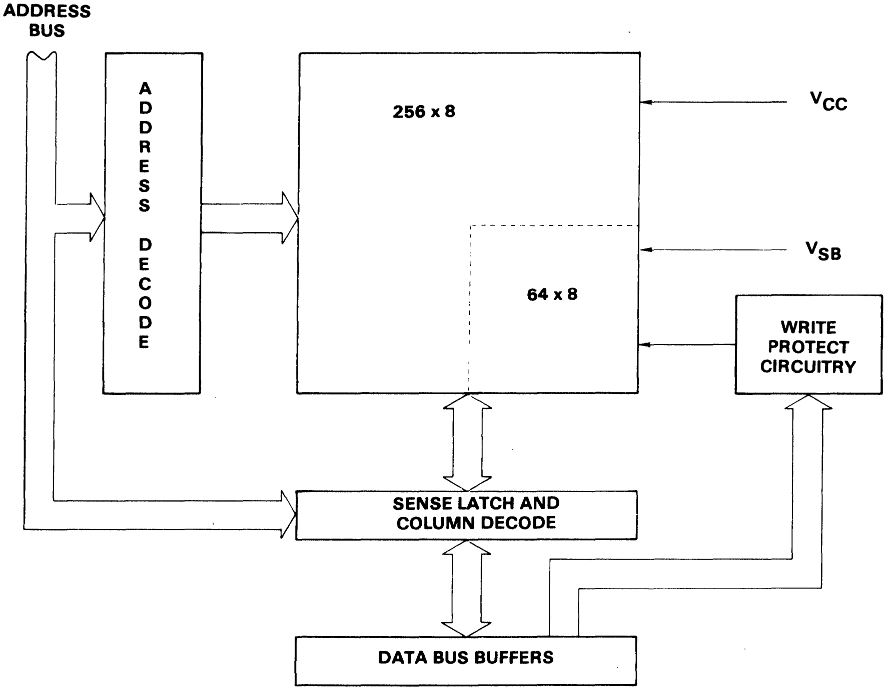
III
Z80 FAMILY
TECHNICAL
MANUALS

4.0 RANDOM ACCESS MEMORY

4.1 OVERVIEW

The MK3886 has 256 bytes of Edge Activated static Random Access Memory. Sixty-four bytes of this RAM can be configured to operate in a low power battery back-up mode when storage of critical data is imperative. Also included with these 64 bytes is a write protection circuit. This circuit is designed to permit only authorized write operations to this area of memory. A register programmable via the CPU enables and disables the write protection.

BLOCK DIAGRAM OF 256 x 8 RAM
Figure 4.0-1



The static memory is a 256 x 8 array. The lower 8 address bus bits are decoded within the chip to provide addressing from 0 to 255. This memory can be decoded on 256 word memory blocks anywhere within a users' system memory map by using the memory chip select input CS_M.

The lowest 64 bytes of the RAM (address 0-63) are both write protected and capable of maintaining data in a low power standby mode. This memory block has an independent power supply input and write control circuitry. If standby power is desired, the standby power source input V_{SB} can be connected to a supply independent of the V_{CC} input. Three nickel - cadmium batteries (typical voltage= 3.75 V) can be used in a trickle charge circuit to provide power. The power requirements are such that less than 8.2 mW are required at a minimum 3.2 V standby supply voltage. Power supply tolerance is not critical. The memory will function over a range of 3.2 V to 5.5 volts. If low power standby is not desired, V_{SB} should be tied to V_{CC}, the Combo chip supply voltage.

4.2 PROGRAMMING THE RAM WRITE PROTECTION REGISTER

In addition to the low power standby mode, a write protect circuit is provided to prevent undesired writing of data into the lower 64 bytes of memory. This circuit functions in three modes. The first mode allows normal Read/Write operation. The second mode enables the memory for only one write operation. The third mode inhibits all attempts to write in the memory protected area.

Port A (A_3 - A_0 = H 'A') is designated as the write protect register. It is WRITE only. Its format is defined in the convention below.

D7	D6	D5	D4	D3	D2	D1	D0
WP7	WP6	WP5	WP4	WP3	WP2	WP1	WP0

The write protect circuit is totally disabled when 66H is written to this port. The memory area will function as normal Read/Write memory.

A 55H written to this Port will allow only one write operation to occur. Subsequent attempts to write will be unsucessfull unless another 55H is first written or the write protect circuit disabled. Each time 55H is written to Port A, the next pulse on the $\overline{\text{WR}}$ input will force the port to H'00'.Therefore, the RAM will be enabled for only one write operation. Note that if an interrupt suspends program execution just after 55H was written, when program execution continues the data value could not be written to the memory location unless the write protect circuit was disabled again. This is caused by the $\overline{\text{WR}}$ input toggling during data manipulation in the interrupt service routine.

Any value written to Port A other than H'66' or H'55' will enable the write protection circuit. If the circuit is enabled and a write operation is attempted, data will not be written to the selected location and no error will be reported. The status of the write protect circuit can be obtained by reading the Status Register Port 5, bit 5. Bit 5 will be cleared if a write protection operation is enabled, and set if it is disabled. Upon receipt of a $\overline{\text{RESET}}$ input, the write protect port is loaded with H'00' thus enabling the write protect circuit.

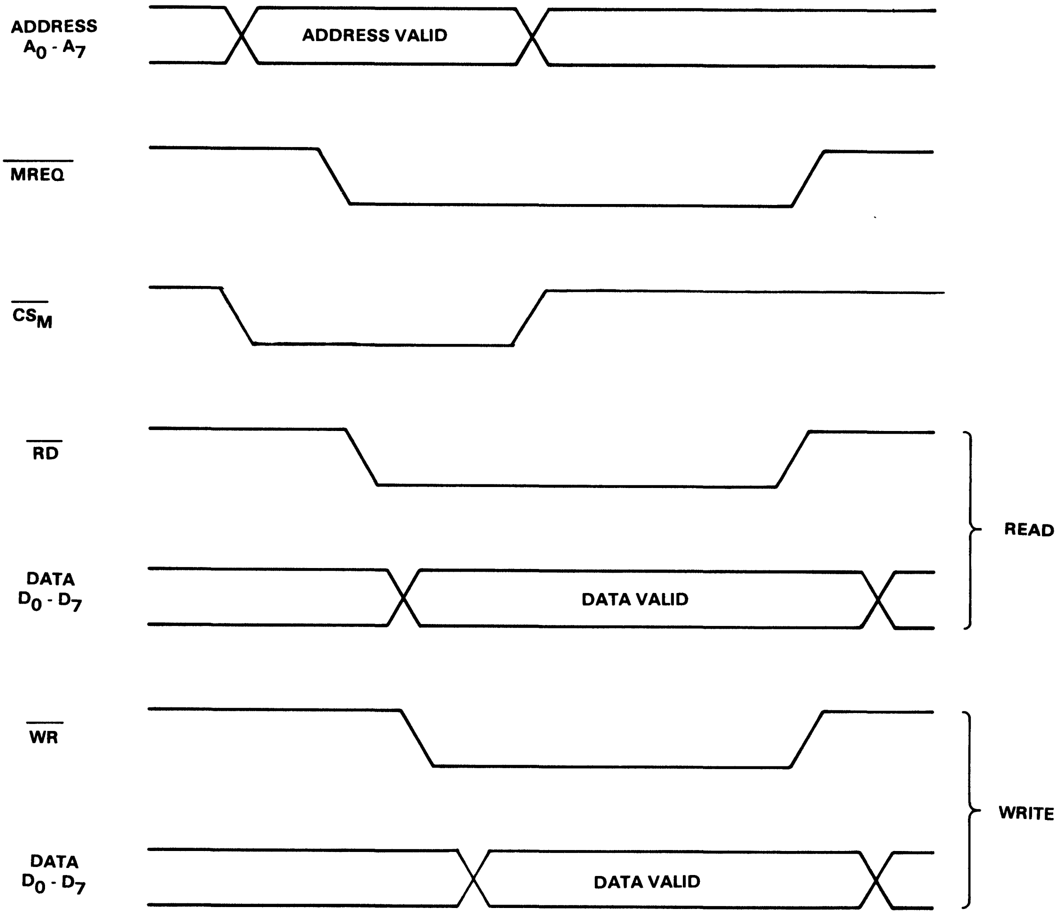
An example of each mode is shown in the Applications Section of this manual.

4.3 MEMORY TIMING

Figure 4.0-2 illustrates timing of Read or Write memory cycles. These cycles are independent of the Timer Clock (TCLK). This feature allows the MK3886 to be CPU independent. The input address lines and chip select are latched and all memory cycle timing is initiated from the falling edge of $\overline{\text{MREQ}}$. Therefore, the address and chip select inputs must be stable before the falling edge of $\overline{\text{MREQ}}$. The status of $\overline{\text{RD}}$ and $\overline{\text{WR}}$ indicated whether a Read or Write cycle will occur. If $\overline{\text{RD}}$ becomes active, a data byte will be gated from the Memory onto the data bus. The data is valid as long as $\overline{\text{RD}}$ is active. If the memory is selected and $\overline{\text{WR}}$ becomes active, data is gated from the bus into the memory on the rising edge of the $\overline{\text{WR}}$ signal.

MEMORY READ/WRITE TIMING

Figure 4.0-2

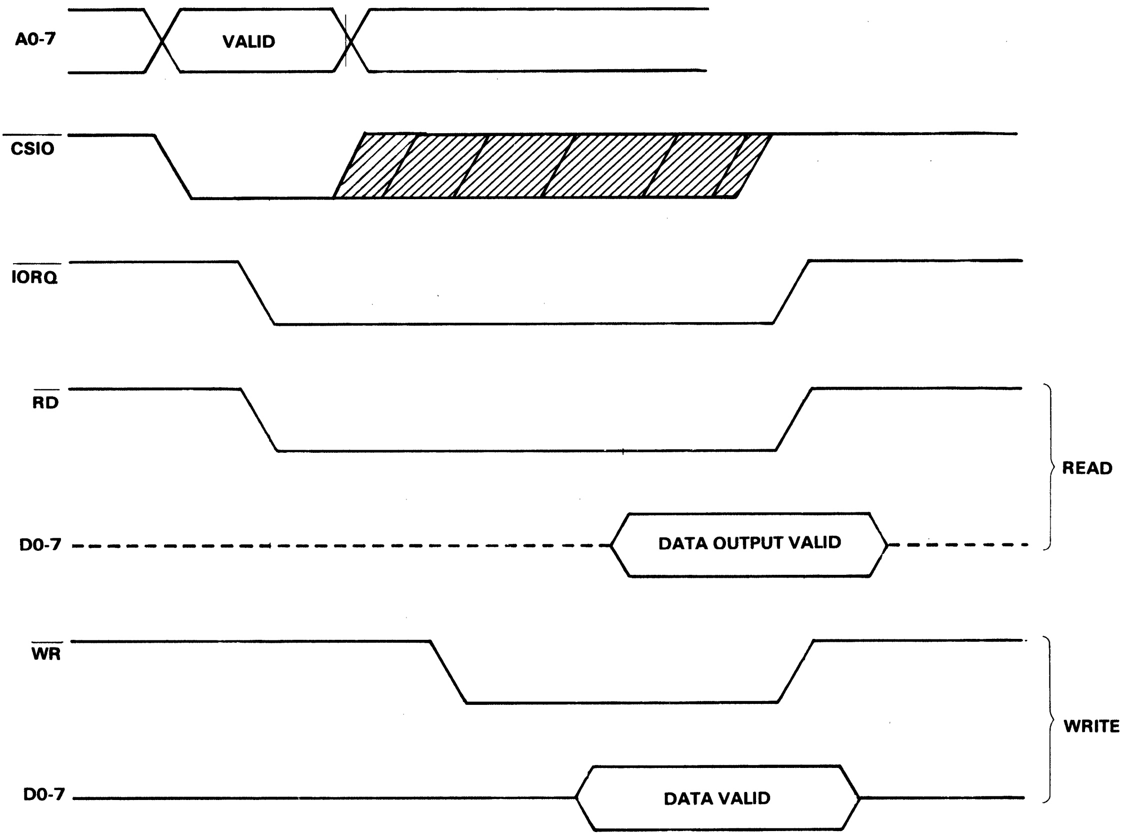


4.4 I/O TIMING

Figure 4.0-3 illustrates timing of I/O Read or Write Operations with the Combo chip. These cycles are independent of the Timer clock (TCLK) input. This feature allows the MK3886 to be CPU independent. The input address lines and chip select are latched and all I/O cycle timing is initiated from the falling edge of $\overline{IORQ}$. Notice that the timing is very similar to the memory timing. The chip select and I/O address lines A3-A0 must be stable before the falling edge of $\overline{IORQ}$. The status of $\overline{RD}$ and $\overline{WR}$ dictate whether a Read or Write cycle will occur. If $\overline{RD}$ becomes active, a data byte will be gated from the selected I/O port onto the data bus. If the selected port is Write only, no data will appear and consequently be interpreted as all "1's" by the CPU. Data is valid on the bus as long as $\overline{RD}$ is active. If an I/O operation is selected and $\overline{WR}$ becomes active, data is gated from the bus into the addressed register port on the rising edge of $\overline{WR}$.

I/O READ/WRITE CYCLE TIMING

Figure 4.0-3



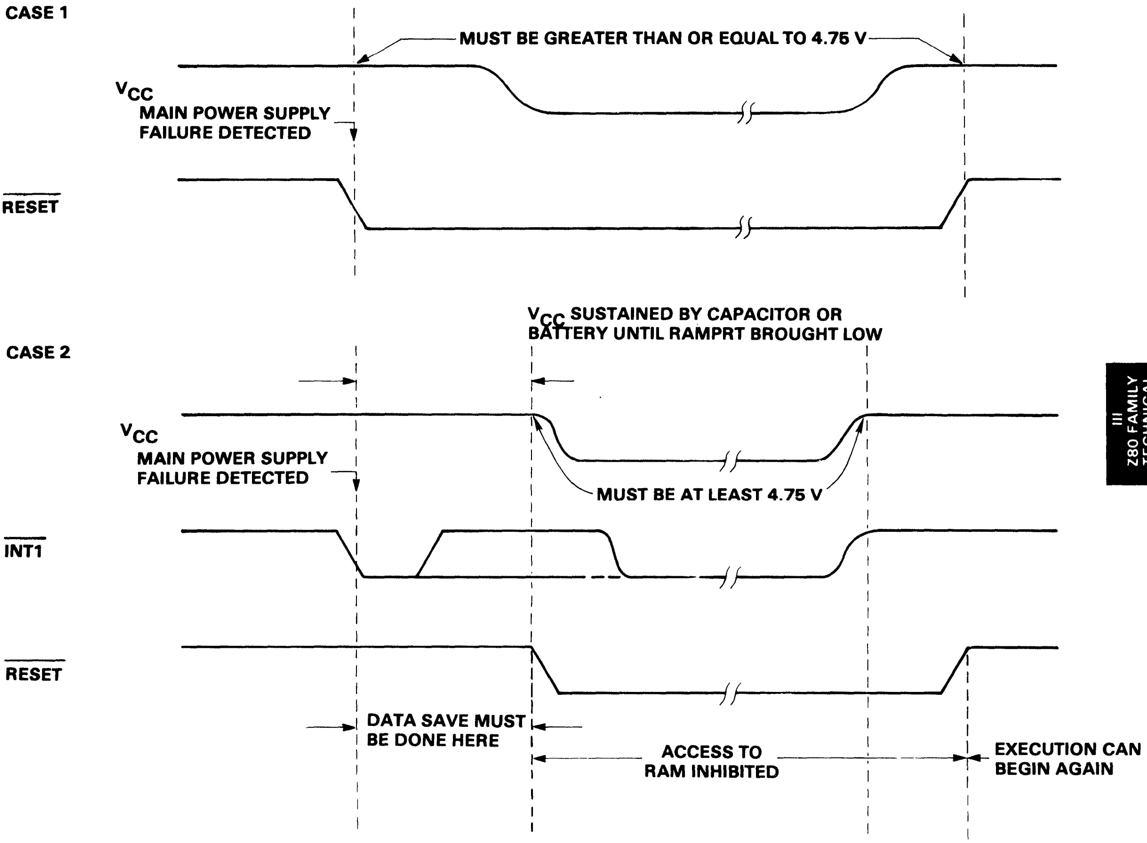
4.5. STANDBY POWER MODE

The MK3886 is designed to provide a low power standby mode for the 64 low order bytes of the on-chip RAM. The standby power source (V_{SB}) is connected to pin 37, RAM Protect Control is tied to the $\overline{\text{RESET}}$ pin and the substrate decoupling capacitor tied to pin 29. V_{CC} must be greater than 4.75 volts before $\overline{\text{RESET}}$ changes states. As the power comes up, $\overline{\text{RESET}}$ should be held low until V_{CC} is above the minimum level.

Figure 4.0-4 illustrates the timing associated with this circuit for two cases. The first case illustrates timing where no save routine is required. In this case whatever data is in the RAM is saved as power goes down (assuming the $\overline{\text{RESET}}$ condition is satisfied). The second case demonstrates how an external interrupt is generated notifying the processor of an impending power failure. All key data must be stored in the RAM. After the save is done, $\overline{\text{RESET}}$ can fall. Remember that upon powerup, this portion of the RAM will be in the Read Only mode.

STANDBY POWER TIMING

Figure 4.0-4

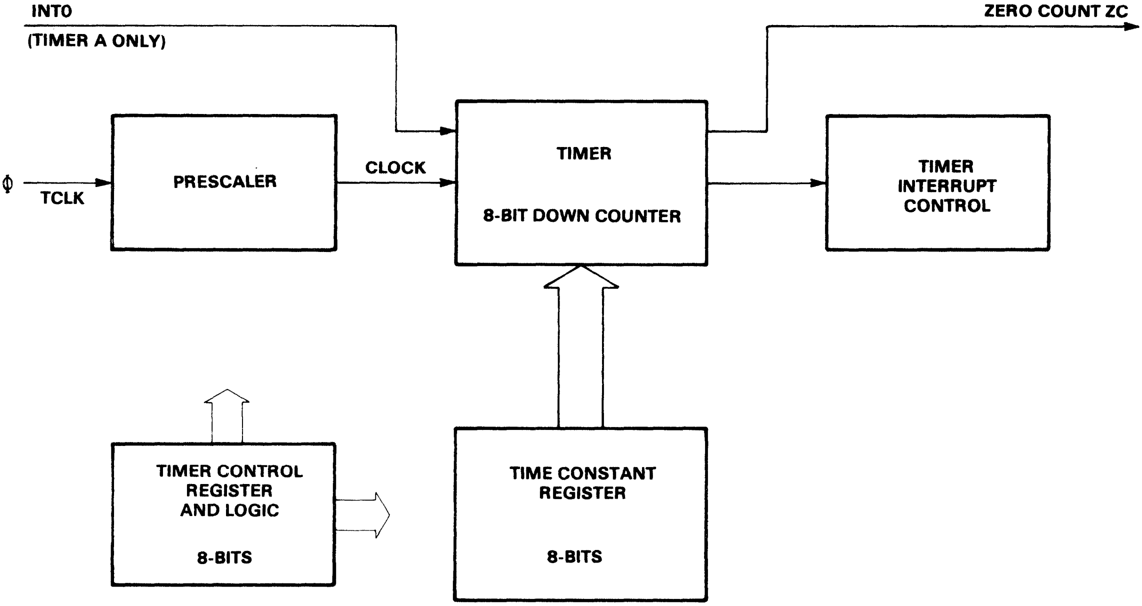

CASE 1
MUST BE GREATER THAN OR EQUAL TO 4.75 V
VCC
MAIN POWER SUPPLY FAILURE DETECTED
RESET
CASE 2
VCC SUSTAINED BY CAPACITOR OR BATTERY UNTIL RAMPRT BROUGHT LOW
VCC
MAIN POWER SUPPLY FAILURE DETECTED
MUST BE AT LEAST 4.75 V
INT1
RESET
DATA SAVE MUST BE DONE HERE
ACCESS TO RAM INHIBITED
EXECUTION CAN BEGIN AGAIN

5.0 TIMERS A AND B

5.1 OVERVIEW

The MK3886 has two programmable timers, designated as Timer A and Timer B. Each timer consists of an 8-bit binary Down Counter, a programmable Prescaler, a Time Constant Register, and a Zero Count output. There are some differences between Timer A and B. These differences basically are due to the fact that the INTO line is tied directly to Timer A's control circuitry. This feature gives Timer A three modes of operation: Interval Timer, Pulse-Width Measurement, and Event Counter. Timer B operates in the Interval Timer Mode only. A basic block diagram which applies to both Timers is illustrated in Fig. 5.0-1. The following description highlights the major functions of each block.

TIMER BLOCK DIAGRAM
Figure 5.0-1



5.1.1. THE CHANNEL CONTROL REGISTER AND LOGIC

The Channel Control Register (8-bit) and Logic, is written to by the CPU to select the modes and parameters of the channel. Within the Combo chip there are two such registers, corresponding to Channel A and Channel B. Each register is assigned a separate I/O port address selected by the state of the lower four address inputs, (A_3-A_0). For specific I/O port assignments for each Timer, refer to the "Programming" section for each timer channel or Section 8.0, "Combo Programming Summary."

5.1.2. THE PRESCALER

Both Timer A and Timer B have a Prescaler which can be programmed via the appropriate Timer Control Register to divide its input, TCLK, by a predetermined prescale value. The output of the Prescaler is then used to clock the Down Counter in the Interval Timer Mode of operation or in the Pulse-Width Measurement Mode of operation. (Pulse-Width Measurement on Timer A only).

5.1.3. THE TIME CONSTANT REGISTER

The Time Constant Register is an 8-bit register used in all count modes and is programmed by the CPU with an integer time constant value of 0 through 255. Under normal operation, this register is loaded into the Down Counter when the Time Constant Register is first written to and is reloaded automatically thereafter whenever the Down Counter counts to zero. (For details of how a time constant is written to a Timer channel, see 5.2.1 "Programming Timer A" and 5.3.1 "Programming Timer B").

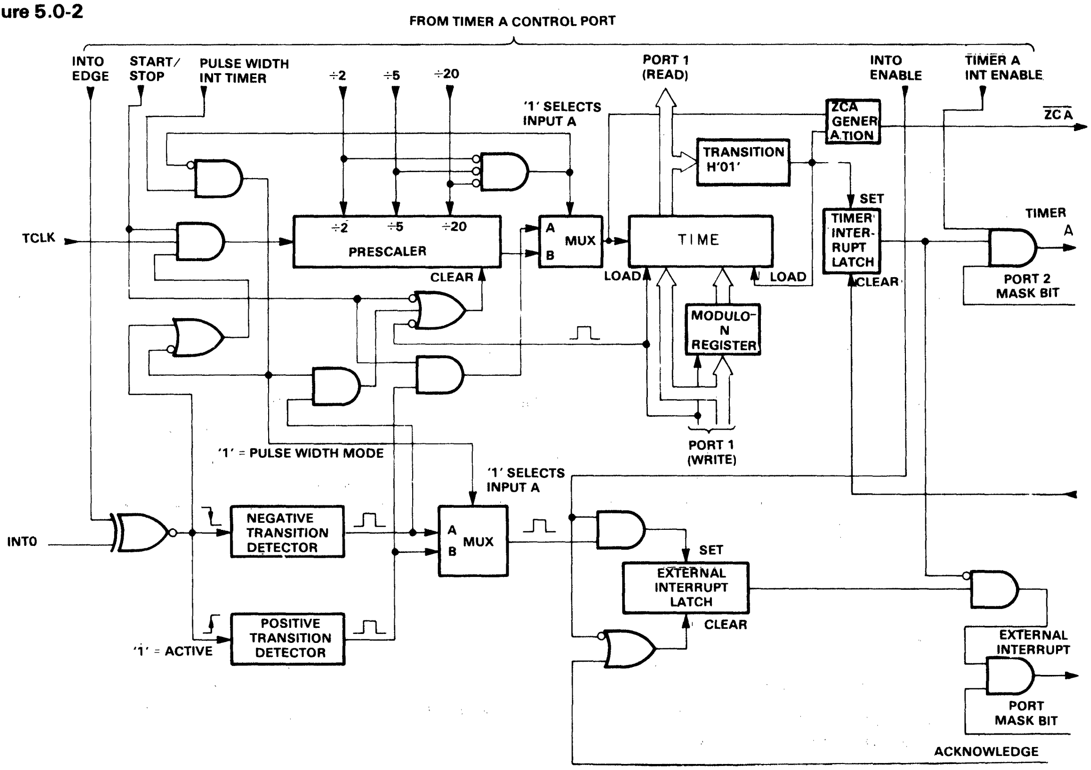
5.1.4. THE DOWN COUNTER

The Down Counter is an 8-bit register used in all count modes. The Down Counter is decremented by the INTO edge in the Event Counter Mode (Timer A only), or by the clock output of the Prescaler in the Interval Timer Mode. At any time, the contents of the Down Counter can be read by performing an I/O READ of the port address assigned to the Down Counter of the selected Timer channel. Either timer channel may be programmed to generate an interrupt request sequence each time the zero count condition in the Down counter is reached.

5.2. TIMER A

Timer A can function in the Interval Timer Mode, the Pulse Width Measurement Mode, or the Event Counter Mode. The Interval Timer mode is used to count TCLK clock periods to generate accurate time intervals. The Pulse Width Measurment Mode is used to accurately measure the duration of a pulse applied to the INTO pin. The Timer begins counting TCLK clock periods when INTO becomes active and stops when it returns to the inactive state. Timer A can also count pulses applied to the INTO pin when it is programmed to operate in the Event Counter mode. A functional diagram of Timer A circuit is shown in Figure 5.0-2.

TIMER A FUNCTIONAL SCHEMATIC
Figure 5.0-2



5.2.1 PROGRAMMING TIMER A

The desired timer operation is selected by loading the Timer A Control Register. This register is designated as Port 0 and is WRITE only.

The format of the Timer A Control Register is as follows:

D7	D6	D5	D4	D3	D2	D1	D0
÷20	÷5	÷2	TIMER A Interrupt Enable	INTO ENABLE	INTO EDGE	PULSE WIDTH INTERVAL TIMER	START STOP

Bit 7, 6, 5 - Prescaler constants.

These bits determine the possible division values selectable for the prescaler. Any of three conditions will cause the prescaler to be reset: whenever $\overline{\text{RESET}}$ is active, whenever the timer is stopped by clearing bit 0 (Start/Stop), or on the trailing edge transition of the INTO pin when in the Pulse Width Measurement Mode. The following table defines the value of the prescaler for various bit assignments.

D7	D6	D5	PRESCALE
0	0	1	÷2
0	1	0	÷5
0	1	1	÷10
1	0	0	÷20
1	0	1	÷40
1	1	0	÷100
1	1	1	÷200

Bit 4 - Timer A Interrupt Enable

When set, this allows Timer A to generate an interrupt request sequence every time the Down Counter reaches a zero count condition.

When cleared interrupts are disabled and any pending interrupt is cleared.

Bit 3 - INTO Interrupt Enable

When set, this allows INTO to generate an interrupt on the active edge selected by Bit 2.

When cleared, interrupts are disabled and any pending interrupt is cleared.

Bit 2 - INTO Edge Active Level

When set, INTO is active high.

When cleared, INTO is active low.

Bit 1 - Pulse Width/Interval Timer Mode Select

When set, the Pulse Width Measurement Mode is selected.

When cleared and bits 7, 6, and 5 are also cleared, the Event Counter Mode is selected. If any prescale bit is set, the Interval Timer Mode is selected.

Bit 0 - Start/Stop

When set, it will enable the Timer for operation.

When cleared, it will stop the counter and reset the prescaler.

Port 1 is designated as the Time Constant Register for Timer A. When a write operation is made to the port, both the time constant register and the Down Counter are loaded. The following is the bit assignment for the Port:

D7	D6	D5	D4	D3	D2	D1	D0
TC7	TC6	TC5	TC4	TC3	TC2	TC1	TC0

The following sections illustrate the operation of the Timer A Control Register in conjunction with each of the three possible timer modes.

5.2.2 INTERVAL TIMER MODE

When Port 0 bit 1 is cleared and at least one prescale bit is set, the Timer operates in the Interval Timer Mode. This mode counts the prescaled system clock and generates a zero count output, ZCA, and an interrupt (if enabled) every time it counts to zero. When bit 0 of the Control Register is set, the Timer will start counting down from the modulo-N value loaded by the Time Constant Register. After counting down to H'01', the Timer returns to the modulo-N value at the next count. On the transition form H'01' to H'N' the Timer sets a timer interrupt request latch. Note that the interrupt request latch is set by the transition to H'N' and not the presence of H'N' in the Timer, thus allowing a full 256 counts if the Time Constant Register is preset to H'00'. If bit 4 of the Control Register is set and the Timer Interrupt is not masked by bit 4 of Port 9, the interrupt request will be passed onto the CPU. However, if bit 4 of the control register is a logic 0, the Interrupt Request is not recorded by the Timer Control logic.

Consider an example in which the Time Constant Register is loaded with H'64' (decimal 100). The timer interrupt request latch will be set and the ZCA output will be pulsed at the 100th count following the timer start and will be repeated precisely on every 100 count interval. If the prescaler is set at ÷5, the timer will count to zero every 500 TCLK periods. For a 2.5MHz TCLK clock, this will produce a 200 micro-second interval. The period of ZCA is given by tc*P*TC where tc is the TCLK period, P is the prescale value, and TC is the time constant value.

The range of possible intervals is from 2 to 51,200 TCLK clock periods (.8 microsecond to 20.48 milliseconds for a 2.5 MHz clock). However, approximately 50 TCLK periods is a practical minimum because of interrupt service time requirements by the CPU. To establish time intervals greater than 51,200 TCLK clock periods is a simple matter of using the timer interrupt service routine to count the number of interrupts, saving the result in a CPU register until the desired interval is achieved. With this technique virtually any time interval, or several time intervals, may be generated.

The Timer may be read at any time and in any mode using an input instruction and may take place "on the fly" without interferring with normal timer operation. Also, the Timer may be stopped at any time by clearing bit 0 of the Control Register. The Timer will hold its current contents indefinitely and will resume counting when bit 0 is again set. Recall however that the prescaler is reset whenever the Timer is stopped; thus a series of starting and stopping will result in cumulative truncation errors.

5.2.3 PULSE WIDTH MEASUREMENT MODE

When Timer A Control Register bit 1 is set (logic 1) and at least one prescale bit is set, the Timer operates in the Pulse Width Measurement Mode. This mode is used for measuring the duration of a pulse applied to the INTO pin. The Timer is stopped and the prescaler is reset whenever INTO is at its inactive level. The active level of INTO is defined by Control Register bit 2; if cleared, INTO is active low; if set, INTO is active high. If bit 0 is set, the prescaler and Timer will start counting whenever a transition is made to the active level on INTO. When INTO returns to the inactive level,the Timer then stops, the prescaler resets, and if bit 3 is set, an external interrupt request latch is set.

As in the Interval Timer Mode, the Timer may be read at any time, or may be stopped at any time by clearing Control Register bit 0. The prescaler and Control Register bit 4 function as previously described and the Timer still functions as an 8-bit binary down counter with the timer interrupt request latch being set on the Timer's transition from H '01' to H 'N'. Note that the INTO pin has nothing to do with loading the Timer; its action is that of automatically starting and stopping the Timer and of generating external interrupts. Pulse widths longer than the prescale value times the modulo-N value are easily measured by using the timer interrupt service routine to store the number of timer interrupts in one or more CPU registers.

As for accuracy, the actual pulse duration is typically slightly longer than the measured value because the status of the prescaler is not readable and is reset when the Timer is stopped. Thus for maximum accuracy it is advisable to use a small division setting for the prescaler.

5.2.4 EVENT COUNTER MODE

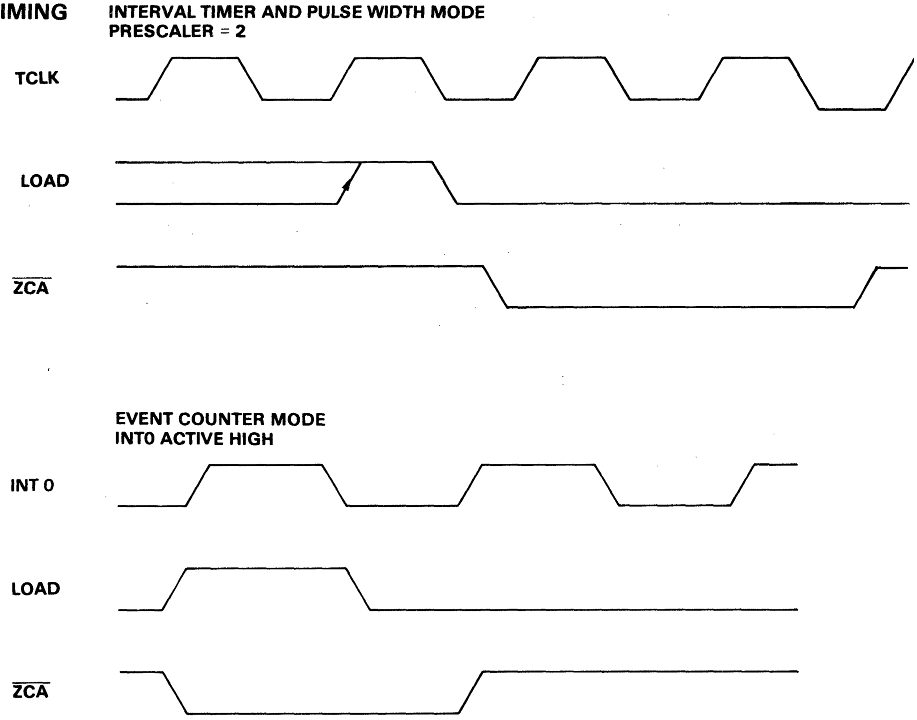
When Timer A Control Register bit 1 is cleared and all prescale bits (5, 6, and 7) are cleared the Timer operates in the Event Counter Mode. This mode is used for counting pulses applied to the INTO pin. If Timer A Control Register bit 0 is set, the Timer will decrement on each transition from the inactive level to the active level on the INTO pin. The prescaler is not used in this mode.

Normally, control Register bit 3 should be kept cleared in the Event Counter Mode; otherwise, external interrupts will be generated on every transition from the inactive level to the active level of the INTO pin.

5.2.5 ZCA Output

Timer A generates an output called ZCA whenever a transition (LOAD) is detected from H'01' to H'N' in the Down Counter indicating that zero count has occurred. An output will be generated on ZCA that is an integral multiple of TCLK cycles. The number of TCLK cycles is a function of the selected value of the Prescaler. For example if the Prescaler is set at 2 then the ZCA output will be active for 2 TCLK cycles. This is illustrated in the timing diagram in Figure 6.4. Since the Prescaler is not used in the Event Counter mode, the ZCA output will only stay active for one INTO clock period. The output state of ZCA is not defined when the Time Constant Register is loaded with H"01". Figure 5.0-3 illustrates two cases for ZCA Output Timing.

ZCA OUTPUT TIMING
Figure 5.0-3



5.3 TIMER B

Timer B is similar in operation to Timer A; however, it functions only as an Interval Timer. Its output, ZCB, is different from Timer A in that it toggles every time it counts down to zero. This feature automatically produces a square wave which is one-half the time out frequency. This allows ZCB to be tied directly to the SRCLK pin to be used as the serial port clock. A functional diagram of Timer B is shown in Figure 5.0-4.

5.3.1 PROGRAMMING TIMER B

Control of the timer circuitry is provided through the Timer B Control Register. It is designated as Port 2 and is WRITE only. The following diagram illustrates the bit assignments within the register.

Timer B Control Register

D7	D6	D5	D4	D3	D2	D1	D0
PS1	PSO	X	X	Start Stop	Timer B Interrupt Enable	X	X

The description of each bit assignment is as follows:

Bits 7 and 6 - PS1 and PSO

These bits determine the division values selectable for the prescaler. The following bit assignments are made for the corresponding prescale values.

PS1	PSO	PRESCALE
0	0	÷2
0	1	÷5
1	0	÷10
1	1	÷20

Bits 5 and 4 - Don't care

Bit 3 - Start/Stop

When set, the timer will be enabled to count.

When cleared, the timer will stop counting and the prescaler will be reset.

Bit 2 - Timer Interrupt Enable

When enabled, this bit allows Timer B to generate an interrupt request sequence every time the Down Counter reaches a zero count condition and if the Timer B Interrupt Mask Register (Port 9) bit 5 is enabled.

When cleared, interrupts are disabled and any pending interrupt is cleared.

Bits 1 and O - Don't Care

5.3.2 TIMER B TIME CONSTANT REGISTER

As with Timer A, Timer B has a single 8-bit Time Constant Register. However, Timer B's Register has two modes of addressing. In the first mode the Time Constant Register and the Down Counter are immediately loaded where in the second mode only the Time Constant Register is loaded. The Former addressing mode is performed with a write command to Port 3; the latter addressing mode is commanded with a write command to Port 4.

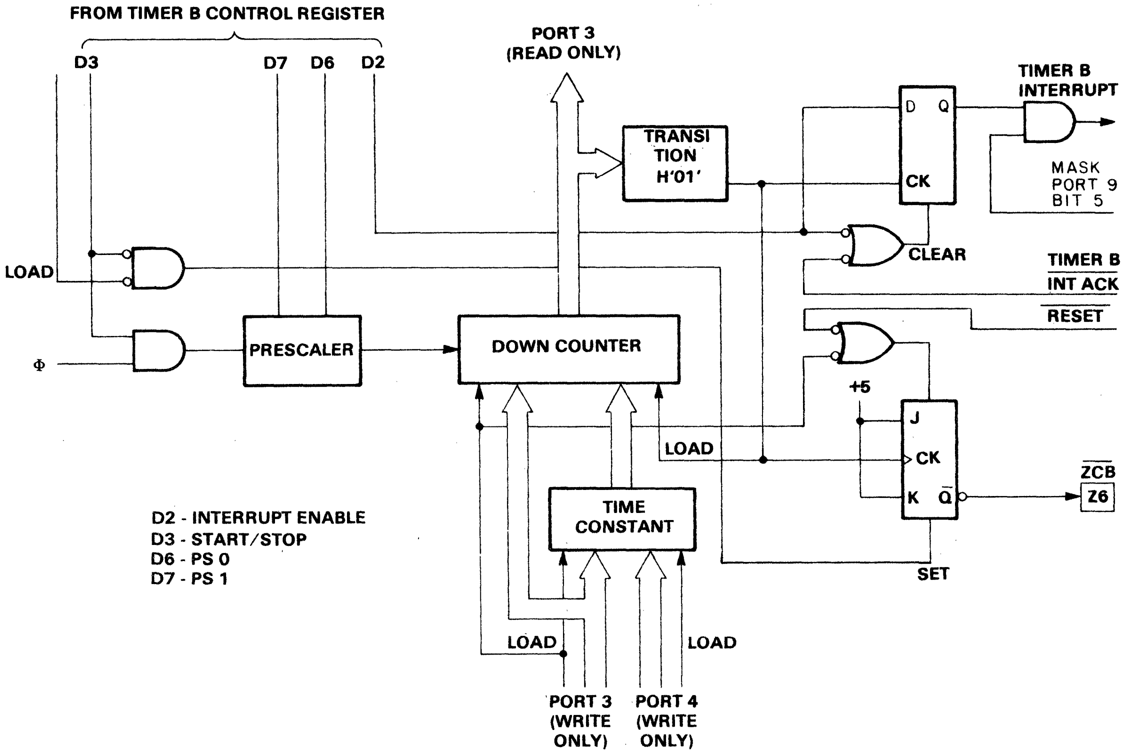
During a write command, Port 3 is loaded with a data byte and its contents are immediately loaded into the Down Counter and Time Constant Register regardless of its present state. When this register is read, the present value of the Down Counter is examined.

Port 4 also addresses Time Constant Register for Timer B. This port is Write only. Writing to this port will not disturb the current timer count. When a new value of Port 4 is loaded, it will not be transferred to the Down Counter until it counts down to zero. With this feature it is possible to generate asymetrical wave forms with the ZCB output. The bit assignments for the port are as follows:

TIMER B TIME CONSTANT REGISTER

D7	D6	D5	D4	D3	D2	D1	DO
TC7	TC6	TC5	TC4	TC3	TC2	TC1	TCO

TIMER B FUNCTIONAL SCHEMATIC
Figure 5.0-4



5.3.3 ZERO COUNT OUTPUT - ZCB

The ZCB output is inverted every time Timer B counts to zero. This can be used to produce a square wave output which is one half the timeout frequency. However, this can be easily modified to generate a pulse train with a variable pulse width and interpulse interval.

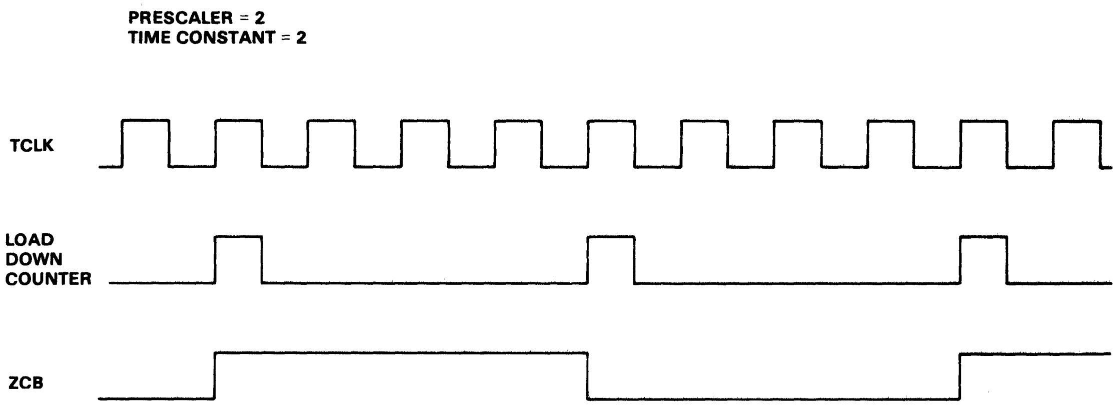
If a square wave is desired, the output period of ZCB is given by the formula:

$$2 * P * t_C * TC$$

where t_C is the period of Timer Clock, TCLK, P is the Prescaler factor, TC is the time constant data word, and 2 is a constant to adjust for the toggle output feature on ZCB. Once the desired Time Constant and Prescale values are loaded and the Timer started, a continous square wave will be produced. Figure 5.0-5 illustrates the timing relationship for the ZCB output with a Prescale and Time Constant value of two.

Writing to the Timer B Counter Register, Port 3, will set ZCB. Writing the Timer B control Register (Port 2) with the Stop bit (3) equal to zero will force ZCB to be a zero. An active RESET signal will also reset ZCB.

ZCB OUTPUT TIMING
Figure 5.0-5



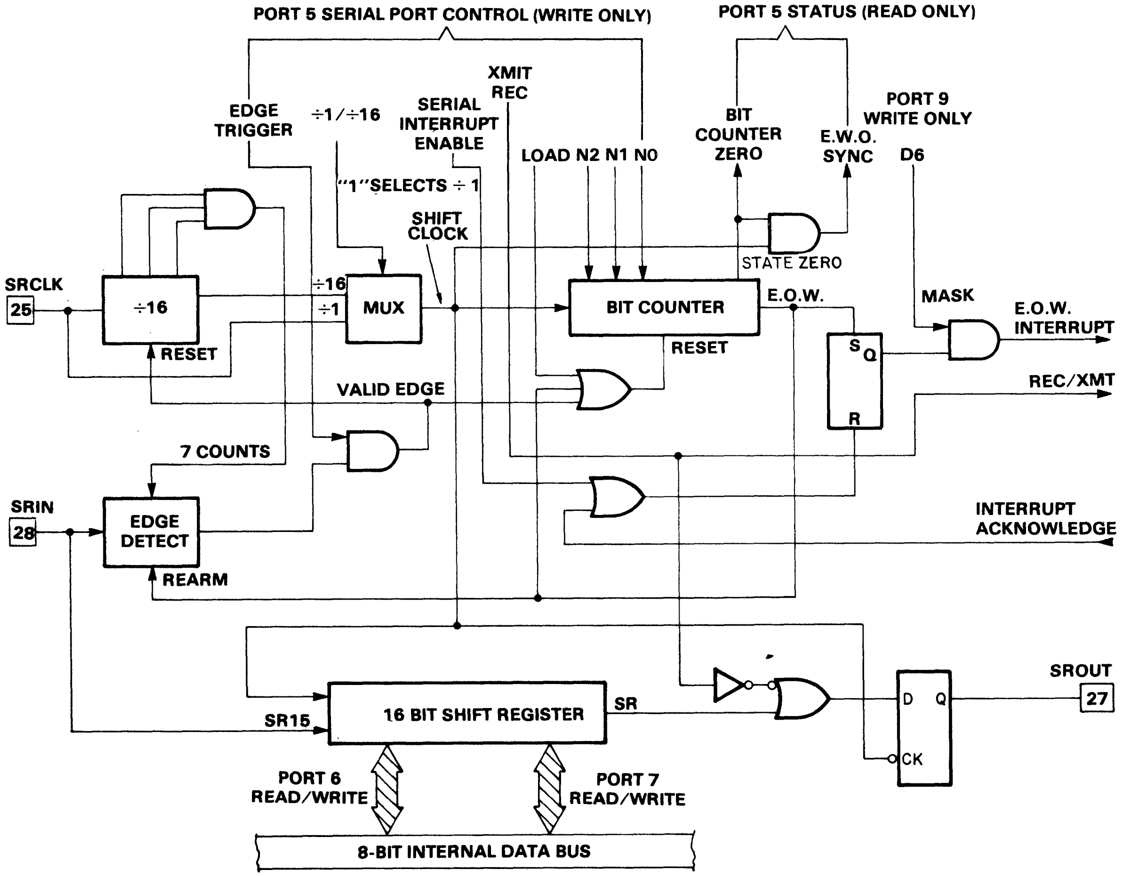
6.0 SERIAL I/O

6.1 OVERVIEW

The MK3886 contains circuitry to perform serial data transfers into and out of the unit under program control. The serial port will allow input and output of either asynchronous or synchronous serial data. The port consists of a 16-bit shift register that can be read from or written to while data is being shifted into or out of the shift register. This port, controlled by the CPU, can provide serial data communications or be used to interface external serial logic. The port uses 3 I/O pins to provide input (SRIN), output (SROUT), and clock (SRCLK) for the serial data.

The block diagram shown in Fig. 6.0-1 depicts the functional operation of the Serial Port. This diagram should be used as a reference throughout the remainder of the Serial Port description.

SERIAL PORT BLOCK DIAGRAM
Figure 6.0-1



6.1.1. SHIFT REGISTER

The Shift Register in the serial port is 16 bits long and can be read or written to at any time. It is addressed by the CPU via two ports that access two 8 bit bytes. The Upper Shift Register is defined as Port 6 and is used to access the most significant 8 bits. The Lower Shift Register is defined as Port 7 and is used to access the least significant 8 bits. Bit 0 is the least significant bit of the Shift Register. It is used to form the serial output, SROUT. Bit 15 is the most significant bit of the Shift Register. It is the bit position where serial input data first enters the Shift Register.

6.1.2. SHIFT CLOCK

The internal SHIFT Clock is used to clock data transfers into and out of the 16-bit Shift Register. It is also used to clock the internal Bit Counter register. In the receive mode, data is clocked into the most significant bit of the upper half of the shift register (Port 6) on the rising edge of the SHIFT clock. In the transmit mode, output data is transferred from the least significant bit of the lower half of the Shift Register to the output (SROUT) latch when the SHIFT clock is at a low level. The SHIFT clock is derived from the SRCLK input and is programmed to operate in a ÷1 mode or a ÷16 mode from the SRCLK input.

6.1.3 BIT COUNTER

The Bit Counter is initally reset to zero and remains in said state until the first bit of a new serial word is clocked into the MSB of the upper half of the shift register. The Bit Counter is then incremented as the successive bits are clocked into the shift register. Counting occurs on the rising edge of SHIFT. When the last bit of the programmed number of bits in the word is received (clocked into the shift register), the Bit Counter will reset itself to zero. An interrupt will be generated (if enabled) on the rising edge of SHIFT that resets the counter at the count of the specified number of bits per word. This allows the CPU to be interrupted at the word rate, not the bit rate. If the edge trigger mode is not selected, the Bit Counter will continue to count and interrupt regardless of whether new data is being input or output. Writing a new control word to the Serial Port automatically resets the Bit Counter to state zero.

6.1.4. EDGE DETECT CIRCUIT

The Edge Detect Circuit is used in conjunction with the asynchronous receive mode in order to prevent false start of word detection. This requires that the edge trigger circuit is enabled by bit D5 of Port 5 and that the ÷16 mode is selected.

When programmed, the ÷16 counter is reset and SHIFT clock is held low until a negative transition on the SRIN pin has occurred. After this edge has been detected, the circuitry will continue to hold SHIFT clock low and will sample the logic level on the SRIN input for 7 SRCLK pulses. If the SRIN input has remained low throughout this time, then the start-of-word is assumed to be valid and SHIFT goes high coincident with the rising edge of the 8th SRCLK pulse. The data on the SRIN input is then clocked into the shift register by the SHIFT clock in the middle of each bit time.

When the preprogrammed number of bits per word are counted by the bit counter, an End-of-Word interrupt occurs. At this time the ÷16 counter will be reset and the Edge Detector rearmed in preparation for the next character. This scheme provides reliable character synchronization with data sampling in the middle of each bit time.

6.1.5. CONTROL CIRCUITRY

The Control Circuitry provides correct timing and coordination of the functions within the Serial Port. Additionally, it is responsible for loading the Bit Counter and processing interrupts for the PORT. Programmer control is directed through Port 5, the Serial Port Control Register. Specifically, the register controls whether the serial port is in the transmit or receive mode, the ÷1 or ÷16 clock select, edge trigger, interrupt enable, and the number of shifts per word.

6.2 PROGRAMMING THE SERIAL PORT

6.2.1 SERIAL CONTROL REGISTER

Control of the Serial Port is provided via the Serial Control Register. This Register is designated as Port 5 and is WRITE only. The bit designation is as follows:

Port 5 - SERIAL CONTROL REGISTER

D7	D6	D5	D4	D3	D2	D1	D0
÷1 ÷16	XMIT RCV	EDGE Trigger	X	Serial Port Interrupt Enable	N2	N1	NO

Bit 7 - ÷1/-16 Select

When set, SRCLK is used as the internal SHIFT clock.

When cleared, SRCLK is divided by 16 to form the internal SHIFT CLOCK.

Bit 6 - XMIT/RCV

If set, the port is configured for the transmit mode.

When cleared, the port is configured for the receive mode.

Bit 5 - Edge Trigger

This bit is used primarily in the ÷16 receive mode for asynchronous data reception. When set, edge trigger mode is selected. When a negative transition is detected on the SRIN Input, the SHIFT CLOCK will begin to count. The state of SRIN is tested on the 7th clock pulse in order to assure a valid Start bit was detected. If the Start is valid, the Shift Register will be shifted on the 8th SRCLK pulse and subsequently every 16 pulses later until the preprogrammed shifts per word occur. At this time an End-of-Word interrupt flag occurs, and the Edge Detect circuit is rearmed.

If cleared the Edge Detect circuit is disabled. SRCLK is either divided by 1 or by 16 to generate SHIFT CLOCK and is not controlled by the edge detect circuit.

Bit 4 - Don't Care

Bit 3 - Serial Interrupt Enable

When set, an interrupt will be generated when the programmed number of shifts per word has occurred in either the transmit or receive mode. A different interrupt vector is generated for the transmit and receive modes.

When cleared, interrupts are disabled. Any pending interrupts will be cleared.

Bits 2, 1, 0 - N2,N1,NO

These bits determine the serial word size for both transmit and receive by the number of shifts per word. The values selected represent the shift configuration needed to interface most serial external logic and synchronous or asynchronous data communication channels. The bit designation is as follows:

N2	N1	N0	Shifts/Word
0	0	0	4
0	0	1	7
0	1	0	8
0	1	1	9
1	0	0	10
1	0	1	11
1	1	0	12
1	1	1	16

6.2.2. SHIFT REGISTER PROGRAMMING

The Shift Register is divided into two 8-bit ports. These bytes are accessed via Ports 6 and 7 which are the Upper and Lower Shift Register bytes respectively. Both ports are Read/Write. Bit designation for the two ports are as follows:

Port 6 - UPPER SHIFT REGISTER

D7	D6	D5	D4	D3	D2	D1	D0
SR15	SR14	SR13	SR12	SR11	SR10	SR9	SR8

Port 7 - LOWER SHIFT REGISTER

D7	D6	D5	D4	D3	D2	D1	D0
SR7	SR6	SR5	SR4	SR3	SR2	SR1	SR0

6.2.3. STATUS REGISTER

A register is provided on the MK3886 that provides information as to the status of particular functions within the chip. This register is Read Only and accessed as Port 5. The bit designation is as follows:

Port 5 - STATUS REGISTER

D7	D6	D5	D4	D3	D2	D1	D0
Bit Counter Zero	End of Word Sync	RAM Protect	X	X	INT2	INT1	INT0

Bit 7 - Bit Counter Zero

When set, this bit indicates the Bit Counter is in the zero state.

Bit 6 - End of Word Sync

When set, this bit indicates that the bit counter of the serial port is in the ZERO state AND the SHIFT clock is at a high level.

Bit 5 - RAM Protect

If set, it indicates the RAM write protection circuitry is enabled. Attempts made to write in the lowest 64 bytes (0-63) of RAM will be unsuccessful. There is no indication if an illegal write was attempted.

If clear, then all areas of RAM can be written to by the CPU.

Bits 2,1, and 0 External Interrupt State.

These bits indicate the current logic level being applied to External Interrupt inputs INT2, $\overline{\text{INT1}}$, and INT0.

6.3 ASYNCHRONOUS RECEIVE MODE

Figure 6.0-2 illustrates the timing for an example using the Serial Port in the asynchronous receive mode. For operation in this mode, the Serial Port Control Register should be programmed for Receive (XMIT/REC=0) and the Edge Trigger bit should be enabled. Also, the ÷16 mode should be selected since this also enables the Edge Detect circuitry. Upon selecting the ÷16 and Edge Trigger, both the Bit Counter and the ÷16 counter are reset and held low until a negative transition occurs on the SRIN pin. After a valid edge has been detected (see section 6.1.4), the SHIFT clock will go high and the bit count will begin at the eighth SRCLK pulse and will continue to clock every sixteenth clock pulse thereafter. When the programmed number of bits have been shifted in, the Bit Counter will be reset to zero and an End-of-Word interrupt will be generated. After the falling edge of SHIFT following the End-of-Word interrupt, the Edge Detect circuitry will be rearmed in preparation for the next word. Thus, if a start bit is present immediately following the time when the Edge Detect circuitry is rearmed, SHIFT clock will again go high approximately one bit-time after the rising edge of SHIFT which reset the Bit Counter and caused the End-of-Word interrupt. In other words, SHIFT can go high again on the eighth SRCLK pulse as soon as the Edge Detect circuitry is rearmed.

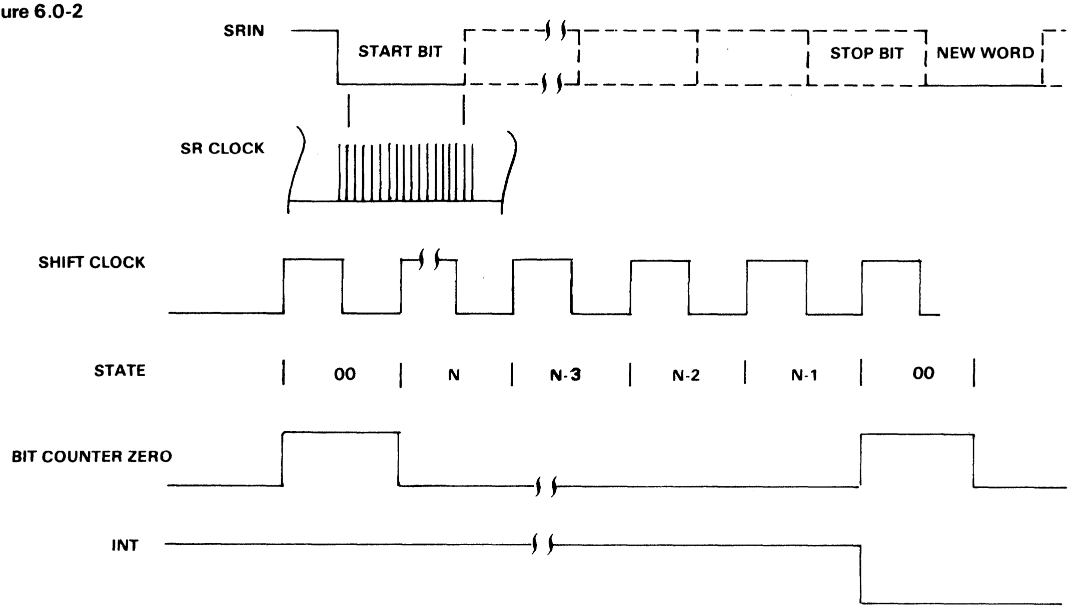
The Shift Register must be read before the next rising SHIFT CLOCK edge; otherwise, the data will be shifted to the right by one bit. Since data is gated directly on the data bus from the Shift Register, it must be read witin one bit time. For a 9600 bps data rate, this would require reading the Register within 104 microseconds from the time that the End-of-Word interrupt is generated.

It is possible to detect a receiver overrun condition by examining the state of the Bit Counter. Bit 7 of Port 5 is designated as Bit Counter Zero (BCZ). After a end of word receive interrupt is processed, the data in the shift register is transferred to processor. If Bit 7 is then read and is a logic 1 no overun error has occurred. If the bit is a logic 0 an error has occurred.

The example in Figure 6.0-2 shows the timing required for asynchronous data reception from a device such as a teletype. Within this data stream is start, stop, and data bits. A typical format requires 1 start bit, 8 data bits and 2 stop bits for a total of 11 bits. All these bits will be residing in the 16 bit shift register when the End of Word interrupt is generated. It is, therefore, necessary to strip the start and stop bits from the data. An example of this is shown in Section 9 concerning MK3886 Applications.

SERIAL PORT TIMING ASYNCHRONOUS RECEIVE

Figure 6.0-2



6.4 SYNCHRONOUS RECEIVE

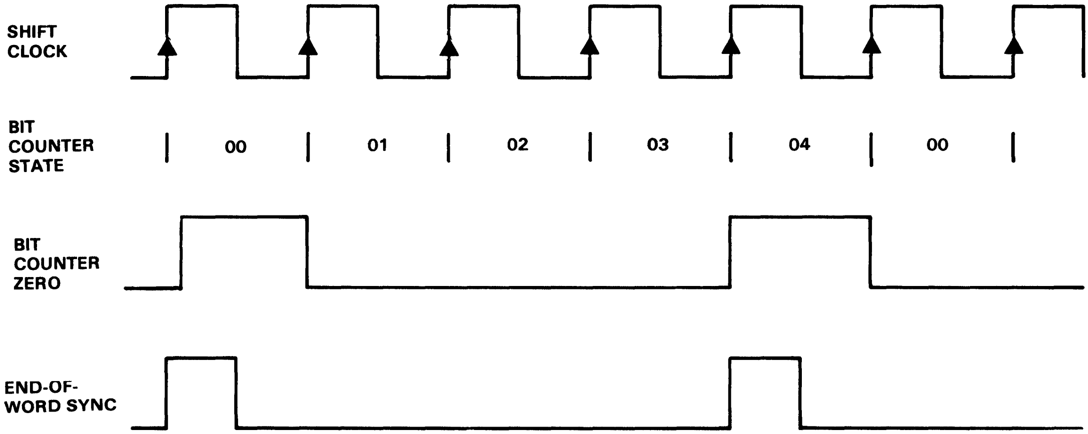
For synchronous operation, the Edge Trigger Mode must not be selected and bit 6 (XMIT/REC) should be a 0. Also, the ÷1 SHIFT Clock mode should be selected to establish the SRCLK input as the synchronization clock for the data stream. Once the Serial Port Control word is written to with Edge Trigger = 0 and the ÷1 SHIFT clock mode selected, then the Serial Port will continuously shift data into the MSB of the upper half of the Shift Register at the ÷1 clock rate and interrupt when the programmed number of bits have been shifted in.

An illustration of receive timing is shown in Figure 6.0-3. This diagram is for a ÷1, no edge triggered, sychronous receive sequence for a 5 bit word. Note the relationship of SHIFT clock, state of the bit counter, and Bit Counter Zero. Since Edge Trigger is not enabled (Port 5, bit 5=0) the Bit counter will continue to count and interrupt at the word rate regardless of whether or not new data is being received. Writing a new control word to the Serial Control Port automatically resets the bit counter to zero. Note that when the End-of-Word interrupt is received, the Shift Register must be read before the rising edge of the next SHIFT clock pulse or data will be shifted to the right. At 9600 bps this requires reading the Shift Register within 104 microseconds of the End-of-Word interrupt. As explained in the previous section on Asynchronous Operation, the Bit Counter zero flag in the Status Register can be used to verify the integrity of the data read from the shift register.

Writing to Port 5 resets the bit counter to zero.

SERIAL PORT SYNCHRONOUS RECEIVE TIMING

Figure 6.0-3



6.5 ASYNCHRONOUS OR SYNCHRONOUS TRANSMIT MODE

The Transmit mode is selected by setting bit 6 (XMIT/REC) of the Serial Port Control word. In the transmit mode data loaded into the Shift Register is enabled out of the Port 7 on the falling edge of SHIFT CLOCK with the least significant bit, SR0, first. Data from the internal data bus is loaded directly into the Shift Register whenever an output is done to either the Upper or Lower Shift Register Ports. A Status flag is also available to insure that data can be written to the port without an underrun condition occuring.

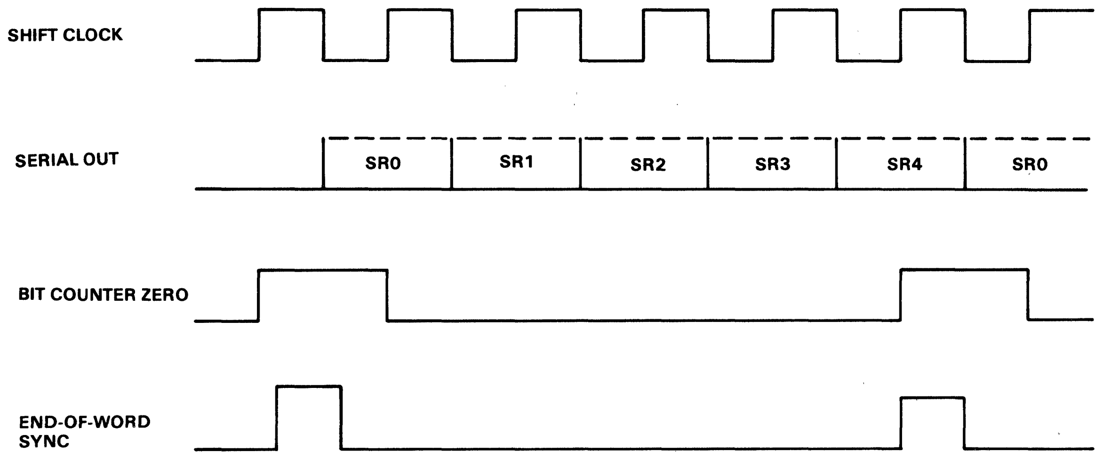
Figure 6.0-4 illustrates serial output timing in the ÷1 mode. This mode can be used for either asynchronous or synchronous data output. This figure illustrates timing relationships necessary to insure correct serial data output.

When the output operation is begun, the data within the shift register is shifted one bit position to the right on the falling edge of the SHIFT CLOCK. SR0, the least significant bit of Port 7, is output first. SR15, the most significant bit of Port 6, is output last. SR15 is output on the 16th SHIFT CLOCK pulse (assuming 16 shifts per word was selected). While the shift register contents are being output on a bit by bit basis, data is simultaneously sampled and input to the shift register through the SRIN pin.

Output data is transferred from SR0 to the output holding buffer when SHIFT CLOCK is low. Therefore to insure valid output data on the SROUT pin for a full bit time, the lower shift register (Port 7) must be written to while SHIFT CLOCK is at a high level. This condition is present when both Bit Counter Zero and SHIFT CLOCK are high. This can be verified by reading Bit 6 of Port 5, which is called End of Word Sync. The above condition must be met in order to assure continuous error free output data. If the Upper and Lower Shift Register Ports are both utilized and are loaded on opposite sides of the rising edge of SHIFT CLOCK, an extra bit will either be added or dropped from the output data stream.

SERIAL PORT TRANSMIT TIMING

Figure 6.0-4



For continous data output, the Shift Register must be loaded within 1/2 bit time of the End-of-Word interrupt. For a 9600 bps output rate, this requires loading the Shift Register within 52 microseconds of the rising edge of SHIFT that signals the End-of-Word condition.

For noncontinuous data output, as is used in Asynchronous data transmission, the user must still take the same precautions in loading the Shift Register to prevent an overrun or underrun condition as described above. One method that can be used to insure valid data out is to use the End-of-word interrupt in the transmit mode to signal the proper time to load the Shift Register. In using this method, the transmit mode would be set and the Serial Interrupt would always be enabled, so the the Serial Port would be continually shifting out data and interrupting at the specified word rate. This would occur regardless of whether or not data had been loaded into the Shift Register. Recall that while the Shift Register is in the Transmit Mode, the SRIN input is being sampled and shifted in. If, for example, it is desirable to have the SROUT Line at a marking condition (logic 1) between asynchronous serial words, then the user should insure that a logic one is present at the SRIN pin.

Another method for sending asynchronous serial data would be to specify edge trigger in a write operation to the Serial Port Control register. This would reset the internal ÷16 counter, thereby stopping the SHIFT clock. It would then be possible to load both halves of the Shift Register without the possibility of an overrun or underrun condition. This would be followed by a second control word specifying no Edge Trigger. This sequence will stop the output process, reload the Shift Register, reset the ÷16 counter, and then re-enable the ÷16 counter and SHIFT clock.

7.0 INTERRUPTS

7.1 OVERVIEW

A total of three external interrupt inputs are provided in the MK3886. Two of these inputs are uncommitted, the third is associated with Timer A. These inputs are edge triggered with Schmitt trigger inputs that allow slow rise time signals to be tied directly to the interrupt request lines. $\overline{\text{INT1}}$ will set an interrupt request latch on a negative edge transition whereas INT2 sets the latch on a positive edge. INTO can be programmed to interrupt on either edge transition.

The Combo's interrupt control logic insures that it acts in accordance with Z80 system interrupt protocol for nested priority interrupt and proper return from interrupt. The priority of any Z80 peripheral is determined by its physical location in a daisy chain configuration. Two signal lines (IEI and IEO) are provided on the Combo and all Z80 peripheral devices to form the system daisy chain. The device closest to the CPU has the highest priority; within the Combo, interrupt priority has been pre-determined for each interrupt source. The following is the internally assigned priority in decreasing order:

1. INT O
2. Timer A
3. Timer B
4. End of Word - Receive
5. End of Word - Transmit
6. $\overline{\text{INT1}}$
7. INT2

III
Z80 FAMILY
TECHNICAL
MANUALS

Each of the interrupt channels on the 3886 can be individually enabled and each provides a unique interrupt vector to the CPU. Additionally, a programmable mask register, accessed via Port 9, allows the interrupt requests on each channel to be selectively blocked without disabling them.

7.2 PROGRAMMING THE INTERRUPTS

The Combo chip can be programmed to request a CPU interrupt every time its interrupt request latch is set. Some time after the interrupt request, the CPU will respond with an interrupt acknowledge and the MK3886 will determine the highest priority channel which is requesting an interrupt within the device. Then if the MK3886's IEI input is active, indicating that it has priority within the system daisy chain, it will put an 8-bit word on the system data bus.

The Combo chip has been designed to operate in two different interrupt modes. These modes are RESTART and VECTOR. RESTART mode is provided for use with 8080 CPU's or the Z80 in Mode 0, VECTOR is used with Interrupt Mode 2 of the Z80 CPU.

If the MK3886 is in the RESTART mode upon receipt of a interrupt acknowledge, the Combo Chip forces a Z80 RESTART instruction on the data bus. The CPU strobes in RESTART and executes the instruction from one of eight locations as defined by the T field in the RESTART instruction. The RESTART locations are 0000, 0008H, ..., 0030H. The interrupt channel assignments specify which restart location is executed. The channel assignments are encoded into the instruction field. The RESTART vector is shown in the convention below.

RESTART

D7	D6	D5	D4	D3	D2	D1	DO
1	1	I2	I1	IO	1	1	1

Where:					RESTART
Z80 Mnemonic	I2	I1	I0	SOURCE	LOCATION
RST 56	1	1	1	INT0	38 H
RST 48	1	1	0	Timer A	30 H
RST 40	1	0	1	Timer B	28 H
RST 32	1	0	0	End of Word Receive	20 H
RST 24	0	1	1	End of Word Transmit	18 H
RST 16	0	1	0	$\overline{\text{INT1}}$	10 H
RST 08	0	0	1	INT2	08 H

The VECTOR mode is a more powerful method of servicing interrupts. When an interrupt request has been acknowledged and the Combo's IEI input is active, indicating it has priority within the system daisy chain, it will place an 8-bit interrupt vector on the system data bus. This interrupt vector is used to form a pointer to a location in memory where the address of the interrupt service routine is stored. The interrupt vector has the following format.

VECTOR

D7	D6	D5	D4	D3	D2	D1	D0
V7	V6	V5	V4	I2	I1	I0	0

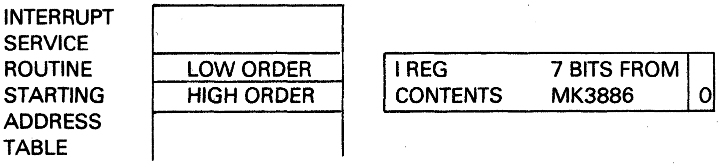
1	1	1	-	INT 0
1	1	0	-	Timer A
1	0	1	-	Timer B
1	0	0	-	E.O.W. Receive
0	1	1	-	E.O.W. Transmit
0	1	0	-	$\overline{\text{INT 1}}$
0	0	1	-	INT 2

The high-order 4 bits of the vector, V7 through V4, are program selectable and are stored in upper bits of Port 8. They should be written to the Combo Chip as part of the initialization process. The next three bits will be provided by the Combo's internal control logic as a binary code corresponding to the highest priority channel requesting an interrupt; finally the low-order bit of the VECTOR will always be zero.

The VECTOR represents the least significant 8 bits of a 16 bit interrupt vector. The most significant 8 bits are supplied by the I register within the Z80-CPU. This 16 bit vector points to an interrupt service routine starting address table. Thus in Mode 2, a single 8-bit vector stored in an interrupting MK3886 can result in an indirect call to any memory location.

MODE 2 INTERRUPT OPERATION

Desired starting address pointed to by:



7-3. LOADING THE INTERRUPT CONTROL AND VECTOR REGISTER

Before an interrupt sequence can begin, the Combo Chip must be programmed to initialize the desired response. Programming is done through two registers, the Interrupt Control and Vector Register and the Interrupt Mask Register. Port 8 is designated as the Interrupt Control and Vector register. The purpose of this register is to store the high order 4 bits of the Interrupt Vector and to provide interrupt control information. Port 9 is designated as the Interrupt Mask Register. The purpose of this register is to selectively block interrupting channels without disabling them.

Port 8 is the Interrupt Control Vector Register. To load this register, the CPU performs a normal I/O write sequence to the port address. This Port is WRITE only. The following diagram shows the position of each bit in the Interrupt Control and Vector Register.

D7	D6	D5	D4	D3	D2	D1	DO
V_7	V_6	V_5	V_4	RESTART/VECTOR	INT1 ENABLE	INT2 ENABLE	X

PORT 8

Bits D7-D4

These bits represent the high order four bits of the Interrupt Vector. When combined with the internal channel assignments, a complete 8 bit vector is specified.

Bit D3 Mode Select

When set the MK3886 will operate in the RESTART mode. This mode is directly compatible with the 8080A or Z80 Mode 0.

When cleared, the MK3886 will operate in vectored interrupt mode (Z80 mode 2).

Bits D2-D1

These bits are used to selectively enable or disable the two external interrupt inputs, $\overline{\text{INT1}}$ and INT2.

When set, they will generate an interrupt request sequence every time the inputs become active (if the appropriate mask bit is enabled).

Interrupt Mask Register

Port 9 is the Interrupt Mask Register. This register allows a mask word to be written to the port that will selectively allow certain channels to interrupt the CPU. This will block the interrupts but will not disable them. This operation can be thought of as a 2-input AND gate. One input of the gate is the mask bit while the other input is the latched interrupt active signal. The mask bit, therefore, determines whether the channel interrupt will be further processed and prioritized within the 3886. Any channel of the 3886 can be prevented from requesting an interrupt by loading a logic 0 in the respective bit position. The Port assignments are shown below.

INTERRUPT MASK REGISTER

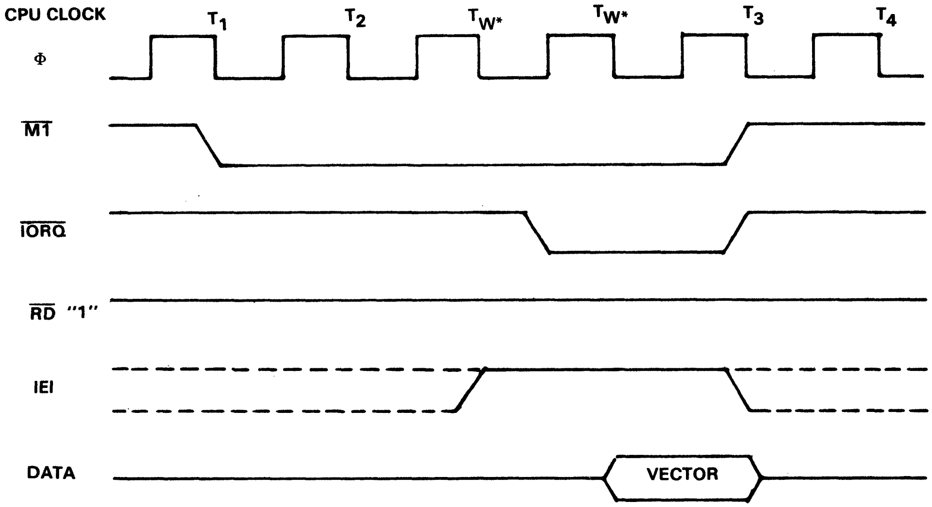
D7	D6	D5	D4	D3	D2	D1	DO
X	SERIAL PORT	TIMER B	TIMER A	INTO	$\overline{\text{INT1}}$	INT2	X

If an interrupt request has been made, and the respective mask bit disabled it will be passed on to the CPU interrupt circuitry when the mask bit is enabled. If this previous interrupt is not desired but future interrupts needed, the appropriate interrupt enable bit must be disabled, then set again. This is done using the respective interrupt enable bits designated in the specific channel's control register.

7.4 INTERRUPT ACKNOWLEDGE CYCLE

Figure 7.0-1 illustrates the timing associated with the Interrupt Acknowledge Cycle. Some time after an interrupt is requested by the Combo, the CPU will send out an interrupt acknowledge ($\overline{\text{M1}}$ and $\overline{\text{IORQ}}$). To insure that the daisy chain enable lines stabilize, channels are inhibited from changing their interrupt request status when $\overline{\text{M1}}$ is active. $\overline{\text{M1}}$ is active about two Φ clock cycles earlier than $\overline{\text{IORQ}}$, and $\overline{\text{RD}}$ is false to distinguish the cycle from an instruction fetch. During this time the interrupt logic of the Combo will determine the highest priority interrupting channel within the Combo places its Interrupt Vector onto the Data Bus when $\overline{\text{IORQ}}$ goes active. Two wait states (T_{W*}) are automatically inserted by the CPU at this time to allow the daisy chain to stabilize. Additional wait states may be added.

INTERRUPT ACKNOWLEDGE CYCLE
Figure 7.0-1



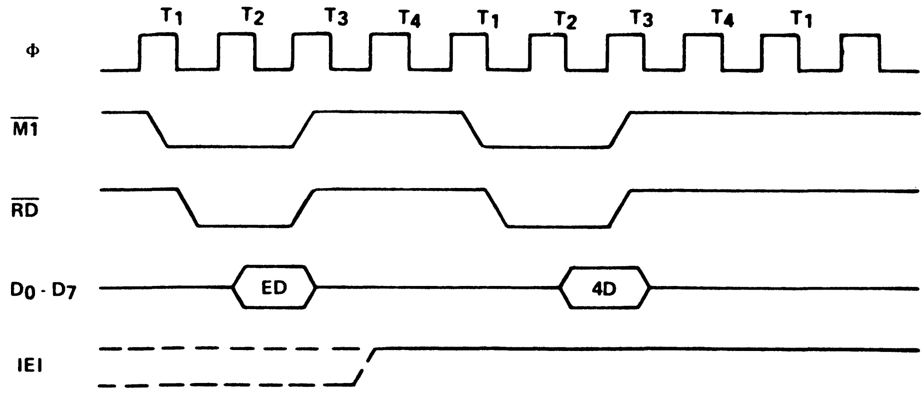
7.5 RETURN FROM INTERRUPT CYCLE

Unlike the other Z80 peripheral circuits, the MK3886 will not respond to the Z80 RETI command. This is due to the manner in which the IEO line in the daisy chain is manipulated by the 3886. The standard Z80 daisy chain protocol dictates that the closest device to the CPU has the highest priority. This priority is determined by the IEI and IEO signals. If a peripheral is granted an interrupt acknowledge, it will force IEO low thus blocking all other lower priority devices. The IEO line will stay low until the interrupt service routine is executed and the CPU issues the two byte RETI code. If IEI is high and IEO is low, the peripheral will decode the RETI instruction. The peripheral being serviced will be reinitialized and its IEO will become active (assuming no other pending interrupts by that device).

Operation with the MK3886 is slightly different. After an interrupt has been acknowledged and an 8-bit word gated onto the data bus by the MK3886, the IEO line will go back high unless another channel of the Combo is requesting an interrupt. Therefore it is now possible for lower priority peripheral circuits to interrupt the CPU. If a Combo interrupt occurred while a lower priority peripheral device was being serviced, a potential conflict could arise if an RETI was to terminate the

RETURN FROM INTERRUPT CYCLE

Figure 7.0-2



Combo's Interrupt Service Routine. Since the IEO line of the 3886 was brought high after the interrupt acknowledge, an RETI instruction issued by the CPU would be decoded and executed by the lower priority peripheral device. Therefore an RET command is the correct method of exiting a Combo Interrupt Service Routine.*

*Rev "C" of the MK3886 should be placed at the end of the daisy chain, otherwise other peripheral devices in the daisy chain might not decode the RETI instruction.

8.0 COMBO PROGRAMMING SUMMARY

The following is a summary of the programmable ports on the MK3886. It is provided for use as quick reference and review of the port channel assignments. Detailed information on programming is provided in the appropriate sections on channel operation (e.g. Timers, Serial Port, etc.)

The most significant digit, D7, is on the far left and least significant digit, DO, on the far right of the register representation. An X indicates an unused bit position.

Port 0 - Timer A Control - Write Only

÷ 20	÷5	÷2						
7	6	5	4	3	2	1	0 ←	Bit No.

Event Counter Mode ←	0	0	0
÷2 Prescale ←	0	0	1
÷5 Prescale ←	0	1	0
÷10 Prescale ←	0	1	1
÷20 Prescale ←	1	0	0
÷40 Prescale ←	1	0	1
÷100 Prescale ←	1	1	0
÷200 Prescale ←	1	1	1

0 → Start/Stop Timer
1 → Pulse Width/Interval Timer
2 → INTO Active Level
3 → INTO External Interrupt Enable
4 → Timer A Interrupt Enable

Port 1 - Timer A Time Constant Register - Write Only

TC7	TC6	TC5	TC4	TC3	TC2	TC1	TCO	
7	6	5	4	3	2	1	0 ←	Bit No.

PORT 1 - Timer A Down Counter - Read Only

7	6	5	4	3	2	1	0 ←	Bit No.

PORT 2 - Timer B Control - Write Only

PSI	PSO	X	X			X	X	
7	6	5	4	3	2	1	0 ←	Bit No.

÷2 Prescale — 0 0
÷5 Prescale — 0 1
÷10 Prescale — 1 0
÷20 Prescale — 1 1

2 — Timer B Interrupt Enable
3 — Start/Stop Timer B

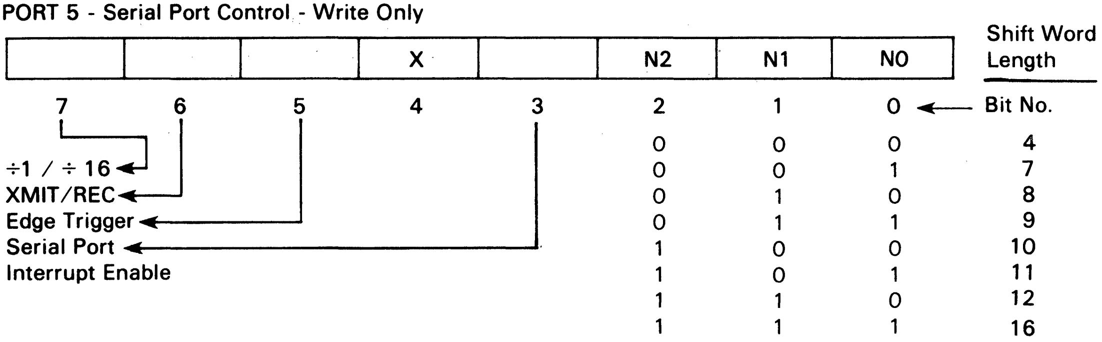
PORT 3 - Timer B Counter - Read/Write

7	6	5	4	3	2	1	0 ←	Bit No.

PORT 4 - Timer B Time Constant Register - Write Only

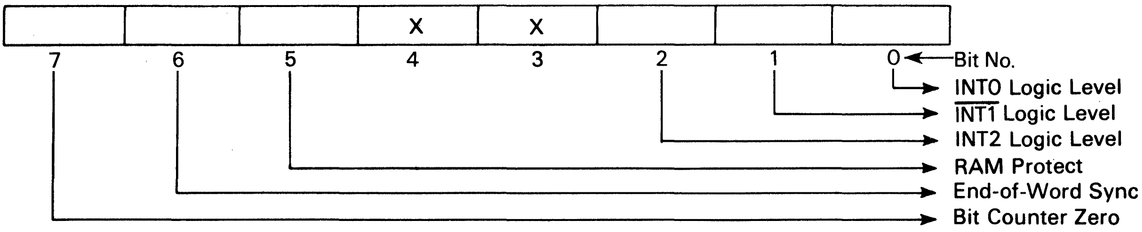
TC7	TC6	TC5	TC4	TC3	TC2	TC1	TCO	
7	6	5	4	3	2	1	0 ←	Bit No.

PORT 5 - Serial Port Control - Write Only



N2	N1	N0	Shift Word Length
0	0	0	4
0	0	1	7
0	1	0	8
0	1	1	9
1	0	0	10
1	0	1	11
1	1	0	12
1	1	1	16

PORT 5 - Status - Read Only



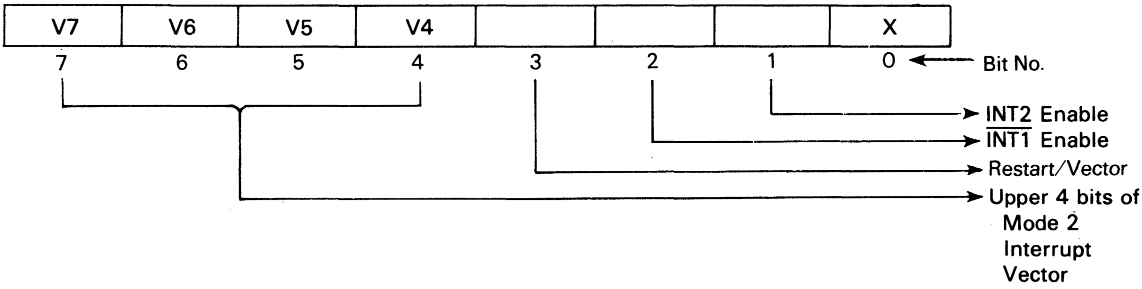
PORT 6 - Upper Shift Register - Read/Write

SR15	SR14	SR13	SR12	SR11	SR10	SR9	SR8

PORT 7 - Lower Shift Register - Read/Write

SR7	SR6	SR5	SR4	SR3	SR2	SR1	SR0

PORT 8 - Interrupt Control and Vector Register - Write Only


V7
V6
V5
V4
X
7
6
5
4
3
2
1
0
Bit No.
INT2 Enable
$\overline{\text{INT1}}$ Enable
Restart/Vector
Upper 4 bits of Mode 2 Interrupt Vector

If the Vector Mode is selected, the following interrupt vector is returned during the interrupt acknowledge cycle.

V7	V6	V5	V4	I2	I1	I0	0	
7	6	5	4	3	2	1	0	← Bit No.

Supplied from PORT 8 (bits 7–4)

1	1	1	0	- INT 0
1	1	0	0	- Timer A
1	0	1	0	- Timer B
1	0	0	0	- E.O.W. Receive
0	1	1	0	- E.O.W. Transmit
0	1	0	0	- $\overline{\text{INT 1}}$
0	0	1	0	- INT2

If the restart mode is selected, a RESTART instruction is returned to the CPU during the interrupt acknowledge cycle in the following format.

1	1	I2	I1	I0	1	1	1	
7	6	5	4	3	2		0	← Bit No.

Where:

Z80 Mnemonic	I2	I1	I0	SOURCE	RESTART LOCATION
RST 56	1	1	1	INTO	38H
RST 48	1	1	0	Timer A	30H
RST 40	1	0	1	Timer B	28H
RST 32	1	0	0	End of Word Receive	20H
RST 24	0	1	1	End of Word Transmit	18H
RST 16	0	1	0	$\overline{\text{INT1}}$	10H
RST 08	0	0	1	INT2	08H

PORT 9 - Interrupt Mask Register - Write Only

X							X	
7	6	5	4	3	2	1	0	← Bit No.

1	INT2
2	$\overline{\text{INT1}}$
3	INTO
4	Timer A Interrupt
5	Timer B Interrupt
6	Serial Port Interrupts

PORT A - Write Protect - Write Only

WP7	WP6	WP5	WP4	WP3	WP2	WP1	WPO

Write one byte: 55_H
Write Uninhibited: 66_H
Write Inhibit: Any value except 55_H or 66_H
All registers are cleared when $\overline{\text{RESET}}$ occurs.
The following is a list in descending order of the priority of each interrupting channel. The highest priority is first

1. INT 0
2. TIMER A
3. TIMER B
4. End of Word - Receive
5. End of Word - Transmit
6. $\overline{\text{INT 1}}$
7. INT 2

9.0 APPLICATIONS

The MK3886 has been designed to function in a variety of ways utilizing the power and flexibility of the functions on chip. The following sections illustrate hardware and software applications of this component.

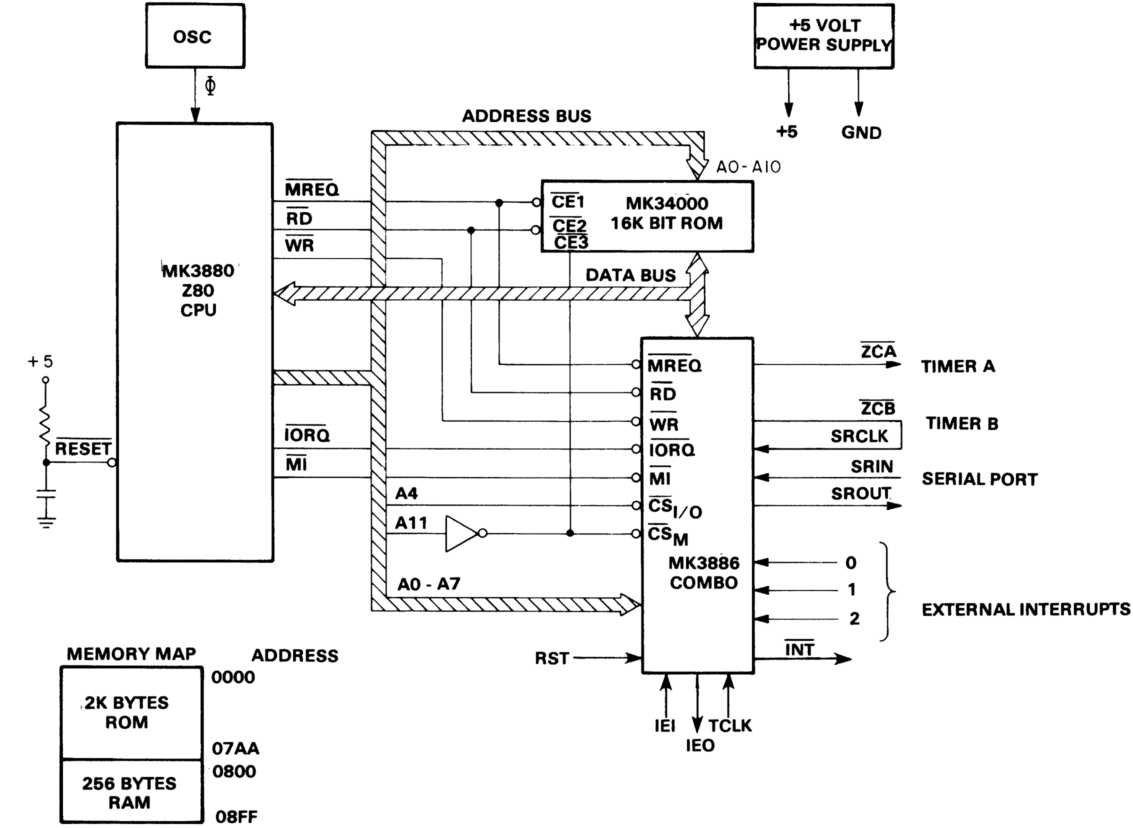
9.1 MINIMUM SYSTEM

Figure 9.0-1 illustrates a minimum Z80 system configuration. Four basic components are needed: the Oscillator, Z80 CPU, PROM or ROM, and the MK3886 Combo Chip. Since the Z80 CPU only requires a single 5-volt supply, most small systems can be implemented only using a single supply. Note that a powerful system can be implemented using only 3 LSI circuits, an oscillator, and a single 5-volt power supply.

This system configuration has 2K x 8 of ROM and 256 bytes of RAM. For this example, address bit A11 separates the ROM space from the RAM space so that it can be used for the chip select function. For larger amounts of external ROM or RAM, a simple TTL decoder would be required to form the chip selects. This memory configuration provides adequate storage space for programs plus providing space to allow scratchpad data storage and implementation of a "stack."

The system also has the Timer, Serial I/O and Interrupt capability of the Combo chip to allow an interface to an outside device. If additional parallel I/O were desired, TTL circuits could be added to accomplish this function. Input data could be gated onto the data bus using any standard tri-state driver while the output data could be latched with any type of standard TTL latch. As another alternative, Z80-PIO could be used to provide I/O capable of operating in an input only, output only, bidirectional or bit control mode.

MINIMUM Z80 SYSTEM
Figure 9.0-1



9.2 WRITE PROTECT

The lowest 64 bytes in the Combo's RAM are write protected. Three modes are available for dynamically configuring the write protect circuit. The configuration is done by writing to Port A, the Write Protect Port. These modes are Read Only, Read/Write and Read/Write one byte.

Upon power up or whenever a $\overline{\text{RESET}}$ occurs, the 64 bytes of RAM are in Read Only mode. Any attempt to Write to this portion of RAM will be inhibited within the MK3886. No error indication will be reported. The status of the Write Inhibit circuit can be determined by examining the RAM Protect bit in Port 5, the status Register. To examine this port, an input and bit test is required. If Bit 5 is set, write protection is enabled, if cleared it is disabled and normal Read/Write operations can continue. A typical sequence would be as follows:

```
IN      A,(5)       ;Input Status Register
BIT     5, A        ;Test Bit 5
JR      NZ, INH     ;Branch if Write Inhibited
                    ;Continue Processing if Read/Write
```

If the Write Protect circuit is enabled and it needs to be disabled for normal Read/Write operations; the following sequence would be used.

```
LD      A, 66H      ;Output Hex '66'
OUT     (OAH), A    ;for Read/Write
```

If only 1 byte is to be written to the memory the following sequence would be used:

```
LD      C,OAH       ;Specify Port to
LD      B,55H       ;Write one Byte
OUT     (C),B       ;Set Control Word
LD      (ADDR),A    ;Write one byte to RAM
EI                  ;Re-enable Interrupts
```

This sequence directs the RAM to write 1 byte and then re-enable the Write Protect circuitry. The control word to request this function is '55'H.

The interrupts are disabled in order to prevent any other write cycles from interrupt subroutines occurring before the Write operation to the RAM. In the example above, the contents of the Accumulator is written to the RAM memory. Also note that a 16 bit load instruction will not execute properly. An example of this would be if the LD (ADDR), A instruction was replaced with LD (ADDR), HL. The contents of the L register would be correctly loaded into ADDR but the H register would not be loaded into ADDR =1 since the write circuitry would now be inhibited for the second write operation.

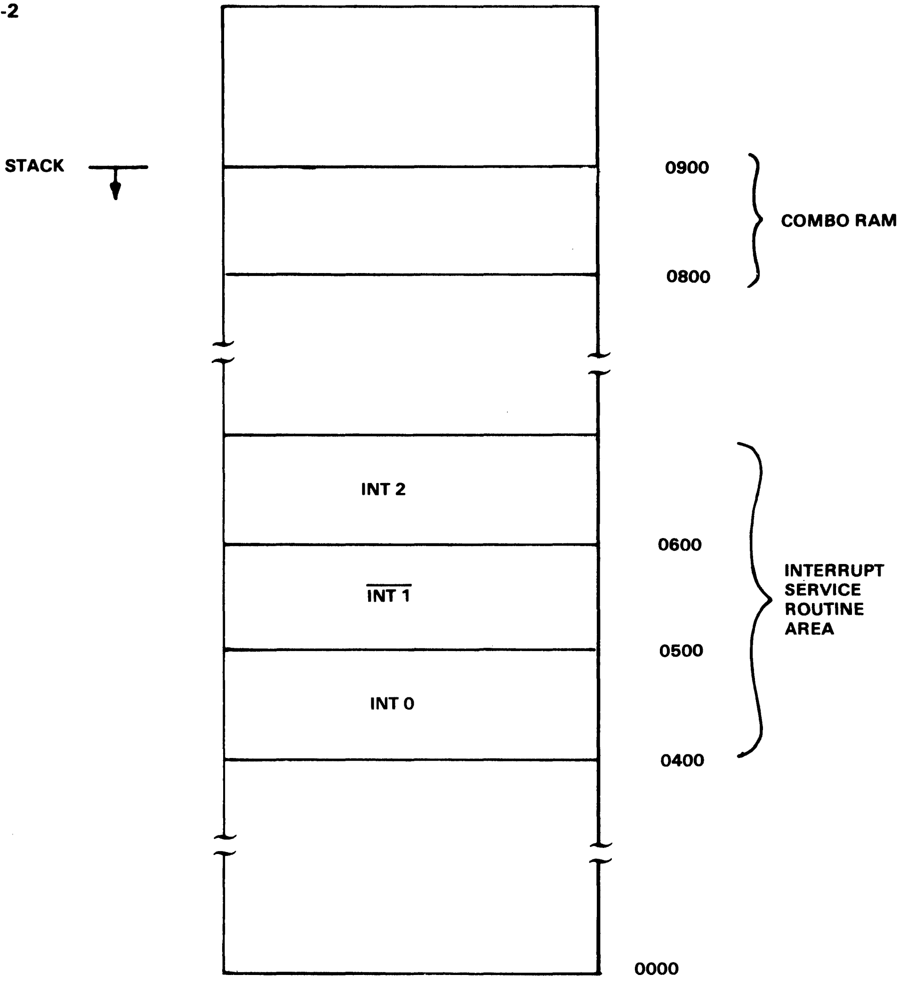
In order to enable the write protection circuit any value except 55H and 66H should be written to the Write Protect Port. For this example all zeros are used. The sequence would be as follows:

```
LD      A,0         ;clear accumulator
OUT     (OAH),A     ;Enable Write Protect Circuitry
```

9.3 INTERRUPT CONTROLLER

Three lines on the MK3886 are used to generate prioritized, vectored, maskable, edge triggered, edge selectable (INTO only), external inputs. The following example illustrates the steps necessary to set up the Z80 CPU to handle Mode 2 interrupts from this device. The memory map for the example is shown in Figure 9.0-2.

MEMORY MAP
Figure 9.0-2



The start bit must always be low. It lasts for 1 bit time and is used to indicate the beginning of a character. The data bits are the actual data character transferred. In this example the character is 8 bits long with the least significant bit being sent out or received first. After the data bits have been transmitted, 2 stop bits are sent.

In order to output asynchronous data, the following steps must be taken.

1. Load the start, data, and stop bits into the output port.
2. Start the shift register
3. Stop after receiving end-of-word transmit to reload more data + start bits + stop bits.

The following is an example routine to execute this task.

```
    INIT             SELECT
    EXX              ;ALT. REGS.
    LD HL,DATA       ;POINT TO DATA
    LD B, NUMBER     ;NUMBER OF BYTES
    EXX              ;MAIN REGS.
    LD A,40H         ;SERIAL INT.
    OUT (9),A        ;NOT MASKED
    LD A,4DH         ;INITIALIZE
    OUT (5),A        ;SERIAL PORT
    CALL XMIT        ;SEND 1st CHAR.
```

ASSUMED: Interrupt mode, Interrupt vector, Baud rate clock, etc. already initialized.

```
INTERRUPT
  BACK EX AF,AF'   ;RESTORE MAIN
       EXX         ;REGISTER SET
       EI          ;ENABLE INT.
       RET         ;BACK TO MAIN
```

ASSUMED: B'contains # of bytes to be transferred
HC' points to bytes to be transferred
Part is programmed in asynchronous serial transmit mode

The following program will allow the MK3886 to operate in asynchronous receive mode. It is assumed that the incoming character will consist of 8 bits per word and have two stop bits. However, a programming shortcut is used in this example to facilitate a more efficient algorithm.

The algorithm assumes that if the edge detect circuit begins correctly then a valid incoming message was being received. Using this assumption, then only 9 shifts per word are required. Shifting in the final 2 stop bits are meaningless since the MK3886 resynchronizes its edge detect circuits automatically whenever an end-of-word interrupt occurs. Therefore, upon the End-of-Word interrupt, the 8 bits of data are in the upper shift register and the start bit is in the most significant bit of the lower shift register. The data can be input directly to the CPU from the upper shift register and the start bit in the lower shift register simply ignored.

The following is an example of implementation of the asynchronous receive operation.

```
INIT    EXX                  ;SELECT ALT. REGS.
        LD B,NUMBER          ;NUMBER OF BYTES
        LD HL,DATA           ;POINT TO STORAGE
        EXX                  ;RESTORE MAIN REGIS.
        LD A,2BH             ;RX. MODE, :16CK,
        OUT (5),A            ;EDGE TRIG., INT. ENABLE
                             ;9 SHIFTS/WORD
```

The data can be input directly to the CPU from the upper shift register and the start bit in the lower shift register simply ignored.

The following is an example of implementation of the asynchronous receive operation.

```
INT       EXX                    ;SELECT ALT.
          LD B,NUMBER            ;NUMBER OF BYTES
          LD HL,DATA             ;POINT TO STORAGE
          EXX                    ;RESTORE MAIN REGIS.
          LD A,2BH               ;RX. MODE, :16 CK,
          OUT (5),A              ;EDGE TRIG., INT. ENABLE
                                 ;9 SHIFTS/WORD

RXINT     EX AF,AF'              ;SELECT ALTERNATE
          EXX                    ;REGISTER SET
          IN A,(6)               ;READ DATA WORD
          LD (HL)A               ;STORE WORD
          INC HL                 ;BUMP POINTER
          DJNZ BACK-$            ;RETURN IF B NOT 0
          LD A,23H               ;ELSE DISABLE
          OUT (5),A              ;RECEIVER
BACK      EX AF,AF'              ;RESTORE MAIN
          EXX                    ;REGISTER SET
          EI                     ;ENABLE INTERRUPT
          REI                    ;BACK TO MAIN

CPEX 1    DI                     ;DISABLE INTERRUPTS
          IM 2                   ;SELECT Z80 MODE 2 INTERRUPT
          LD HL,CRAM+255         ;ESTABLISH STACK
          LD SP,HL               ;AT TOP OF COMBO RAM
          LD HL,IVT              ;FETCH TABLE START ADDR.
          LD A,H                 ;SET UPPER 8 BITS
          LD I,A                 ;OF INT. VECT. TABLE
          LD A,L                 ;SET LOWER 8 BITS
```

The Z80 software routine will perform the following functions in this example:

(1) Select the Vector Interrupt Mode
(2) Establish the Stack at location 'FFFF'H
(3) Create the interrupt vector by loading the I and V registers in CPU and Combo respectively
(4) Enable INT 0 - 3 interrupt enable and mask register
(5) Specify the starting address of each INT service routine at each interrupt vector.
(6) Enable system interrupts

The following routine will perform the above tasks tne ready the Z80 CPU for interrupt from the MK3886. Remember that an RET, not an RETI, is the correct instruction to be executed upon completion of an interrupt service routine.

MOSTEK MACRO-80 ASSEMBLER V2.2

```
LOC    OBJ.CODE   STMT-NR          SOURCE-STMT PASS2 CPEX1  CPEX1  CPEX1  REL
                     1             NAME     CPEX1
                            ; COMBO PROGRAMMING EXAMPLE
                            ;
                            ; THIS ROUTINE ENABLES THE THREE EXTERNAL INTERRUPTS
                            ; ON THE COMBO CHOP. ALL OTHER INTERRUPTS ARE LEFT
                            ; DISABLED.
                            ;
                            ; EQUATES
         =0800       9      CRAM   EQU      0800H         ; START OF COMBO RAM
         =0000      10      CPORT  EQU      0             ; COMBO PORT MAP START ADDR.
         =0200      11      INTO   DEFL     0             ; INTO SERVICE ROUTINE START
         =0300      12      INT1   DEFL     02100H        ; INT1 SERVICE ROUTINE START
         =0400      13      INT2   DEFL     01200H        ; INT2 SERVICE ROUTINE START

0100                15             ORG      CRAM
                            ; INTERRUPT VECTOR TABLE
0100                17      IVT    DEFS     2             RESERVE FOR FUTURE VECTOR
0102   0004         18             DEFW     INT2          ; INT2 VECTOR
0104   0003         19      DEFW   INT1     ; INT1 VECTOR
0106                20             DEFS     8             ; RESERVE: OTHER COMBO VECT.
010E   0002         21             DEFW     INTO          ; INTO VECTOR
                            ;
0000                23             ORG      0
                            ; INITIALIZATION ROUTINE
0000   F3           25             DI                     ; DISABLE INTERRUPTS
0001   ED5E         26             IM       2             ; SELECT Z80 MODE 2 INTERRUPT
0003   3E01         27             LD       A,01H         ; SET UPPER 8 BITS
0005   ED47         28             LD       I,A           ;   OF INTERRUPT VECTOR TABLE
0007   21FF08       29             LD       HL,CRAM+255   ; ESTABLISH STACK
000A   F9           30             LD       SP,HL         ;   AT TOP OF COMBO RAM
000B   3E06         31             LD       A,06          ; LOAD VECTOR AND
000D   D306         32             OUT      (CPORT+6),A   ;   ENABLE INT 1 + 2
000F   3E0C         33             LD       A,0CH
0011   D300         34             OUT      (CPORT+0),A   ; ENABLE INTO, ACT. LEV. = 1
0013   3E00         35             LD       A,0           ; DISABLE
0015   D302         36             OUT      (CPORT+2),A   ;   OTHER
0017   D305         37             OUT      (CPORT+5),A   ;     COMBO INTERRUPTS
0019   FB           38             EI                     ; ENAB
```

```
                CPEX1
        ; INITIALIZATION ROUTINE
                EI                        ; DISABLE INTERRUPTS
                IM      2                 ; SELECT Z80 MODE INTERRUPTS
                LD      HL,CRAM+255       ; ESTABLISH STACK
                LD      SP,HL             ;  AT TOP OF COMBO RAM
                LD      HL,IVT            ; FETCH TABLE START ADDR.
                LD      A,H               ; SET UPPER 8 BITS
                LD      I,A               ;    OF INT. VECT. TABLE
                LD      A,L               ; SET LOWER 8 BITS
                AND     0F0H              ; MASK LOWER 4 BITS
                OR      07H               ; SELECT VECTOR MODE, ENABLE INT 1+2
                OUT     (CPORT+8),A       ; SET INT VECT + CTL REG
                LD      A,0CH
                OUT
```

9.4 TIMER CONSTANTS FOR USE AS BAUD RATE GENERATOR

The ZCB output of Timer B can be used to generate square waves that can be utilized as the clock input for the serial port. The following table gives a list of prescale values and time constant values for Timer B necessary to generate the correct frequencies for standard baud rates. The data are tabulated for use in the ÷16 modes by the serial port. These calculations are based upon a clock frequency of 2.458 MHz. This frequency was selected since it is a multiple of many baud rate frequencies as well as being close to the maximum frequency for the standard 2.5 MHz Z80 chip set. The period of clock was determined by using the following formula for the ACB output period:

$$2 * P^{*}\, t_C * TC$$

Where P is the Prescale factor, t_C is the period of the Timer clock TCLK, TC is the Timer B Time Constant data word loaded in Port 4, and 2 is a constant to adjust for the toggle output feature on ZCB.

			ACTUAL	DESIRED
			BAUD RATE	BAUD RATE
BAUD RATE	PRESCALE	TIME CONSTANT	CLOCK FREQ.	CLOCK FREQ.
110	5	139	1769	1760
300	2	128	4801	4800
600	2	64	9602	9600
1200	2	32	19203	19200
2400	2	16	38406	38400
4800	2	8	76813	76800
9600	2	4	153625	153600

9.5 TIMER B AS A VARIABLE PULSE WIDTH TIMER

Timer B can be used to produce asymetrical square wave outputs. Using Port 4, the Timer B Time Constant Register, a pulse train of variable pulse widths and interpulse intervals can be established. This simply requires reloading the time constant register with alternating values on each occurance that Timer B counts to zero. For example, consider a system using a 2.5MHz Time Clock. A continuing pulse train with a 1 millisecond pulse width and a 2 millisecond interpulse interval is desired. To obtain this output, the Prescaler would be loaded to divide by 20 and the Time Constant Register alternately loaded with 125 and 250. An example of this program is illustrated below:

The program is divided into two parts: Initialization and Interrupt Service Routine. The Initialization routine assumes that the interrupt vector location was programmed and the stack area has been established. Basically this routine sets up the Timer, conditions the ZCB output, and starts the timer. The interrupt service routine tests to see if 125 or 250 was last loaded into the Timer Constant Register. The alternate value is then loaded into that register and then the subroutine is exited.

```
CPEX2

; INITIALIZATION

DI                    ; DISABLE INTERRUPTS
IM 2                  ; Z80 MODE 2 INTERRUPT
LD HL,IVT             ; FETCH TABLE START ADDR
LD A,H                ; SET UPPER 8 BITS
LD I,A                ; OF INTERRUPT VECTOR TABLE
LD A,L                ; SET LOWER 8 BITS &
OUT (CPORT+8),A       ; DISABLE INT1 & 2
LD HL,CRAM+255        ; ESTABLISH STACK
LD SP,HL              ; AT TOP OF COMBO RAM
LD A,0
OUT (CPORT +0),A      ; DISABLE TIMER A & INT0
OUT (CPORT+5),A       ; DISABLE SERIAL PORT
LD A,010H
OUT (CPORT +9),A      ; UNMASK TIM. B, MASK REST
LD A,T1
OUT (CPORT+),A        ; START T1 HIGH PULSE,ZCB=1
LD A,0C0H
OUT (CPORT+2),A       ; TIM. B ENABLE, PRESCALE=20
LD A,T2
OUT (CPORT+4),A       ; LOAD NEXT TC FOR LOW PULSE
LD (TCVAL),A          ; SAVE NEXT TC
```

MOSTEK MACRO-80 ASSEMBLER V2.2

```
LOC   OBJ. CODE   STMT-NR SOURCE-STMT PASS2  CPEX2   CPEX2   CPEX2  REL
                      1                  NAME     CPEX2
                             ; COMBO PROGRAMMING EXAMPLE
                             ; THIS ROUTINE PROGRAMS TIMER B AS AN ASYMETRICAL
                             ; PULSE GENERATOR. ALL OTHER INTERRUPTS ARE DISABLED.
                             ;
                             ; EQUATES
        =0400         7      CRAM    EQU      0800H          ; START ADDR. OF COMBO RAM
        =0000         8      CPORT   EQU      0              ; START ADDR. OF COMBO PORTS
        =007D         9      T1      EQU      125            ; TC FOR HIGH T1 = 1 MS.
        =00FA        10      T2      EQU      250            ; TC FOR LOW T2 = 2 MS
                             ;
0400                 12              ORG      CRAM
0400'                13                       1
                             ;
                             ; INTERRUPT VECTOR TABLE
0100                 17      IVT     DEFS     2              ; RESERVE FOR FUTURE VECTOR
0102                 18              DEFS     8              ; RESERVE; OTHER COMBO VECT.
010A    0002'        19              DEFW     TBI            ; TIMER B INT. START ADDR.
010C                 20              DEFS     4              ; RESERVE: OTHER COMBO VECT.
                             TCVAL   DEFS     1              ; CURRENT TIME CONSTANT VAC.
0000                 22              ORG      0
                             ; INITIALIZATION
0000    F3           24              EI                      ; DISABLE INTERRUPTS
0001    ED5E         25              IM       2              ; Z80 MODE 2 INTERRUPT
0003    3E01         26              LD       HA,01          ; SET UPPER 8 BITS
0005    ED47         27              LD       I,A            ;  OF INT. VECTOR TABLE
0007    21FF04       28              LD       HL,CRAM+255    ; ESTABLISH STACK
000A    F9           29              LD       SP,HL          ;  AT TOP OF COMBO RAM
000B    3E08         30              LD       A,08H          ; SET INTERRUPT VECTOR &
000D    D308         31              OUT      (CPORT+8),A    ;  DISABLE INT1 & 2
```

```
000F     3E00        32                         LD      A,0
0011     D300        33                         OUT     (CPORT+0),A    ; DISABLE TIMER A & INTO
0013     D305        34                         OUT     (CPORT+5),A    ; DISABLE SERIAL PORT
0015     3E10        35                         LD      A,010H
0017     D309        36                         OUT     (CPORT+9),A    ; UNMASK TIM. B, MASK REST
001F     D302        40                         OUT     (CPORT+2),A    ; TIM. B ENABLE, PRESCALE 20
0021     3EFA        41                         LD      A,T2
0023     D304        42                         OUT     (CPORT+4),A    ; LOAD TC FOR LOW PULSE LENGTH
0025     FB          43                         EI
                      ;
                      ;
                      ;
```

MOSTEK MACRO-80 ASSEMBLER V2.2

```
LOC    OBJ. CODE    STMT-NR  SOURCE-STMT   PASS2  CPEX2   CPEX2    CPEX2  REL

0200                 48                     ORG     020H
                                      ; TIMER B INTERRUPT SERVICE ROUTINE
0200'    F5          50              TBI    PUSH    AF             ; SAVE ACCUMULATOR, FLAGS
0201     E5          51                     PUSH    HL             ; SAVE HL REGISTER PAIR
0202     210004'     52                     LD      HL,TCVAL       ; POINT TO CURRENT TC VALUE
0205     3E7D        53                     LD      A,T1           ; GET T1 VALUE
0207     BE          54                     CP      (HL)           ; COMPARE TO TC VALUE
0208     2002        55                     JR      NZ,TBI1-$      ; IF NOT =, LOAD T1 INTO TC
020A     3EFA        36                     LD      A,T2           ; OTHERWISE LOAD T2 INTO TC
020A     3EFA        36                     LD      A,T2           ; OTHERWISE LOAD T2 INTO TC
020C'    D304        57              TBI1   OUT     (CPORT+4),A    ; LOAD TIME CONSTANT
020E     320004'     58                     LD      (TCVAL),A      ; STORE CURRENT TC IN BUFFER
0211     F1          59                     POP     AF             ; RESTORE
0212     E1          60                     POP     HL             ;      CPU REGISTERS
0213     FB          61                     EI                     ; ENABLE INTERRUPTS
0214     C9          62                     RET                    ; RETURN
```

9.6. SERIAL ASYNCHRONOUS I/O

The MK3886 is capable of half duplex serial asynchronous transmission and reception. Since this port functions basically as a serial to parallel and parallel to serial converter, some software is needed to add or strip the start and stop bits and to align the data word. The following example illustrates a program sequence necessary to output an 8-bit ASCII word. It is assumed that the input clock frequency has been generated and is input at SRCLK. An 8-bit asynchronous data word has the following format.

START	1	2	3	4	5	6	7	8	STOP	STOP

The start bit must always be low. It lasts for 1 bit time and is used to indicate the beginning of a character. The data bits are the actual data character transferred. In this example the character is 8 bits long with the least significant bit being sent out or received first. After the data bits have been transmitted, a stop bit is sent. During this period the stop bits are high for 1 or 2-bit times allowing for both the transmit and receive terminals to asynchronize.

In order to output asynchronous data, the following steps must be taken.

1. Load the start, data and stop bits into the output port.
2. Start the shift register
3. Stop after receiving end-of-word transmit to reload more data

The following is an example routine to execute this task.

```
INIT                    SELECT
EXX                     ;ALT. REGS.
LD HL,DATA              ;POINT TO DATA
LD B, NUMBER            ;NUMBER OF BYTES
EXX                     ;MAIN REGS.
LD A,40H                ;SERIAL INT.
OUT (9),A               ;NOT MASKED
LD A,4DH                ;INITIALIZE
OUT (5),A               ;SERIAL PORT
CALL XMIT               ;SEND 1st CHAR.
```

ASSUMED: Interrupt mode, Interrupt vector, Baud rate clock, etc. already initialized.

```
INTERRUPT

XMIT EX AF,AF'          ;SELECT ALT.
EXX                     ;REGISTERS
LD A,(HL)               ;GET DATA
SLA A                   ;MAKE START BIT
OUT (7),A               ;OUT TO S.R. LO
LD A,OFFH               ;MAKE STOP BITS
RL A                    ;RESTORE DATA M.S.B.
OUT (6),A               ;OUT TO S.R. HI
INC HL                  ;BUMP POINTER
DJNZ BACK-$             ;JUMP IF MORE DATA
LD A,25H                ;ELSE DISABLE
OUT (5),A               ;TRANSMITTER

BAC, EX AF,AF'          ;RESTORE MAIN
EXX                     ;REGISTER SET
EI                      ;ENABLE INT.
RET                     ;BACK TO MAIN
```

ASSUMED: B' contains # of bytes to be transferred
HL' points to bytes to be transferred

Part is programmed in asynchronous serial transmit mode

The following program will allow the 3886 to operate in asynchronous receive mode. It is assumed that the incoming character will consist of 8 bits per word and have two stop bits. However, a programming shortcut is used in this example to facilitate a more efficient algorithm.

The algorithm assumes that if the edge detect circuit begins correctly then a valid incoming message was being received. Using this assumption, then only 9 shifts per word are required. Shifting in the final 2 stop bits are meaning less since the 3886 resynchronizes its edge detect circuits automatically whenever an end-of-word interrupts occurs. Therefore, upon the End-of-Word interrupt, the 8 bits of data are in the upper shift register and the start bit is in the most significant bit of the lower shift register. The data can be input directly to the CPU from the upper shift register and the start bit in the lower shift register simply ignored.

The following is an example of implementation of the asynchronous receive operation.

```
INIT      EXX               ;SELECT ALT. REGS.
          LD B,NUMBER       ;NUMBER OF BYTES
          LD HL,DATA        ;POINT TO STORAGE
          EXX               ;RESTORE MAIN REGIS.
          LD A,2BH          ;RX. MODE, :16 CK,
          OUT (5),A         ;EDGE TRIG., INT. ENABLE
                            ;9 SHIFTS/WORD

RXINT     EXX AF,AF'        ;SELECT ALTERNATE
          EXX               ;REGISTER SET
          IN A,(6)          ;READ DATA WORD
          LD (HL), A        ;STORE WORD
          INC HL            ;BUMP POINTER
          DJNZ BACK-$       ;RETURN IF B NOT 0
          LD A,23H          ;ELSE DISABLE
          OUT (5),A         ;RECEIVER
BACK      EX AF,AF'         ;REGISTER MAIN
          EXX               ;REGISTER SET
          EI                ;ENABLE INTERRUPT
          RET               ;BACK TO MAIN
```

10.0 ABSOLUTE MAXIMUM RATINGS

Temperature Under Bias	Specified Operating Range
Storage Temperature	-65°C to + 150°C
Voltage on Any Pin With Respect to Ground	-0.3V to +7V
Power Dissipation	0.8W

10.1 D.C. CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = 5V ± 5% unless otherwise specified

SYM	PARAMETERS	MIN	MAX	UNIT	TEST CONDITION
V_{ILC}	Clock Input Low Voltage	-0.3	0.8	V	
V_{IHC}	Clock Input High Voltage (1)	2.2	VCC'+.3	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{IH}	Input High Voltage	2.0	VCC	V	
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 2.0 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -250 μA
I_{CC}	Power Supply Current		120	mA	Outputs Open
I_{LI}	Input Leakage Current		± 10	μA	V_{IN} = 0 to V_{CC}
I_{LOH}	Tri-State Output Leakage Current In Float		10	μA	V_{OUT} = 2.4 to V_{CC}
I_{LOL}	Tri-State Output Leakage Current In Float		-10	μA	V_{OUT} = 0.4V
V_{SB}	Standby V_{CC} for RAM a	3.2	5.5	V	
I_{SB}	Standby Current		6.0	mA	V_{RAM} = 5.5V
			3.7	mA	V_{RAM} = 3.2V

10.2 CAPACITANCE

T_A = 25°C,

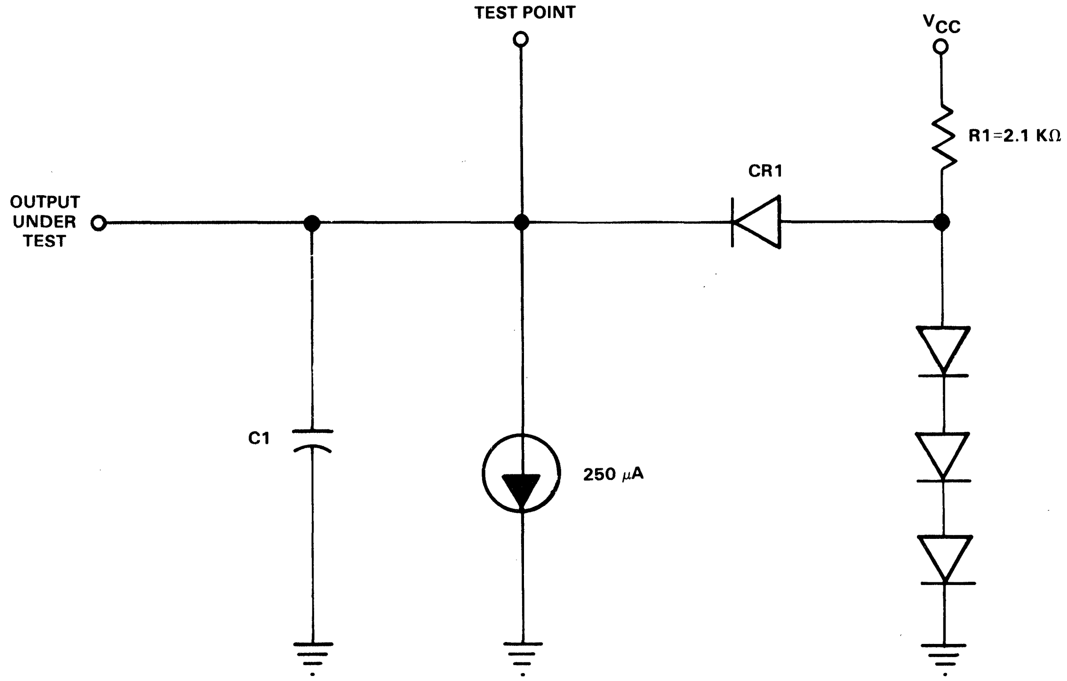
SYM	PARAMETER	MAX	UNIT	TEST CONDITION
C_{IN}	Input Capacitance	5	pF	Unmeasured Pins Returned to Ground
C_{OUT}	Output Capacitance	10	pF	

*COMMENT

Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

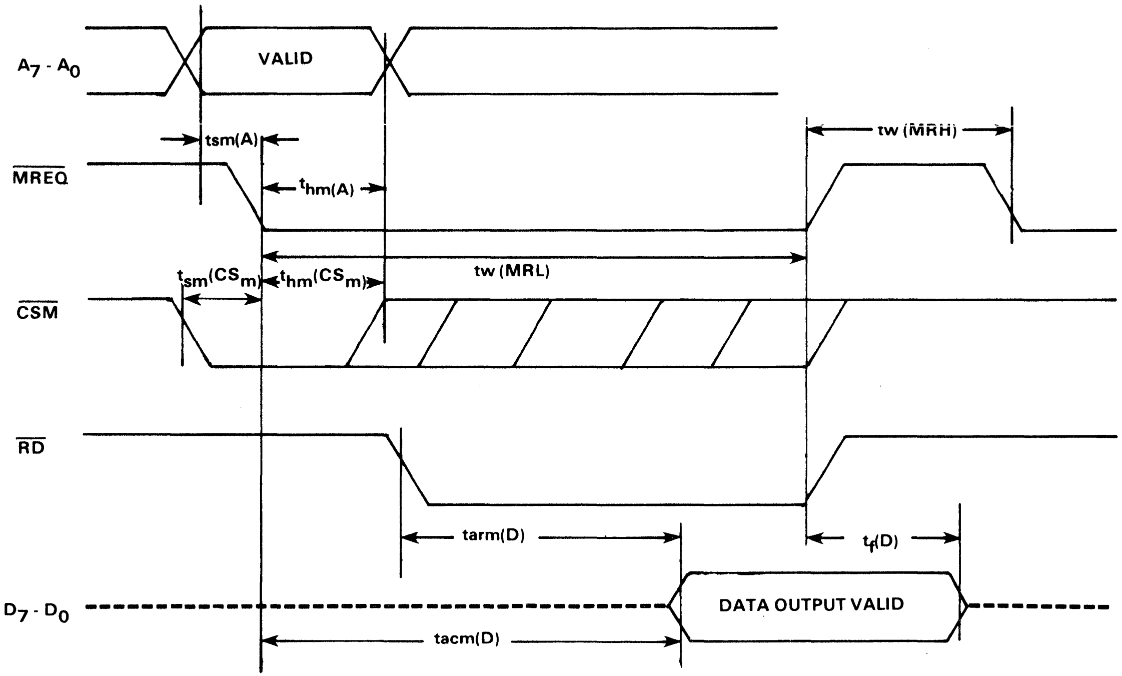
LOAD CIRCUIT FOR OUTPUT

Figure 10.0-1



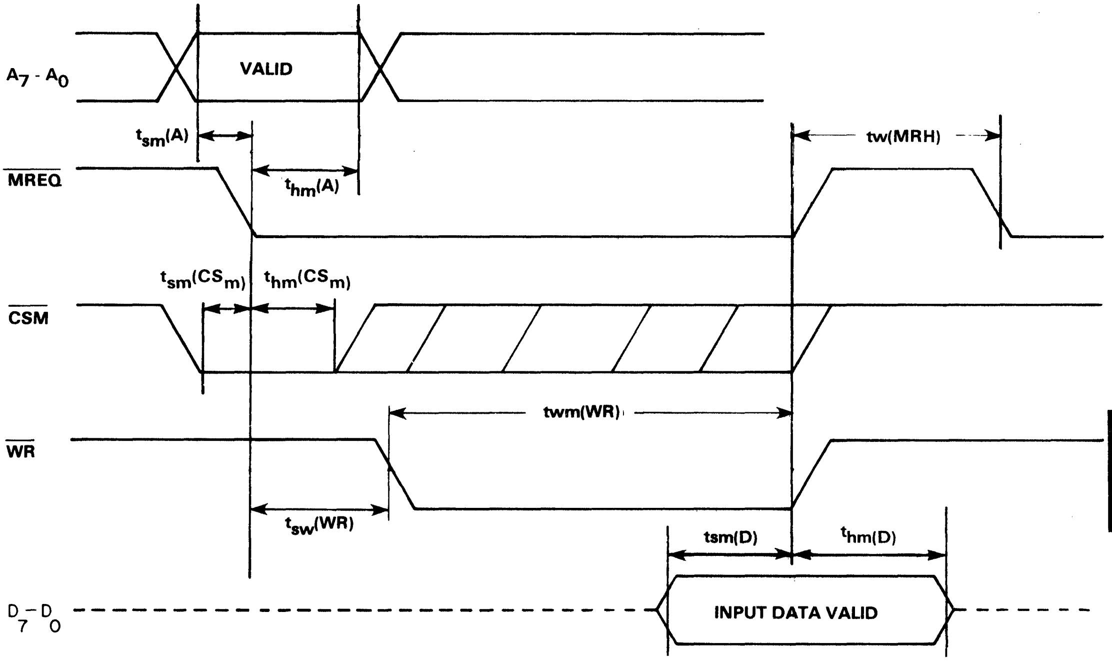
MEMORY READ TIMING

Figure 10.0-2a



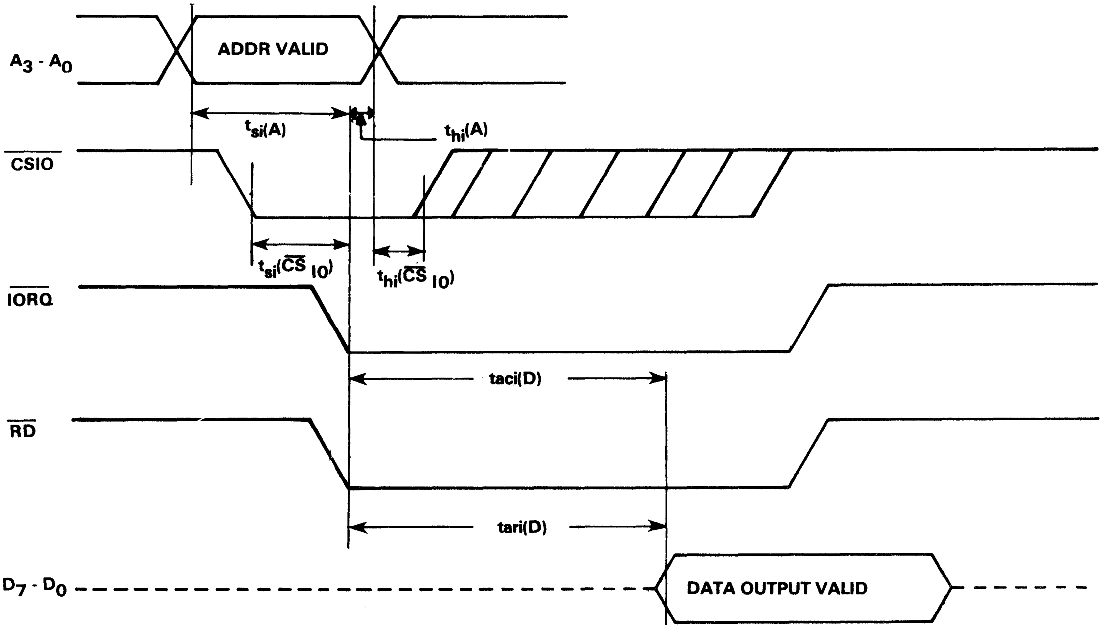
MEMORY WRITE TIMING

Figure 10.0-2b



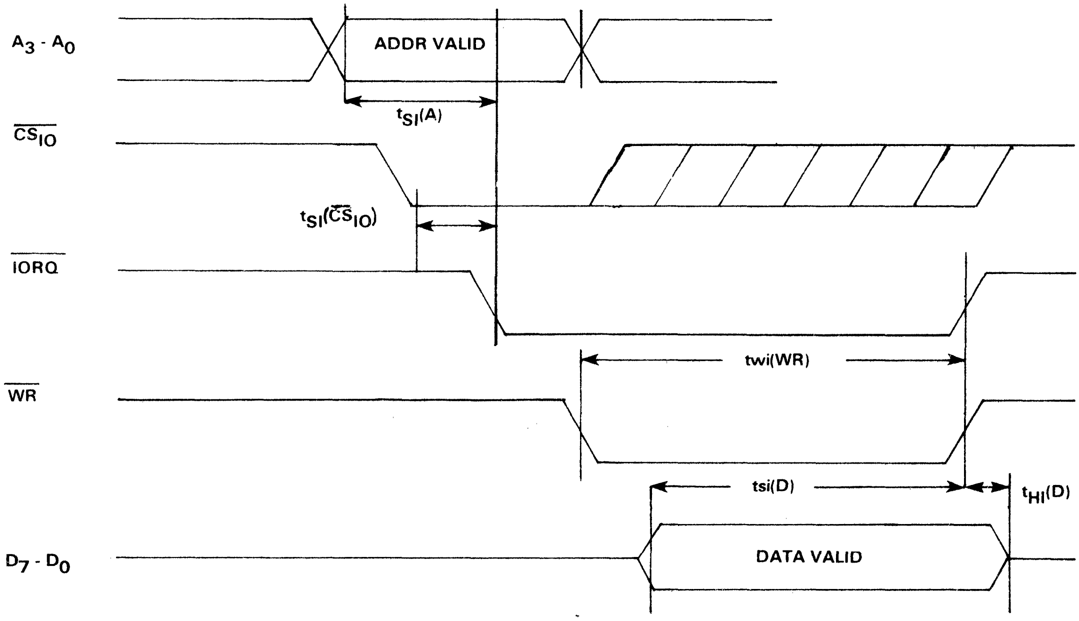
I/O READ TIMING

Figure 10.0-2c



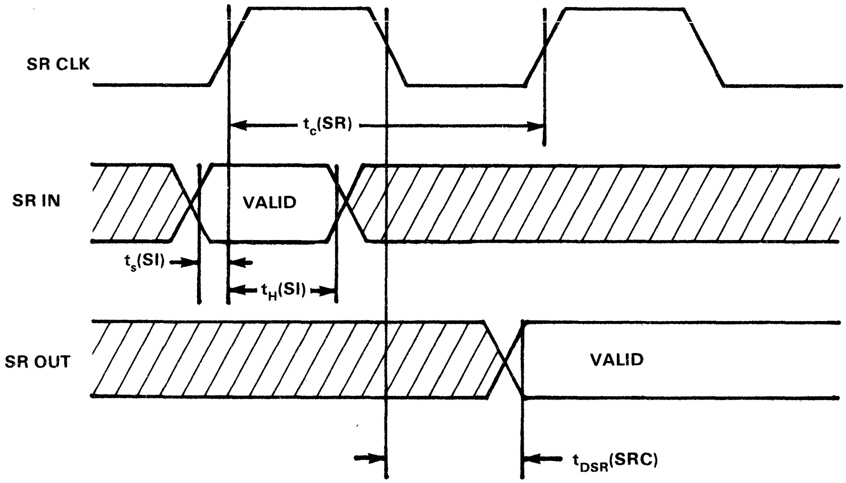
I/O WRITE TIMING

Figure 10.0-2d



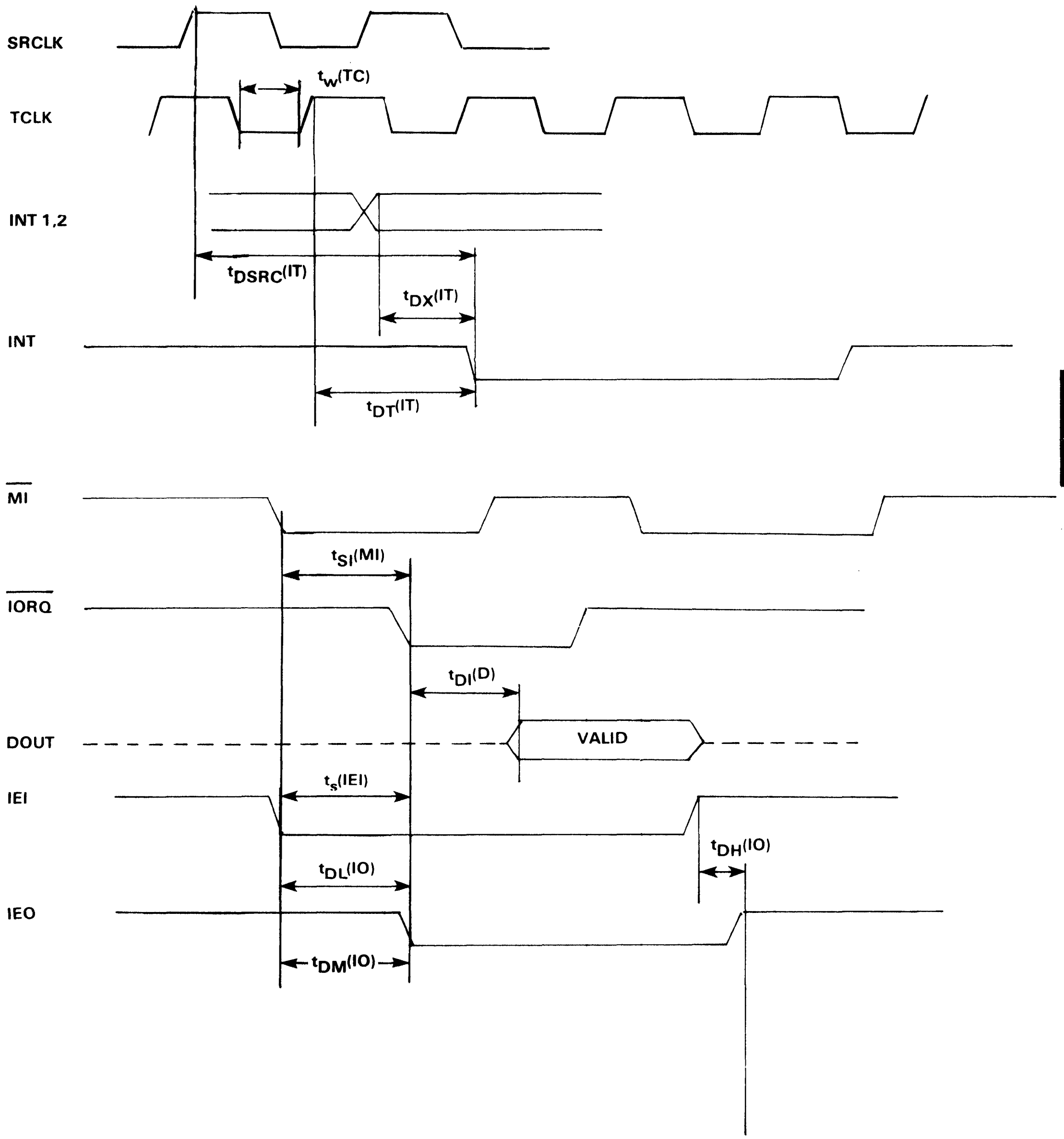
SERIAL I/O TIMING

Figure 10.0-2f



INTERRUPT RESPONSE TIMING

Figure 10.0-2e



10.3 A.C. CHARACTERISTICS

T_A= 0°C to 70°C, V_C = +5V ± 5V unless otherwise noted.

SIGNAL	SYMBOL	PARAMETER	MK3886		MK3886-4		UNIT	CONDITIONS
			MIN	MAX	MIN	MAX		
A_7-A_0	$t_{SM}(A)$	Address Setup time to falling edge of $\overline{\text{MREQ}}$	100		45		n.s.	
	$t_{HM}(A)$	Address Hold time from falling edge of $\overline{\text{MREQ}}$	40		40		n.s.	
	$t_{SI}(A)$	Address Setup time to falling edge of $\overline{\text{IORQ}}$	100		45		n.s.	
	$t_{HI}(A)$	Address Hold time from falling edge of $\overline{\text{IORQ}}$	40		40		n.s.	
$\overline{\text{CS}}_m$	$t_{SM}(\overline{\text{CS}}_M)$	$\overline{\text{CS}}_M$ setup time to falling edge of $\overline{\text{MREQ}}$	85		40		n.s.	
	$t_{HM}(\overline{\text{CS}}_M)$	$\overline{\text{CS}}_M$ Hold time from falling edge of $\overline{\text{MREQ}}$	40		40		n.s.	
$\overline{\text{CS}}_{IO}$	$t_{SI}(\overline{\text{CS}}_{IO})$	$\overline{\text{CS}}_{10}$ Setup time to falling edge of $\overline{\text{IORQ}}$	85		50		n.s.	
	$t_{HI}(\overline{\text{CS}}_{IO})$	$\overline{\text{CS}}_{IO}$ Hold time from falling edge of $\overline{\text{IORQ}}$	40		40		n.s.	
D_7-D_0	$t_{ACM}(D)$	Data Output Delay from $\overline{\text{MREQ}}$ during memory read		350		220	n.s.	Load=50pF + 1TTL Load
	$t_{ACI}(D)$	Data Output Delay from $\overline{\text{IORQ}}$ during I/O read		380		335	n.s.	
	$t_{ARM}(D)$	Data Output Delay from $\overline{\text{RD}}$ to data valid during memory cycle		200		200	n.s.	
	$t_{ARI}(D)$	Data Output Delay from $\overline{\text{RD}}$ to data valid during I/O cycle		380		335	n.s.	
	$t_{SI}(D)$	Data Setup time to rising $\overline{\text{IORQ}}$ or WR during I/O $\overline{\text{WRITE}}$	265		265		n.s.	
	$t_{HI}(D)$	Data hold time from rising $\overline{\text{IORQ}}$ or WR during I/O $\overline{\text{WRITE}}$	30		30		n.s.	
	$t_{SM}(D)$	Data Setup time to rising $\overline{\text{MREQ}}$ or $\overline{\text{WR}}$ during memory $\overline{\text{WRITE}}$	200		110		n.s.	
	$t_{HM}(D)$	Data hold time from rising $\overline{\text{MREQ}}$ or $\overline{\text{WR}}$ during memory $\overline{\text{WRITE}}$	30		30		n.s.	
	$t_{DI}(D)$	Data Output delay from falling $\overline{\text{IORQ}}$ during interrupt acknowledge		340		340	n.s.	
	$t_F(D)$	Delay to float		110			n.s.	
$\overline{\text{WR}}$	$t_{WI}(WR)$	$\overline{\text{WR}}$ pulse width low (I/O cycle)	250		250		n.s.	
	$t_{WM}(WR)$	$\overline{\text{WR}}$ pulse width low (Mem cycle)	100		100		n.s.	
$\overline{\text{MI}}$	$t_{SI}(M1)$	$\overline{\text{M1}}$ Setup time to falling $\overline{\text{IORQ}}$ during interrupt acknowledge	250		250		n.s.	
IEI	$t_S(IEI)$	Setup to falling $\overline{\text{IORQ}}$ DURING interrupt acknowledge	200		140		n.s.	
IEO	$t_{DH}(IO)$	IEO Delay Time from rising edge of IEI		160		160	n.s.	Load=50pF + 1TTL Load
	$t_{DL}(IO)$	IEO Delay Time from falling edge of IEI		130		130	n.s.	
	$t_{DM}(IO)$	IEO Delay from falling edge of $\overline{\text{M1}}$ (interrupt just prior to $\overline{\text{M1}}$)		190		190	n.s.	

10.3 AC CHARACTERISTICS (continued)

SIGNAL	SYMBOL	PARAMETER	MK3886		MK3886-4		UNIT	CONDITIONS
			MIN	MAX	MIN	MAX		
MREQ	t_W(MRL) t_W(MRH)	Pulse Width, MREQ Low Pulse Width, MREQ High	480 190		300			
	t_{DX}(IT)	Delay to falling INT from external interrupt active transition		400		400	n.s.	
INT	t_{DT}(IT)	Delay to INT from Timer Interrupt at rising TCLK		600		600	n.s.	
	t_{DSRC}(IT)	Delay to INT from rising edge of SRCLK		900		900	n.s.	
ZCA	t_{WC}(ZCA)	Width of ZCA pulse	P* t_{CLK} - $t_{CLK/2}$ - 100				n.s.	
SRCLK	t_C(SR)	Period of SRCLK	3.3		3.3		μs.	
	t_S(SI)	SRIN setup with respect to SRCLK edge	250		250		n.s.	
	t_H(SI)	SRIN hold time from SRCLK (x1 mode)	150		150		n.s.	
	$t_{DSR(SRC)}$	Delay to SROUT from SRCLK		2.5		1.9	n.s.	
TCLK	t_C(TC)	Period of Timer Clock	400		250		n.s.	
	t_W(TC)	Timer Clock Low time	180	220	110	140	n.s.	

ORDERING INFORMATION

PART NO.	DESIGNATOR	PACKAGE TYPE	MAXIMUM CLOCK FREQUENCY	TEMPERATURE RANGE
MK3886N	Z80-COMBO	PLASTIC	2.5 MHz	0° to 70°C
MK3886P	Z80-COMBO	CERAMIC	2.5 MHz	0° to 70°C
MK3886J	Z80-COMBO	CERDIP	2.5 MHz	0° to 70°C
MK3886N-4	Z80-COMBO	PLASTIC	4.0 MHz	0° to 70°C
MK3886P-4	Z80-COMBO	CERAMIC	4.0 MHz	0° to 70°C
MK3886J-4	Z80-COMBO	CERDIP	4.0 MHz	0° to 70°C

1981 Z80 MICROCOMPUTER DATA BOOK

I Table of Contents

I TABLE OF CONTENTS

II General Information

II GENERAL INFORMA-TION

III Z80 Family Technical Manuals

III Z80 FAMILY TECHNICAL MANUALS

IV Z80 Family Data Sheets

IV Z80 FAMILY DATA SHEETS

V Z80 Development Equipment

V Z80 DEVELOP-MENT EQUIPMENT

VI Z80 Microcomputer Application Notes

VI Z80 MICRO-COMPUTER APPLICATION NOTES

VII Z80 Microcomputer Peripherals

VII Z80 MICRO-COMPUTER PERIPHERALS

VIII Z80 Military/ Hi-Rel

VIII Z80 MILITARY/ HI-REL

MOSTEK®

Z80 MICROCOMPUTER

Direct Memory Access Controller MK3883

FEATURES

□ Transfers, searches and search/transfers in byte-at-a-time, burst or continuous modes. Cycle length and edge timing can be programmed to match the speed of any port.

□ Dual port addresses (source and destination) generated for memory-to-I/O, memory-to-memory, or I/O-to-I/O operations. Addresses may be fixed or automatically incremented/decremented.

□ Next-operation loading without disturbing current operations via buffered starting-address registers. An entire previous sequence can be repeated automatically.

□ Extensive programmability of functions. CPU can read complete channel status.

□ Standard Z80 Family bus-request and prioritized interrupt-request daisy chains implemented without external logic. Sophisticated, internally modifiable interrupt vectoring.

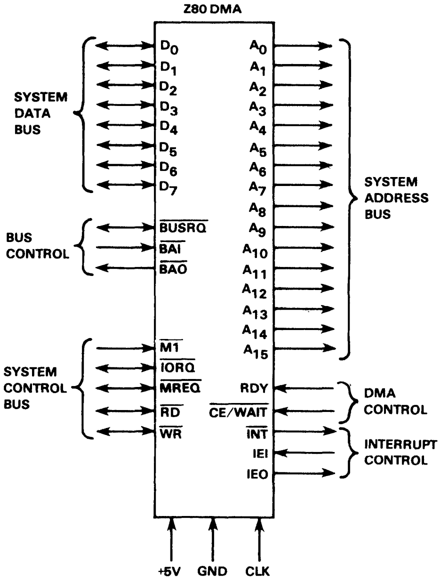
□ Direct interfacing to system buses without external logic.

GENERAL DESCRIPTION

The MK3883 Z80 DMA (Direct Memory Access) is a powerful and versatile device for controlling and processing transfers of data. Its basic function of managing CPU-independent transfers between two ports is augmented by an array of features that optimize transfer speed and control with little or no external logic in systems using an 8- or 16-bit data bus and a 16-bit address bus.

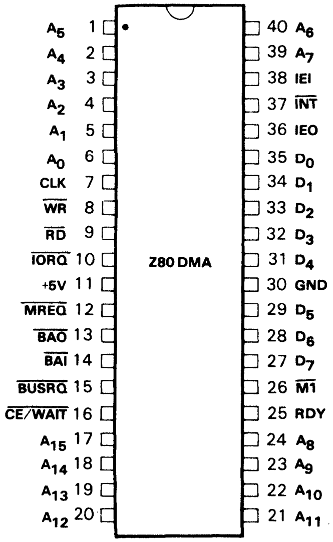
PIN FUNCTIONS

Figure 1



PIN ASSIGNMENTS

Figure 2



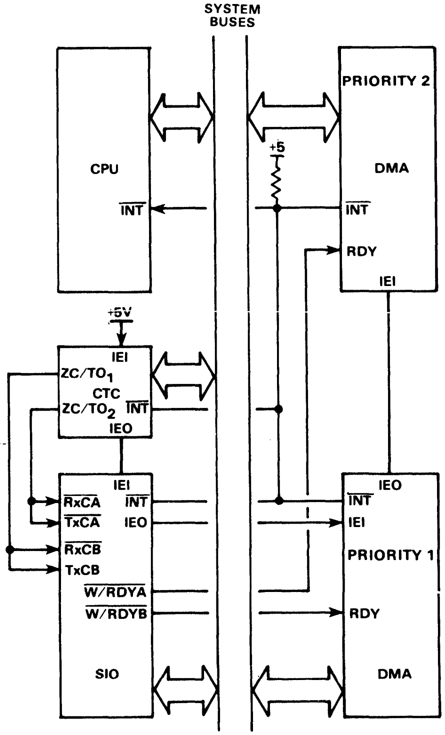
Transfers can be done between any two ports (source and destination), including memory-to-I/O, memory-to-memory, and I/O-to-I/O. Dual port addresses are automatically generated for each transaction and may be either fixed or incrementing/decrementing. In addition, bit-maskable byte searches can be performed either concurrently with transfers or as an operation in itself.

The MK3883 Z80 DMA contains direct interfacing to and independent control of system buses, as well as sophisticated bus and interrupt controls. Many programmable features, including variable cycle timing and auto-restart minimize CPU software overhead. They are especially useful in adapting this special-purpose transfer processor to a broad variety of memory, I/O and CPU environments.

The MK3883 Z80 DMA is an n-channel silicon-gate depletion-load device packaged in a 40-pin plastic, ceramic DIP, or CERDIP. It uses a single +5V power supply and the standard Z80 Family single-phase clock.

Z80 ENVIRONMENT WITH MULTIPLE DMA CONTROLLERS

Figure 3



FUNCTIONAL DESCRIPTION

Classes of Operation

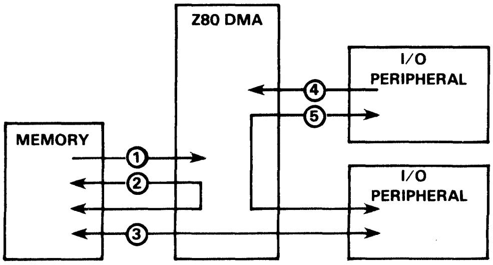
The MK3883 Z80 DMA has three basic classes of operation:

- Transfers of data between two ports (memory or I/O peripheral)
- Searches for a particular 8-bit maskable byte/at a single port in memory or an I/O peripheral
- Combined transfers with simultaneous search between two ports

Figure 4 illustrates the basic functions served by these classes of operation.

BASIC FUNCTIONS OF THE Z80 DMA

Figure 4



1. Search memory
2. Transfer memory-to-memory (optional search)
3. Transfer memory-to-I/O (optional search)
4. Search I/O
5. Transfer I/O-to-I/O (optional search)

During a transfer, the DMA assumes control of the system control, address, and data buses. Data is read from one addressable port and written to the other addressable port, byte by byte. The ports may be programmed to be either system main memory or peripheral I/O devices. Thus, a block of data may be written from one peripheral to another, from one area of main memory to another, or from a peripheral to main memory and vice versa.

During a search-only operation, data is read from the source port and compared byte by byte with DMA-internal register containing a programmable match byte. This match byte may optionally be masked so that only certain bits within the match byte are compared. Search rates up to 1.25M bytes per second can be obtained with the 2.5MHz MK3883 Z80 DMA or 2M bytes per second with the 4MHz MK3883-4 Z80 DMA.

In combined searches and transfers, data is transferred between two ports while simultaneously searching for a bit-maskable byte match.

Data transfers or searches can be programmed to stop or interrupt under various conditions. In addition, CPU-readable status bits can be programmed to reflect the condition.

Modes of Operation

The MK3883 Z80 DMA can be programmed to operate in one of three transfer and/or search modes:

- Byte-at-a-time: data operations are performed one byte at a time. Between each byte operation the system buses are released to the CPU. The buses are requested again for each succeeding byte operation.
- Burst: data operations continue until a port's Ready line to the DMA goes inactive. The DMA then stops and releases the system buses after completing its current byte operation.
- Continuous: data operations continue until the end of the programmed block of data is reached before the system buses are released. If a port's Ready line goes inactive before this occurs, the DMA simply pauses until the Ready line comes active again.

In all modes, once a byte of data is read into the DMA, the operation on the byte will be completed in an orderly fashion, regardless of the state of other signals (including a port's Ready line).

Due to the DMA's high-speed buffered method of reading data, operations on one byte are not completed until the next byte is read in. Consequently, total transfer or search block lengths must be two or more bytes, and that block lengths programmed into the DMA must be one byte less than the desired block length (count is N-1 where N is the block length).

Commands and Status

The Z80 DMA has several writeable control registers and readable status registers available to the CPU. Control bytes can be written to the DMA while the DMA is enabled or disabled, but the act of writing a control byte to the DMA disables the DMA until it is again enabled by a specific command. Status bytes can also be read at any time, but writing the Read Status command or the Read Mask command disables the DMA.

Control bytes to the DMA include those which effect immediate command actions such as enable, disable, reset, load starting-address buffers, continue, clear counters, clear status bits and the like. In addition, many mode-setting control bytes can be written, including mode and class of operation, port configuration, starting addresses, block length, address counting rule, match and match-mask byte, interrupt conditions, interrupt vector, status-affects-vector condition, pulse counting, auto restart, Ready-line and Wait-line rules, and read mask.

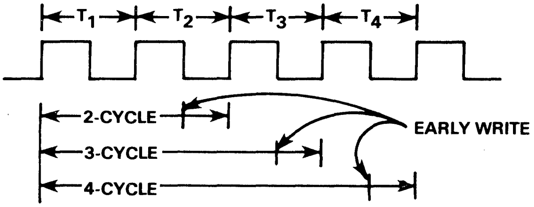
Readable status registers include a general status byte reflecting Ready-line, end-of-block, byte-match and interrupt conditions, as well as Dual-byte registers for the current byte count, Port A address and Port B address.

Variable Cycle

The Z80 DMA has the unique feature of programmable operation-cycle length. This is valuable in tailoring the DMA to the particular requirements of other system components (fast or slow) and maximizes the data-transfer rate. It also eliminates external logic for signal conditioning.

There are two aspects to the variable cycle feature. First, the entire read and write cycles (periods) associated with the source and destination ports can be independently programmed as 2, 3 or 4 T-cycles long (more if Wait cycles are used), thereby increasing or decreasing the speed with which all DMA signals change (Figure 5).

VARIABLE CYCLE LENGTH
Figure 5



Second, the four signals in each port specifically associated with transfers of data (I/O Request, Memory Request, Read and Write) can each have its active trailing edge terminated one-half T-cycle early. This adds a further dimension of flexibility and speed, allowing such things as shorter-than-normal Read or Write signals that go inactive before data starts to change.

Address Generation

Two 16-bit addresses are generated by the Z80 DMA for every transfer or search operation, one address for the source Port A and another for the destination Port B. Each address can be either variable or fixed. Variable addresses can increment or decrement from the programmed starting address. The fixed-address capability eliminates the need for separate enabling wires to I/O ports.

Port addresses are multiplexed onto the system address bus, depending on whether the DMA is reading the source port or writing to the destination port. Two readable address counters (2-bytes each) keep the current address of each port.

Auto Restart

The starting addresses of either port can be reloaded automatically at the end of a block. This option is selected by the Auto Restart control bit. The byte counter is cleared when the addresses are reloaded.

The Auto Restart feature relieves the CPU of software overhead for repetitive operations such as CRT refresh and many others. Moreover, the CPU can write different starting addresses into buffer registers during transfers causing the Auto Restart to begin at a new location.

Interrupts

The MK3883 Z80 DMA can be programmed to interrupt the CPU on four conditions:

- Interrupt on Ready (before requesting bus)
- Interrupt on Match
- Interrupt on End of Block
- Interrupt on Match at End of Block

Any of these interrupts cause an interrupt-pending status bit to be set, and each of them can optionally alter the DMA's interrupt vector. Due to the buffered constraint mentioned under "Modes of Operation," interrupts on Match at End of Block are caused by matches to the byte just prior to the last byte in the block.

The DMA shares the Z80 family's elaborate interrupt scheme, which provides fast interrupt service in real-time applications. In a Z80 CPU environment, the DMA passes its internally modifiable 8-bit interrupt vector to the CPU, which adds an additional eight bits to form the memory address of the interrupt-routine table. This table contains the address of the beginning of the interrupt routine itself.

In this process, CPU control is transferred directly to the interrupt routine, so that the next instruction executed after an interrupt acknowledge is the first instruction of the interrupt routine itself.

Pulse Generation

External devices can keep track of how many bytes have been transferred by using the DMA's pulse output, which provides a signal at 256-byte intervals. The interval sequence may be offset at the beginning by 1 to 255 bytes.

The interrupt line outputs the pulse signal in a manner that prevents misinterpretation by the CPU as an interrupt request, since it only appears when the Bus Request and Bus Acknowledge lines are both active.

PIN DESCRIPTIONS

A_0-A_{15}. System Address Bus (output, 3-state). Addresses generated by the DMA are sent to both source and destination ports (main memory or I/O peripherals) on these lines.

$\overline{\text{BAI}}$. Bus Acknowledge In (input, active Low). Signals that the system buses have been released for DMA control. In multiple-DMA configurations, the $\overline{\text{BAI}}$ pin of the highest priority DMA is normally connected to the Bus Acknowledge pin of the CPU. Lower-priority DMAs have their $\overline{\text{BAI}}$ connected to the $\overline{\text{BAO}}$ of a higher-priority DMA.

$\overline{\text{BAO}}$. Bus Acknowledge Out (output, active Low). In a multiple-DMA configuration, this pin signals that no other higher-priority DMA has requested the system busses. $\overline{\text{BAI}}$ and $\overline{\text{BAO}}$ form a daisy chain for multiple-DMA priority resolution over bus control.

$\overline{\text{BUSRQ}}$. Bus Request (bidirectional, active Low, open drain). As an output, it sends requests for control of the system address bus, data bus and control bus to the CPU. As an input when multiple DMAs are strung together in a priority daisy chain via $\overline{\text{BAI}}$ and $\overline{\text{BAO}}$, it senses when another DMA has requested the buses and causes this DMA to refrain from bus requesting until the other DMA is finished. Because it is a bidirectional pin, there cannot be any buffers between this DMA and any other DMA. It can, however, have a unidirectional into the CPU. A pull-up resistor is connected to this pin.

$\overline{\text{CE/WAIT}}$. Chip Enable and Wait (input, active Low). Normally this functions only as a $\overline{\text{CE}}$ line, but it can also be programmed to serve a $\overline{\text{WAIT}}$ function. As a $\overline{\text{CE}}$ line from the CPU, it becomes active when $\overline{\text{WR}}$ and $\overline{\text{IORQ}}$ are active and the I/O port address on the system address bus is the DMA's address, thereby allowing a transfer of control or command bytes from the CPU to the DMA. As a $\overline{\text{WAIT}}$ line from memory or I/O devices, after the DMA has received a bus-request acknowledge from the CPU, it causes wait states to be inserted in the DMA's operation cycles thereby slowing the DMA to a speed that matches the memory or I/O device.

CLK. System clock (input). Standard Z80 single-phase clock at 2.5MHz (MK3883) or 4.0MHz (MK3883-4). For slower system clocks, a TTL gate with a large pullup resistor may be adequate to meet the timing and voltage level specification. For higher-speed systems, use a clock driver with an active pullup to meet the V_{IH} specification and risetime requirements.

D_0-D_7. System Data Bus (bidirectional, 3-state). Commands from the CPU, DMA status, and data from memory or I/O peripherals are transferred on these lines.

IEI. Interrupt Enable In (input, active High). This is used with IEO to form a priority daisy chain when there is more than one interrupt-driven device. A High on this line indicates that no other device of higher priority is being serviced by a CPU interrupt service routine.

IEO. Interrupt Enable Out (output, active High). IEO is High only if IEI is High and the CPU is not servicing an interrupt from this DMA. Thus, this signal blocks lower-priority devices from interrupting while a higher-priority device is being serviced by its CPU interrupt service routine.

$\overline{\text{INT}}$. Interrupt Request (output, active Low, open drain). This requests a CPU interrupt. The CPU acknowledges the interrupt by pulling its $\overline{\text{IORQ}}$ output Low during an $\overline{\text{M1}}$ cycle. It is typically connected to the $\overline{\text{INT}}$ pin of the CPU with a pullup resistor and tied to all other $\overline{\text{INT}}$ pins in the system.

$\overline{\text{IORQ}}$. Input/Output Request (bidirectional, active Low, 3-state). As an input, this indicates that the lower half of the address bus holds a valid I/O port address for transfer of control or status bytes from or to the CPU, respectively; this DMA is the addressed port if its $\overline{\text{CE}}$ pin and its $\overline{\text{WR}}$ or $\overline{\text{RD}}$ pins are simultaneously active. As an output, after the DMA has taken control of the system busses, it indicates that the lower half of the address bus holds a valid port address for another I/O device involved in a DMA transfer of data. When $\overline{\text{IORQ}}$ and $\overline{\text{M1}}$ are both active simultaneously, an interrupt acknowledge is indicated.

$\overline{\text{M1}}$. Machine Cycle One (input, active Low). Indicates that the current CPU machine cycle is an instruction fetch. It is used by the DMA to decode the return-from-interrupt instruction (RETI) (ED-4D) sent by the CPU. During two-byte instruction fetches, $\overline{\text{M1}}$ is active as each opcode byte is fetched. An interrupt acknowledge is indicated when both $\overline{\text{M1}}$ and $\overline{\text{IORQ}}$ are active.

$\overline{\text{MREQ}}$. Memory Request (bidirectional, active Low, 3-state). This indicates that the address bus holds a valid address for a memory read or write operation. As an input, it indicates that control or status information from or to memory is to be transferred to the DMA, if the DMA's $\overline{\text{CE}}$ and $\overline{\text{WR}}$ or $\overline{\text{RD}}$ lines are simultaneously active. As an output, after the DMA has taken control of the system buses, it indicates a DMA transfer request from or to memory.

$\overline{\text{RD}}$. Read (bidirectional, active Low, 3-state). As an input, this indicates that the CPU wants to read status bytes from the DMA's read registers. As an output, after the DMA has taken control of the system buses, it indicates a DMA-controlled read from a memory or I/O port address.

$\overline{\text{WR}}$. Write (bidirectional, active Low, 3-state). As an input, this indicates that the CPU wants to write control or command bytes to the DMA write registers. As an output, after the DMA has taken control of the system busses, it indicates a DMA-controlled write to a memory or I/O port address.

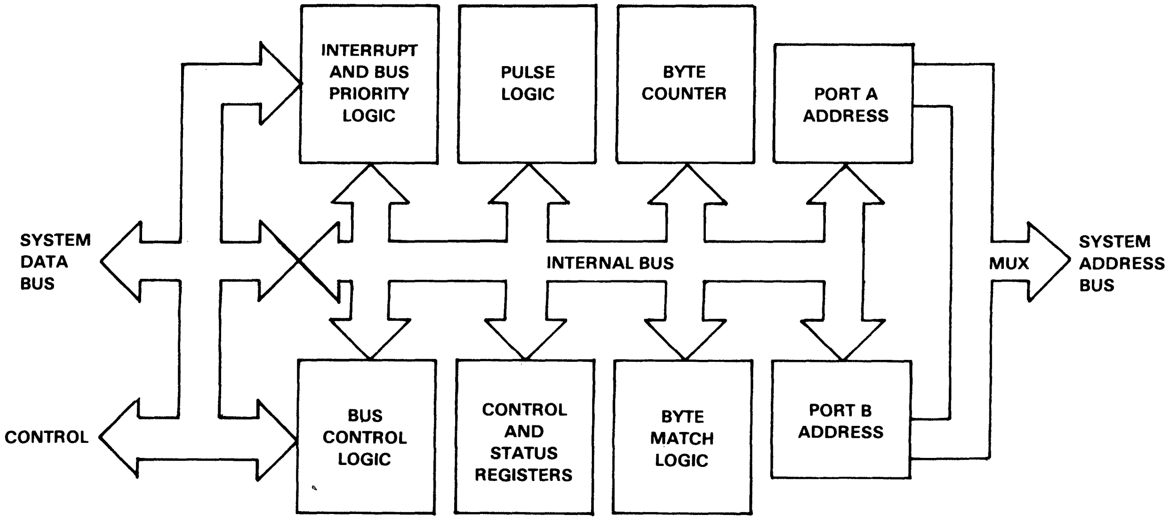
RDY. Ready (input, programmable active Low or High). This is monitored by the DMA to determine when a peripheral device associated with a DMA port is ready for a read or write operation. Depending on the mode of DMA operation (byte, burst or continuous), the RDY line indirectly controls DMA activity by causing the $\overline{\text{BUSRQ}}$ line to go Low or High.

INTERNAL STRUCTURE

The internal structure of the MK3883 Z80 DMA includes driver and receiver circuitry for interfacing with an 8-bit system data bus, a 16-bit system address bus, and system control lines (Figure 6). In a Z80 CPU environment, the DMA can be tied directly to the analogous pins on the CPU (Figure 7) with no additional buffering, except for the $\overline{\text{CE}}/\overline{\text{WAIT}}$ line.

The DMA's internal data bus interfaces with the system data bus and services all internal logic and registers. Addresses generated from this logic for Ports A and B (source and destination) of the DMA's single transfer channel are multiplexed onto the system address bus.

BLOCK DIAGRAM
Figure 6



Specialized logic circuits in the DMA are dedicated to the various functions of external bus interfacing, internal bus control, byte matching, byte counting, periodic pulse generation, CPU interrupts, bus requests, and address generation. A set of twenty-one writeable control registers and seven readable status registers provide the means by which the CPU governs and monitors the activities of these logic circuits. All registers are eight bits wide, with double-byte information stored in adjacent registers. The two starting-address registers (two bytes each) for Ports A and B are buffered.

The 21 writeable control registers are organized into seven base-register groups, most of which have multiple registers. The base registers in each writeable group contain both control/command bits and pointer bits that can be set to address other registers within the group. The seven readable status registers have no analogous second-level registers.

The registers are designated as follows, according to their base-register groups:

WR0-WR6 - Write Register groups 0 through 6 (7 base registers plus 14 associated registers)

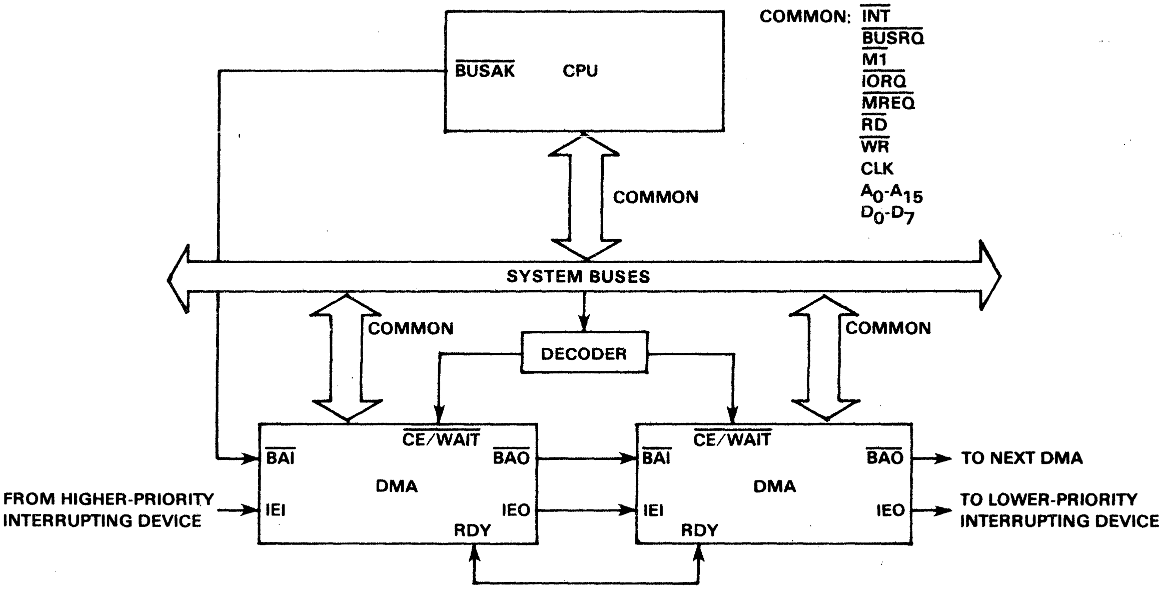
RR0-RR6 - Read Registers 0 through 6

Writing to a register within a write-register group involves first writing to the base register, with the appropriate pointer bits set, then writing to one or more of the other registers within the group. All seven of the readable status registers are accessed sequentially according to a programmable mask contained in one of the writeable registers. The section entitled "Programming" explains this in more detail.

A pipelining scheme is used for reading data in. The programmed block length is the number of bytes compared to the byte counter, which increments at the end of each cycle. In searches, data byte comparisons with the match byte are made during the read cycle of the next byte. Matches are, therefore, discovered only after the next byte is read in.

In multiple-DMA configurations, interrupt-request daisy chains are prioritized by the order in which their IEI and IEO lines are connected. The system bus, however, may not be pre-empted. Any DMA that gains access to the system buses keeps them until it is finished.

MULTIPLE-DMA INTERCONNECTION TO THE Z80 CPU
Figure 7



WRITE REGISTERS

WR0	Base register byte Port A starting address (low byte) Port A starting address (high byte) Block length (low byte) Block length (high byte)
WR1	Base register byte Port A variable-timing byte
WR2	Base register byte Port B variable-timing byte
WR3	Base register byte Mask byte Match byte
WR4	Base register byte Port B starting address (low byte) Port B starting address (high byte) Interrupt control byte Pulse control byte Interrupt vector
WR5	Base register byte
WR6	Base register byte Read mask

READ REGISTERS

RR0	Status byte
RR1	Byte counter (low byte)
RR2	Byte counter (high byte)
RR3	Port A address counter (low byte)
RR4	Port A address counter (high byte)
RR5	Port B address counter (low byte)
RR6	Port B address counter (high byte)

PROGRAMMING

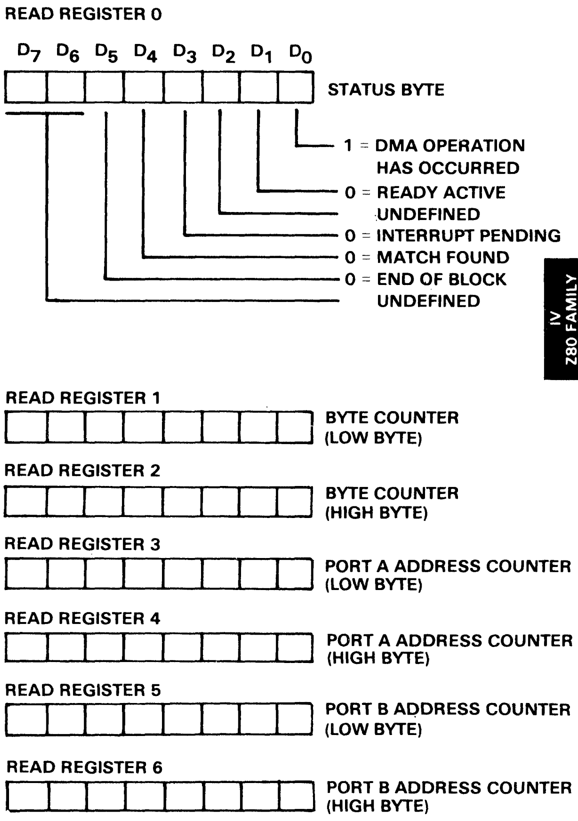
The Z80 DMA has two programmable fundamental states: (1) an enabled state, in which it can gain control of the system buses and direct the transfer of data between ports, and (2) a disabled state, in which it can initiate neither bus requests or data transfers. When the DMA is powered up or reset by any means, it is automatically placed into the disabled state. Program commands can be written to it by the CPU in either state, but this automatically puts the DMA in the disabled state, which is maintained until an enabled command is issued by the CPU. The CPU must program the DMA in advance of any data search or transfer by addressing it as an I/O port and sending a sequence of control bytes using an Output instruction (such as OTIR for the Z80 CPU).

Writing

Control or command bytes are written into one or more of the Write Register groups (WR0-WR6) by first writing to the base register byte in that group. All groups have base registers and most groups have additional associated registers. The associated registers in a group are sequentially accessed by first writing a byte to the base register containing register-group identification and pointer bits (1's) to one or more of that base register's associated registers.

READ REGISTERS

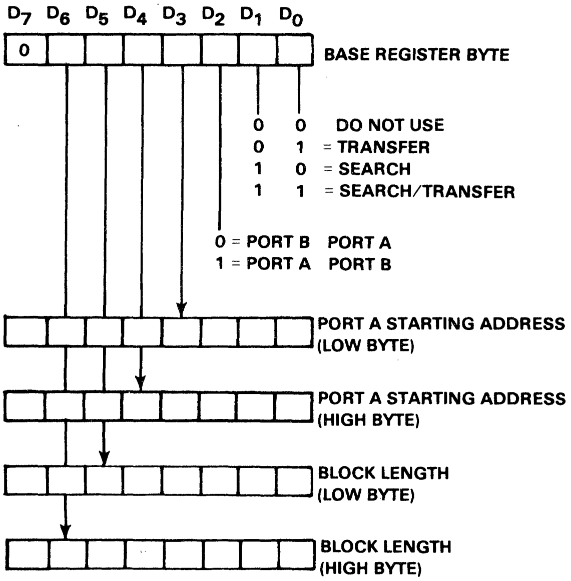
Figure 8a.



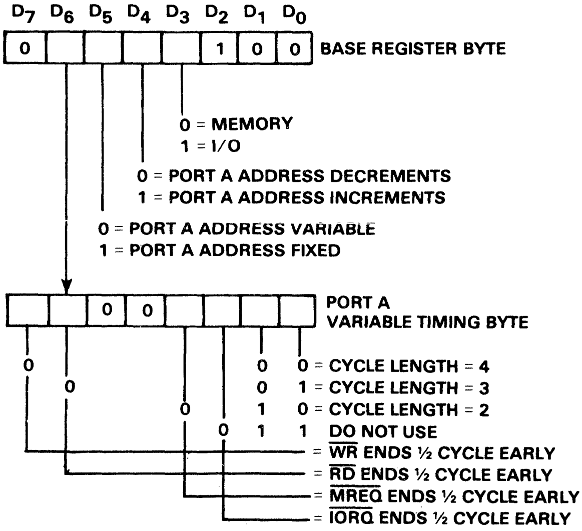
This is illustrated in Figure 8. In this figure, the sequence in which associated registers within a group can be written to is shown by the vertical position of the associated registers. For example, if a byte written to the DMA contains the bits that identify WR0 (bits D0, D1 and D7), and also contains 1's in the bit positions that point to the associated "Port A Starting Address (low byte)" and "Port A Starting Address (high byte)" then the next two bytes written to the DMA will be stored in these two registers, in that order.

WRITE REGISTERS
Figure 8b

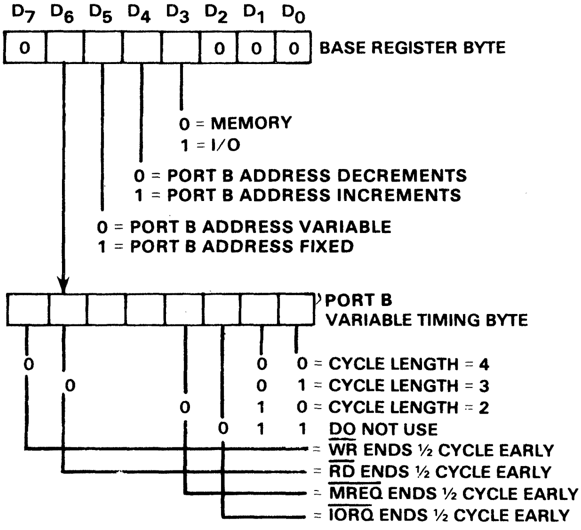
WRITE REGISTER 0



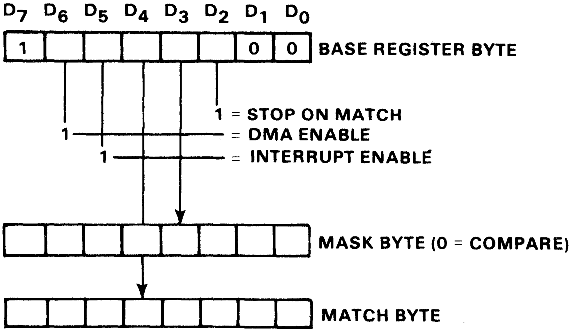
WRITE REGISTER 1


D7 D6 D5 D4 D3 D2 D1 D0
0 1 0 0
BASE REGISTER BYTE
0 = MEMORY
1 = I/O
0 = PORT A ADDRESS DECREMENTS
1 = PORT A ADDRESS INCREMENTS
0 = PORT A ADDRESS VARIABLE
1 = PORT A ADDRESS FIXED
0 0
PORT A VARIABLE TIMING BYTE
0 0 = CYCLE LENGTH = 4
0 1 = CYCLE LENGTH = 3
1 0 = CYCLE LENGTH = 2
1 1 DO NOT USE
0 = WR ENDS ½ CYCLE EARLY
0 = RD ENDS ½ CYCLE EARLY
0 = MREQ ENDS ½ CYCLE EARLY
0 = IORQ ENDS ½ CYCLE EARLY

WRITE REGISTER 2


D7 D6 D5 D4 D3 D2 D1 D0
0 0 0 0
BASE REGISTER BYTE
0 = MEMORY
1 = I/O
0 = PORT B ADDRESS DECREMENTS
1 = PORT B ADDRESS INCREMENTS
0 = PORT B ADDRESS VARIABLE
1 = PORT B ADDRESS FIXED
PORT B VARIABLE TIMING BYTE
0 0 = CYCLE LENGTH = 4
0 1 = CYCLE LENGTH = 3
1 0 = CYCLE LENGTH = 2
1 1 DO NOT USE
0 = WR ENDS ½ CYCLE EARLY
0 = RD ENDS ½ CYCLE EARLY
0 = MREQ ENDS ½ CYCLE EARLY
0 = IORQ ENDS ½ CYCLE EARLY

WRITE REGISTER 3



Reading

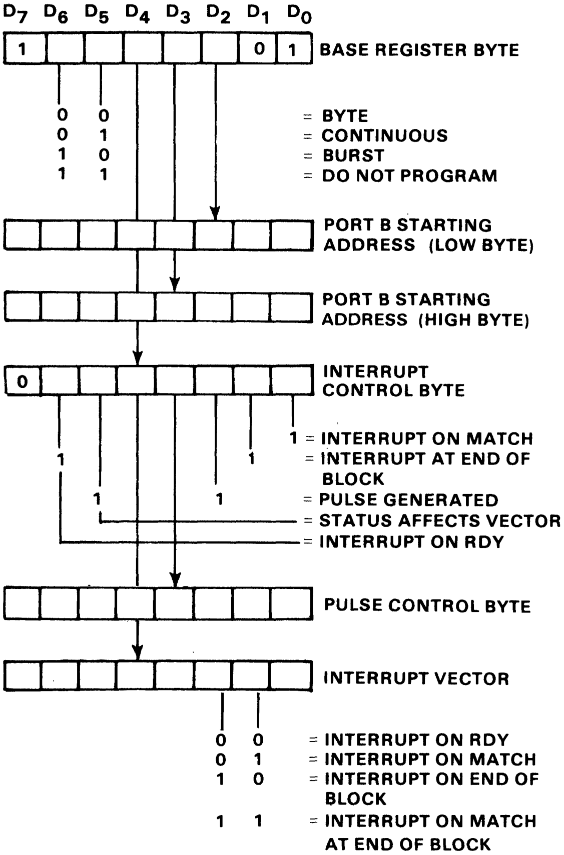
The Read Registers (RR0-RR6) are read by the CPU by addressing the DMA as an I/O port using an Input instruction (such as INIR for the Z80 CPU). The readable bytes contain DMA status, byte-counter values, and port addresses since the last DMA reset. The registers are always read in a fixed sequence beginning with RR0 and ending with RR6. However, the register read in this sequence is determined by programming the Read Mask in WR6. The sequence of reading is initialized by writing an Initiate Read Sequence or Set Read Status command to WR6. After a Reset DMA, the sequence must be initialized with the Initiate Read Sequence command or a Read Status command. The sequence of reading all registers that are not excluded by the Read Mask register must be completed before a new Initiate Read Sequence or Read Status command.

Fixed-Address Programming

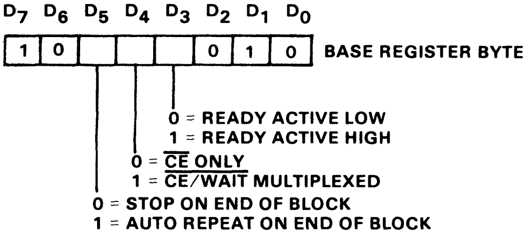
A special circumstance arises when programming a destination port to have a fixed address. The load command in WR6 only loads a fixed address to a port selected as the source, not to a port selected as the destination. Therefore, a fixed destination address must be loaded by temporarily declaring it a fixed-source

WRITE REGISTERS
Figure 8b

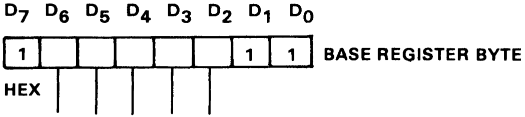
WRITE REGISTER 4



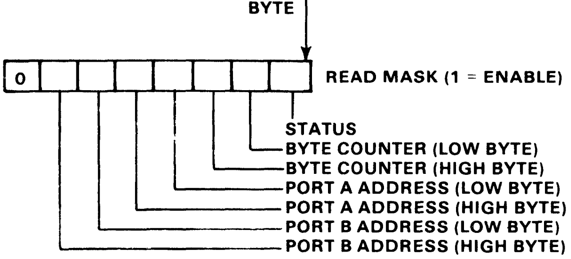
WRITE REGISTER 5



WRITE REGISTER 6



HEX							
C3	1	0	0	0	0	=	RESET INTERRUPT CIRCUITRY, DISABLE INTERRUPT AND BUS REQUEST LOGIC, UNFORCE INTERNAL READY CONDITION, DISABLE "MUXCE" AND STOP AUTO REPEAT.
C7	1	0	0	0	1	=	RESET PORT A TIMING TO STANDARD Z80 CPU TIMING.
CB	1	0	0	1	0	=	RESET PORT B TIMING TO STANDARD Z80 CPU TIMING.
CF	1	0	0	1	1	=	LOAD STARTING ADDRESS FOR BOTH PORTS, CLEAR BYTE COUNTER.
D3	1	0	1	0	0	=	ADDRESS CONTINUE FROM PRESENT LOCATIONS, CLEAR BYTE COUNTER.
AB	0	1	0	1	0	=	ENABLE INTERRUPTS.
AF	0	1	0	1	1	=	DISABLE INTERRUPTS.
A3	0	1	0	0	0	=	RESET AND DISABLE INTERRUPT CIRCUITS (LIKE RETI) AND UNFORCE THE INTERNAL READY CONDITON.
87	0	0	0	0	1	=	ENABLE DMA } BOTH AFFECT ALL OPERATIONS EXCEPT INTERRUPTS, BUT DO NOT RESET ANY FUNCTIONS.
83	0	0	0	0	0	=	DISABLE DMA
A7	0	1	0	0	1	=	INITIATE READ SEQUENCE TO THE FIRST REGISTER DESIGNATED AS READABLE BY THE READ MASK REGISTER.
BF	0	1	1	1	1	=	SET READ STATUS SO NEXT READ IS FROM STATUS REGISTER.
B3	0	1	1	0	0	=	FORCE AN INTERNAL READY CONDITION INDEPENDENT "OF THE RDY" INPUT. (USED FOR MEMORY-TO-MEMORY OPERATIONS WHERE NO RDY SIGNAL IS NEEDED. THIS COMMAND DOES NOT FUNCTION IN THE "BYTE-AT-A-TIME" MODE).
8B	0	0	0	1	0	=	CLEAR MATCH AND END OF BLOCK STATUS BITS.
B7	0	1	1	0	1	=	ENABLE AFTER RETI SO DMA REQUESTS BUS ONLY AFTER RECEIVING A RETI. MUST BE FOLLOWED BY AN ENABLE DMA COMMAND.
BB	0	1	1	1	0	=	READ MASK IS THE FOLLOWING BYTE



0 READ MASK (1 = ENABLE)
STATUS
BYTE COUNTER (LOW BYTE)
BYTE COUNTER (HIGH BYTE)
PORT A ADDRESS (LOW BYTE)
PORT A ADDRESS (HIGH BYTE)
PORT B ADDRESS (LOW BYTE)
PORT B ADDRESS (HIGH BYTE)

SAMPLE DMA PROGRAM

Figure 9

COMMENTS	D_7	D_8	D_5	D_4	D_3	D_2	D_1	D_0	HEX
WR0 sets DMA to receive block length, Port A starting address and temporarily sets Port B as source.	0	1 Block Length Upper Follows	1 Block Length Lower Follows	1 Port A Upper Address Follows	1 Port A Lower Address Follows	0 B→A Temporary for Loading B Address	0 Transfer, No Search	1	79
Port A address (lower)	0	1	0	1	0	0	0	0	50
Port A address (upper)	0	0	0	1	0	0	0	0	10
Block length (lower)	0	0	0	0	0	0	0	0	00
Block length (upper)	0	0	0	1	0	0	0	0	10
WR1 defines Port A as peripheral with fixed address.	0	0 No Timing Follows	0 Address Changes	1 Address Increments	0 Port is Memory	1 This is Port A	0	0	14
WR2 defines Port B as peripheral with fixed address.	0	0 No Timing Follows	1 Fixed Address	0	1 Port is I/O	0 This is Port B	0	0	28
WR4 sets mode to Burst, sets DMA to expect Port B address.	1	1 Burst Mode	0	0 No Interrupt Control Byte Follows	0 No Upper Address	1 Port B Lower Address Follows	0	1	C5
Port B address (lower)	0	0	0	0	0	1	0	1	05
WR5 sets Ready active High	1	0	0 No Auto Restart	0 No Wait States	1 RDY Active High	0	1	0	8A
WR6 loads both Port addresses and resets block counter.*	1	1	0	0 Load	1	1	1	1	CF
WR0 sets Port A as source.*	0	0	0 No Address of Block Length Bytes	0	0	1 A→B	0 Transfer, No Search	1	05
WR6 reloads Port addresses and resets block counter	1	1	0	0 Load	1	1	1	1	CF
WR6 enables DMA to start operation.	1	0	0	0 Enable DMA	0	1	1	1	87

NOTE: The actual number of bytes transferred is one more than specified by the block length.
*These commands are necessary only in the case of a fixed destination address.

address and subsequently declaring the true source as such, thereby implicitly making the other a destination.

The following example illustrates the steps in this procedure, assuming that transfers are to occur from a variable-address source (Port A) to a fixed-address destination (Port B):

1. Temporarily declare Port B as source in WR0.
2. Load Port B address in WR6.
3. Declare Port A as source in WR0.
4. Load Port A address in WR6.
5. Enable DMA in WR6.

Figure 9 illustrates a program to transfer data from memory (Port A) to a peripheral device (Port B). In this example, the Port A memory starting address is 1050$_H$ and the Port B peripheral fixed address is 05$_H$. Note that the data flow is 1001$_H$ bytes—one more than specified by the block length. The table of DMA commands may be stored in consecutive memory locations and transferred to the DMA with an output instruction such as the Z80 CPU's OTIR instruction.

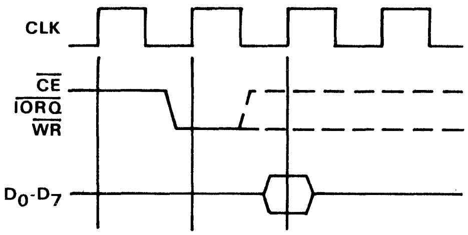
INACTIVE STATE TIMING (DMA as CPU Peripheral)

In its inactive state, the DMA is addressed by the CPU as an I/O peripheral for write and read (control and status) operations. Write timing is illustrated in Figure 10.

Reading of the DMA's status byte, byte counter or port address counters is illustrated in Figure 11. These operations require less than three T-cycles. The $\overline{CE}$, $\overline{IORQ}$ and $\overline{RD}$ lines are made active over two rising edges of CLK, and data appears on the bus approximately one T-cycle after they become active.

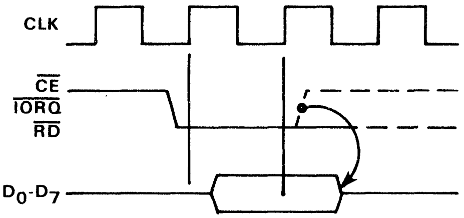
CPU-TO-DMA WRITE CYCLE

Figure 10



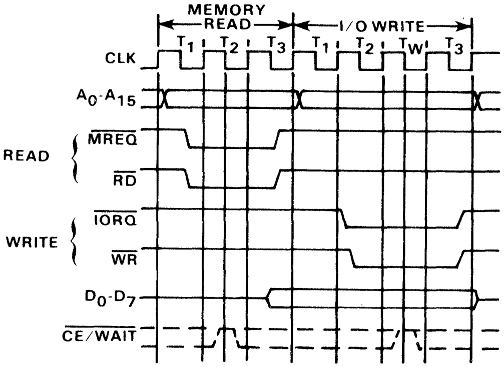
CPU-TO-DMA READ CYCLE

Figure 11



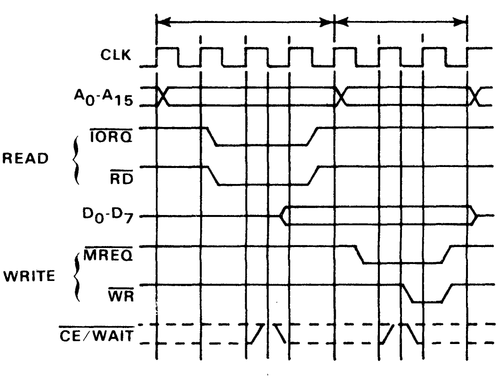
MEMORY-TO-I/O TRANSFER

Figure 12



I/O-TO-MEMORY TRANSFER

Figure 13



ACTIVE STATE TIMING (DMA as Bus Controller)

The DMA is active when it takes control of the system bus and begins transferring data.

Default Read and Write Cycles

By default, and after reset the DMA's timing of read and write operations is exactly the same as the Z80 CPU's timing of read and write cycles for memory and I/O peripherals, with one exception: during a read cycle, data is latched on the falling edge of T_3 and held on the data bus across the boundary between read and write cycles, through the end of the following write cycle.

Figure 12 illustrates the timing for memory-to-I/O port transfers and Figure 13 illustrates I/O-to-memory transfers. Memory-to-memory and I/O-to-I/O transfer timings are simply permutations of these diagrams.

The default timing uses three T-cycles for memory transactions and four T-cycles for I/O transactions, which include one automatically inserted wait cycle between T_2 and T_3. If the $\overline{\text{CE/WAIT}}$ line is programmed to act as WAIT line during the DMA's active state, it is sampled on the falling edge of T_2 for memory transactions and the falling edge of T_W for I/O transactions. If $\overline{\text{CE/WAIT}}$ is low during this time another T-cycle is added, during which the $\overline{\text{CE/WAIT}}$ line will again be sampled. The duration of transactions can thus be indefinitely extended.

Variable Cycle and Edge Timing

The Z80 DMA's default operation-cycle length for the source (read) port and destination (write) port can be independently programmed. This variable-cycle feature allows read or write cycles consisting of two, three or four T-cycles (more if Wait cycles are inserted), thereby increasing or decreasing the speed of all signals generated by the DMA. In addition, the trailing edges of the $\overline{\text{IORQ}}$, $\overline{\text{MREQ}}$, $\overline{\text{RD}}$ and $\overline{\text{WR}}$ signals can be independently terminated one-half cycle early. Figure 14 illustrates this.

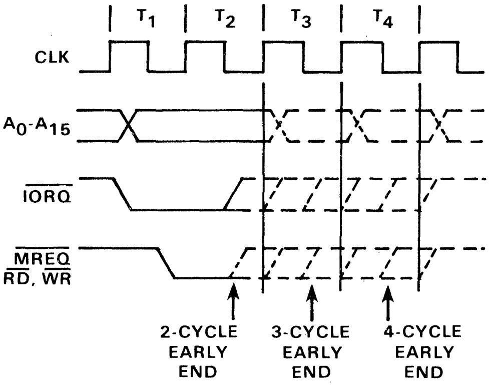
In the variable-cycle mode, unlike default timing, $\overline{\text{IORQ}}$ comes active one-half cycle before $\overline{\text{MREQ}}$, $\overline{\text{RD}}$ and $\overline{\text{WR}}$. $\overline{\text{CE/WAIT}}$ can be used to extend only the 3 or 4 T-cycle variable cycles. It is sampled at the falling edge of T_2 for 3- or 4-cycle memory cycles, and at the falling edge of T_3 for 4-cycle I/O cycles.

During transfers, data is latched on the clock edge causing the rising edge of $\overline{\text{RD}}$ and held until the end of the write cycle.

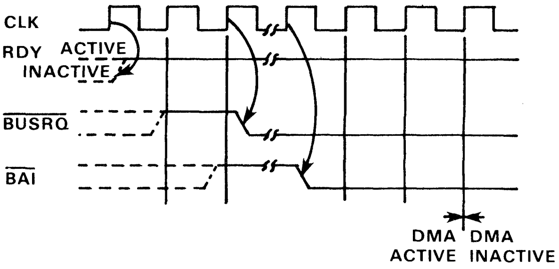
Bus Requests

Figure 15 illustrates the bus request and acceptance timing. The RDY line, which may be programmed active

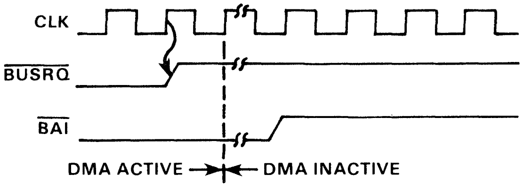
VARIABLE-CYCLE AND EDGE TIMING
Figure 14



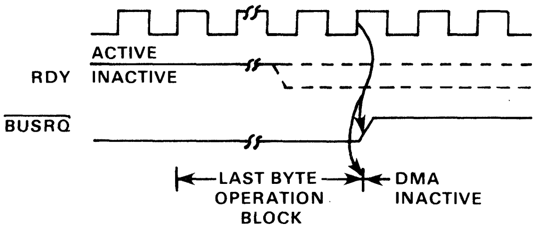
BUS REQUEST AND ACCEPTANCE
Figure 15



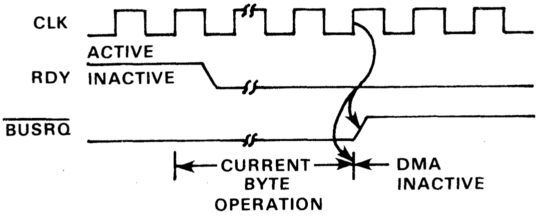
BUS RELEASE (BYTE-AT-A-TIME MODE)
Figure 16



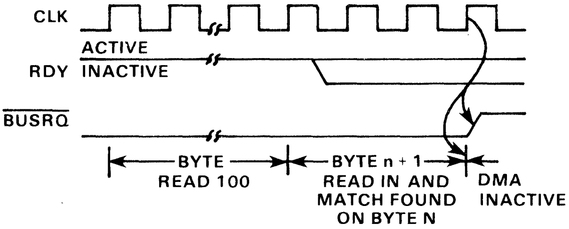
BUS RELEASE (CONTINUOUS MODE)
Figure 17



BUS RELEASE WHEN NOT READY (BURST MODE)
Figure 18



BUS RELEASE ON MATCH (BURST AND CONTINUOUS MODES)
Figure 19



High or Low, is sampled on every rising edge of CLK. If it is found to be active, and the bus is not in use by any other device, the following rising edge of CLK drives BUSRQ low. After receiving BUSRQ, the CPU acknowledges on the BAI input either directly or through a multiple-DMA daisy chain. When a low is detected on BAI for two consecutive rising edges of CLK, the DMA will begin transferring data on the next rising edge of CLK.

Bus Release Byte-at-a-Time

In Byte-at-a-Time mode, BUSRQ is brought high on the rising edge of CLK prior to the end of each read cycle (search-only) or write cycle (transfer and transfer/search) as illustrated in Figure 16. This is done regardless of the state of RDY. There is no possibility of confusion when a Z80 CPU is used since the CPU cannot begin an operation until the following T-cycle. Most other CPUs are not bothered by this either, although note should be taken of it. The next bus request for the next byte will come after both BUSRQ and BAI have returned high.

Bus Release at End of Block

In Burst and Continuous modes, an end of block causes BUSRQ to go High usually on the same rising edge of CLK in which the DMA completes the transfer of the data block (Figure 17). The last byte in the block is transferred even if RDY goes inactive before completion of the last byte transfer.

Bus Release on Not Ready

In Burst Mode, when RDY goes inactive it causes $\overline{\text{BUSRQ}}$ to go High on the next rising edge of CLK after the completion of its current byte operation (Figure 18). The action on $\overline{\text{BUSRQ}}$ is thus somewhat delayed from action on the RDY line. The DMA always completes its current byte operation in an orderly fashion before releasing the bus.

By contrast, $\overline{\text{BUSRQ}}$ is not released in Continuous mode when RDY goes inactive. Instead, the DMA idles after completing the current byte operation, awaiting an active RDY again.

Bus Release on Match

If the DMA is programmed to stop on match in Burst or Continuous modes, a match causes $\overline{\text{BUSRQ}}$ to go inactive on the rising edge of CLK after the next byte following the match (Figure 19). Due to the pipelining scheme, matches are determined while the next byte is being read. Matches at End-of-Block are, therefore, actually matches to the byte immediately preceding the last byte in the block.

The RDY line can go inactive after the matching operation begins without affecting this bus-release timing.

Interrupts

Timings for interrupt acknowledge and return from interrupt are the same as timings for these in other Z80 peripherals.

Interrupt on RDY (interrupt before requesting bus) does not directly affect the $\overline{\text{BUSRQ}}$ line. Instead, the interrupt service routine must handle this by issuing the following commands to WR6:

1. Enable after Return From Interrupt (RETI) Command —Hex B7
2. Enable DMA—Hex 87
3. A RETI instruction that resets the IUS latch in the Z80 DMA

ELECTRICAL CHARACTERISTICS

ABSOLUTE MAXIMUM RATINGS

Operating Ambient Temperature Under Bias	As Specified Under "Ordering Information"
Storage Temperature	-65°C to +150°C
Voltage on any pin with respect to ground	-0.3V to +7V
Power Dissipation	1.5W

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

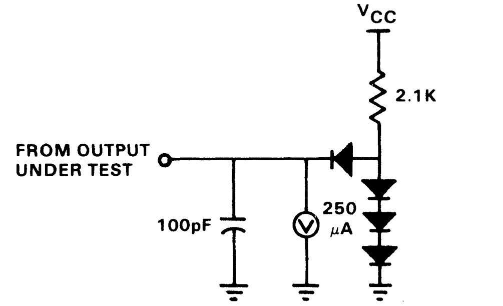
STANDARD TEST CONDITIONS

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin. Standard conditions are as follows:

- $+4.75 \leq V_{CC} \leq +5.25V$
- GND = 0V
- $0°C \leq T_A \leq +70°C$

All AC parameters assume a load capacitance of 100pF max. Timing references between two output signals assume a load difference of 50pF max.

Figure 20



DC CHARACTERISTICS

SYM	PARAMETER	MIN	MAX	UNIT	TEST CONDITION
V_{ILC}	Clock Input Low Voltage	-0.3	0.80	V	
V_{IHC}	Clock Input High Voltage	V_{CC}-.6	5.5	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	

DC CHARACTERISTICS

SYM	PARAMETER	MIN	MAX	UNIT	TEST CONDITION
V_{IH}	Input High Voltage	2.0	5.5	V	
V_{OL}	Output Low Voltage		0.4	V	I_{OL}=3.2mA for $\overline{BUSRQ}$ I_{OL}=2.0mA for all others
V_{OH}	Output High Voltage	2.4		V	I_{OH}=250μA
I_{CC}	Power Supply Current MK3883 MK3883-4		 150 200	 mA mA	
I_{LI}	Input Leakage Current		± 10	μA	V_{IN} = 0 to V_{CC}
I_{LOH}	Tri-State Output Leakage Current in Float		10	μA	V_{OUT}=2.4 to V_{CC}
I_{LOL}	Tri-State Output Leakage Current in Float		-10	μA	V_{OUT}=0.4V
I_{LD}	Data Bus Leakage Current in Input Mode		± 10	μA	$0 \leq V_{IN} \leq V_{CC}$

V_{CC} = 5V ± 5% unless otherwise specified, over specified temperature range.

CAPACITANCE

SYM	PARAMETER	MIN	MAX	UNIT	TEST CONDITION
C	Clock Capacitance		35	pF	Unmeasured Pins
C_{IN}	Input Capacitance		10	pF	Returned to Ground
C_{OUT}	Output Capacitance		10	pF	

f = 1MHz, over specified temperature range

INACTIVE STATE AC CHARACTERISTICS

(See Figure 21)

			MK3883		MK3883-4		
NO	SYM	PARAMETER	MIN	MAX	MIN	MAX	UNIT
1	TcC	Clock Cycle Time	400	4000	250	4000	ns
2	TwCh	Clock Width (High)	170	2000	105	2000	ns
3	TwCl	Clock Width (Low)	170	2000	105	2000	ns
4	TrC	Clock Rise Time		30		30	ns
5	TfC	Clock Fall Time		30		30	ns
6	Th	Hold Time for Any Specified Setup Time	0		0		ns
7	TsC(Cr)	$\overline{IORQ}$, $\overline{WR}$, $\overline{CE}$ ↓ to Clock ↑ Setup	280		145		ns
8	TdDO(RDF)	$\overline{RD}$ to Data Output Delay		500		380	ns
9	TsWM(Cr)	Data In to Clock ↑ Setup ($\overline{WR}$ or $\overline{M1}$)	50		50		ns
10	TdCf(DO)	$\overline{IORQ}$ ↓ to Data Out Delay (INTA Cycle)		340		160	ns

INACTIVE STATE AC CHARACTERISTICS (Continued)

NO	SYM	PARAMETER	MK3883		MK3883-4		UNIT
			MIN	MAX	MIN	MAX	
11	TdRD(Dz)	$\overline{\text{RD}}$↑ to Data Float Delay (output buffer disable)		160		110	ns
12	TsIEI(IORQ)	IEI ↓ to $\overline{\text{IORQ}}$ ↓ Setup (INTA Cycle)	140		140		ns
13	TdIEOr(IEIr)	IEI ↑ to IEO ↑ Delay		210		160	ns
14	TdIEOf(IEIf)	IEI ↓ to IEO ↓ Delay		190		130	ns
15	TdM1(IEO)	$\overline{\text{M1}}$↓to IEO↓ Delay (interrupt just prior to $\overline{\text{M1}}$↓)		300		190	ns
16	TsM1f(Cr)	$\overline{\text{M1}}$↓ to Clock ↑ Setup	210		90		ns
17	TsM1r(Cf)	$\overline{\text{M1}}$ ↑ to Clock ↓ Setup	20		0		ns
18	TsRD(C)	$\overline{\text{RD}}$ ↓ to Clock ↑ Setup ($\overline{\text{M1}}$ Cycle)	240		115		ns
19	TdI(INT)	Interrupt Cause to $\overline{\text{INT}}$ ↓Delay ($\overline{\text{INT}}$ generated only when DMA is inactive)		500		500	ns
20	TdBAIr (BAOr)	$\overline{\text{BAI}}$ ↑ to $\overline{\text{BAO}}$ ↑ Delay		200		150	ns
21	TdBAIf (BAOf)	$\overline{\text{BAI}}$ ↓ to $\overline{\text{BAO}}$ ↓ Delay		200		150	ns

ACTIVE STATE AC CHARACTERISTICS

(See Figure 22)

NO	SYM	PARAMETER	MK3883		MK3883-4	
			MIN(ns)	MAX(ns)	MIN(ns)	MAX(ns)
1	TcC	Clock Cycle Time	400		250	
2	TwCh	Clock Width (High)	180	2000	110	2000
3	TwCl	Clock Width (Low)	180	2000	110	2000
4	TrC	Clock Rise Time		30		20
5	TfC	Clock Fall Time		30		20
6	TdA	Address Output Delay		145		110
7	TdC(Az)	Clock ↑ to Address Float Delay		110		90
8	TsA(MREQ)	Address to $\overline{\text{MREQ}}$ ↓ Setup (Memory Cycle)	(2)+(5)-75		(2)+(5)-75	
9	TsA(IRW)	Address Stable to $\overline{\text{IORQ}}$, $\overline{\text{RD}}$, $\overline{\text{WR}}$ ↓ Setup (I/O Cycle)	(1)-80		(1)-70	
10	TdRW(A)	$\overline{\text{RD}}$, $\overline{\text{WR}}$ ↑ to Addr. Stable Delay	(3)+(4)-40		(3)+(4)-50	
*11	TdRW(Az)	$\overline{\text{RD}}$, $\overline{\text{WR}}$ ↑ to Addr. Float	(3)+(4)-60		(3)+(4)-45	
12	TdCf(DO)	Clock ↓ to Data Out Delay		230		150

ACTIVE STATE AC CHARACTERISTICS

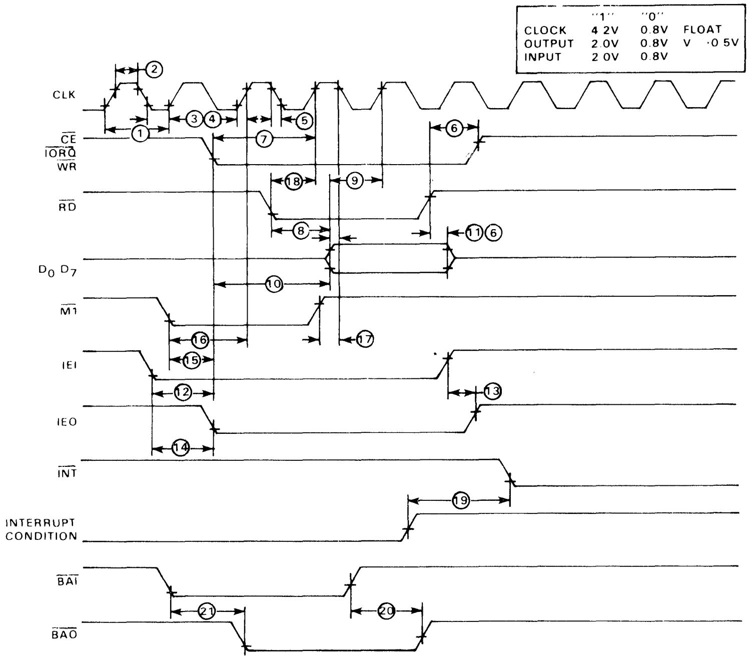
NO	SYM	PARAMETER	MK3883		MK3883-4	
			MIN(ns)	MAX(ns)	MIN(ns)	MAX(ns)
13	TdCr(Dz)	Clock ↑ to Data Float Delay (Write Cycle)		90		90
14	TsDI(Cr)	Data In to Clock ↑ Setup (Read cycle when falling edge ends read)	50		35	
15	TsDI(Cf)	Data In to Clock ↓ Setup (Read cycle when falling edge ends read)	60		50	
16	TsDO(WfM)	Data Out to $\overline{\text{WR}}$ ↓ Setup (Memory Cycle)	(1)-210		(1)-170	
17	TsDO(WfI)	Data Out to $\overline{\text{WR}}$ ↓ Setup (I/O cycle)	100		100	
18	TdWr(DO)	$\overline{\text{WR}}$ ↑ to Data Out Delay	(3)+(4)-80		(3)+(4)-70	
19	Th	Hold Time for Any Specified Setup Time	0		0	
*20	TdCr(Mf)	Clock ↑ to $\overline{\text{MREQ}}$ ↓ Delay		100		85
21	TdCf(Mf)	Clock ↓ to $\overline{\text{MREQ}}$ ↓ Delay		100		85
22	TdCr(Mr)	Clock ↑ to $\overline{\text{MREQ}}$ ↑ Delay		100		85
23	TdCf(Mr)	Clock ↓ to $\overline{\text{MREQ}}$ ↑ Delay		100		85
24	TwM1	$\overline{\text{MREQ}}$ Low Pulse Width	(1)-40		(1)-30	
25	TwMh	$\overline{\text{MREQ}}$ High Pulse Width	(2)+(5)-30		(2)+(5)-20	
26	TdCr(If)	Clock ↑ to $\overline{\text{IORQ}}$ ↓ Delay		90		75
27	TdCf(If)	Clock ↓ to $\overline{\text{IORQ}}$ ↓ Delay		110		85
28	TdCr(Ir)	Clock ↑ to $\overline{\text{IORQ}}$ ↑ Delay		100		85
29	TdCf(Ir)	Clock ↓ to $\overline{\text{IORQ}}$ ↑ Delay		110		85
30	TdCr(Rf)	Clock ↑ to $\overline{\text{RD}}$ ↓ Delay		100		85
31	TdCf(Rf)	Clock ↓ to $\overline{\text{RD}}$ ↓ Delay		130		95
32	TdCr(Rr)	Clock ↑ to $\overline{\text{RD}}$ ↑ Delay		100		85
33	TdCf(Rr)	Clock ↓ to $\overline{\text{RD}}$ ↑ Delay		110		85
34	TdCr(Wf)	Clock ↑ to $\overline{\text{WR}}$ ↓ Delay		80		65
35	TdCf(Wf)	Clock ↓ to $\overline{\text{WR}}$ ↓ Delay		90		80
*36	TdCr(Wr)	Clock ↑ to $\overline{\text{WR}}$ ↑ Delay		100		80
37	TdCf(Wr)	Clock ↓ to $\overline{\text{WR}}$ ↑ Delay		100		80
38	TwWl	$\overline{\text{WR}}$ Low Pulse Width	(1)-40		(1)-30	
39	TsWA(Cf)	$\overline{\text{WAIT}}$ to Clock ↓ Setup	70		70	
40	TdCr(B)	Clock ↑ to $\overline{\text{BUSRQ}}$ Delay		100		100
41	TdCr(Iz)	Clock ↑ to $\overline{\text{IORQ}}$, $\overline{\text{MREQ}}$, $\overline{\text{RD}}$, $\overline{\text{WR}}$ Float Delay		100		80

NOTES:

1. Numbers in parentheses are other parameter-numbers in this table; their values should be substituted in equations.
2. All equations imply DMA default (standard) timing.
3. Data must be enabled onto data bus when RD is active.
4. Asterisk(*) before parameter number means the parameter is not illustrated in the AC Timing Diagrams.

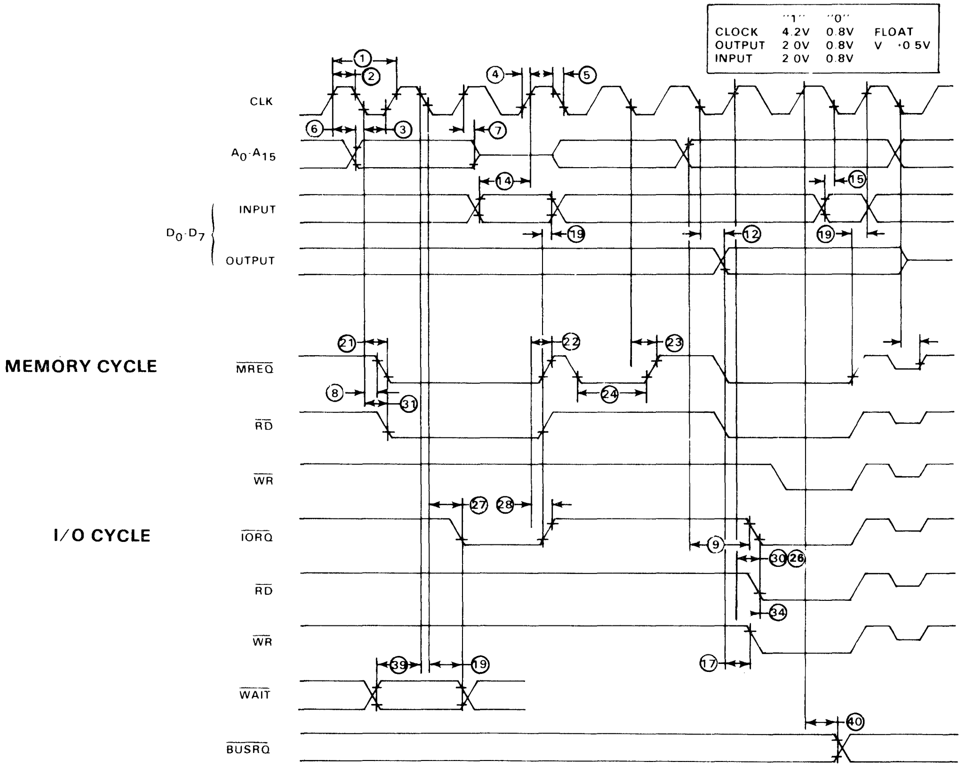
INACTIVE STATE CHARACTERISTICS

Figure 21



ACTIVE STATE CHARACTERISTICS

Figure 22



ORDERING INFORMATION

PART NO.	PACKAGE TYPE	MAX CLOCK FREQUENCY	TEMPERATURE RANGE
MK3883N	Z80-DMA Plastic	2.5 MHz	0°C to +70°C
MK3883P	Z80-DMA Ceramic	2.5 MHz	0°C to +70°C
MK3883J	Z80-DMA CERDIP	2.5 MHz	0°C to +70°C
MK3883N-10	Z80-DMA Plastic	2.5 MHz	-40°C to +85°C
MK3883P-10	Z80-DMA Ceramic	2.5 MHz	-40°C to +85°C
MK3883J-10	Z80-DMA CERDIP	2.5 MHz	-40°C to +85°C
MK3883N-4	Z80A-DMA Plastic	4 MHz	0°C to +70°C
MK3883P-4	Z80A-DMA Ceramic	4 MHz	0°C to +70°C
MK3883J-4	Z80A-DMA CERDIP	4 MHz	0°C to +70°C

PRELIMINARY

MOSTEK®

Z80 MICROCOMPUTER

Serial Input/Output Controller MK3884

FEATURES

- □ Two independent full-duplex channels, with separate control and status lines for modems or other devices.
- □ Data rates of 0 to 500K bits/second in the x1 clock mode with a 2.5MHz clock (MK3884 Z80 SIO), or 0 to 800K bits/second with a 4.0MHz clock (MK3884-4 Z80 SIO).
- □ Asynchronous protocols: everything necessary for complete messages in 5, 6, 7 or 8 bits/character. Includes variable stop bits and several clock-rate multipliers; break generation and detection; parity; overrun and framing error detection.
- □ Synchronous protocols: everything necessary for complete bit- or byte-oriented messages in 5, 6, 7 or 8 bits/character, including IBM Bisync, SDLC, HDLC, CCITT-X.25 and others. Automatic CRC generation/-checking, sync character and zero insertion/-deletion, abort generation/detection and flag insertion.
- □ Receiver data registers quadruply buffered, transmitter registers doubly buffered.
- □ Highly sophisticated and flexible daisy-chain interrupt vectoring for interrupts without external logic.

DESCRIPTION

The MK3884 Z80 SIO Serial Input/Output Controller is a dual-channel data communication interface with extraordinary versatility and capability. Its basic functions as a serial-to-parallel, parallel-to-serial converter/controller can be programmed by a CPU for a broad range of serial communication applications.

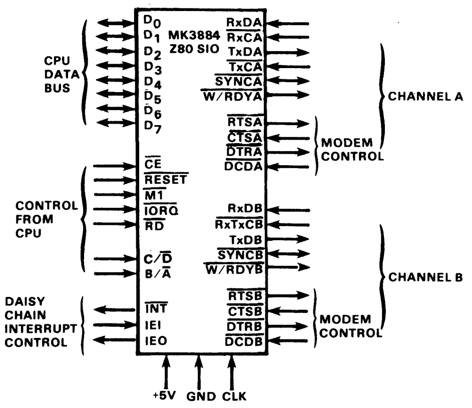
The device supports all common asynchronous and synchronous protocols, byte- or bit-oriented, and performs all of the functions traditionally done by UARTs, USARTs and synchronous communication controllers combined, plus additional functions traditionally performed by the CPU. Moreover, it does this on two fully-independent channels, with an exceptionally sophisticated interrupt structure that allows very fast transfers.

Full interfacing is provided for CPU or DMA control. In addition to data communication, the circuit can handle virtually all types of serial I/O with fast (or slow) peripheral devices. While designed primarily as a member of the Z80 family, its versatility makes it well suited to many other CPUs.

The Z80 SIO is an n-channel silicon-gate depletion-load device packaged in a 40-pin plastic, ceramic DIP, or CERDIP. It uses a single +5V power supply and the standard Z80 family single-phase clock.

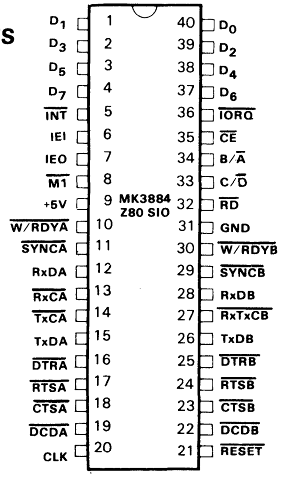
MK3884 Z80 SIO PIN FUNCTIONS

Figure 1



MK3884 Z80 SIO PIN ASSIGNMENTS

Figure 2



PIN DESCRIPTIONS

Figures 1 through 6 illustrate the three pin configurations (bonding options) available in the SIO. The constraints of a 40-pin package make it impossible to bring out the Receive Clock ($\overline{RxC}$), Transmit Clock ($\overline{TxC}$), Data Terminal Ready ($\overline{DTR}$) and Sync ($\overline{SYNC}$) signals for both channels. Therefore, either Channel B lacks a signal or two signals are bonded together in the three bonding options offered:

- MK3887 Z80 SIO lacks $\overline{SYNCB}$
- MK3885 Z80 SIO lacks $\overline{DTRB}$
- MK3884 Z80 SIO has all four signals, but $\overline{TxCB}$ and $\overline{RxCB}$ are bonded together

The pin descriptions are as follows:

B/$\overline{A}$. Channel A Or B Select (input, High selects Channel B). This input defines which channel is accessed during a data transfer between the CPU and the SIO. Address bit A_0 from the CPU is often used for the selection function.

C/$\overline{D}$. Control Or Data Select (input, High selects Control). This input defines the type of information transfer performed between the CPU and the SIO. A High at this input during a CPU write to the SIO causes the information on the data bus to be interpreted as a command for the channel selected by B/$\overline{A}$. A Low at C/$\overline{D}$ means that the information on the data bus is data. Address bit A_1 is often used for this function.

$\overline{CE}$. Chip Enable (Input, active Low). A Low level at this input enables the SIO to accept command or data input from the CPU during a write cycle, or to transmit data to the CPU during a read cycle.

CLK. System Clock (input). The SIO uses the standard Z80 System Clock to synchronize internal signals. This is a single-phase clock.

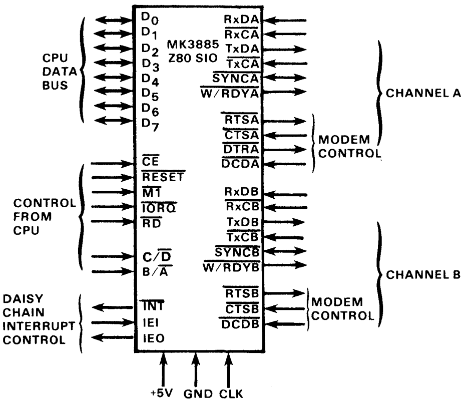
$\overline{CTSA}$, $\overline{CTSB}$. Clear To Send (inputs, active Low). When programmed as Auto Enables, a Low on these inputs enables the respective transmitter. If not programmed as Auto Enables, these inputs may be programmed as general-purpose inputs. Both inputs are Schmitt-trigger buffered to accommodate slow-risetime signals. The SIO detects pulses on these inputs and interrupts the CPU on both logic level transitions. The Schmitt-trigger buffering does not guarantee a specified noise-level margin.

D_0-D_7. System Data Bus (bidirectional, 3-state). The system data bus transfers data and commands between the CPU and the Z80 SIO. D_0 is the least significant bit.

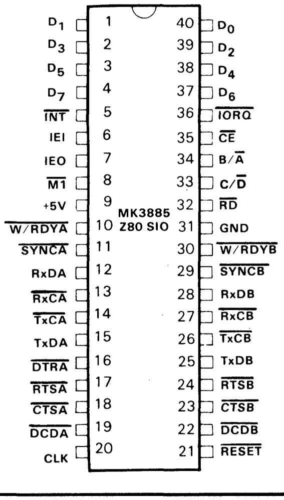
$\overline{DCDA}$, $\overline{DCDB}$. Data Carrier Detect (inputs, active Low). These pins function as receiver enables if the SIO is programmed for Auto Enables; otherwise they may be used as general-purpose input pins. Both pins are Schmitt-trigger buffered to accommodate slow-risetime signals. The SIO detects pulses on these pins and interrupts the CPU on both logic level transitions. Schmitt-trigger buffering does not guarantee a specific noise-level margin.

$\overline{DTRA}$, $\overline{DTRB}$. Data Terminal Ready (outputs, active Low). These outputs follow the state programmed into Z80 SIO. They can also be programmed as general-purpose outputs.
In the MK3885 bonding option, $\overline{DTRB}$ is omitted.

MK3885 Z80 SIO PIN FUNCTIONS
Figure 3



MK3885 Z80 SIO PIN ASSIGNMENTS
Figure 4



IEI. Interrupt Enable In (input, active High). This signal is used with IEO to form a priority daisy chain when there is more than one interrupt-driven device. A High on this line indicates that no other device of higher priority is being serviced by a CPU interrupt service routine.

IEO. Interrupt Enable Out (output, active High). IEO is High only if IEI is High and the CPU is not servicing an interrupt from this SIO. Thus, this signal blocks lower priority devices from interrupting while a higher priority device is being serviced by its CPU interrupt service routine.

$\overline{\mathrm{INT}}$. Interrupt Request (output, open drain, active Low). When the SIO is requesting an interrupt, it pulls $\overline{\mathrm{INT}}$ Low.

$\overline{\mathrm{IORQ}}$. Input/Output Request (input from CPU, active Low). $\overline{\mathrm{IORQ}}$ is used in conjunction with B/$\overline{\mathrm{A}}$, C/$\overline{\mathrm{D}}$, $\overline{\mathrm{CE}}$ and $\overline{\mathrm{RD}}$ to transfer commands and data between the CPU and the SIO. When $\overline{\mathrm{CE}}$, $\overline{\mathrm{RD}}$ and $\overline{\mathrm{IORQ}}$ are all active, the channel selected by B/$\overline{\mathrm{A}}$ transfers data to the CPU (a read operation). When $\overline{\mathrm{CE}}$ and $\overline{\mathrm{IORQ}}$ are active, but $\overline{\mathrm{RD}}$ is inactive, the channel selected by B/$\overline{\mathrm{A}}$ is written to by the CPU with either data or control information as specified by C/$\overline{\mathrm{D}}$. As mentioned previously, if $\overline{\mathrm{IORQ}}$ and $\overline{\mathrm{M1}}$ are active simultaneously, the CPU is acknowledging an interrupt and the SIO automatically places its interrupt vector on the CPU data bus if it is the highest priority device requesting an interrupt.

$\overline{\mathrm{M1}}$. Machine Cycle (input from Z80 CPU, active Low). When $\overline{\mathrm{M1}}$ is active and $\overline{\mathrm{RD}}$ is also active, the Z80 CPU is fetching an instruction from memory; when $\overline{\mathrm{M1}}$ is active while $\overline{\mathrm{IORQ}}$ is active, the SIO accepts $\overline{\mathrm{M1}}$ and $\overline{\mathrm{IORQ}}$ as an interrupt acknowledge if the SIO is the highest priority device that has interrupted the Z80 CPU.

$\overline{\mathrm{RxCA}}$, $\overline{\mathrm{RxCB}}$. Receiver Clocks (inputs). Receive data is sampled on the rising edge of $\overline{\mathrm{RxC}}$. The Receive Clocks may be 1, 16, 32 or 64 times the data rate in asynchronous modes. These clocks may be driven by the Z80 CTC Counter Timer Circuit for programmable baud rate generation. Both inputs are Schmitt-trigger buffered (no noise level margin is specified).

In the MK3884 bonding option, $\overline{\mathrm{RxCB}}$ is bonded together with $\overline{\mathrm{TxCB}}$.

$\overline{\mathrm{RD}}$. Read Cycle Status (input from CPU, active Low). If $\overline{\mathrm{RD}}$ is active, a memory or I/O read operation is in progress. $\overline{\mathrm{RD}}$ is used with B/$\overline{\mathrm{A}}$, $\overline{\mathrm{CE}}$ and $\overline{\mathrm{IORQ}}$ to transfer data from the SIO to the CPU.

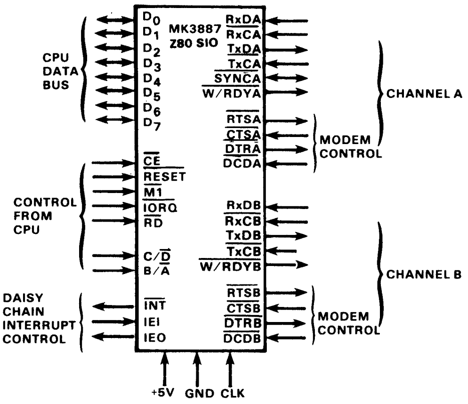
RxDA, RxDB. Receive Data (inputs, active High). Serial data at TTL levels.

$\overline{\mathrm{RESET}}$. Reset (input, active Low). A Low $\overline{\mathrm{RESET}}$ disables both receivers and transmitters, forces TxDA and TxDB marking, forces the modem controls High and disables all interrupts. The control registers must be rewritten after the SIO is reset and before data is transmitted or received.

$\overline{\mathrm{RTSA}}$, $\overline{\mathrm{RTSB}}$. Request To Send (outputs, active Low). When the RTS bit in Write Register 5 (Figure 14) is set, the $\overline{\mathrm{RTS}}$ output goes Low. When the $\overline{\mathrm{RTS}}$ bit is reset in the Asynchronous mode, the output goes High after the transmitter is empty. In Synchronous modes, the $\overline{\mathrm{RTS}}$ pin strictly follows the state of the RTS bit. Both pins can be used as general-purpose outputs.

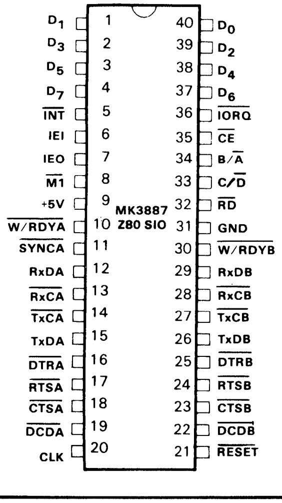
MK3887 Z80 SIO PIN FUNCTIONS

Figure 5



MK3887 Z80 SIO PIN ASSIGNMENTS

Figure 6



$\overline{\text{SYNCA}}$, $\overline{\text{SYNCB}}$. Synchronization (inputs/outputs, active Low). These pins can act either as inputs or outputs. In the asynchronous receive mode, they are inputs similar to $\overline{\text{CTS}}$ and $\overline{\text{DCD}}$. In this mode, the transitions on these lines affect the state of the Sync/-Hunt status bit in Read Register 0 (Figure 13), but have no other function. In the External Sync mode, these lines also act as inputs. When external synchronization is achieved, $\overline{\text{SYNC}}$ must be driven Low on the second rising edge of $\overline{\text{RxC}}$ after that rising edge of $\overline{\text{RxC}}$ on which the last bit of the sync character was received. In other words, after the sync pattern is detected, the external logic must wait for two full Receive Clock cycles to activate the $\overline{\text{SYNC}}$ input. Once $\overline{\text{SYNC}}$ is forced Low, it should be kept Low until the CPU informs the external synchronization detect logic that synchronization has been lost or a new message is about to start. Character assembly begins on the rising edge of $\overline{\text{RxC}}$ that immediately precedes the falling edge of $\overline{\text{SYNC}}$ in the External Sync mode.

In the internal synchronization mode (Monosync and Bisync), these pins act as outputs that are active during the part of the receive clock ($\overline{\text{RxC}}$) cycle in which sync characters are recognized. The sync condition is not latched, so these outputs are active each time a sync pattern is recognized, regardless of character boundaries.

In the MK3887 bonding option, $\overline{\text{SYNCB}}$ is omitted.

$\overline{\text{TxCA}}$, $\overline{\text{TxCB}}$. Transmitter Clocks (inputs). TxD changes from the falling edge of $\overline{\text{TxC}}$. In asynchronous modes, the Transmitter Clocks may be 1, 16, 32 or 64 times the data rate; however, the clock multiplier for the transmitter and the receiver must be the same. The Transmit Clock inputs are Schmitt-trigger buffered for relaxed rise- and fall-time requirements (no noise level margin is specified). Transmitter Clocks may be driven by the Z80 CTC Counter Timer Circuit for programmable baud rate generation.

In the MK3884 bonding option, $\overline{\text{TxCB}}$ is bonded together with $\overline{\text{RxCB}}$.

TxDA, TxDB. Transmit Data (outputs, active High). Serial data at TTL levels.

$\overline{\text{W/RDYA}}$, $\overline{\text{W/RDYB}}$. Wait/Ready A, Wait/Ready B (outputs, open drain, when programmed for Wait function; driven High and Low when programmed for Ready function). These dual-purpose outputs may be programmed as Ready lines for a DMA controller or as Wait lines that synchronize the CPU to the SIO data rate. The reset state is open drain.

FUNCTIONAL CAPABILITIES

The functional capabilities of the Z80 SIO can be described from two different points of view: as a data communications device, it transmits and receives serial data in a wide variety of data-communication protocols; as a Z80 family peripheral, it interacts with the Z80 CPU and other peripheral circuits, sharing the data, address and control buses, as well as being a part of the Z80 interrupt structure. As a peripheral to other microprocessors, the SIO offers valuable features such as non-vectored interrupts, polling and simple handshake capability.

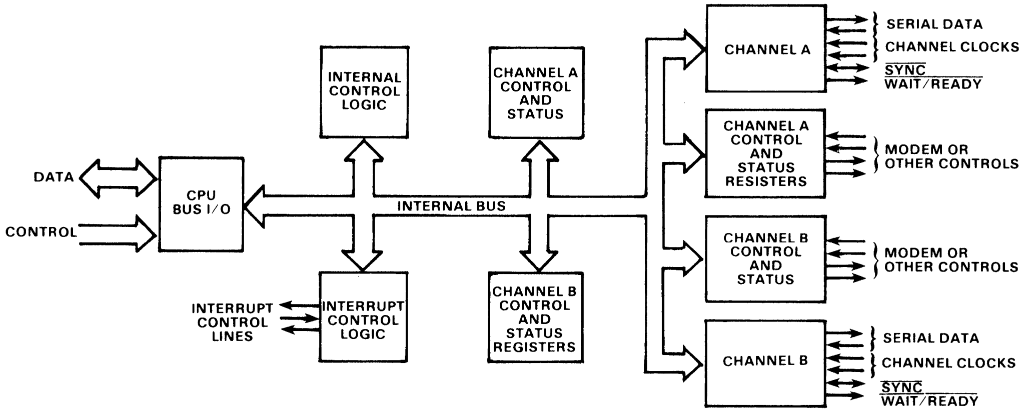
Figure 8 illustrates the conventional devices that the SIO replaces.

The first part of the following discussion covers SIO data-communication capabilities; the second part describes interactions between the CPU and the SIO.

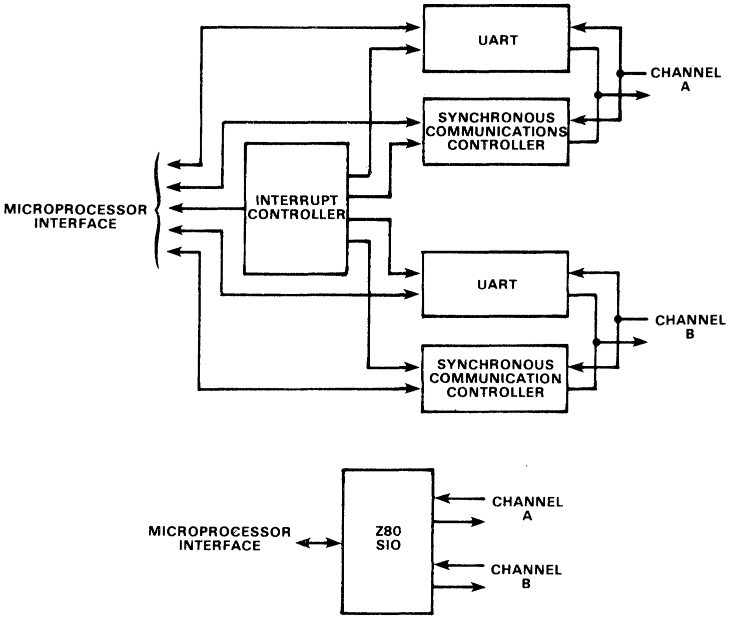
DATA COMMUNICATION CAPABILITIES

The SIO provides two independent full-duplex channels that can be programmed for use in any common asynchronous or synchronous data-communication

Block Diagram
Figure 7



Conventional Devices Replaced by the Z80 SIO
Figure 8



protocol. Figure 9 illustrates some of these protocols. The following is a short description of them. A more detailed explanation of these modes can be found in the MK3884 Z80 SIO Technical Manual.

Asynchronous Modes. Transmission and reception can be done independently on each channel with five to eight bits per character, plus optional even or odd parity. The transmitters can supply one, one-and-a-half or two stop bits per character and can provide a break output at any time. The receiver break-detection logic interrupts the CPU both at the start and end of a received break. Reception is protected from spikes by a transient spike-rejection mechanism that checks the signal one-half a bit time after a Low level is detected on the receive data input (RxDA or RxDB in Figure 5). If the Low does not persist—as in the case of a transient—the character assembly process is not started.

Framing errors and overrun errors are detected and buffered together with the partial character on which they occurred. Vectored interrupts allow fast servicing of error conditions using dedicated routines. Furthermore, a built-in checking process avoids interpreting a framing error as a new start bit: a framing error results in the addition of one-half a bit time to the point at which the search for the next start bit is begun.

The SIO does not require symmetric transmit and receive clock signals—a feature that allows it to be used with MK3882 Z80 CTC or many other clock sources. The transmitter and receiver can handle data at a rate of 1, 1/16, 1/32 or 1/64 of the clock rate supplied to the receive and transmit clock inputs. In asynchronous modes, the SYNC pin may be programmed as an input that can be used for functions such as monitoring a ring indicator.

Synchronous Modes. The SIO supports both byte-oriented and bit-oriented synchronous communication. Synchronous byte-oriented protocols can be handled in several modes that allow character synchronization with an 8-bit sync character (Monosync), any 16-bit sync pattern (Bisync) or with an external sync signal. Leading sync characters can be removed without interrupting the CPU.

Five-, six- or seven-bit sync characters are detected with 8- or 16-bit patterns in the SIO by overlapping the larger pattern across multiple in-coming sync characters, as shown in Figure 10.

CRC checking for synchronous byte-oriented modes is delayed by one character time so the CPU may disable CRC checking on specific characters. This permits implementation of protocols such as IBM Bisync.
Both CRC-16 ($X^{16} + X^{15} + X^2 + 1$) and CCITT ($X^{16} + X^{12} + X^5 + 1$) error checking polynomials are supported. In all non-SDLC modes, the CRC generator is initialized to 0's; in SDLC modes, it is initialized to 1's. The SIO can be used for interfacing to peripherals such as hard--sectored floppy disk, but it cannot generate or check CRC for IBM-compatible soft-sectored disks. The SIO also provides a feature that automatically transmits CRC data when no other data is available for transmission. This allows very high-speed transmissions under

DMA control with no need for CPU intervention at the end of a message. When there is no data or CRC to send in synchronous modes, the transmitter inserts 8- or 16-bit sync characters regardless of the programmed character length.

The SIO supports synchronous bit-oriented protocols such as SDLC and HDLC by performing automatic flag sending, zero insertion and CRC generation. A special command can be used to abort a frame in transmission. At the end of a message the SIO automatically transmits the CRC and trailing flag when the transmit buffer becomes empty. If a transmit underrun occurs in the middle of a message, an external/status interrupt warns the CPU of this status change so that an abort may be issued. One to eight bits per character can be sent, which allows reception of a message with no prior information about the character structure in the information field of a frame.

The receiver automatically synchronizes on the leading flag of a frame in SDLC or HDLC, and provides a synchronization signal on the SYNC pin; an interrupt can also be programmed. The receiver can be programmed to search for frames addressed by a single byte to only a specified user-selected address or to a global broadcast address. In this mode, frames that do not match either the user-selected or broadcast address are ignored. The number of address bytes can be extended under software control. For transmitting data, an interrupt on the first received character or on every character can be selected. The receiver automatically deletes all zeroes inserted by the transmitter during character assembly. It also calculates and automatically checks the CRC to validate frame transmission. At the end of transmission, the status of a received frame is available in the status registers.

The SIO can be conveniently used under DMA control to provide high-speed reception or transmission. In reception, for example, the SIO can interrupt the CPU when the first character of a message is received. The CPU then enables the DMA to transfer the message to memory. The SIO then issues an end-of-frame interrupt and the CPU can check the status of the received message. Thus, the CPU is freed for other service while the message is being received.

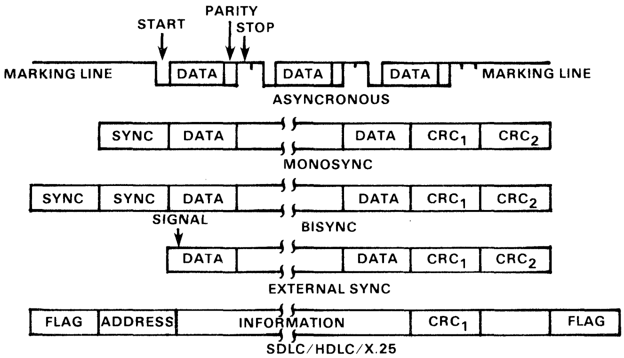
I/O INTERFACE CAPABILITIES

The SIO offers the choice of polling, interrupt (vectored or non-vectored) and block-transfer modes to transfer data, status and control information to and from the CPU. The block-transfer mode can also be implemented under DMA control.

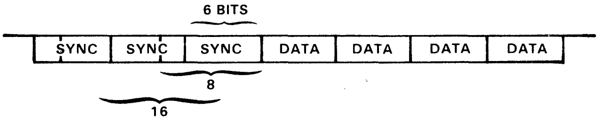
Polling. Two status registers are updated at appropriate times for each function being performed (for example, CRC error-status valid at the end of a message). When the CPU is operated in a polling fashion, one of the SIO's two status registers is used to indicate whether the SIO has some data or needs some data. Depending on the contents of this register, the CPU will either write data, read data, or just go on. Two bits in the register indicate that a data transfer is needed. In addition, error and other conditions are indicated. The second status register (special receive conditions) does not have to be read in a polling sequence, until a character has been received. All interrupt modes are disabled when operating the device in a polled environment.

Interrupts. The SIO has an elaborate interrupt scheme to provide fast interrupt service in real-time applications. A control register and a status register in Channel B contain the interrupt vector. When programmed to do

Z80 SIO PROTOCOLS
Figure 9



VARIABLE LENGTH SYNC CHARACTERS
Figure 10



so, the SIO can modify three bits of the interrupt vector in the status register so that it points directly to one of eight interrupt service routines in memory, thereby servicing conditions in both channels and eliminating most of the needs for a status-analysis routine.

Transmit interrupts, receive interrupts and external/status interrupts are the main sources of interrupts. Each interrupt source is enabled under program control, with Channel A having a higher priority than Channel B, and with receive, transmit and external/status interrupts prioritized in that order within each channel. When the transmit interrupt is enabled, the CPU is interrupted by the transmit buffer becoming empty. (This implies that the transmitter must have had a data character written into it so it can become empty.) The receiver can interrupt the CPU in one of two ways:

- Interrupt on first received character
- Interrupt on all received characters

Interrupt-on-first-received-character is typically used with the block-transfer mode. Interrupt-on-all-received-characters has the option of modifying the interrupt vector in the event of a parity error. Both of these interrupt modes will also interrupt under special receive conditions on a character or message basis (end-of-frame interrupt in SDLC, for example). This means that the special-receive condition can cause an interrupt only if the interrupt-on-first-received-character or interrupt-on-all-received-characters mode is selected. In interrupt-on-first-received-character, an interrupt can occur from special-receive conditions (except parity error) after the first-received-character interrupt (example: receive-overrun interrupt).

The main function of the external/status interrupt is to monitor the signal transitions of the Clear To Send ($\overline{\text{CTS}}$), Data Carrier Detect ($\overline{\text{DCD}}$) and Synchronization ($\overline{\text{SYNC}}$) pins (Figures 1 through 6). In addition, an external/status interrupt is also caused by a CRC-sending condition or by the detection of a break sequence (asynchronous mode) or abort sequence (SDLC mode) in the data stream. The interrupt caused by the break/abort sequence allows the SIO to interrupt when the break/abort sequence is detected or terminated. This feature facilitates the proper termination of the current message, correct initialization of the next message, and the accurate timing of the break/abort condition in external logic.

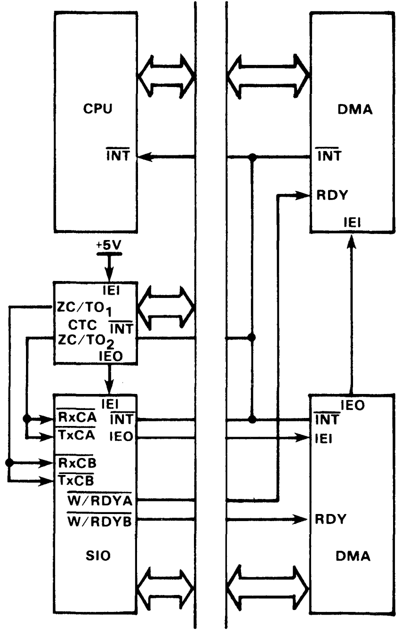
In a Z80 CPU environment (Figure 11), SIO interrupt vectoring is "automatic": the SIO passes its internally-modifiable 8-bit interrupt vector to the CPU, which adds an additional 8 bits from its interrupt-vector (I) register to form the memory address of the interrupt-routine table. This table contains the address of the beginning of the interrupt routine itself. The process entails an indirect transfer of CPU control to the interrupt routine, so that the next instruction executed after an interrupt acknowledge by the CPU is the first instruction of the interrupt routine itself.

CPU/DMA Block Transfer. The SIO's block-transfer mode accommodates both CPU block transfers and DMA controllers (Z80 DMA or other designs). The block-transfer mode uses the Wait/Ready output signal, which is selected with three bits in an internal control register. The Wait/Ready output signal can be programmed $\overline{\text{WAIT}}$ line in the CPU block-transfer mode or as a READY line in the DMA block-transfer mode.

To a DMA controller, the SIO $\overline{\text{READY}}$ output indicates that the SIO is ready to transfer data to or from memory. To the CPU, the $\overline{\text{WAIT}}$ output indicates that the SIO is not ready to transfer data, thereby requesting the CPU to extend the I/O cycle.

TYPICAL Z80 ENVIRONMENT

Figure 11



ARCHITECTURE

DESCRIPTION

The internal structure of the device includes a Z80 CPU interface, internal control and interrupt logic, and two full-duplex channels. Each channel contains its own set of control and status (write and read) registers, and control and status logic that provides the interface to modems or other external devices.

The registers for each channel are designated as follows:

WR0-WR7 — Write Registers 0 through 7
RR0-RR2 — Read Registers 0 through 2

The register group includes five 8-bit control registers, two sync-character registers and two status registers. The interrupt vector is written into an additional 8-bit register (Write Register 2) in Channel B that may be read through another 8-bit register (Read Register 2) in Channel B. The bit assignment and functional grouping of each register is configured to simplify and organize the programming process. Table 1 lists the functions assigned to each read or write register.

Read Register Functions

RR0	Transmit/Receive buffer status, interrupt status and external status
RR1	Special Receive Condition status
RR2	Modified interrupt vector (Channel B only)

Write Register Functions

WR0	Register pointers, CRC initialize, initialization commands for the various modes, etc.
WR1	Transmit/Receive interrupt and data transfer mode definition.
WR2	Interrupt vector (Channel B only)
WR3	Receive parameters and control
WR4	Transmit/Receive miscellaneous parameters and modes
WR5	Transmit parameters and controls
WR6	Sync character or SDLC address field
WR7	Sync character or SDLC flag

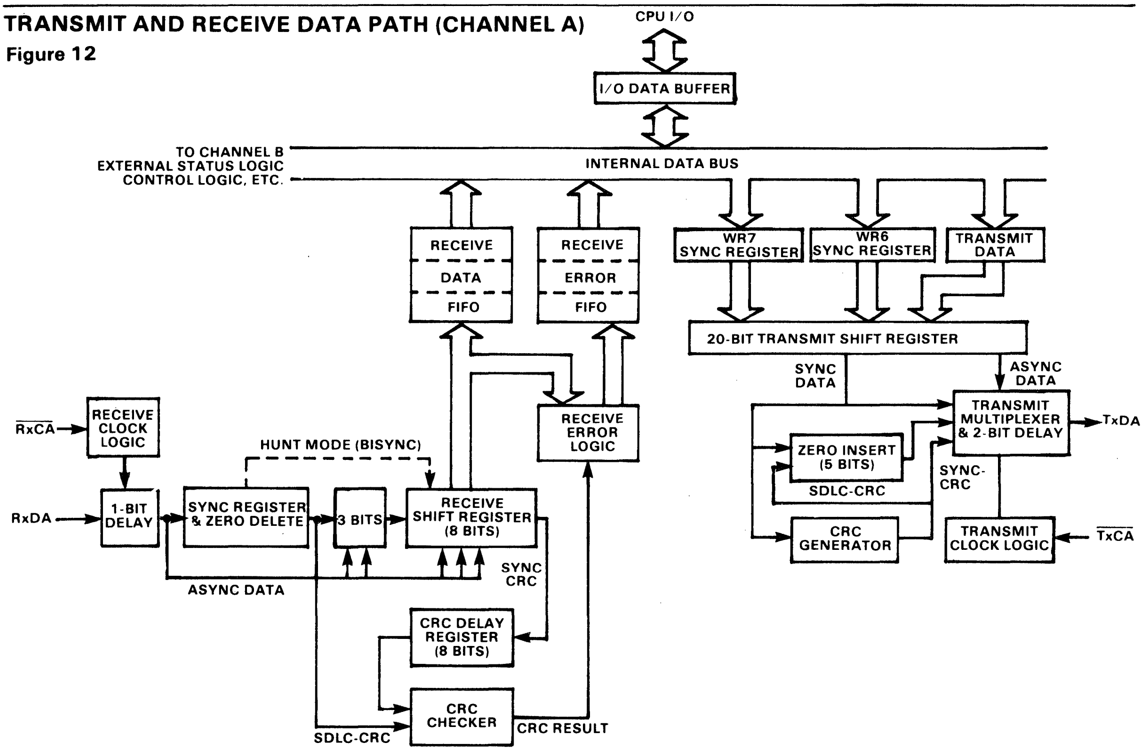
The logic for both channels provides formats, synchronization and validation for data transferred to and from the channel interface. The modem control inputs, Clear To Send ($\overline{CTS}$) and Data Carrier Detect ($\overline{DCD}$), are monitored by the external control and status logic under program control. All external control-and-status-logic signals are general-purpose in nature and can be used for functions other than modem control.

Data Path. The transmit and receive data path illustrated for Channel A in Figure 12 is identical for both channels. The receiver has three 8-bit buffer registers in a FIFO arrangement, in addition to the 8-bit receive shift register. This scheme creates additional time for the CPU to service an interrupt at the beginning of a block of high-speed data. Incoming data is routed through one of several paths (data or CRC) depending on the selected mode and—in asynchronous modes—the character length.

The transmitter has an 8-bit transmit data buffer register that is loaded from the internal data bus, and a 20-bit transmit shift register that can be loaded from the sync-character buffers or from the transmit data register. Depending on the operational mode, outgoing data is routed through one of four main paths before it is transmitted from the Transmit Data output (TxD).

The system program first issues a series of commands that initialize the basic mode of operation and then other commands that qualify conditions within the selected

TRANSMIT AND RECEIVE DATA PATH (CHANNEL A)
Figure 12



mode. For example, the asynchronous mode, character length, clock rate, number of stop bits, even or odd parity might be set first; then the interrupt mode; and finally, receiver or transmitter enable.

Both channels contain registers that must be programmed via the system program prior to operation. The channel-select input (B/$\overline{A}$) and the control/data input (C/$\overline{D}$) are the command-structure addressing controls, and are normally controlled by the CPU address bus. Figures 15 and 16 illustrate the timing relationships for programming the write registers and transferring data and status.

Read Registers. The SIO contains three read registers for Channel B and two read registers for Channel A (RR0-RR2 in Figure 13) that can be read to obtain the status information; RR2 contains the internally--modifiable interrupt vector and is only in the Channel B register set. The status information includes error conditions, interrupt vector and standard communications-interface signals.

To read the contents of a selected read register other than RR0, the system program must first write the pointer byte to WR0 in exactly the same way as a write register operation. Then, by executing a read instruction, the contents of the addressed read register can be read by the CPU.

The status bits of RR0 and RR1 are carefully grouped to simplify status monitoring. For example, when the interrupt vector indicates that a Special Receive Condition interrupt has occured, all the appropriate error bits can be read from a single register (RR1).

Write Registers. The SIO contains eight write registers for Channel B and seven write registers for Channel A (WR0-WR7 in Figure 14) that are programmed separately to configure the functional personality of the channels; WR2 contains the interrupt vector for both channels and is only in the Channel B register set. With the exception of WR0, programming the write registers require two bytes. The first byte is to WR0 and contains three bits (D_0-D_2) that point to the selected register; the second byte is the actual control word that is written into the register to configure the SIO.

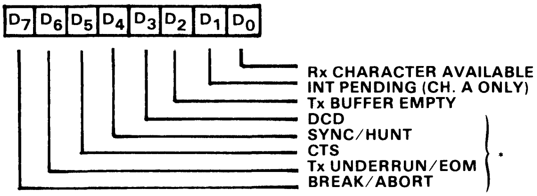
WR0 is a special case in that all of the basic commands can be written to it with a single byte. Reset (internal or external) initializes the pointer bits D_0-D_2 to point to WR0. This implies that a channel reset must always point to WR0.

The SIO must have the same clock as the CPU (same phase and frequency relationship, not necessarily the same driver).

READ REGISTER BIT FUNCTIONS

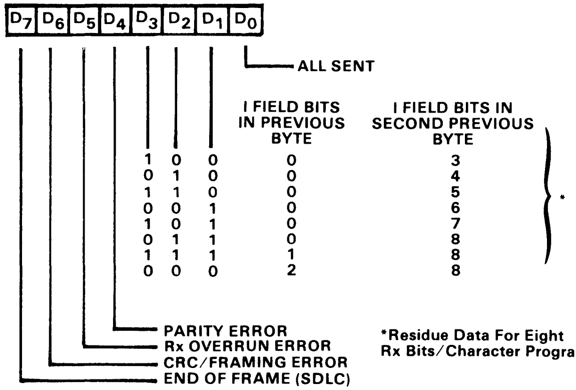
Figure 13

READ REGISTER 0



*Used With External/Status Interrupt Mode

READ REGISTER 1

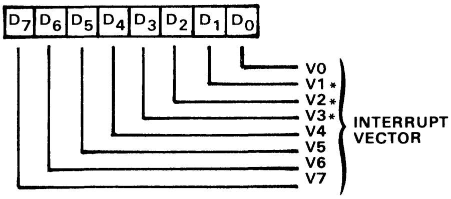


			I FIELD BITS IN PREVIOUS BYTE	I FIELD BITS IN SECOND PREVIOUS BYTE
1	0	0	0	3
0	1	0	0	4
1	1	0	0	5
0	0	1	0	6
1	0	1	0	7
0	1	1	0	8
1	1	1	1	8
0	0	0	2	8

*Residue Data For Eight Rx Bits/Character Programmed

Used With Special Receive Condition Mode

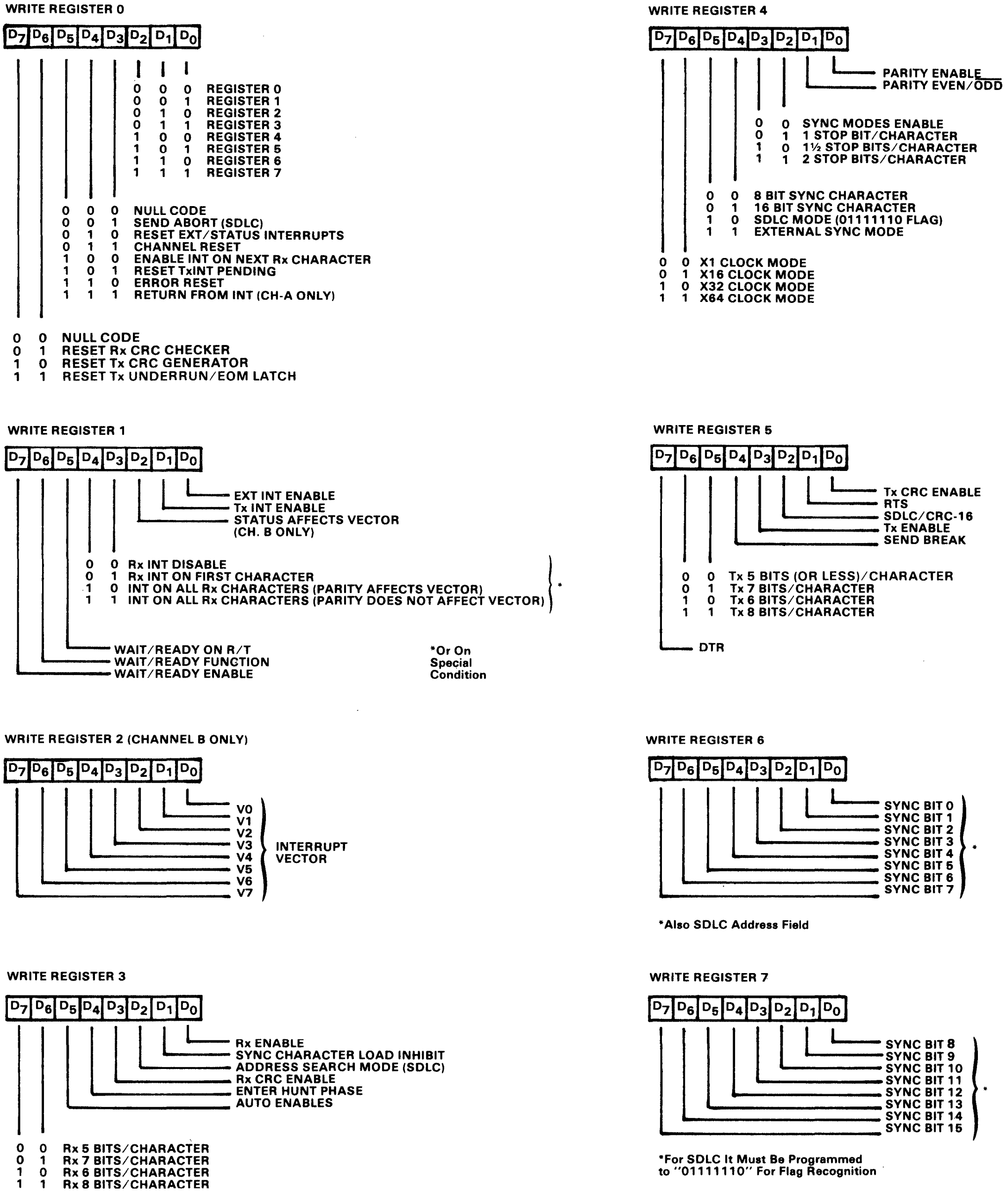
READ REGISTER 2



*Variable if Status Affects Vector is Programmed

WRITE REGISTER BIT FUNCTIONS

Figure 14



WRITE REGISTER 0

D7 D6 D5 D4 D3 D2 D1 D0

D2	D1	D0	
0	0	0	REGISTER 0
0	0	1	REGISTER 1
0	1	0	REGISTER 2
0	1	1	REGISTER 3
1	0	0	REGISTER 4
1	0	1	REGISTER 5
1	1	0	REGISTER 6
1	1	1	REGISTER 7

D5	D4	D3	
0	0	0	NULL CODE
0	0	1	SEND ABORT (SDLC)
0	1	0	RESET EXT/STATUS INTERRUPTS
0	1	1	CHANNEL RESET
1	0	0	ENABLE INT ON NEXT Rx CHARACTER
1	0	1	RESET TxINT PENDING
1	1	0	ERROR RESET
1	1	1	RETURN FROM INT (CH-A ONLY)

D7	D6	
0	0	NULL CODE
0	1	RESET Rx CRC CHECKER
1	0	RESET Tx CRC GENERATOR
1	1	RESET Tx UNDERRUN/EOM LATCH

WRITE REGISTER 1

D7 D6 D5 D4 D3 D2 D1 D0

- D0: EXT INT ENABLE
- D1: Tx INT ENABLE
- D2: STATUS AFFECTS VECTOR (CH. B ONLY)

D4	D3	
0	0	Rx INT DISABLE
0	1	Rx INT ON FIRST CHARACTER
1	0	INT ON ALL Rx CHARACTERS (PARITY AFFECTS VECTOR) *
1	1	INT ON ALL Rx CHARACTERS (PARITY DOES NOT AFFECT VECTOR) *

- D5: WAIT/READY ON R/T
- D6: WAIT/READY FUNCTION
- D7: WAIT/READY ENABLE

*Or On Special Condition

WRITE REGISTER 2 (CHANNEL B ONLY)

D7 D6 D5 D4 D3 D2 D1 D0

- D0: V0
- D1: V1
- D2: V2
- D3: V3
- D4: V4
- D5: V5
- D6: V6
- D7: V7

V0–V7: INTERRUPT VECTOR

WRITE REGISTER 3

D7 D6 D5 D4 D3 D2 D1 D0

- D0: Rx ENABLE
- D1: SYNC CHARACTER LOAD INHIBIT
- D2: ADDRESS SEARCH MODE (SDLC)
- D3: Rx CRC ENABLE
- D4: ENTER HUNT PHASE
- D5: AUTO ENABLES

D7	D6	
0	0	Rx 5 BITS/CHARACTER
0	1	Rx 7 BITS/CHARACTER
1	0	Rx 6 BITS/CHARACTER
1	1	Rx 8 BITS/CHARACTER

WRITE REGISTER 4

D7 D6 D5 D4 D3 D2 D1 D0

- D0: PARITY ENABLE
- D1: PARITY EVEN/$\overline{\text{ODD}}$

D3	D2	
0	0	SYNC MODES ENABLE
0	1	1 STOP BIT/CHARACTER
1	0	1½ STOP BITS/CHARACTER
1	1	2 STOP BITS/CHARACTER

D5	D4	
0	0	8 BIT SYNC CHARACTER
0	1	16 BIT SYNC CHARACTER
1	0	SDLC MODE (01111110 FLAG)
1	1	EXTERNAL SYNC MODE

D7	D6	
0	0	X1 CLOCK MODE
0	1	X16 CLOCK MODE
1	0	X32 CLOCK MODE
1	1	X64 CLOCK MODE

WRITE REGISTER 5

D7 D6 D5 D4 D3 D2 D1 D0

- D0: Tx CRC ENABLE
- D1: RTS
- D2: SDLC/CRC-16
- D3: Tx ENABLE
- D4: SEND BREAK

D6	D5	
0	0	Tx 5 BITS (OR LESS)/CHARACTER
0	1	Tx 7 BITS/CHARACTER
1	0	Tx 6 BITS/CHARACTER
1	1	Tx 8 BITS/CHARACTER

- D7: DTR

WRITE REGISTER 6

D7 D6 D5 D4 D3 D2 D1 D0

- D0: SYNC BIT 0
- D1: SYNC BIT 1
- D2: SYNC BIT 2
- D3: SYNC BIT 3
- D4: SYNC BIT 4
- D5: SYNC BIT 5
- D6: SYNC BIT 6
- D7: SYNC BIT 7

SYNC BIT 0–7: *

*Also SDLC Address Field

WRITE REGISTER 7

D7 D6 D5 D4 D3 D2 D1 D0

- D0: SYNC BIT 8
- D1: SYNC BIT 9
- D2: SYNC BIT 10
- D3: SYNC BIT 11
- D4: SYNC BIT 12
- D5: SYNC BIT 13
- D6: SYNC BIT 14
- D7: SYNC BIT 15

SYNC BIT 8–15: *

*For SDLC It Must Be Programmed to "01111110" For Flag Recognition

Read Cycle. The timing signals generated by a Z80 CPU input instruction to read a data or status byte from the SIO are illustrated in Figure 15.

Write Cycle. Figure 16 illustrates the timing and data signals generated by a Z80 CPU output instruction to write a data or control byte into the SIO.

Interrupt-Acknowledge Cycle. After receiving an interrupt-- request signal from an SIO ($\overline{\text{INT}}$ pulled Low), the Z80 CPU sends an interrupt-acknowledge sequence ($\overline{\text{M1}}$ Low, and $\overline{\text{IORQ}}$ Low a few cycles later) as in Figure 17. The SIO contains an internal daisy-chained interrupt structure for prioritizing nested interrupts for the various functions of its two channels, and this structure can be used within an external user-defined daisy chain that prioritizes several peripheral circuits.

The IEI of the highest-priority device is terminated High. A device that has an interrupt pending or under service forces its IEO Low. For devices with no interrupt pending or under service, IEO = IEI.

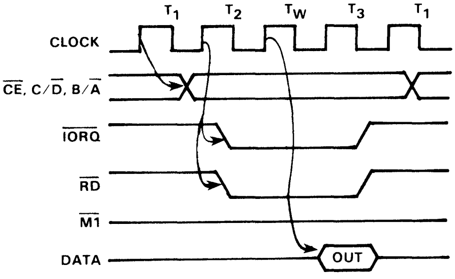
To insure stable conditions in the daisy chain, all interrupt status signals are prevented from changing while M1 is Low. When IORQ is Low, the highest priority interrupt requestor (the one with IEI High) places its interrupt vector on the data bus and sets its internal interrupt-under-service latch.

Return From Interrupt Cycle. Figure 18 illustrates the return from interrupt cycle. Normally, the Z80 CPU issues a RETI (return from interrupt) instruction at the end of an interrupt service routine. RETI is a 2-byte opcode (ED-4D) that resets the interrupt-under-service latch in the SIO to terminate the interrupt that has just been processed. This is accomplished by manipulating the daisy chain in the following way.

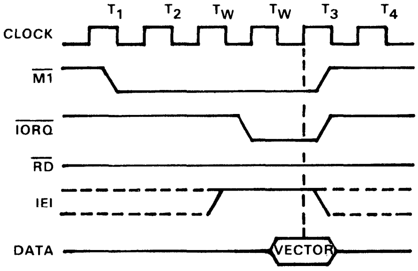
The normal daisy-chain operation can be used to detect a pending interrupt; however, it cannot distinguish between an interrupt under service and a pending unacknowledged interrupt of a higher priority. Whenever "ED" is decoded, the daisy chain is modified by forcing High the IEO of any interrupt that has not yet been acknowledged. Thus the daisy chain identifies the device presently under service as the only one with an IEI High and an IEO Low. If the next opcode byte is "4D", the interrupt-under-service latch is reset.

The ripple time of the interrupt daisy chain (both the High-to-Low and the Low-to-High transitions) limits the number of devices that can be placed in the daisy chain. Ripple time can be improved with carry-look-ahead, or by extending the interrupt-acknowledge cycle. For further information about techniques for increasing the number of daisy-chained devices, refer to the MK3880 Z80 CPU Product Specification.

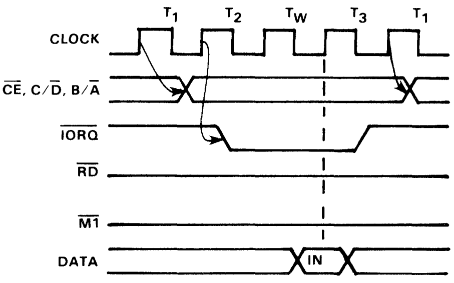
READ CYCLE
Figure 15



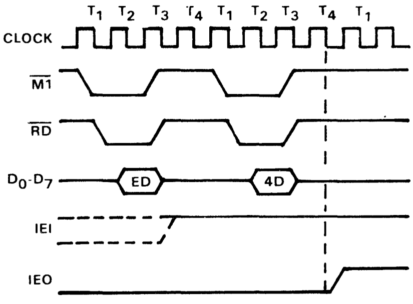
INTERRUPT ACKNOWLEDGE CYCLE
Figure 17



WRITE CYCLE
Figure 16



RETURN FROM INTERRUPT CYCLE
Figure 18



ABSOLUTE MAXIMUM RATINGS

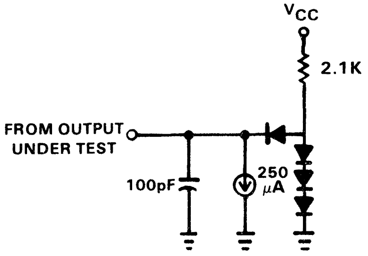
Voltages on all inputs and outputs with respect to GND . . . . –0.3V to +7.0V
Operating Ambient Temperature. . . . . .As Specified in Ordering Information
Storage Temperature . . . . –65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

STANDARD TEST CONDITIONS

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin. Standard conditions are as follows:

- $+4.75V \leq V_{CC} \leq +5.25V$
- GND = 0V
- T_A as specified in Ordering Information



All AC parameters assume a load capacitance of 100 pF max. Timing references between two output signals assume a load difference of 50 pF max.

DC CHARACTERISTICS

SYM	PARAMETER	MIN	MAX	UNIT	TEST CONDITION
V_{ILC}	Clock Input Low Voltage	-0.3	+0.80	V	
V_{IHC}	Clock Input High Voltage	V_{CC} -0.6	+5.5	V	
V_{IL}	Input Low Voltage	-0.3	+0.8	V	
V_{IH}	Input High Voltage	+2.0	+5.5	V	
V_{OL}	Output Low Voltage		+0.4	V	I_{OL} = 2.0mA
V_{OH}	Output High Voltage	+2.4		V	I_{OH} = -250 µA
I_{LI}	Input Leakage Current	-10	±10	µA	$0 < V_{IN} < V_{CC}$
I_Z	3-State Output/Data Bus Input Leakage Current	-10	+10	µA	$0 < V_{IN} < V_{CC}$
$I_{L(SY)}$	$\overline{\text{SYNC}}$ Pin Leakage Current	-40	+10	µA	$0 < V_{IN} < V_{CC}$
I_{CC}	Power Supply Current		100	mA	

Overall specified temperature and voltage range.

CAPACITANCE

SYM	PARAMETER	MIN	MAX	UNIT	TEST CONDITION
C	Clock Capacitance		40	pF	Unmeasured pins returned to ground
C_{IN}	Input Capacitance		10	pF	
C_{OUT}	Output Capacitance		10	pF	

Over specified temperature range; f = 1MHz

AC ELECTRICAL CHARACTERISTICS

See Figure 19

NUMBER	SYM	PARAMETER	MK3884		MK3884-4		UNIT
			MIN	MAX	MIN	MAX	
1	TcC	Clock Cycle Time	400	4000	250	4000	ns
2	TwCh	Clock Width (High)	170	2000	105	2000	ns
3	TfC	Clock Fall Time		30		30	ns
4	TrC	Clock Rise Time		30		30	ns
5	TwCl	Clock Width (Low)	170	2000	105	2000	ns
6	TsAD(C)	$\overline{CE}$, C/$\overline{D}$, B/$\overline{A}$ to Clock ↑ Setup Time	160		145		ns
7	TsCS(C)	$\overline{IORQ}$, $\overline{RD}$ to Clock ↑ Setup Time	240		115		ns
8	TdC(DO)	Clock ↑ to Data Out Delay		240		220	ns
9	TsDI(C)	Data In to Clock ↑ Setup (Write or M1 Cycle)	50		50		ns
10	TdRD(DOz)	$\overline{RD}$ ↑ to Data Out Float Delay		230		110	ns
11	TdIO(DOI)	$\overline{IORQ}$ ↓ to Data Out Delay (INTA Cycle)		340		160	ns
12	TsM1(C)	$\overline{M1}$ to Clock ↑ Setup Time	210		90		ns
13	TsIEI(IO)	IEI to $\overline{IORQ}$ ↓ Setup Time ($\overline{INTA}$ Cycle)	200		140		ns
14	TdM1(IEO)	$\overline{M1}$ ↓ to IEO ↓ Delay (interrupt before $\overline{M1}$)		300		190	ns
15	TdIEI(IEOr)	IEI ↑ to IEO ↑ Delay (after ED decode)		150		100	ns
16	TdIEI(IEOf)	IEI ↓ to IEO ↓ Delay		150		100	ns
17	TdC(INT)	Clock ↑ to $\overline{INT}$ ↓ Delay		200		200	ns
18	TdIO (W/RWf)	$\overline{IORQ}$ ↓ or $\overline{CE}$ ↓ to $\overline{W/RDY}$ ↓ Delay (Wait Mode)		300		210	ns
19	TdC (W/RR)	Clock ↑ to $\overline{W/RDY}$ ↓ Delay (Ready Mode)		120		120	ns
20	TdC (W/RWz)	Clock ↓ to $\overline{W/RDY}$ Float Delay (Wait Mode)		150		130	ns
21	Th	Any unspecified Hold when Setup is specified	0		0		ns

NOTE: Timings are referenced from minimum V_{IH} or maximum V_{IL} for inputs and from minimum V_{OH} or maximum V_{OL} for outputs.

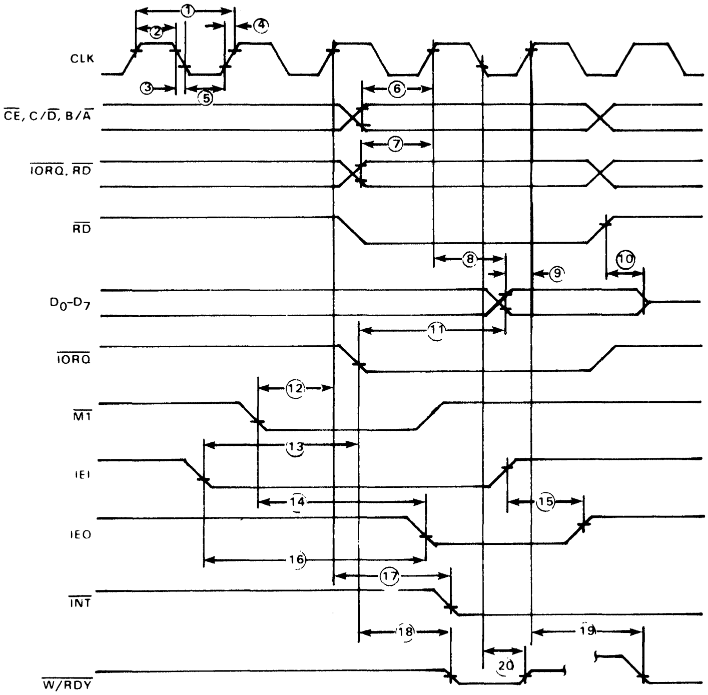
AC ELECTRICAL CHARACTERISTICS (continued)

See Figure 20

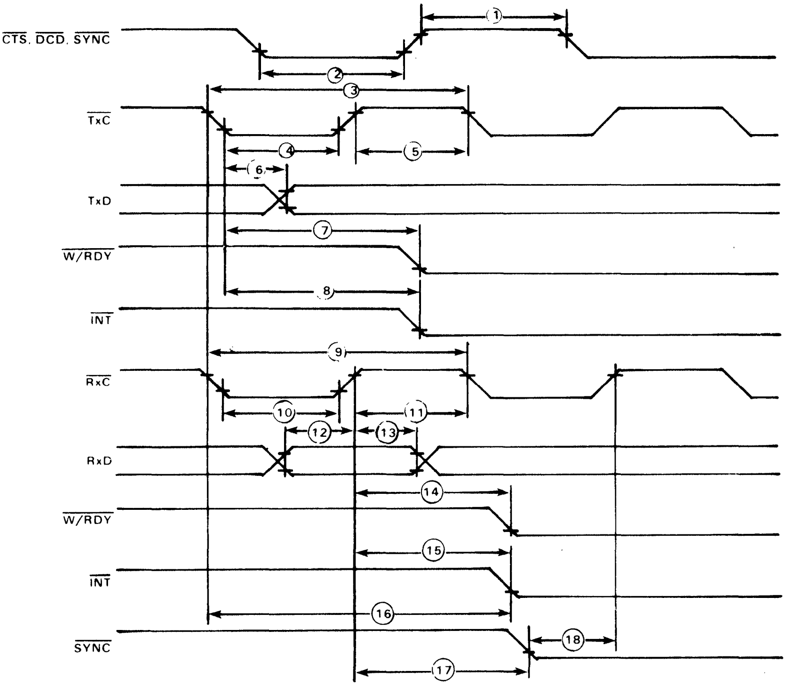
NUMBER	SYM	PARAMETER	MK3884		MK3884-4		UNIT
			MIN	MAX	MIN	MAX	
1	TwPh	Pulse Width (High)	200		200		ns
2	TwPl	Pulse Width (Low)	200		200		ns
3	TcTxC	$\overline{\text{TxC}}$ Cycle Time	400	∞	400	∞	ns
4	TwTxCl	$\overline{\text{TxC}}$ Width (Low)	180	∞	180	∞	ns
5	TwTxCh	$\overline{\text{TxC}}$ Width (High)	180	∞	180	∞	ns
6	TdTxC(TxD)	$\overline{\text{TxC}}$ ↓ to TxD Delay (x1 Mode)		400		300	ns
7	TdTxC (W/RRf)	$\overline{\text{TxC}}$ ↓ to $\overline{\text{W/RDY}}$ ↓ Delay (Ready Mode)	5	9	5	9	Clk Periods*
8	TdTxC(INT)	$\overline{\text{TxC}}$ ↓ to $\overline{\text{INT}}$ ↓ Delay	5	9	5	9	Clk Periods*
9	TcRxC	$\overline{\text{RxC}}$ Cycle Time	400	∞	400	∞	ns
10	TwRxCl	$\overline{\text{RxC}}$ Width (Low)	180	∞	180	∞	ns
11	TwRxCh	$\overline{\text{RxC}}$ Width (High)	180	∞	180	∞	ns
12	TsRxD(RxC)	RxD to $\overline{\text{RxC}}$ ↑ Setup Time (x1 Mode)	0		0		ns
13	ThRxD(RxC)	$\overline{\text{RxC}}$ ↑ to RxD Hold time (x1 Mode)	140		140		ns
14	TdRxC (W/RRf)	$\overline{\text{RxC}}$ ↑ to $\overline{\text{W/RDY}}$ ↓ Delay (Ready Mode)	10	13	10	13	Clk Periods*
15	TdRxC(INT)	$\overline{\text{RxC}}$ ↑ to $\overline{\text{INT}}$ ↓ Delay	10	13	10	13	Clk Periods*
16	TdTxC(INT)	$\overline{\text{TxC}}$ ↓ to $\overline{\text{INT}}$ ↓ Delay	5	9	5	9	Clk Periods*
17	TdRxC (SYNC)	$\overline{\text{RxC}}$ ↑ to $\overline{\text{SYNC}}$ ↓ Delay (Output Modes)	4	7	4	7	Clk Periods*
18	TsSYNC (RxC)	$\overline{\text{SYNC}}$ ↓ to $\overline{\text{RxC}}$ ↑ Setup (External Sync Modes)	-100		-100		ns

In all modes, the System Clock rate must be at least five times the maximum data rate.
RESET must be active a minimum of one complete Clock Cycle.
*System Clock

AC ELECTRICAL CHARACTERISTICS
Figure 19



AC ELECTRICAL CHARACTERISTICS
Figure 20



ORDERING INFORMATION

PART NO.	PACKAGE TYPE	MAX CLOCK FREQUENCY	TEMPERATURE RANGE
MK3884N Z80-SIO/0	Plastic	2.5MHz	0°C to +70°C
MK3884J Z80-SIO/0	CERDIP	2.5MHz	0°C to +70°C
MK3884P Z80-SIO/0	Ceramic	2.5MHz	0°C to +70°C
MK3884N-10 Z80-SIO/0	Plastic	2.5MHz	-40°C to +85°C
MK3884J-10 Z80-SIO/0	CERDIP	2.5MHz	-40°C to +85°C
MK3884P-10 Z80-SIO/0	Ceramic	2.5MHz	-40°C to +85°C
MK3884N-4 Z80A-SIO/0	Plastic	4MHz	0°C to +70°C
MK3884J-4 Z80A-SIO/0	CERDIP	4MHz	0°C to +70°C
MK3884P-4 Z80A-SIO/0	Ceramic	4MHz	0°C to +70°C
MK3885N Z80-SIO/1	Plastic	2.5MHz	0°C to +70°C
MK3885J Z80-SIO/1	CERDIP	2.5MHz	0°C to +70°C
MK3885P Z80-SIO/1	Ceramic	2.5MHz	0°C to +70°C
MK3885N-10 Z80-SIO/1	Plastic	2.5MHz	-40°C to +85°C
MK3885J-10 Z80-SIO/1	CERDIP	2.5MHz	-40°C to +85°C
MK3885P-10 Z80-SIO/1	Ceramic	2.5MHz	-40°C to +85°C
MK3885N-4 Z80A-SIO/1	Plastic	4MHz	0°C to +70°C
MK3885J-4 Z80A-SIO/1	CERDIP	4MHz	0°C to +70°C
MK3885P-4 Z80A-SIO/1	Ceramic	4MHz	0°C to +70°C
MK3887N Z80-SIO/2	Plastic	2.5MHz	0°C to +70°C
MK3887J Z80-SIO/2	CERDIP	2.5MHz	0°C to +70°C
MK3887P Z80-SIO/2	Ceramic	2.5MHz	0°C to +70 C
MK3887N-10 Z80-SIO/2	Plastic	2.5MHz	-40°C to +85°C
MK3887J-10 Z80-SIO/2	CERDIP	2.5MHz	-40°C to +85°C
MK3887P-10 Z80-SIO/2	Ceramic	2.5MHz	-40°C to +85°C
MK3887N-4 Z80A-SIO/2	Plastic	4MHz	0°C to +70°C
MK3887J-4 Z80A-SIO/2	CERDIP	4MHz	0°C to +70°C
MK3887P-4 Z80A-SIO/2	Ceramic	4MHz	0°C to +70°C

NOTE: Refer to the section on pin descriptions for explanation of the differences between the MK3884, MK3885, and MK3887.

MOSTEK®

Z80 MICROCOMPUTER DEVICES

MK3886 COMBO CHIP

FEATURES

□ 256 x 8 static RAM - 64 bytes of which can operate in write protected or low power standby mode

□ Two programmable timers which operate from an independent clock source.

□ Three external interrupt channels with programmable vector for each channel.

□ Serial I/O port - synchronous or asynchronous operation with end of word interrupt

□ Z80 compatible daisy chain interrupt structure

□ Single 5 volt supply (±5%)

GENERAL DESCRIPTION

The MK3886 Combo chip is a Z80 microprocessor peripheral containing a combination of features that enables a user to have great flexibility with a single component. The chip contains 256 bytes of RAM, two timers, a serial Input/Output port, and three external interrupt inputs. Additionally, the interrupt vector and priority circuitry can be software programmable to configure the chip for special user requirements. The Combo chip utilizes N-channel silicon gate depletion load technology and is packaged in a 40-pin DIP.

FUNCTIONAL PIN DESCRIPTION

D_7 - D_0. CPU Data Bus (bi-directional, tri-state). This eight bit bus is used to transfer data and control information between the MK3886 and CPU. D0 is the least significant bit.

A_7 - A_0. Address Input Bus (input, active high). The eight address input lines are connected to the CPU address bus and are used to select either the RAM memory address or an I/O port.

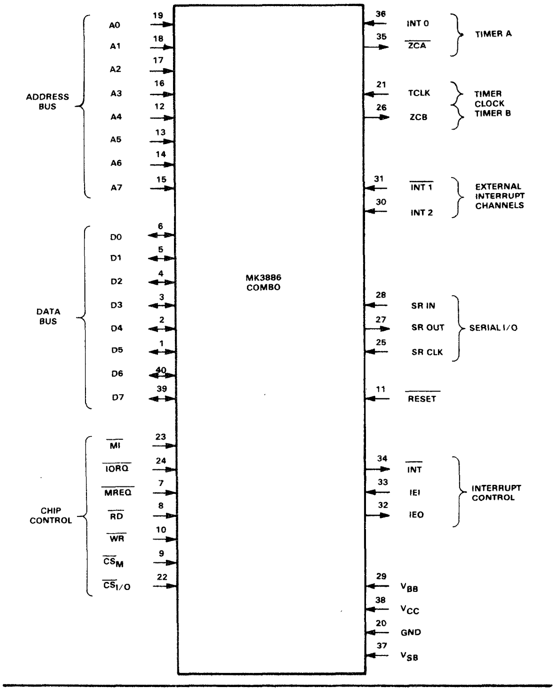
$\overline{M1}$. Machine Cycle One from CPU (input, active low). This signal from the CPU is used as a sync pulse during an interrupt acknowledge cycle. When $\overline{M1}$ is active and $\overline{IORQ}$ is active, the CPU is acknowledging an interrupt.

$\overline{IORQ}$. Input/Output Request from Z80-CPU (input, active low). The $\overline{IORQ}$ signal is used in conjunction with the $\overline{CS}_{I/O}$, $\overline{RD}$, and $\overline{WR}$ signals to transfer commands and data between the CPU and the Combo Chip. When $\overline{CS}_{I/O}$, $\overline{RD}$, and $\overline{IORQ}$ are active, the contents of the port addressed from address inputs A3-A0 will be placed on the data bus (a read operation). When $\overline{CS}_{I/O}$, $\overline{WR}$, and $\overline{IORQ}$ are active, the port addressed by address inputs A3-A0 will be loaded with the contents of the data bus. Also, if $\overline{IORQ}$ and $\overline{M1}$ are active simultaneously, the CPU is acknowledging an interrupt. During this time, the interrupting device within the Combo Chip will place its interrupt vector on the CPU data bus if it is the highest priority device requesting an interrupt.

$\overline{MREQ}$. Memory Request from CPU (input, active low). The $\overline{MREQ}$ signal is used in conjunction with the $\overline{CS}$, $\overline{RD}$, and $\overline{WR}$ signals to either read or write to the RAM location addressed from A_7 - A_0.

$\overline{RD}$. Read Cycle Status from the CPU (input, active low). An active $\overline{RD}$ in conjunction with $\overline{CS}_M$ and $\overline{MREQ}$ signals that a memory read is in progress. The MK3886 responds by placing the contents of the address RAM Location on the data bus. If $\overline{CS}_{I/O}$ and $\overline{IORQ}$ are active simultaneously with $\overline{RD}$, the MK3886 responds by placing the I/O port contents on the data bus.

PIN CONNECTIONS



$\overline{\text{WR}}$. Write Cycle Status (input, active low). An active $\overline{\text{WR}}$, $\overline{\text{CS}}_M$, and $\overline{\text{MREQ}}$ indicates that a memory write is in progress to the RAM. If an active $\overline{\text{WR}}$, $\overline{\text{CS}}_{I/O}$, and $\overline{\text{IORQ}}$ occur, it indicates an I/O write is in progress.

$\overline{\text{CS}}_M$. Chip Select RAM (input, active low). $\overline{\text{CS}}_M$ selects the 256 x 8 RAM for a Memory Read or Write cycle. $\overline{\text{CS}}_M$ is usually decoded from the CPU Address Bus.

$\overline{\text{CS}}_{I/O}$. Chip Select I/O (input, active low). $\overline{\text{CS}}_{I/O}$ selects the MK3886 for an I/O Read or Write cycle. $\overline{\text{CS}}_{I/O}$ is usually decoded from the CPU Address Bus.

$\overline{\text{INT}}$. Interrupt Request (output, open drain, active low). This signal is active low to signal the CPU whenever an interrupt is pending from one of the seven interrupt channels within the MK3886.

IEI. Interrupt Enable In (input, active high). This signal is used to help form an interrupt priority daisy chain when more than one peripheral device in the system has interrupting capability. A high level on this pin indicates that no other interrupting devices of higher priority in the daisy chain are requesting interrupt service from the CPU or being serviced by an interrupt subroutine.

IEO. Interrupt Enable Out (output, active high). The IEO signal, in conjunction with IEI, is used to form an interrupt priority daisy chain. IEO is high only if IEI is high and an interrupt is not being requested from any Combo channel.

$\overline{\text{INT1}}$, INT2. External Interrupt lines. These two edge triggered interrupt lines are uncommitted to system activity and can be used to flag the CPU as desired by external activity. $\overline{\text{INT1}}$ generates an interrupt on a negative edge whereas INT2 generates an interrupt on a positive edge transition.

TCLK. Timer Clock (input). This signal phase clock is used as an input to the two timers on the MK3886. The TCLK input may be asynchronous with respect to the Read and Write control signals.

SRIN. Serial Data Input (input). This signal is the input data line to the Combo chip's serial port.

SROUT. Serial Data Output (output). This signal is the output data line from the serial port.

SRCLK. Serial Port clock (input). This is the clock used to derive the shift register clock in order to shift data into or out of the serial port.

INTO. External Interrupt 0 (input, programmed as active high or low). INTO is used in conjunction with Timer A to operate in the Pulse Width Measurement Mode or Event Counter Mode. It can also be used as an External Interrupt input and can be programmed to generate an interrupt on either a positive or negative edge transition. Because this interrupt is referenced from TCLK, TCLK must be present to allow interrupts.

$\overline{\text{ZCA}}$. Zero Count Out (output, active low). This signal is a zero count output which pulses low for an integral multiple of Timer clock cycles. The number of clock cycles is a function of the selected prescale value.

ZCB. Zero Count Out (output, active high). The Zero Count output pin is toggled every time Timer B counts to zero. This produces a square wave which is half the time out frequency. This output can be connected to SRCLK to drive the serial port. This pin can also be programmed via timer B to provide asymetrical output wave forms.

$\overline{\text{RESET}}/\overline{\text{RAMPRT}}$. $\overline{\text{RESET}}$ (input, active low). $\overline{\text{RESET}}$ disables all interrupts, masks all interrupts, stops both timers, and prevents any memory access from occuring. It also forces SROUT and ZCA to a logical 1 output condition. ZCB is forced to a logical zero condition. The $\overline{\text{RAMPRT}}$ is the RAM protect control signal. When it is brought low, the RAM is disabled and therefore protected against any alternations.

V_{BB}. Substrate Decoupling. This pin allows a .01 microfarad capacitor to be tied to the substrate to allow additional substrate decoupling when powering V_{CC} up and down.

V_{SB}. Low Power Standby Supply. This voltage is used to power the 64 bytes of low power RAM. The voltage can vary from 3.2 to 5.6 volts D.C. and is independent of V_{CC}.

MK3886 ARCHITECTURE

A block diagram of the MK3886 Combo chip is shown in Figure 1. The internal structure of the Combo chip consists of a latched CPU bus interface, internal control logic, 256 bytes of random access memory, serial I/O port, two timers, and three external interrupt channels. This part is not restricted for use only in Z80 CPU systems but can easily be adapted to other bus-oriented systems.

The internal control logic of the Combo chip receives and decodes the control sequences to be performed from the CPU bus. Three types of access cycles may occur: memory read/write, I/O read/write, and interrupt acknowledge. None of these operations are dependent upon the timer clock, TCLK, but rather on timing conditions present on the control input lines. The TCLK input is only used to drive the two timers. The control logic and I/O ports can be addressed via ten registers. These internal registers serve to configure the chip for proper operation and provide a means for exchange of control and data information between the CPU and Combo chip.

Four individual microcomputer component features can be identified within the Combo Chip. These features are Read/Write memory, counting and timing channels, serial input/output, and external interrupt inputs. These combinations enable the MK3886 to function as a key element in a minimum component Z80 system.

The memory consists of 256 bytes of static RAM. The lower 64 bytes have two special features. Write protection and standby power. The write protection protects this memory from undesired write operations. The low power standby RAM provides a method to preserve important data during a loss of system power.

Two versatile software programmable timers are provided. The Timer clock (TCLK) input is used by both timers to provide an accurate time base. In addition to the zero count output on both timers, Timer A has an external interrupt input linked directly to its control circuitry which provides two additional timer modes.

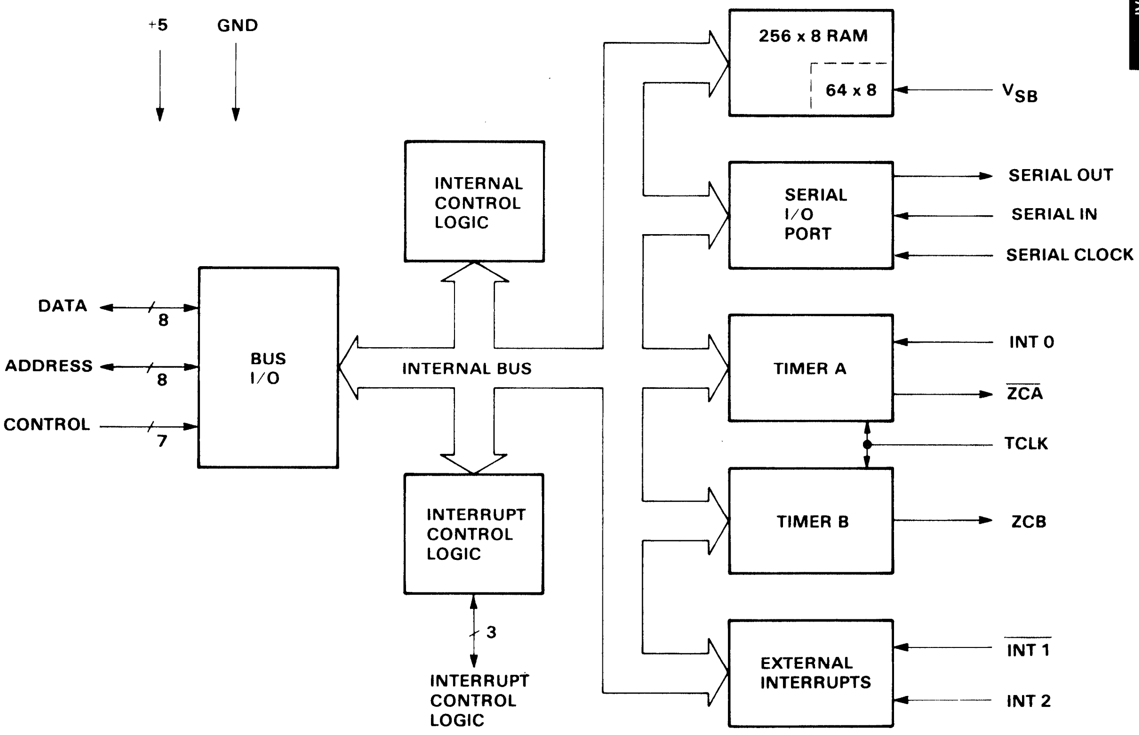
The serial port allows input and output of serial data in either asynchronous or synchronous modes. The port is basically a 16 bit shift register that can be read from or written to while the data is being shifted at a rate determined by the external serial clock. This port can be used to provide serial data communications or to interface to external serial logic such as shift registers or serial memories (CCD).

Three external interrupt lines are provided in order to allow prioritized, vectored, maskable, edge triggered external interrupt inputs. All interrupt lines are TTL compatible with Schmitt trigger buffered input circuits.

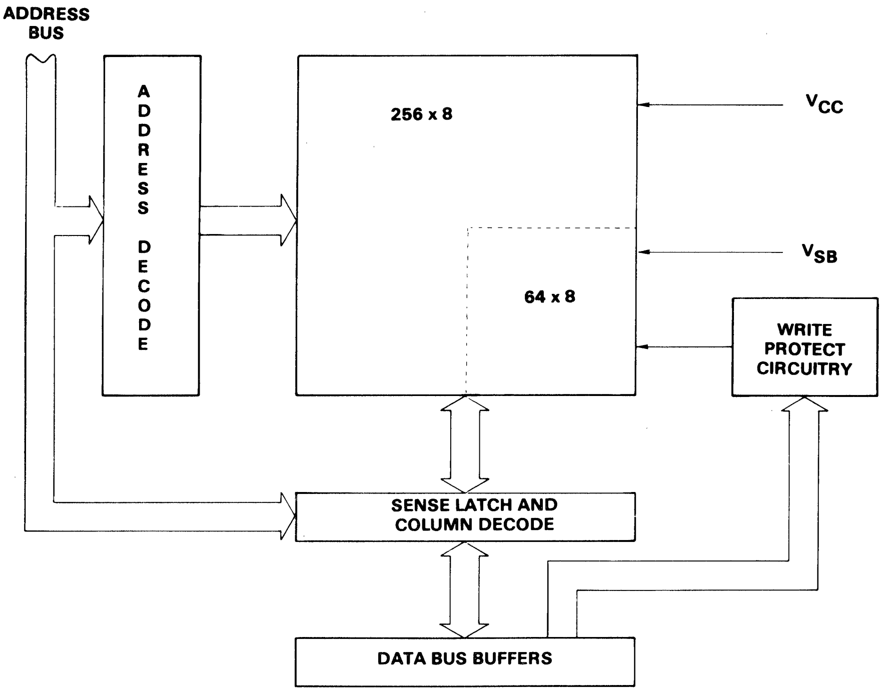
The interrupt control logic section handles all CPU interrupt protocol for nested priority interrupt structures. Priority is determined in two ways. First, an internal priority has been assigned for the 7 channels capable of generating an interrupt. This priority is listed in section 7. Secondly, priority of any component device is determined by its physical location in a daisy chain configuration. Two lines are provided in each Combo chip to form this daisy chain with the device closest to the CPU having the highest priority.

The MK3886 requires only a single +5 volt supply. However, a second input, V_{SB}, is provided to supply power to the low power standby RAM. This supply input can be tied either to a battery back up supply or simply to V_{CC} if low power standby is not required.

MK3886 BLOCK DIAGRAM
Figure 1



BLOCK DIAGRAM OF 256 x 8 RAM
Figure 2



RANDOM ACCESS MEMORY

The MK3886 has 256 bytes of Edge Activated static Random Access Memory. Sixty-four bytes of this RAM can be configured to operate in a low power battery back-up mode when storage of critical data is imperative. Also included with these 64 bytes is a write protection circuit. This circuit is designed to permit only authorized write operations to this area of memory. A register programmable via the CPU enables and disables the write protection.

The static memory is a 256 x 8 array. The lower 8 address bus bits are decoded within the chip to provide addressing from 0 to 255. This memory can be decoded on 256 word memory blocks anywhere within a user's system memory map by using the memory chip select input $\overline{CS_M}$.

The lowest 64 bytes of the RAM (address 0-63) are both write protected and capable of maintaining data in a low power standby mode. This memory block has an independent power supply input and write control circuitry. If standby power is desired the standby power source input V_{SB} can be connected to a supply independent of the V_{CC} input. Three nickel-cadmium batteries (typical voltage = 3.75V) can be used in a trickle charge circuit to provide power. The power requirements are such that less than 8.2mW are required at a minimum 3.2V standby supply voltage.Power supply tolerance is not critical. The memory will function over a range of 3.2V to 5.5 volts. If low power standby is not desired, V_{SB} should be tied to V_{CC}, the Combo chip supply voltage.

TIMERS A AND B

The MK3886 has two programmable timers, designated as Timer A and Timer B. Each timer consists of an 8-bit binary Down Counter, a programmable Prescaler, a Timer Constant Register, and a Zero Count output. There are some differences between Timer A and B. These differences basically are due to the fact that the INTO line is tied directly to Timer A's control circuitry. This feature gives Timer A three modes of operation: Interval Timer, Pulse-Width Measurement, and Event Counter. Timer B operates in the Interval Timer Mode only. A basic block diagram which applies to both Timers is illustrated in Figure 3. The following description highlights the major functions of each block.

The Channel Control Register (8-bit) and Logic is written to by the CPU to select the modes and parameters of the channel. Within the Combo chip there are two such registers, corresponding to Channel A and Channel B. Each register is assigned a separate I/O port address selected by the state of the lower four address inputs, (A_3 - A_0). For specific I/O port assignments for each Timer, refer to the section entitled "Combo Programming Summary."

Both Timer A and Timer B have a Prescaler which can be programmed via the appropriate Timer Control Register to divide its input TCLK, by a predetermined prescale value. The output of the Prescaler is then used to clock the Down Counter in the Interval Timer Mode of operation or in the Pulse-Width Measurement Mode of operation. (Pulse-Width Measurement on Timer A only.)

The Time Constant Register is an 8-bit register used in all count modes and is programmed by the CPU with an integer time constant value of 0 through 255. Under normal operation, this register is loaded into the Down counter when the Time Constant Register is first written to and is reloaded automatically thereafter whenever the Down Counter counts to zero.

The Down Counter is an 8-bit register used in all count modes. The Down Counter is decremented by the INTO edge in the Event Counter Mode (Timer A only), or by the clock output of the prescaler in the Interval Timer Mode. At any time, the contents of the Down Counter can be read by performing an I/O READ of the port address assigned to the Down Counter of the selected Timer channel. Either timer channel may be programmed to generate an interrupt request sequence each time the zero count condition in the Down counter is reached.

SERIAL I/O

The MK3886 contains circuitry to perform serial data transfers into and out of the unit under program control. The serial port will allow input and output of either asynchronous or synchronous serial data. The port consists of a 16-bit shift register that can be read from or written to while data is being shifted into or out of the shift register. This port, controlled by the CPU, can provide input (SRIN), output (SROUT), and clock (SRCLK) for the serial data.

The block diagram shown in Figure 4 depicts the functional operation of the Serial Port.

SHIFT REGISTER

The Shift Register in the serial port is 16 bits long and can be read or written to at any time. It is addressed by the CPU via two ports that access two 8 bit bytes. The Upper Shift Register is defined as Port 6 and is used to access the most significant 8 bits. The Lower Shift register is defined as Port 7 and is used to access the least significant 8 bits. Bit 0 is the lease significant bit of the Shift Register. It is used to form the serial output, SROUT. Bit 15 is the most significant bit of the Shift Register. It is the bit position where serial input data first enters the Shift Register.

INTERRUPTS

A total of three external interrupt inputs are provided in the MK3886. Two of these inputs are uncommitted, the third is associated with Timer A. These inputs are edge triggered with Schmitt trigger inputs that allow slow rise signals to be tied directly to the interrupt request lines. $\overline{\text{INT1}}$ will set an interrupt request latch on a negative edge transition whereas INT2 sets the latch on a positive edge. INTO can be programmed to interrupt on either transition.

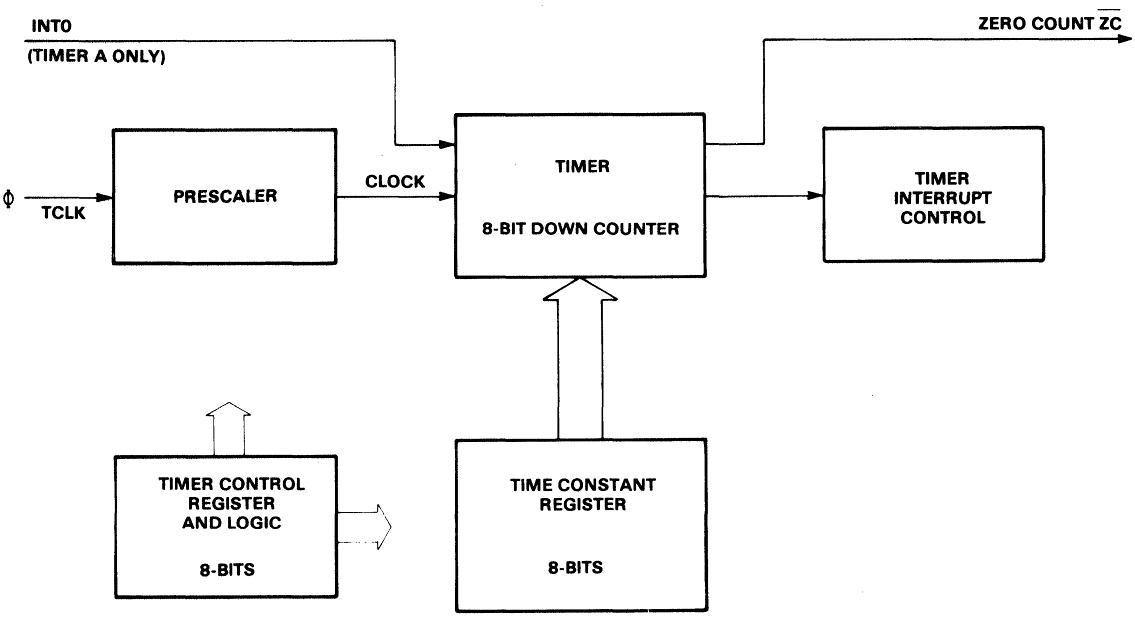
The Combo's interrupt control logic insures that it acts in accordance with Z80 system interrupt protocol for nested priority interrupt and proper return from interrupt. The priority of any Z80 peripheral is determined by its physical location in a daisy chain configuration. Two signal lines (IEI and IEO) are provided on the Combo and all Z80 peripheral devices to form the system daisy chain. The device closest to the CPU has the highest priority; within the Combo, interrupt priority has been pre-detemined for each interrupt source. The following is the internally assigned priority in decreasing order:

1. INTO
2. Timer A
3. Timer B
4. End of Word - Receive
5. End of Word - Transmit
6. $\overline{\text{INT1}}$
7. INT2

Each of the interrupt channels on the MK3886 can be individually enabled and each provides a unique interrupt vector to the CPU. Additionally, a programmable mask register accessed via Port 9 allows the interrupt requests on each channel to be selectively blocked without disabling them.

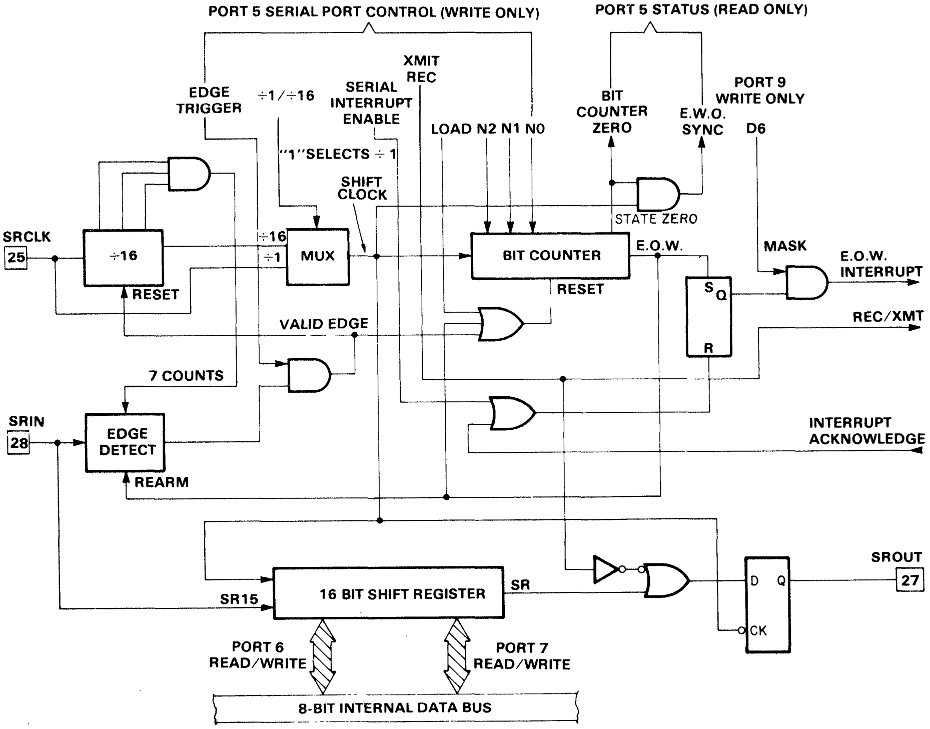
TIMER BLOCK DIAGRAM

Figure 3



PORT BLOCK DIAGRAM

Figure 4



COMBO PROGRAMMING SUMMARY

The following is a summary of the programmable ports on the MK3886. It is provided for use as a quick reference and review of the port channel assignments.

The most significant digit, D7, is on the far left and least significant digit, D0, on the far right of the register representation. An X indicates an unused bit position.

Port 0 - Timer A Control - Write Only ADDRESS: '00'H

÷20	÷5	÷2	T4	T3	T2	T1	TO
7	6	5	4	3	2	1	0 ← Bit No.

Event Counter Mode	0 0 0
÷2 Prescale	0 0 1
÷5 Prescale	0 1 0
÷10 Prescale	0 1 1
÷20 Prescale	1 0 0
÷40 Prescale	1 0 1
÷100 Prescale	1 1 0
÷200 Prescale	1 1 1

TO - Start/Stop Timer
T1 - Pulse Width/Internal Timer
T2 - INTO Active Level
T3 - INTO Ext. Int. Enable
T4 - Timer A Int. Enable

PORT 1 - Timer A Time Constant Register - Write Only
ADDRESS: '01'H

TC7	TC6	TC5	TC4	TC3	TC2	TC1	TCO
7	6	5	4	3	2	1	0 ← Bit No.

PORT 1 - Timer A Down Counter - Read Only
ADDRESS: '01'H

7	6	5	4	3	2	1	0 ← Bit No.

PORT 2 - Timer B Control - Write Only
ADDRESS: '02'H

PSI	PSO	X	X			X	X
7	6	5	4	3	2	1	0 ← Bit No.

÷2 Prescale -	0 0
÷5 Prescale -	0 1
÷10 Prescale -	1 0
÷20 Prescale -	1 1

2 → Timer B Interrupt Enable
3 → Start/Stop Timer B

PORT 3 - Timer B Counter - Read/Write
ADDRESS: '03'H

7	6	5	4	3	2	1	0 ← Bit No.

PORT 4 - Timer B Time Constant Register - Write Only
ADDRESS: '04'H

TC7	TC6	TC5	TC4	TC3	TC2	TC1	TCO
7	6	5	4	3	2	1	0 ← Bit No.

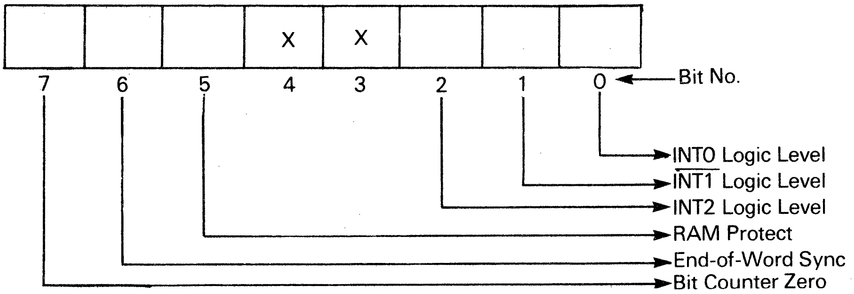
PORT 5 - Serial Port Control - Write Only
ADDRESS: '05'H

			X		N2	N1	NO
7	6	5	4	3	2	1	0 ← Bit No.

3 → Serial Port/ Interrupt Enable
5 → Edge Trigger
6 → XMIT/RCV
7 → ÷1/ ÷16

N2	N1	NO	Shift Word Length
0	0	0	4
0	0	1	7
0	1	0	8
0	1	1	9
1	0	0	10
1	0	1	11
1	1	0	12
1	1	1	16

PORT 5 - Status - Read Only
ADDRESS: '05'H



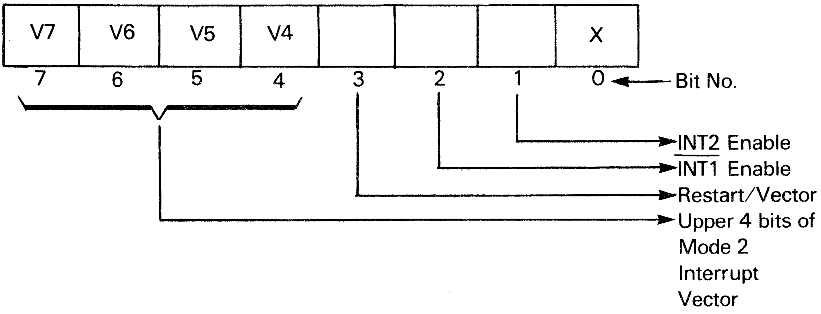
PORT 6 - Upper Shift Register - Read/Write
ADDRESS: '06'H

SR15	SR14	SR13	SR12	SR11	SR10	SR9	SR8

PORT 7 - Lower Shift Register - Read/Write
ADDRESS: '07'H

SR7	SR6	SR5	SR4	SR3	SR2	SR1	SR0

PORT 8 - Interrupt Control and Vector Register - Write
Only ADDRESS: '08'H



V7
V6
V5
V4
X
7
6
5
4
3
2
1
0
Bit No.
INT2 Enable
$\overline{\text{INT1}}$ Enable
Restart/Vector
Upper 4 bits of Mode 2 Interrupt Vector

If the Vector Mode is selected, the following interrupt vector is returned during the interrupt acknowledge cycle.

V7	V6	V5	V4	I2	I1	I0	0	
7	6	5	4	3	2	1	0	← Bit No.
				1	1	1	0	- INT 0
				1	1	0	0	- Timer A
				1	0	1	0	- Timer B
				1	0	0	0	- E.O.W. Receive
				0	1	1	0	- E.O.W. Transmit
				0	1	0	0	- $\overline{\text{INT1}}$
				0	0	1	0	- INT2

Supplied from PORT 8

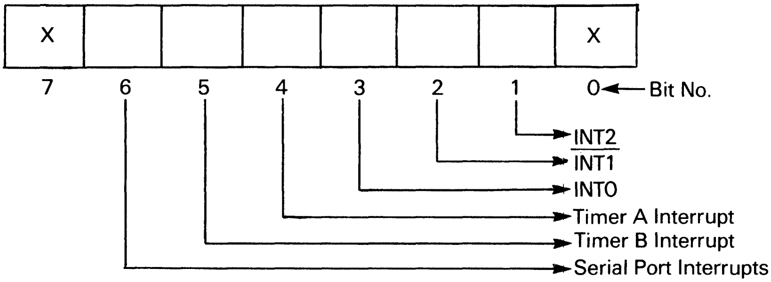
If the restart mode is selected, a RESTART instruction is returned to the CPU during the interrupt acknowledge cycle in the following format.

1	1	I2	I1	I0	1	1	1	
7	6	5	4	3	2	1	0	← Bit No.

Where:

Z80 Mnemonic	12	11	10	SOURCE	RESTART LOCATION
RST 56	1	1	1	INTO	38H
RST 48	1	1	0	Timer A	30H
RST 40	1	0	1	Timer B	28H
RST 32	1	0	0	End of Word Receive	20H
RST 24	0	1	1	End of Word Transmit	18H
RST 16	0	1	0	$\overline{\text{INT1}}$	10H
RST 08	0	0	1	INT2	08H

PORT 9 - Interrupt Mask Register - Write Only
ADDRESS: '09'H



PORT A - Write Protect - Write Only
ADDRESS: '0A'H

WP7	WP6	WP5	WP4	WP3	WP2	WP1	WPO

Write one byte: 55_H
Write Uninhibited: 66_H
Write Inhibit: Any value except 55_H or 66_H
All registers are cleared when $\overline{\text{RESET}}$ occurs.
The following is a list in descending order of the priority of each interrupting channel. The highest priority is first.

1. INT O
2. TIMER A
3. TIMER B
4. End of Word - Recieve
5. End of Word - Transmit
6. $\overline{\text{INT 1}}$
7. INT 2

ELECTRICAL SPECIFICTIONS

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias	Specified Operating Range
Storage Temperature	-65°C to +150°C
Voltage on Any Pin With Respect to Ground	-0.3 V to +7 V
Power Dissipation	0.8W

Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = 5V ± 5% unless otherwise specified

SYM	PARAMETER	MIN	MAX	UNIT	TEST CONDITION
V_{ILC}	Clock Input Low Voltage	-0.3	0.8	V	
V_{IHC}	Clock Input High Voltage	2.2	V_{CC} + .3	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 2.0mA
V_{OL}	Output High Voltage	2.4		V	I_{OH} = -250μA
I_{CC}	Power Supply Current		120	mA	Outputs Open
I_{LI}	Input Leakage Current		±10	μA	V_{IN} = 0 to V_{CC}
I_{LOH}	Tri-State Output Leakage Current in Float		10	μA	V_{OUT} = 2.4 to V_{CC}.
I_{LOL}	Tri-State Output Leakage Current in Float		-10	μA	V_{OUT} = 0.4V
V_{SB}	Standby V_{CC} for RAM	3.2	5.5	V	
I_{SB}	Standby Current		6.0 3.7	mA mA	V_{RAM} = 5.5V V_{RAM} = 3.2V

NOTE: All voltages are referenced to ground.

CAPACITANCE

T_A = 25°C

SYM	PARAMETER	MAX	UNIT	TEST CONDITION
C_{IN}	Input Capacitance	10	pF	Unmeasured Pins
C_{OUT}	Output Capacitance	10	pF	Returned to Ground

A.C. CHARACTERISTICS

T_A= 0°C to 70°C, V_C = +5V ± 5V unless otherwise noted.

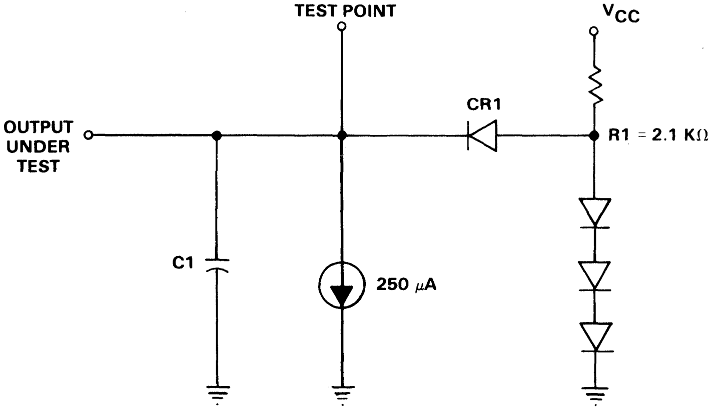
SIGNAL	SYMBOL	PARAMETER	MK3886		MK3886-4		UNIT	CONDITIONS
			MIN	MAX	MIN	MAX		
A_7-A_0	$t_{SM}(A)$	Address Setup time to falling edge of $\overline{MREQ}$	100		45		n.s.	
	$t_{HM}(A)$	Address Hold time from falling edge of $\overline{MREQ}$	40		40		n.s.	
	$t_{SI}(A)$	Address Setup time to falling edge of $\overline{IORQ}$	100		45		n.s.	
	$t_{HI}(A)$	Address Hold time from falling edge of $\overline{IORQ}$	40		40		n.s.	
$\overline{CS}_m$	$t_{SM}(\overline{CS}_M)$	$\overline{CS}_M$ setup time to falling edge of $\overline{MREQ}$	85		40		n.s.	
	$t_{HM}(\overline{CS}_M)$	$\overline{CS}_M$ Hold time from falling edge of $\overline{MREQ}$	40		40		n.s.	
$\overline{CS}_{IO}$	$t_{SI}(\overline{CS}_{IO})$	$\overline{CS}_{10}$ Setup time to falling edge of $\overline{IORQ}$	85		50		n.s.	
	$t_{HI}(\overline{CS}_{IO})$	$\overline{CS}_{IO}$ Hold time from falling edge of $\overline{IORQ}$	40		40		n.s.	
D_7-D_0	$t_{ACM}(D)$	Data Output Delay from $\overline{MREQ}$ during memory read		350		220	n.s.	Load=50pF + 1TTL Load
	$t_{ACI}(D)$	Data Output Delay from $\overline{IORQ}$ during I/O read		380		335	n.s.	
	$t_{ARM}(D)$	Data Output Delay from $\overline{RD}$ to data valid during memory cycle		200		200	n.s.	
	$t_{ARI}(D)$	Data Output Delay from $\overline{RD}$ to data valid during I/O cycle		380		335	n.s.	
	$t_{SI}(D)$	Data Setup time to rising $\overline{IORQ}$ or $\overline{WR}$ during I/O $\overline{WRITE}$	265		265		n.s.	
	$t_{HI}(D)$	Data hold time from rising $\overline{IORQ}$ or $\overline{WR}$ during I/O $\overline{WRITE}$	30		30		n.s.	
	$t_{SM}(D)$	Data Setup time to rising $\overline{MREQ}$ or $\overline{WR}$ during memory $\overline{WRITE}$	200		110		n.s.	
	$t_{HM}(D)$	Data hold time from rising $\overline{MREQ}$ or $\overline{WR}$ during memory $\overline{WRITE}$	30		30		n.s.	
	$t_{DI}(D)$	Data Output delay from falling $\overline{IORQ}$ during interrupt acknowledge		340		340	n.s.	
	$t_F(D)$	Delay to float		110			n.s.	
$\overline{WR}$	$t_{WI}(WR)$	$\overline{WR}$ pulse width low (I/O cycle)	250		250		n.s.	
	$t_{WM}(WR)$	$\overline{WR}$ pulse width low (Mem cycle)	100		100		n.s.	
$\overline{MI}$	$t_{SI}(M1)$	$\overline{M1}$ Setup time to falling $\overline{IORQ}$ during interrupt acknowledge	250		250		n.s.	
IEI	$t_S(IEI)$	Setup to falling $\overline{IORQ}$ During interrupt acknowledge	200		140		n.s.	
IEO	$t_{DH}(IO)$	IEO Delay Time from rising edge of IEI		160		160	n.s.	Load=50pF + 1TTL Load
	$t_{DL}(IO)$	IEO Delay Time from falling edge of IEI		130		130	n.s.	
	$t_{DM}(IO)$	IEO Delay from falling edge of $\overline{M1}$ (interrupt just prior to $\overline{M1}$)		190		190	n.s.	
	$t_D(IO)$	Delay to rising IEO from rising $\overline{IORQ}$ during interrupt acknowledge		-60			n.s.	

SIGNAL	SYMBOL	PARAMETER	MK3886		MK3886-4		UNIT	CONDITIONS
			MIN	MAX	MIN	MAX		
$\overline{\text{MREQ}}$	t_W(MRL)	Pulse Width, $\overline{\text{MREQ}}$ Low	480		300		n.s.	
	t_W(MRH)	Pulse Width, $\overline{\text{MREQ}}$ High	190		160		n.s.	
	t_{DX}(IT)	Delay to falling $\overline{\text{INT}}$ from external interrupt active transition		900		900	n.s.	
$\overline{\text{INT}}$	t_{DT}(IT)	Delay to $\overline{\text{INT}}$ from Timer Interrupt at rising TCLK		600		600	n.s.	
	t_{DSRC}(IT)	Delay to $\overline{\text{INT}}$ from rising edge of SRCLK		900		900	n.s.	
ZCA	t_{WC}(ZCA)	Width of ZCA pulse (max)		P* t_{CLK} - $t_{CLK/2}$ - 100 n.s.				
SRCLK	t_C(SR)	Period of SRCLK	3.3		3.3		µs.	
	t_S(SI)	SRIN setup with respect to SRCLK edge	250		250		n.s.	
	$t_{DSR(SRC)}$	Delay to SROUT from SRCLK		2.5		1.9	µs.	
TCLK	t_C(TC)	Period of Timer Clock	400		250		n.s.	
	t_W(TC)	Timer Clock Low Time	180	220	110	140	n.s.	

NOTE: All AC parameters assume a load capacitance of 100 pF max. Timing references between two output signals assume a load difference of 50 pF max. Timings are referenced from minimum V_{IH} or maximum V_{IL} for inputs and from minimum V_{OH} or maximum V_{OL} for outputs.

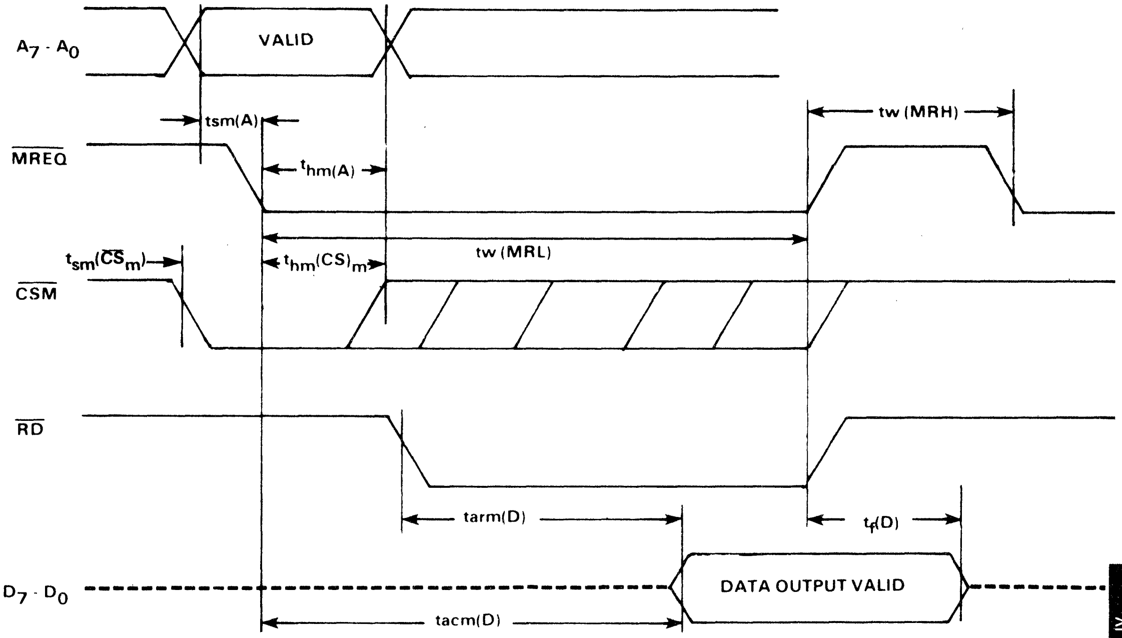
LOAD CIRCUIT FOR OUTPUT

Figure 5



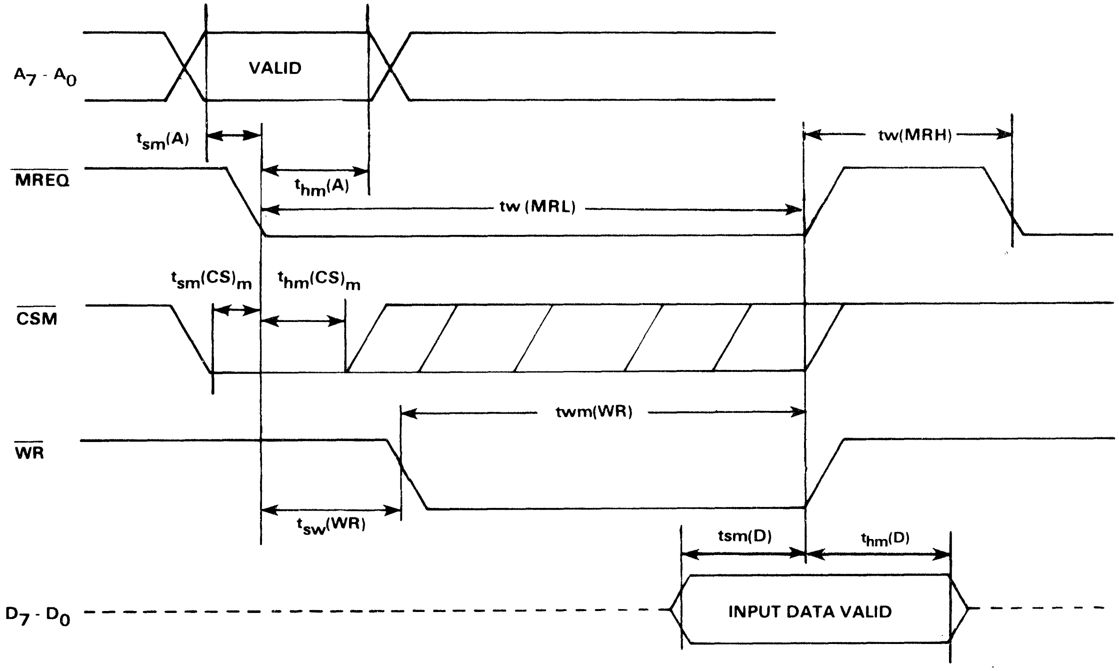
MEMORY READ TIMING

Figure 6



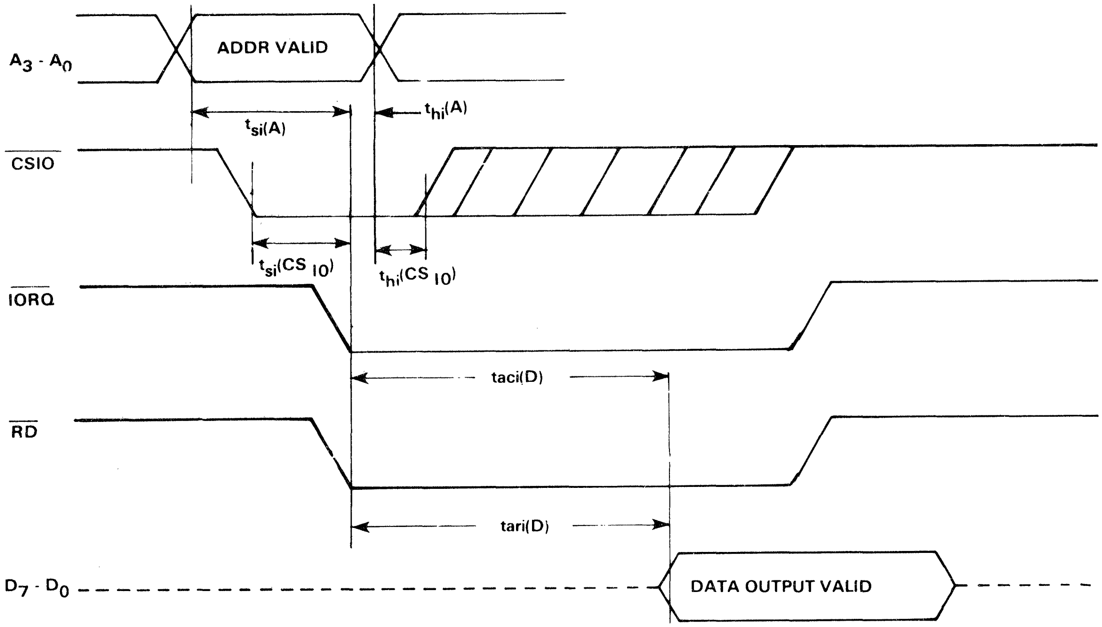
MEMORY WRITE TIMING

Figure 7



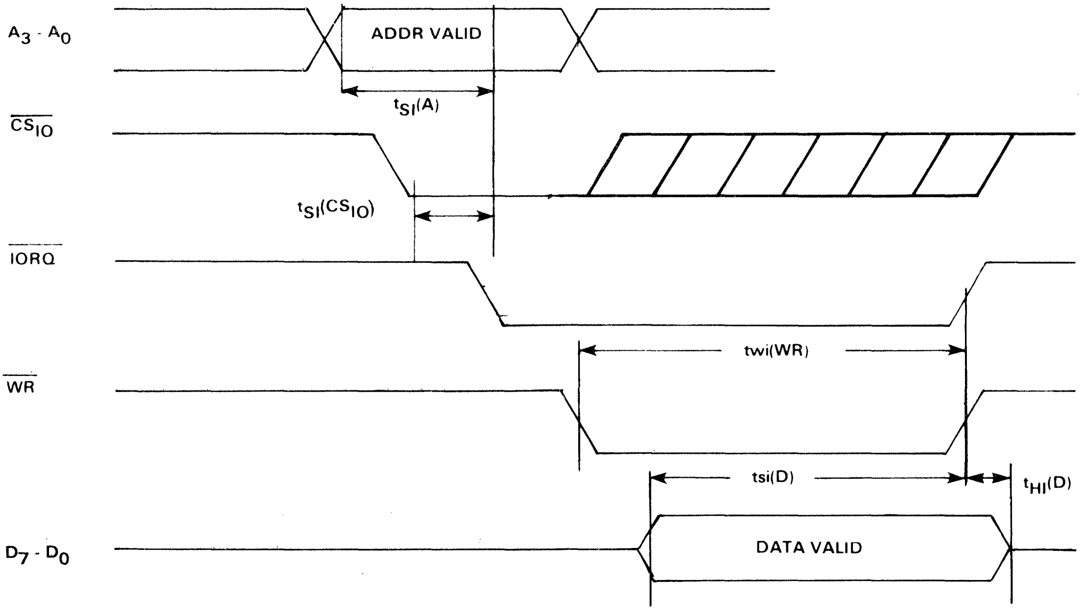
I/O READ TIMING

Figure 8



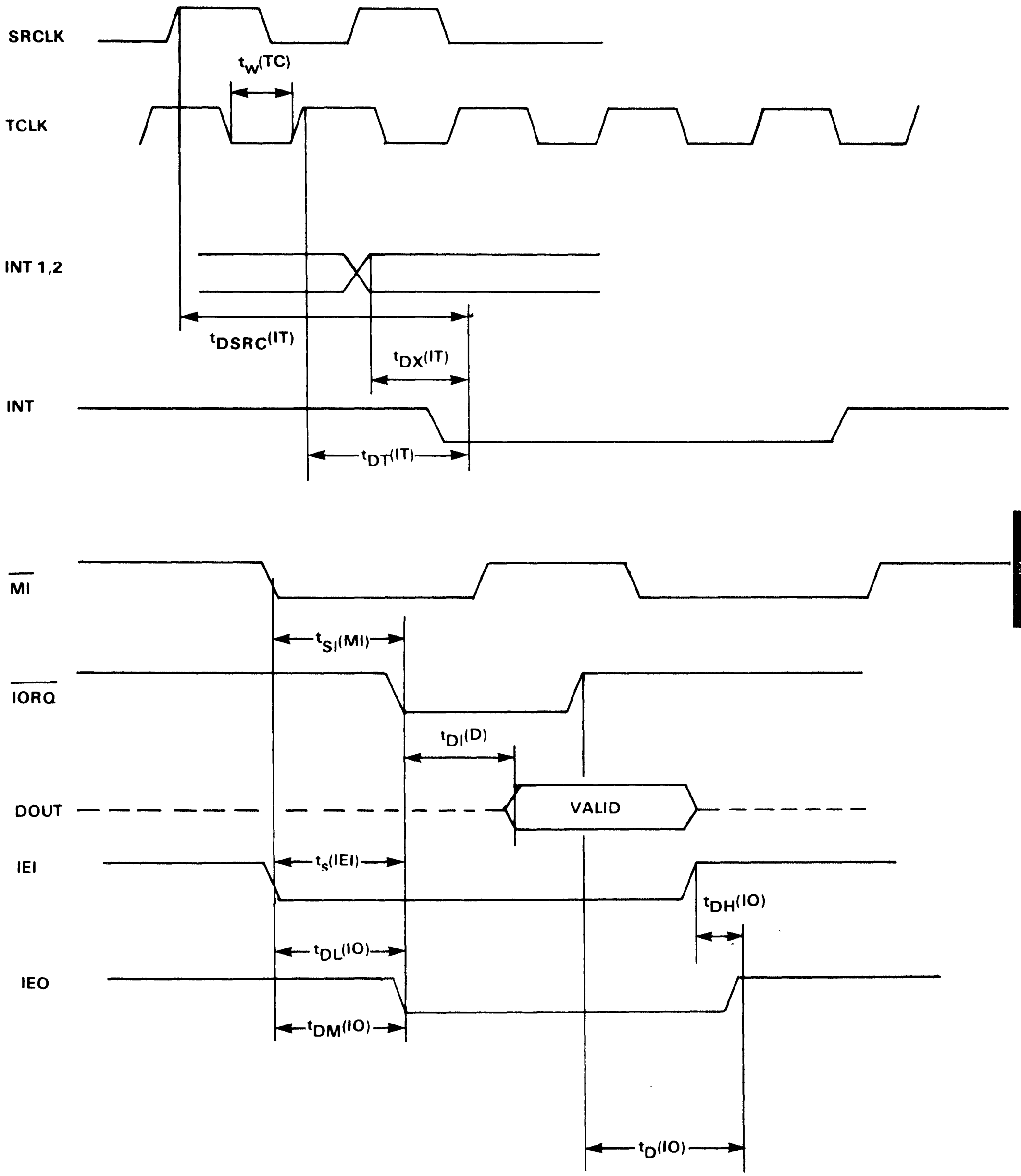
I/O WRITE TIMING

Figure 9



INTERRUPT RESPONSE TIMING

Figure 10



ORDERING INFORMATION

PART	DESIGNATOR	PACKAGE TYPE	MAXIMUM CLOCK FREQUENCY	TEMPERATURE RANGE
MK3886N	Z80-COMBO	PLASTIC	2.5 MHz	0° to 70°C
MK3886P	Z80-COMBO	CERAMIC	2.5 MHz	0° to 70°C
MK3886J	Z80-COMBO	CERDIP	2.5 MHz	0° to 70°C
MK3886N-4	Z80-COMBO	PLASTIC	4.0 MHz	0° to 70°C
MK3886P-4	Z80-COMBO	CERAMIC	4.0 MHz	0° to 70°C
MK3886J-4	Z80-COMBO	CERDIP	4.0 MHz	0° to 70°C

1981 Z80 MICROCOMPUTER DATA BOOK

I Table of Contents

I TABLE OF CONTENTS

II General Information

II GENERAL INFORMATION

III Z80 Family Technical Manuals

III Z80 FAMILY TECHNICAL MANUALS

IV Z80 Family Data Sheets

IV Z80 FAMILY DATA SHEETS

V Z80 Development Equipment

V Z80 DEVELOP-MENT EQUIPMENT

VI Z80 Microcomputer Application Notes

VI Z80 MICRO-COMPUTER APPLICATION NOTES

VII Z80 Microcomputer Peripherals

VII Z80 MICRO-COMPUTER PERIPHERALS

VIII Z80 Military/Hi-Rel

VIII Z80 MILITARY/HI-REL

MOSTEK®

Z80 MICROCOMPUTER SYSTEMS

MATRIX™ Microcomputer Development System

INTRODUCTION

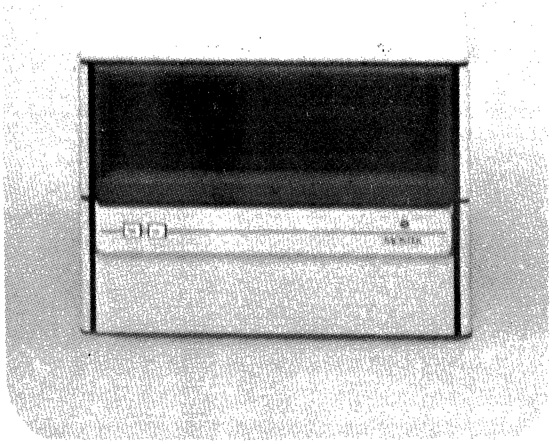
The Mostek MATRIX™ is a complete state-of-the-art, floppy disk-based computer. Not only does it provide all the necessary tools for software development, but it provides complete hardware/software debug through Mostek's AIM™ series of in-circuit emulation cards for the Z80 and the 3870 family of single chip microcomputers. The MATRIX has at its heart the powerful OEM-80E (Single Board Computer), the RAM-80BE (RAM I/O add-on board), and the FLP-80E (floppy disk controller board). Because these boards and software are available separately to OEM users, the MATRIX serves as an excellent test bed for developing systems applications.

The disk-based system eliminates the need for other mass storage media and provides ease of interface to any peripheral normally used with computers. The file-based structure for storage and retrieval consolidates the data base and provides a reliable portable media to speed and facilitate software development.

The FLP-80DOS Disk Operating System is designed for maximum flexibility both in use and expansion to meet a multitude of end user or OEM needs. FLP-80DOS is compatible with Mostek's SD and MD Series of OEM boards, allowing software designed on the MATRIX to be directly used in OEM board applications.

Development System Features

The MATRIX is an excellent integration of both hardware and software development tools for use throughout the complete system design and development phase. The software development is begun by using the combination of Mostek's Text Editor with "roll in-roll out" virtual memory operation and the Mostek relocating assembler. Debug can then proceed inside the MATRIX domain using its resources as if they were in the final system. Using combinations of the Monitor, Designer's Debugging Tool, execution time breakpoints, and single step/ multistep operation along with a formatted memory dump provides control for attacking those tough problems. The use of the Mostek AIM options provides extended debug with versatile hardware breakpoints on memory or port locations, a buffered in-circuit emulation cable for extending the software debug into its own natural hardware environment, and a history memory to capture bus transactions in real time for later examination.

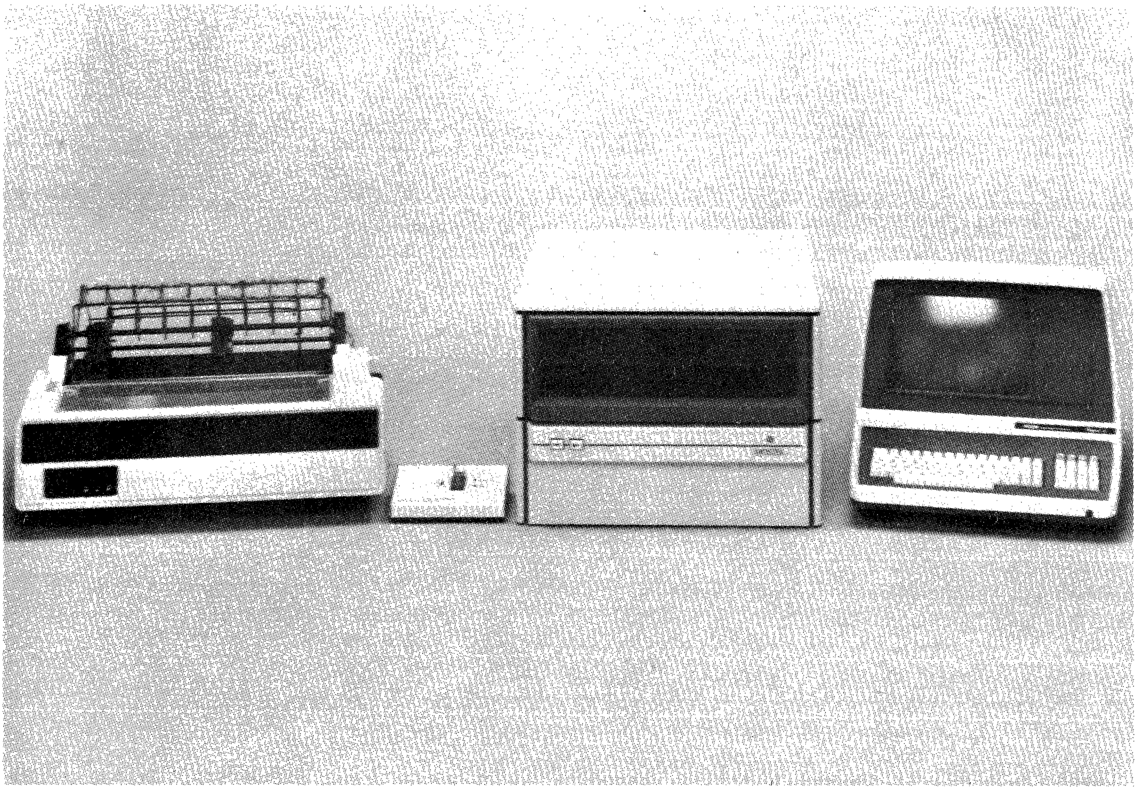


The relocatable and linking feature of the assembler enables the use of contemporary modular design techniques whereby major system alterations can be made in small tractable modules. Using the Linker, the small modules can be combined to form a run-time module without major reassembly of the entire program.

Package System Features

From a system standpoint, the MATRIX has been designed to be the basis of an end product small business/industrial computer. The flexibility provided in the FLP-80DOS operating system permits application programs to be as diverse as a high level language compiler to a supervisory control system in the industrial environment. Other hardware options are available, with even more to be added. Expansion of the disk drive units to a total of four single-sided or double-sided units provides up to two megabytes of storage. This computer uses the third generation Z80 processor supported with the power of a complete family of peripheral chips. Through the use of its 158 instructions, including 16-bit arithmetic, bit manipulation, advanced block moves and interrupt handling, almost any application from communication concentrators to qeneral purpose accounting systems is made easy.

V
Z80
DEVELOP-
MENT
EQUIPMENT



OEM Features

The hardware and software basis for the MATRIX is also available separately to the OEM purchaser. Through a software licensing agreement, all Mostek Software can be utilized on these OEM series of cards.

MATRIX RESIDENT SOFTWARE (FLP-80DOS)

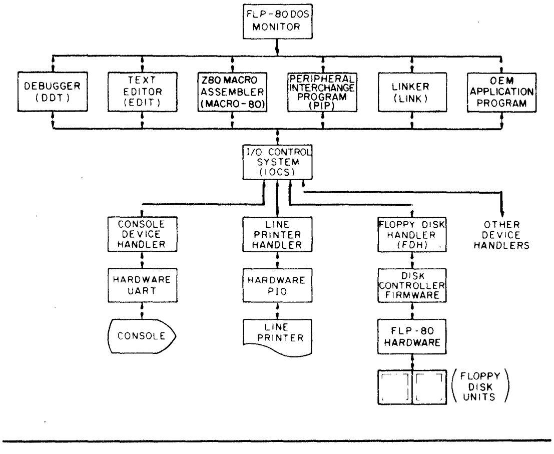
A totally integrated package of resident software is offered in conjunction with the MATRIX consisting of:

Monitor
Text Editor
Z80 Assembler
Linker
DDT-80 with extended debug through AIM modules
Peripheral Interchange Program
Floppy Disk Handler
I/O Control System
Device Driver Library
Batch Mode Operation

Monitor

The FLP-80DOS Monitor is the environment from which all activity in the system initiates. From the Monitor, any system routine such as PIP or a user-generated program is begun by simply entering the program name. FLP-80DOS I/O is done in terms of logical unit numbers, as is commonly done in FORTRAN. A set of logical units is pre-assigned to default I/O drivers upon power up or reset. From the console the user can reassign any logical unit to any new I/O device and can also display logical unit assignments. Executable file creation can be done by the Save command; printable absolute object files can be produced using the Dump command.

MATRIX BLOCK DIAGRAM



Text Editor

The Text Editor permits editing/creating of any source file independent of the language being written. The Editor is both line and string oriented to give maximum utility and user flexibility. The Editor, through its virtual memory "roll in-roll out" technique, can edit a file whose length is limited only by maximum diskette storage. Included in the repertoire of 15 commands are macro commands to save time when encountering a redundant editing task. The Editor is also capable of performing in one operation all the commands which will fit into an 80-column command buffer.

Summary of Editor Commands

Advance N	- Advance line pointer N line
Backup N	- backs up N lines
Change N/S1/S2	- change N occurrences of String 1 to String 2
Delete N	- Delete current line plus next N-1 lines of text
Exchange N	- Exchanges current line plus next N-1 lines with lines to be inserted while in insert mode
Get file	- Reads another file and inserts it into the file being edited after the current line
Insert	- Place Editor in insert mode. Text will be inserted after present line
Line N	- Place line pointer on Line N.
Macro 1 or Macro 2	- Defines Macro 1 or Macro 2 by the following string of Text Editor commands.
Put N file	- Outputs N lines of the file being edited to another disk file.
Quit	- Stores off file under editing process and returns to Monitor environment
Search N/S1	- Searches from existing pointer location until NTH occurrence of string S1 is located and prints it.
Top	- Inerts records at top of file before first line.
Verify N	- Print current record to console plus next N-1 records while advancing pointer N records ahead.
Write N	- Prints current records plus next N-1 records to source output device while advancing pointer N records.
eXecute N	- Executes Macro 1 or Macro 2 as defined by Macro command.
Breakpoint	- sets software trap in user code for interrupting execution in order to examine CPU registers
Register	- displays contents of user's registers
Offset	- enters address adder for debug of relocatable modules
Fill	- fills specified portion of memory with 8 bit byte
Verify	- compares two blocks of memory
Walk	- software single step/multistep
Quit	- returns to Monitor

Debuggers for other processors have similar or enhanced capability and are included with the appropriate AIM.™

Z80 ASSEMBLER

The Z80 Resident Assembler generates relocatable or absolute object code from source files. The assembler recognizes all 158 Z80 instructions as well as 20 powerful pseudo operators. The object code generated is absolute or relocatable format. With the relocating feature, large programs can be easily developed in smaller sections and linked using the Linker. Because the assembler utilizes the I/O Control System, object modules or list modules can be directed to disk files, paper tape, console, or line printer. Portability of output media eliminates the requirement for a complete set of peripherals at every software/hardware development system. The assembler run-time options include sorted symbol table generation, no list, no object, pass 2 only, quit, cross reference table, and reset symbol table. The assembler is capable of handling 14 expression operators including logical, shift, multiplication, division, addition and subtraction operations. These permit complex expressions to be resolved at assembly time by the assembler rather than manually by the programmer. Comments can be placed anywhere but must be integrated with the listing file but can be directed to the console device. In addition, assembler pseudo operators are:

GLOBAL	- for global symbol definition.
PSECT operator	- to generate relocatable or absolute modules
IF expression	- conditional assembly IF expression is true
INCLUDE dataset	- to include other datasets (files) as in-line source code anywhere in source file.

V
Z80
DEVELOP-
MENT
EQUIPMENT

Linker

The Linker program provides the capability of linking assembler-generated, absolute or relocatable object modules together to create a binary or run-time file. This process permits generation of programs which may require the total memory resources of the system. The linking process includes the library search option which, if elected, will link in standard library object files from disk to resolve undefined global symbols. Another option selects a complete global symbol cross-reference listing.

DDT

The Designer's Debugging Tool consists of commands for facilitating an otherwise difficult debugging process. The MATRIX rapid source changes through the editor and re-assemblies, followed by DDT operations close the loop on the debug cycle. The DDT commands include:

Memory - display, update, or tabulate memory
Port - display, update or tabulate I/O ports
Execute - execute user's program
Hexadecimal - performs 16 bit add/sub
Copy - copy one block to another

Peripheral Interchange Program

PIP provides complete file maintenance activity for operations such as copy file from disk to disk, disk to peripheral, or any peripheral to any other peripheral supporting both file-structured and character-oriented devices. Key operations such as renaming, appending, and erasing files also exist along with status commands for diskette ID and vital statistics. PIP can search the diskette directories for any file or a file of a specific name, extension, and user number. The PIP operations are:

Append - appends file 1 to file 2 without changing file 1.

Copy - copies input files or data from an input device to an output file or device. The Copy command can be used for a variety of purposes such as listing files, concatenating individual files, or copying all the files on a single file from one disk unit (e.g. DKO) to a second disk unit (e.g. DK1)

Date - allows the specifying of the date in day, month, and year format. The date specified will be used to date tag any file which is created or edited.

Directory - lists the directory of a specified disk unit (DKO, DKI, etc.). The file name, extension, and user number and creation or edited date are listed for each file in the directory. The user can also request listing-only files of a specified name, only files of a specified extension, or only files of a specified user number. The list device can be any device supported by the system as well as a file.

Erase - erases a single file or files from a diskette in a specified disk unit. The user has the option to erase all files, only files of a specified file name, or only files of a specified user number.

Format - takes completely unformatted soft-sectored diskettes, formats to IBM 3740, and prepares to be a system diskette. Operation is performed on diskette unit 1 and a unique 11-character name is assigned to that diskette.

Init - initializes maps in the disk handler when a new diskette has been changed while in the PIP environment.

Rename - renames a file, its extension, and user number to a file of name X, extension Y, and user Z.

Status - lists all vital statistics of a disk unit to any device. These include the number of allocated records, the number of used records, and the number of bad records

Quit - returns to Monitor Environment.

DOS/Disk Handler

The heart of the FLP-80DOS software package is the Disk Operating System. Capable of supporting up to 4 single-density, single or double-sided units, the system provides a file-structure orientation timed and optimized for rapid storage and retrieval. Program debug is enhanced by complete error reporting supplied with the DOS. Additionally, extensive error recovery and bad sector allocation insure data and file integrity. The DOS not only provides file reading and writing capability, but special pointer manipulation, record deletions, record insertions, skip records both forward and backward as well as directory manipulation such as file creation, renaming, and erasure. The DOS is initiated by a calling vector which is a subset of the I/O control system vector or through the standard IOCS calling sequence to elect buffer allocation, blocking, and deblocking of data to a user-selectable, logical record type.

A unique dynamic allocation algorithm makes optimal use of disk storage space. Run time (Binary) files are given first priority to large blocks of free space to eliminate overhead in operating system and overlay programs. The algorithm marks storage fragments as low priority and uses them only when the diskette is nearing maximum capacity. The DOS permits 7 files to be opened for operations at any one time, thus permitting execution of complex application programs.

I/O Control System

The I/O Control System provides a central facility from which all calls to I/O can be structured. This permits a system applications program to dissolve any device dependence by utilizing the logical unit approach of large, main-frame computers. For example, a programmer may want to structure the utility to use logical unit No. 5 as the list device which normally in the system defaults to the line printer. He may, however, assign at run time a different device for logical unit No. 5. The application program remains unchanged.

Interface by a user to IOCS is done by entering a device mnemonic in a table and observing the calling sequence format. IOCS supplies a physical buffer of desired length, handles buffer allocation, blocking, deblocking, and provides a logical record structure as specified by the user.

Batch - Mode Operation

In Batch-Mode Operation, a command file is built on disk or assigned to a peripheral input device such as a card reader. The console input normally taken from the keyboard is taken from this batch device or batch file. While operating under direction from a batch file, the console output prompts the user as normal or the prompting can be directed to any other output device. The Batch operation is especially useful for the execution of redundant procedures not requiring constant attention of the operator.

MATRIX SYSTEM SPECIFICATIONS

- Z80 CPU.
- 4K byte PROM bootstrap and Z80 debugger
- 60K bytes user RAM. (56K contiguous)
- 8 x 8 bit I/O ports (4 x PIO) with user-definable drivers/receivers
- Serial port, RS 232 and 20 mA current loop.
- 4 channel counter/timer (CTC).
- 2 single-density, single-sided disk drives; 250K bytes per floppy disk.
- 3 positions for AIM modules, A/D cards, Serial Interface, etc.
- Device drivers for paper tape readers, punches, card readers, line printers, Silent 700's, Teletypes and CRT's are included. Others can be added.
- PROM programmer I/O port. Programmer itself is optional.
- Bus compatible with Mostek SD/E series of OEM boards.

HARDWARE DESCRIPTION OEM-80E CPU Module

The OEM-80E provides the essential CPU power of the system. While using the Z80 as the central processing unit, the OEM-80E is provided with other Z80 family peripheral chip support. Two Z80 PIO's give 4 completely programmable 8 bit parallel I/O ports with handshake from which the standard system peripherals are interfaced. Also on the card is the Z80-CTC counter time circuit which has 3 free flexible channels to perform critical counting and timing functions. Along with 16K of RAM, the OEM-80 provides 5 ROM/PROM sockets which can be utilized for 10/20K of ROM or 5/10K PROM. Four sockets contain the firmware portion of FLP-80DOS. The remaining socket can be strapped for other ROM/PROM elements.

RAM-80BE

The RAM-80BE adds additional memory with Mostek's MK4116 16K dynamic memory along with more I/O. These two fully programmable 8-bit I/O ports with handshake provide additional I/O expansion as system RAM memory needs grow. Standard system configuration is 48K bytes for a system total of 60K bytes user RAM (56K contiguous).

FLP-80E

Integral to the MATRIX system is the floppy controller. The FLP-80E is a complete IBM 3740 single-density/double-sided controller for up to 4 drives. The controller has 128 bytes of FIFO buffer resulting in a completely interruptable disk system.

OPTIONAL MODULES COMPATIBLE WITH MATRIX

AIM-Z80BE (6.0MHz max. clock rate)

The AIM-Z80AE is an improved Z80 In-Circuit-Emulation module usable at Z80-CPU clock rates of up to 4MHz. The AIM-Z80AE is a two processor solution to In Circuit Emulation which utilizes a Z80-CPU in the buffer box for accurate emulation at high clock rates with minimum restrictions on the target system. The AIM-Z80AE provides real time emulation (no WAIT states) while providing full access to RESET, NMI and INT control lines. Eight single byte software breakpoints (in RAM) are provided as well as one hardware trap (RAM or ROM). The emulation RAM on the AIM-Z80A is mappable into the target system in 256 byte increments. A 1024 word x 48 bit history memory is triggerable by the hardware intercept and can be read back to the terminal to provide a formated display of the Z80-CPU address, data, and control busses during the execution of the program under test. Several trigger options are available to condition the loading of the history memory.

AIM-7XE

the AIM-7XE module provides debug and in-circuit emulation capabilities for the 3870 series microcomputers on the MATRIX. Multiple breakpoint capability and single-step operation allow the designer complete control over the execution of the 3870 Series microcomputer.

Register, Port display, and modification capability provides information needed to find system "bugs." All I/O is in the user's system connected to AIM-7XE by a 40-pin interface cable.

The debugging operation is controlled by a mnemonic debugger which controls the interaction between the Z80 host computer and the 3870 slave. It includes a history module for the last 1024 CPU cycles and also supports all 3870 family circuits.

Assembly and linking is done using the MACRO-70 Assembler and the standard FLP-80DOS linker.

MECHANICAL SPECIFICATIONS

Overall Dimensions:
CPU subsystem - 8" High x 21" wide x 22" deep
(20.3cm x 53.3 cm x 55.8cm)
Disk subsystem - 8" High x 21" wide x 22" deep
(20.3cm x 53.3 cm x 55.8cm)

Humidity: up to 90% relative, noncondensing.

Material: Structural Foam (Noryl)

Weight: CPU Subsystem 25 lbs (11.3 Kg)
Disk Subsystem 50 lbs (22.7 Kg)

Fan Capacity: 115 CFM

Card Cage: Six slots DIN 41612 type connectors

Operating Temperature: +10°C to +35°C

ELECTRICAL SPECIFICATIONS

INPUT 100/115/230 volts AC ± 10%
50 Hz (MK78189) or 60Hz (MK78188)

OUTPUT

CPU subsystem	+5 VDC at 12A max. +12 VDC at 1.7A max. -12 VDC at 1.7A max.
Disk subsystem	+5VDC at 3.0A max. -5 VDC at 0.5A max. +24 VDC at 3.4A max.

ORDERING INFORMATION

BASIC SYSTEM NO.

NAME	DESCRIPTION	PART NO.
MATRIX™	Z80 floppy disk based microcomputer with 60K bytes of RAM (56K bytes contiguous RAM), 4K bytes PROM bootstrap, two 250K byte single density floppy disk drives with Operations Manual. Includes the software package of FLP-80DOS distributed on diskette. Requires signed license agreement with purchase order.	MK78188 (60Hz) MK78189 (50Hz)
MATRIX™	Operations Manual Only	MK79730
FLP-80DOS	Operations Manual Only	MK78557

IN-CIRCUIT EMULATION MODULES

NAME	DESCRIPTION	PART NO.
AIM-Z80AE	4.0 MHz RAM based Z80 in-circuit emulator with expanded history trace, buffer box, cables and Operations Manual 16K Bytes emulation RAM 32K Bytes emulation RAM	 MK78181-1 MK78181-2
AIM-Z80AE	Operations Manual only	MK79650
AIM-7XE	RAM based in-circuit emulator for the 3870 series of single-chip microcomputers (3870, 3872, 3874 and 3876) with cables and Operations Manual.	MK79077
AIM-7XE	Operations Manual only	MK79579

SOFTWARE-FULLY SUPPORTED

NAME	DESCRIPTION	PART NO.
MACRO-70	Relocatable 3870/F8 MACRO assembler which speeds up development of 3870/F8 programs through use of MACRO to run on MATRIX with Operations Manual. Requires signed license agreement with purchase order.	MK79085
MACRO-70	Operations Manual Only	MK79658
MACRO-80	Operations Manual Only	MK79635

NAME	DESCRIPTION	PART NO.
ANSI BASIC	ANSI BASIC interpreter with random disk access for the MATRIX microcomputer including operations manual. Requires signed license agreement with purchase order.	MK78157
ANSI BASIC	Operations Manual only	MK79623

SOFTWARE-LEVEL 2 UNSUPPORTED

NAME	DESCRIPTION	PART NO.
MOSTEK FORTRAN IV	FORTRAN IV compiler (Z80 object code) for the MATRIX microcomputer with Operations Manual. Requires signed license agreement with purchase order.	MK78158
MOSTEK FORTRAN IV	Operations Manual only MK79644	MK79643
LIBRARY	Vol. 1 of Z80 Software Library including FLP-80DOS utilities, sort, 8080 to Z80 source translator, word processor program, LLL BASIC (6K). 23 Programs total including source, object, and binary.	MK78164

PERIPHERALS AND CABLES

NAME	DESCRIPTION	PART NO.

MOSTEK VT	110-9600 Baud CRT with upper and lowercase character set. Includes cable (78152) to MATRIX. 110/115 volt 50/60 Hz 230 volt 50/60 Hz	MK78190-1(60Hz)
		MK78190-2(50Hz)

MOSTEK LP	7 x 7 dot MATRIX printer with 120 character LP per second operation. Includes interface cable to MATRIX.
	100/115 volt model 50/60 Hz 230 volt model 50/60Hz
		MK78191-1(60Hz)
		MK78191-2(50Hz)

PPG-8/16	Programmer for 2708, 2758 and 2716 PROM Includes interfacing cables to MATRIX.	MK79081-1
SD-WW	Wire wrap card compatible with MATRIX.	MK79063
SD-EXT	Extender card compatible with MATRIX.	MK79062
LP-CABLE	Interface cable from MATRIX Microcomputer to Centronics 306 or 702 printer	MK79089
PPG-CABLE	Interface cables from MATRIX to PPG-8/16 PROM programmer (MK79081).	MK79090

Standard License Agreement and Registration Form

All Mostek Corporation software products are sold on condition that the purchaser agrees to the following terms:

1. The Purchaser agrees not to sell, provide, give away, or otherwise make available to any unauthorized persons, all or any part of, the Mostek software products listed below, including, but not restricted to: object code, source code, and program listings. This license allows the purchaser to operate the software product only on the system referenced by serial number below. Mostek retains title to all Mostek software products including diskettes and tapes.

2. The Purchaser may at any time demonstrate the normal operation of the Mostek software product to any person.

3. Purchaser may not copy materials furnished with Mostek software products but copies may be obtained from Mostek. No part of the Mostek software products may be copied by Purchaser in printed or machine-readable form unless for the purpose of study, modification or back-up. Purchaser will place Mostek's copyright notice on all copies and maintain records, available at Mostek's request, of the location of the copies.

4. Mostek's sole obligation shall be to make available to Purchaser all published modifications or updates made by Mostek to licensed software products which are published and made generally available within one (1) year from date of purchase, provided Purchaser has complied with the Software License Agreement and Registration Form.

5. In no event will Mostek be held liable for any loss, expense or damage, of any kind whatsoever, direct or indirect, including as a result of Mostek's negligence. Mostek shall not be liable for any incidental damages, consequential damages and lost profits, arising out of or connected in any manner with any of Mostek's software products described below.

6. MOSTEK MAKES NO WARRANTIES OF ANY KIND, WHETHER STATUTORY, WRITTEN, ORAL, EXPRESSED OR IMPLIED (INCLUDING WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE AND MERCHANTABILITY AND WARRANTIES ARISING FROM COURSE OR DEALING OR USAGE OF TRADE) WITH RESPECT TO THE SOFTWARE DESCRIBED BELOW.

7. Any license under this agreement may be terminated for breach by one month's prior written notice and Purchaser will promptly return all copies of any parts of Mostek Software products.

List the following Software Products subject to this agreement

Mostek Product Number (MK#)	Product Name	Mostek Product Number (MK#)	Product Name

AGREED TO:

PURCHASER

Name (PRINT)

Signature (PARTY)

Title Date

MOSTEK CORPORATION

Name

Title Date

PLEASE PRINT FOLLOWING INFORMATION:

Company Name Date

Address

City State Zip Code

Country Telephone ( )

Mostek Disk System Serial # or Mostek Disk Controller Serial #

or □ Non Mostek Hardware

Date of Purchase

Place of Purchase

If you are purchasing this product from a Distributor, print the Distributor's name below and return this form to the Distributor; The distributor will provide a purchase order # and will then forward this form to the address below.

Distributor Name ____

If you are purchasing this product direct from Mostek, check the Customer PO # box, provide your purchase order #, and return this form to the address below.

Purchase Order # to Mostek □ Customer PO #
or
□ Distributor PO #
PO # ____

RETURN THIS FORM TO:

Software Librarian, MS #510
Micro System Department
Mostek Corporation
1215 W. Crosby Road
P.O. Box 169
Carrollton, Texas 75006

*NOTE: Mostek will not ship this software product to customer until this signed form is received by the Mostek software librarian.

MOSTEK®

PERIPHERAL

CRT

MK78190-1, MK78190-2

FEATURES

□ Interfaces directly to MATRIX™

□ All 128 ASCII codes

□ 32 displayable control codes (in monitor mode)

□ Displays up to 96 characters, including lower case

□ Keyboard layout similar to that of typewriter

□ Separate 18 key numeric pad

□ Switch-selectable inverse video

□ Cursor addressing

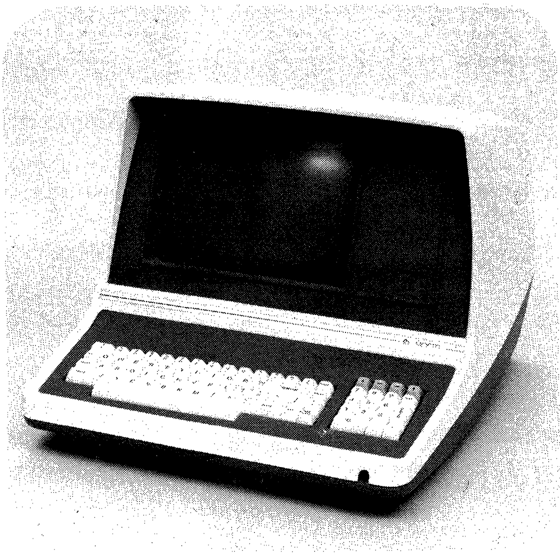
□ EIA interface

□ Baud rates up to 9.6 KB

□ Auxiliary unidirectional EIA output controlled by DC2 (on) and DC4 (off)

□ 5 x 8 Dot Matrix

CRT PHOTO



DESCRIPTION

The Mostek CRT is a high-performance, keyboard display unit that is fully compatible with the MATRIX™ microcomputer system.

The character set consists of 96 displayable upper and lower-case characters with lower-case descenders. The display may be switch-selected to be standard video (white on black) or reverse video (black on white).

The Mostek CRT can be interfaced to any computer system that provides a RS-232 serial asynchronous interface.

OPERATING CHARACTERISTICS

TERMINAL CONTROL

	Keyboard	Remote Command
CLEAR SCREEN	•	•
CLEAR TO END OF LINE	•	•
CLEAR TO END OF SCREEN	•	•
AUDIBLE ALARM	•	•
BACKSPACE	•	•
KEYBOARD LOCK		•
KEYBOARD UNLOCK		•
TAB		•
MONITOR MODE	•	•

CURSOR CONTROL

	Keyboard	Remote Command
CURSOR ADDRESS (XY)	•	•
INCREMENTAL CURSOR CONTROL	•	•
HOME CURSOR	•	•

SPECIFICATION

DISPLAY CHARACTERISTICS

Characters per line: 80
Lines per display: 24
Screen capacity: 1920 characters
All 128 ASCII codes
96 displayable characters including lower case
32 displayable control codes
Character size: 5 x 8 dot matrix
Refresh rate: 50/60 frames/sec
Cursor: Block, Flashing Block, Underline, or Flashing Underline

INTERFACE

Full or Half Duplex (W.E. modem 103A compatible or W.E. Modem 202C/D using character turnaround).

EIA RS-232-C connector.

Eight Baud Rates: 110, 150, 300, 1200, 1800, 2400, 4800, 9600.

Parity: Odd, Even, 1, 0 or off

No. of Stop Bits: one (two at 110 Baud)

EXTERNAL CONTROLS

Auto Scroll
Contrast
Power On/Off
Half Duplex/Full Duplex
Auto LF/CR Control
Reverse Video or Standard Video
Upper/Lower Case
Parity
Baud rate
EIA or Current Loop

ELECTRICAL

Power consumption: 60 watts, nominal
Power input: 115 V, 60 Hz; 115 V, 50 Hz

MECHANICAL

Size (nominal): 15 in. (38 cm) high, 18.5 in. (47 cm) wide, 23.25 in. (59 cm) deep

Weight: 38 lbs. (17 kg)

ENVIRONMENTAL

Temperature: 10°C to 40°C
Storage Temperature: 0°C to 85°C
Humidity: 10 to 90% relative, non-condensing

ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NO.
CRT	Mostek CRT terminal featuring all 128 ASCII codes, 96 displayable characters including lower case, 80 characters by 24 lines, typewriter-like keyboard layout, cursor addressing, EIA interface and Baud rates to 9.6 K Baud. Includes RS-232 interface cable (MK78152).	MK78190-1
CRT-50	Same as above but for 50 Hz operation.	MK78190-2
SDE-RMC6 to CRT	CRT interface cable only.	MK78152
MD-232 DCE-C	CRT to MDX-SIO, MDX-DEBUG or MDX-EPROM/UART	MK77955

MOSTEK®

PERIPHERAL

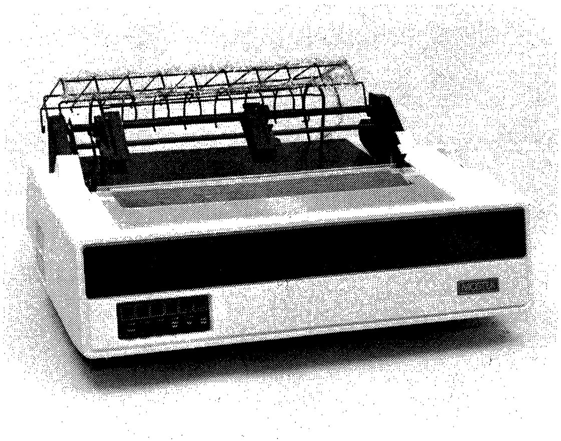
Line Printer

MK78191-1, MK78191-2

FEATURES

- □ Interfaces directly to MATRIX™
- □ Prints 120 characters per second
- □ Up to 132 characters per line
- □ Prints original plus five copies
- □ Character elongation
- □ Eight inches per second paper slew rate
- □ Ribbon cartridge
- □ 7x7 dot matrix, 64-character ASCII
- □ Tractor feed/Pin feed platen
- □ Parallel interfce

LINE PRINTER PHOTO



DESCRIPTION

The Mostek line printer is a state-of-the-art microprocessor-controlled, dot matrix line printer that prints at the rate of 120 characters per second. The printer has a maximum print width of 132 characters with a horizontal format of ten characters per inch and six lines per inch vertical. Elongated (double-width) characters are software-selectable.

The Mostek line printer interfaces directly to the MATRIX™ Microcomputer System and can be interfaced easily to other computer systems supporting parallel I/O.

SPECIFICATIONS

Print performance - Minimum throughout

Printer	Print Speed (cps)	Max Print	10Char/ Line (lpm)	80Char/ Line (lpm)	132Char/ Line (lpm)
702	120	132...	200	74	47

Character
7x7 dot matrix ....

Format
Ten Characters per inch horizontal
Six Lines per inch vertical
Elongated (double-width) characters software-selectable

Forms Handling
Tractor feed, for rear or bottom feed forms
8 ips slew rate
Usable paper 4 in. (102 mm) to 17.3 in. (439 mm) width
Paper tension adjustment

Ribbon System
Ribbon cartridge
Continuous ribbon 9/16 in. (14 mm) wide, 20 yards (18.3 meters) long.
Mobius loop allows printing on upper and lower portion on alternate passes.

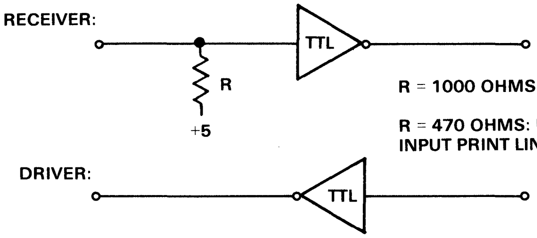
Panel Indicators
Power On: Indicates AC power is applied to printer.
Select: Indicates printer can receive date.
Alert: Indicates operator-correctable error condition.

Operator Controls
Select/deselect
Forms thickness
Top of form
Horizontal forms positioning
Vertical forms positioning
Power ON/OFF
Single line feed
Paper empty override
Self-test

INTERFACE DRIVERS AND RECEIVERS

ALL INPUT/OUTPUT SIGNALS ARE TTL COMPATIBLE

LO: 0.4 VOLTS HI: 2.4 VOLTS

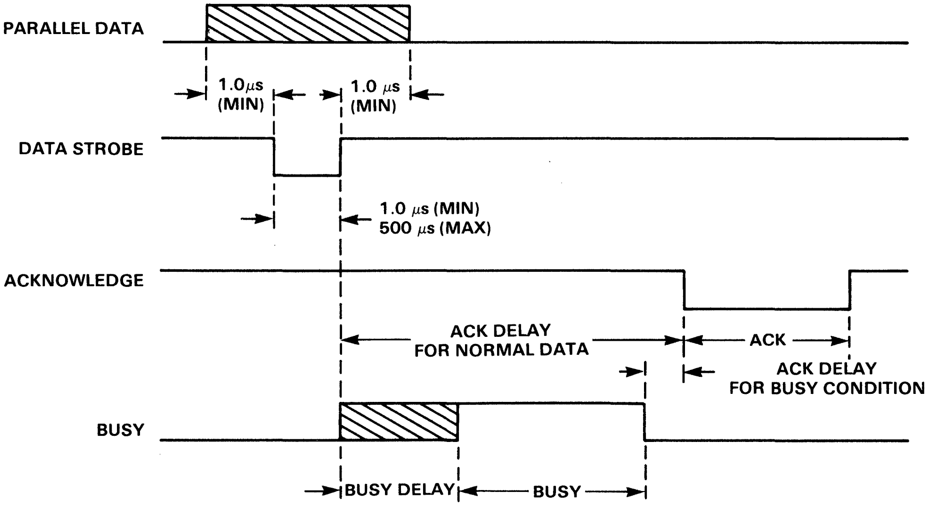


R = 1000 OHMS: DATA LINES

R = 470 OHMS: DATA STROBE AND INPUT PRINT LINES

CONNECTOR: AMPHENOL 57 40360 SERIES, 36-PIN
(CENTRONICS 31310019)

INTERFACE TIMING



Internal Controls

Auto motor control:	Turns stepping motors off when no data is received.
Electronic top of form:	Allows paper to space to top of form when command is received.

Preset for 11 in. (279 mm) or 12 in. (305 mm) forms Opt. VFU must be used for other form lengths.

Data Input

7- or 8-bit ASCII parallel; microprocessor electronics; TTL levels with strobe.
Acknowledge pulse indicates that data was received.

INTERFACING

Electrical Requirements

50/60 Hz, 115/230 VAC; -10%/-15% of Nominal Tappable Transformer (100, 110, 115, 120, 200, 220, 230, 240 VAC).

Physical Dimensions

	Model 702
Weight:	60 lbs. (27 Kg)
Width:	24.5 in. (622 mm)
Height:	8 in. (203 mm)
Depth:	18 in. (457 mm)

Temperature

Operating: 40° to 100°F (4.4° to 37.7°C)
Storage: -10° to 160°F (-40° to 71.1°C)

Humidity

Operating: 20% to 90% (No condensation)
Storage: 5% to 95% (No condensation)

Normal Data Input Timing	**ACK Delay** **ACK**	**2 - 6 µsec** **4 µsec**
BUSY CONDITION TIMING	BUSY DELAY	0 - 1.5 µsec
	ACK DELAY	1 - 6 µsec
	ACK	4 µsec
	BUSY DURATION:	
	Line Feed	350 - 500 µsec
	Verfical Tab (1-in.)	135 - 145 msec
	Form Feed (11-in.)	1.48 - 1.50 sec
	Delete	160 - 400 µsec
	Bell	0
	Select*	0 - 1.5 µsec
	Deselect	Unit Printer is selected
	Printer	8.33 msec/char; plus 148 msec non-printing time/line

*No busy if inhibit prime on select option is used.

ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NO.
LP	Mostek line printer featuring 120 cps operation, 7x7 dot matrix, 10 cpi, and paper slew rate of 8 ips. Includes MATRIX™ cable, 60 Hz operation.	MK78191-1
LP-50	Same as above but for 50 Hz operation.	MK78191-2
MD-CPRT-C	MATRIX System or MDX-PIO to Centronics Line Printer Interface cable.	MK79089

MOSTEK®

PERIPHERAL

PPG 8/16-PROM Programmer

MK79181-1

FEATURES

□ Programs, reads, and verifies 2708-, 2758-, and 2716-type PROMs (2758 and 2716 PROMS must be 5-Volt only type)

□ Interfaces to MATRIX and MDX-PIO

□ Driver software included on system diskette for FLP-80DOS

□ Zero-insertion-force socket

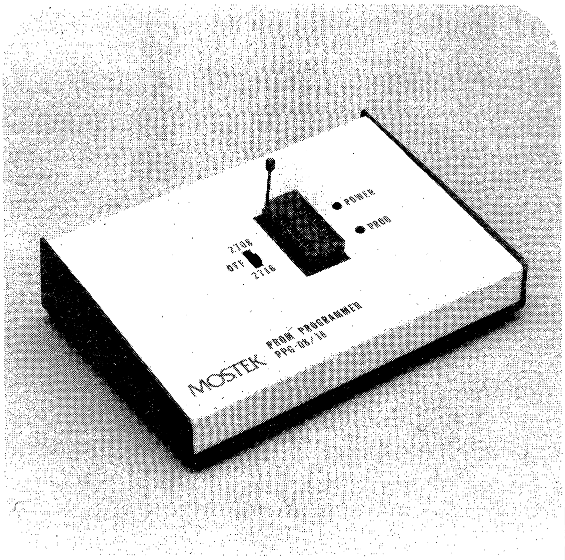
□ Power and programming indicators

DESCRIPTION

The PPG-8/16 PROM Programmer is a peripheral which provides a low-cost means of programming 2708, 2758, or 2716 PROMs. It is compatible with Mostek's MATRIX Microcomputer Development System and the MDX-PIO. The PPG-8/16 has a generalized computer interface (two 8-bit I/O ports) allowing it to be controlled by other types of host computers with user-generated driver software. A complete set of documentation is provided with the PPG-8/16 which describes the internal operation and details user's operating procedures

The PPG-8/16 is available in a metal enclosure for use with the MATRIX™ and the MDX-PIO. Interface cables for either the MATRIX or MDX-PIO must be purchased separately.

PPG 8/16 PHOTO

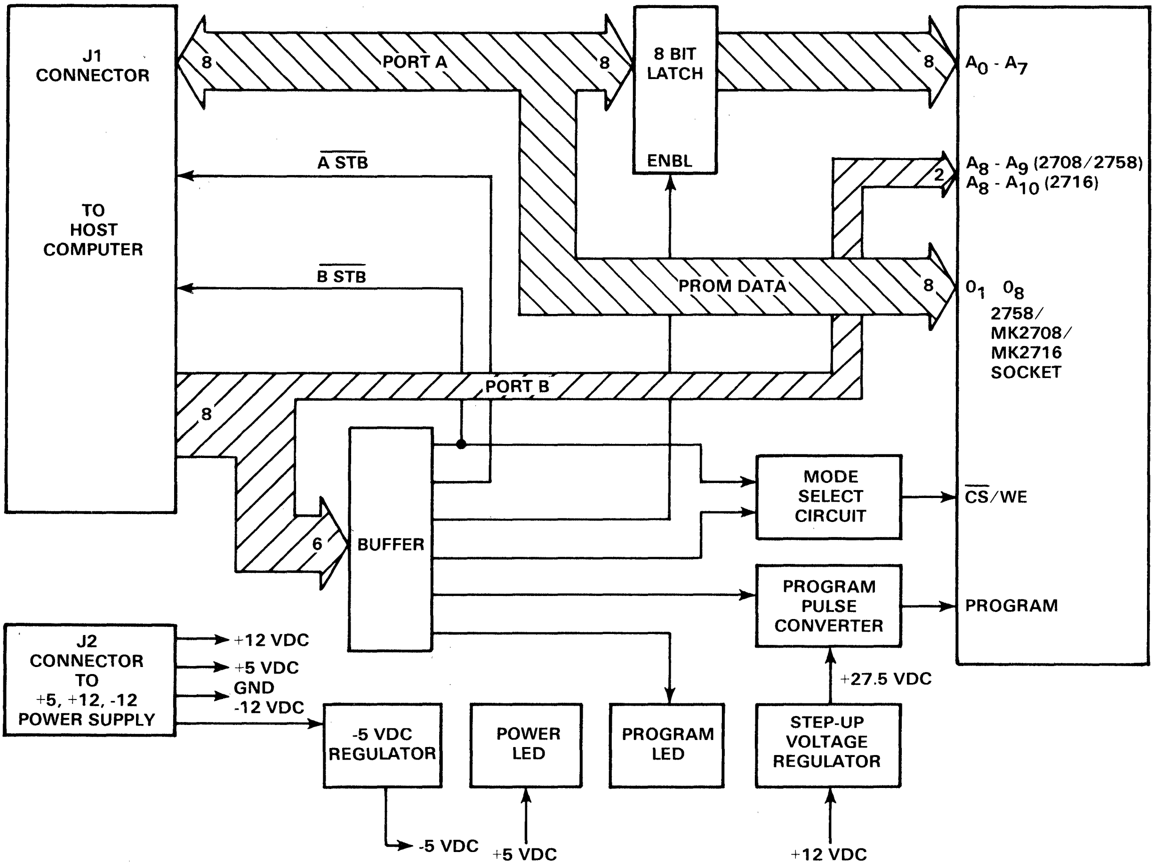


SOFTWARE DESCRIPTION

The driver software accomplishes four basic operations. These are (1) loading data into host computer memory, (2) reading the contents of a PROM into host computer memory, (3) programming a PROM from the contents of the host computer memory, and (4) verifying the contents of a PROM with the contents of the host computer memory.

The driver software is provided on the FLP-80DOS system diskette. The user documentation provided with the PPG-8/16 fully explains programming procedures to enable a user to develop a software driver on a different host computer.

PPG8/16 BLOCK DIAGRAM



INTERFACE

25-pin control connector (D type)
40-pin control connector (0.1-in. centers card edge) for AID-80F, SDB-80, SDB-50/70, or MATRIX™
12-pin power connector (0.156-in. centers card edge)
All control signals are TTL-compatible

POWER REQUIREMENTS

+12 VDC at 250mA typical
+5 VDC at 100mA typical
-12 VDC at 50mA typical

OPERATING TEMPERATURE

0°C -60°C

PROGRAMMING TIME

2708 - 2.5 minutes
2758 - 0.9 minutes
2716 - 1.8 minutes

ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NO.
PPG-8/16	PROM Programmer for 2708/2758/2716 PROMs with Operations Manual for interface with MATRIX.	MK79081-1
MATRIX to PPG-8/16	PPG-8/16 Interface Cable for MATRIX	MK79090
MD-PPG-C	PPG-8/16 Interface Cable for MDX-PIO	MK77957
	PPG-8/16 Operations Manual	MK79603

*NOTE: The PPG-8/16 will only program the 2708, 2758, and 2716 PROMs. The 2758 and 2716 are 5 Volt only type PROMs. THE PPG-8/16 WILL NOT PROGRAM THE TI2716 MULTIPLE-VOLTAGE 2K x 8 PROM.

MOSTEK®

SOFTWARE DISK BASED

ANSI BASIC Software Interpreter

MK78157

FEATURES

□ Meets ANSI standard on BASIC (X3.60 - 1978)

□ Direct access to CPU I/O Ports

□ Ability to read or write any memory location (PEEK, POKE)

□ Arrays with up to 255 dimensions

□ Dynamic allocation and deallocation of arrays

□ IF . . . THEN . . . ELSE and IF . . . GO TO (both if's may be nested)

□ Direct (immediate) execution of statements

□ Error trapping, with error messages in English

□ Four variable types: Integer, string, real and double-precision real

□ Long variable names significant up to 40 characters

□ Full PRINT USING capabilities for formatted output

□ Extensive program editing facilities

□ Trace facililties

□ Can call any number of assembly-language subroutines

□ Boolean (logical) operations

□ Supports up to six sequential and random access files on floppy disk

□ Variable record length in random access files from one to 128 bytes/record

□ Complete set of file manipulation statements

□ Occupies only 23K bytes, not including operating system

□ Supports console and line printer I/O

□ Allows console output to be redirected to the line printer

□ WHILE . . . WEND structured construct

□ Programs can be saved on disk in a protected format that cannot be listed on console

DESCRIPTION

Mostek ANSI BASIC is an extensive implementation of Microsoft BASIC for the Z80 microprocessor. Its features are comparable to the BASICs found on minicomputers and large mainframes. Mostek ANSI BASIC is among the fastest microprocessor BASICs available. Designed to operate on Mostek Systems with FLP-80DOS V2.1 and with 48K bytes or more memory, Mostek BASIC provides a sophisticated software development tool.

Mostek ANSI BASIC is implemented as an interpreter and is highly suitable for user-interactive processing. Programs and data are stored in a compressed internal format to maximize memory utilization. In a 64K system, 28K of user's program and data storage area are available.

Unique features include long variable names, substring assignments and hexadecimal and octal constants. Many other features ease the task of programming complex functions. The Programmer is seldom limited by array size (up to 255 dimensions, with run-time allocation and deallocation) or I/O restrictions. Full PRINT USING capabilities allow formatted output, while both input and output may be performed with multiple sequential and random files on floppy disk as well as with the CPU I/O ports. Editing, error trapping, and trace facilities greatly simplify program debugging.

Commands:

AUTO	CLEAR	CONT	DELETE	EDIT
FILES	LIST	LLIST	LOAD	MERGE
NEW	NULL	RENUM	RESET	RUN
SAVE	SYSTEM	TRON	TROFF	WIDTH

Program Statements:

CALL	CHAIN	COMMON	DEF DBL	DEF FN
DEFINT	DEFSNG	DEFSTR	DEFUSR	DIM
END	ERASE	ERROR	FOR . . NEXT	GOSUB . . . RETURN
GOTO	IF . . . THEN(ELSE)	IF . . . GOTO	LET	ON ERROR GOTO
ON . . . GOSUB	ON . . . GOTO	OPTION BASE	RANDOMIZE	
REM	RESUME	STOP	SWAP	
WHILE . . . WEND				

Input/Output Statements:

CLOSE	DATA	FIELD	GET	INPUT
INPUT#	KILL	LINE INPUT	LINE INPUT#	LPRINT
LPRINT USING	LSET	NAME	OPEN	OUT
PRINT	PRINT USING	PRINT#	PRINT# USING WRITE	PUT
READ	RESTORE	RESET	RSET	WRITE#

Operators:

=	-	+	*	/
^	\	>	<	<=
>=	<>	MOD	NOT	AND
OR	XOR	IMP	EQU	

Arithmetic Functions:

ABS	ATN	CDBL	CINT	COS	CSNG
EXP	ERR	ERL	FIX	FRE	INT
LOG	RND	SGN	SIN	SQR	TAN
USR	VARPTR				

String Functions:

ASC	CHR$	HEX$	INSTR	LEFT	LEN
MID$	OCT	RIGHT$	SPACE$	SPC$	STR$
STRINGS	VAL				

Input/Output Functions:

CVI	CVS	CVD	DSKF	EOF	INP
INPUT$	LOC	LOF	LOG	LPOS	MKD$
MKI$	MKS$	PEEK	POKE	POS	TAB
WAIT					

ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NO.
Mostek ANSI BASIC	BASIC INTERPRETER high-level language to run on FLP-80DOS. Requires 48K or more bytes of memory.	MK78157
	BASIC Operation Manual Only	MK79708

In order to receive Mostek ANSI BASIC, the Mostek BASIC non-disclosure agreement must be signed and returned with each purchase order.

MOSTEK®

SOFTWARE DISK BASED

FORTRAN IV Compiler

MK78158

FEATURES

□ All of ANSI standard FORTRAN IV (X3.9-1966) except complex data type

□ Generates relocatable linkable object code

□ Subroutines may be compiled separately and stored in a system library

□ Compiles several hundred statements per minute in a single pass

□ Enhancements include

1. LOGICAL variables which can be used as integer quantities
2. LOGICAL DO loops for tighter, faster execution of small-valued integer loops
3. Mixed-mode arithmetic
4. Hexadecimal constants
5. Literals and Holleriths allowed in expressions
6. Logical operations on integer data. .AND., .OR., .NOT. and .XOR. can be used for 16-bit or 8-bit Boolean operations
7. READ/WRITE End-of-File or Error Condition transfer. END=n and ERR=n (where n is the statement number) can be included in READ or WRITE statements to transfer control to the specified statement on detection of an error or end-of-file condition
8. ENCODE/DECODE for FORMAT operations to memory

□ Long descriptive error messages

□ Extended optimizations

□ Z80-assembly-language subprograms may be called from FORTRAN programs

DESCRIPTION

Mostek's FORTRAN IV Compiler package provides new capabilities for users of Z80-based microcomputer systems. Mostek FORTRAN is comparable to FORTRAN compilers on large mainframes and minicomputers. All of ANSI Standard FORTRAN X3.9-1966 is included except the COMPLEX data type. Therefore, users may take advantage of the many applications programs already written in FORTRAN.

FORTRAN IV COMPILER PHOTO

Mostek FORTRAN IV is unique in that it provides a microprocessor FORTRAN development package that generates relocatable object modules. This means that only the subroutines and system routines required to run FORTRAN programs are loaded before execution. Subroutines can be placed in a system library so that users can develop a common set of subroutines that are used in their programs. Also, if only one module of a program is changed, it is necessary to re-compile only that module.

The standard library of subroutines supplied with FORTRAN includes:

ABS	IABS	DABS	AINT
INT	IDINT	AMOD	MOD
AMAXO	AMAX1	MAXO	MAX1
DMAX1	AMINO	AMIN1	MINO
MIN1	DMIN1	FLOAT	IFIX
SIGN	ISIGN	DSIGN	DIM
IDIM	SNGL	DBLE	EXP
DEXP	ALOG	DLOG	ALOG10
DLOG10	SIN	DSIN	COS
DCOS	TANH	SQRT	DSQRT
ATAN	DATAN	ATAN2	DATAN2
DMOD	PEEK	POKE	INP
OUT			

The library also contains routines for 32-bit and 64-bit floating point addition, subtraction, multiplication, division, etc. These routines are among the fastest available for performing these functions on the Z80.

A minimum system size of 48K bytes (including FLP-80DOS) is required to provide efficient optimization. The Mostek FORTRAN compiler optimizes the generated object code in several ways:

1. Common subexpression elimination. Common subexpressions are evaluated once, and the value is substituted in later occurrences of the subexpression.

2. Peephole Optimization. Small sections of code are replaced by more-compact, faster code in special cases.

3. Constant folding. Integer constant expressions are evaluated at compile time.

4. Branch Optimizations. The number of conditional jumps in arithmetic and logical IFs is minimized.

Long descriptive error messages are another feature of the compiler. For instance:

?Statement unrecognizable

is printed if the compiler scans a statement that is not an assignment or other FORTRAN statement. The last twenty characters scanned before the detected error are also printed.

As an option, the compiler generates a fully symbolic listing of the machine language to be generated. At the end of the listing, the compiler produces an error summary and tables showing the addresses assigned to labels, variables and constants.

LINKER

A relocating linking loader (LINK-80) and a library manager (LIB-80) are included in the Mostek FORTRAN package.

LINK-80 resolves internal and external references between the object modules loaded and also performs library searches for system subroutines and generates a load map of memory showing the locations of the main program, subroutines and common areas.

LIBRARY MANAGER

LIB-80 allows users to customize libraries of object modules. LIB-80 can be used to insert, replace or delete object modules within a library, or create a new library from scratch. Library modules and the symbol definitions they contain may also be listed.

XCPM UTILITY

A utility program (XCPM) is included which allows the user to copy FORTRAN source programs from CP/M diskettes to FLP-80DOS diskettes. At this point the programs can be compiled using the Mostek FORTRAN compiler.

FTRANS UTILITY

FTRANS allows the user to convert object programs produced by the Mostek Z80 assembler to a form that is linkable to FORTRAN programs.

DESIGNATOR	DESCRIPTION	PART NO.
Mostek FORTRAN IV	FORTRAN IV high-level compiler to run on FLP-80DOS. Requires 48K bytes of RAM. Includes Operations Manual.	MK78158
	Mostek FORTRAN IV Operations Manual only	MK79643

MOSTEK®

SOFTWARE DISK BASED

FLP-80DOS

MK78142, MK77962

INTRODUCTION

The Mostek FLP-80DOS software package is designed for the Mostek dual floppy disk Z80 Development System or an MD board system. Further information on this system can be found in the MATRIX™ Data Sheet. FLP-80DOS includes:

□ Monitor
□ Debugger
□ Text Editor
□ Z80 Assembler
□ Relocating Linking Loader
□ Peripheral Interchange Program
□ Linker
□ A Generalized I/O System For Peripherals

These programs provide state-of-the-art software for developing Z80 programs as well as establishing a firm basis for OEM products.

MONITOR

The Monitor provides user interface from the console to the rest of the software. The user can load and run system programs, such as the Assembler, using one simple command. Programs in object and binary format can be loaded into and dumped from RAM. All I/O is done via channels which are identified by Logical Unit Numbers. The Monitor allows any software device handler to be assigned to any Logical Unit Number. Thus, the software provides complete flexibility in configuring the system with different peripherals. The Monitor also allows two-character mnemonics to represent 16-bit address values. Using mnemonics simplifies the command language. Certain mnemonics are reserved for I/O device handlers such as 'DK' for the flexible disk handler. The user can create and assign his own mnemonics at any time from the console, thus simplifying the command language for his own use. The Monitor also allows "batch mode operation" from any input device or file.

The Monitor commands are:

$ASSIGN -	assign a Logic Unit Number to a device.
$CLEAR -	remove the assignment of a Logical Unit Number to a device.
$RTABLE -	print a list of current Logic Unit Number-to-Device assignments.
$DTABLE -	print default Logical Unit Number-to-Device assignments.
$LOAD -	load object modules into RAM.
$GTABLE -	print a listing of global symbol table.
$GINIT -	initialize global symbol table.
$DUMP -	dump RAM to a device in object format.
$GET -	load a binary file into RAM from disk.
$SAVE -	save a binary file on disk.
$BEGIN -	start execution of a loaded program.
$INIT -	initialize disk handler.
$DDT -	enter DDT debug environment.
IMPLIED RUN COMMAND -	get and start execution of a binary file.

FLP-80DOS BOARD PHOTO

DESIGNER'S DEVELOPMENT TOOL - DDT

The DDT debugger program is supplied in a combination of ... on the FLP-80DOS diskette.... and absolute Z80 programs. Standard commands allow displaying and modifying memory and CPU registers, setting breakpoints, and executing programs. Mnemonics are used to represent Z80 registers, thus simplifying the command language.

The allowed commands are:

B - Insert a breakpoint in user's program.
C - Copy contents of a block of memory to another location in memory.
E - Execute a program.
F - Fill an area of RAM with a constant.
H - 16-bit hexadecimal arithmetic.
L - Locate and print every occurrence of an 8-bit pattern.
M - Display, update, or tabulate the contents of memory.
P - Display or update the contents of a port.
R - Display the contents of the user's register.
S - Hardware single step - requires Mostek's AIM-80 board or AIM-Z80A board.
W - Software single step.
V - Verify memory (compare two blocks and print differences).

TEXT EDITOR -EDIT

The FLP-80DOS Editor permits random-access editing of ASCII character strings. The Editor works on blocks of characters which are rolled in from disk. It can be used as a line-or character-oriented editor. Individual characters may be located by position or context. Each edited block is automatically rolled out to disk after editing. Although the Editor is used primarily for creating and modifying Z80 assembly language source statements, it may be applied to any ASCII text delimited by "carriage returns".

The Editor has a pseudo-macro command processing option. Up to two sets of commands may be stored and processed at any time during the editing process. The Editor allows the following commands:

An - Advance record pointer n records.
Bn - Backup record pointer n records.
Cn dS1dS2d - Change string S1 to string S2 for n occurrences.
Dn - Delete the next n records.
En - Exchange current records with records to be inserted.
Fn - If n = 0, reduce printout to console device (for TTY and slow consoles).
I - Insert records.
Ln - Go to line number n.
Mn - Enter commands into one of two alternate command buffers (pseudo-macro).
Q - Quit - Return to Monitor.
Sn dS1d -Search for nth occurrence of string S1.
T - Insert records at top of file before first record.
Vn - Output n records to console device.
Wn - Output n records to Logical Unit Number five (LUN 5) with line numbers.
Xn - Execute alternate command buffer n.

Z80 ASSEMBLER - ASM

The FLP-80DOS Assembler reads standard Z80 source mnemonics and pseudo-ops and outputs an assembly listing and object code. The assembly listing shows address, machine code, statement number, and source statement. The code is in industry-standard hexadecimal format modified for relocatable, linkable assemblies.

The Assembler supports conditional assemblies, global symbols, relocatable programs, and a printed symbol table. It can assemble any length program, limited only by a symbol table size of over 400 symbols. Expressions involving arithmetic and logical operations are allowed. Although normally used as a two-pass assembler, the Assembler can also be run as a single-pass assembler or as a learning tool. The following pseudo-ops are supported:

COND - same as IF.
DEFB - define byte.
DEFL - define label.
DEFM - define message (ASCII).
DEFS - define storage.
DEFW - define word.
END - end statement.
ENDC - same as ENDIF.
ENDIF - end of conditional assembly.
EQU - equate label.
GLOBAL - global symbol definition.
IF - conditional assembly.
INCLUDE - include another file within an assembly.
NAME - program name definition.
ORG - program origin.
PSECT - program section definition.
EJECT - eject a page of listing.
TITLE - place heading at top of each page of listing.
LIST - turn listing on.
NLIST - turn listing off.

RELOCATING LINKING LOADER - RLL

The Mostek FLP-80DOS Relocating Linking Loader provides state-of-the-art capability for loading programs into memory. Loading and linking of any number of relocatable or nonrelocatable object modules is done in one pass. A non-relocatable module is always loaded at its starting address as defined by the ORG pseudo-op during assembly. A relocatable object module can be positioned anywhere in memory at an offset address.

The Loader automatically links and relocates global symbols which are used to provide communication or linkage between program modules. As object modules are loaded, a table containing global symbol references and definitions is built up. The symbol table can be printed to list all global symbols and their load address. The number of object modules which can be loaded by the Loader is limited only by the amount of RAM available for the modules and the symbol table.

The Loader also loads industry-standard non-relocatable, non-linkable object modules.

LINKER - LINK

The Linker provides capability for linking object modules together and creating a binary (RAM image) file on disk. A binary file can be loaded using the Monitor GET or IMPLIED RUN command. Modules are linked together using global symbols for communication between modules. The linker produces a global symbol table and a global cross reference table which may be listed on any output device.

The Linker also provides a library search option for all global symbols undefined after the specified object modules are processed. If a symbol is undefined, the Linker searches the disk for an object file having the file-name of the symbol. If the file is found, it is linked with the main module in an attempt to resolve the undefined symbol.

PERIPHERAL INTERCHANGE PROGRAM - PIP

The Peripheral Interchange Program provides complete file maintenance facilities for the system. In addition, it can be used to copy information from any device or file to any other device or file. The command language is easy to use and resembles that used on DEC minicomputers. The following commands are supported:

COMMAND	FUNCITON
APPEND	Append files.
COPY	Copy files from any device to another device or file.
DIRECT	List Directory of specified Disk Unit.
ERASE	Delete a file.
FORMAT	Format a disk.
INIT	Initialize the disk handler.
RENAME	Rename a file.
STATUS	List number of used and available sectors on specified disk unit.
QUIT	Return to Monitor.

The first letter only of each command may be used.

DISK OPERATING SOFTWARE

The disk software, as well as being the heart of the MATRIX development system, can be used directly in OEM applications. The software consists of two programs which provide a complete disk handling facility.

INPUT/OUTPUT CONTROL SYSTEM - IOCS

The first package is called the I/O Control System (IOCS). This is a generalized blocker/deblocker which can interface to any device handler. Input and output can be done via the IOCS in any of four modes:

1. Single-byte transfer.
2. Line at a time, where the end of a line is defined by carriage return.
3. Multibyte transfers, where the number of bytes to be transferred is defined as the logical record length.
4. Continuous transfer to end-of-file, which is used for binary (RAM-image) files.

The IOCS provides easy application of I/O oriented packages to any device. There is one entry point, and all parameters are passed via a vector defined by the calling program. Any given handler defines the physical attributes of its device which are, in turn, used by the IOCS to perform blocking and deblocking.

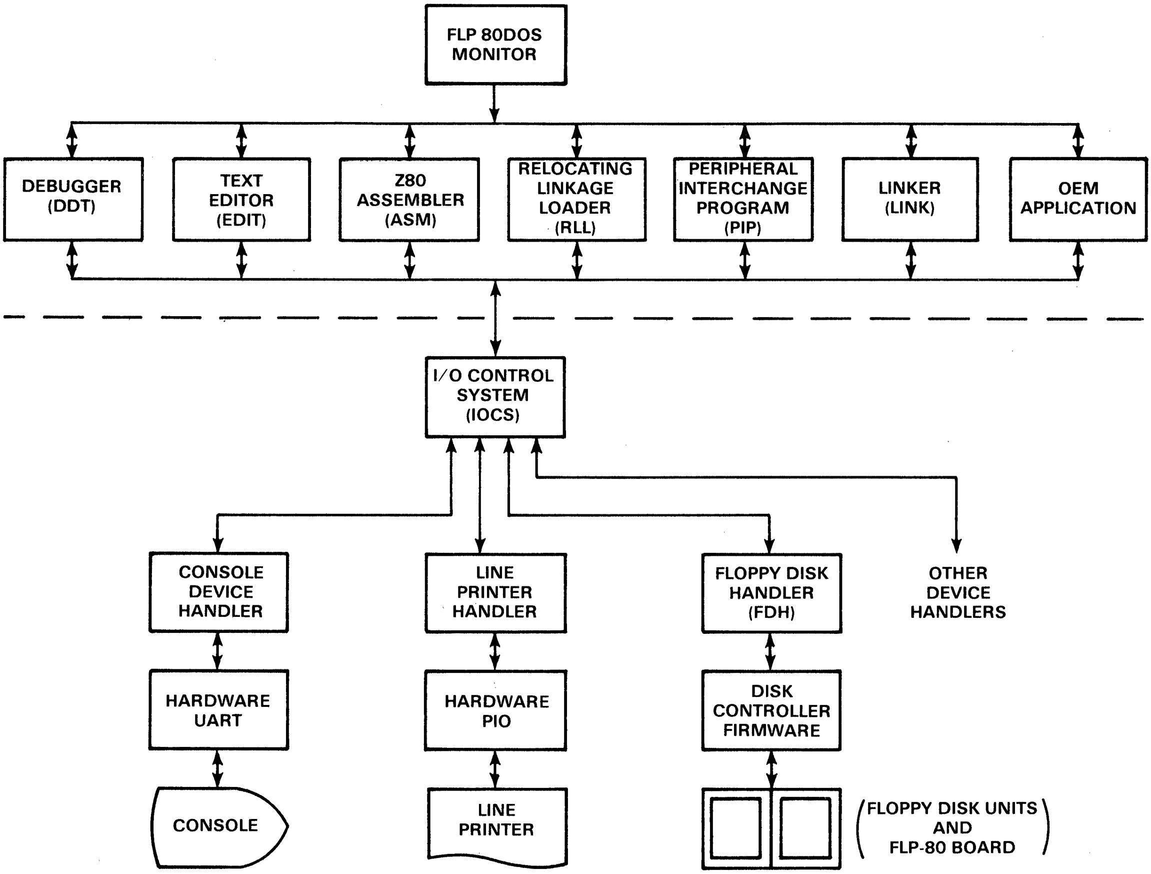
FLOPPY DISK HANDLER - FDH

The Floppy Disk Handler (FDH) interfaces from the IOCS to a firmware controller for up to four floppy disk units. The FDH provides a sophisticated command structure to handle advanced OEM products. The firmware controller interfaces to Mostek's FLP-80E Controller Board. The disk format is IBM 3740 soft sectored. The software can be easily adapted to double-sided and double-density disks. The Floppy Disk Handler commands include:

— erase file
— create file
— open file
— close file
— rename file
— rewind file
— read next n sectors
— reread current sector
— read previous sector
— skip forward n sectors
— skip backward n sectors
— replace (rewrite) current sector
— delete n sectors

The FDH has advanced error recovery capability. It supports a bad sector map and an extensive directory which allows multiple users. The file structure is doubly-linked to increase data integrity on the disk, and a bad file can be recovered from either its start or end.

FLP-80 DOS BLOCK DIAGRAM



ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NO.
FLP-80DOS	SDE based development system software (SD PROMs)	MK78142
FLP-80DOS	MD based development system software (MD PROMs)	MK77962
	FLP-80DOS Operations Manual Only	MK78557

MOSTEK®

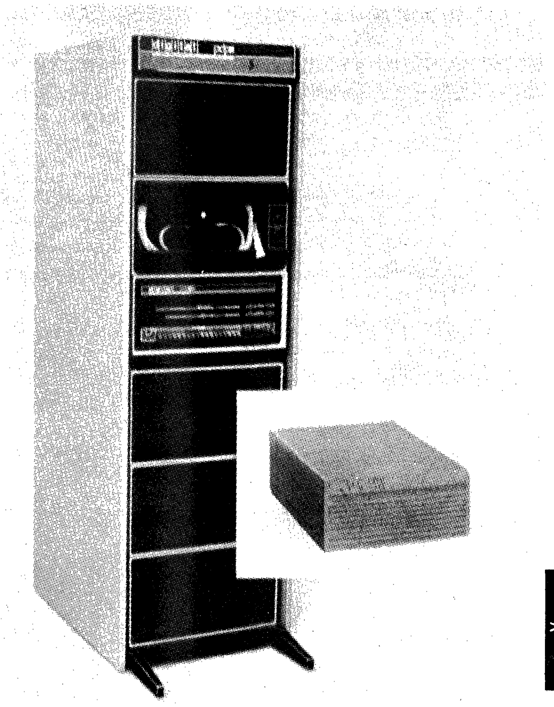
Z80 MICROCOMPUTER SOFTWARE SUPPORT

FORTRAN IV Cross Assembler (XFOR-80)

FEATURES

- □ ANSI-FORTRAN IV Source
- □ Executes on most 8-32 bit word length machines
- □ Cross Assembler is machine independent for:
  - Character representation (ASCII or BCD)
  - Numerical representation (1's or 2's complement)
- □ I/O logical device assignments are user definable
- □ 2 pass assembly easily accomodated if no secondary storage is available
- □ Memory required: 20K words (typical)
- □ Assembles all standard Z80 source statements and MACROs
- □ Size of program to be assembled is limited only by memory available for symbol table.
- □ Includes the following pseudo-ops:

● ORG	–	Program Origin
● EQU	–	Equate
● DEFL	–	Define Label ('Set')
● DEFM	–	Define Message (ASCII Text)
● DEFB	–	Define Byte
● DEFW	–	Define Word
● DEFS	–	Define Storage
● END	–	End Statement
● MACR	–	MACRO Definition
● ENDM	–	End MACRO Definition



DESCRIPTION

The XFOR-80 is a Cross Assembler for assembling Z80 source programs into the corresponding machine code for the Z80 microprocessor.

The XFOR-80 Cross Assembler is written in ANSI FORTRAN IV. It may be compiled and executed on any computer system which has at least a 20K words memory for program storage. The Cross Assembler is independent of machine character representation (ASCII, BCD, etc.) and numerical representation (2's complement, 1's complement, etc.) Logical device assignments are set up in the source of the main program module, and may be easily changed to suit the installation. Also, if no secondary storage is available the main program may be changed to accomodate reading of the user input for the second phase of the assembly.

ORDERING INFORMATION

The XFOR-80 is available directly from MOSTEK by filling out a copy of the Software Licensing Agreement printed on the back of this data sheet and returning it with the appropriate payment on Customer Purchase Order to:

MOSTEK CORPORATION
Microprocessor Systems Dept.
1215 West Crosby Road
Carrollton, Texas 75006

Order Number	Description
MK 78117	X FOR-80

STANDARD SOFTWARE LICENSE AGREEMENT

All Mostek Corporation products are sold
on condition that the Purchaser agrees to
the following terms:

1. The Purchaser agrees not to sell, provide, give away, or otherwise make available to any unauthorized persons, all or any part of, the Mostek software products listed below; including, but not restricted to: object code, source code and program listings.

2. The Purchaser may at any time demonstrate the normal operation of the Mostek software product to any person.

3. All software designed, developed and generated independently of, and not based on, Mostek's software by purchaser shall become the sole property of purchaser and shall be excluded from the provisions of this Agreement. Mostek's software which is modified with the written permission of Mostek and which is modified to such an extent that Mostek agrees that it is not recognizable as Mostek's software shall become the sole property of purchaser.

4. Purchaser shall be notified by Mostek of all updates and modifications made by Mostek for a one-year period after purchase of said Mostek software product. Updated and/or modified software and manuals will be supplied at the current cataloged prices.

5. In no event will Mostek be held liable for any loss, expense or damage, of any kind whatsoever, direct or indirect, regardless of whether such arises out of the law of torts or contracts, or Mostek's negligence, including incidental damages, consequential damages and lost profits, arising out of or connected in any manner with any of Mostek's software products described below.

6. MOSTEK MAKES NO WARRANTIES OF ANY KIND, WHETHER STATUTORY, WRITTEN, ORAL, EXPRESSED OR IMPLIED (INCLUDING WARRANTIES OF FITNESS FOR A PARTIC–ULAR PURPOSE AND MERCHANTABILITY AND WARRANTIES ARISING FROM COURSE OF DEALING OR USAGE OF TRADE) WITH RESPECT TO THE SOFTWARE DESCRIBED BELOW.

The Following Software Products Subject To this Agreement:

Order Number   Description   Price*

Ship To: Bill To:

Method of Shipment: Customer P.O. Number

Agreed To:

PURCHASER   MOSTEK CORPORATION

By: By:

Title: Title:

Date: Date:

*Prices Subject To Change Without Notice

MOSTEK®

SOFTWARE DISK BASED

LIB-80-V1

MK78164

FEATURES

□ Includes 23 useful subroutines and programs for the Z80, including:
- Lawrence Livermore Lab's Basic
- Generalized sort program for up to eight fields per record
- 8080 - Z80 source-code converter
- Fast disk-to-disk copy utility
- Hexadecimal Dump Utility to dump memory on files
- Assembly Language Formatter Utility to format Z80 source into columns
- Word Processor Program Version 2.0, used to format documents
- Disk Recovery Utility used to recover bad disk files

□ All programs are supplied in source, object, and binary format with complete documentation on a standard FLP-80DOS diskette

□ Requires FLP-80DOS Version 2.0 or higher

DESCRIPTION

The Mostek FLP-80DOS Software Library is a collection of programs of general utility that run under FLP-80DOS Version 2.0 or higher. These programs are used quite extensively at Mostek. They are being offered in source format on diskette so that the user may not only use them as supplied, but may use them as a base for individually-tailored software.

This software library differs from other libraries in that all programs in the library have been developed or modified in-house. All programs in the library are in use at Mostek and all have some utility.

The FLP-80DOS Software Library Volume 1 consists of a User's Guide and two diskettes containing the source and binary (or object for subroutines) forms for each one of the twenty-three included programs. In order to reduce the cost of the library, printed source listing are not supplied. The user can obtain a source listing easily by assembling the required source program. A brief User's Guide is a part of each program source.

The FLP-80DOS Software Library is a "Level 2" product. "Level 2" software products are supplied by Mostek but are not supported in the areas of technical assistance or updates.

ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NO.
LIB-80 Volume 1	Software Library including source, object, and binary formats on diskette, and a printed user's guide.	MK78164
	Software Library Operation Manual Only	MK79621

MOSTEK®

SOFTWARE DISK BASED

MACRO-80

MK78165

FEATURES

□ Assembles standard Z80 instruction set to produce relocatable, linkable, object modules

□ Provides nested conditional assembly, an extensive expression evaluation capability, and an extended set of assembler pseudo-ops:

ORG	- origin
EQU	- equate
DEFL	- set/define macro label
DEFM	- define message
DEFB	- define byte
DEFW	- define word
DEFS	- define storage
END	- end of program
GLOBAL	- global symbol definition
NAME	- module name definition
PSECT	- program section definition
IF/ENDIF	- conditional assembly
INCLUDE	- include another file in source module
LIST/NLIST	- list on/off
CLIST	- code listing only of macro expansions
ELIST	- list/no list of macro expansions
EJECT	- eject a page of listing
TITLE	- place title on listing

□ Provides options for obtaining a printed cross-reference listing, terminating after pass one if errors are encountered, redefining standard Z80 opcodes via macros, and obtaining an unused-symbol reference table.

□ Provides the most advanced macro handling capability in the microcomputer market which includes:
- optional arguments
- default arguments
- looping capability
- global/local macro labels
- nested/recursive expansions
- integer/boolean variables
- string manipulation
- conditional expansion based on symbol definition
- call-by-value facility
- expansion of code-producing statements only
- expansion of macro-call statement only

□ Listing and object modules can be output on disk files or any device.

□ Compatible with other Mostek Z80 assemblers and FLP-80DOS Version 2.0 or higher. Requires 32K or more of system RAM.

DESCRIPTION

MACRO-80 is an advanced upgrade from the FLP-80DOS Assembler (ASM). In addition to its macro capabilities, it provides for nested conditional assembly and allows symbol lengths of any number of characters. It supports global symbols, relocatable programs, a symbol cross-reference listing, and an unused-symbol reference table. MACRO-80 is upward compatible with all other Mostek Z80 assemblers.

The Mostek Z80 Macro Assembler (MACRO-80) is designed to run on the Mostek Dual-Disk Development System with 32K or more of RAM. It requires FLP-80DOS, Version 2.0 or higher. Macro pseudo-ops include the following:

MACRO/MEND	- define a macro
MNEXT	- step to next argument
MIF	- evaluate expression and branch to local macro label if true
MGOTO	- branch to local macro label
MEXIT	- terminate macro expansion
MERROR	- print error message in listing
MLOCAL	- define local macro label

Predefined macro-related parameters include the following:

%NEXP	- current number of this expansion
%NARC	- number of arguments passed to expansion
#PRM	- expand last-used argument
%NPRM	- number of last-used argument
%NCHAR	- number of characters in argument

The operations manual describes in detail all facilites available in MACRO-80 and provides a host of examples and sample print-outs.

ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NO.
MACRO-80	Z80 Macro Assembler, binary program supplied on a standard FLP-80DOS diskette, with Operations Manual.	MK78165
	MACRO-80 Operations Manual	MK79635

MOSTEK®

DEVELOPMENT SYSTEMS EMULATION BOARDS

AIM-Z80BE

MK78205

HARDWARE FEATURES

□ Direct interface to Mostek's development system

□ In-circuit emulation of 2.5, 4.0, and 6.0 MHz Z80 microprocessors.

□ Real-time execution (6 MHz - no wait states)

□ Flexible breakpoints (hardware and eight single-byte software)

□ Single-step execution

□ 32K bytes emulation RAM

□ Memory Mappable into Target system in 256 byte blocks

□ Illegal write to memory detection

□ Nonexistent memory access detect

□ Forty-eight channel by 1024 words history memory

□ Event counter

□ Delay counter

□ Execution T-state timer

□ Keyboard escape function

SOFTWARE FEATURES

□ Simple-to-use, single character commands

□ Flexible display format includes disassembly of opcodes

□ System configuration parameters stored on disk for future use

DESCRIPTION

AIM-Z80BE is an advanced development tool which provides debug assistance for both software and hardware via in-circuit emulation of the Z80 microprocessor. Use of the AIM-Z80BE is completely transparent to the user's final system configuration (referred to as the Target). No memory space or ports are used and all signals including $\overline{\text{RESET}}$, $\overline{\text{INT}}$ and $\overline{\text{NMI}}$ are functional during emulation.

AIM-Z80BE SYSTEM PHOTOGRAPH

Single-step circuitry allows the user to execute Target instructions one at a time to see the exact effect of each instruction. Single step is functional in ROM as well as RAM.

Sixteen K bytes of emulation RAM may be mapped into the Target memory map at any desired address so that software may be developed even before Target memory is available.

Breakpoint-detect circuitry allows real-time execution to proceed to any desired point in the user's program and then terminate with all registers and status information saved so that execution may later be resumed. Real-time execution may also be terminated at any time by enabling the Escape Key. EVENT and DELAY counters give added flexibility for viewing the exact point of interest in the user's program.

A 48-channel history circuit will simultaneously record any bus transaction which the user may desire to see. The address bus, data bus and control signals plus eighteen external probes which can be used to monitor the Target system's circuitry at other points are recorded by the History circuit.

*Trademark of Mostek Corporation

AIM-Z80BE INITIALIZATION
EXAMPLE 1

```
$AIMZ80(CR)             <Run AIM-Z80BE control program      (1)
AIMZ80 VERSION 1.0      <Sign on message                    (2)
,I E000,EFFF(CR)        <Initialize Target memory map       (3)
IS THIS BLOCK SYSTEM MEMORY?  (Y/N) Y (CR)
ARE WRITES ALLOWED?  (Y/N) Y (CR)
,I (CR)                 <Display Target memory map          (4)
S = SYSTEM MEMORY  T = TARGET MEMORY  P = WRITES PROTECTED

  0000  T  T  T  T    T  T  T  T    T  T  T  T    T  T  T  T
  1000  T  T  T  T    T  T  T  T    T  T  T  T    T  T  T  T
  2000  T  T  T  T    T  T  T  T    T  T  T  T    T  T  T  T
  3000  T  T  T  T    T  T  T  T    T  T  T  T    T  T  T  T
  4000  T  T  T  T    T  T  T  T    T  T  T  T    T  T  T  T
  5000  T  T  T  T    T  T  T  T    T  T  T  T    T  T  T  T
  6000  T  T  T  T    T  T  T  T    T  T  T  T    T  T  T  T
  7000  T  T  T  T    T  T  T  T    T  T  T  T    T  T  T  T
  8000  T  T  T  T    T  T  T  T    T  T  T  T    T  T  T  T
  9000  T  T  T  T    T  T  T  T    T  T  T  T    T  T  T  T
  A000  T  T  T  T    T  T  T  T    T  T  T  T    T  T  T  T
  B000  T  T  T  T    T  T  T  T    T  T  T  T    T  T  T  T
  C000  T  T  T  T    T  T  T  T    T  T  T  T    T  T  T  T
  D000  T  T  T  T    T  T  T  T    T  T  T  T    T  T  T  T
  E000  S  S  S  S    S  S  S  S    S  S  S  S    S  S  S  S
  F000  T  T  T  T    T  T  T  T    T  T  T  T    T  T  T  T
```

USING THE AIM-Z80BE

AIM-Z80BE is partitioned on three modules. The Control and History modules are installed directly into the development system. Cables from these modules connect to the Buffer module which plugs directly into the Target system's Z80 CPU socket. After AIM-Z80BE is installed, the development system is powered up and the system booted up as normal. All development system software and hardware are still functional. The software to control AIM-Z80BE (AIMZ80) is initialized by using the implied run command. AIMZ80 will sign on, take control of the Target system and allow the user to initialize the Target system and use any of the AIMZ80 commands to load, test, and debug his Target program. An example of the use of some of the AIMZ80 commands is given in the following examples.

In the above example the system was initialized by entering AIMZ80 (1), the sign on message was printed (2), and then the I command was used to map Target address space E000 through EFFF to use the System Emulation RAM (3). The I command with no operands was then used to display the current Target memory map configuration (4).

SOFTWARE BREAKPOINT
EXAMPLE 2

```
,M E000 (CR)                    <Examine Target memory starting at address E000     (1)
E000 3E (CR)
E001 00 (CR)
E002 3C (CR)
E003 3C (CR)
E004 C3 (CR)
E005 00 (CR)
E006 E0 .

,B E004 (CR)                    <Set software breakpoint at address E004            (2)
,E E000 (CR)                    <Begin execution at address E000                    (3)
,SWBP ENCOUNTERED               <Software breakpoint encountered                    (4)
 PC    AF    I IF   BC    DE    Hl   A'F'  B'C'  D'E'  H'L'   IX    IY    SP
E004  0200  0040  02FF  0000  0200  0041  0101  FFFF  0200  FFFF  FFFF  01E9

,T 4, -4 (CR)                   <Tabulate 4 history samples starting 4 samples     (5)
                                 before the breakpoint
OFFS ADDR DB DISASSEMBLY          TYPE  PA8------PA0    PB8------PB0
-004 E000 3E LD A,0               FETC  1 1111  1111    1 1111  1111
-003 E001 00                      MRD   1 1111  1111    1 1111  1111
-002 E002 3C INC A                FETC  1 1111  1111    1 1111  1111              (6)
-001 E003 3C INC A                FETC  1 1111  1111    1 1111  1111
+000 E004 5B BREAKPOINT CODE      FETC  1 1111  1111    1 1111  1111
,B C,A (CR)                     <Clear all breakpoints                              (7)
```

In this example, the M command is used to examine Target memory locations E000 through E006 (1), then a software breakpoint is set at address E004 (2), and execution is started at address E000 (3). The Software breakpoint is encountered at (4), and the current state of the Z80 registers is displayed. The T command is then used to tabulate 4 history samples starting 4 samples prior to the breakpoint last encountered (5). Carriage Return is then entered to display the next sample which is the breakpoint code (6). Next all breakpoints are cleared (7).

SPECIFYING HISTORY OPTIONS
EXAMPLE 3

```
,B H,0 (CR)                  <Specify history options                    (1)


                   PA7----PA0  A15-----------------A0
TRIGGER WORD IS:   XXXX  XXXX  0000  0000  0000  0000
UPDATE:            XXXX  XXXX  1110  0000  0000  0000                    (2)


TRIGGER STROBE IS (MRD,LE)  ;TO CHANGE SELECT ONE:                  (3)
      MRD(0)    MWR(1)    MREQ(2)    IORD(3)
     IOWR(4)   IORQ(5)    INTA(6)    PA8(7)    [ ,LE(8) ,TE(9)]

EVENT COUNT IS:  0001 2(CR)                                              (4)

DELAY COUONT IS:  000 4(CR)                                              (5)

HISTORY CLOCK IS (MRD MWR IORD IOWR )  ;TO CHANGE SELECT ANY:            (6)
      MRD(0)    MWR(1)    MRF(2)
     IORD(3)   IOWR(4)   PA8+(5)    PA8-(6) 0,1,2,3,4(CR)

HISTORY CLOCK ENABLE IS (ALL) CYCLES ;TO CHANGE SELECT ONE:              (7)
      ALL(0)      DMA(1)     CPU(2)    PA7L(3)     PA7H(4)
                TWORD(6)          [ , PB8L(8)    ,PB8H(9)]
```

In this example, the H,O option is selected to specify History or Hardware breakpoint options (1). The system then allows the user to update the trigger word (2), the trigger strobe (3), the EVENT COUNT (4), the DELAY COUNT (5), the history clock source (6), and finally the history clock enable signal.

HARDWARE BREAKPOINT
EXAMPLE 4

```
,B E000,H(CR)          < Set hardware breakpoint at E000                 (1)
,E E000 (CR)           < Begin execution at E000                         (2)
,HWBP ENCOUNTERED      < Hardware breakpoint encountered                 (3)


 PC   AF    I IF  BC    DE    HI    A'F'  B'C'  D'E'  H'L'  IX    IY    SP
E003 0100  0040  02FF  0000  0200  0041  0101  FFFF  0200  FFFF  FFFF  01E9


,T(CR)                 < Tabulate history starting at earliest sample.   (4)


OFFS ADDR DB DISASSEMBLY    TYPE    PA8------PA0    PBS------PB0      (5)
-00B E000 3E LD   A,0       FETC    1 1111 1111     1 1111 1111
-00A 0038 3E                RFSH    1 1111 1111     1 1111 1111
-009 E001 00                MRD     1 1111 1111     1 1111 1111
-008 E002 3C INC  A         FETC    1 1111 1111     1 1111 1111
-007 0039 3C                RFSH    1 1111 1111     1 1111 1111
-006 E003 3C INC  A         FETC    1 1111 1111     1 1111 1111
-005 003A 3C                RFSH    1 1111 1111     1 1111 1111
-004 E004 C3 JP   ;0E000H   FETC    1 1111 1111     1 1111 1111
-003 003B C3                RFSH    1 1111 1111     1 1111 1111
-002 E005 00                MRD     1 1111 1111     1 1111 1111
-001 E006 E0                MRD     1 1111 1111     1 1111 1111
+000 E000 3E LD   A,0       FETC    1 1111 1111     1 1111 1111
+001 003C 3E                RFSH    1 1111 1111     1 1111 1111
+002 E001 00                MRD     1 1111 1111     1 1111 1111
+003 E002 3C INC  A         FETC    1 1111 1111     1 1111 1111
+004 003D 3C                RFSH    1 1111 1111     1 1111 1111
END OF HISTORY
```

In this example, a hardware breakpoint is set at address E000 (1), and execution is started at address E000 (2). After the EVENT COUNT and DELAY COUNT are satisfied, the hardware breakpoint is encountered (3) and the Z80 registers are printed. Next the T command is used to tabulate the history memory starting at the earliest sample (4). The history memory tabulation gives the offset from the breakpoint, the address bus, the data bus, the disassembled opcodes, the type of bus cycle, the A probes and the B probes (5).

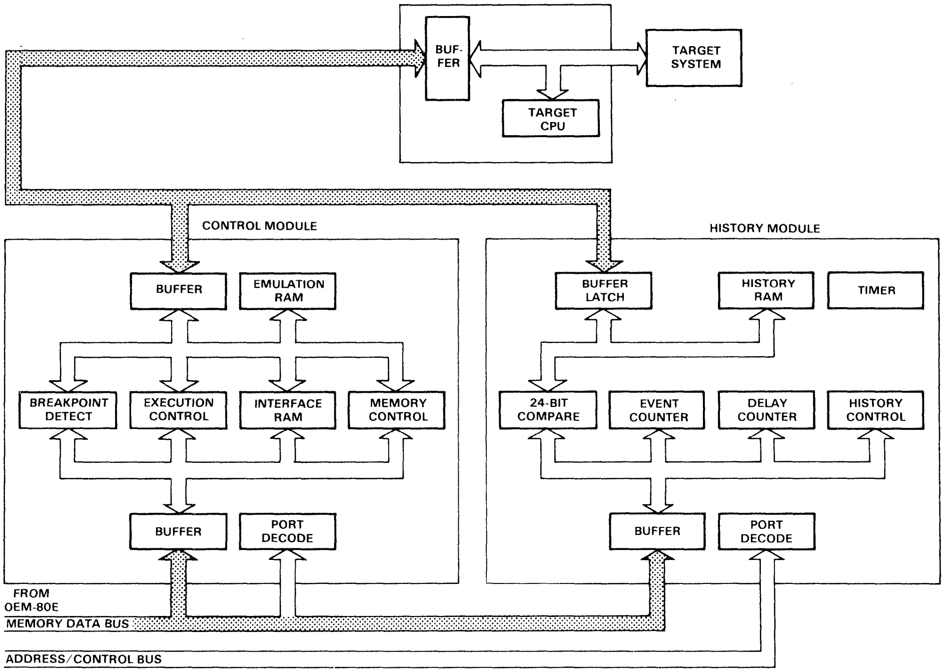
BLOCK DIAGRAM DESCRIPTION

As shown in the block diagram, AIM-Z80BE consists of three modules. The buffer module, which contains the Target Z80 CPU, plugs directly into the Target system CPU socket. Address, data and control signals are buffered and cabled to the Control and History modules which are installed in the development system.

The control module has the circuitry for detecting the breakpoint conditions. It forces execution to begin in the System Interface RAM which is loaded with an interface program and is shadowed into the Target memory space. This control program makes the Target CPU a slave to the development system. When the user desires to resume execution, the control program activates the execution control circuit and execution resumes at the desired address. The 16K byte emulation RAM may be mapped to appear at any address space in the Target memory map.

The History module has a 24-bit comparator circuit to detect the hardware breakpoint condition, the event counter, and the delay counter. Sampling into the 48 channel-by-1024 word history RAM is controlled by the History control circuit. The Timer circuit is used to count Target processor clocks for logging elapsed execution time.

AIM-Z80BE BLOCK DIAGRAM



AIM-Z80BE SOFTWARE

AIMZ80 is the software designed to operate the AIM-Z80BE system on Mostek's Dual Floppy Disk Microcomputers. It is supplied on standard FLP-80DOS diskette. The software has the same command structure as other Mostek debuggers. The commands available with AIMZ80 are summarized below. Designations s,f, and d stand for operands.

,B s,f	Set hardware or software breakpoint at memory location s.
,C s,f, d	Copy the Target memory block s through f to Target memory starting at d.
,D s,f	Dump the Target memory block s through f to any desired disk file.
,E s,f	Begin real time execution at Target memory address s with an optional breakpoint set at f.
,F s,f,d	Fill the Target memory block s through f with data d.
,G s	Get binary file s and load it into Target memory.
,H	Hexadecimal arithmetic.
,I s,f	Initialize the memory map for the Target memory block s through f.
,L s,f,d	Locate data d in Target memory range s through f.
,M s	Display and update Target memory at location s.
,M s,f,d	Tabulate Target memory locations s through f. Option d specifies disassemble of Target memory.
,O s	Set relative offset equal to s for all address operands. Extremely useful in debugging relocatable modules.
,P s	Display and update Target port number s.
,Q	Quit and return to FLP-80DOS Monitor.
,R s,f	Display Target registers, Option s specifies the number of registers to be displayed and option f specifies if the heading is to be included.
,S s,f	Single step starting at Target location s for f number of steps.
,T s,f	Tabulate s locations of the history RAM starting at an offset of f locations from the breakpoint address.
,V s,f,d	Verify Target memory block s through f against file d
,W s	Write in parallel to logical unit s all output.

Target system programs are developed using the Mostek resident assembler or Macro assembler and linked using the resident Linker. AIMZ80 is then used to complete debugging on the user's Target system.

ELECTRICAL SPECIFICATIONS

Operating Temperature Range 0°C to +50°C

Target Power Supply Requirements (typical)
+5V 5% @ 500mA

System Power Supply Requirements (typical)
+5V 5% @ 2.5A
+12V 5% @ 100mA
-12V 10% @10mA

Interface- MATRIX and SYS-80F compatible

Operating Frequency - 500KHz to 4MHz (Z80 PHI clock)

Target Interface - All signals meet the specifications for the MK3880-6 (6.0 MHz CPU) with the following exceptions: exceptions:

1. The output low voltage is 0.5 v max at 1.8 mA for the ADDRESS, DATA, $\overline{\text{IORQ}}$, $\overline{\text{RFSH}}$, $\overline{\text{HALT}}$, and $\overline{\text{BUSAK}}$ signals.
2. The input low current is 400μa max for the PHI clock, $\overline{\text{RESET}}$, $\overline{\text{INT}}$, $\overline{\text{NMI}}$, and DATA signals.
3. The input high current is 20μa max for the PHI clock, $\overline{\text{RESET}}$, $\overline{\text{INT}}$, $\overline{\text{NMI}}$, and DATA signals.
4. The signals $\overline{\text{M1}}$, $\overline{\text{MREQ}}$, $\overline{\text{RD}}$, and $\overline{\text{WR}}$ have a **maximum of 25 ns added propagation delay.**
5. The input signals $\overline{\text{RESET}}$, $\overline{\text{INT}}$, and $\overline{\text{NMI}}$ have a maximum of 45ns added propagation delay.

ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NO.
AIM-Z80BE	Includes the AIM-Z80BE-16 in circuit eulation control module, histroy module, buffer modules, cabling operation manual and software on diskette. The emulation control module is capable of simulating up to 32K bytes of Target memory.	MK78205
	AIM-Z80BE Operations manual only.	MK79650

1981 Z80 MICROCOMPUTER DATA BOOK

I Table of Contents

II General Information

III Z80 Family Technical Manuals

IV Z80 Family Data Sheets

V Z80 Development Equipment

VI Z80 Microcomputer Application Notes

VII Z80 Microcomputer Peripherals

VIII Z80 Military/ Hi-Rel

VI Z80 MICRO-COMPUTER APPLICATION NOTES

MOSTEK®

ADD SERIAL COMMUNICATION CAPABILITY TO THE 8086/8088 FAMILY USING THE Z80 SIO

Application Note

INTRODUCTION

Since its introduction, the SIO (MK3884/5/7) has been widely recognized as one of the most powerful serial communications devices commercially available. The SIO is a dual-channel, multifunction peripheral device capable of handling a wide variety of serial data communications requirements in microcomputer systems. The system designer can configure the SIO for the personality of almost any system, as each channel is software programmable. The communications power of the SIO does not need to be limited to those applications where a Z80 CPU is being used. To more fully explain how it can be used with other processors is the basis of this application note. The information presented here can be used to adapt the use of the SIO not only to the 8086/8088 family but also to other microprocessors. All references made to the 8086 hereafter apply also to the 8088, as the hardware and software implementation is identical for each.

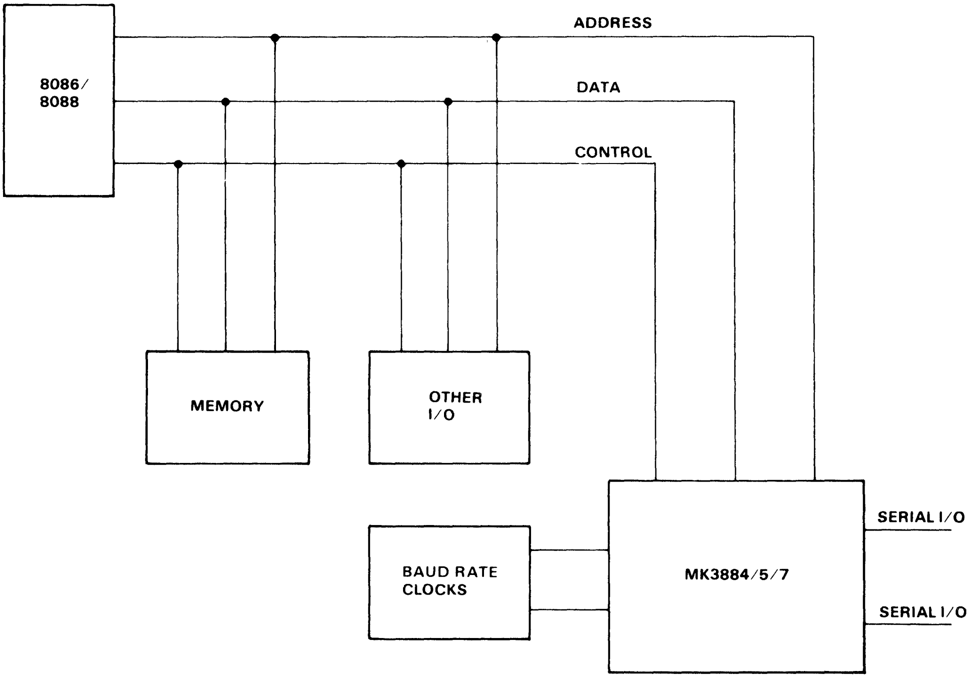
DESCRIPTION

The MK3884/5/7 (SIO) is a dual full duplex USART device that includes special logic to allow it to handle special purpose serial protocols such as SDLC and HDLC. Three bonding options are available to the user allowing alternate signals to be brought out to some pins for a particular application. For example, in some cases the receive and transmit clocks for the second channel are both needed because they will be at different frequencies. A user desiring this can use the MK3887 but the $\overline{\text{SYNC}}$ signal must be sacrificed. Other alternative signal selections are brought out on the MK3884 and MK3885. This flexibility and the fully dual nature of the SIO give it particular appeal in designing serial communications equipment.

The system that will be considered is a very simplified 8086 based minimum mode system which will use the SIO as a serial communications port interfaced to a standard ASCII terminal. It is assumed that the device will be memory mapped to avoid the use of the limited I/O instructions of

SYSTEM BLOCK DIAGRAM

Figure 1



VI
Z80
MICRO-
COMPUTER
APPLICATION
NOTES

the 8086 family. Interrupts will be used for the receive side of channel A to signal the processor that another character has been received. Figure 1 shows a simplified block diagram of the system.

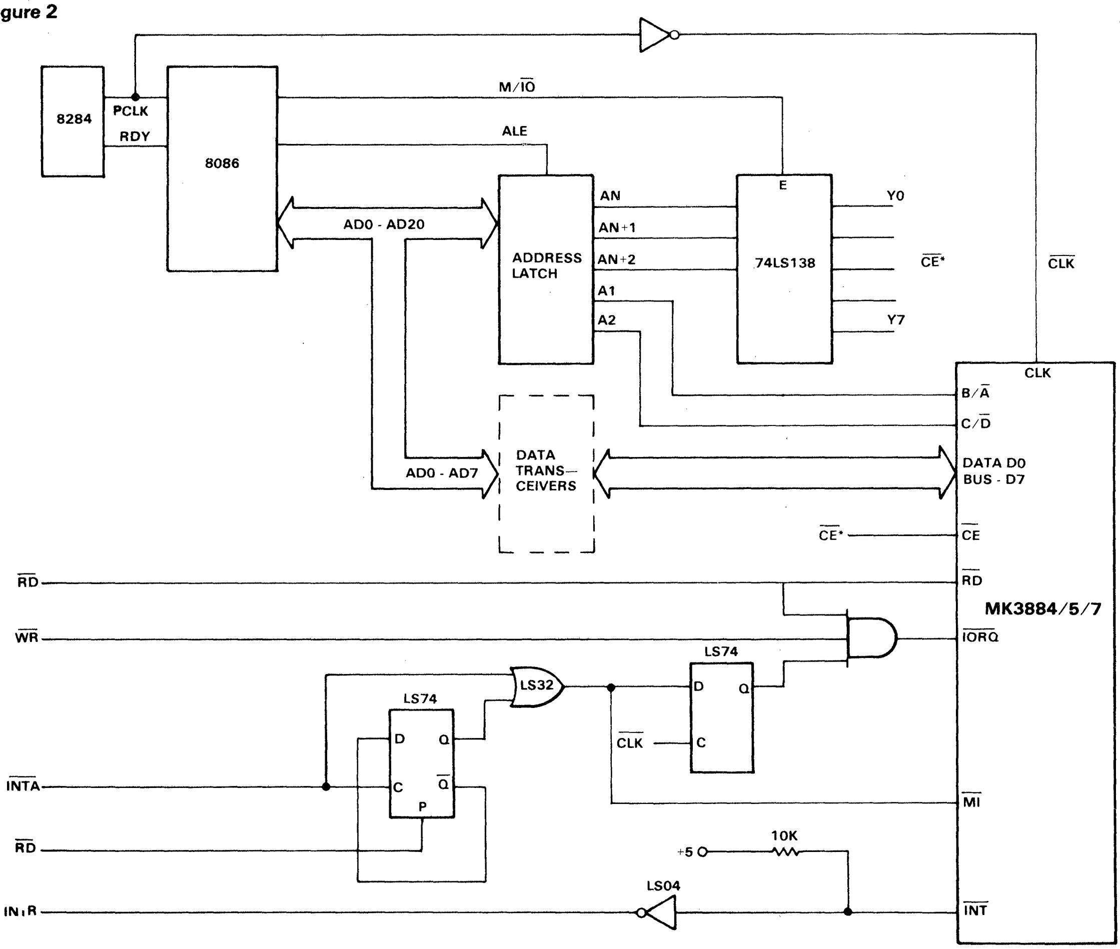
There are several control signals used by the SIO which are peculiar to the Z80 family of devices and as such, must be given special consideration in using this device with another processor.

1. $\overline{\text{A}}$/B and C/$\overline{\text{D}}$	(inputs)
2. $\overline{\text{CE}}$	(input)
3. $\overline{\text{M1}}$	(input-needed for interrupts)
4. $\overline{\text{IORQ}}$	(input)
5. $\overline{\text{RD}}$	(input)
6. $\overline{\text{INT}}$	(output)

The most readily obtainable signals are $\overline{\text{A}}$/B and C/$\overline{\text{D}}$ corresponding to A1 and A2 of the address bus, respectively. Note that the address bus must first be latched by the ALE signal from the 8086. (NOTE: A0 is not an address but a bank select signal.) For the sake of programming, we will consider the SIO as a memory mapped device as suggested earlier. We will also want to use the interrupt capability of the device and that will be covered later. Since the address signals are latched out of the processor first, there is not a timing problem with the interface to the SIO if the chip enable signal is decoded well before any data is available. The $\overline{\text{CE}}$ signal is derived from address decoding the upper address bits. This can be done using a BIPOLAR PROM or a decoder such as the 74LS138 as shown in Figure 2. It is important to note that the $\overline{\text{CE}}$ signal will be valid only during a memory cycle through the use of the M/$\overline{\text{IO}}$ line from the 8086. In a maximum 8086 system this must be done with the signals from the Bus Controller Device.

$\overline{\text{M1}}$ is used by the SIO only during the interrupt acknowledge cycle and the interrupt return cycle. To handle interrupts with the 8086, the $\overline{\text{INT}}$ signal is inverted between the SIO and the processor to form the INTR signal with it's logic true high. When the 8086 detects an interrupt, it issues an $\overline{\text{INTA}}$ (interrupt acknowledge) signal twice; first in conjunction with LOCK to lock out any possible outside bus requests and then a second time to initiate the interrupt

8086 MINIMUM SYSTEM USING THE MK3884/5/7
Figure 2



vector response. This second $\overline{\text{INTA}}$ is detected by excluding the first $\overline{\text{INTA}}$ using a toggle flip-flop which is then applied to the SIO as $\overline{\text{M1}}$. This $\overline{\text{M1}}$ is delayed using a flip-flop and gated with $\overline{\text{CE}}$ to form the necessary $\overline{\text{IORQ}}$ signal. The combination of the $\overline{\text{M1}}$ followed by the $\overline{\text{IORQ}}$ signals the SIO that an interrupt acknowledge cycle is being performed. At that point the SIO will place its programmed interrupt vector on the data bus for use by the processor. At the end of the interrupt service the SIO would normally be reset by the detection of the two byte opcode ED and 4D. With a non-Z80 processor the SIO can be reset by the software interrupt reset command. This will be discussed later.

The $\overline{\text{IORQ}}$ signal is used to signal the SIO that an I/O instruction is being performed. Since we have assumed that the device will be memory mapped, the $\overline{\text{IORQ}}$ signal can be simply the $\overline{\text{RD}}$ or $\overline{\text{WR}}$ signal gated with the delayed $\overline{\text{M1}}$ signal discussed previously. If the $\overline{\text{CE}}$ signal is not fully decoded, a problem can occur if other devices are similarly accessed. Therefore, it is suggested that full decoding be used to allow unrestricted use of the remainder of the memory space.

The $\overline{\text{RD}}$ (read) signal comes directly from the 8086 and is applied to the SIO as such. This is to insure that the device timing is maintained.

The pull-up resistor and inverter on the $\overline{\text{INT}}$ (interrupt) line are necessary to insure that the INTR signal to the 8086 is logic false (low) for all conditions except an actual interrupt where the SIO pulls $\overline{\text{INT}}$ low. If more than one SIO were being used, the $\overline{\text{INT}}$ signals from all the devices would be wire-ored together before going to the inverter.

In most designs the introduction of the logic delays in the control signals will not adversely affect the timing necessary for proper operation of the SIO with the 8086. However, if the SIO is to be on a peripheral card or isolated from the processor by data, address, and control buffers, then the timing interrelations for the SIO as called out on the data sheets must be carefully examined to insure that proper set up and delay times are maintained. This is especially critical in the interrupt acknowledge cycle where the SIO is attempting to pass a vector back to the processor. Some caution, therefore, must be taken in respect to buffer direction & tri-state control.

SOFTWARE

Software needed to drive the SIO is most easily implemented as a block of command and data bytes that are transferred to the SIO using a memory block move type of operation. This is the primary reason for memory mapping the device. The actual programming of the SIO will be covered in another application note, but it is important to point out that the sequence of initialization commands and control commands for interrupt servicing is critical. Interrupt service routines are similar to other 8086 type services with the exception that the service routine for the SIO must have a Write Register 0, Command 7, as its last instruction to reset the internal interrupt hardware of the SIO so another interrupt can be generated.

OPERATION

The maximum clock frequency of the SIO is 4 MHz, and for test purposes the clock speed was 3 MHz. This clock must be synchronized to the clock of the 8086 and inverted to insure proper data transfers. This will limit the 8086 to a clock of 4 MHz, unless special speed-up/slow-down hardware is designed to synchronize the data transfers when differing clock rates are used. Under normal operation of the SIO and 8086 at 4 MHz, the SIO will not have a problem keeping pace with the processor because of its ability to transfer at very high data rates. Caution should be taken if the SIO clock is not 50% duty cycle to ensure the clock high (min.) and clock low (min.) signals are not violated. If synchronous data transfers at very high clock rates are to be done with the SIO, it may be necessary to use a direct memory access controller to load the transmit buffer of the SIO to insure that it does not become prematurely empty. If the transmit buffer does become empty before the end of the transmission, the block of data being transmitted may be terminated before all the data has been transferred. Successful operation of the SIO in very high speed systems has demonstrated that the SIO is capable of the most sophisticated data transfers with the least interface to the processor of a system.

PROGRAM EXAMPLE

Table 1 illustrates an echo program showing the initialization and transfer of data between the 8086 CPU, the Z80 SIO, and a terminal. Proper initialization includes first resetting B and loading the interrupt vector '10'H to the SIO. In the event of an interrupt, '10'H will be read by the 8086, multiplied by 4 internally, and hence provide an interrupt look up table located at '40'H as indicated in Table 1. As shown, channel A is configured for data communications at 9600 baud, no parity, one stop bit, and 8 bits / character. Reference should be made to the Mostek MK3884 technical manual for proper SIO initialization procedures. Worthy of note are the command strings incorporated for SIO initialization. This technique is analogous to the efficient Z80 OTIR instruction. Also of interest and necessity is the software return from interrupt. When the Z80 SIO is in an interrupt driven environment, it must see an RETI instruction (ED 4D) on its data bus in order to reset the internal interrupt logic. In non-Z80 CPU environments, however, the interrupt reset may be effected by writing a '38'H to the appropriate command/status register of the SIO. Program execution loops until interrupted by the data terminal, causing an interrupt to the starting address, '002B'H in this example.

Table 1

SI086

```
LINE    SOURCE

         **************************************************************
        * THIS PROCEDURE (SUBROUTINE) ASSUMES THE SEGMENT REGISTERS *
        * AND STACK POINTER ARE SET UP.                             *
        ************************************************************
        INT_VECTS       SEGMENT AT 0
                        ORG     40 H
                        DD      ECHO
        INT_VECTS       ENDS
        ;
        ;
        CODENAME                SEGMENT
                        ASSUME  CS:CODENAME,DS:CODENAME, SS:CODENAME,ES:CODENAME
        ;
        ;   THE SIO IS MEMORY MAPPED AT LOCATIONS 1000H-1006H
        ;
        SIO_ADATA       EQU     1000H               ;ADDRESS OF SIO CHANNEL A DATA
        SIO_BDATA       EQU     SIO_ADATA+2         ;ADDRESS OF SIO CHANNEL B DATA
        SIO_ACS         EQU     SIO_ADATA+4         ;ADDRESS OF SIO CHANNEL A
        ;                                                   COMMAND/STATUS
        SIO_BCS         EQU     SIO_ADATA+6         ;ADDRESS OF SIO CHANNEL B
        ;                                                   COMMAND/STATUS
        ;
        ;  B COMMAND STRING TO INITIALIZE CHANNEL B
        ;
        B_COMM          DB      18,02H,10H


        ;               18H     CHANNEL B RESET,WRITE REGISTER 0
        ;               02H     SET POINTER TO WRITE REGISTER 2
        ;               10H     SIO INTERRUPT VECTOR
        ;
        ;  A COMMAND STRING TO INITIALIZE CHANNEL A
        ;
        A_COM           DB      18H,04H,44H,01H,18H,03H,0C1H,05H,68H




        ;               18H     CHANNEL A RESET,WRITE REGISTER 0
        ;               04H     SET POINTER TO WRITE REGISTER 4
        ;               44H     16X CLOCK MODE,1 STOP BIT,NO PARITY
        ;               01H     SET POINTER TO WRITE REGISTER 1
        ;               18H     INTERRUPT ON ALL Rx CHARACTERS
        ;               03H     SET POINTER TO WRITE REGISTER 3
        ;               C1H     Rx 8 BITS/CHARACTER,Rx ENABLE
```

```
LINE    SOURCE

        ;               05H     SET POINTER TO WRITE REGISTER 5
        ;               68H     Tx 8 BITS/CHARACTER,Tx ENABLE
        ;
        ;   START OF SIO INITIALIZATION PROCEDURE
        ;
        SIO_INIT        PROC    NEAR
        START:          MOV     BX,SIO_BCS          ;CHANNEL B COMMAND ADDRESS
                        MOV     SI,OFFSET B_COM     ;ADDRESS OF COMMAND STRING
                        MOV     CX,LENGTH B_COM     ;STRING LENGTH IN COUNT REGISTER
                        CLD                         ;CLEAR DIRECTION FLAG,AUTO-INC.
        CBLD:           LODS    B_COM               ;    CHANNEL B
                        MOV     [BX],AL             ;    INITIALIZATION
                        LOOP    CBLD                ;    LOOP UNTIL CX=0
                        MOV     BX,SIO_ACS          ;CHANNEL A COMMAND ADDRESS
                        MOV     SI,OFFSET A_COM     ;ADDRESS OF COMMAND STRING
                        MOV     CX,LENGTH A_COM     ;STRING LENGTH IN COUNT REGISTER
        CALD:           LODS    A_COM               ;    CHANNEL A
                        MOV     [BX],AL             ;    INITIALIZATION
                        LOOP    CALD                ;    LOOP UNTIL CX=0
                        STI                         ;SET INTERRUPT FLAG
                        RET                         ;RETURN TO CALLING PROGRAM
        SIO_INIT        ENDP
        ;
        ;   END OF INITIALIZATION OF SIO
        ;
        ;   INTERRUPT SERVICE ROUTINE TO ECHO BACK CHARACTERS TO THE TERMINAL
        ;
        SIO ISR         PROC    NEAR
        ECHO:           MOV     BX,SIO ADATA        ;CHANNEL A DATA ADDRESS
                        MOV     AL,[BX]             ;GET CHARACTER
                        MOV     [BX],AL             ;ECHO TO TERMINAL
                        MOV     BX,SIO_ACS          ;CHANNEL A C/S ADDRESS
                        MOV     BYTE PTR [BX],38H           ;SOFTWARE RETURN
        ;                                                    FROM INTERRUPT
                        IRET                        ;RETURN TO INTERRUPTED PROGRAM
        SIO_ISR         ENDP
        ;
        ;   END OF INTERRUPT SERVICE ROUTINE, CHARACTER RECEIVED
        ;   FROM TERMINAL IN AL REGISTER.
        ;
        CODENAME                ENDS
                        END
```

MOSTEK®

Z80 INTERFACING TECHNIQUES FOR DYNAMIC RAM

Application Note

INTRODUCTION

Since the introduction of second generation microprocessors, there has been a steady increase in the need for larger RAM memory for microcomputer systems. This need for larger RAM memory is due in part to the availability of higher level languages such as PL/M, PL/Z, FORTRAN, BASIC and COBOL. Until now, when faced with the need to add memory to a microcomputer system, most designers have chosen static memories such as the 2102 1Kx1 or possibly one of the new 4Kx1 static memories. However, as most mini or mainframe memory designers have learned, 16-pin dynamic memories are often the best overall choice for reliability, low power, performance, and board density. This same philosophy is true for a microcomputer system. Why then have microcomputer designers been reluctant to use dynamic memory in their system? The most important reason is that second generation microprocessors such as the 8080 and 6800 do not provide the necessary signals to easily interface dynamic memories into a microcomputer system.

Today, with the introduction of the Z80, a true third generation microprocessor, not only can a microcomputer designer increase system throughput by the use of more powerful instructions, but he can also easily interface either static or dynamic memories into the microcomputer system. This application note provides specific examples of how to interface 16-pin dynamic memories to the Z80.

OPERATION OF 16-PIN DYNAMIC MEMORIES

The 16-pin dynamic memory concept, pioneered by MOSTEK, uses a unique address multiplexing technique which allows memories as large as 16, 384 bits x 1 to be packaged in a 16-pin package. For example the MK4027 (4,096x1 dynamic MOS RAM) and the MK4116 (16,384x1 dynamic MOS RAM) both use address multiplexing to load the address bits into memory. The MK4027 needs 12 address bits to select 1 out of 4,096 locations, while the MK4116 requires 14 bits to select 1 out of 16,384. The internal memories of the MK4027 and MK4116 can be thought of as a matrix. The MK4027 matrix can be thought of as 64x64, and the MK4116 as 128x128. To select a particular location, a row and column address is supplied to the memory. For the MK4027, address bits A_0-A_5 are the row address, and bits A_6-A_{11} are the column addresses. For the MK4116, address bits A_0-A_6 are the row address, and A_7-A_{13} are the column address. The row and column addresses are strobed into the memory by two negative going clocks called Row Address Strobe ($\overline{RAS}$) and Column Address Strobe ($\overline{CAS}$). By the use of $\overline{RAS}$ and $\overline{CAS}$, the address bits are latched into the memory for access to the desired memory location.

Dynamic memories store their data in the form of a charge on a small capacitor. In order for the dynamic memory to retain valid data, this charge must be periodically restored. The process by which data is restored in a dynamic memory is known as refreshing. A refresh cycle is performed on a row of data each time a read or write cycle is performed on any bit within the given row. A row consists of 64 locations for the MK4027 and 128 locations for the MK4116. The refresh period for the MK4027 and the MK4116 is 2ms which means that the memory will retain a row of data for 2ms without a refresh. Therefore, to refresh all rows within 2ms, a refresh cycle must be executed every 32µs (2ms÷64) for the MK4027 and 16µs (2ms÷128) for the MK4116.

To ensure that every row within a given memory is refreshed within the specified time, a refresh row address counter must be implemented either in external hardware or as an internal CPU function as in the Z80. (Discussed in more detail under Z80 Refresh Control and Timing.) The refresh row address counter should be incremented each time that a refresh cycle is executed. When a refresh is performed, all RAMs in the system should be loaded with the refresh row address. For the MK4027 and the MK4116, a refresh cycle consists of loading the refresh row address on the address lines and then generating a $\overline{RAS}$ for all RAMs in the system. This is known as a $\overline{RAS}$ only refresh. The row that was addressed will be refreshed in each memory. The $\overline{RAS}$ only refresh prevents a conflict between the outputs of all the RAMs by disabling the output on the MK4116, and maintaining the output state from the previous memory cycle on the MK4027.

Z80 TIMING AND MEMORY CONTROL SIGNALS

The Z80 was designed to make the job of interfacing to dynamic memories easier. One of the reasons the Z80 makes dynamic memory interfacing easier is because of the number of memory control signals that are available to the designer. The Z80 control signals associated with memory operations are:

MEMORY REQUEST ($\overline{\text{MREQ}}$) - Memory request signal indicating that the address bus holds a valid memory address for a memory read, memory write, or memory refresh cycle.

READ ($\overline{\text{RD}}$) - Read signal indicating that the CPU wants to read data from memory or an I/O device. The addressed I/O device or memory should use this signal to gate data onto the CPU data bus.

WRITE ($\overline{\text{WR}}$) - Write signal indicating that the CPU data bus hold valid data to be stored in the addressed memory or I/O device.

REFRESH ($\overline{\text{RFSH}}$) - Refresh signal indicates that the lower 7 bits of the address bus contain a refresh address for dynamic memories and the current MREQ signal should be used to generate a refresh cycle for all dynamic memories in the system.

Figures 1a, 1b, and 1c show the timing relationships of the control signals, address bus, data bus and system clock Φ. By using these timing diagrams, a set of equations can be derived to show the worst case access times needed for dynamic memories with the Z80 operating at 2.5MHz.

The access time needed for the op code fetch cycle and the memory read cycle can be computed by equations 1 and 2.

(1) $t_{\text{ACCESS OP CODE}} = 3(t_c/2) - t_{DL\overline{\Phi}(MR)} - t_{S\Phi(D)}$

where: t_C = Clock period

$t_{DL\overline{\Phi}(MR)}$ = $\overline{\text{MREQ}}$ delay from falling edge of clock.

$t_{s\Phi(D)}$ = Data setup time to rising edge of clock during op code fetch cycle.

let: $t_C = 400\text{ns}$; $t_{DL\overline{\Phi}(MR)} = 100\text{ns}$; $t_{s\Phi} = 50\text{ns}$

then: $\underline{t_{\text{ACCESS OP CODE}} = 450\text{ns}}$

(2) $t_{\text{ACCESS MEMORY READ}} = 4(t_c/2) - t_{DL\Phi(MR)} - t_{S\overline{\Phi}(D)}$

where: t_C = Clock period

$t_{DL\overline{\Phi}(MR)}$ = $\overline{\text{MREQ}}$ delay from falling edge of clock

$t_{S\overline{\Phi}(D)}$ = Data Setup time to falling edge of clock

let: $t_C = 400\text{ns}$; $t_{DL\ (MR)} = 100\text{ns}$; $t_{S\ (D)\,\overline{\Phi}} = 60\text{ns}$

then: $\underline{t_{\text{ACCESS MEMORY READ}} = 640\text{ns}}$

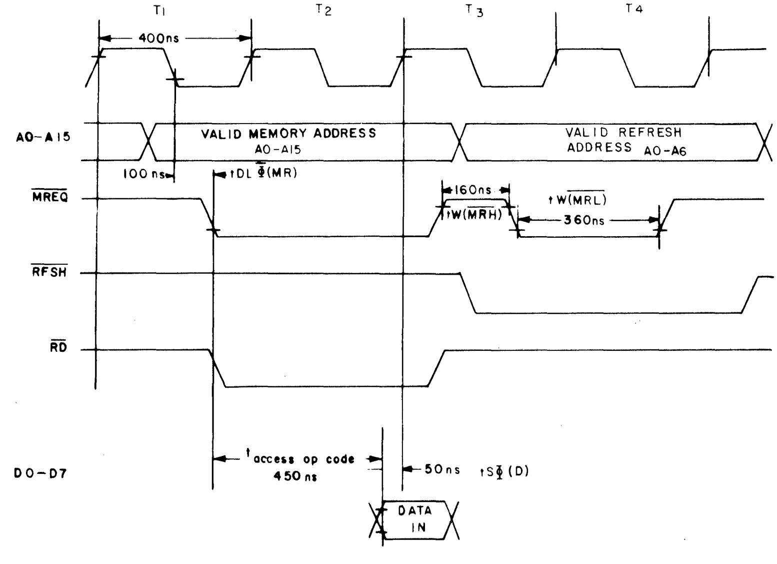
The access times computed in equations 1 and 2 are overall worst case access times required by the CPU. The overall access times must include all TTL buffer delays and the access time for the memory device. For example, a typical dynamic memory design would have the following characteristics, (see Figure 2).

The example in Figure 2 shows an overall access time of 336ns. This would more than satisfy the 450ns required for the op code fetch and the 640ns required for a memory read.

CPU $\overline{\text{MREQ}}$ buffer delay.	12ns	(8T97)
Memory gating and timing delays	40ns	
Memory device access time	250ns	(MK4027/4116-4)
Memory data bus buffer delay	17ns	(8T28)
CPU data bus buffer delay.	17ns	(8T28)
	336ns	

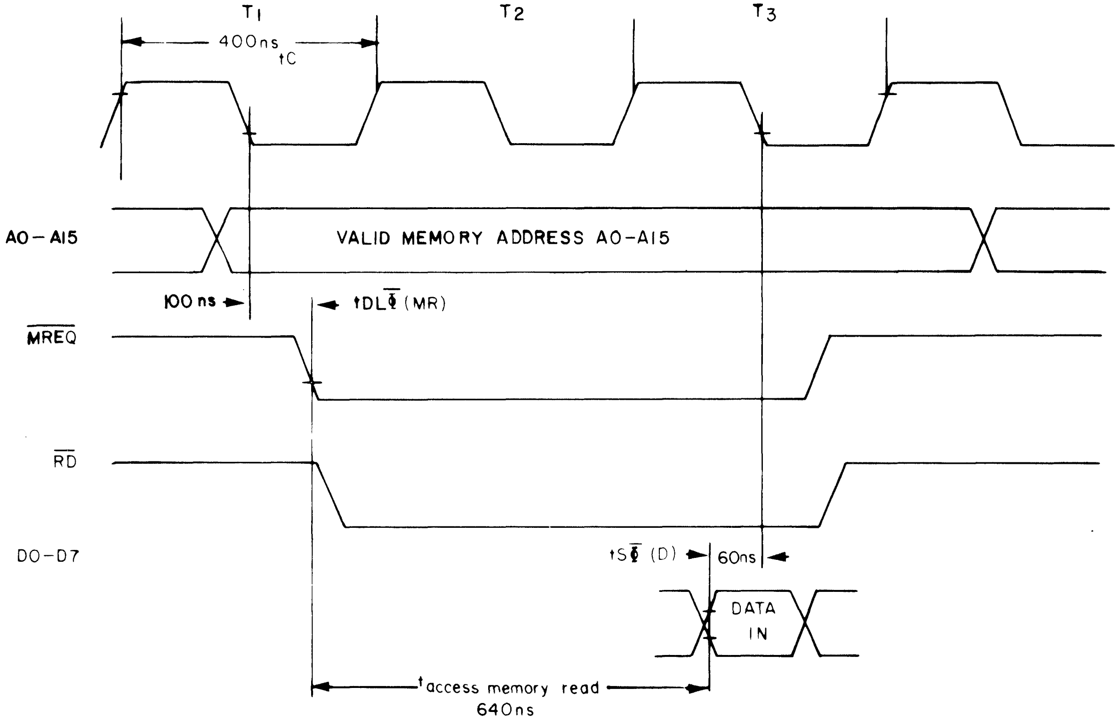
OP CODE FETCH TIMING

Figure 1a.



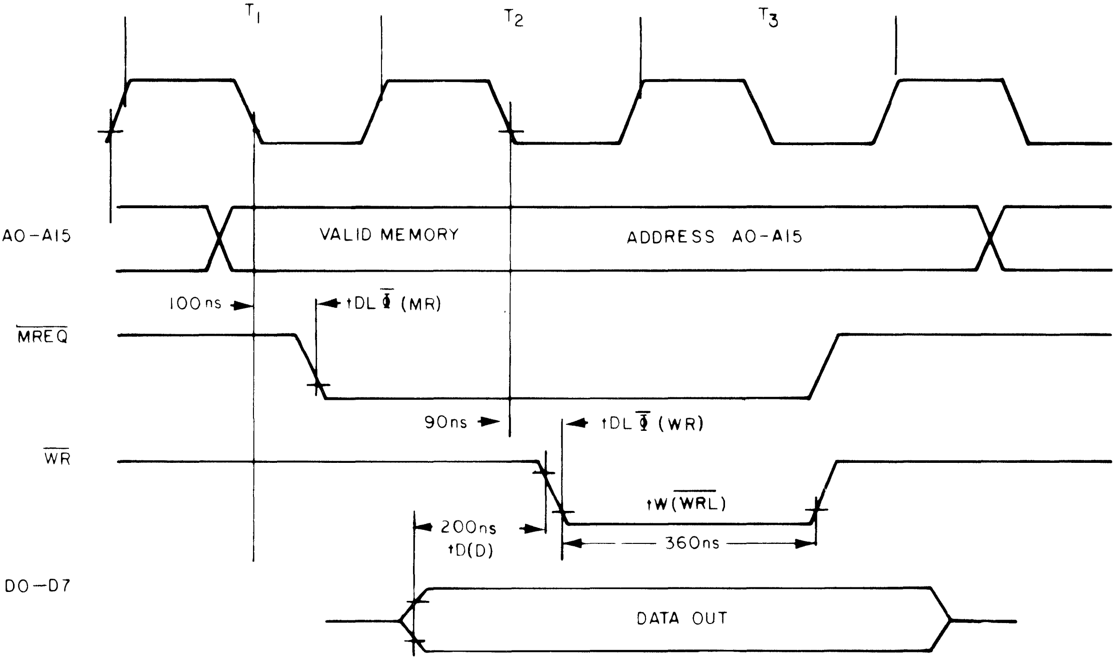
MEMORY READ TIMING

Figure 1b.



MEMORY WRITE TIMING

Figure 1c.



Z80 REFRESH CONTROL AND TIMING

One of the most important features provided by the Z80 for interfacing to dynamic memories is the execution of a refresh cycle every time an op code fetch cycle is performed. By placing the refresh cycle in the op code fetch, the Z80 does not have to allocate time in the form of "wait states" or by "stretching" the clock to perform the refresh cycle. In other words, the refresh cycle is "totally transparent" to the CPU and does not decrease the system throughput (see Figure 1a). The refresh cycle is transparent to the CPU because, once the op code has been fetched from memory during states T_1 and T_2, the memory would normally be idle during states T_3 and T_4.

Therefore, by placing the refresh in the T_3 and T_4 states of the op code fetch, no time is lost for refreshing dynamic memory. The critical timing parameters involving the Z80 and dynamic memories during the refresh cycle are: $t_{W(MRH)}$ and $t_{W(MRL)}$. The parameter known as $t_{W(MRH)}$ refers to the time that $\overline{\text{MREQ}}$ is high during the op code fetch between the fetch of the op code and the refresh cycle. This time is known as "precharge" for dynamic memories and is necessary to allow certain internal nodes of the RAM to be charged-up for another memory cycle. The equation for the minimum $t_{W(MRH)}$ time period is:

(3) $t_{W(MRH)} = t_{W(\Phi H)} + t_f - 30$

where: $t_{W(\Phi H)}$ is clock pulse width high
t_f is clock fall time

let: $t_{W(\Phi H)} = 180\text{ns}$; $t_f = 10\text{ns}$

then: $\underline{t_{W(MRH)} = 160\text{ns (min)}}$

A $t_{W(MRH)}$ of 160ns is more than adequate to meet the worst case precharge times for most dynamic RAMs. For example, the MK4027-4 and the MK4116-4 require a 120ns precharge.The other refresh cycle parameter of importance to dynamic RAMs is $t_{W(MRL)}$, (the time that $\overline{\text{MREQ}}$ is low during the refresh cycle). This time is important because $\overline{\text{MREQ}}$ is used to directly generate $\overline{\text{RAS}}$. The equation for the minimum time period is:

(4) $t_{W(MRL)} = t_c - 40$

where: t_c is the clock period

let: $t_c = 400\text{ns}$

then: $\underline{t_{W(MRL)} = 360\text{ns}}$

A 360ns $t_{W(MRL)}$ exceeds the 250ns min $\overline{\text{RAS}}$ time required for the MK4027-4 and the MK4116-4.

By controlling the refresh internally with the Z80, the designer must be aware of one limitation. The limitation is that to refresh memory properly, the Z80 CPU must be able to execute op codes since the refresh cycle occurs during the op code fetch. The following conditions cause the execution of op codes to be inhibited, and will destroy the contents of dynamic memory.

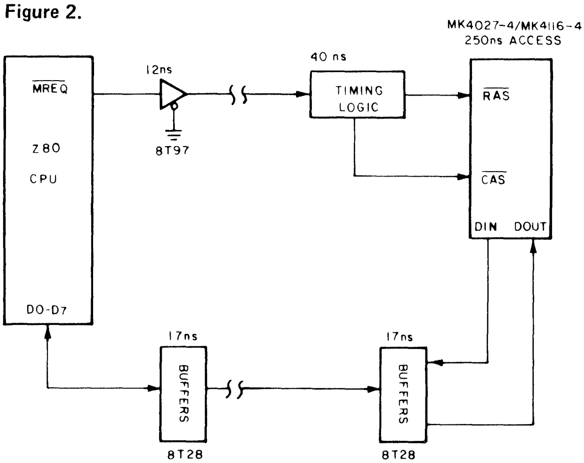
(1) Prolonged reset > 1ms
(2) Prolonged wait state operation > 1ms
(3) Prolonged bus acknowledge (DMA) > 1ms
(4) Φ clock of < 1.216 MHz for 16K RAMs
< .608 MHz for 4K RAMs

The clocks rate in number 4 are based on the Z80 continually executing the worst case instruction which is an EX (SP), HL that executes in 19 T states. Therefore, by operating the Z80 at or above these clocks frequencies, the user is ensured that the dynamic memories in the system will be refreshed properly.

Remember to refresh memory properly, the Z80 must be able to execute op codes!

DELAY FOR A TYPICAL MEMORY SYSTEM

Figure 2.



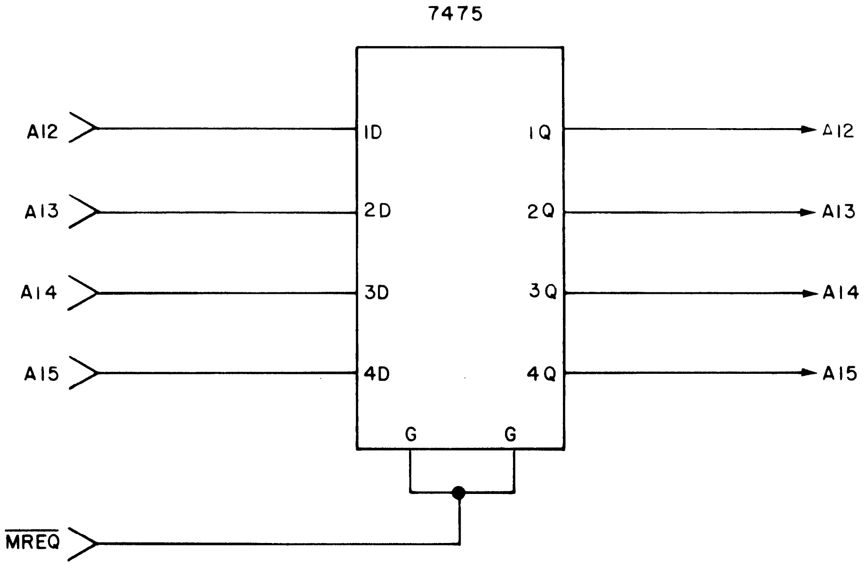
SUPPORT CIRCUITS FOR DYNAMIC MEMORY INTERFACE

Two support circuits are necessary to ensure reliable operation of dynamic memory with the Z80.

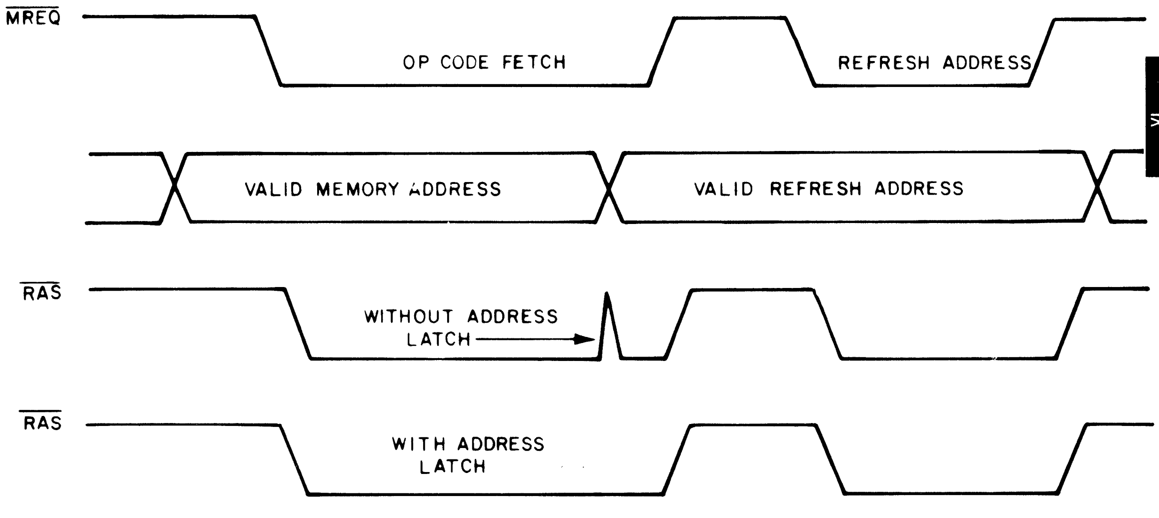
The first of these circuits is an address latch shown in Figure 3. The latch is used to hold addresses A_{12}-A_{15} while $\overline{\text{MREQ}}$ is active. This action is necessary because the Z80 does not ensure the validity of the address bus at the end of the op code fetch (see Figure 4). This action does not directly affect dynamic memories because they latch addresses internally. The problem comes from the address decoder which generates $\overline{\text{RAS}}$. If the address lines which drive the decoder are allowed to change while $\overline{\text{MREQ}}$ is low, then a "glitch" can occur on the $\overline{\text{RAS}}$ line or lines (if more than one row of RAMs are used) which may have the effect of destroying one row of data.

The second support circuit is used to generate a power on and short manual reset pulse. Recall from the discussion under Z80 Timing and Memory Con-

ADDRESS LATCH
Figure 3.



RAS TIMING WITH AND WITHOUT ADDRESS LATCH
Figure 4.

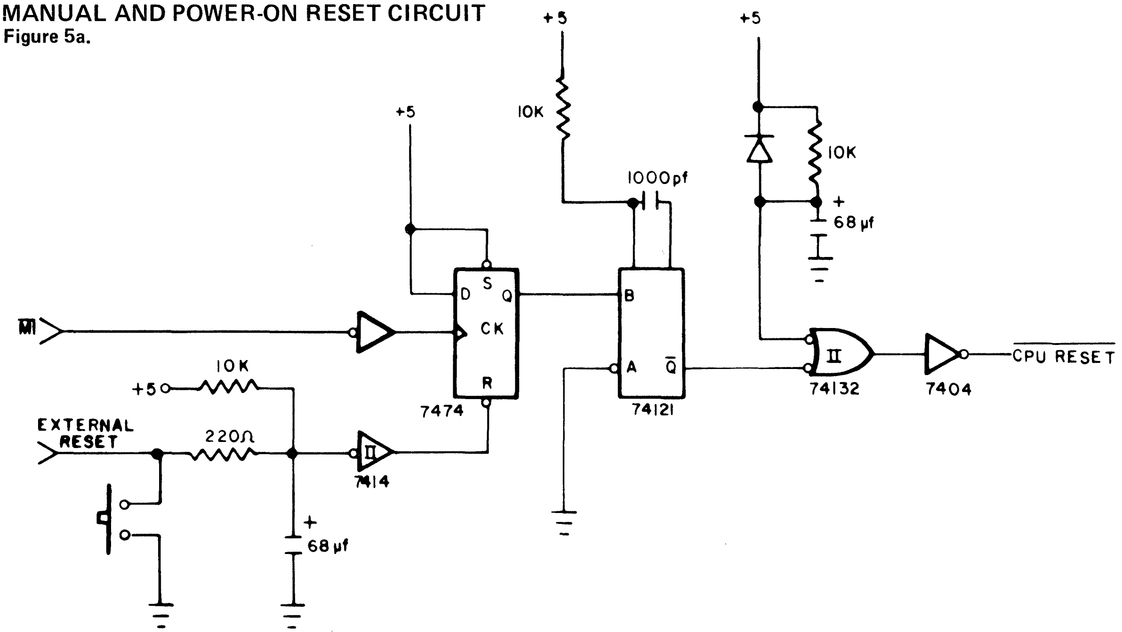


trol Signals that one of the conditions that will cause dynamic memory to be destroyed is a reset pulse of duration greater than 1ms. The circuit shown in Figure 5a can be used to generate a short reset pulse from either a push button or an external source. Additionally the manual reset is synchronized to the start of an M1 cycle so that the reset will not fall during the middle of a memory cycle. Along with the manual reset, the circuit will also generate a power on reset.

If it is not necessary that the contents of the dynamic memory be preserved, then the reset circuit shown in Figure 5b may be used to generate a manual or power on reset.

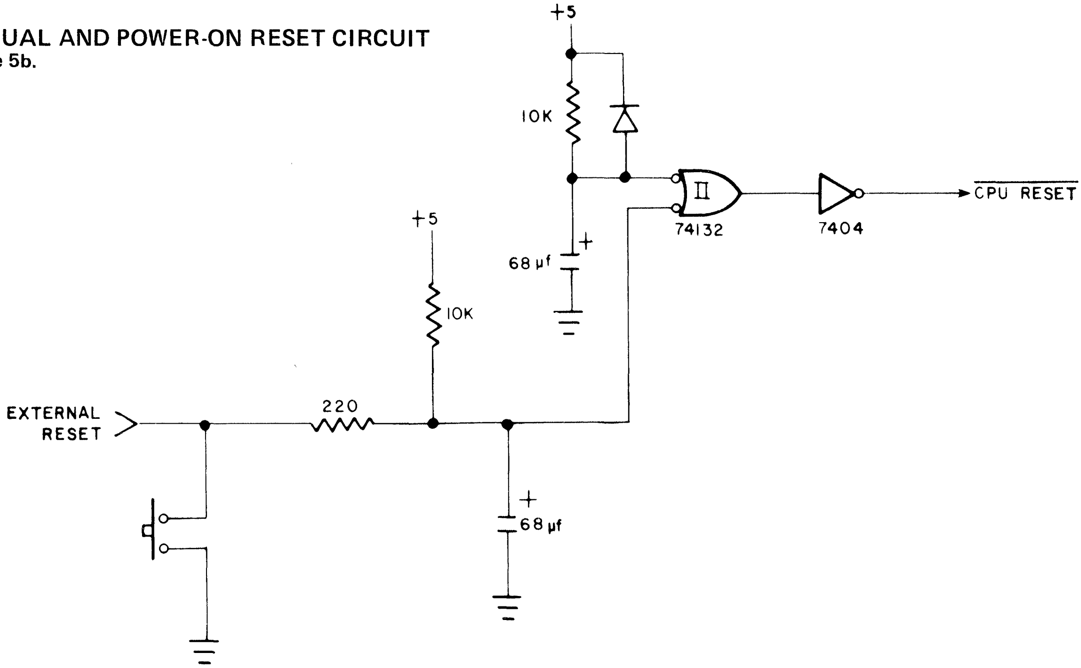
MANUAL AND POWER-ON RESET CIRCUIT

Figure 5a.



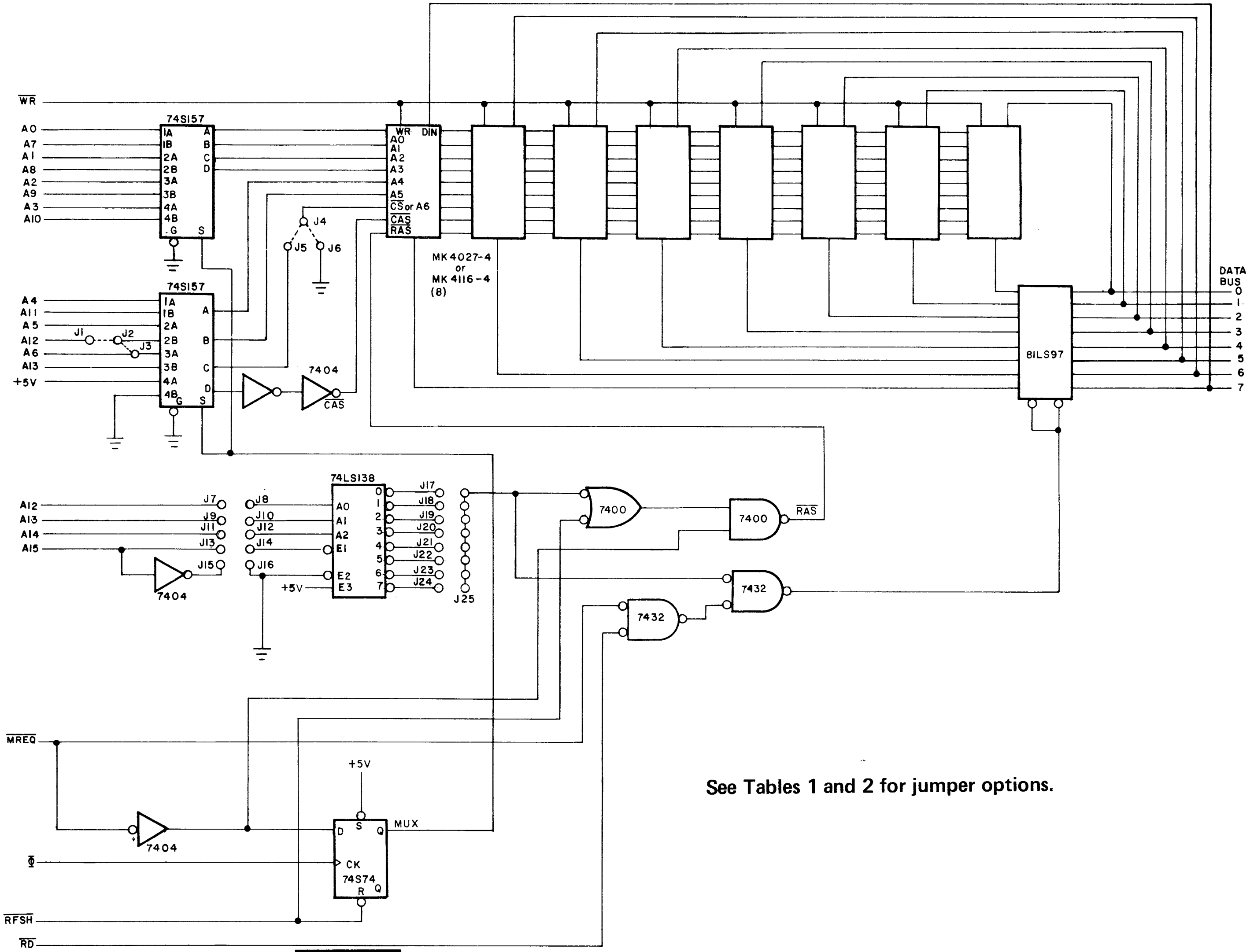
MANUAL AND POWER-ON RESET CIRCUIT

Figure 5b.



DESIGN EXAMPLE NO. 1 SCHEMATIC DIAGRAM

Figure 6.



See Tables 1 and 2 for jumper options.

DESIGN EXAMPLES FOR INTERFACING THE Z80 TO DYNAMIC MEMORY

To illustrate the interface between the Z80 and dynamic memory, two design examples are presented. Example number 1 is for a 4K/16Kx8 memory and the example number 2 is a 16K/64Kx8 memory.

Design Example Number 1: 4K/16Kx8 Memory

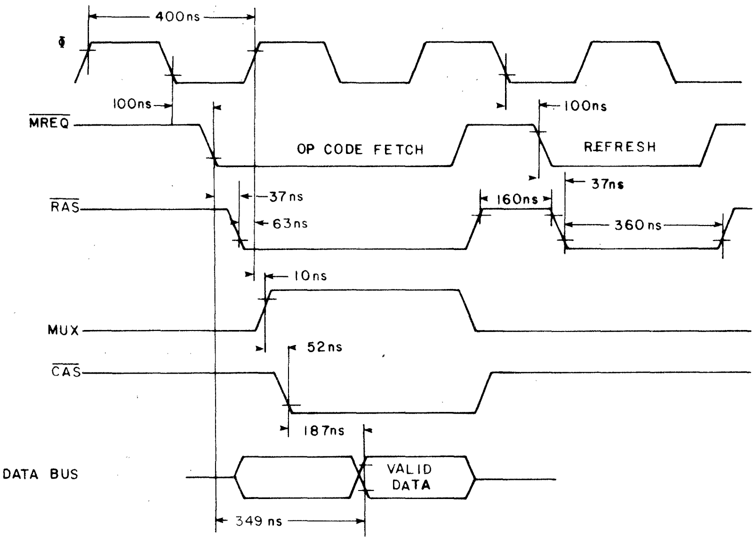
This design example describes a 4K/16Kx8 memory that is best suited for a small single board Z80 based microcomputer system. The memory devices used in the example are the MK4027 (4,096x1 MOS Dynamic RAM) and the MK4116 (16,384x1 MOS Dynamic RAM). A very important feature of this design is the ease in which the memory can be expanded from a 4Kx8 to a 16Kx8 memory. This is made possible by the use of jumper options which configure the memory for either the MK4027 or the MK4116. See Table 1 and 2 for jumper options.

Figure 6 shows the schematic diagram for the 4K/16Kx8 memory. A timing diagram for the Z80 control signals and memory control signals is shown in Figure 7. The operation of the circuit may be described as follows: $\overline{\text{RAS}}$ is generated by NANDing MREQ with RFSH + ADDRESS DECODE. $\overline{\text{RFSH}}$ is generated directly from the Z80 while address decode comes from the 74LSI38 decoder. Address decode indicates that the address on the bus falls within the memory boundaries of the memory. If an op code fetch or memory read is being executed the 81LS97 output buffer will be enabled at approximately the same time as $\overline{\text{RAS}}$ is generated for the memory array. The output buffer is enabled only during an op code fetch or memory read when $\overline{\text{ADDRESS DECODE}}$, $\overline{\text{MREQ}}$, and $\overline{\text{RD}}$ are all low. The switch multiplexer signal (MUX) is generated on the rising edge of Φ after $\overline{\text{MREQ}}$ has gone low during an op code fetch, memory read or memory write. After MUX is generated and the address multiplexers switch from the row address to column address, $\overline{\text{CAS}}$ will be generated. $\overline{\text{CAS}}$ comes from one of the outputs of the multiplexer and is delayed by two gate delays to ensure that the proper column address set-up time will be achieved. Once $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ have been generated for the memory array, the memory will then access the desired location for a read or write operation.

7404	22ns	Generate $\overline{\text{RAS}}$ from $\overline{\text{MREQ}}$
7400	15ns	
	63ns	$\overline{\text{RAS}}$ to rising edge of Φ
74S74	10ns	Φ to MUX
74S157	15ns	Generate $\overline{\text{CAS}}$ from MUX
7404	22ns	
7404	15ns	
t_{CAC}	165ns	$\overline{\text{CAS}}$ access time
81LS97	22ns	Output buffer delay
	349ns	Worst case access

DESIGN EXAMPLE NO. 1 MEMORY TIMING

Figure 7.



The worst case access time required by the CPU for the op code fetch is 450ns (from equation 1); therefore, the circuit exceeds the required access time by 101ns (worst case).

The circuit shown in Figure 6 provides excellent performance when used as a small on board memory. The memory size should be held at eight devices because there is not sufficient timing margin to allow the interface circuit to drive a larger memory array.

Design Example Number 2: 16Kx8 Memory

This design example describes a 16K/64Kx8 memory which is best suited for a Z80 based microcomputer system where a large amount of RAM is desired. The memory devices used in this example are the same as for the first example, the MK4027 and the MK4116. Again as with the first example, the memory may be expanded from a 16Kx8 to a 64Kx8 by reconfiguring jumpers. See Table 3 and 4 for jumper options.

Figure 8 shows the schematic diagram for the 16K/64K memory. A timing diagram is shown in Figure 9. The operation of the circuit can be described as follows: $\overline{\text{RAS}}$ is generated by NANDing MREQ with ADDRESS DECODE (from the two 74LS138s) + RFSH. Only one row of RAMs will receive a $\overline{\text{RAS}}$ during an op code fetch, memory read or memory write. However, a $\overline{\text{RAS}}$ will be generated for all rows within the array during a refresh cycle. $\overline{\text{MREQ}}$ is inverted and fed into a TTL compatible delay line to generate MUX and $\overline{\text{CAS}}$. (This particular approach differs from the method used in example number 1 in that all memory timing is referenced to MREQ, whereas the circuit in example number 1 bases its memory timing from both $\overline{\text{MREQ}}$ and the clock. Both methods offer good results, however, the TTL delay line approach offers the best control over the memory timing.) MUX is generated 65ns later and is used to switch the 74157 multiplexers from the row to the column address. The 65ns delay was chosen to allow adequate margin for the row address hold time t_{RAH}. At 110ns, $\overline{\text{CAS}}$ is generated from the delay line and NANDed with RFSH, which inhibits a $\overline{\text{CAS}}$ during refresh cycle. After $\overline{\text{CAS}}$ is applied to the memory, the desired location is then accessed. A worst case access timing analysis for the circuit shown in Figure 8 can be computed as follows:

74LS14	22ns	Generate $\overline{\text{RAS}}$ from $\overline{\text{MREQ}}$
74LS00	15ns	
delay line	50ns	MUX from $\overline{\text{RAS}}$
delay line	45ns	$\overline{\text{CAS}}$ delay from MUX
7400	20ns	
t_{CAC}	165ns	Access time from $\overline{\text{CAS}}$
8833	30ns	Output buffer delay
	347ns	

The required access time from the CPU is 450ns (from equation 1). This leaves 103ns of margin for additional CPU buffers on the control and address lines. This particular circuit offers excellent results for an application which requires a large amount of RAM memory. As mentioned earlier, the memory timing used in this example offers the best control over the memory timing and would be ideally suited for an application which required direct memory access (DMA).

4K x 8 CONFIGURATION(MK4027) JUMPER

Table 1

CONNECT: J13 to J14

ADDRESS	CONNECT
0000-0FFF	J17 to J25
1000-1FFF	J18 to J25
2000-2FFF	J19 to J25
3000-3FFF	J20 to J25
4000-4FFF	J21 to J25
5000-5FFF	J22 to J25
6000-6FFF	J23 to J25
7000-7FFF	J24 to J25

Connect: J2 to J3
J4 to J6
J7 to J8
J9 to J10
J11 to J12

CONNECT: J14 to J15

ADDRESS	CONNECT
8000-8FFF	J17 to J25
9000-9FFF	J18 to J25
A000-AFFF	J19 to J25
B000-BFFF	J20 to J25
C000-CFFF	J21 to J25
D000-DFFF	J22 to J25
E000-EFFF	J23 to J25
F000-FFFF	J24 to J25

16K x 8 CONFIGURATION (MK4116) JUMPER CONNECTIONS

Table 2

CONNECT: J1 to J2
J4 to J5
J8 to J11
J10 to J13
J12 to J16
J14 to J16

ADDRESS	CONNECT
0-3FFF	J17 to J25
4000-7FFF	J18 to J25
8000-BFFF	J19 to J25
C000-FFFF	J20 to J25

16K x 8 CONFIGURATION (MK4027)

Table 3

CONNECT: J1 to J3
J5 to J6
J7 to J8
J9 to J10
J11 to J12
J13 to J14

ADDRESS:	0-3FFF	ADDRESS:	4000-7FFF	ADDRESS:	8000-BFFF	ADDRESS:	C000-FFFF
CONNECT:	J24 to J25	CONNECT:	J16 to J17	CONNECT:	J40 to J41	CONNECT:	J32 to J33
	J26 to J27		J18 to J19		J42 to J43		J34 to J35
	J28 to J29		J20 to J21		J44 to J43		J36 to J37
	J30 to J31		J22 to J23		J46 to J47		J38 to J39

64K x 8 CONFIGURATION(MK4116)

Table 4

CONNECT: J1 to J2
J4 to J5
J8 to J11
J10 to J13
J12 to J15
J14 to J15

ADDRESS:	0-FFFF
CONNECT:	J32 to J33
	J34 to J35
	J36 to J37
	J38 to J39

SYSTEM PERFORMANCE CHARACTERISTICS

Table 5

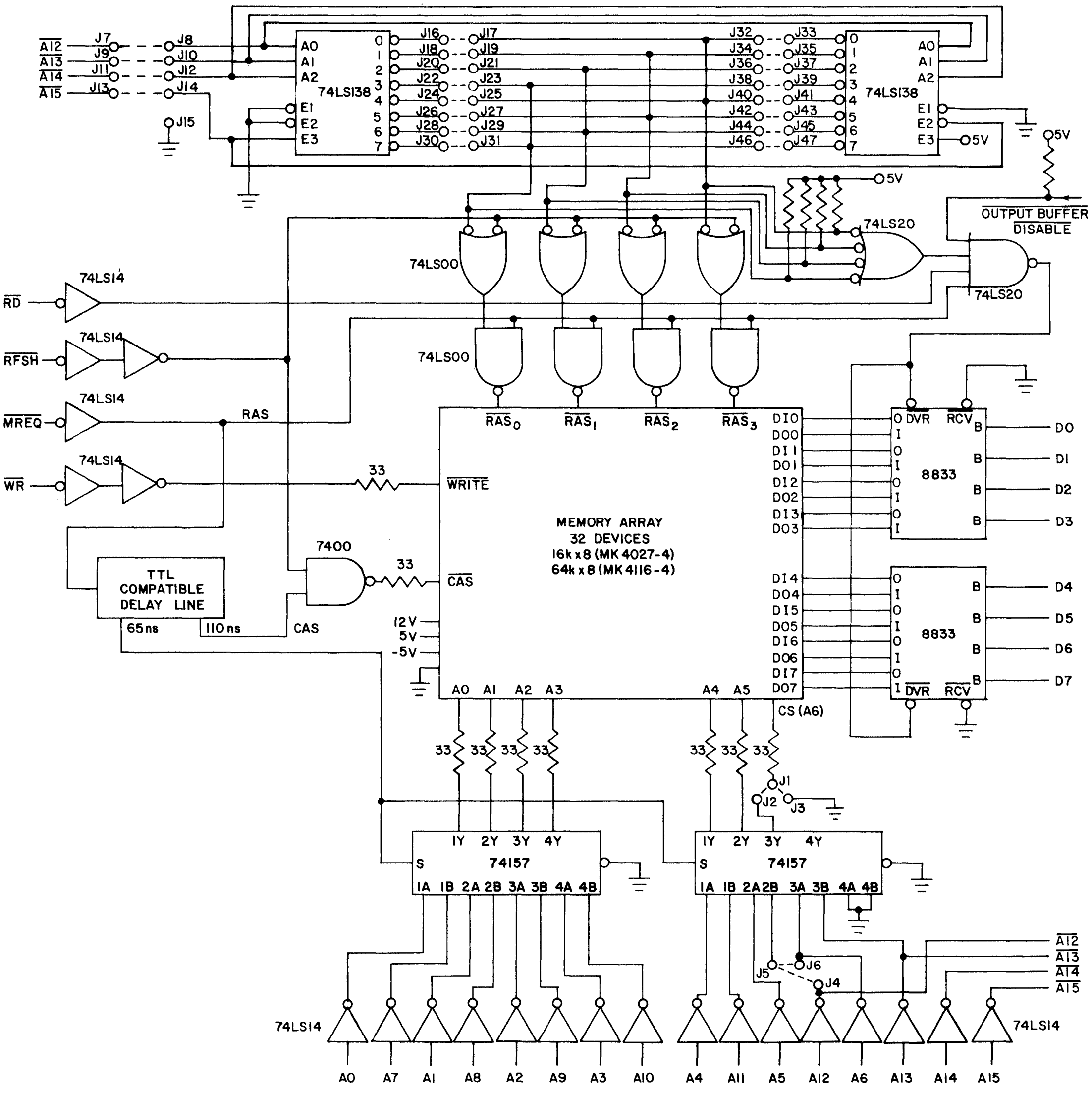
The system characteristics for the preceeding design examples are shown in Table 5.

EXAMPLE #	MEMORY CAPACITY	MEMORY ACCESS	POWER REQUIREMENTS
1	4K/16Kx8	349ns max.	+12V @ 0.0250 A max. +5V @ 0.422 A max.* -5V @ 0.030 A max.
2	16K/64Kx8	347ns max.	+12V @ 0.600 A max. +5V @ 0.550 A max. * -5V @ 0.030 A max.

*All power requirements are max.; operating temperature 0°C to 70°C ambient, max +12V current computed with Z80 executing continuous op code fetch cycles from RAM at 1.6 μs intervals.

DESIGN EXAMPLE NO. 2 SCHEMATIC DIAGRAM

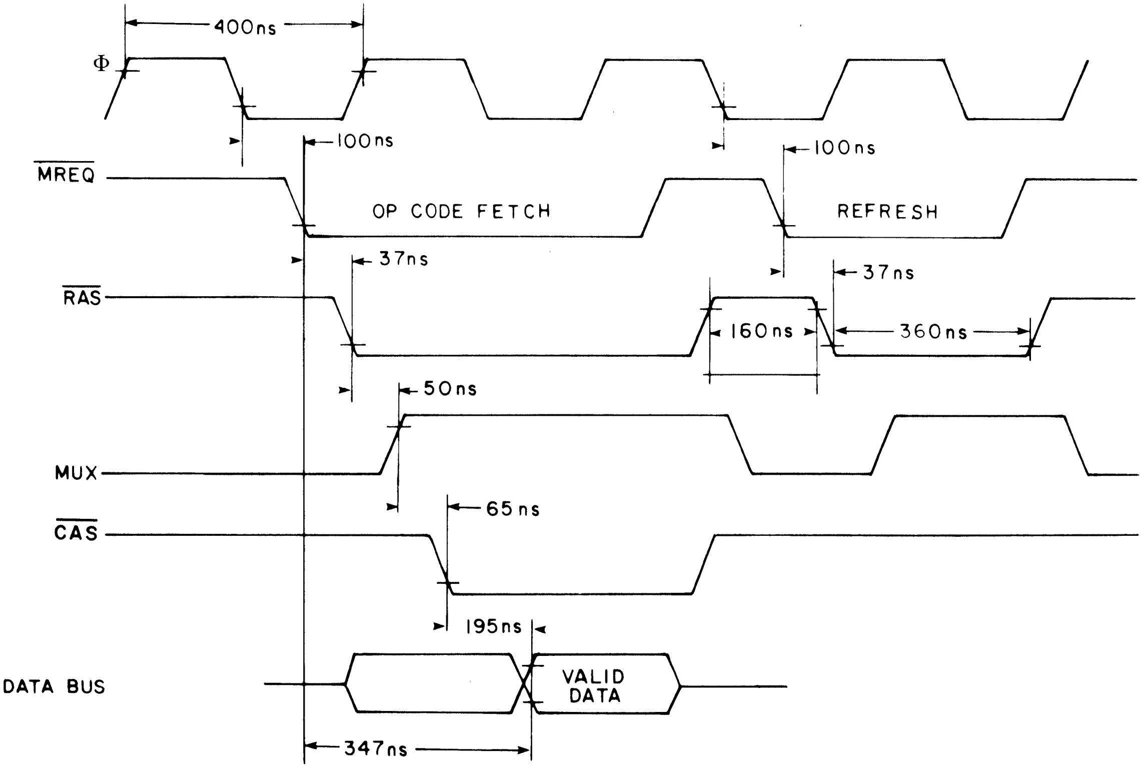
Figure 8.



FOR JUMPER OPTIONS SEE TABLES 3 AND 4

DESIGN EXAMPLE NO. 2 MEMORY TIMING

Figure 9.



PRINTED CIRCUIT LAYOUT

One of the most important parts of a dynamic memory design is the printed circuit layout. Figure 10 illustrates a recommended layout for 32 devices. A very important factor in the P.C. layout is the power distribution. Proper power distribution on the V_{DD} and V_{BB} supply lines is necessary because of the transient current characteristics which dynamic memories exhibit. To achieve proper power distribution, V_{DD}, V_{BB}, V_{CC} and ground should be laid out in a grid to help minimize the power distribution impedance. Along with good power distribution, adequate capacitive bypassing for each device in the memory array is necessary. In addition to the individual by-passing capacitors, it is recommended that each supply (V_{BB}, V_{CC} and V_{DD}) be bypassed with an electrolytic capacitor 20μF.

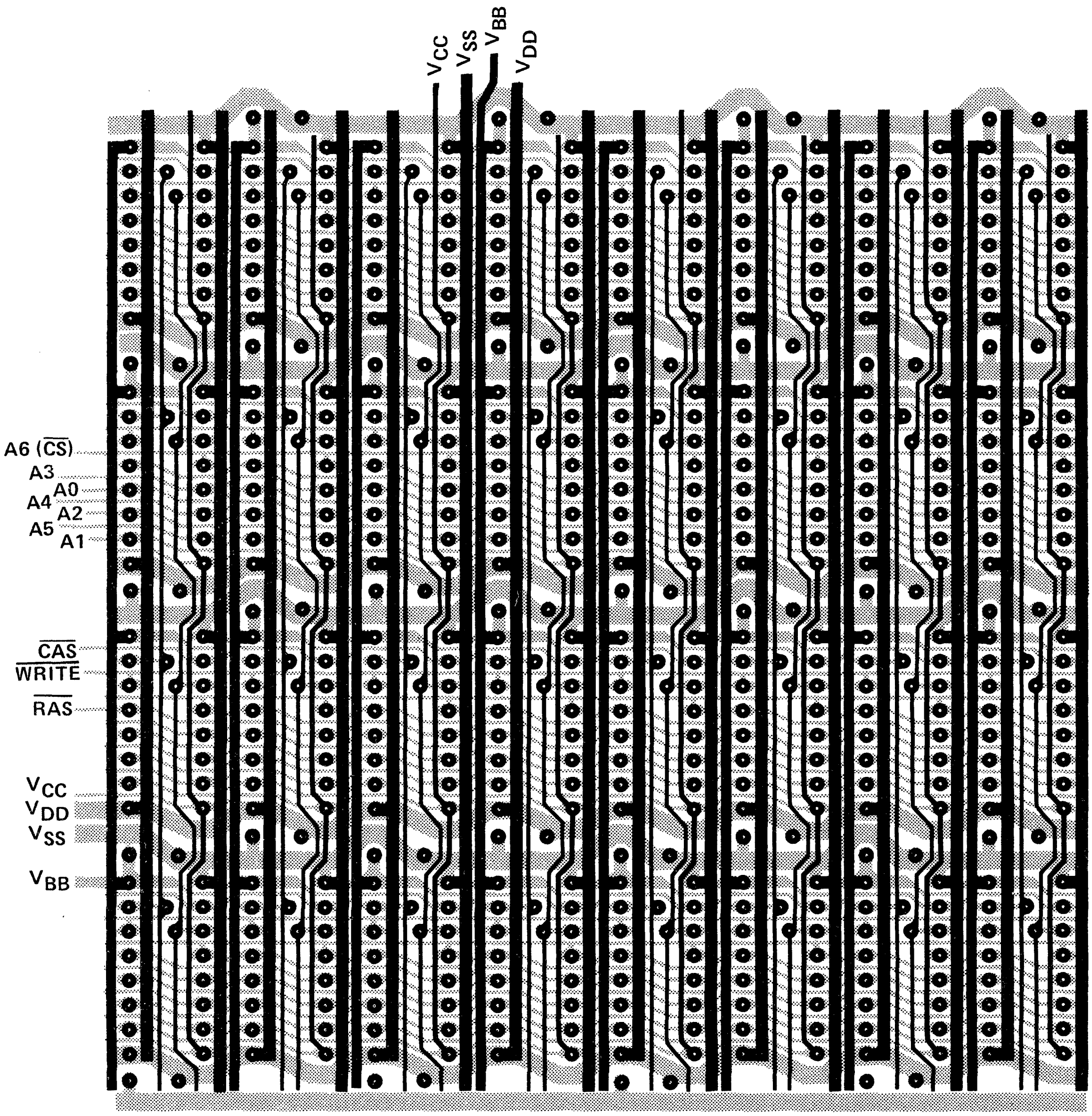
By using good power distribution techniques and using the recommended number of bypassing capacitors, the designer can minimize the amount of noise in the memory array. Other layout considerations are the placement of signal lines. Lines such as address, chip select, column address strobe, and write should be bussed together as rows; then, bus all rows together at one end of the array. Interconnection between rows should be avoided. Row address strobe lines should be bussed together as a row, then connected to the appropriate $\overline{\text{RAS}}$ driver. TTL drivers for the memory array signals should be located as close as possible to the array to help minimize signal noise.

For a large memory array such as the one shown in design example number 2, series terminating resistors should be used to minimize the amount of negative undershoot. These resistors should be used on the address lines, $\overline{\text{CAS}}$ and $\overline{\text{WRITE}}$, and have values between 20 Ω to a 33 Ω .

The layout for a 32 device array can be put in a 5" x 5" area on a two sided printed circuit board.

SUGGESTED P. C. LAYOUT FOR MK4027 or MK4116

Figure 10.



4MHz Z80 DYNAMIC MEMORY INTERFACE CONSIDERATIONS

A 4MHz Z80 is available for the microcomputer designer who needs higher system throughput. Considerations which must be faced by the designer when interfacing the 4MHz Z80 to dynamic memory are the need for memories with faster access times and for providing minimum RAM precharge time. The access times required for dynamic memory interfaced to a 4MHz Z80 can be computed from equations 1 and 2 under Z80 Timing and Memory Control Signals.

Access time for op code fetch for 4MHz Z80,
let: t_C = 250ns; $t_{DL\overline{\Phi}(MR)}$ = 75ns; $t_{s\Phi(D)}$ = 35ns
then: $t_{ACCESS\ OP\ CODE}$ = 265ns

Access time for memory read for 4MHz Z80,
let: t_C = 250ns; $t_{DL\overline{\Phi}(MR)}$ = 75ns; $t_{S\overline{\Phi}(D)}$ = 50ns
then: $t_{ACCESS\ MEMORY\ READ}$ = 375ns

The problem of faster access times can be solved by using 200ns memories such as the MK4027-3 or MK4116-3. Depending on the number of buffer delays in the system, the designer may have to use 150ns memories such as the MK4027-2 or MK4116-2. The most critical problem that exists when interfacing dynamic memory to the 4MHz Z80 is the RAM precharge time (trp). This parameter is called $t_{W(MRH)}$ on the Z80 and can be computed by the following equation.

(4) $t_{W(RH)} = t_{W(\Phi H)} + t_f$-20ns
let: $t_{W(\Phi H)}$ = 110ns; t_f = 5ns
then: $t_{W(MRH)}$ = 95ns

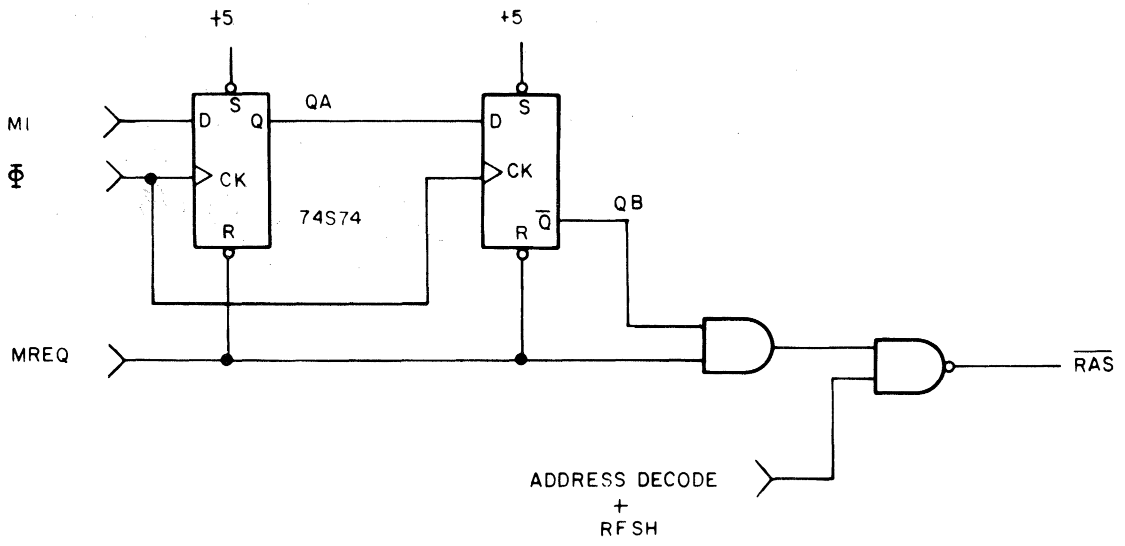
A $t_{W(MRH)}$ of 95ns will not meet the minimum precharge time of the MK4027-2 or MK4116-2 which is 100ns. The MK4027-3 and MK4116-3 require a 120ns precharge. Figure 11 shows a circuit that will lengthen the $t_{W(MRH)}$ pulse from 95ns to a minimum of 126ns while only inserting one gate delay into the access timing chain. Figure 12 shows the timing for the circuit of Figure 11. The operation of the circuit in Figure 11 can be explained as follows: The D flip flops are held in a reset condition until MREQ goes to its active state. After MREQ goes active, on the next positive clock edge, the D input of U1 and U2 will be transferred to the outputs of the flip flops. Output QA will go high if M1 was high when Φ clocked U1. Output QB will go low on the next positive going clock edge, which will cause the output of U3 to go low and force the output of U4, which is $\overline{RAS}$, high. The flip flops will be reset when MREQ goes inactive.

The circuit shown in Figure 11 will give a minimum of 126ns precharge for dynamic memories, with the Z80 operating at 4MHz. The 126ns $t_{W(MRH)}$ is computed as follows.

110ns	$t_{W(\Phi H)}$ - clock pulse width high (min)
5ns	t_F - clock full time (min)
20ns	$t_{DL\overline{\Phi}(MR)}$ - MREQ delay (min)
-9ns	74S74 delay (min)
126ns	$t_{W(MRH)}$ modified (min)

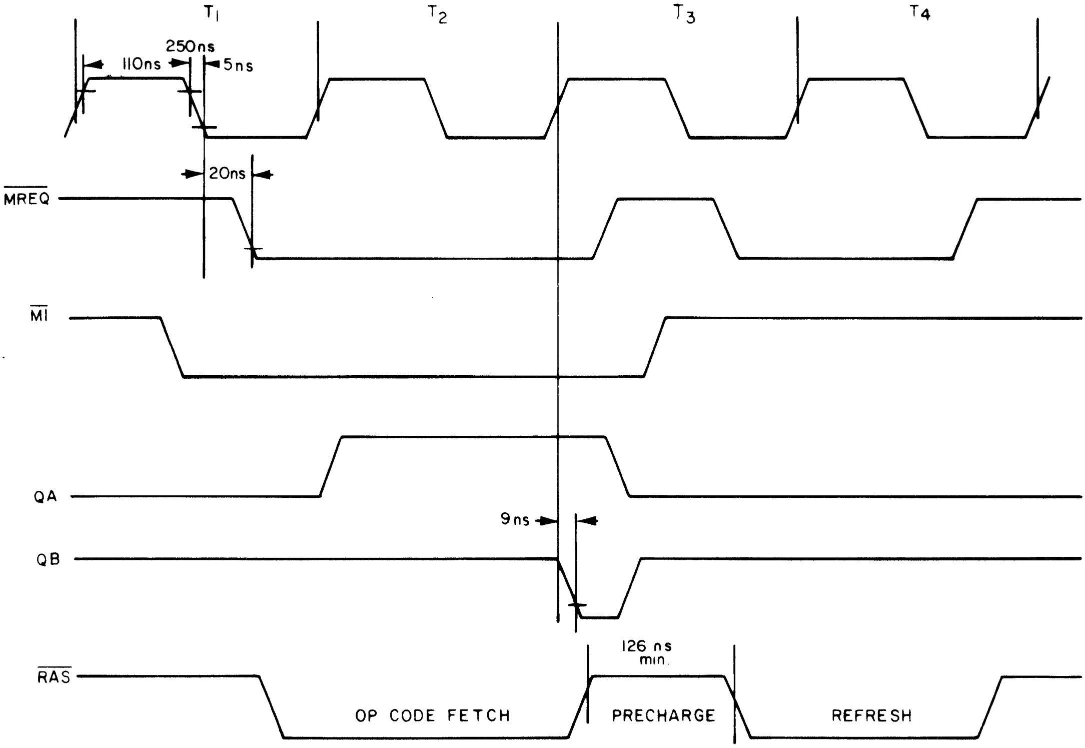
4MHz Z80 PRECHARGE EXTENDER FOR DYNAMIC MEMORIES

Figure 11



TIMING DIAGRAM FOR 4MHz Z80 PRECHARGE EXTENDER

Figure 12



APPENDIX

MEMORY TEST ROUTINE

This section is intended to give the microcomputer designer a memory diagnostic suitable for testing memory systems such as the ones shown in Section VI.

The routine is a modified address storage test with an incrementing pattern. A complete test requires 256$_{10}$ passes, which will execute in less than 4 minutes for a 16Kx8 memory. If an error occurs, the program will store the pattern in location '2C'H and the address of the error at locations '2D'H and '2E'H.

The program is set up to test memory starting at location '2F'H up to the end of the block of memory defined by the bytes located at 'OC'H and 'OD'H. The test may be set up to start at any location by modifying locations '03'H - '04'H and '11'H - '12'H with the starting address that is desired.

```
                          MXRTS  LISTING                  PAGE   0001
    LOC    OBJ CODE    STMT SOURCE STATEMENT

                       0001 ;TRANSLATED FROM DEC 1976 INTERFACE MAGAZINE
                       0002 ;
                       0003 ;THIS IS A MODIFIED ADDRESS STORAGE TEST WITH AN
                       0004 ;INCREMENTING PATTERN
                       0005 ;
                       0006 ;256 PASSES MUST BE EXECUTED BEFORE THE MEMORY IS
                       0007 ;COMPLETELY TESTED.
                       0008 ;
                       0009 ;IF AN ERROR OCCURS, THE PATTERN WILL BE STORED
                       0010 ;AT LOCATION '002C'H AND THE ADDRESS OF THE
                       0011 ;ERROR LOCATION WILL BE STORED AT '002D'H AND
                       0012 ;'002E'H.
                       0013 ;
```

MEMORY TEST ROUTINE (Cont'd.)

```
                     0014 ;THE CONTENTS OF LOCATIONS '000C'H AND '001D'H
                     0015 ;SHOULD BE SELECTED ACCORDING TO THE FOLLOWING
                     0016 ;MEMORY SIZE TO BE TESTED
                     0017 ;
                     0018 ;TOP OF MEMORY TO
                     0019 ;BE TESTED                     VALUE OF EPAGE
                     0020 ;
                     0021 ;      4K                                '10'H
                     0022 ;      8K                                '20'H
                     0023 ;     16K                                '40'H
                     0024 ;     32K                                '80'H
                     0025 ;     48K                                'C0'H
                     0026 ;     64K                                'FF'H
                     0027 ;
                     0028 ;THE PROGRAM IS SET UP TO START TESTING AT
                     0029 ;LOCATION '002F'H. THE STARTING ADDRESS FOR THE
                     0030 ;TEST CAN BE MODIFIED BY CHANGING LOCATIONS
                     0031 ;'0003-0004'H AND '0011-0012'H.
                     0032 ;
                     0033 ;TEST TIME FOR A 16K X 8 MEMORY IS APPROX. 4 MIN
                     0034 ;
0000                 0035        ORG   0000H
0000    0600         0036        LD    B,0        ;CLEAR B PATRN MODIFIER
                     0037 ;LOAD UP MEMORY
0002    212F00       0038 LOOP:  LD    HL,START   ;GET STARTING ADDR
0005    7D           0039 FILL:  LD    A,L        ;LOW BYTE TO ACCM
0006    AC           0040        XOR   H          ;XOR WITH HIGH BYTE
0007    A8           0041        XOR   B          ;XOR WITH PATTERN
0008    77           0042        LD    (HL),A     ;STORE IN ADDR
0009    23           0043        INC   HL         ;INCREMENT ADDR
000A    7C           0044        LD    A,H        ;LOAD HIGH BYTE OF ADDR
000B    FE10         0045        CP    EPAGE      ;COMPARE WITH STOP ADDR
000D    C20500       0046        JP    NZ,FILL    ;NOT DONE,GO BACK
                     0047 ;READ AND CHECK TEST DATE
0010    212F00       0048        LD    HL,START   ;GET STARTING ADDR
0013    7D           0049 TEST:  LD    A,L        ;LOAD LOW BYTE
0014    AC           0050        XOR   H          ;XOR WITH HIGH BYTE
0015    A8           0051        XOR   B          ;XOR WITH MODIFIER
0016    BE           0052        CP    (HL)       ;COMPARE WITH MEMORY LOC
0017    C22500       0053        JP    NZ,FXIT    ;ERROR EXIT
001A    23           0054        INC   HL         ;UPDATE MEMORY ADDRESS
001B    7C           0055        LD    A,H        ;LOAD HIGH BYTE
001C    FE10         0056        CP    EPAGE      ;COMPARE WITH STOP ADDR
001E    C21300       0057        JP    NZ,TEST    ;LOOP BACK
0021    04           0058        INC   B          ;UPDATE MODIFIER



                                  MXRTS  LISTING                 PAGE   0002
 LOC    OBJ CODE     STMT SOURCE STATEMENT

0022    C30200       0059        JP    LOOP       ;RST WITH NEW MODIFIER
                     0060 ;ERROR EXIT
0025    222D00       0061 FXIT:  LD    (BYTE),HL  ;SAVE ERROR ADDRESS
0028    322C00       0062        LD    (PATRN),A  ;SAVE BAD PATTERN
002B    76           0063        HALT             ;FLAG OPERATOR
002C                 0064 PATRN: DEFS  1
002D                 0065 BYTE:  DEFS  2
002F    2F00         0066 START: DEFW  $
                     0068 EPAGE: EQU   10H        ;SET UP FOR 4K TEST
                     0069        END
```

MOSTEK®

APPLYING THE Z80 SIO IN ASYNCHRONOUS DATA COMMUNICATIONS

Application Note

Serial asynchronous data links, probably the most prevelent mode of data communications in existence today, require versitile, easy to interface communications devices. The Z80 SIO is just such a device. Although it is just as equally suited in virtually all serial protocol environments, no compromises were made in asynchronous applications. The Z80 SIO operating features include:

□ Data communications rates of up to 800K bits/s

□ Three FIFO receive data buffers per channel

□ Full duplex operation

□ Break generation and detection

□ Parity, overrun, and framing error detection

□ Polled or interrupt driven

The most salient of the SIO's many features is its capability to operate using prioritized vector interrupts, offering unparalleled speed and efficiency in maximizing data throughput. Although the SIO can be operated in polled as well as interrupt modes, the latter shall be emphasized in the following discussion due to its inherent power and versatility.

In order to better understand use of the SIO in serial data communications, a look at the internal organization of the chip would be helpful. Figure 1 depicts the functional logic of one of the SIO channels. As shown, there are a total of 11 registers accessible by the programmer. "Write Registers" (WR0-WR7), as they are referred to, are used to configure the SIO to the desired type of protocol and includes such information as data rates, parity information, word length, etc. In addition, three "Read Registers" (RR0-RR2) are provided for monitoring data flow and error conditions. For a detailed description of these registers, as well as the entire MK3884, reference should be made to the MK3884 Z80 SIO Technical Manual. In the receiver section, notice that data flows from the receive shift register to the three receive buffers.

These registers are configured in a first in, first out (FIFO) arrangement, thus providing the data link with additional overrun protection. Associated with each receive buffer is a corresponding error buffer, enabling the programmer to poll the various Read Registers and ascertain error conditions corresponding to the data. This concept is illustrated in Figure 2. Recieve Buffer 3 contains data, has no associated errors, and will be read next by the CPU, as it is at the top of the stack. Receive Buffer 2 has a parity error associated with it's data word, indicated by the "1" in the parity column. Similarly, Receive Buffer 1 has an associated overrun error condition, indicating it has been overwritten at least once. This type of FIFO arrangement allows the programmer three full receive word-times to read the SIO before losing any data, which is extremely advantageous when the programmer must perform numerous housekeeping functions. The SIO is also capable of full duplex operation, illustrated in Figure 2 by separate data paths for the transmitter and receiver. Notice the separate transmit and receive clock inputs for situations requiring different clock rates. A $\overline{\text{SYNC}}$ input is provided as a general purpose input in asynchronous communications, and is used to establish synchronization in monosync and bisync communications. Finally, all standard modem control signals are present for handshaking including $\overline{\text{DCD}}$, $\overline{\text{DTR}}$, $\overline{\text{CTS}}$, and $\overline{\text{RTS}}$.

The SIO interfaces easily with the Z80 CPU, and generally requires little, if any modification of control signals when used with other CPU's. Figures 3A and 3B show the typical interconnections and addressing techniques between the SIO and CPU. Note that the C/$\overline{\text{D}}$ (control data) and B/$\overline{\text{A}}$ (Channel B/A) pins may be connected to A0 and A1 of the address bus, respectively. Figure 3B further illustrates SIO addressing, where even numbered addresses decode the channel (A or B) and define a data operation. Conversely, odd numbered addresses define a control operation to the addressed port.

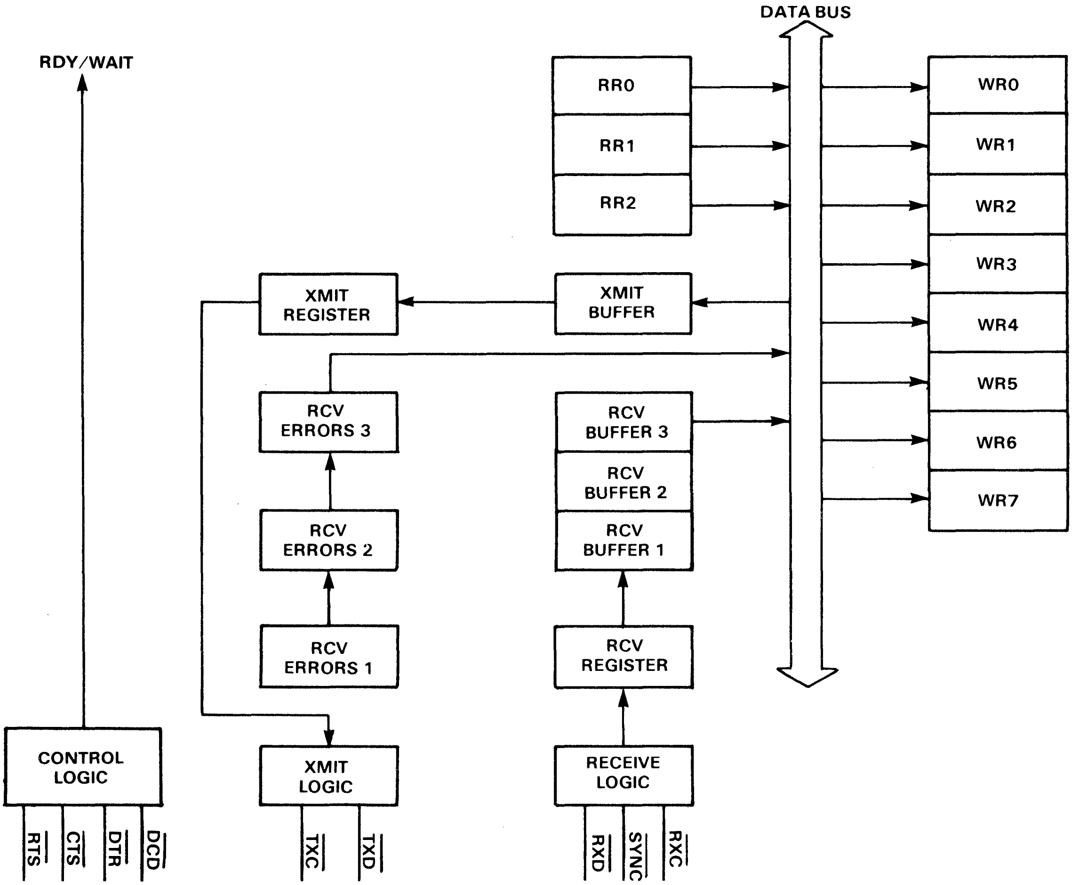
There are also two clock considerations that deserve attention. 1) The SIO is a synchronous device, whose clock (Φ) must be identical to that of the CPU clock. 2) Although the SIO is capable of high data rates, care should be taken to ensure that the system clock (Φ) is at least 5 times the data rate, as specified in the data book.

THE ASYNCHRONOUS MODEL

In discussing the use of the Z80 SIO in Async communications, the illustration in Figure 4 depicts the method in which the SIO receiver logic assembles and transmitter logic sends serial data asynchronously. The data stream consists of one start bit, a variable length data word (selectable 5-8 bits), an optional partity bit, and selectable stop bits (1, 1½ or 2). Note that the data word is sent low order bit first and must be right justified if less than 8 data bits are contained in the data field.

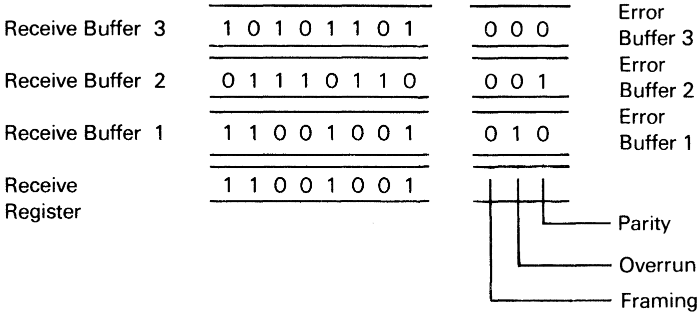
FUNCTIONAL LOGIC OF AN SIO CHANNEL

Figure 1



SIO RECEIVE FIFO DATA/ERROR BUFFERS

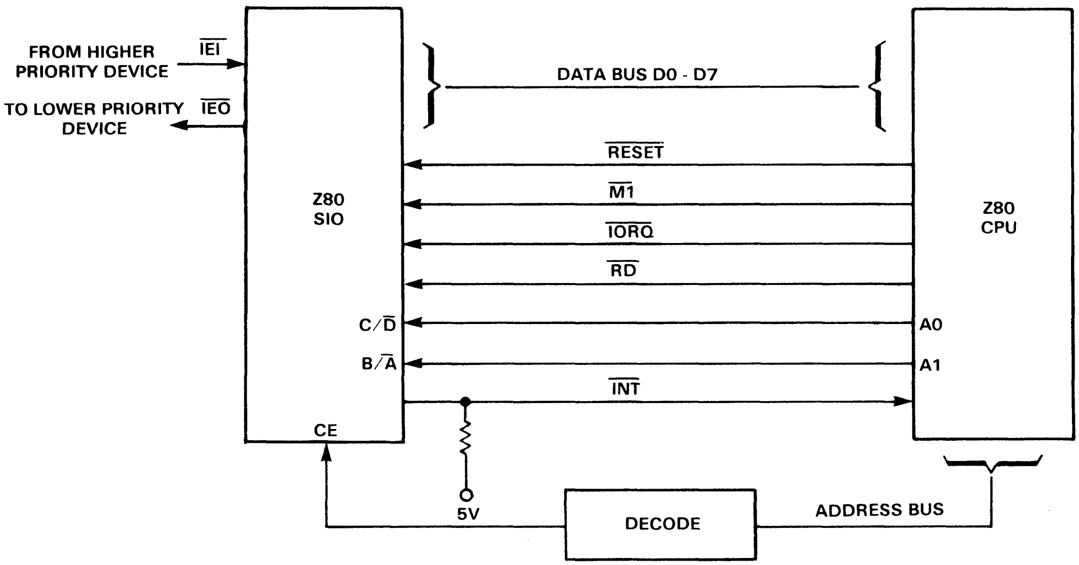
Figure 2



Receive Buffer 3	1 0 1 0 1 1 0 1	0 0 0	Error Buffer 3
Receive Buffer 2	0 1 1 1 0 1 1 0	0 0 1	Error Buffer 2
Receive Buffer 1	1 1 0 0 1 0 0 1	0 1 0	Error Buffer 1
Receive Register	1 1 0 0 1 0 0 1		

Z80 CIO - CPU INTERCONNECTIONS

Figure 3A



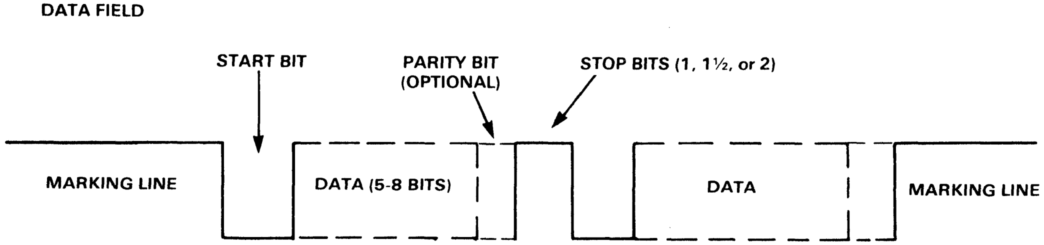
Z80 SIO-CPU INTERCONNECTIONS

Figure 3B

Address Label	$\overline{CE}$	B/$\overline{A}$	C/$\overline{D}$	
SIO ADD	0	0	0	Channel A XMIT/RCV ADDRESS
SIO ADD +1	0	0	1	Channel A READ/WRITE ADDRESS
SIO ADD +2	0	1	0	Channel B XMIT/RCV ADDRESS
SIO ADD +3	0	1	1	Channel B READ/WRITE ADDRESS

ASYNCHRONOUS FORMAT

Figure 4



ASYNCHRONOUS PROGRAMMING

The design of a serial data communications link utilizing the SIO will comprise three basic software modules:

- Initialization
- Data transfer
- Error detection and recovery

The program flow for initialization is illustrated by the flow diagram in Figure 5. This procedure consists of writing a string of control bytes to the SIO using the pointer register, WR0. Each control byte is preceded by the pointer register byte, which tells the SIO which of 8 write registers is to be addressed. Initialization is executed in this alternating pointer register/control-data fashion until the SIO is configured as desired, as shown in Table 1.

Although SIO initialization is not necessarily order dependent, the logical order as depicted in Figure 5 and Table 1 is highly recommended.

Also, a word of caution concerning the use of WR0 is appropriate at this time. As WR0 has a dual function - that of a pointer to other control registers and commands (i.e., reset channel, error reset, etc.) to the SIO - it is possible to perform both of these functions simultaneously. This is not recomended because following each command, the internal register pointer resets to zero, thus preventing the ensuing control word from loading properly. Each command issued should address WR0, as pointed out in the example.

Data transfer and error handling methods are presented in there simplest form in Table 1. The first eight bytes of code initialize the SIO, which consists of initializing the CPU internal registers B, C, and HL with the table length, port address, and table address, respectively. Notice how efficiently the use of the OTIR instruction transfers the entire block of data to the SIO. Although channel A is the active channel being used in this example, channel B must also be accessed, as shown. This is because the WR2 and the status affects vector bit are active in channel B only.

Another instruction of interest is the "EI" instruction, both because of it's existence and placement. Whenever the CPU acknowledges an interrupt, interrupts within the CPU are disabled and remain so until an "EI" instruction is executed. Hence, the placement of "EI" in the program example forms a non-nested interrupt structure. Conversly, placing "EI" at the beginning of a subroutine would constitute nested interrupts, as other devices could now cause interrupts.

The interrupts themselves may be initiated by the SIO in many different ways. The transmitter and receiver interrupts are initiated when the transmit buffer empty and receiver character available bits are set. In the case of receive errors, interrupt requests are made (if programmed to do so) when any of several special error conditions exist. As shown in the program initialization (Table 1), the special effects vector is enabled, allowing the SIO to modify the returned vector, indicating either a) transmit buffer full, b) receive-character available, c) External/Status change, or d) special receive conditions. This powerful feature further reduces programming overhead and thus allows greater efficiency and data throughput. Also, at the programmers discretion is the ability to initiate data transfer automatically by monitoring the modem control signals. This is effected by the SIO detecting $\overline{\text{DCD}}$ and $\overline{\text{CTS}}$ in an active state which, in turn, enables the receiver and transmitter, respectively. Also, if External interrupts are enabled as they are in the example, interrupts are generated upon transition of $\overline{\text{DCD}}$ or $\overline{\text{CTS}}$. This feature is useful in initiating line turn-around and detecting break conditions. Once External/Status Interrupts have been acknowledged, they must be reset by writing to register 0 of the appropriate channel. Note also in the initialization procedure that immediately following a chip or channel reset, the "Reset External/Status Interrupts" command should be executed to prevent possible spurious interrupts.

Should the programmer choose not to operate in an interrupt mode, all of the aforementioned conditions would have to be polled by reading the appropriate read register (RR0-RR2). When operating in this mode, the proper sequence of checking the SIO for receive characters would be:

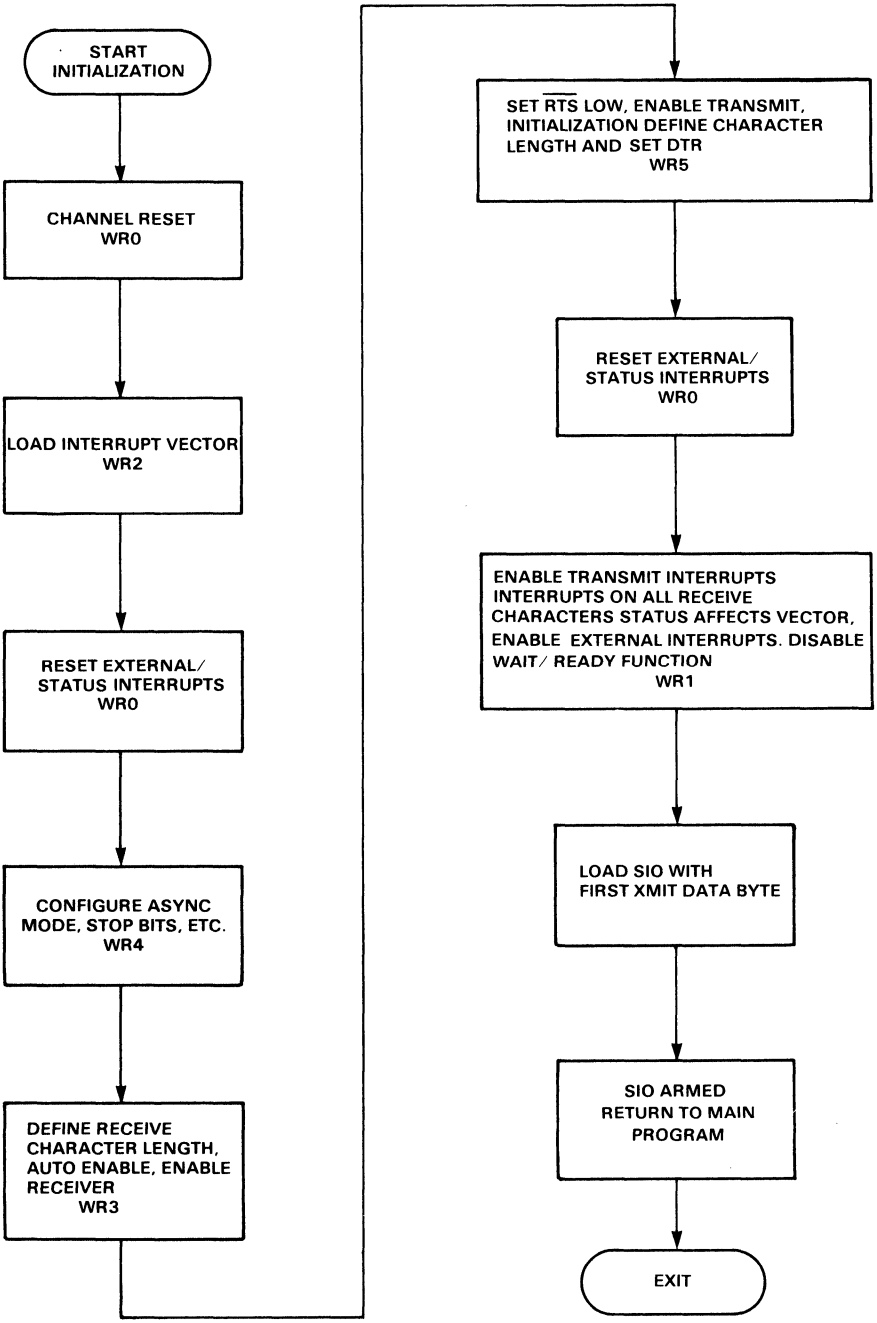
1. Read RR0; determine if a character is available.
2. If so, interrogate RR1 to ascertain error status.
3. Read the DATA.

The status of errors should be checked before reading the data to preserve the proper error to data word correspondence. An example of reading the Read Registers is given in Table 1 under Error Handler. As illustrated, RR1 is accessed by first performing a "write" to WR0 which points to register 1, and followed by a "READ" operation. Once the status byte is in the accumulator, each of the pertinent error bits are interrogated using the "bit" instruction. Associated with each type of error is it's error routine which takes the appropriate recovery action. When interrupts are used, as in this example, care should be taken within each Error Routine to perform an Error Reset command, thus allowing future error interrupts to occur.

The preceding example should equip the user with a "guide" for programming the SIO, not only in asynchronous communications, but synchronous and SDLC/HDLC as well. Of course, the latter protocols deserve special attention, and are covered in detail in the MK3884 Technical Manual. As demonstrated, the SIO, when taken advantage of, can be an extremely powerful device in any data communications link.

LOGICAL FLOW OF SIO INITIALIZATION

Figure 5


START INITIALIZATION
CHANNEL RESET WR0
LOAD INTERRUPT VECTOR WR2
RESET EXTERNAL/ STATUS INTERRUPTS WR0
CONFIGURE ASYNC MODE, STOP BITS, ETC. WR4
DEFINE RECEIVE CHARACTER LENGTH, AUTO ENABLE, ENABLE RECEIVER WR3
SET $\overline{RTS}$ LOW, ENABLE TRANSMIT, INITIALIZATION DEFINE CHARACTER LENGTH AND SET DTR WR5
RESET EXTERNAL/ STATUS INTERRUPTS WR0
ENABLE TRANSMIT INTERRUPTS INTERRUPTS ON ALL RECEIVE CHARACTERS STATUS AFFECTS VECTOR, ENABLE EXTERNAL INTERRUPTS. DISABLE WAIT/ READY FUNCTION WR1
LOAD SIO WITH FIRST XMIT DATA BYTE
SIO ARMED RETURN TO MAIN PROGRAM
EXIT

TYPICAL PROGRAM EXAMPLE

Table 1

```
LABEL   SOURCE STATEMENT  COMMENTS

UNIT    LD B, LENG B      ; LENGTH of Table, CH B
        LD C, SIOCTL +2   ; Port Address, CH B
        LD HL, CTLTB      ; TABLE Address, CH B
        OTIR              ; Initialized CH B
        LD B, LENG A      ; Length of Table, CH A
        LD C, SIOCTL      ; Port Address, CH A
        LD HL, CTLTA      ; Table Address, CH A
        OTIR              ; Initialize CH A

CTRLTA  DEFB '18' H       ; WR0, RESET CH A
         DEFB '10' H      ; WR0, Reset External/Status
                            Interrupts
        DEFB '04' H       ; Pointer to WR4
        DEFB '40' H       ; X16 CLK, ODD Parity, 2 stop
                            bits
        DEFB '03' H       ; Pointer to WR3
        DEFB '61' H       ; 7 bits/char, receive and
                            auto enable
        DEFB '05' H       ; Pointer to WR5
        DEFB 'AA' H       ; Set RTS, DTR; 7 bits/char.,
                            enable Xmit
        DEFB '10' H       ; WR0; reset EXT/STATUS
                            INT.
        DEFB '01' H       ; Pointer to WR1
        DEFB '17' H       ; Enable external and
                            transmit interrupts,
                            status affects vector,
                            interrupt on all RCV
                            characters.

CTLTB   DEFB '18' H       ; WR0, Reset CH B
        DEFB '10' H       ; WR0, Reset External/Status
                            Interrupts
        DEFB '02' H       ; Pointer to WR2
        DEFB '00' H       ; Load Interrupt Vector
        DEFB '01' H       ; Pointer to WR1
        DEFB '14' H       ; Status affects vector
```

TRANSMIT DATA HANDLER

```
LABEL   SOURCE STATEMENT  COMMENTS

XMTINT  EX AF,AF'         ; Save Registers
        LD A,(T BUF)      ; Load Character
        OUT (SIODAT),A    ; Ship it Out
        EX AF,AF'         ; Restore Registers
        EI                ; Re-enable interrupts
        RETI
```

RECEIVE DATA HANDLER

```
RCVINT  EX AF,AF'         ; Save Registers
        IN A,(SIODAT)     ; Read Character
        LD (RBUF),A       ; Save in Memory
        EX AF,AF'         ; Restore Registers
        EI
        RETI
```

ERROR HANDLER

```
INTERR  EX AF,AF'         ; Save Registers
        LD A, '01' H      ; Set Pointer to
        OUT (SIOCTL),A    ; Reg. 1
        IN A,(SIODAT)     ;
        BIT 6,A           ;
        JR Z, FMER        ; Framing Error
        BIT 5,A           ;
        JR Z,ORER         ; Overrun Error
        JP PAER           ; Parity Error
        EX AF,AF'         ; Restore Registers
        EI
        RETI
```

MOSTEK®

USE OF THE MK3805 CLOCK/RAM

Application Note

INTRODUCTION

Many microprocessor applications require a real time clock and/or memory that can be battery powered with very low power drain. A typical application might be an automobile trip computer, where the clock could provide the time of day and the memory would be used to retain vital information when the ignition switch is off. The interfacing technique needs to be kept as simple as possible so as to minimize the required overhead in software, and it should minimize the number of pins required in order that other I/O requirements can be efficiently accommodated.

FEATURES

Mostek's CLOCK/RAM microcomputer peripheral chip satisfies all of these requirements. The device, designated MK3805, contains a real-time clock/calendar, 24 bytes of static RAM, an on-chip oscillator and communicates serially with the microcomputer via a simple interface protocol. The MK3805 is fabricated using CMOS technology, thus insuring very low power consumption.

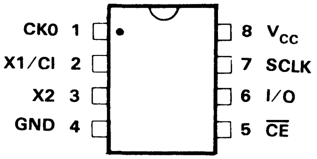
The real-time clock/calendar provides all timekeeping functions. It contains registers for seconds, minutes, hours, day, date, month, and year. The end of the month date is automatically adjusted for months with less than 31 days. The clock operates in either the 24 hour or 12 hour format with an AM/PM indicator. Since the MK3805 is designed to interface to a microcomputer, the alarm function is easily accommodated in the microcomputer, should it be required.

The on-chip oscillator provides the clock source for the clock/calendar. It incorporates a programmable divider so that a wide variety of crystal frequencies can be accommodated. The oscillator also has an output available that is designed to serve as the clock generator for the microcomputer. A separately programmable divider provides several different output frequencies for any given crystal frequency. This feature can eliminate having to use a separate crystal or external oscillator for the microcomputer, thereby reducing system cost.

Interfacing the CLOCK/RAM with a microcomputer is greatly simplified using asynchronous serial communication. Only 3 lines are required to communicate with the CLOCK/RAM: (1) $\overline{CE}$ (chip enable), (2) I/O (data line), and (3) SCLK (shift register clock). Data can be transferred to and from the CLOCK/RAM one byte at a time, or in a burst of up to 24 bytes.

PINOUT DIAGRAM

Figure 1



PIN	NAME	DESCRIPTION
1	CKO	System clock (output).
2	X1/CI	Crystal or external clock (input).
3	X2	Crystal (input).
4	GND	Ground.
5	$\overline{CE}$	Chip enable (input, active low).
6	I/O	Data I/O (input/output).
7	SCLK	Shift register clock (input).
8	V_{CC}	Positive supply voltage.

PINOUT DESCRIPTION

Figure 1 is a pinout diagram of the MK3805. It is packaged in an 8-pin DIP to conserve PC board space. A brief description of the function of each pin is listed.

TECHNICAL DESCRIPTION

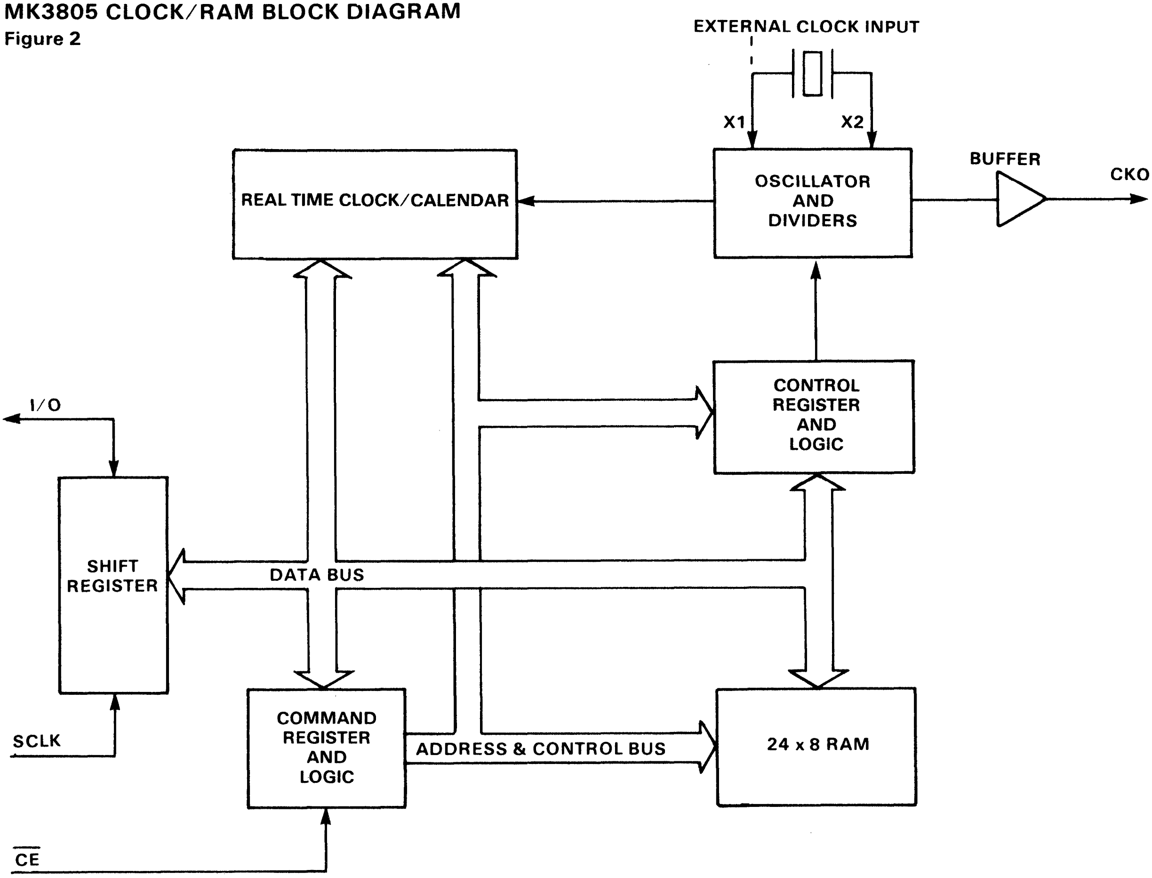
Figure 2 is a block diagram of the CLOCK/RAM chip. The main components are the oscillator and divider, the real time clock/calendar, the static RAM, the command register and logic, the control register and logic, and the serial shift register.

The shift register is used to communicate with the outside world. Data on the I/O line is either input or output on each shift register clock pulse when the chip is enabled. If the chip is in the input mode, the data on the I/O line is input to the shift register on the rising edge of SCLK. If in the output mode, data is shifted out onto the I/O line on the falling edge of SCLK.

The command register receives the first byte input by the shift register after $\overline{CE}$ goes true (low). This byte must be the command byte and will direct further operations within the CLOCK/RAM. The command specifies whether subsequent transfers will be read or written, and what register or RAM location will be involved.

MK3805 CLOCK/RAM BLOCK DIAGRAM

Figure 2



The control register has bits defined which control the divider for the internal real-time clock and the external system clock. One bit serves as the write protect control flag, preventing accidental write operations during power-up or power-down situations.

The real-time clock/calendar is accessed via seven registers. These registers contain seconds, minutes, hours, day, date, month, and year information. Certain bits within these registers also control a run/stop function, 12/24 hour clock mode, and indicate AM or PM (12 hour mode only). These registers can be accessed either randomly in byte mode, or sequentially in burst mode.

The static RAM is organized as 24 bytes of 8-bits each. They can be accessed either randomly in byte mode, or sequentially in burst mode.

The reader should refer to the MK3805 data sheet for operating specifications and detailed timing information.

DATA TRANSFERS

Data transfer is accomplished under control of the $\overline{CE}$ and SCLK inputs by an external microcomputer. Each transfer consists of a single byte (COMMAND) input followed by a single or multiple byte input or output (as defined by the command byte).

The general format for the command byte is shown in Figure 3. The most significant bit (bit 7) must be a logical 1; bit 6 specifies a clock function if logical 0 or a RAM function if logical 1. Bits 1-5 specify the clock register(s) or RAM location(s) to be accessed. The least significant bit (bit 0) specifies a write operation if a logical 0 or a read operation if a logical 1.

In the clock burst mode, all clock, calendar, and control registers are transferred beginning with register 0 (seconds) and ending with register 7 (control). Unless terminated early, this burst mode requires that $\overline{CE}$ be true and 72 SCLK cycles be supplied. This mode may be terminated at any time by taking $\overline{CE}$ false. This mode is specified by setting all address bits in the command byte to a logical 1.

In the RAM burst mode, all RAM locations are transferred beginning with location 0 and ending with location 23 (017H). Unless terminated early, this burst mode transfer

Figure 3

COMMAND, REGISTER, DATA FORMAT SUMMARY

I. GENERAL COMMAND FORMAT:

7	6	5	4	3	2	1	0
1	$R/\overline{C}$	A_4	A_3	A_2	A_1	A_0	$R/\overline{W}$

II. CLOCK COMMAND FORMAT:

	7	6	5	4	3	2	1	0
SEC	1	0	0	0	0	0	0	$R/\overline{W}$
MIN	1	0	0	0	0	0	1	$R/\overline{W}$
HR	1	0	0	0	0	1	0	$R/\overline{W}$
DATE	1	0	0	0	0	1	1	$R/\overline{W}$
MONTH	1	0	0	0	1	0	0	$R/\overline{W}$
DAY	1	0	0	0	1	0	1	$R/\overline{W}$
YEAR	1	0	0	0	1	1	0	$R/\overline{W}$
CONTROL	1	0	0	0	1	1	1	$R/\overline{W}$
CLOCK BURST	1	0	1	1	1	1	1	$R/\overline{W}$

IV. CLOCK PROGRAMMING MODEL:

	7	6	5	4	3	2	1	0
00-59	STOP	10 SEC			SEC			
00-59	10 MIN				MIN			
01-12 00-23	12/ 24	0	10 A/P	HR	HR			
01-28/29 01-30 01-31	T_1	0	10 DATE		DATE			
01-12	0	0	0	10 M	MONTH			
01-07	T_2	0	0	0	0	DAY		
0-99	10 YEAR				YEAR			
	WP	C_1	C_0	X_4	X_3	X_2	X_1	X_0

NOTES:

WP Write protect.
X_4 - X_0 Program dividers for real time clock.
C_1 - C_0 Program dividers for clock output.
T_2 - T_1 Test bits (normally set to 0).

III. RAM COMMAND FORMAT:

	7	6	5	4	3	2	1	0
RAM 0	1	1	0	0	0	0	0	$R/\overline{W}$
				● ● ●				
RAM 23	1	1	1	0	1	1	1	$R/\overline{W}$
RAM BURST	1	1	1	1	1	1	1	$R/\overline{W}$

requires that $\overline{\mathrm{CE}}$ be true and 200 SCLK cycles be supplied. This mode may be terminated at any time by taking $\overline{\mathrm{CE}}$ false. This mode is specified by setting all address bits in the command byte to a logical 1.

Refer to Figure 3 for a summary of the command, register, and data formats.

POWER-ON STATES

When the MK3805 is first powered up, all eight clock registers come up to a pre-defined state. These are listed below. The RAM locations contain unspecified data.

Clock:

Seconds	00	
Minutes	00	
Hours	00	
Date	01	
Month	01	
Day	01	
Year	00	
Halt	1	(clock stopped)
12/24 Hour	0	(24 hour mode)

Control:

Write Protect	1	(protect on)
C0 & C1	01	(CKO = crystal frequency /2)
X3 & X4	00	(crystal frequency is binary: 2^h)
X0, X1 & X2	000	(divide by 2^{23})

SERIAL TIMING

The timing sequence for data transfer with the CLOCK/RAM is started when $\overline{\mathrm{CE}}$ goes low (see Figure 4). After $\overline{\mathrm{CE}}$ goes low, the next 8 SCLK cycles will input the command byte of the proper format. If the most significant bit (bit 7) is a logical 0, the command byte will be ignored, as will all SCLK cycles until $\overline{\mathrm{CE}}$ goes high and returns low to signify the start of a new transfer. Command bits are input on the rising edge of SCLK.

Input data will be input on the rising edge of the next 8 SCLK cycles (per byte if burst mode is specified). Additional SCLK cycles will be ingored, should they inadvertently occur.

Output data will be output on the falling edge of the next 8 SCLK cycles (per byte if burst mode is specified). Additional SCLK cycles will retransmit the information, thereby permitting continuous transmission of clock information for certain applications.

A data transfer will terminate if $\overline{\mathrm{CE}}$ goes high, and the transfer must be reinitiated by the proper command when $\overline{\mathrm{CE}}$ goes low again. The I/O pin will be in the high impedance state when $\overline{\mathrm{CE}}$ is high.

DESIGN EXAMPLE

As a demonstration of the software and hardware interfacing for the CLOCK/RAM chip, the design of a demonstration used for electronic shows is given here. The hardware used was a standard CRT terminal, an MK38P73 single chip microcomputer, the MK3805 CLOCK/RAM chip, and some miscellaneous parts to interface to the CRT. Refer to Figure 5 for a schematic of the circuit used. Note how simple the design is. The MK3805 interfaces directly to the MK38P73 via 3 pins, and it provides the clock input to the MK38P73 via a fourth pin.

HARDWARE DESCRIPTION

The MK38P73 is an 8-bit single-chip microcomputer with 4 parallel ports, a serial port, 128 bytes of RAM, and 2K bytes of EPROM (in the form of a piggy back 2716). Because the serial communications with the CLOCK/RAM uses a simple shift register type interface, the serial port of the 38P73 is not used here. It remains free for serial communications with the CRT.

The MK3805 is interfaced to the microcomputer via port 4. This is done to take advantage of the $\overline{\mathrm{STB}}$ line associated with that port. The $\overline{\mathrm{STB}}$ line goes low for a short time after each output to port 4 instruction is executed. This normally would be used to strobe data into an output device attached to the port. In this example, the $\overline{\mathrm{STB}}$ line provides the SCLK pulse to the CLOCK/RAM shift register to clock data into and out of the chip. By using this line, toggling another port bit to strobe data in and out is not required. Such an interface to other microcomputers is straightforward.

The CLOCK/RAM chip also provides the clock source for the microcomputer. By selecting a crystal frequency of 3.6864 MHz and setting the CKO divider to divide by 1, the serial port on the MK38P73 operates at standard Baud rates (9600, 4800, 2400, 1200, etc.).

The 75150 and 1489 chips convert the TTL level signals output by the microcomputer to RS-232 levels in order that the circuit can be interfaced to a standard CRT.

SOFTWARE DESCRIPTION

The heart of the software is the subroutine labeled 'CLKRAM'. This subroutine provides all the necessary software interfacing to the CLOCK/RAM.

Before calling the subroutine, the necessary parameters must be set up in the proper registers. The ISAR is used as a pointer to where the data is to be read from or written to in the MK38P73 RAM area.

The scratchpad register 'CMD' must contain the command to be sent to the CLOCK/RAM. (See the description of the command given earlier.)

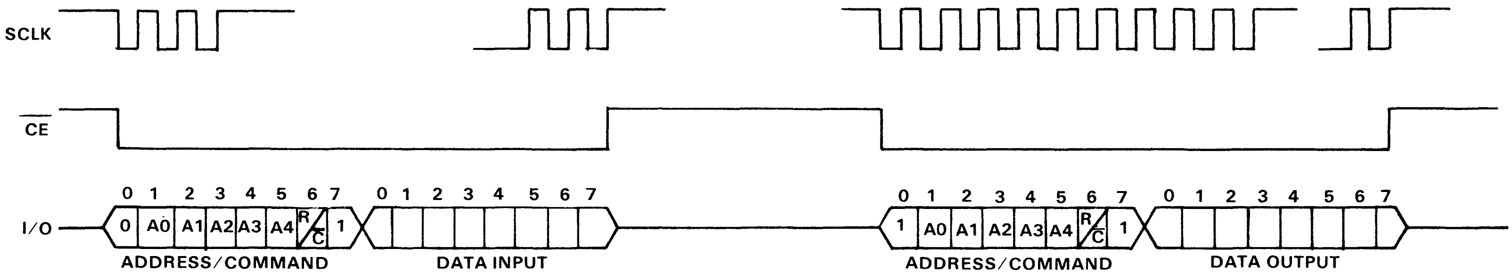
The bit pattern for enabling the CLOCK/RAM must be

MK3805 RAM DATA TRANSFER SUMMARY

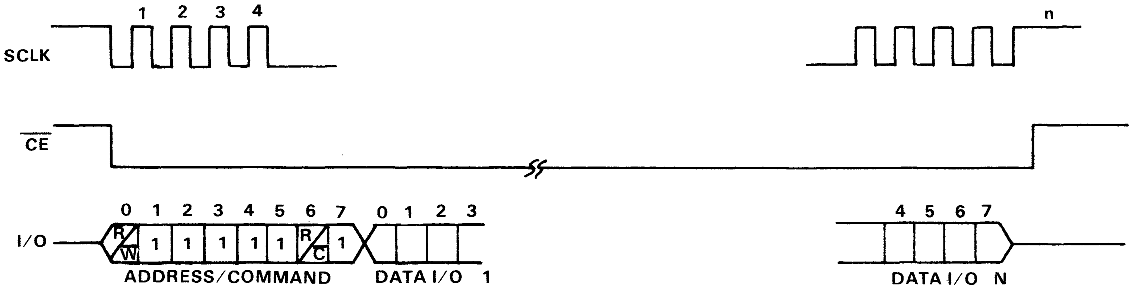
Figure 4

Notes: 1) Data input sampled on rising edge of clock.
2) Data output changes on falling edge of clock.
3) Rising edge of CE terminates operation and resets command register.

I. SINGLE BYTE TRANSFER



II. BURST MODE TRANSFER



FUNCTION	N	n
CLOCK	8	72
RAM	24	200

stored in the scratchpad register 'CHIPEN'. This bit pattern should contain a logic 1 in the bit position that corresponds to the port 4 line tied to the CLOCK/RAM $\overline{CE}$ pin. All other bits should be 0. This technique allows multiple serial microcomputer peripheral chips to be tied together with common I/O and SCLK lines, with a separate port line for each device $\overline{CE}$.

The subroutine also provides an option for using the port 4 pins not used by the CLOCK/RAM interface for any other purpose. To accomplish this, a copy of whatever is written to port 4 by other routines must be kept in the scratchpad register 'PT4IMG'. This option is not used in this example.

The main demonstration routine (listing 1) is quite basic. Its purpose is to print the features of the CLOCK/RAM on the CRT, then read the clock and display it's contents once every second. A reentry point is provided in order that the clock/calendar settings may be changed after power up. (See the flowchart in Figure 6.)

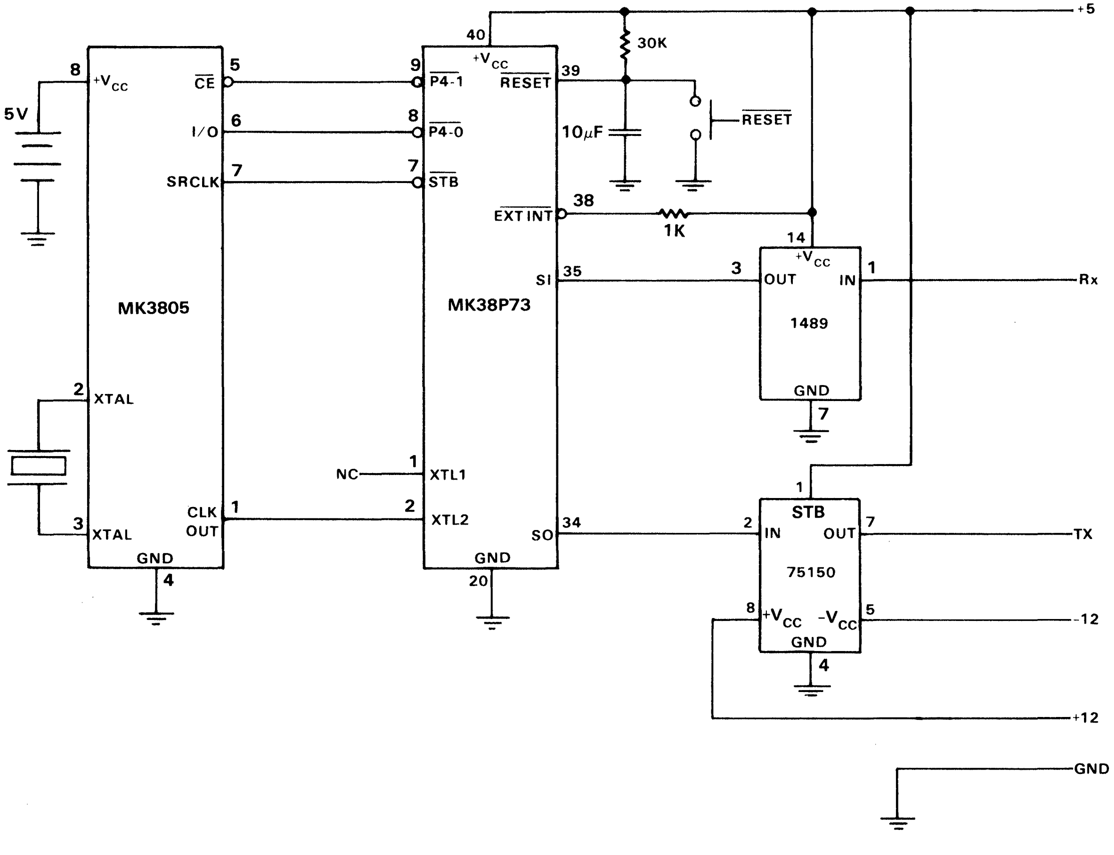
When power is applied to the microcomputer, it resets and begins execution of the program at location 0000H. The code at this point initializes the system and checks for valid CLOCK/RAM data. This condition is indicated by the state of the write protect bit in the control byte. If the bit is set to a logical 1, then the CLOCK/RAM has also just been powered up. This indicates that the registers contain invalid data and should be initialized before continuing. If the bit is reset to a logical 0, the CLOCK/RAM did not just power up, and the data in its registers should be valid.

After the clock data is verified, the routine prints a message consisting of CLOCK/RAM features. The timer is then set to interrupt once every 1/36 second so that the time, etc., may be updated on the CRT screen. The routine then just waits for an interrupt from the timer or the keyboard.

When a timer interrupt occurs, the service routine checks to see if 1 second has elapsed since the last service. If not, it resets the timer and returns to the wait for interrupt state. If 1 second has gone by, the routine proceeds to erase the

SCHEMATIC OF DEMONSTRATION CIRCUIT

Figure 5



time, etc., from the top of the screen and print new data obtained from the CLOCK/RAM. The timer is then reset and returns to the wait for interrupt state.

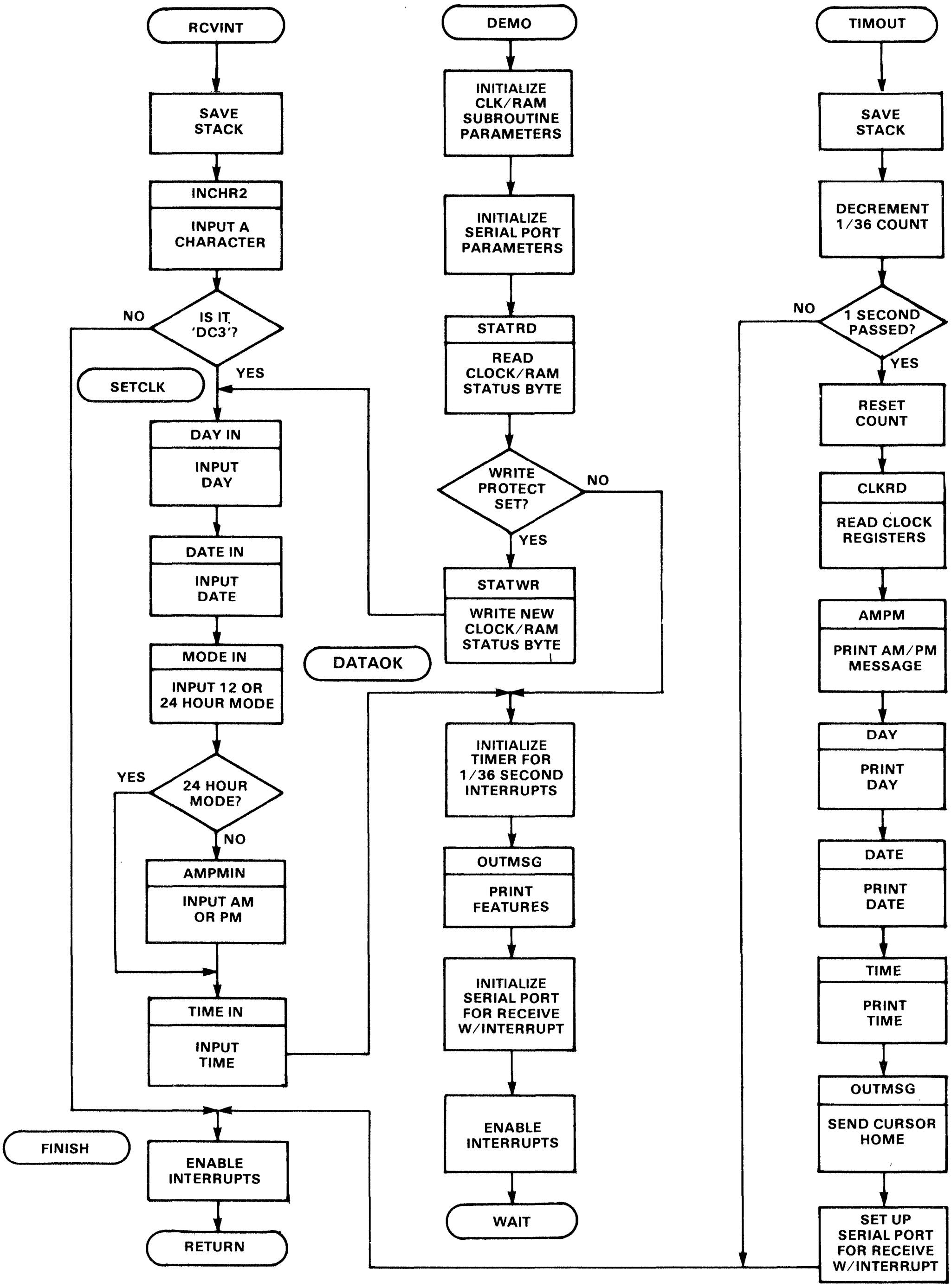
When a receiver interrupt occurs, the serial port contains a valid character from the keyboard. The service routine checks to see if it is a 'DC3' (control-S) character. If not, the routine returns to the wait for interrupt state. If it is, the routine goes to the clock set entry point of the main routine and the user is allowed to set the clock and calendar values. The main routine entered in this fashion is executed similarly to a power on reset with the CLOCK/RAM write protect bit set to a logical 1.

The CLOCK/RAM subroutine (listing 2) was designed to send the command to the CLOCK/RAM chip and then transfer the number of data bytes specified by the command.

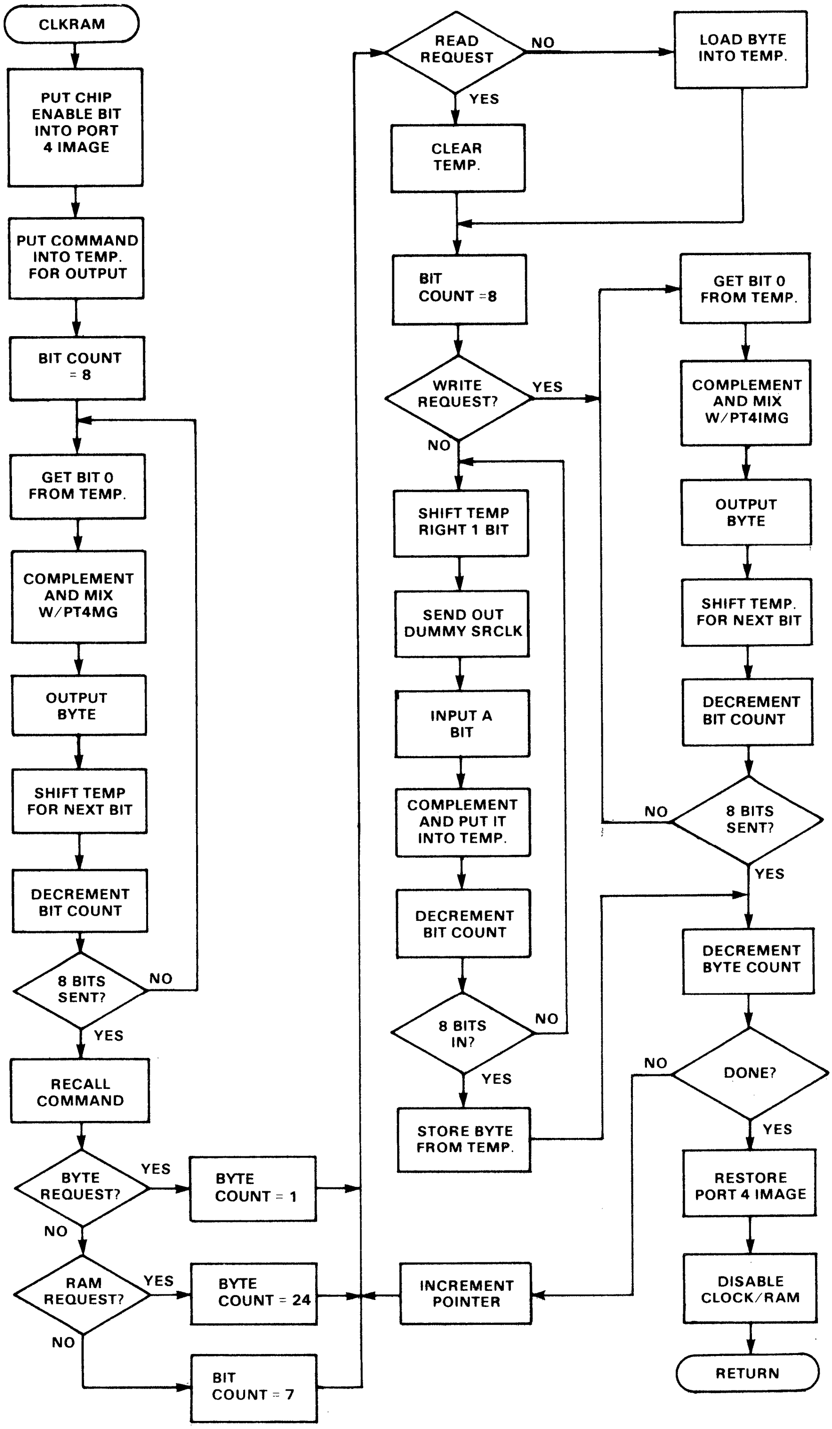
As seen in the flowchart (Figure 7), either 1, 7, or 24 bytes of data may be transferred between the microcomputer and the CLOCK/RAM. The command sent to the subroutine is exactly the command sent to the CLOCK/RAM, so there is no confusion as to the format of the command byte. When this routine is called, the ISAR must be pointing to the scratchpad RAM area where the data transferred is to be read from or written to. Note that only 7 bytes are transferred in a clock burst. This is to eliminate reading and writing the control register every time.

MAIN ROUTINE FLOWCHART

Figure 6



CLKRAM SUBROUTINE FLOWCHART
Figure 7



LISTING 1 - DEMO PROGRAM

```
CLOCK/RAM DEMONSTRATION MODULE    F8/3870 MACRO CROSS ASSM.  V2.2
LOC  OBJ.CODE      STMT-NR SOURCE-STMT PASS2 DEMO   DEMO   DEMO   ABS

                         1         TITLE CLOCK/RAM DEMONSTRATION MODULE
                         2         NAME DEMO
                         3         PSECT ABS
                         4         GLOBAL CLKRAM
                           *
                           * THIS MODULE MUST BE LINKED WITH THE CLOCK/RAM MODULE
                           * TO CREATE A WORKING PROGRAM.
                           *
                           *
                           **********************************
                           *                                *
                           * DEMO FOR MK3805 CLOCK/RAM CHIP *
                           *                                *
                           **********************************
                           ************************************
                           *                                  *
                           * SCRATCH PAD REGISTER DEFINITIONS *
                           *                                  *
                           ************************************
                           *
                           * GLOBAL REGISTERS. THESE REGISTERS MUST BE THE SAME
                           * AS IN THE CLOCK/RAM MODULE.
                           *
        =0000           25 PT4IMG  EQU  00H          ;PORT 4 IMAGE STORAGE
        =0001           26 CHIPEN  EQU  01H          ;CHIP ENABLE STORAGE
        =0002           27 CMD     EQU  02H          ;COMMAND STORAGE
                           *
                           * LOCAL REGISTERS. THESE REGISTERS DO NOT NEED TO BE
                           * MADE KNOWN TO THE CLOCK/RAM MODULE.
                           *
        =0003           32 TEMP    EQU  03H          ;TEMPERARY STORAGE
        =0004           33 CNTSAV  EQU  04H          ;DIGIT COUNT SAVE
        =0005           34 DCOUNT  EQU  05H          ;DIGIT COUNTER
        =0006           35 TIMCNT  EQU  06H          ;TIMER COUNTER
        =0007           36 CTRL    EQU  07H          ;CLOCK/RAM CONTROL STORAGE
        =0010           37 SECOND  EQU  10H          ;SECOND BUFFER
        =0011           38 MINUTE  EQU  11H          ;MINUTE BUFFER
        =0012           39 HOUR    EQU  12H          ;HOUR BUFFER
        =0013           40 DAY     EQU  13H          ;DAY BUFFER
        =0014           41 DATE    EQU  14H          ;DATE BUFFER
        =0015           42 MONTH   EQU  15H          ;MONTH BUFFER
        =0016           43 YEAR    EQU  16H          ;YEAR BUFFER
                           *
                           ********************
                           *                  *
                           * PORT DEFINITIONS *
                           *                  *
                           ********************
                           *
        =0004           51 CRDATA  EQU  04H          ;CLOCK/RAM DATA PORT
        =0006           52 TICTRL  EQU  06H          ;TIMER, INTERUPT CTRL PORT
        =0007           53 TIMER   EQU  07H          ;TIMER PORT
        =000C           54 RXCTRL  EQU  0CH          ;SERIAL CONTROL PORT
        =000D           55 RXSTAT  EQU  0DH          ;SERIAL STATUS PORT
        =000E           56 MSBYTE  EQU  0EH          ;SERIAL MSB PORT
        =000F           57 LSBYTE  EQU  0FH          ;SERIAL LSB PORT
                           *
                           *********************
                           *                   *
                           * ASCII DEFINITIONS *
                           *                   *
                           *********************
                           *
        =0004           65 EOT     EQU  04H          ;END OF TEXT
        =000A           66 LF      EQU  0AH          ;LINE FEED
        =000C           67 FF      EQU  0CH          ;FORM FEED
        =000D           68 CR      EQU  0DH          ;CARIAGE RETURN
        =0013           69 DC3     EQU  13H          ;DEVICE CONTROL 3 (^S)
        =001B           70 ESC     EQU  1BH          ;ESCAPE
```

```
                              *************
                              *           *
                              * CONSTANTS *
                              *           *
                              *************
                              *
                              * DAYS OF THE WEEK
                              *
      =0001                80 SUN      EQU  1             ;SUNDAY IS DAY 1
      =0002                81 MON      EQU  2             ;MONDAY IS DAY 2
      =0003                82 TUES     EQU  3             ;TUESDAY IS DAY 3
      =0004                83 WED      EQU  4             ;WEDNESDAY IS DAY 4
      =0005                84 THURS    EQU  5             ;THURSDAY IS DAY 5
      =0006                85 FRI      EQU  6             ;FRIDAY IS DAY 6
      =0007                86 SAT      EQU  7             ;SATURDAY IS DAY 7
                              *
                              * MONTHS OF THE YEAR
                              *
      =0001                90 JAN      EQU  1             ;JANUARY IS MONTH 1
      =0002                91 FEB      EQU  2             ;FEBRUARY IS MONTH 2
      =0003                92 MARCH    EQU  3             ;MARCH IS MONTH 3
      =0004                93 APRIL    EQU  4             ;APRIL IS MONTH 4
      =0005                94 MAY      EQU  5             ;MAY IS MONTH 5
      =0006                95 JUNE     EQU  6             ;JUNE IS MONTH 6
      =0007                96 JULY     EQU  7             ;JULY IS MONTH 7
      =0008                97 AUG      EQU  8             ;AUGUST IS MONTH 8
      =0009                98 SEPT     EQU  9             ;SEPTEMBER IS MONTH 9
      =000A                99 OCT      EQU  10            ;OCTOBER IS MONTH 10
      =000B               100 NOV      EQU  11            ;NOVEMBER IS MONTH 11
      =000C               101 DEC      EQU  12            ;DECEMBER IS MONTH 12
                              *
                              * COUNTER VALUES
                              *
      =0000               105 ZERO     EQU  0             ;COUNT IS 0
      =0001               106 ONE      EQU  1             ;COUNT IS 1
      =0002               107 TWO      EQU  2             ;COUNT IS 2
      =0003               108 THREE    EQU  3             ;COUNT IS 3
      =0004               109 FOUR     EQU  4             ;COUNT IS 4
      =0005               110 FIVE     EQU  5             ;COUNT IS 5
      =0006               111 SIX      EQU  6             ;COUNT IS 6
      =0007               112 SEVEN    EQU  7             ;COUNT IS 7
      =0008               113 EIGHT    EQU  8             ;COUNT IS 8
      =0009               114 NINE     EQU  9             ;COUNT IS 9
      =000A               115 TEN      EQU  10            ;COUNT IS 10
      =0010               116 TENBCD   EQU  10H           ;BCD VALUE OF 10
                              *
                              * BCD MASKS
                              *
      =000F               120 LSD      EQU  0FH           ;MASK FOR ONE'S DIGIT
      =00F0               121 MSD      EQU  0F0H          ;MASK FOR TEN'S DIGIT
                              *
                              * LEAP YEAR MASKS
                              *
      =0013               125 LEAP1    EQU  13H           ;MASK TO CHECK FOR ?
      =0012               126 LEAP2    EQU  12H           ;MASK TO CHECK FOR ?
                              *
                              * ISAR MASK
```

```
                          *
        =003F        130 ISMASK  EQU  3FH        ;MASK TO 6 BITS
                          *
                          * CLOCK/CALENDAR MASKS
                          *
        =0080        134 HALT    EQU  80H        ;HALT FLAG IS BIT 7 OF SECON
                                                  S
        =0070        135 SECMSD  EQU  70H        ;SECONDS TEN'S DIGIT
        =000F        136 SECLSD  EQU  0FH        ;SECONDS ONE'S DIGIT
        =0070        137 MINMSD  EQU  70H        ;MINUTES TEN'S DIGIT
        =000F        138 MINLSD  EQU  0FH        ;MINUTES ONE'S DIGIT
        =0080        139 MODE    EQU  80H        ;12/24 HOUR MODE IS BIT 7 OF
                                                  HOURS
        =0020        140 AMPM    EQU  20H        ;AM/PM FLAG IS BIT 5 OF HOUR
        =0030        141 HR2MSD  EQU  30H        ;24 HOUR MODE TEN'S DIGIT
        =0010        142 HR1MSD  EQU  10H        ;12 HOUR MODE TEN'S DIGIT
        =000F        143 HRLSD   EQU  0FH        ;HOURS ONE'S DIGIT
        =0007        144 DAYLSD  EQU  07H        ;DAY MASK
        =0030        145 DATMSD  EQU  30H        ;DATE TEN'S DIGIT
        =000F        146 DATLSD  EQU  0FH        ;DATE ONE'S DIGIT
        =0010        147 MNMSD   EQU  10H        ;MONTH TEN'S DIGIT
        =000F        148 MNLSD   EQU  0FH        ;MONTH ONE'S DIGIT
        =00F0        149 YRMSD   EQU  0F0H       ;YEARS TEN'S DIGIT
        =000F        150 YRLSD   EQU  0FH        ;YEARS ONE'S DIGIT
                          *
                          * TIMER VALUES
                          *
        =0024        154 MAXCNT  EQU  36         ;TIMER MAXIMUM COUNT
        =00EA        155 TMCTRL  EQU  0EAH       ;TIMER CONTROL BYTE
                          *
                          * CHIP ENABLE BITS
                          *
        =0001        159 DATA    EQU  01H        ;DATA BIT IS BIT 0
        =0002        160 CE1     EQU  02H        ;CHIP ENABLE BIT IS BIT 1
                          *
                          * PARITY FOR TRANSMITTER
                          *
        =00FE        164 PARITY  EQU  0FEH       ;PARITY (BIT 0) IS 'SPACE'
                          *
                          * SERIAL PORT VALUES
                          *
        =000B        168 BAUD    EQU  0BH        ;BAUD RATE = 9600
        =00A2        169 XMIT    EQU  0A2H       ;TRANSMIT COMMAND
        =00B0        170 RCV     EQU  0B0H       ;RECIEVE COMMAND
        =00B1        171 RCVI    EQU  0B1H       ;RECIEVE W/INTERUPT
                          *
                          * CLOCK/RAM VALUES
                          *
        =0000        175 CRCTRL  EQU  00H        ;CLK/RAM CONTROL BYTE
        =0002        176 CRCHIP  EQU  02H        ;CLK/RAM CHIP ENABLE BYTE
        =008F        177 RDSTAT  EQU  8FH        ;READ CLK/RAM STATUS
        =008E        178 WRSTAT  EQU  8EH        ;WRITE CLK/RAM STATUS
        =00BF        179 RDCLK   EQU  0BFH       ;READ CLOCK REGISTERS
        =00BE        180 WRCLK   EQU  0BEH       ;WRITE CLOCK REGISTERS
```

```
                          ******************
                          *                *
                          * INITIALIZATION *
                          *                *
                          ******************
                          *
                          * FUNCTION:
                          * THIS IS THE START OF THE DEMO PROGRAM. WHEN THE
                          * MICROCOMPUTER RESETS DUE TO POWER UP OR A HARDWARE
                          * (PUSH BUTTON) RESET, THIS CODE IS ENTERED. THE
                          * INITIALIZATION CONSISTS OF CLEARING ALL SCRATCH PAD
                          * REGISTERS, SETTING UP THE CHIP ENABLE PARAMETER,
                          * SETTING THE SERIAL PORT BAUD RATE AND PARITY,
                          * AND CHECKING IF THE CLOCK DATA IS VALID. IF IT IS
                          * NOT VALID, THE ROUTINE CONTINUE ON TO SET THE CLOCK.
                          * OTHERWISE, THE DATA IS ASSUMED OK.
                          *
                          * ENTRY STATUS:
                          * THE CPU HAS BEEN RESET.
                          *
                          * EXIT STATUS:
                          * IF THE CLOCK DATA IS VALID, THEN THE ROUTINE EXITS
                          * TO THE DATA OK ROUTINE. OTHERWISE, THE ROUTINE
                          * EXITS TO THE SET CLOCK ROUTINE.
                          *
                          * CLEAR SCRATCH PAD
                          *
0000                  209         ORG  0000H
0000 70               210         CLR               ;CLEAR ALL SCRATCH PAD
0001 0B               211 INIT    LR   IS,A         ;PUT POINTER INTO ISAR
0002 70               212         CLR               ;CLEAR THAT LOACTION
0003 5C               213         LR   S,A          ;
0004 0A               214         LR   A,IS         ;BUMP POINTER
0005 1F               215         INC               ;BUMP POINTER
0006 213F             216         NI   ISMASK       ;MASK TO 6 BITS
0008 94F8             217         BNZ  INIT         ;GO IF NOT DONE
                          *
                          * SET UP CLOCK/RAM SUBROUTINE PARAMETERS.
                          *
000A 2002             221         LI   CRCHIP       ;SET CLK/RAM CHIP ENABLE
000C 51               222         LR   CHIPEN,A     ;
                          *
                          * INITALIZE SERIAL PORT PARAMETERS.
                          *
000D 200B             226         LI   BAUD         ;SET SERIAL BAUD RATE
000F BC               227         OUTS RXCTRL       ;
0010 20FE             228         LI   PARITY       ;SET PARITY TO 'SPACE'
0012 BE               229         OUTS MSBYTE       ;
                          *
                          * CHECK IF CLOCK/RAM HAS JUST BEEN POWERED UP. IF SO,
                          * INITIALIZE AND SET THE CLOCK. IF NOT, THEN THE CLOCK
                          * DATA SHOULD BE VALID.
                          *
0013 2802AE           235         PI   STATRD       ;READ CLK/RAM STATUS
0016 47               236         LR   A,CTRL       ;CHECK WRITE PROTECT BIT
0017 F7               237         NS   CTRL         ;
0018 8169             238         BP   DATAOK       ;BRANCH IF DATA GOOD
```

```
                          *
                          * CLOCK/RAM JUST POWERED UP, SO INITALIZE IT.
                          *
001A 2802B6           242        PI   STATWR      ;WRITE CLK/RAM STATUS
001D 29006C           243        JMP  SETCLK      ;SET CLOCK
```

```
                                 ***********************************
                                 *                                 *
                                 * TIMER INTERRUPT SERVICE ROUTINE *
                                 *                                 *
                                 ***********************************
                                 *
                                 * FUNCTION:
                                 * THE TIMER INTERRUPT SERVICE ROUTINE IS ENTERED EVERY
                                 * TIME THE HARDWARE TIMER TIMES OUT (APPROXIMATELY
                                 * EVERY 1/36 SECONDS.) THE TIMER COUNTER IS
                                 * DECREMENTED TO DETERMINE IF 1 SECOND HAS PASSED
                                 * SINCE THE LAST SCREEN UPDATE. IF NOT, THE ROUTINE
                                 * TERMINATES. IF SO, NEW DATA IS READ FROM THE CLOCK/
                                 * RAM AND THE SCREEN IS UPDATED.
                                 *
                                 * ENTRY STATUS:
                                 * THE TIMER HAS TIMED OUT.
                                 *
                                 * EXIT STATUS:
                                 * IF 1 SECOND HAS NOT PASSED, THEN THE COUNTER IS
                                 * DECREMENTED. OTHERWISE, THE COUNTER IS RESET AND
                                 * THE NEW TIME IS READ FFROM THE CLOCK/RAM AND
                                 * PRINTED.
                                 *
020                         269            ORG   0020H
020 08                      270            LR    K,P          ;SAVE STACK
021 00                      271            LR    A,KU         ;
022 06                      272            LR    QU,A         ;
023 01                      273            LR    A,KL         ;
024 07                      274            LR    QL,A         ;
                                 *
                                 * CHECK IF 1 SECOND HAS PASSED SINCE LAST INTERRUPT.
                                 *
025 36                      278            DS    TIMCNT       ;DECREMENT COUNT
026 941C                    279            BNZ   FINISH       ;BRANCH IF NOT ZERO
                                 *
                                 * IT HAS, SO RESET COUNTER, READ NEW CLOCK DATA AND
                                 * DISPLAY IT.
                                 *
028 2024                    284            LI    MAXCNT       ;RESET COUNT
02A 56                      285            LR    TIMCNT,A
02B 2802C1                  286            PI    CLKRD        ;READ CLOCK REGISTERS
02E 2801A4                  287            PI    AMPMOT       ;PRINT AM/PM MESSAGE
031 2801C0                  288            PI    DAYOT        ;PRINT DAY
034 2801CC                  289            PI    DATEOT       ;PRINT DATE
037 2801F7                  290            PI    TIMEOT       ;PRINT TIME
03A 2A05EB                  291            DCI   HOME         ;SEND CURSOR HOME
03D 28029F                  292            PI    OUTMSG
                                 *
                                 * PUT SERIAL PORT BACK IN RECEIVE MODE AND RETURN
                                 * FROM INTERRUPT.
                                 *
40 20B1                     297            LI    RCVI         ;ENABLE RCV INTERUPT
42 BD                       298            OUTS  RXSTAT
43 1B                       299 FINISH     EI                 ;ENABLE INTERUPTS
44 0D                       300            LR    P0,Q         ;RETURN
```

```
                                  *************************************
                                  *                                   *
                                  * RECEIVER INTERRUPT SERVICE ROUTINE *
                                  *                                   *
                                  *************************************
                                  *
                                  * FUNCTION:
                                  * THE RECEIVER INTERRUPT SERVICE ROUTINE IS ENTERED
                                  * EVERY TIME A CHARACTER IS RECEIVED IN THE SERIAL
                                  * PORT. THE CHARACTER IS CHECKED FOR 'DC3' (CONTROL
                                  * S). IF NOT A 'DC3', THEN THE ROUTINE IS TERMINATED.
                                  * OTHERWISE, THE USER IS ALLOWED TO SET THE CLOCK
                                  * VALUES.
                                  *
                                  * ENTRY STATUS:
                                  * A CHARACTER HAS BEEN RECEIVED FROM THE KEYBOARD.
                                  *
                                  * EXIT STATUS:
                                  * IF THE CHARACTER WAS NOT A 'DC3', THEN A RETURN
                                  * FROM INTERRUPT IS DONE. OTHERWISE, THE ROUTINE
                                  * EXITS TO THE SET CLOCK ROUTINE.
                                  *
0060                        324          ORG  0060H
0060 08                     325          LR   K,P          ;SAVE STACK
0061 00                     326          LR   A,KU         ;
0062 06                     327          LR   QU,A         ;
0063 01                     328          LR   A,KL         ;
0064 07                     329          LR   QL,A         ;
                                  *
                                  * CHECK FOR 'DC3' FROM KEYBOARD. SET THE CLOCK IF
                                  * THIS KEY FOUND.
                                  *
0065 280287                 334          PI   INCHR2       ;GET CHARACTER
0068 2513                   335          CI   DC3          ;CHECK FOR 'DC3'
006A 94D8                   336          BNZ  FINISH       ;BRANCH IF NOT
                                  *
                                  * WAS 'DC3', SO FALL THROUGH TO SET CLOCK.
```

```
                                    *****************
                                    *               *
                                    * SET THE CLOCK *
                                    *               *
                                    *****************
                                    *
                                    * FUNCTION:
                                    * THIS ROUTINE ALLOWS THE USER TO SET THE CLOCK AND
                                    * CALENDAR SETTINGS.
                                    *
                                    * ENTRY STATUS:
                                    * EITHER THE CLOCK DATA WAS INVALID AT POWER UP OR
                                    * THE USER ENTERED A 'DC3' FROM THE KEYBOARD.
                                    *
                                    * EXIT STATUS:
                                    * ALL CLOCK/CALENDAR SETTINGS ARE SET.
                                    *
006C 68                         357 SETCLK  LISL SECOND.AND.7 ;POINT TO CLOCK BUFFER
006D 62                         358         LISU SECOND.SHR.3 ;
006E 2800AB                     359         PI   DAYIN        ;SET DAY OF WEEK
0071 280126                     360         PI   DATEIN       ;SET DATE IN CALENDAR
0074 280096                     361         PI   MODEIN       ;SET 12/24 HOUR MODE
0077 8104                       362         BP   SET1         ;BRANCH IS 24 HOUR MODE
0079 2800BC                     363         PI   AMPMIN       ;SET AM/PM FLAG
007C 2800D0                     364 SET1    PI   TIMEIN       ;SET TIME IN CLOCK
007F 2802C9                     365         PI   CLKWR        ;WRITE DATA TO CLOCK
                                    *
                                    * CLOCK NOW SET, SO FALL THROUGH TO START INTERRUPTS.
```

```
                               **************************
                               *                        *
                               * SET UP FOR INTERRUPTS *
                               *                        *
                               **************************
                               *
                               * FUNCTION:
                               * THIS ROUTINE INITIALIZES THE TIMER AND SERIAL PORT
                               * AND ENABLES INTERRUPTS.
                               *
                               * ENTRY STATUS:
                               * EITHER THE DATA WAS VALID AT POWER UP, OR THE CLOCK
                               * HAS JUST BEEN SET.
                               *
                               * EXIT STATUS:
                               * THE TIMER AND RECEIVER INTERRUPTS ARE THE ONLY EXIT.
                               *
0082 70                    386 DATAOK  CLR                 ;CLEAR TIMER
0083 B7                    387         OUTS TIMER          ;
0084 2024                  388         LI   MAXCNT         ;SET COUNTER
0086 56                    389         LR   TIMCNT,A       ;
0087 20EA                  390         LI   TMCTRL         ;SET TIMER CONTROL
0089 B6                    391         OUTS TICTRL         ;
008A 2A02DF                392         DCI  SIGNON         ;PRINT FEATURES
008D 28029F                393         PI   OUTMSG         ;
0090 20B1                  394         LI   RCVI           ;ENABLE RCV INTERUPT
0092 BD                    395         OUTS RXSTAT         ;
0093 1B                    396         EI                  ;ENABLE INTERUPTS
0094 90FF                  397 STOP    BR   STOP           ;WAIT FOR INTERUPT
```

```
                                  ***********************************
                                  *                                 *
                                  * 12/24 HOUR MODE INPUT SUBROUTINE *
                                  *                                 *
                                  ***********************************
                                  *
                                  * FUNCTION:
                                  * THIS SUBROUTINE ASKS THE USER IF THE MODE IS TO BE
                                  * 12 OR 24 HOUR FORMAT. THE ANSWER IS AQUIRED, AND THE
                                  * PROPER MODE IS SET.
                                  *
                                  * ENTRY STATUS:
                                  * NONE.
                                  *
                                  * EXIT STATUS:
                                  * THE MODE IS SET FOR 12 OR 24 HOUR OPERATION.
                                  *
0096 08                       416 MODEIN  LR   K,P          ;SAVE STACK
0097 00                       417         LR   A,KU         ;
0098 06                       418         LR   QU,A         ;
0099 01                       419         LR   A,KL         ;
009A 07                       420         LR   QL,A         ;
009B 2A0611                   421         DCI  MODMSG       ;PRINT MODE MESSAGE
009E 28029F                   422         PI   OUTMSG       ;
00A1 6A                       423         LISL HOUR.AND.7   ;POINT TO HOURS
00A2 280234                   424         PI   DIGIT2       ;GET DIGIT (0-1)
00A5 15                       425         SL   4            ;PUT INTO BIT 7
00A6 13                       426         SL   1            ;
00A7 13                       427         SL   1            ;
00A8 13                       428         SL   1            ;
00A9 5C                       429         LR   S,A          ;STORE IT AT HOURS
00AA 0D                       430         LR   P0,Q         ;RETURN
```

```
                          ************************
                          *                      *
                          * DAY INPUT SUBROUTINE *
                          *                      *
                          ************************
                          *
                          * FUNCTION:
                          * THIS SUBROUTINE ASKS THE USER FOR THE DAY AND
                          * INPUTS THE ANSWER.
                          *
                          * ENTRY STATUS:
                          * NONE.
                          *
                          * EXIT STATUS:
                          * THE DAY OF THE WEEK IS IN THE DAY BUFFER.
                          *
00AB 08               448 DAYIN   LR   K,P         ;SAVE STACK
00AC 00               449         LR   A,KU        ;
00AD 06               450         LR   QU,A        ;
00AE 01               451         LR   A,KL        ;
00AF 07               452         LR   QL,A        ;
00B0 2A05F0           453         DCI  DAYMSG      ;PRINT DAY MESSAGE
00B3 28029F           454         PI   OUTMSG      ;
00B6 6B               455         LISL DAY.AND.7   ;POINT TO DAY
00B7 280222           456         PI   DIGIT7      ;GET DIGIT (1-7)
00BA 5C               457         LR   S,A         ;STORE IT AT DAY
00BB 0D               458         LR   P0,Q        ;RETURN
```

```
                          *********************************
                          *                               *
                          * AM/PM SELECT INPUT SUBROUTINE *
                          *                               *
                          *********************************
                          *
                          * FUNCTION:
                          * THIS SUBROUTINE ASKS THE USER FOR THE AM OR PM
                          * SETTING. THE ANSWER IS AQUIRED AND THE PROPER MODE
                          * IS SET. THIS ROUTINE IS CALLED IN THE 12 HOUR
                          * MODE ONLY.
                          *
                          * ENTRY STATUS:
                          * NONE.
                          *
                          * EXIT STATUS:
                          * THE AM/PM FLAG IS SET OR RESET IN THE HOUR BUFFER.
                          *
00BC 08               478 AMPMIN  LR   K,P         ;SAVE STACK
00BD 00               479         LR   A,KU        ;
00BE 06               480         LR   QU,A        ;
00BF 01               481         LR   A,KL        ;
00C0 07               482         LR   QL,A        ;
00C1 2A0631           483         DCI  AMPMSG      ;PRINT AM/PM MESSAGE
00C4 28029F           484         PI   OUTMSG      ;
00C7 6A               485         LISL HOUR.AND.7  ;POINT TO HOURS
00C8 280234           486         PI   DIGIT2      ;GET DIGIT (0-1)
00CB 15               487         SL   4           ;PUT INTO BIT 5
00CC 13               488         SL   1           ;
00CD EC               489         XS   S           ;
00CE 5C               490         LR   S,A         ;STORE IT AT HOURS
00CF 0D               491         LR   P0,Q        ;RETURN
```

```
                          *************************
                          *                       *
                          * TIME INPUT SUBROUTINE *
                          *                       *
                          *************************
                          *
                          * FUNCTION:
                          * THIS SUBROUTINE ASKS THE USER FOR THE TIME. IT
                          * INPUTS THE TIME AND SETS THE CLOCK UP ACCORDINGLY.
                          * THE TIME IS INFUT IN THE HR:MIN:SEC FORMAT. LEADING
                          * ZEROS MUST BE INPUT.
                          *
                          * ENTRY STATUS:
                          * NONE.
                          *
                          * EXIT STATUS:
                          * THE TIME OF DAY IS SET IN THE HOUR, MINUTE, AND
                          * SECOND BUFFER.
                          *
00D0 08               512 TIMEIN  LR   K,P         ;SAVE STACK
00D1 00               513         LR   A,KU        ;
00D2 06               514         LR   QU,A        ;
00D3 01               515         LR   A,KL        ;
00D4 07               516         LR   QL,A        ;
00D5 2A0648           517         DCI  TIMMSG      ;PRINT TIME MESSAGE
00D8 28029F           518         PI   OUTMSG      ;
                          *
                          * CHECK IF 12 OR 24 HOUR MODE.
                          *
00DB 6A               522         LISL HOUR.AND.7  ;POINT TO HOURS
00DC 4C               523         LR   A,S         ;CHECK IF 24 HOUR MODE
00DD FC               524         NS   S           ;
00DE 8115             525         BP   HOUR24      ;BRANCH IF SO
                          *
                          * 12 HOUR MODE, SO VALID HOURS ARE 01-12.
                          *
00E0 280234           529         PI   DIGIT2      GET CIGIT (0-1)
00E3 840B             530         BZ   HOUR0X      ;BRANCH IF 0 ENTERED
00E5 15               531         SL   4           ;STORE IT AT TENS
00E6 EC               532         XS   S           ;
00E7 5C               533         LR   S,A         ;
00E8 280239           534         PI   DIGIT3      ;GET DIGIT (0-2)
00EB EC               535 HOUR1   XS   S           ;STORE IT AT UNITS
00EC 5C               536         LR   S,A         ;
00ED 901B             537         BR   MIN         ;GO TO MINUTES
00EF 28022A           538 HOUR0X  PI   DIGIT9      ;GET DIGIT (1-9)
00F2 90F8             539         BR   HOUR1       ;STORE IT AND CONTINUE
                          *
                          * 24 HOUR MODE, SC VALID HOURS ARE 00-23.
                          *
00F4 280239           543 HOUR24  PI   DIGIT3      ;GET DIGIT (0-2)
00F7 15               544         SL   4           ;STORE IT AT TENS
00F8 5C               545         LR   S,A         ;
00F9 2520             546         CI   TWC.SHL.4   ;SEE IF DIGIT WAS '2'
00FB 8408             547         BZ   HOUR2X      ;BRANCH IF SO
00FD 28024D           548         PI   DIGIT0      ;GET DIGIT (0-9)
0100 EC               549 HOUR2   XS   S           ;STORE IT AT UNITS
```

```
0101 5C                550            LR    S,A            ;
0102 9006              551            BR    MIN            ;GO TO MINUTES
0104 28023E            552 HOUR2X     PI    DIGIT4         ;GET DIGIT (0-3)
0107 90F8              553            BR    HOUR2          ;STORE AND CONTINUE
                           *
                           * VALID MINUTES ARE 00-59.
                           *
0109 28026A            557 MIN        PI    OUTCOL         ;PRINT COLON SEPARATOR
010C 69                558            LISL  MINUTE.AND.7 ;POINT TO MINUTES
010D 280243            559            PI    DIGIT6         ;GET DIGIT (0-5)
0110 15                560            SL    4              ;STORE IT AT TENS
0111 5C                561            LR    S,A            ;
0112 28024D            562            PI    DIGIT0         ;GET DIGIT (0-9)
0115 EC                563            XS    S              ;STORE IT AT UNITS
0116 5C                564            LR    S,A            ;
                           *
                           * VALID SECONDS ARE 00-59
                           *
0117 28026A            568            PI    OUTCOL         ;PRINT COLON SEPARATOR
011A 68                569            LISL  SECOND.AND.7 ;POINT TO SECONDS.
011B 280243            570            PI    DIGIT6         ;GET DIGIT (0-5)
011E 15                571            SL    4              ;STORE IT AT TENS
011F 5C                572            LR    S,A            ;
0120 28024D            573            PI    DIGIT0         ;GET DIGIT (0-9)
0123 EC                574            XS    S              ;STORE IT AT UNITS
0124 5C                575            LR    S,A            ;
0125 0D                576            LR    P0,Q           ;RETURN
```

```
                            *************************
                            *                       *
                            * DATE INPUT SUBROUTINE *
                            *                       *
                            *************************
                            *
                            * FUNCTION:
                            * THIS SUBROUTINE ASKS THE USER FOR THE DATE. IT
                            * INPUTS THE DATE AND SETS THE CALENDAR ACCORDINGLY.
                            * THE DATE IS INPUT IN THE YR:MNTH:DAY FORMAT.
                            * LEADING ZEROS MUST BE INPUT.
                            *
                            * ENTRY STATUS:
                            * NONE.
                            *
                            * EXIT STATUS:
                            * THE DATE IS IN THE YEAR, MONTH, AND DATE BUFFER.
                            *
0126 08                  596 DATEIN  LR   K,P           ;SAVE STACK
0127 00                  597         LR   A,KU          ;
0128 06                  598         LR   QU,A          ;
0129 01                  599         LR   A,KL          ;
012A 07                  600         LR   QL,A          ;
012B 2A05FD              601         DCI  DATMSG        ;PRINT DATE MESSAGE
012E 28029F              602         PI   OUTMSG        ;
                            *
                            * VALID YEARS ARE 00-99.
                            *
0131 6E                  606         LISL YEAR.AND.7    ;POINT TO YEAR
0132 28024D              607         PI   DIGITO        ;GET DIGIT (0-9)
0135 15                  608         SL   4             ;STORE IT AT TENS
0136 5C                  609         LR   S,A           ;
0137 28024D              610         PI   DIGITO        ;GET DIGIT (0-9)
013A EC                  611         XS   S             ;STORE IT AT UNITS
013B 5C                  612         LR   S,A           ;
                            *
                            * VALID MONTHS ARE 01-12.
                            *
013C 28026A              616         PI   OUTCOL        ;PRINT COLON SEPARATER
013F 6D                  617         LISL MONTH.AND.7   ;POINT TO MONTH
0140 280234              618         PI   DIGIT2        ;GET DIGIT (0-1)
0143 15                  619         SL   4             ;STORE IT AT TENS
0144 5C                  620         LR   S,A           ;
0145 8408                621         BZ   MNTHOX        ;BRANCH IF DIGIT IS '0'
0147 280239              622         PI   DIGIT3        ;GET DIGIT (0-2)
014A EC                  623 MONTH1  XS   S             ;STORE IT AT UNITS
014B 5C                  624         LR   S,A           ;
014C 9006                625         BR   DDATE         ;GO TO DATE
014E 28022A              626 MNTHOX  PI   DIGIT9        ;GET DIGIT (1-9)
0151 90F8                627         BR   MONTH1        ;STORE AND CONTINUE
                            *
                            * CHECK MONTH. IF MONTH IS FEBURARY, ALLOW 28 OR
                            * 29 DAYS IN THE MONTH. IF MONTH IS APRIL, JUNE,
                            * SEPTEMBER OR NOVEMBER, ALLOW 30 DAYS IN THE
                            * MONTH. FOR OTHER MONTHS, ALLOW 31 DAYS.
                            *
0153 28026A              634 DDATE   PI   OUTCOL        ;PRINT COLON SEPARATOR
```

```
0156 4E            635          LR   A,D          ;GET MONTH, POINT DATE
0157 2502          636          CI   FEB          ;CHECK IF 'FEBRUARY'
0159 842F          637          BZ   FEBXX        ;BRANCH IF SO
                       *
                       * NOT FEBRUARY, SO ALLOW 30 OR 31 DAYS.
                       *
015B 28023E        641          PI   DIGIT4       ;GET DIGIT (0-3)
015E 15            642          SL   4            ;STORE IT AT TENS
015F 5C            643          LR   S,A          ;
0160 840B          644          BZ   DAY0X        ;BRANCH IF DIGIT WAS 0
0162 2530          645          CI   THREE.SHL.4  ;CHECK IF DIGIT WAS '3'
0164 840C          646          BZ   DAY3X        ;BRANCH IF SO
0166 28024D        647 DDATE3   PI   DIGIT0       ;GET DIGIT (0-9)
0169 EC            648 DDATE1   XS   S            ;STORE IT AT UNITS
016A 5C            649          LR   S,A          ;
016B 0D            650          LR   P0,Q         ;RETURN
016C 28022A        651 DAY0X    PI   DIGIT9       ;GET DIGIT (1-9)
016F 90F9          652          BR   DDATE1       ;STORE AND RETURN
                       *
                       * CHECK FOR APRIL, JUNE, SEPTEMBER AND NOVEMBER.
                       *
0171 6D            656 DAY3X    LISL MONTH.AND.7  ;POINT TO MONTH
0172 2004          657          LI   4            ;LOOP COUNT = 4
0174 55            658          LR   DCOUNT,A     ;
0175 4E            659          LR   A,D          ;GET MONTH, POINT DATE
0176 2A02DB        660          DCI  TAB30        ;POINT TO 30-DAY TABLE
0179 8D            661 DLOOP    CM                ;CHECK IF IN TABLE
017A 8409          662          BZ   DAY30        ;BRANCH IF SO
017C 35            663          DS   DCOUNT       ;DECREMENT COUNT
017D 94FB          664          BNZ  DLOOP        ;BRANCH IF NOT DONE
017F 280234        665          PI   DIGIT2       ;GET DIGIT (0-1)
                       *
                       * 31 DAY MONTH, SO ALLOW DAYS OF 01-31.
                       *
0182 90E6          669          BR   DDATE1       ;STORE AND RETURN
                       *
                       * 30 DAY MONTH, SC ALLOW DAYS OF 01-30.
                       *
0184 28022F        673 DAY30    PI   DIGIT1       ;GET DIGIT (0)
0187 90E1          674          BR   DDATE1       ;STORE AND RETURN
                       *
                       * FEBRUARY, SO ALLOW 28 OR 29 DAYS.
                       *
0189 280239        678 FEBXX    PI   DIGIT3       ;GET DIGIT (0-2)
018C 15            679          SL   4            ;STORE IT AT TENS
018D 5C            680          LR   S,A          ;
018E 84DD          681          BZ   DAY0X        ;BRANCH IF DIGIT WAS 0
0190 2520          682          CI   TWO.SHL.4    ;CHECK IF DIGIT WAS '2'
0192 94D3          683          BNZ  DDATE3       ;BRANCH IF NOT
                       *
                       * CHECK IF IT IS A LEAP YEAR.
                       *
0194 6E            687          LISL YEAR.AND.7   ;POINT TO YEAR
0195 4C            688          LR   A,S          ;GET YEAR
0196 6C            689          LISL DATE.AND.7   ;POINT TO DATE
0197 2113          690          NI   LEAP1        ;CHECK IF LEAP YEAR
0199 84CC          691          BZ   DDATE3       ;BRANCH IF IT IS
```

```
019B 2312              692          XI    LEAP2        ;CHECK AGAIN
019D 84C8              693          BZ    DDATE3       ;BRANCH IF IT IS
                           *
                           * NOT A LEAP YEAR, SC ALLOW DAYS OF 01-28.
                           *
019F 280248            697          PI    DIGIT8       ;GET DIGIT (0-8)
01A2 90C6              698          BR    DDATE1       ;STORE AND RETURN
```

```
                               ***************************
                               *                         *
                               * AM/PM PRINT SUBROUTINE  *
                               *                         *
                               ***************************
                               *
                               * FUNCTION:
                               * THIS SUBROUTINE PRINTS THE MESSAGE 'GOOD MORNING'
                               * IF THE AM/PM BIT IS CLEAR, OR 'GOOD AFTERNOOD' IF
                               * THE AM/PM BIT IS SET.
                               *
                               * ENTRY STATUS:
                               * THE MODE AND AM/PM BITS MUST BE IN THE HOUR BUFFER.
                               *
                               * EXIT STATUS:
                               * IF THE MODE IS 12 HOUR, THEN THE 'GOOD MORNING' OR
                               * 'GOOD AFTERNOON' MESSAGE WAS PRINTED (DEPENDING ON
                               * THE STATUS OF THE AM/PM BIT.) OTHERWISE, THE FIRST
                               * LINE OF THE CRT WAS BLANKED.
                               *
01A4 08                    720 AMPMOT  LR   K,P          ;SAVE STACK
01A5 2A0512                721         DCI  GOODPT       ;CURSOR TO LINE 1
01A8 28029F                722         PI   OUTMSG       ;
                               *
                               * CHECK IF IN 12 OR 24 HOUR MODE. SKIP THIS
                               * ROUTINE IF 24 HOUR MODE.
                               *
01AB 6A                    727         LISL HOUR.AND.7   ;POINT TO HOURS
01AC 4C                    728         LR   A,S          ;CHECK 12/24 HOUR BIT
01AD FC                    729         NS   S            ;SET FLAGS
01AE 8110                  730         BP   MLTRY1       ;BRANCH IF 24 HOUR
                               *
                               * 12 HOUR MODE, SO CHECK AM/PM FLAG. PRINT 'GOOD
                               * MORNING' IF AM, 'GOOD AFTERNOON' IF PM.
                               *
01B0 13                    735         SL   1            ;CHECK AM/PM FLAG
01B1 13                    736         SL   1
01B2 8106                  737         BP   AMPM1        ;BRANCH IF AM
01B4 2A0527                738         DCI  GDAFTR       ;POINT TO PM MSG
01B7 9004                  739         BR   AMPM2        ;CONTINUE
01B9 2A0519                740 AMPM1   DCI  GDMORN       ;POINT TO AM MSG
01BC 28029F                741 AMPM2   PI   OUTMSG       ;PRINT MESSAGE
01BF 0C                    742 MLTRY1  PK                ;RETURN
```

```
                                *************************
                                *                       *
                                * DAY PRINT SUBROUTINE  *
                                *                       *
                                *************************
                                *
                                * FUNCTION:
                                * THIS SUBROUTINE PRINTS THE DAY.
                                *
                                * ENTRY STATUS:
                                * THE DAY OF THE WEEK MUST BE IN THE DAY BUFFER.
                                *
                                * EXIT STATUS:
                                * THE DAY IS PRINTED ON THE CRT.
                                *
01C0 08                     759 DAYOT    LR   K,P            ;SAVE STACK
01C1 2A0537                 760          DCI  DAYPT          ;CURSOR TO LINE 3
01C4 28029F                 761          PI   OUTMSG         ;
01C7 6B                     762          LISL DAY.AND.7      ;POINT TO DAY
01C8 280295                 763          PI   FNDOUT         ;PRINT DAY MESSAGE
01CB 0C                     764          PK                  ;RETURN
```

```
                                ***************************
                                *                         *
                                * DATE PRINT SUBROUTINE *
                                *                         *
                                ***************************
                                *
                                * FUNCTION:
                                * THIS SUBROUTINE PRINTS THE DATE.
                                *
                                * ENTRY STATUS:
                                * THE DATE MUST BE IN THE YEAR, MONTH, AND DATE
                                * BUFFERS.
                                *
                                * EXIT STATUS:
                                * THE DATE IS PRINTED ON THE CRT.
                                *
01CC 08                     782 DATEOT   LR   K,P          ;SAVE STACK
01CD 2A0576                 783          DCI  DATEPT       ;CURSOR TO LINE 5
01D0 28029F                 784          PI   OUTMSG       ;
01D3 6D                     785          LISL MONTH.AND.7  ;POINT TO MONTH
                                *
                                * MAKE BCD MONTH BINARY.
                                *
01D4 4C                     789          LR   A,S          ;GET MONTH
01D5 2110                   790          NI   TENBCD       ;SEE IF MONTH > 9
01D7 4C                     791          LR   A,S          ;RECALL MONTH
01D8 8405                   792          BZ   DATE1        ;BRANCH IF <= 9
01DA 210F                   793          NI   MNLSD        ;KEEP ONLY LSD
01DC 240A                   794          AI   TEN          ;ADD 10
01DE 5C                     795 DATE1    LR   S,A          ;PUT IT ALL BACK
                                *
                                * FIND MONTH IN MESSAGE AREA AND PRINT IT.
                                * THEN PRINT DATE AND YEAR.
                                *
01DF 280295                 800          PI   FNDOUT       ;PRINT MONTH
01E2 6C                     801          LISL DATE.AND.7   ;POINT TO DATE
01E3 280278                 802          PI   OUTMSD       ;PRINT DATE
01E6 28027E                 803          PI   OUTLSD       ;
01E9 2A05DF                 804          DCI  SEPAR        ;PRINT SEPARATER
01EC 28029F                 805          PI   OUTMSG       ;
01EF 6E                     806          LISL YEAR.AND.7   ;POINT TO YEAR
01F0 280278                 807          PI   OUTMSD       ;PRINT YEAR
01F3 28027E                 808          PI   OUTLSD       ;
01F6 0C                     809          PK                ;RETURN
```

VI
Z80
MICRO-
COMPUTER
APPLICATION
NOTES

```
                              *************************
                              *                       *
                              * TIME PRINT SUBROUTINE *
                              *                       *
                              *************************
                              *
                              * FUNCTION:
                              * THIS SUBROUTINE PRINTS THE TIME.
                              *
                              * ENTRY STATUS:
                              * THE TIME MUST BE IN THE HOUR, MINUTE, AND SECOND
                              * BUFFERS.
                              *
                              * EXIT STATUS:
                              * THE TIME IS PRINTED ON THE CRT.
                              *
01F7 08                   827 TIMEOT  LR   K,P          ;SAVE STACK
01F8 2A05E4               828         DCI  TIMEPT       ;CURSOR TO LINE 7
01FB 28029F               829         PI   OUTMSG       ;
                              *
                              * CHECK IF 12 OR 24 HOUR MODE. FOR 12 HOUR MODE,
                              * FLAGS MUST BE STRIPPED FROM HOURS BYTE.
                              *
01FE 6A                   834         LISL HOUR.AND.7   ;POINT TO HOURS
01FF 4C                   835         LR   A,S          ;CHECK 12/24 HOUR BIT
0200 FC                   836         NS   S            ;
0201 8105                 837         BP   MLTRY2       ;BRANCH IF 24 HOUR
0203 4C                   838         LR   A,S          ;STRIP FLAGS FROM HOURS
0204 211F                 839         NI   HR1MSD+HRLSD ;
0206 5C                   840         LR   S,A          ;
                              *
                              * PRINT HOURS, MINUTES AND SECONDS.
                              *
0207 280278               844 MLTRY2  PI   OUTMSD       ;PRINT HOURS
020A 28027E               845         PI   OUTLSD       ;
020D 28026A               846         PI   OUTCOL       ;PRINT COLON
0210 69                   847         LISL MINUTE.AND.7 ;POINT TO MINUTES
0211 280278               848         PI   OUTMSD       ;PRINT MINUTES
0214 28027E               849         PI   OUTLSD       ;
0217 28026A               850         PI   OUTCOL       ;PRINT COLON
021A 68                   851         LISL SECOND.AND.7 ;POINT TO SECONDS
021B 280278               852         PI   OUTMSD       ;PRINT SECONDS
021E 28027E               853         PI   OUTLSD       ;
0221 0C                   854         PK                ;RETURN
```

```
                                  ************************
                                  *                      *
                                  * GET DIGIT SUBRCUTINE *
                                  *                      *
                                  ************************
                                  *
                                  * FUNCTION:
                                  * THIS SUBROUTINE, WITH IT'S VARIOUS ENTRY POINTS,
                                  * GETS A DIGIT FROM THE KEYBOARD AND ECHOS IT TO THE
                                  * CRT.
                                  *
                                  * ENTRY STATUS:
                                  * THE RANGE IS SPECIFIED BY A CALL TO THE APPROPRIATE
                                  * ENTRY PCINT.
                                  *
                                  * NORMAL EXIT STATUS:
                                  * THE DIGIT IS ECHOED TO THE CRT, AND RETURNED
                                  * IN A AS A BINARY VALUE.
                                  *
                                  * ERROR EXIT STATUS:
                                  * THE CHARACTER IS NOT ECHOED TO THE CRT, AND THE
                                  * ROUTINE LOOPS BACK FOR ANOTHER CHARACTER UNTIL
                                  * A CHARACTER THAT IS IN THE RANGE IS INPUT.
                                  *
                                  * GET DIGIT (1-7) SUBROUTINE
                                  *
0222 08                       882 DIGIT7  LR   K,P         ;SAVE STACK
0223 2007                     883         LI   SEVEN       ;COUNT = 7
0225 2A02D2                   884 DGT     DCI  TAB19       ;POINT TO SECOND TABLE
0228 902A                     885         BR   DIGITT      ;GO GET A DIGIT
                                  *
                                  * GET DIGIT (1-9) SUBROUTINE
                                  *
022A 08                       889 DIGIT9  LR   K,P         ;SAVE STACK
022B 2009                     890         LI   NINE        ;COUNT = 9
022D 90F7                     891         BR   DGT         ;GO GET A DIGIT
                                  *
                                  * GET DIGIT (0) SUBROUTINE
                                  *
022F 08                       895 DIGIT1  LR   K,P         ;SAVE STACK
0230 2001                     896         LI   ONE         ;COUNT = 1
0232 901D                     897         BR   DIGIT       ;GO GET A DIGIT
                                  *
                                  * GET DIGIT (0-1) SUBROUTINE
                                  *
0234 08                       901 DIGIT2  LR   K,P         ;SAVE STACK
0235 2002                     902         LI   TWO         ;COUNT = 2
0237 9018                     903         BR   DIGIT       ;GO GET A DIGIT
                                  *
                                  * GET DIGIT (0-2) SUBROUTINE
                                  *
0239 08                       907 DIGIT3  LR   K,P         ;SAVE STACK
023A 2003                     908         LI   THREE       ;COUNT = 3
023C 9013                     909         BR   DIGIT       ;GO GET A DIGIT
                                  *
                                  * GET DIGIT (0-3) SUBROUTINE
                                  *
```

```
0023E 08                913 DIGIT4  LR   K,P         ;SAVE STACK
```

```
023F 2004               914         LI   FOUR        ;COUNT = 4
0241 900E               915         BR   DIGIT       ;GO GET A DIGIT
                            *
                            * GET DIGIT (0-5) SUBROUTINE
                            *
0243 08                 919 DIGIT6  LR   K,P         ;SAVE STACK
0244 2006               920         LI   SIX         ;COUNT = 6
0246 9009               921         BR   DIGIT       ;GO GET A DIGIT
                            *
                            * GET DIGIT (0-8) SUBROUTINE
                            *
0248 08                 925 DIGIT8  LR   K,P         ;SAVE STACK
0249 2009               926         LI   NINE        ;COUNT = 9
024B 9004               927         BR   DIGIT       ;GO GET A DIGIT
                            *
                            * GET DIGIT (0-9)
                            *
024D 08                 931 DIGIT0  LR   K,P         ;SAVE STACK
024E 200A               932         LI   TEN         ;COUNT = 10
0250 2A02D1             933 DIGIT   DCI  TAB09       ;POINT TO 0-9 TABLE
                            *
                            * SAVE COUNT AND POINTER IN CASE A CHARACTER IS
                            * ENTERED WHICH IS NOT WITHIN RANGE.
                            *
0253 54                 938 DIGITT  LR   CNTSAV,A    ;SAVE COUNT FOR ERROR
0254 11                 939         LR   H,DC        ;SAVE POINTER FOR ERROR
0255 55                 940 DGTBAD  LR   DCOUNT,A    ;SAVE COUNT
0256 10                 941         LR   DC,H        ;POINT TO TABLE
0257 280283             942         PI   INCHR       ;GET A CHARACTER
025A 8D                 943 DGTLOP  CM               ;SEE IF IT IS IN TABLE
025B 8407               944         BZ   DGTOK       ;BRANCH IF IT IS
025D 35                 945         DS   DCOUNT      ;DECREMENT COUNT
025E 94FB               946         BNZ  DGTLOP      ;BRANCH IF NOT DONE
0260 44                 947         LR   A,CNTSAV    ;RESET COUNTER
0261 90F3               948         BR   DGTBAD      ;TRY AGAIN
                            *
                            * GOT A VALID CHARACTER, SO ECHO IT TO SCREEN
                            * AND MAKE IT BINARY.
                            *
0263 28026D             953 DGTOK   PI   OUTCHR      ;ECHO CHARACTER
0266 43                 954         LR   A,TEMP      ;MAKE IT BCD
0267 210F               955         NI   LSD         ;
0269 0C                 956         PK               ;RETURN
```

```
                          *******************************
                          *                             *
                          * CHARACTER OUTPUT SUBROUTINE *
                          *                             *
                          *******************************
                          *
                          * FUNCTION:
                          * THIS SUBROUTINE, WITH IT'S VARIOUS ENTRY POINTS,
                          * OUTPUTS THE SPECIFIED CHARACTER TO THE CRT.
                          *
                          * ENTRY STATUS:
                          * THE CHARACTER TO BE OUTPUT IS DETERMINED BY THE
                          * ENTRY POINT TO THE ROUTINE.
                          *
                          * EXIT STATUS:
                          * THE CHARACTER IS OUTPUT TO THE CRT.
                          *
                          * OUTPUT COLON SUBROUTINE
                          *
026A 203A             977 OUTCOL  LI    ':'         ;LOAD COLON INTO TEMP
026C 53               978         LR    TEMP,A
                          *
                          * CHARACTER OUTPUT SUBROUTINE
                          *
                          * THE CHARACTER IS OUTPUT FROM REGISTER TEMP.
                          *
026D 20A2             984 OUTCHR  LI    XMIT        ;PUT INTO XMIT MODE
026F BD               985         OUTS RXSTAT       ;
0270 43               986         LR    A,TEMP      ;GET CHARACTER
0271 13               987         SL    1           ;START BIT = 0
0272 BF               988         OUTS LSBYTE       ;SEND IT
0273 AD               989 LOOP1   INS   RXSTAT      ;WAIT TILL IT'S SENT
0274 13               990         SL    1           ;
0275 81FD             991         BP    LOOP1       ;
0277 1C               992         POP               ;RETURN
                          *
                          * OUTPUT MOST SIGNIFICANT DIGIT SUBROUTINE
                          *
                          * THE DIGIT IS OUTPUT FROM BITS 7-4 OF THE BYTE AT
                          * THE LOCATION POINTED TO BY ISAR.
                          *
0278 4C               999 OUTMSD  LR    A,S         ;GET MSD
0279 14              1000         SR    4           ;
027A 2230            1001 ASCII   OI    030H        ;MAKE IT ASCII
027C 90EF            1002         BR    OUTCOL+2    ;SEND IT OUT
                          *
                          * OUTPUT LEAST SIGNIFICANT DIGIT SUBROUTINE
                          *
                          * THE DIGIT IS OUTPUT FROM BITS 3-0 OF THE BYTE AT
                          * THE LOCATION POINTED TO BY ISAR.
                          *
027E 4C              1009 OUTLSD  LR    A,S         ;GET LSD
027F 210F            1010         NI    LSD         ;
0281 90F8            1011         BR    ASCII       ;MAKE IT ASCII AND PRINT
```

```
                               ******************************
                               *                            *
                               * CHARACTER INPUT SUBROUTINE *
                               *                            *
                               ******************************
                               *
                               * FUNCTION:
                               * THIS SUBROUTINE INPUTS A CHARACTER FROM THE
                               * KEYBOARD.
                               *
                               * ENTRY STATUS:
                               * NONE.
                               *
                               * EXIT STATUS:
                               * THE CHARACTER IS RETURNED IN A IN ASCII FORMAT.
                               *
0283 20B0                 1029 INCHR    LI   RCV          ;PUT INTO RCV MODE
0285 BD                   1030          OUTS RXSTAT       ;
0286 AF                   1031          INS  LSBYTE       ;CLEAR READY BIT
0287 AD                   1032 INCHR2   INS  RXSTAT       ;WAIT TILL INPUT READY
0288 81FE                 1033          BP   INCHR2       ;
028A AF                   1034          INS  LSBYTE       ;GET BITS 1 AND 0
028B 14                   1035          SR   4            ;
028C 12                   1036          SR   1            ;
028D 12                   1037          SR   1            ;
028E 53                   1038          LR   TEMP,A       ;SAVE THEM
028F AE                   1039          INS  MSBYTE       ;GET BITS 7 THRU 2
0290 13                   1040          SL   1            ;
0291 13                   1041          SL   1            ;
0292 E3                   1042          XS   TEMP         ;MIX BITS INTO BYTE
0293 53                   1043          LR   TEMP,A       ;SAVE INPUT
0294 1C                   1044          POP               ;RETURN
```

```
                               ****************************
                               *                          *
                               * PRINT MESSAGE SUBROUTINE *
                               *                          *
                               ****************************
                               *
                               * FUNCTION:
                               * THIS SUBROUTINE PRINTS THE MESSAGE WHOSE NUMBER I
                               * IN THE LOCATION AT THE POINTER.
                               *
                               * ENTRY STATUS:
                               * ISAR MUST POINT TO THE LOCATION CONTAINING THE
                               * NUMBER OF THE MESSAGE TO BE PRINTED. (TYPICALLY
                               * THE NUMBER OF THE MONTH OR DAY.) DC MUST POINT TO
                               * THE START OF THE STRING OF MESSAGES. EACH MESSAGE
                               * MUST END WITH AN 'EOT' CHARACTER.
                               *
                               * EXIT STATUS:
                               * THE APPROPRIATE MESSAGE WAS PRINTED ON THE CRT.
                               *
0295 3C                   1066 FNDOUT  DS   S             ;DECREMENT MSG COUNT
0296 8408                 1067         BZ   OUTMSG        ;BRANCH IF FOUND
0298 16                   1068 FNDLOP  LM                 ;GET CHARACTER
0299 2504                 1069         CI   EOT           ;CHECK FOR END OF TEXT
029B 94FC                 1070         BNZ  FNDLOP        ;BRANCH IF NOT FOUND
029D 90F7                 1071         BR   FNDOUT        ;ELSE, CHECK COUNT
                               *
                               * MESSAGE LOCATED, SO FALL THROUGH TO PRINT IT.
```

```
                            ****************************
                            *                          *
                            * MESSAGE OUTPUT SUBROUTINE *
                            *                          *
                            ****************************
                            *
                            * FUNCTION:
                            * THIS SUBROUTINE PRINTS THE MESSAGE STARTING AT TH
                            * POINTER.
                            *
                            * ENTRY STATUS:
                            * DC MUST POINT TO THE START OF THE MESSAGE TO BE
                            * PRINTED. IT MUST END WITH AN 'EOT' CHARACTER.
                            *
                            * EXIT STATUS:
                            * THE MESSAGE IS PRINTED ON THE CRT.
                            *
029F 20A2              1092 OUTMSG   LI   XMIT        ;PUT INTO XMIT MODE
02A1 8D                1093          CUTS RXSTAT
02A2 16                1094 LOOP3    LM               ;GET CHARACTER
02A3 2504              1095          CI   EOT         ;CHECK FOR END OF TEXT
02A5 84CD              1096          BZ   LOCP1       ;BRANCH IF END
02A7 13                1097          SL   1           ;START BIT = 0
02A8 BF                1098          OUTS LSBYTE      ;SENT CHARACTER
02A9 AD                1099 LOOP4    INS  RXSTAT      ;WAIT TILL READY FOR NEXT
02AA 81FE              1100          BP   LOCP4       ;BRANCH IF NOT READY
02AC 90F5              1101          BR   LOCP3       ;NEXT CHARACTER
```

```
                               ******************************
                               *                            *
                               * CLOCK/RAM SUBROUTINE PATCHES *
                               *                            *
                               ******************************
                               *
                               * FUNCTION:
                               * THESE PATCHES ARE TO SET UP THE COMMAND REGISTER
                               * FOR THE CLOCK/RAM SUBROUTINE. THE DIFFERENT ENTRY
                               * POINTS SET UP DIFFERENT COMMANDS.
                               *
                               * ENTRY STATUS:
                               * FOR WRITE COMMANDS, THE DATA MUST BE IN THE CLOCK
                               * BUFFER AREAS.
                               *
                               *
                               * EXIT STATUS:
                               * THE DATA IS TRANSFERED BETWEEN SCRATCH PAD AND TH
                               * CLOCK/RAM.
                               * READ CLOCK/RAM STATUS SUBROUTINE
                               *
                               * READ CLOCK/RAM STATUS SUBRUTINE
                               *
02AE 60                   1126 STATRD   LISU CTRL.SHR.3   ;POINT TO CTRL REG
02AF 6F                   1127          LISL CTRL.AND.7   ;
02B0 208F                 1128          LI   RDSTAT       ;SET UP COMMAND
02B2 52                   1129          LR   CMD,A        ;
02B3 29FFFF               1130          JMP  CLKRAM       ;EXECUTE IT
                               *
                               * WRITE CLOCK/RAM STATUS SUBROUTINE
                               *
02B6 60                   1134 STATWR   LISU CTRL.SHR.3   ;POINT TO CTRL REG
02B7 6F                   1135          LISL CTRL.AND.7   ;
02B8 208E                 1136          LI   WRSTAT       ;SET UP COMMAND
02BA 52                   1137          LR   CMD,A        ;
02BB 2000                 1138          LI   CRCTRL       ;SET UP CONTROL BYTE
02BD 57                   1139          LR   CTRL,A       ;
02BE 2902B4               1140          JMP  CLKRAM       ;EXECUTE IT
                               *
                               * READ CLOCK SUBROUTINE
                               *
02C1 62                   1144 CLKRD    LISU SECOND.SHR.3 ;POINT TO CLOCK BUFFER
02C2 68                   1145          LISL SECOND.AND.7 ;
02C3 20BF                 1146          LI   RDCLK        ;SET UP COMMAND
02C5 52                   1147          LR   CMD,A        ;
02C6 2902BF               1148          JMP  CLKRAM       ;EXECUTE IT
                               *
                               * WRITE CLOCK SUBROUTINE
                               *
02C9 62                   1152 CLKWR    LISU SECOND.SHR.3 ;POINT TO CLOCK BUFFER
02CA 68                   1153          LISL SECOND.AND.7 ;
02CB 20BE                 1154          LI   WRCLK        ;SET UP COMMAND
02CD 52                   1155          LR   CMD,A        ;
02CE 2902C7               1156          JMP  CLKRAM       ;EXECUTE IT
```

```
                             *****************
                             *               *
                             * PROGRAM TABLES *
                             *               *
                             *****************
                             *
                             * DIGIT CHECK TABLE
                             *
02D1 30              1166 TAB09   DEFB '0'
02D2 31323334        1167 TAB19   DEFB '1','2','3','4','5','6','7','8','9'
     35363738
     39
                             *
                             * TABLE OF 30 DAY MONTHS
                             *
02DB 04060911        1171 TAB30   DEFB 4,6,9,11H
                             *
                             *******************
                             *                 *
                             * PROGRAM MESSAGES *
                             *                 *
                             *******************
                             *
                             * FEATURES MESSAGE
                             *
02DF 0C1B592A        1181 SIGNON  DEFB FF,ESC,'Y','*',' '
     20
02E4 2A2A2A2A        1182         DEFM '**********     PRESENTING MOSTEK''S '
     2A2A2A2A
     2A2A2020
     20202050
     52455345
     4E54494E
     47204D4F
     5354454B
     275320
0307 4E455720        1183         DEFM 'NEW CLOCK/RAM PERIPHERAL CHIP     '
     434C4F43
     4B2F5241
     4D205045
     52495048
     4552414C
     20434849
     50202020
     2020
0329 2A2A2A2A        1184         DEFM '**********'
     2A2A2A2A
     2A2A
0333 0D0A0A0A        1185         DEFB CR,LF,LF,LF
0337 46454154        1186         DEFM 'FEATURES:'
     55524553
     3A
0340 0D0A0A          1187         DEFB CR,LF,LF
0343 2A20434D        1188         DEFM '* CMOS DESIGN FOR EXTREMELY LOW POWER
     4F532044
     45534947
     4E20464F
     52204558
     5452454D
     454C5920
     4C4F5720
     504F5745
     52
0368 20434F4E       1189           DEFM ' CCNSUMPTION.'
     53554D50
     54494F4E
     2E
0375 000A           1190           DEFB CR,LF
0377 2A204153       1191           DEFM '* ASYNCHRONOUS SERIAL COMMUNICATION AT'
     594E4348
     524F4E4F
     55532053
     45524941
     4C20434F
     4D4D554E
     49434154
     494F4E20
     4154
039D 20564952       1192           DEFM ' VIRTUALLY ANY BAUD RATE.'
     5455414C
     4C592041
     4E592042
     41554420
     52415445
     2E
03B6 000A           1193           DEFB CR,LF
03B8 2A203132       1194           DEFM '* 12/24 HOUR CLCCK/CALENDAR WITH AUTO'
     2F323420
     484F5552
     20434C4F
     434B2F43
     414C454E
     44415220
     57495448
     20415554
     4F
03DD 2041444A       1195           DEFM ' ACJUST FOR SHORT MCNTHS AND LEAP'
     55535420
     464F5220
     53484F52
     54204D4F
     4E544853
     20414E44
     204C4541
     50
03FE 20594541       1196           DEFM ' YEARS.'
     52532E
0405 000A           1197           DEFB CR,LF
0407 2A203234       1198           DEFM '* 24 BYTES OF RAM FOR POWER DOWN'
     20425954
     4553204F
     46205241
     4D20464F
     5220504F
     57455220
     444F574E
0427 2053544F          1199          DEFM ' STORAGE OF VITAL INFORMATION.'
     52414745
     204F4620
     56495441
     4C20494E
     464F524D
     4154494F
     4E2E
0445 0D0A              1200          DEFB CR,LF
0447 2A204F4E          1201          DEFM '* ON CHIP OSCILLATOR THAT PROVIDES A'
     20434849
     50204F53
     43494C4C
     41544F52
     20544841
     54205052
     4F564944
     45532041
046B 20434C4F          1202          DEFM ' CLOCK SIGNAL FOR YOUR MICROPROCESSOR.'
     434B2053
     49474E41
     4C20464F
     5220594F
     5552204D
     4943524F
     50524F43
     4553534F
     522E
0491 0D0A              1203          DEFB CR,LF
0493 2A205349          1204          DEFM '* SIMPLE INTERFACING TO ANY'
     4D504C45
     20494E54
     45524641
     43494E47
     20544F20
     414E59
04AE 204D4943          1205          DEFM ' MICROPROCESSOR.'
     524F5052
     4F434553
     534F522E
04BE 0D0A0A0A          1206          DEFB CR,LF,LF,LF
04C2 2A2A2A2A          1207          DEFM '**********     SEE YOUR MOSTEK REPRE'
     2A2A2A2A
     2A2A2020
     20202053
     45452059
     4F555220
     4D4F5354
     454B2052
     45505245
04E6 53454E54          1208          DEFM 'SENTATIVE FOR FURTHER DETAILS     '
     41544956
     4520464F
     52204655
     52544845
```

```
LOC  OBJ.CODE      STMT-NR SOURCE-STMT PASS2 DEMO    DEMO    DEMO    ABS

     52204445
     5441494C
     53202020
     20
0507 2A2A2A2A          1209         DEFM '**********'
     2A2A2A2A
     2A2A
0511 04                1210         DEFB EOT
                            *
                            * AM/PM MESSAGES
                            *
0512 1B592020          1214 GOODPT  DEFB ESC,'Y',' ',' ',ESC,'K',EOT
     1B4B04
0519 474F4F44          1215 GDMORN  DEFM 'GOOD MORNING!'
     204D4F52
     4E494E47
     21
0526 04                1216         DEFB EOT
0527 474F4F44          1217 GDAFTR  DEFM 'GOOD AFTERNOON!'
     20414654
     45524E4F
     4F4E21
0536 04                1218         DEFB EOT
                            *
                            * DAY MESSAGES
                            *
0537 1B592220          1222 DAYPT   DEFB ESC,'Y','"',' ',ESC,'K',EOT
     1B4B04
053E 53554E44          1223         DEFM 'SUNDAY'
     4159
0544 04                1224         DEFB EOT
0545 4D4F4E44          1225         DEFM 'MONDAY'
     4159
054B 04                1226         DEFB EOT
054C 54554553          1227         DEFM 'TUESDAY'
     444159
0553 04                1228         DEFB EOT
0554 5745444E          1229         DEFM 'WEDNESDAY'
     45534441
     59
055D 04                1230         DEFB EOT
055E 54485552          1231         DEFM 'THURSDAY'
     53444159
0566 04                1232         DEFB EOT
0567 46524944          1233         DEFM 'FRIDAY'
     4159
056D 04                1234         DEFB EOT
056E 53415455          1235         DEFM 'SATURDAY'
     52444159
                            *
                            * MONTH MESSAGES
                            *
0576 1B592420          1239 DATEPT  DEFB ESC,'Y','$',' ',ESC,'K',EOT
     1B4B04
057D 4A414E55          1240         DEFM 'JANUARY '
     41525920
0585 04                1241         DEFB EOT
```

```
0586 46454252        1242            DEFM 'FEBRUARY '
     55415259
     20
058F 04              1243            DEFB EOT
0590 4D415243        1244            DEFM 'MARCH '
     4820
0596 04              1245            DEFB EOT
0597 41505249        1246            DEFM 'APRIL '
     4C20
059D 04              1247            DEFB EOT
059E 4D415920        1248            DEFM 'MAY '
05A2 04              1249            DEFB EOT
05A3 4A554E45        1250            DEFM 'JUNE '
     20
05A8 04              1251            DEFB EOT
05A9 4A554C59        1252            DEFM 'JULY '
     20
05AE 04              1253            DEFB EOT
05AF 41554755        1254            DEFM 'AUGUST '
     535420
05B6 04              1255            DEFB EOT
05B7 53455054        1256            DEFM 'SEPTEMBER '
     454D4245
     5220
05C1 04              1257            DEFB EOT
05C2 4F43544F        1258            DEFM 'OCTOBER '
     42455220
05CA 04              1259            DEFB EOT
05CB 4E4F5645        1260            DEFM 'NOVEMBER '
     4D424552
     20
05D4 04              1261            DEFB EOT
05D5 44454345        1262            DEFM 'DECEMBER '
     4D424552
     20
05DE 04              1263            DEFB EOT
                             *
                             * YEAR SEPARATOR MESSAGE
                             *
05DF 2C203139        1267 SEPAR      DEFM ', 19'
05E3 04              1268            DEFB EOT
                             *
                             * SEND CURSOR TO TIME LINE MESSAGE
                             *
05E4 1B592620        1272 TIMEPT     DEFB ESC,'Y','&',' ',ESC,'K',EOT
     1B4B04
                             *
                             * SEND CURSOR HOME MESSAGE
                             *
05EB 1B59376F        1276 HOME       DEFB ESC,'Y','7','o',EOT
     04
                             *
                             * PROMPT MESSAGES
                             *
05F0 0C              1280 DAYMSG     DEFB FF
05F1 44415920        1281            DEFM 'DAY (1-7)? '
     28312D37
```

```
LOC  OBJ.CODE      STMT-NR SOURCE-STMT PASS2 DEMO   DEMO   DEMO   ABS

     293F20
05FC 04                1282          DEFB EOT
                            *
05FD 0D0A              1284 DATMSG   DEFB CR,LF
05FF 44415445          1285          DEFM 'DATE (YR:MO:DA)? '
     20285952
     3A4D4F3A
     4441293F
     20
0610 04                1286          DEFB EOT
                            *
0611 0D0A              1288 MODMSG   DEFB CR,LF
0613 4D4F4445          1289          DEFM 'MODE (0=24 HOUR, 1=12 HOUR)? '
     2028303D
     32342048
     4F55522C
     20313D31
     3220484F
     5552293F
     20
0630 04                1290          DEFB EOT
                            *
0631 0D0A              1292 AMPMSG   DEFB CR,LF
0633 414D2F50          1293          DEFM 'AM/PM (0=AM, 1=PM)? '
     4D202830
     3D414D2C
     20313D50
     4D293F20
0647 04                1294          DEFB EOT
                            *
0648 0D0A              1296 TIMMSG   DEFB CR,LF
064A 54494D45          1297          DEFM 'TIME (HR:MN:SC)? '
     20284852
     3A4D4E3A
     5343293F
     20
055B 04                1298          DEFB EOT
                            *
065C                   1300          END
```

```
CLOCK/RAM DEMONSTRATION MODULE    F8/3870 MACRO CROSS ASSM.  V2.2
 NAME TYP VALUE DEF   REFERENCES        PASS2 DEMO   DEMO   DEMO   ABS

AMPM      0020  140
AMPM1  '  0189  740    737
AMPM2  '  018C  741    739
AMPMIN '  00BC  478    363
AMPMOT '  01A4  720    287
AMPMSG '  0631 1292    483
APRIL     0004   93
ASCII  '  027A 1001   1011
AUG       0008   97
BAUD      000B  168    226
CE1       0002  160
CHIPEN    0001   26    222*
CLKRAM E  02CF           4 1130 1140 1148 1156
CLKRD  '  02C1 1144    286
CLKWR  '  02C9 1152    365
CMD       0002   27   1129*1137*1147*1155*
CNTSAV    0004   33    938* 947
CR        000D   68   1185 1187 1190 1193 1197 1200 1203 1206 1284 1288 1292
                      1296
CRCHIP    0002  176    221
CRCTRL    0000  175   1138
CRDATA    0004   51
CTRL      0007   36    236  237 1126 1127 1134 1135 1139*
DATA      0001  159
DATAOK '  0082  386    238
DATE      0014   41    689  801
DATE1  '  01DE  795    792
DATEIN '  0126  596    360
DATEOT '  01CC  782    289
DATEPT '  0576 1239    783
DATLSD    000F  146
DATMSD    0030  145
DATMSG '  05FD 1284    601
DAY       0013   40    455  762
DAYOX  '  016C  651    644  681
DAY30  '  0184  673    662
DAY3X  '  0171  656    646
DAYIN  '  00AB  448    359
DAYLSD    0007  144
DAYMSG '  05F0 1280    453
DAYOT  '  01C0  759    288
DAYPT  '  0537 1222    760
DC3       0013   69    335
DCOUNT    0005   34    658* 663* 940* 945*
DDATE  '  0153  634    625
DDATE1 '  0169  648    652  669  674  698
DDATE3 '  0166  647    683  691  693
DEC       000C  101
DGT    '  0225  884    891
DGTBAD '  0255  940    948
DGTLOP '  025A  943    946
DGTOK  '  0263  953    944
DIGIT  '  0250  933    897  903  909  915  921  927
DIGIT0 '  024D  931    548  562  573  607  610  647
DIGIT1 '  022F  895    673
DIGIT2 '  0234  901    424  486  529  618  665
DIGIT3 '  0239  907    534  543  622  678
```

```
CLOCK/RAM DEMONSTRATION MODULE    F8/3870 MACRO CROSS ASSM.  V2.2
 NAME TYP VALUE DEF   REFERENCES      PASS2 DEMO   DEMO   DEMO   ABS

DIGIT4 '  023E  913    552  641
DIGIT6 '  0243  919    559  570
DIGIT7 '  0222  882    456
DIGIT8 '  0248  925    697
DIGIT9 '  022A  889    538  626  651
DIGITT '  0253  938    885
DLOOP  '  0179  561    664
EIGHT     0008  113
EOT       0004   65   1069 1095 1210 1214 1216 1218 1222 1224 1226 1228 1230
                      1232 1234 1239 1241 1243 1245 1247 1249 1251 1253 1255
                      1257 1259 1261 1263 1268 1272 1276 1282 1286 1290 1294
                      1298
ESC       001B   70   1181 1214 1214 1222 1222 1239 1239 1272 1272 1276
FEB       0002   91    636
FEBXX  '  0189  678    637
FF        000C   67   1181 1280
FINISH '  0043  299    279  336
FIVE      0005  110
FNDLOP '  0298 1068   1070
FNDOUT '  0295 1066    763  800 1071
FOUR      0004  109    914
FRI       0006   85
GDAFTR '  0527 1217    738
GDMORN '  0519 1215    740
GOODPT '  0512 1214    721
HALT      0080  134
HOME   '  05EB 1276    291
HOUR      0012   39    423  485  522  727  834
HOUR0X '  00EF  538    530
HOUR1  '  00EB  535    539
HOUR2  '  0100  549    553
HOUR24 '  00F4  543    525
HOUR2X '  0104  552    547
HR1MSD    0010  142    839
HR2MSD    0030  141
HRLSD     000F  143    839
INCHR  '  0283 1029    942
INCHR2 '  0287 1032    334 1033
INIT   '  0001  211    217
ISMASK    003F  130    216
JAN       0001   90
JULY      0007   96
JUNE      0006   95
LEAP1     0013  125    690
LEAP2     0012  126    692
LF        000A   66   1185 1185 1185 1187 1187 1190 1193 1197 1200 1203 1206
                      1206 1206 1284 1288 1292 1296
LOOP1  '  0273  989    991 1096
LOOP3  '  02A2 1094   1101
LOOP4  '  02A9 1099   1100
LSBYTE    000F   57    988 1031 1034 1098
LSD       000F  120    955 1010
MARCH     0003   92
MAXCNT    0024  154    284  388
MAY       0005   94
MIN    '  0109  557    537  551
MINLSD    000F  138
```

```
CLOCK/RAM DEMONSTRATION MODULE    F8/3870 MACRO CROSS ASSM.  V2.2
 NAME TYP VALUE DEF   REFERENCES        PASS2 DEMO    DEMO    DEMO    ABS

MINMSD     0070  137
MINUTE     0011   38    558  847
MLTRY1 '   01BF  742    730
MLTRY2 '   0207  844    837
MNLSD      000F  148    793
MNMSD      0010  147
MNTHOX '   014E  626    621
MODE       0080  139
MODEIN '   0096  416    361
MODMSG '   0611 1288    421
MON        0002   81
MONTH      0015   42    617  656  785
MONTH1 '   014A  623    627
MSBYTE     000E   56    229 1039
MSD        00F0  121
NINE       0009  114    890  926
NOV        000B  100
OCT        000A   99
ONE        0001  106    896
OUTCHR '   026D  984    953
OUTCOL '   026A  977    557  568  616  634  846  850 1002
OUTLSD '   027E 1009    803  808  845  849  853
OUTMSD '   0278  999    802  807  844  848  852
OUTMSG '   029F 1092    292  393  422  454  484  518  602  722  741  761  784
                        805  829 1067
PARITY     00FE  164    228
PT4IMG     0000   25
RCV        00B0  170   1029
RCVI       00B1  171    297  394
RDCLK      00BF  179   1146
RDSTAT     008F  177   1128
RXCTRL     000C   54    227
RXSTAT     000D   55    298  395  985  989 1030 1032 1093 1099
SAT        0007   86
SECLSD     000F  136
SECMSD     0070  135
SECOND     0010   37    357  358  569  851 1144 1145 1152 1153
SEPAR  '   05DF 1267    804
SEPT       0009   98
SET1   '   007C  364    362
SETCLK '   006C  357    243
SEVEN      0007  112    883
SIGNON '   02DF 1181    392
SIX        0006  111    920
STATRD '   02AE 1126    235
STATWR '   02B6 1134    242
STOP   '   0094  397    397
SUN        0001   80
TAB09  '   02D1 1166    933
TAB19  '   02D2 1167    884
TAB30  '   02DB 1171    660
TEMP       0003   32    954  978* 986 1038*1042 1043*
TEN        000A  115    794  932
TENBCD     0010  116    790
THREE      0003  108    645  908
THURS      0005   84
TICTRL     0006   52    391
```

```
CLOCK/RAM DEMONSTRATION MODULE    F8/3870 MACRO CROSS ASSM.  V2.2
 NAME TYP VALUE DEF   REFERENCES        PASS2 DEMO    DEMO    DEMO    ABS

TIMCNT    0006   35    278* 285* 389*
TIMEIN '  00D0  512    364
TIMEOT '  01F7  827    290
TIMEPT '  05E4 1272    828
TIMER     0007   53    387
TIMMSG '  0648 1296    517
TMCTRL    00EA  155    390
TUES      0003   82
TWO       0002  107    546  682  902
WED       0004   83
WRCLK     00BE  180   1154
WRSTAT    008E  178   1136
XMIT      00A2  169    984 1092
YEAR      0016   43    606  687  806
YRLSD     000F  150
YRMSD     00F0  149
ZERO      0000  105
```

LISTING 2 - CLOCK/RAM COMMUNICATIONS SUBROUTINE

```
CLOCK/RAM COMMUNICATION MODULE    F8/3870 MACRO CROSS ASSM.  V2.2
LOC  OBJ.CODE      STMT-NR SOURCE-STMT PASS2 CLKRAM CLKRAM CLKRAM REL

                         1         TITLE CLOCK/RAM COMMUNICATION MODULE
                         2         NAME CLKRAM
                         3         PSECT REL
                         4         GLOBAL CLKRAM
                           *
                           * THIS MODULE MUST BE LINKED WITH OTHER MODULES
                           * IN ORDER TO CREATE A WORKING PROGRAM.
                           *
                           ***************************************
                           *                                     *
                           * CLOCK/RAM COMMUNICATION SUBROUTINE *
                           *                                     *
                           ***************************************
                           *
                           * THIS SUBROUTINE IS CALLED BY THE APPLICATION
                           * PROGRAM TO SEND AND RECIEVE DATA TO AND FROM THE
                           * CLOCK/RAM CHIP. WHEN CALLED, THE COMMAND TO BE
                           * EXECUTED MUST BE IN THE SCRATCH-PAD REGISTER
                           * 'CMD', THE CHIP ENABLE CODE MUST BE IN REGISTER
                           * 'CHIPEN' AND THE ISAR MUST POINT TO THE TOP OF
                           * THE DATA AREA.
                           *
                           * THIS ROUTINE ALLOWS THE PORT 4 BITS THAT ARE NOT
                           * USED FOR CHIP ENABLE LINES TO BE USED FOR OTHER
                           * PURPOSES. TO DO THIS, AN IMAGE OF WHATEVER IS
                           * WRITTEN TO THE PORT BY OTHER ROUTINES MUST BE
                           * KEPT IN REGISTER 'PT4IMG'. IN THIS WAY, THOSE
                           * PORT LINES NOT USED BY THIS ROUTINE WILL NOT BE
                           * ALTERED. HOWEVER, ANY OF THE PORT 4 LINES THAT
                           * ARE USED FOR THE CLOCK/RAM MUST ALWAYS BE LEFT
                           * AT A LOGICAL 0.
                           *
                           * COMMAND BYTE FORMAT:
                           * BIT 7 - MUST BE 1
                           * BIT 6 - SOURCE/DESTINATION (1=RAM, 0=CLOCK)
                           * BITS 5 THRU 1 - ADDRESS
                           * BIT 0 - DIRECTION (1=READ, 0=WRITE)
                           *
                           * FOR BYTE MODE, THE ADDRESS OF THE BYTE IS PUT
                           * INTO THE ADDRESS FIELD OF THE COMMAND. FOR BURST
                           * MODE, THE ADDRESS SHOULD BE 01FH. NOTE THAT A
                           * CLOCK BURST FUNCTION TRANSFERS ONLY THE 7 CLOCK
                           * BYTES. IT DOES NOT TRANSFER THE CONTROL BYTE.
                           * VALID ADDRESSES FOR THE COMMAND BYTE (FOR BYTE
                           * MODE) ARE:
                           * CLOCK - 0 THRU 07H
                           * RAM   - 0 THRU 017H
                           *
                           * CHIP ENABLE CONTROL BYTE FORMAT:
                           * BIT 7 THRU 1 - CONTROLS PORT 4 BITS 7 THRU 1
                           * BIT 0 - MUST BE 0 (USED FOR DATA I/O LINE)
                           *
                           * TO SELECT A CLOCK/RAM CHIP WITH IT'S /CE PIN
                           * TIED TO A PORT 4 PIN, THE CORRESPONDING BIT
                           * POSITION SHOULD BE SET TO A 1 (ALL OTHER BITS
                           * SHOULD BE 0).
                           *
```

```
                                   * CALLING SEQUENCE:
                                   * 1) DATA SHOULD BE IN DATA AREA (WRITE ONLY)
                                   * 2) LOAD ISAR TO POINT TO BOTTOM OF DATA AREA
                                   * 3) CHIPEN BYTE SHOULD BE IN REGISTER 'CHIPEN'
                                   * 4) COMMAND BYTE SHOULD BE IN REGISTER 'CMD'
                                   * 5) PORT 4 IMAGE SHOULD BE IN REGISTER 'PT4IMG'
                                   * 6) CALL CLKRAM
                                   * 7) RETURN WITH DATA AREA FILLED (READ ONLY)
                                   *
                                   * PORT 4 IS USED FOR ALL I/O SO THAT IT'S /STROBE
                                   * SERVES AS THE SHIFT REGISTER CLOCK (SRCLK) TO
                                   * THE CLOCK/RAM.
                                   *
                                   * AS PRESENTED HERE, THIS SUBROUTINE MUST NOT BE
                                   * INTERUPTED. BUT THE USER MAY EASILY MODIFY THE
                                   * CODE TO SUPPORT INTERUPTS.
```

```
                          *************
                          *           *
                          * CONSTANTS *
                          *           *
                          *************
                          *
                          * GLOBAL REGISTERS. THESE REGISTERS MUST BE THE SAME
                          * AS IN THE APPLICATION MODULE(S).
                          *
      =0000            84 PT4IMG  EQU  0            ;PORT 4 IMAGE STORAGE
      =0001            85 CHIPEN  EQU  1            ;CHIP ENABLE STORAGE
      =0002            86 CMD     EQU  2            ;COMMAND STORAGE
                          *
                          * LOCAL REGISTERS. THESE REGISTERS DO NOT NEED TO BE
                          * MADE KNOWN TO THE APPLICATION MODULE(S). HOWEVER,
                          * THEY ARE DISTROYED. SO THE APPLICATION MODULE(S)
                          * SHOULD NOT KEEP NEEDED INFORMATION IN THEM.
                          *
      =0003            93 TEMP    EQU  3            ;TEMPERARY STORAGE
      =0004            94 BITCNT  EQU  4            ;BIT COUNTER
      =0005            95 BYTCNT  EQU  5            ;BYTE COUNTER
                          *
                          * PORT DEFINITIONS
                          *
      =0004            99 PORT4   EQU  4            ;PORT 4
                          *
                          * BIT MASK DEFINITIONS
                          *
      =0001           103 BIT0    EQU  01H          ;BIT 0 MASK
      =0080           104 BIT7    EQU  80H          ;BIT 7 MASK
                          *
                          * COUNTER VALUES
                          *
      =0001           108 ONE     EQU  1            ;COUNT IS 1
      =0007           109 SEVEN   EQU  7            ;COUNT IS 7
      =0008           110 EIGHT   EQU  8            ;COUNT IS 8
      =0018           111 TWFOUR  EQU  24           ;COUNT IS 24
                          *
                          * COMMAND BIT DEFINITIONS
                          *
      =0001           115 RDWR    EQU  01H          ;READ/WRITE IS BIT 0
      =003E           116 ADR     EQU  3EH          ;ADDRESS IS BITS 1-5
      =0040           117 CKRM    EQU  40H          ;CLOCK/RAM IS BIT 6
```

```
                          *****************************
                          *                           *
                          * START OF CLOCK/RAM DRIVER *
                          *                           *
                          *****************************
                          *
0000'41               125 CLKRAM  LR   A,CHIPEN     ;PUT CHIP ENABLE INTO PT4I
0001 E0               126         XS   PT4IMG       ;
0002 50               127         LR   PT4IMG,A     ;
                          *
                          * SEND OUT COMMAND TO CLOCK/RAM
                          *
0003 42               131         LR   A,CMD        ;GET COMMAND
0004 53               132         LR   TEMP,A       ;SAVE COMMAND FOR OUTPUT
0005 2008             133         LI   EIGHT        ;BIT COUNT = 8
0007 54               134         LR   BITCNT,A     ;
0008'43               135 BLOOP   LR   A,TEMP       ;GET COMMAND BYTE
0009'2101             136 BLOOP1  NI   BIT0         ;MASK OFF ALL BUT BIT 0
000B 2301             137         XI   BIT0         ;COMPLEMENT BIT 0
000D E0               138         XS   PT4IMG       ;MIX IT WITH CONTROL BYTE
000E B4               139         OUTS PORT4        ;SEND IT OUT
000F 43               140         LR   A,TEMP       ;SHIFT FOR NEXT BIT
0010 12               141         SR   1            ;
0011 53               142         LR   TEMP,A       ;
0012 34               143         DS   BITCNT       ;DECREMENT BIT COUNT
0013 94F5             144         BNZ  BLOOP1       ;BRANCH IF NOT DONE
                          *
                          * SET BYTE COUNT TO PROPER LENGTH
                          *
0015 42               148         LR   A,CMD        ;GET COMMAND
0016 213E             149         NI   ADR          ;MASK OFF ALL BUT ADDRESS
0018 253E             150         CI   ADR          ;CHECK IF BYTE OR BURST
001A 940D             151         BNZ  BYTE         ;BRANCH IF BYTE
001C 42               152         LR   A,CMD        ;GET COMMAND BACK
001D 13               153         SL   1            ;CHECK RAM/CLOCK BIT
001E 9105             154         BM   RAM          ;BRANCH IF RAM
0020'2007             155 CLOCK   LI   SEVEN        ;CLOCK, SO BYTE COUNT = 7
0022 9007             156         BR   CONT         ;CONTINUE
0024'2018             157 RAM     LI   TWFOUR       ;RAM, SO BYTE COUNT = 24
0026 9003             158         BR   CONT         ;CONTINUE
0028'2001             159 BYTE    LI   ONE          ;BYTE, SO BYTE COUNT = 1
002A'55               160 CONT    LR   BYTCNT,A     ;
                          *
                          * MAIN BYTE TRANSFER LOOP
                          *
002B'42               164 MLOOP   LR   A,CMD        ;CHECK READ/WRITE BIT
002C 2101             165         NI   RDWR         ;
002E 70               166         CLR               ;
002F 9402             167         BNZ  XFER         ;BRANCH IF READ DIRECTION
0031 4C               168         LR   A,S          ;WRITE, SO LOAD BYTE
0032'53               169 XFER    LR   TEMP,A       ;
0033 2008             170         LI   EIGHT        ;BIT COUNT = 8
0035 54               171         LR   BITCNT,A     ;
0036 42               172         LR   A,CMD        ;CHECK READ/WRITE BIT
0037 2101             173         NI   RDWR         ;
0039 841B             174         BZ   WRITE        ;BRANCH IF WRITE DIRECTIO
                          *
```

```
                              * READ A BYTE
                              *
003B'43                   178 READ    LR   A,TEMP       ;SHIFT FOR NEXT BIT
003C'12                   179 READ1   SR   1            ;
003D 53                   180         LR   TEMP,A       ;
003E 40                   181         LR   A,PT4IMG     ;SEND OUT DUMMY CLOCK
003F B4                   182         OUTS PORT4        ;
0040 A4                   183         INS  PORT4        ;INPUT DATA BIT
0041 2101                 184         NI   BIT0         ;MASK ALL EXCEPT DATA BIT
0043 70                   185         CLR               ;IF DATA=1, FORCE BIT-7=0
0044 9403                 186         BNZ  READ2        ;BRANCH IF DATA = 1
0046 2080                 187         LI   BIT7         ;DATA=0, FORCE BIT-7=1
0048'E3                   188 READ2   XS   TEMP         ;MIX WITH PREVIOUS BITS
0049 34                   189         DS   BITCNT       ;DECREMENT BIT COUNT
004A 94F1                 190         BNZ  READ1        ;BRANCH IF NOT 8 BITS
004C 5C                   191         LR   S,A          ;STORE BYTE
                              *
                              * CHECK IF ALL BYTES WERE TRANSFERED
                              *
004D'35                   195 ENDCK   DS   BYTCNT       ;DECREMENT BYTE COUNT
004E 8415                 196         BZ   EXIT         ;BRANCH IF DONE
0050 0A                   197         LR   A,IS         ;INCREMENT POINTER
0051 1F                   198         INC               ;
0052 0B                   199         LR   IS,A         ;
0053 90D7                 200         BR   MLOOP        ;LOOP BACK FOR NEXT BYTE
                              *
                              * WRITE A BYTE
                              *
0055'43                   204 WRITE   LR   A,TEMP       ;GET DATA BYTE
0056'2101                 205 WRITE1  NI   BIT0         ;MASK OFF ALL BUT BIT 0
0058 2301                 206         XI   BIT0         ;COMPLEMENT BIT 0
005A E0                   207         XS   PT4IMG       ;MIX IT WITH CONTROL BYTE
005B B4                   208         OUTS PORT4        ;SEND IT OUT
005C 43                   209         LR   A,TEMP       ;SHIFT FOR NEXT BIT
005D 12                   210         SR   1            ;
005E 53                   211         LR   TEMP,A       ;
005F 34                   212         DS   BITCNT       ;DECREMENT BIT COUNT
0060 94F5                 213         BNZ  WRITE1       ;BRANCH IF NOT 8 BITS
0062 90EA                 214         BR   ENDCK        ;CONTINUE
                              *
                              * EXIT FROM SUBRCUTINE
                              *
0064'41                   218 EXIT    LR   A,CHIPEN     ;RESTORE PORT 4 IMAGE
0065 E0                   219         XS   PT4IMG       ;
0066 50                   220         LR   PT4IMG,A     ;
0067 B4                   221         OUTS PORT4        ;DISABLE CHIP
0068 1C                   222         POP               ;FINISHED
```

```
CLOCK/RAM COMMUNICATION MODULE    F8/3870 MACRO CROSS ASSM.  V2.2
 NAME TYP VALUE DEF   REFERENCES        PASS2 CLKRAM CLKRAM CLKRAM REL

ADR        003E  116    149  150
BIT0       0001  103    136  137  184  205  206
BIT7       0080  104    187
BITCNT     0004   94    134* 143* 171* 189* 212*
BLOOP  '   0008  135
BLOOP1 '   0009  136    144
BYTCNT     0005   95    160* 195*
BYTE   '   0028  159    151
CHIPEN     0001   85    125  218
CKRM       0040  117
CLKRAM I   0000  125      4
CLOCK  '   0020  155
CMD        0002   86    131  148  152  164  172
CONT   '   002A  160    156  158
EIGHT      0008  110    133  170
ENDCK  '   004D  195    214
EXIT   '   0064  218    196
MLOOP  '   002B  164    200
ONE        0001  108    159
PORT4      0004   99    139  182  183  208  221
PT4IMG     0000   84    126  127* 138  181  207  219  220*
RAM    '   0024  157    154
RDWR       0001  115    165  173
READ   '   003B  178
READ1  '   003C  179    190
READ2  '   0048  188    186
SEVEN      0007  109    155
TEMP       0003   93    132* 135  140  142* 165* 178  180* 188  204  209  211*
TWFOUR     0018  111    157
WRITE  '   0055  204    174
WRITE1 '   0056  205    213
XFER   '   0032  169    167
```

LISTING 3 - LOAD MAP AND GLOBAL CROSS REFERENCE

```
LOAD MAP

DK1:DEMO   .OBJ[1]     ABS     BEG ADDR 0000     END ADDR 065B
DK1:CLKRAM.OBJ[1]      REL     BEG ADDR 065C     END ADDR 06C4


 GLOBAL CROSS REFERENCE TABLE

 SYMBOL ADDR   REFERENCES
 CLKRAM 065C   02CF 02C7 02BF 02B4
```

MOSTEK®

USING MK3807 VCU IN A MICROPROCESSOR ENVIRONMENT

Application Note

INTRODUCTION

MK3807, the programmable CRT Video Control Unit (VCU), is a user programmable 40-pin n-channel MOS/LSI chip containing the logic functions required to generate all the timing signals for the formatting and presentation of interlaced or non-interlaced video data on a standard or non-standard CRT monitor.

All the formatting, such as horizontal, vertical, and composite sync, characters per data row and per frame are totally user programmable. The data row counter has been designed to facilitate scrolling.

Programming is accomplished by loading seven 8 bit control registers directly off an 8 bit bidirectional data bus. Four register address lines and a chip enable line provide complete microprocessor compatibility for program controlled set up. The device can also be "self loaded" via an external PROM tied on the data bus. (See Figure 1).

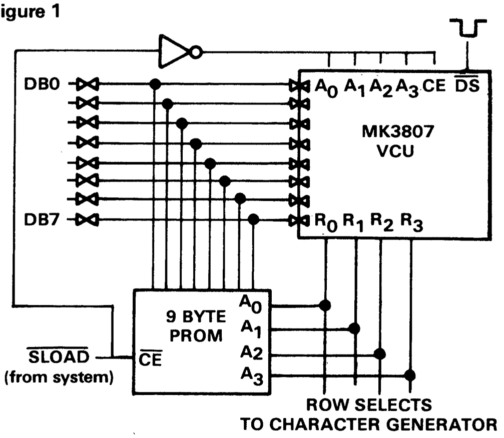
In addition to the seven control registers, two additional registers are provided to store the cursor character and row addresses for generation of the cursor video signal. The contents of these two registers can be read out onto the bus for update by the program or used by the microprocessor as two memory locations. (See Figure 2).

PROGRAM REGISTERS

The VCU contains 9 working registers (7 control registers and 2 data location registers).

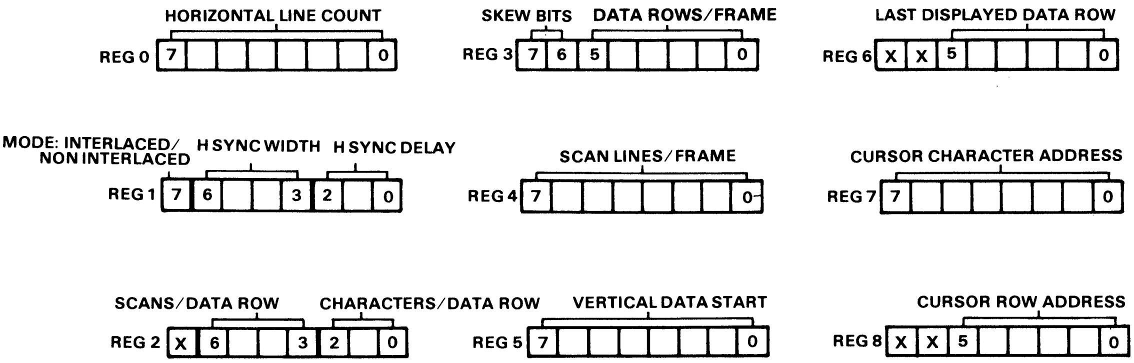
SELF LOADING SCHEME FOR VCU SET-UP

Figure 1



BIT ASSIGNMENT

Figure 2



REGISTER 0

This 8 bit register contains the number of character times for 1 horizontal period of the TV raster scan. For example, using American Standard Television (63.5 μs per line) at a character time of 500 ns, the value for this register would be 63.5 divided by .5 = 127. The number in this register is normally 1.25 times the number of characters per line displayed on this screen. The value loaded into this register is the binary equivalent of 126 (127-1). Since character times are counted from zero instead of one, the value loaded into this register is one less than the actual number of character times. (Refer to Figure 3 for timing diagrams).

REGISTER 1

This register contains 3 fields of information. The most significant bit (7) is the interlace bit. If this bit is set to a 1, Interlace mode is indicated; if set to a 0, Non-Interlace mode is indicated. The next 4 bits (6-3) define the number of character times for the width of the horizontal sync pulse. For example, using American Standard Television (4.5 μs) and a character time of 500 ns indicates that it would require 9 character times, therefore the binary equivalent 9 would be loaded in these bits. The least significant 3 bits (2-0) are used to specify the horizontal sync delay. This is commonly called the Front Porch and is the period between the end of active video to the beginning of the horizontal sync pulse. The value here is not critical and can be used to position the video horizontally on the screen.

REGISTER 2

This register contains both the number of characters to be displayed per line as well as the number of scans per character. Bit 7 is not used (B7 = X). Bits 6 through 3 define the number of scans per character. For example, using a 7 X 9 dot matrix character generator, the normal number of scans might be 12. Therefore, using 12 scans per character, the binary equivalent of eleven (12-1) is inserted into this field. The least significant 3 bits (2-0) contain a 3 bit code which defines the number of characters per line. The VCU is pre-programmed for 20, 32, 40, 64, 72, 80, 96, and 132 characters per line. The 3 bit binary number used in this field determines the particular format, for example, 80 characters being the 6th value would be coded as a binary 5 (101).

CHARACTERS/DATA ROW

DB2	DB1	DB0	
0	0	0	= 20
0	0	1	= 32
0	1	0	= 40
0	1	1	= 64
1	0	0	= 72
1	0	1	= 80
1	1	0	= 96
1	1	1	= 132

REGISTER 3

This register contains both the propagation delay compensation field (skew bits) as well as the data row fields. Bits 7 and 6 are used to adjust the blanking, cursor position and sync delay so as to compensate for either 0, 1 or 2 character time propagation delays of the character generator and the frame buffer RAM.

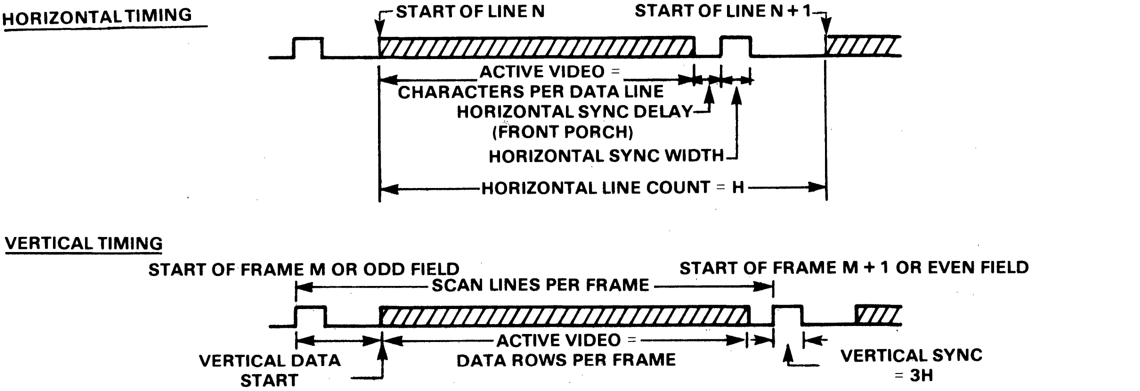
SKEW BITS

		Sync/Blank Delay	Cursor Delay
DB7	DB6	(Character Times)	
0	0	0	0
1	0	1	0
0	1	2	1
1	1	2	2

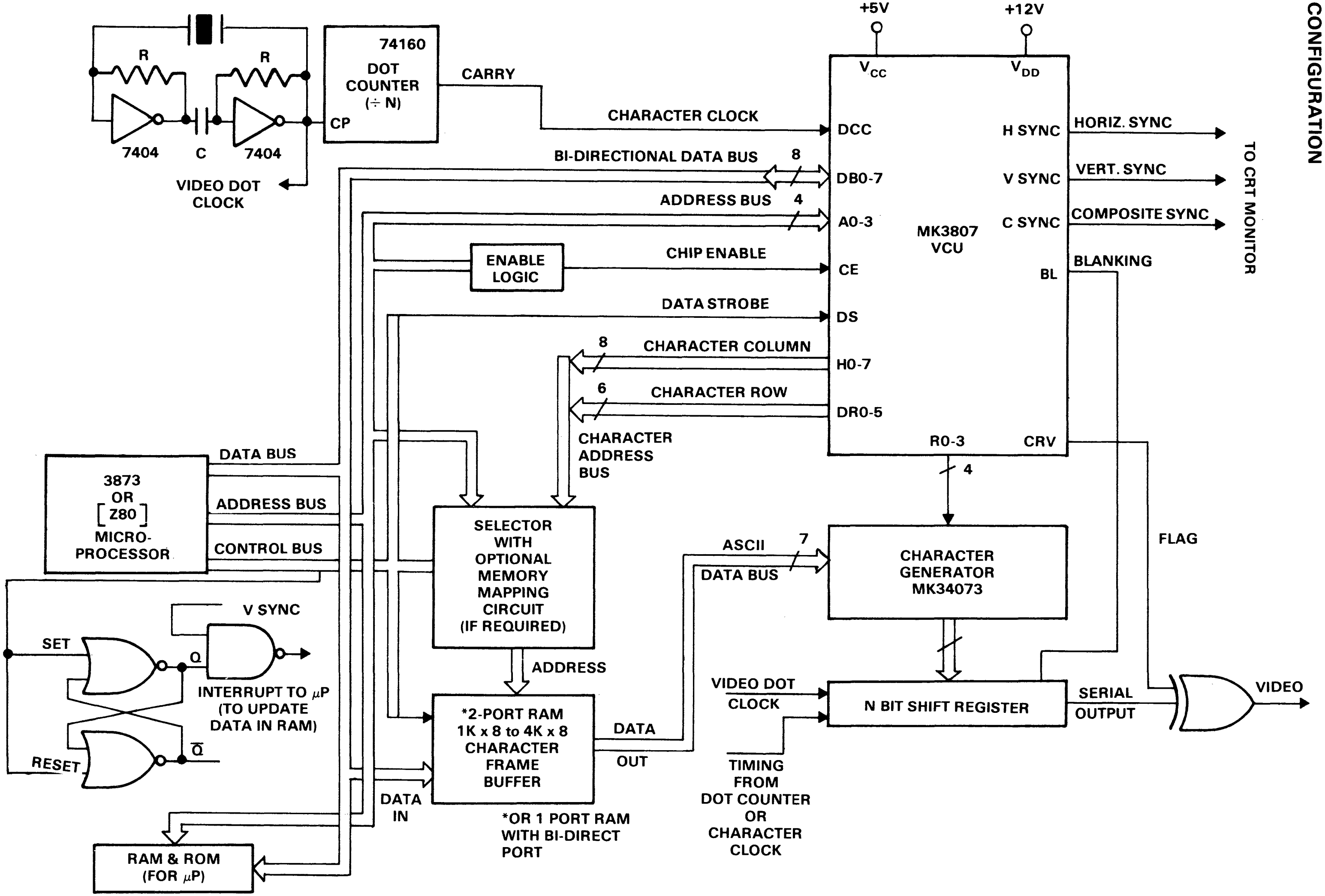
The 6 least significant bits (5-0) define the number of data rows to be displayed on the screen. The number of rows begins at 000000 (single row) and continues to 111111 (64 rows).

HORIZONTAL AND VERTICLE TIMING

Figure 3



VCU μP CONFIGURATION
Figure 4



REGISTER 4

This 8 bit register defines the number of raster lines in the field (frame). Care should be taken when programming this register to make sure that the product of the scans per data row times the number of data rows is less than the number of raster scans. There are 2 methods of programming this register. In the interlaced mode subtract 513 from the number of raster lines desired and divide by 2. For example, for 525 scans, the register should contain the number 6. In the non-interlaced mode subtract the number 256 from the desired number of raster lines and divide by 2. For example, for 262 raster lines, the value is 3.

REGISTER 5

This register defines the number of raster lines between the beginning of the vertical sync pulse and the start of the first data row being displayed. Typically, values of 20 or 21 lines are used. Higher values can be used to position data lower on the screen to a maximum 255. This is called Vertical Data Start and is the sum of Vertical Sync and Vertical Scan Delay.

REGISTER 6

The least significant 6 bits (5-0) of this register define the last data row to be displayed on the screen. Bits 7 and 6 are not used. This feature is useful for both scrolling and positioning of data. For example, if the display was set for 24 data rows, normally row 0 would be on top of the screen and row 23 would be at the bottom. If the scroll register (register 6) contained the number 15, then row 15 would be at the bottom and row 16 would be at the top of the screen. Row 23 and row 0 would be contiguous in the middle of the screen.

REGISTER 7

This 8 bit register contains the character number at which the cursor is to be addressed. For example, if the last character of an 80 character per line display were to be cursored, the binary equivalent of 79 would be in this register.

REGISTER 8

The least significant 6 bits (5-0) of this register define the data row for the cursor; similar to Register 7.

BASIC DISPLAY CONFIGURATION

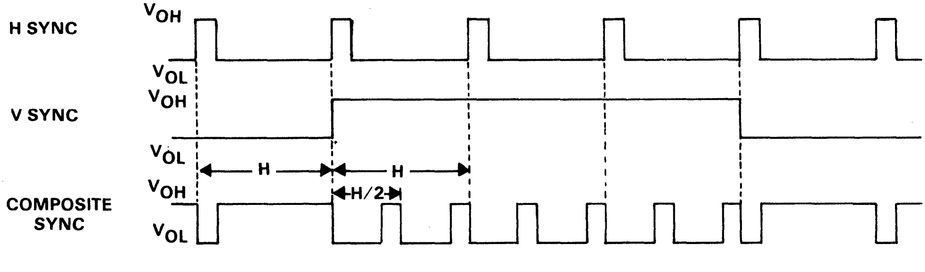
Figure 4 shows the basic configuration for a Bus Oriented, microprocessor based, CRT display system utilizing Mostek's MK3807, the Programmable CRT Video Control Unit (VCU). Either a standard or a non-standard CRT monitor may be used. The user programmable VCU provides Horizontal Sync, Vertical Sync and Composite Sync with serrations, to the monitor's sync deflection circuitry. (Figure 5 shows the composite sync timing). A serial output character generator provides video dot clock frequency data to the Z axis video input of the monitor.

In addition to the VCU, character generator, and shift register, the display system requires a crystal oscillator and a dot counter, typically consisting of two gates of a 7404 and a crystal as well as a 74160 (or equivalent) dot counter. The dot counter divisor (N) is set for the number of horizontal bits in the character plus the number of dots desired for spacing (i.e., for a 7 bit wide character + 2 dots of spacing N = 9). The carry output of the dot counter pulses once per character (character clock) and is fed into the MK3807 DCC (pin 12) input. This enables the VCU to keep track of the character positions as well as generate the entire video timing chain. At the same time the output of the oscillator is fed into the video dot clock input of the shift register of the Video Signal Generator.

An 8 bit bidirectional Data Bus (DB0-DB7), a 4 bit Address Bus (A0-A3), a Chip Enable and a Data Strobe are used in programming the VCU. These buses connect to the microprocessor Data Bus and Address Bus. The VCU appears to the microprocessor as 16 memory or I/O locations. Page logic (high order address bit decoder) connects the Address Bus to the Chip Enable (CE) thereby determining where in the microprocessor memory space the VCU will be located. The Data Strobe (DS) signal is connected to the microprocessor Control Bus. This is used to read or write via the Data Bus, as well as to activate control functions.

COMPOSITE SYNC TIMING DIAGRAM

Figure 5



The VCU raster scan counter outputs (R0-R3) are connected directly to the raster line address inputs of the character generator. This 4 bit address indicates which raster line of the selected character is to be parallel loaded into the shift register. The bit pattern, along with the additional blank spaces, is then shifted out of the video output at the video dot clock rate. The blanking signal can be connected to retrace blanking logic to provide both horizontal and vertical blanking of the video signal to the CRT monitor. The load/shift signals for character generator logic can be derived from the outputs of the dot counter (74160) or taken directly from the character clock (DCC, pin 12 of 3807).

HOW TO USE ROW-COLUMN ADDRESSING

The VCU outputs the character position via the character counter outputs (H0-H7) and the data row counter outputs (DR0-DR5). These outputs define the character column and row location. They are used to address a character frame buffer RAM in which the frame image is stored. Since the VCU keeps counting horizontal addresses (H0-H7) during both horizontal and vertical blanking, dynamic RAMs may be refreshed.

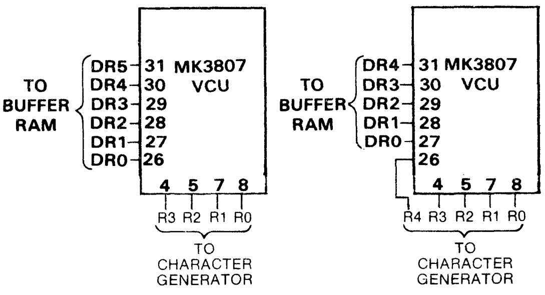
Many advantages are realized using Row-Column (X-Y) Addressing. Among these are:

Oversize Characters

Character fonts with heights greater than 16 dots (raster lines) can be achieved. This is done by using the LSB of the row counter (DR0) as the MSB of the raster scan counter (R4), and then moving the remaining bits of the row counter down one bit (DR1 becomes DR0, etc.). This is achieved by connecting the pins of the VCU in a different configuration. No additional components are required. This is shown in Figure 6. In addition, the VCU must be programmed for twice the desired number of data rows; thus using the above configuration (Figure 6), 32 rows of data with up to 32 lines per character (or 16 rows of data with up to 64 lines per character) can be accomplished.

USING THE VCU WITH CHARACTER FONTS OF HEIGHTS GREATER THAN 16 DOTS (LINES)

Figure 6



A. 64 ROWS OF 16 LINES **B. 32 ROWS OF 32 LINES**

Page Scrolling

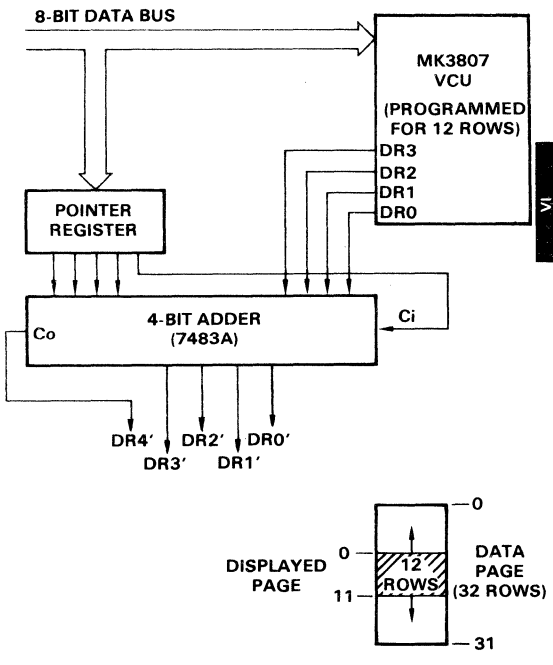
Scrolling a smaller page through a larger page (1K in 4K) can be done on a row by row basis. If the DR0-DR5 lines are offset by a pointer register, the smaller page can be moved up or down inside the larger page by the offset number of rows. This is shown in Figure 7. In this example, if the pointer register contains zero, the VCU will address the first 12 lines of the 32 line page. When the pointer register contains ten, the VCU will address rows 10 to 21. Thus, by loading the pointer register (from the microprocessor data bus), the display can scroll row by row through the data base.

Software Addressing

Most programmers use X — Y (row-column) addressing when writing software for CRT terminals. This makes it easier to blank the bottom line when scrolling, changing cursor positions, etc. Therefore, by having row-column addressing in the VCU, the address bus of the microprocessor can also have the preferred row-column addressing, and the two buses can be mapped together as shown in Figure 8. Without this feature, a software algorithm would have to convert a row-column address to binary address every time the microprocessor wanted to access the frame buffer. This algorithm usually requires a 16 bit multiplication. Thus the VCU, by utilizing row-column addressing, can save significant overhead and program execution time.

SCROLLING A 12 ROW PAGE THRU A 32 ROW PAGE

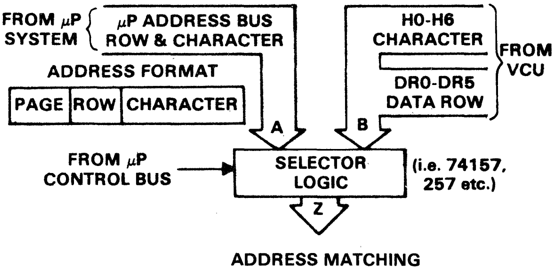
Figure 7



MEMORY MULTIPLEXING

The character column and character row outputs combine to form the character address bus. This bus, along with the microprocessor address bus, is connected to a 2 X 1 selector which addresses the character frame buffer RAM. Figure 8 shows the selector and the mapping for the various formats of the standard VCU. Numerous methods are available to build 2 X 1 selectors. One low-cost technique uses three 74157 or equivalent (74LS157 or 257, 9322, etc.) quad 2 X 1 selector chips. Figure 8 tabulates the mapping on to the microprocessor address bus into the selector with the DR and H lines of the VCU. The output of the selector (Z), is decomposed into two fields, row (Y) and column or character (X). Refer to Table 1.

ADDRESS BUS MAPPING
Figure 8



Memory Addressing

When the number of characters per row is non-binary, i.e. 80, addressing the frame buffer RAM is wasteful of memory. To solve this problem and still retain the advantages of row-column addressing, an address mapping is performed. The output of the selector (Z) is connected to another 74157 quad 2 X 1 selector chip or equivalent. Figures 6A, B, and C show the connection for 12 rows (1K), 24 rows (2K), and 48 rows (4K) of 80 characters. Figure 5 shows the mapping technique. The first 64 characters are mapped directly and the next 16 characters (H6 = 1) are mapped in a higher part of the RAM. The microprocessor address (row and column), is overlayed onto the VCU address bus (row and column) via the selector. The output of the selector maps into the frame buffer. Thus, every character is addressed by its row and column from both the microprocessor and the VCU. The

ADDRESS BUS MAPPING
Table 1

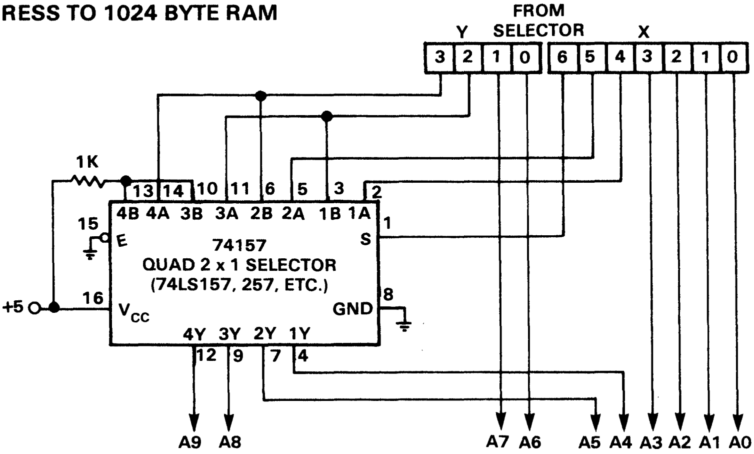
	SELECTOR													
µP ADDRESS BUS (UNUSED BITS ARE FOR PAGE LOCATION)	INPUT (A)	AB12	AB11	AB10	AB9	AB8	AB7	AB6	AB5	AB4	AB3	AB2	AB1	AB0
20 & 32 CHARACTERS/LINE														
FUNCTIONS		ROW								CHARACTER				
VCU OUTPUTS	INPUT (B)			DR5	DR4	DR3	DR2	DR1	DR0	H4	H3	H2	H1	H0
SELECTOR OUTPUTS	OUTPUT (Z)			Y5	Y4	Y3	Y2	Y1	Y0	X4	X3	X2	X1	X0
40 & 64 CHARACTERS/LINE														
FUNCTIONS		ROW							CHARACTER					
VCU OUTPUTS	INPUT (B)		DR5	DR4	DR3	DR2	DR1	DR0	H5	H4	H3	H2	H1	H0
SELECTOR OUTPUTS	OUTPUT (Z)		Y5	Y4	Y3	Y2	Y1	Y0	X5	X4	X3	X2	X1	X0
72, 80 & 96 CHARACTERS/LINE														
FUNCTIONS		ROW						CHARACTER						
VCU OUTPUTS	INPUT (B)	DR5	DR4	DR3	DR2	DR1	DR0	H6	H5	H4	H3	H2	H1	H0
SELECTOR OUTPUTS	OUTPUT (Z)	Y5	Y4	Y3	Y2	Y1	Y0	X6	X5	X4	X3	X2	X1	X0
132 CHARACTERS/LINE														
FUNCTIONS		ROW					CHARACTER							
VCU OUTPUTS	INPUT (B)	DR4	DR3	DR2	DR1	DR0	H7	H6	H5	H4	H3	H2	H1	H0
SELECTOR OUTPUTS	OUTPUT (Z)	Y4	Y3	Y2	Y1	Y0	X7	X6	X5	X4	X3	X2	X1	X0

MEMORY MAPPING CIRCUITS FOR 72 OR 80 CHARACTERS/LINE

Figure 9

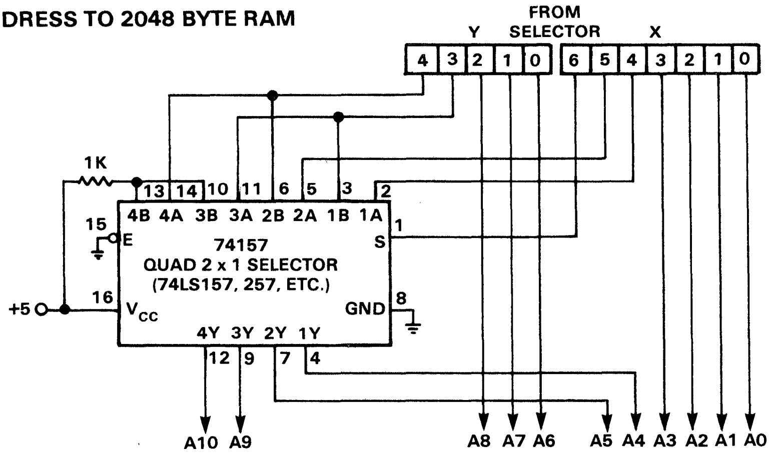
10 BIT BINARY ADDRESS TO 1024 BYTE RAM 12 LINES INTO 1K

Figure 9A



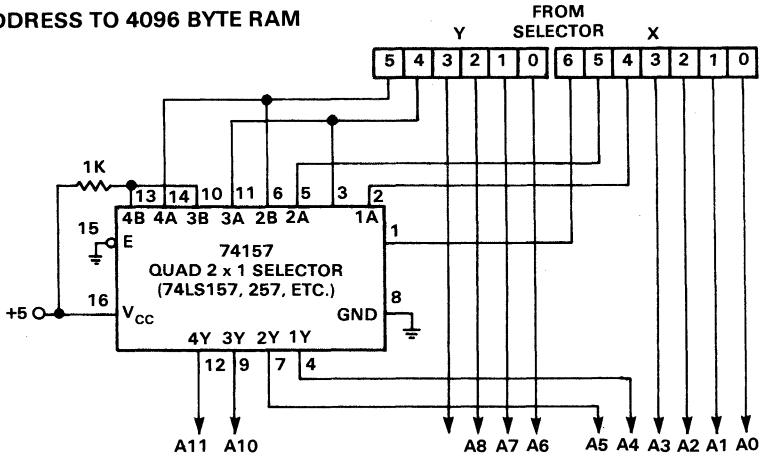
11 BIT BINARY ADDRESS TO 2048 BYTE RAM 24 LINES INTO 2K

Figure 9B



12 BIT BINARY ADDRESS TO 4096 BYTE RAM 48 LINES INTO 4K

Figure 9C



same memory location will be accessed whether the identical address originates from the microprocessor or VCU address bus.

OPERATION

The character frame buffer RAM is initially loaded via the microprocessor data and address buses (see Figure 1). After the microprocessor has loaded the character frame buffer RAM with a complete page, the selector flip-flop is switched (via the microprocessor control bus) so that the RAM is addressed by the character address bus of the VCU. In this mode the VCU operates independent of the microprocessor by addressing the character frame buffer RAM which sends the ASCII data to the CRT character generator. The selected character is then further decomposed by the raster scan counter (RO-R3), from the VCU, and loaded into the character generator shift register. This bit pattern is then serially shifted out at the video dot clock frequency and the data can be encoded so as to compose the video signal.

One possible way to change the data in the frame buffer (which is in microprocessor address space but physically separate) is: whenever the data in the character frame buffer is to be changed or updated, the microprocessor (via the control bus) sets an external flip-flop. The output of this flip-flop is ANDed with the vertical sync signal from the VCU. When this occurs an interrupt is generated to the microprocessor. This alerts the microprocessor to the fact that the vertical blanking interval has begun; it then switches the address selector (via control bus) so that the character frame buffer is now addressed by the microprocessor instead of the VCU. Since the system is in the vertical blanking interval, the screen is blank at this time. Using the American standard of 63.5 μs. per horizontal line and a typical value of 21 horizontal lines for the blanking interval, this gives the system 1.33 ms. in which the microprocessor can change data in the character frame buffer. If this time is not sufficient, the 1.33 ms. window will appear every 1/60 of a second allowing the microprocessor to change part of the RAM data each time.

After the microprocessor has completed its updating of the character frame buffer RAM, it resets the external flip-flop (via the control bus) and switches the selector back to the character address bus of the VCU. Then the microprocessor goes about its normal system operation without being interrupted or having its thrughput slowed down. This is because the VCU refreshes the CRT independently with the character frame buffer RAM, supplying the data, while the microprocessor operates at full speed with its own RAM and ROM. This method is more efficient for microprocessor throughput and control as opposed to having to DMA (cycle steal) or interrupt the processor continually, thereby reducing its throughput.

SYNC-LOCK

Some applications require adding alphanumeric characters (text) or graphics to the same screen as closed circuit or external (off-the-air) video. Figure 11 illustrates a simple technique of externally synchronizing the VCU using 2 chips (7474 and 7402 or equivalent). The external video can come from a closed circuit television system, off-the-air television, or some other video display system. The technique involves stopping the character clock (DCC) when the VCU sync occurs and restarting it when the external sync occurs. In this way, the VCU will be synchronized to the external video. One requirement for the reliable operation of this system is that the VCU horizontal and vertical sync rates must be programmed to be slightly faster than the external sync rate (i.e., the horizontal line counter register of the VCU must be programmed to be less than 63.5 μs., which is the American TV horizontal rate).

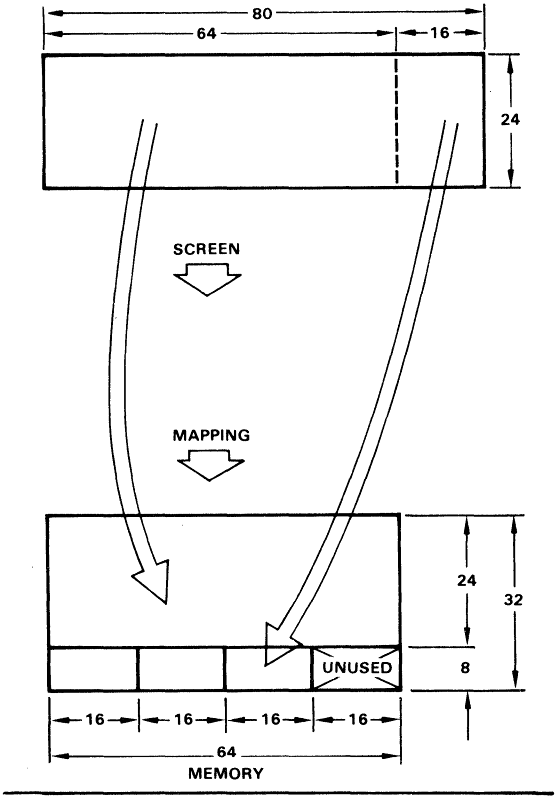
HOW TO PROGRAM THE MK3807 VCU

In order to pick the correct video dot clock frequency and to program the registers in the VCU, it is first necessary to determine several key parameters. Among these parameters are: the vertical refresh rate, the number of horizontal raster lines per frame, the number of characters per line and the format of the characters.

Tables 2A, B list work sheets which give the designer an

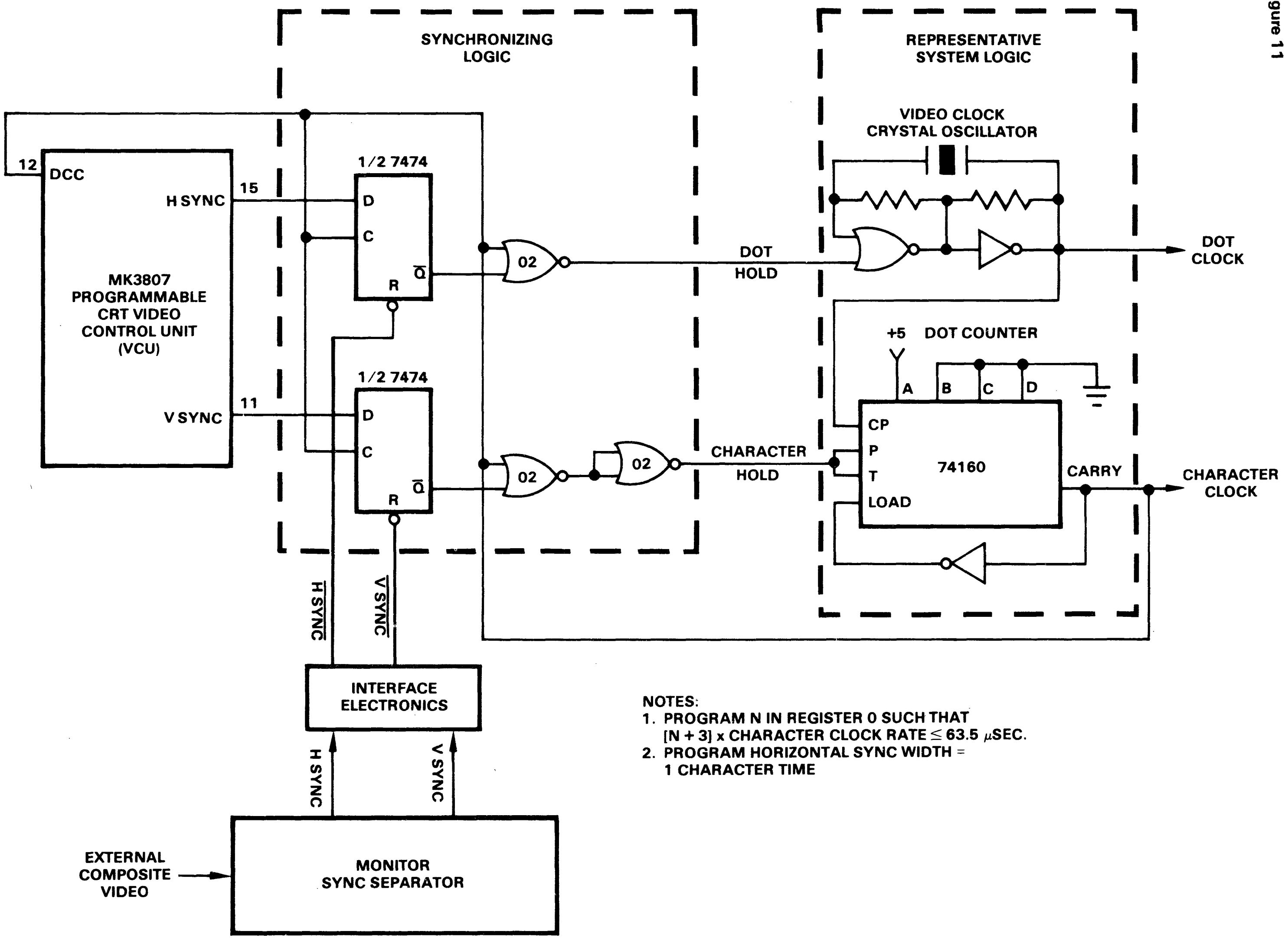
ADDRESS COMPRESSION SCHEME FOR 80 CHARACTERS/LINE

Figure 10



MK3807 EXTERNAL VIDEO SYNCHRONIZING CIRCUIT

Figure 11



NOTES:
1. PROGRAM N IN REGISTER 0 SUCH THAT [N + 3] x CHARACTER CLOCK RATE ≤ 63.5 μSEC.
2. PROGRAM HORIZONTAL SYNC WIDTH = 1 CHARACTER TIME

orderly method of determining the frequencies and register contents from the above parameters. In order to demonstrate its use, typical examples will be shown.

EXAMPLE FOR 80 CHARACTERS BY 24 ROWS

A 7 X 9 character matrix is chosen as it is the most popular for the display of both upper and lower case characters. Also, a non-interlaced system is chosen. The character block of 9 X 12 allows for a 2 dot space between characters and a 3 line space between data rows. The impact of the character block size on the horizontal frequency and the video clock rate will be shown below. A frame refresh rate of 60Hz is chosen for this example. These numbers can be modified for 50Hz systems.

This system will have 24 rows of data and 80 characters per data row. Thus, there are (24 X 12) 288 active scan lines.

The monitor chosen for this example is capable of accepting a composite video signal or separate TTL horizontal and vertical sync pulses. The sum of the horizontal sync delay (front porch), horizontal sync pulse, and horizontal scan delay (back porch) is the horizontal blanking interval. This interval is required as a window in the horizontal scan period to allow retrace. The retrace time is internal to the CRT monitor; this time is a function of monitor horizontal scan components. This time, at a minimum, is the time it takes the display to return from the right to the left hand side of the display. The retrace time is less than the horizontal blanking interval. The horizontal blanking interval is normally about 20% of the total horizontal scanning period. See Figure 12 for horizontal and verticle timing, and Figure 13 for derived register bit assignments.

In an 80 character per data row system, this would give 20 character times for the sum of the Front Porch, Horizontal Sync Pulse, and Back Porch. In the example of table 2C, a sum of 22 character time is used to illustrate that some flexibility exists in the choice of these parameters.

The vertical scanning frequency can be obtained by counting the total number of horizontal lines. The total number of scan lines generated for a vertical field equals the number of data rows times the number of lines per character plus the vertical sync delay plus the vertical sync pulse plus the vertical scan delay.

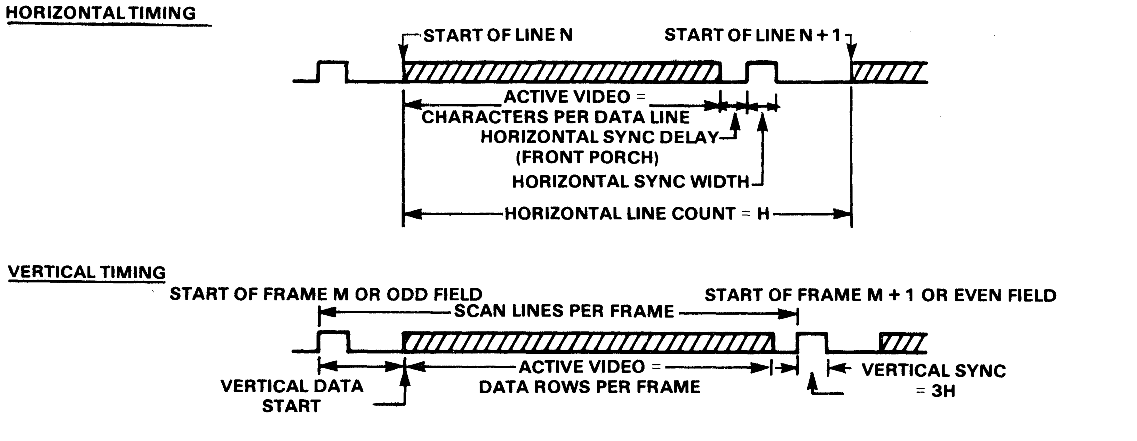
Vertical sync delay is the number of scan lines delay before vertical sync. Vertical sync pulse width should be expresed in scan line units. The VCU is fixed at the standard vertical sync width of 3 horizontal scan lines (3H). Scan line delay is the delay between vertical sync and the display information in scan line units. The sum of the vertical sync and the 2 delays in the vertical blanking interval is normally 5% to 8% of the total number of scan lines.

The vertical period (for 60Hz vertical refresh rate) can be calculated as: 1 divided by 60Hz = 16.67 ms.

Thus, the vertical blanking period (at 8%) equals 1.3 ms. In the example of table 2C, the sum of the "Front Porch, Vertical Sync Pulse, and Back Porch" are 22 scan lines long. Again, some flexibility exists in the choice of these parameters.

Adding the displayed lines (24 X 12 = 288) plus the vertical blanking interval (0 + 3 + 19 = 22), 310 horizontal scan lines are required. These 310 lines must be repeated 60 times a second (every 16.67 ms.). Thus 18,600 horizontal scan lines per second is the horizontal frequency. It can now be seen that any further increase in the number of scan lines per data character block will cause a direct increase in the horizontal frequency, possibly to a point beyond the monitor's specification.

HORIZONTAL AND VERTICAL TIMING
Figure 12



MK3807 VCU WORK SHEET

Table 2A

1. H CHARACTER MATRIX (No. of Dots): .... ________

2. V CHARACTER MATRIX (No. of Horiz. Scan Lines): .... ________

3. H CHARACTER BLOCK (Step 1 + Desired Horiz. Spacing = No. in Dots): .... ________

4. V CHARACTER BLOCK (Step 2 + Desired Vertical Spacing = No. in Horiz.
   Scan Lines): .... ________

5. VERTICAL FRAME (REFRESH) RATE (Freq. in Hz): .... ________

6. DESIRED NO. OF CHARACTER ROWS: .... ________

7. TOTAL NO. OF ACTIVE "VIDEO DISPLAY" SCAN LINES
   (Step 4 x Step 6 = No. in Horiz. Scan Lines): .... ________

8. VERT. SYNC DELAY (No. in Horiz. Scan Lines): .... ________

9. VERT. SYNC (No. in Horiz. Scan Lines; T = ______ μs*): .... ________

10. VERT. SCAN DELAY (No. in Horiz. Scan Lines; T = ______ ms*): .... ________

11. TOTAL VERTICAL FRAME (Add steps 7 thru 10 = No. in Horiz. Scan Lines): .... ________

12. HORIZONTAL SCAN LINE RATE (Step 5 x step 11 = Freq. in KHz): .... ________

13. DESIRED NO. OF CHARACTERS PER HORIZ. ROW: .... ________

14. HORIZ. SYNC DELAY (No. in Character Time Units; T = ______ μs**): .... ________

15. HORIZ. SYNC (No. in Character Time Units; T = ______ μs**): .... ________

16. HORIZ. SCAN DELAY (No. in Character Time Units; T = ______ μs**): .... ________

17. TOTAL CHARACTER TIME UNITS IN (1) HORIZ. SCAN LINE
    (Add Steps 13 thru 16): .... ________

18. CHARACTER RATE (Step 12 x Step 17 = Freq. in MHz): .... ________

19. CLOCK (DOT) RATE (Step 3 x Step 18 = Freq. in MHz): .... ________

*Vertical Interval
**Horizontal Interval

MK3807 VCU WORK SHEET

Table 2B

	ADDRESS				
REG. #	**A3—A0**	**FUNCTION**	**BIT ASSIGNMENT**	**HEX.**	**DEC.**
0	0000	HORIZ. LINE COUNT ______	□□□□□□□□	______	______
1	0001	INTERLACE ______ H SYNC WIDTH ______ H SYNC DELAY ______	□□□□□□□□	______	______
2	0010	SCANS/DATA ROW ______ CHARACTERS/ROW ______	☒□□□□□□□	______	______
3	0011	SKEW CHARACTERS ______ DATA ROWS ______	□□□□□□□□	______	______
4	0100	SCANS/FRAME ______ X = ______	□□□□□□□□	______	______
5	0101	VERTICAL DATA START = 3 + VERTICAL SCAN DELAY: SCAN DELAY ______ DATA START ______	□□□□□□□□	______	______
6	0110	LAST DISPLAYED DATA ROW (= DATA ROWS)	☒☒□□□□□□	______	______

MK3807 VCU WORK SHEET

Table 2C

1. H CHARACTER MATRIX (No. of Dots): 7

2. V CHARACTER MATRIX (No. of Horiz. Scan Lines): 9

3. H CHARACTER BLOCK (Step 1 + Desired Horiz. Spacing = No. in Dots): 9

4. V CHARACTER BLOCK (Step 2 + Desired Vertical Spac.ıg = No. in Horiz. Scan Lines): 12

5. VERTICAL FRAME (REFRESH) RATE (Freq. in Hz): 60

6. DESIRED NO. OF CHARACTER ROWS: 24

7. TOTAL NO. OF ACTIVE "VIDEO DISPLAY SCAN LINES" (Step 4 x Step 6 = No. in Horiz. Scan Lines): 288

8. VERT. SYNC DELAY (No. in Horiz. Scan Lines): 0

9. VERT. SYNC (No. in Horiz. Scan Lines; T = 161.29 µs*): 3

10. VERT. SCAN DELAY (No. in Horiz. Scan Lines; T = 1.02 ms*): 19

11. TOTAL VERTICAL FRAME (Add steps 7 thru 10 = No. in Horiz. Scan Lines): 310

12. HORIZONTAL SCAN LINE RATE (Step 5 x step 11 = Freq. in KHz): 18.6

13. DESIRED NO. OF CHARACTERS PER HORIZ. ROW: 80

14. HORIZ. SYNC DELAY (No. in Character Time Units; T = 2.11 µs**): 4

15. HORIZ. SYNC (No. in Character Time Units; T = 4.74 µs**): 9

16. HORIZ. SCAN DELAY (No. in Character Time Units; T = 4.74 µs**): 9

17. TOTAL CHARACTER TIME UNITS IN (1) HORIZ. SCAN LINE (Add Steps 13 thru 16): 102

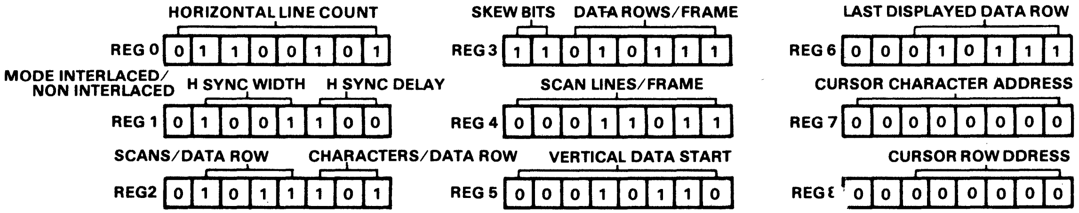
18. CHARACTER RATE (Step 12 x Step 17 = Freq. in MHz): 1.8972

19. CLOCK (DOT) RATE (Step 3 x Step 18 = Freq. in MHz): 17.0748

*Vertical Interval
**Horizontal Interval

BIT ASSIGNMENT

Figure 13



XTAL Frequency

At a frequency of 18.6KHz a scan line takes 53.76 μs. In this time 102 characters (80 displayed + 22 blanked) have to be accessed. Thus the character time is 527.06 ns (53.76 μs/102). Since each character is 9 dots in this example (7 character and 2 blank), the dot period is 58.56 ns (527.06 ns/9). The inverse of the dot period is the video dot clock XTAL frequency. For this example, the video dot clock XTAL is 1/58.56 ns = 17.0748 MHz (53.76 μs/102). Since each character is 9 dots in this example (7 character and 2 blank), the dot period increases in the video clock rate, possibly to a point beyond the monitor's specification.

A more detailed example, using 40 character by 12 row format, follows.

Having chosen the display format and display monitor, the actual settings for the VCU registers can now be established. See Table 2C.

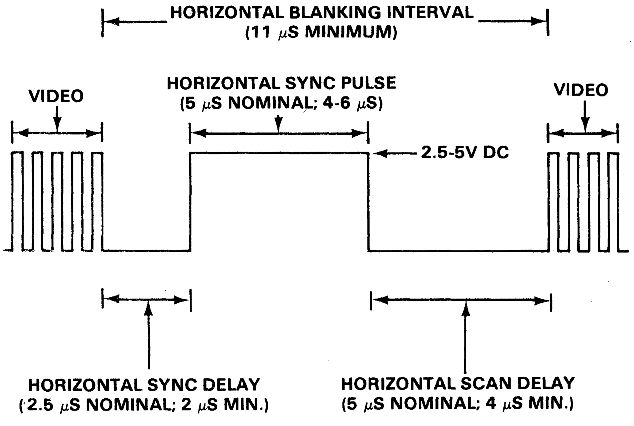
EXAMPLE FOR 40 CHARACTER BY 12 ROWS

Using the VCU worksheet (Table 2A), steps 1 and 2 determine the character matrix. In this example, a 7 X 9 dot matrix will be used, thus in step 1, 7 dots are used horizontally and in step 2, 9 scan lines are used vertically. This defines the character size (other character sizes might be 5 X 7 etc.). Steps 3 and 4 determine the character block size. The character block is composed of the character matrix along with both the horizontal and vertical blank spaces between characters. Step 3 shows the H character block for this example to be 7 dots from step 1 plus 2 additional dots for blank space, giving a total of 9. Step 4 shows the vertical height (V character block) being 9 lines from step 2, plus 3 additional raster lines for vertical spacing, giving a total of 12. The next parameter is the vertical frame refresh rate and this example uses the American Standard of 60Hz (in this example the non-interlace mode will also be used).

As this example uses twelve rows of data, step 6 indicates 12. Step 7 determines the number of active video display raster scan lines. This is determined by taking the number of raster scan lines from step 4 and multiplying that by the number of data rows in step 6, thus giving us the number of displayed horizontal scan lines. In this example, multiply 12 raster lines per data row by 12 data rows to give 144 active video raster scan lines.

The next portion of this example is dependent upon the characteristics of the video monitor being used. For the purposes of this example a standard sync driven video monitor using RS-170 non-interlace sync is used. In accordance with the standard for this monitor, the vertical sync pulse width will be between 180 and 200 μs. with 190 μs. as the nominal value. In addition, the vertical blanking interval, which is made up of the vertical sync pulse and the 2 delays , is defined as being 1 ms. minimum. The same monitor specification defines the horizontal sync pulse width as being between 4 and 6 μs. with 5 μs. as the nominal horizontal sync pulse width. In addition, the horizontal sync delay or front porch is defined as 2.5 μs.

MONITOR HORIZONTAL TIMING
Figure 14

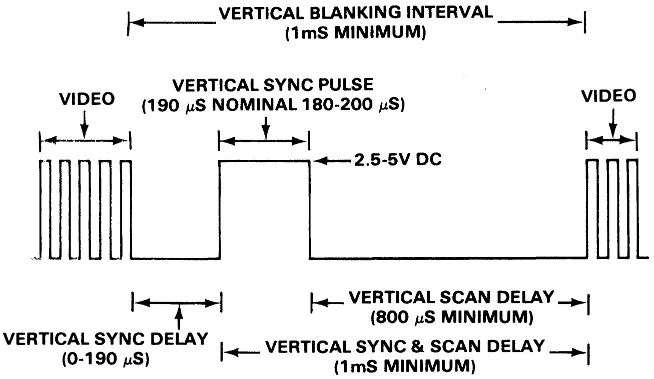


nominally with a 2 µs. minimum. At the same time, the horizontal blanking interval, which is composed of the front porch, horizontal sync pulse, and the back porch is defined as 11 µs. minimum. See Figures 14 and 15.

The monitor characteristics determine the values for steps 9 and 10. Step 9 lists the vertical sync pulse width. The VCU has a fixed vertical sync pulse width of 3 horizontal raster scan lines (3H). Later, the period of a horizontal raster scan line will be determined and verified that this meets the RS-170 specification. Enough time must be allowed for vertical retrace and some blanking at the top of the screen. This is indicated in step 10 as the vertical scan delay. The VCU can be programmed for a vertical scan delay between 0 and 255 raster scan lines to allow utilization of various types of monitors, as well as to position the data vertically on the screen. For purposes of this example, a vertical scan delay of 19 raster lines is chosen. After the horizontal period is determined, it can be verified that these values comply with the specification. Step 11 is the total number of raster lines per frame or, in other words, the number of raster lines per vertical refresh time. Normally, this will be determined by adding to the number of displayed scan lines, the vertical sync pulse width, the vertical scan delay, and the vertical sync delay which has not yet been determined. However, in this case, since the example uses a standard monitor, it is possible to work backwards. Therefore, for step 11 we will enter 262 raster lines per frame (a typical number of raster lines/field of a standard monitor). Now work backwards to step 8 and determine the vertical sync delay. This is the number of raster lines between the last displayed video raster line and the beginning of vertical sync. Subtracting 144, 19, and 3 from 262 leaves 96, thus for step 8, 96 horizontal lines is the vertical sync delay. We have now determined the vertical timing waveform for this example. The next part of the example is to determine the horizontal scan line rate or how many raster lines per second will be displayed. This is determined by multiplying the vertical frame refresh rate from step 5; in this case 60 frames per second by the total number of raster lines per frame from step 11, in this case 262. The product will be 15,720 raster lines per second. This is the horizontal scan rate. The horizontal period is determined by taking the inverse of horizontal scan rate, 1 divided by 15,720 Hz is 63.6132 µs. This is the time of 1 horizontal raster line. This information is now used to go back and check on meeting the specifications in steps 9 and 10. Step 9 lists 3 horizontal lines as the vertical sync pulse width. 3 X 63.6132 µs. yields 190.84 µs. This is the nominal value specified for the monitor. Step 10 lists the vertical scan delay as 19 raster lines multiplying that by 63.61 µs. yields 1.21 ms., thus the values picked for the above parameters meet the specification for the monitor.

In step 13 the desired number of active display characters per horizontal data row is listed. 40 character per row have been chosen. Steps 14, 15 and 16 are now selected using the horizontal period and the monitor specifications. Step 14 is the horizontal sync delay or front porch. In this case 2 character times. The period of a character will be determined later in this example which will be used to verify that this parameter meets the RS-170 specification given earlier. In step 15 the horizontal sync width is chosen to be 4 character times and in step 16 the horizontal scan delay is

MONITOR VERTICAL TIMING
Figure 15



chosen to also be 4 character times. Step 17 is the total number of character times per horizontal scan line and this is determined by adding steps 13 through 16, thus we add 40+2+4+4=50 character times per horizontal scan line. In step 18 the character rate is determined by multiplying the horizontal line rate of step 12 by the total character units per horizontal line, thus, 15,720 X 50 = 786,000 characters per second. The character period is the inverse of the character rate, thus 1 over 786,000 yields a character period of 1.272 μs. This information is used to verify steps 14, 15, and 16. In step 14 the horizontal sync delay was chosen as 2 character units. 2 times 1.272 μs yields 2.54 μs. Step 15, the horizontal sync width was 4 character units. 4 times 1.272 μs. yields 5.089 μs. and similarly, step 16, four character units also is 5.089 μs. These three values are in agreement with the specification for the monitor. The next step is to determine the video dot clock frequency. It is determined by multiplying the number of dots per character from step 3 by the character rate in step 18, 9 X 786 KHz = 7.074 MHz. Thus, the crystal frequency required for this example is 7.074 MHz and the dot clock counter divisor N is 9 (from step 3).

Register Programming

Register 0 (Horizontal Line Count) determines the total number of character units per horizontal line. From step 17 we have determined that there would be 50 character units per line. This register is loaded with (N — 1) the decimal number 49.

Register 1 contains 3 fields. The first field is the most significant bit and this determines the interlaced or non-interlaced mode of operation. This example uses the non-interlaced mode, therefore, bit 7 is loaded with a 0. The next field is the horizontal sync pulse width and this field is bits 6 through 3. Step 15 determines that the horizontal sync width is 4 character times. Therefore the binary equivalent of 4 is loaded into these bits. Thus bits 6 through 3 are loaded with 0100. The third field is the horizontal sync delay, step 14 determines that this is 2 character time units. Therefore, bits 2 through 0 are loaded with 010.

Register 2 contains 2 fields, with the most significant bit unused. Bits 6 through 3 determine the scans per data row. In this example from step 4, there will be 12 raster lines per data row, and from the VCU data sheet note this is an N + 1 register. Therefore the decimal number eleven is loaded into bits 6 through 3. the second field is characters per data row, bits 2 through 0. In this example 40 active characters per data row was chosen. The VCU data sheet specifies that 010 in this field will give 40 characters per data row, thus bits 2 through 0 are loaded with 010.

Register 3 also contains 2 fields. The first field, bits 7 and 6, are the skew bits. These bits allow the hardware designer to

MK3807 VCU WORK SHEET

1. H CHARACTER MATRIX (No. of Dots): 7
2. V CHARACTER MATRIX (No. of Horiz. Scan Lines): 9
3. H CHARACTER BLOCK (Step 1 + Desired Horiz. Spacing = No. in Dots): 9
4. V CHARACTER BLOCK (Step 2 + Desired Vertical Spacing = No. in Horiz. Scan Lines): 12
5. VERTICAL FRAME (REFRESH) RATE (Freq. in Hz): 60
6. DESIRED NO. OF CHARACTER ROWS: 12
7. TOTAL NO. OF ACTIVE "VIDEO DISPLAY SCAN LINES" (Step 4 x Step 6 = No. in Horiz. Scan Lines): 144
8. VERT. SYNC DELAY (No. in Horiz. Scan Lines): 96
9. VERT. SYNC (No. in Horiz. Scan Lines; T = 190.84 μs*): 3
10. VERT. SCAN DELAY (No. in Horiz. Scan Lines; T = 1.21 ms*): 19
11. TOTAL VERTICAL FRAME (Add steps 7 thru 10 = No. in Horiz. Scan Lines): 262
12. HORIZONTAL SCAN LINE RATE (Step 5 x step 11 = Freq. in KHz): 15.72
13. DESIRED NO. OF CHARACTERS PER HORIZ. ROW: 40
14. HORIZ. SYNC DELAY (No. in Character Time Units; T = 2.54 μs**): 2
15. HORIZ. SYNC (No. in Character Time Units; T = 5.09 μs**): 4
16. HORIZ. SCAN DELAY (No. in Character Time Units; T = 5.09 μs**): 4
17. TOTAL CHARACTER TIME UNITS IN (1) HORIZ. SCAN LINE (Add Steps 13 thru 16): 50
18. CHARACTER RATE (Step 12 x Step 17 = Freq. in MHz): .... .786
19. CLOCK (DOT) RATE (Step 3 x Step 18 = Freq. in MHz): 7.074

*Vertical Interval
**Horizontal Interval

use a slower buffer RAM memory and allow compensation for slower character generator access times. In the example shown as well as most typical applications, these bits are set for 2 character time delays, therefore bit 7 and bit 6 will both contain a 1. The other field is data rows per frame, bits 5 through 0. In Step 6 there are 12 data rows per frame, and the VCU data sheet specifies that this is an N + 1 register. Thus the decimal number eleven is loaded in bits 5 through 0.

Register 4 determines the number of horizontal raster lines per frame. From this example, step 11, specifies there are 262 raster lines per frame. The VCU data sheet specifies that there are two modes of loading this register. In the non-interlace mode (this example) the equation 2X + 256 is equal to 262. Thus, X is equal to 3. The decimal number 3 is loaded into register 4.

Register 5 is the vertical start of data. From steps 9 and 10 in the example the vertical data start is 22 raster lines, thus the decimal number 22 is loaded into register 5.

Register 6 is the last displayed data row. This register is used for multi-line scrolling and for initialization purposes is set to the same data as in register 3, the data rows per frame. Thus, the decimal number eleven is loaded into register 6.

The following will illustrate the use of register 6 for multi-line scrolling:

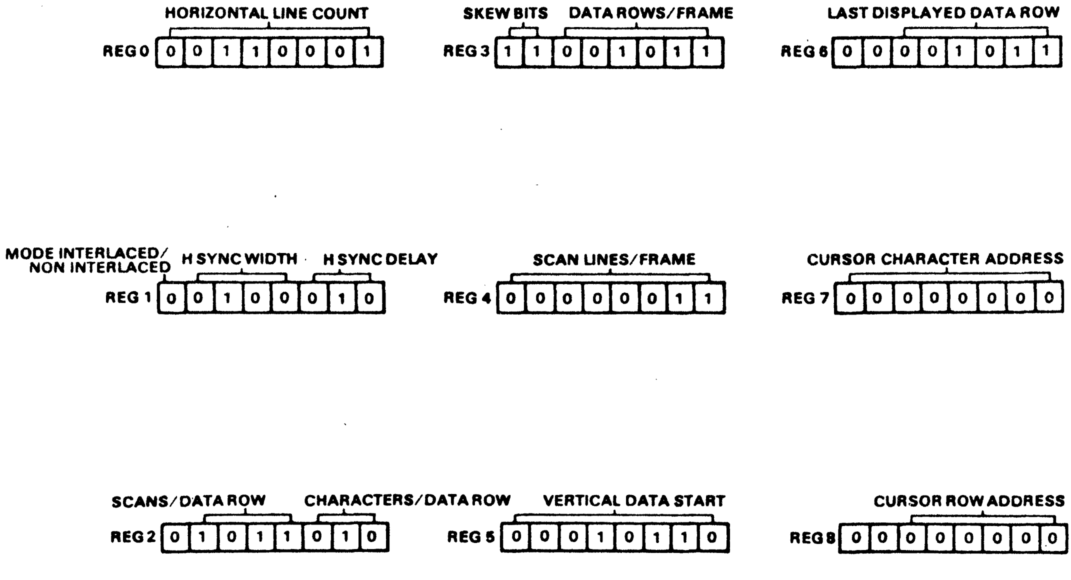
> Using 12 rows of data with row 0 on top of the screen and row 11 on the bottom and as programmed in register 6 with eleven, this will be the case. Now, if another number is programmed into register 6, such as 5, data row 5 will be on the bottom of the screen, while data row 6 will be on the top followed by data row 7, 8, through to 11, followed by row 0 through 5.

Register 7 is the cursor character address. It is initialized to 0, thus it is now set to the beginning of the data row.

Register 8 is also initialized to 0. This is the cursor row address and is set to the top data row. The 2 cursor addresses (X-Y) coincide at the upper left hand corner of the screen. See the VCU work sheet on page 16.

The above is only a typical example of how to determine the frequencies, program the frequencies, and program the registers of the VCU. This is shown for illustrative purposes only and designers/programmers should determine these values for their specific CRT requirements.

BIT ASSIGNMENT CHART



HORIZONTAL LINE COUNT
REG 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |

SKEW BITS DATA ROWS/FRAME
REG 3 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |

LAST DISPLAYED DATA ROW
REG 6 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |

MODE INTERLACED/ NON INTERLACED H SYNC WIDTH H SYNC DELAY
REG 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |

SCAN LINES/FRAME
REG 4 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

CURSOR CHARACTER ADDRESS
REG 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SCANS/DATA ROW CHARACTERS/DATA ROW
REG 2 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |

VERTICAL DATA START
REG 5 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |

CURSOR ROW ADDRESS
REG 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

APPLICATION NOTES

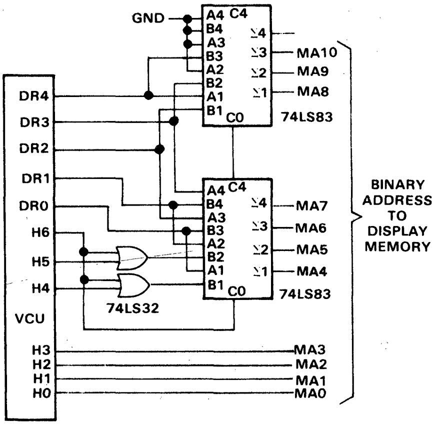
Conversion of Row Column to Binary Address

With only slightly more complicated circuitry than required by memory mapping, the row column addressing outputs of the VCU may be readily changed to binary address outputs. For data formats that use 48 or 80 visible characters per data row, this can be done by the addition of two 74LS83's and a 74LS32 (or equivalent) in some formats or by the addition of the one 256x8 PROM. Figure 16 below shows the implementation for an 80 character by 24 data row display using the adders. Figure 17 is an implementation using a bipolar PROM.

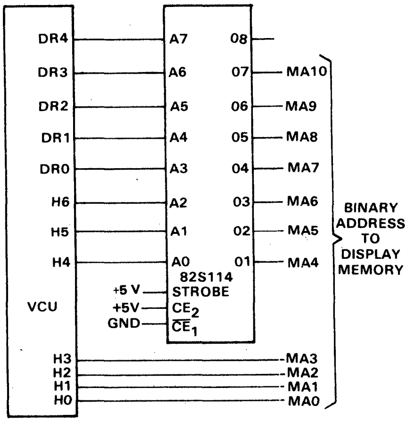
In essence the adders are used to add groups of 16. Since there are 5 groups of 16 in each data row of 80 characters, the adders effectively multiply the data row count (DR0-DR4) by 5 to obtain the starting binary address for each row. This is done by adding DR0-DR4 to itself shifted two positions to the left. Within each data row, H6, H5, and H4 are used to add from 0 to 4 groups of 16. The PROM configuration is merely a table look-up implementation of the adder configuration.

The PROM configuration can be programmed to provide binary addresses for any number of groups of 16 characters per data row (i.e., 48, 80, 96, 112, 144, 160). Table 3 shows some typical mapping for an 80x24 display.

80x24 DISPLAY WITH BINARY ADDRESS USING 74LS83 ADDERS
Figure 16



80x24 DISPLAY WITH BINARY ADDRESS USING 256x8 PROM
Figure 17



TYPICAL MAPPING OF 80x24 DISPLAY
Table 3

ADDRESS TABLE

DR4	DR3	DR2	DR1	DR0	H6	H5	H4	H3	H2	H1	H0	ROW	COL	MA10	MA9	MA8	MA7	MA6	MA5	MA4	MA3	MA2	MA1	MA0	ADDR. (BIN)
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	16	0	0	0	0	0	0	1	0	0	0	0	16
0	0	0	0	0	1	0	0	1	1	1	1	0	79	0	0	0	0	1	0	0	1	1	1	1	79
0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0	0	0	80
0	0	0	0	1	1	0	0	1	1	1	1	1	79	0	0	0	1	0	0	1	1	1	1	1	159
0	0	0	1	0	0	0	0	0	0	0	0	2	0	0	0	0	1	0	1	0	0	0	0	0	160
0	0	0	1	0	1	0	0	1	1	1	1	2	79	0	0	0	1	1	1	0	1	1	1	1	239
0	0	0	1	1	0	0	0	0	0	0	0	3	0	0	0	0	1	1	1	1	0	0	0	0	240
1	0	1	1	1	0	0	0	0	0	0	0	23	0	1	1	1	0	0	1	1	0	0	0	0	1840
1	0	1	1	1	1	0	0	1	1	1	1	23	79	1	1	1	0	1	1	1	1	1	1	1	1919

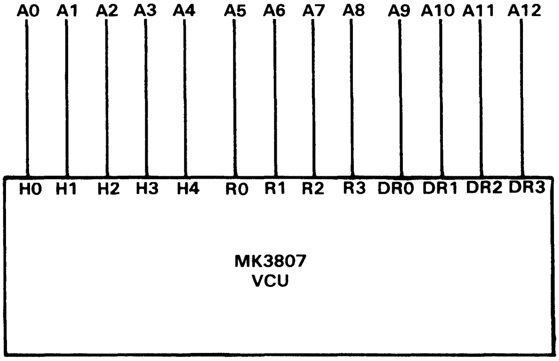
USING THE VCU FOR A 256 X 256 DOT GRAPHIC DISPLAY

The VCU can be used for dot matrix graphic displays as well as alphanumeric displays. The following is an example of a 256 x 256 dot matrix graphic display using the raster line counter outputs (R0-R3) as part of the RAM addressing.

For this example the character width (the dot counter divisor) should be 8 dots. The VCU should be programmed (See Figure 18) for:

Characters per data row = 32
Scans per data row = 16
Data rows per frame = 16

USING THE VCU FOR A 256 x 256 DOT GRAPHIC DISPLAY
Figure 18



A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12
H0 H1 H2 H3 H4 R0 R1 R2 R3 DR0 DR1 DR2 DR3
MK3807
VCU

USING THE VCU FOR MORE THAN 128 CHARACTERS PER ROW AND MORE THAN 32 ROWS

Due to pin limitation, the most significant character count output of the VCU is multiplexed with the most significant bit of the data row counter. When the horizontal line count is greater than 128, this output (H7/DR5) automatically becomes H7. On the surface, this creates a limitation of no more than 32 data rows.

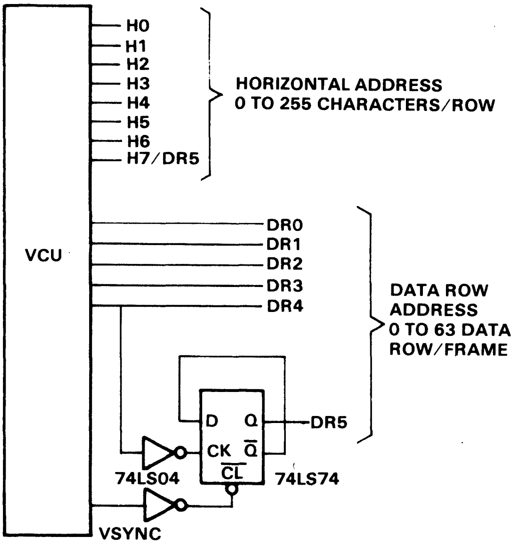
In actual fact, the row column addressing of the VCU permits the display of more than 128 characters per row and more than 32 rows per frame with only two inverters and one D-type flip flop. In the following example, the display format will be 132 characters per row by 35 data rows.

The horizontal row address will appear on outputs H0 to H7. Data row outputs DR0 to DR4 will provide five of the six bits required for the data row addressing. The circuit shown in Figure 19 will generate the required sixth row address bit.

There are many other applications of the VCU other than the alphanumeric CRT terminal as shown above.

Because of the speed and flexibility of the device, it can be used to generate television pictures (with gray scale and color), facsimile, slow-scan TV, frame storage, scan conversion, etc. Since the VCU generates composite sync (with serrations), the serial video can be combined with the composite sync to produce composite video (RS-170).

USING THE VCU FOR MORE THAN 128 CHARACTERS PER ROW AND MORE THAN 32 ROWS
Figure 19



VI
Z80
MICRO-
COMPUTER
APPLICATION
NOTES

1981 Z80 MICROCOMPUTER DATA BOOK

I Table of Contents

II General Information

III Z80 Family Technical Manuals

IV Z80 Family Data Sheets

V Z80 Development Equipment

VI Z80 Microcomputer Application Notes

VII Z80 Microcomputer Peripherals

VIII Z80 Military/ Hi-Rel

I TABLE OF CONTENTS

II GENERAL INFORMA-TION

III Z80 FAMILY TECHNICAL MANUALS

IV Z80 FAMILY DATA SHEETS

V Z80 DEVELOP-MENT EQUIPMENT

VI Z80 MICRO-COMPUTER APPLICATION NOTES

VII Z80 MICRO-COMPUTER PERIPHERALS

VIII Z80 MILITARY/ HI-REL

ADVANCE INFORMATION

MOSTEK®

CMOS MICROCOMPUTER CLOCK/RAM

MK3805N

FEATURES

□ Real-time clock counts seconds, minutes, hours, date of the month, day of the week, month, and year. Every 4th year, February has 29 days.

□ Serial I/O for minimum pin count (8 pins)

□ 24 x 8 RAM for scratchpad data storage

□ Simple Microcomputer interface

□ High speed shift clock independent of crystal oscillator frequency

□ Single byte or multiple byte (Burst Mode) data transfer capability for read or write of clock or RAM data.

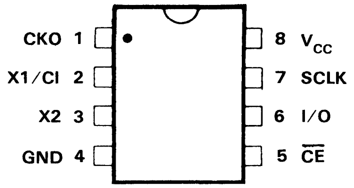
□ TTL Compatible (V_{CC} = 5V)

□ Low-power CMOS

□ $I_{CC} \leq 2mA$ (V_{CC} = 5 V)

□ $+3V \leq V_{CC} \leq 9.5V$

PIN OUT



PIN	NAME	DESCRIPTION
1	CKO	Buffered System Clock Output
2	X1/CI	Crystal or External Clock Input
3	X2	Crystal Input
4	GND	Power Supply Pin
5	$\overline{CE}$	Chip Enable for Serial I/O Transfer
6	I/O	Data Input/Output Pin
7	SCLK	Shift Clock for Serial I/O Transfer
8	V_{CC}	Power Supply Pin

GENERAL DESCRIPTION

Many microprocessor applications require a real-time clock and/or memory that can be battery powered with very low power drain. The MK3805N is specifically designed for these applications. The device contains a real-time clock/calendar, 24 bytes of static RAM, an on-chip oscillator, and it communicates with the microprocessor via a simple serial interface. The MK3805N is fabricated using CMOS technology, thus insuring very low power consumption.

The real-time clock/calendar provides seconds, minutes, hours, day, date, month, and year information to the microprocessor. The end of the month date is automatically adjusted for months with less than 31 days, including correction for leap year every 4 years. The clock operates in either the 24 hour or 12 hour format with an AM/PM indicator.

The on-chip oscillator provides a real-time clock source for the clock/calendar. It incorporates a programmable divider so that a wide variety of crystal frequencies can be accommodated. The oscillator also has an output available that can be connected to the microprocessor clock input. A separately programmable divider provides several different output frequencies for any given crystal frequency. This feature can eliminate having to use a separate crystal or external oscillator for the microprocessor, thereby reducing system cost.

Interfacing the CLOCK/RAM with a microprocessor is greatly simplified using asynchronous serial communication. Only 3 lines are required to communicate with the CLOCK/RAM: (1) $\overline{CE}$ (chip enable), (2) I/O (data line) and (3) SCLK (shift register clock). Data can be transferred to and from the CLOCK/RAM one byte at a time or in a burst of up to 24 bytes.

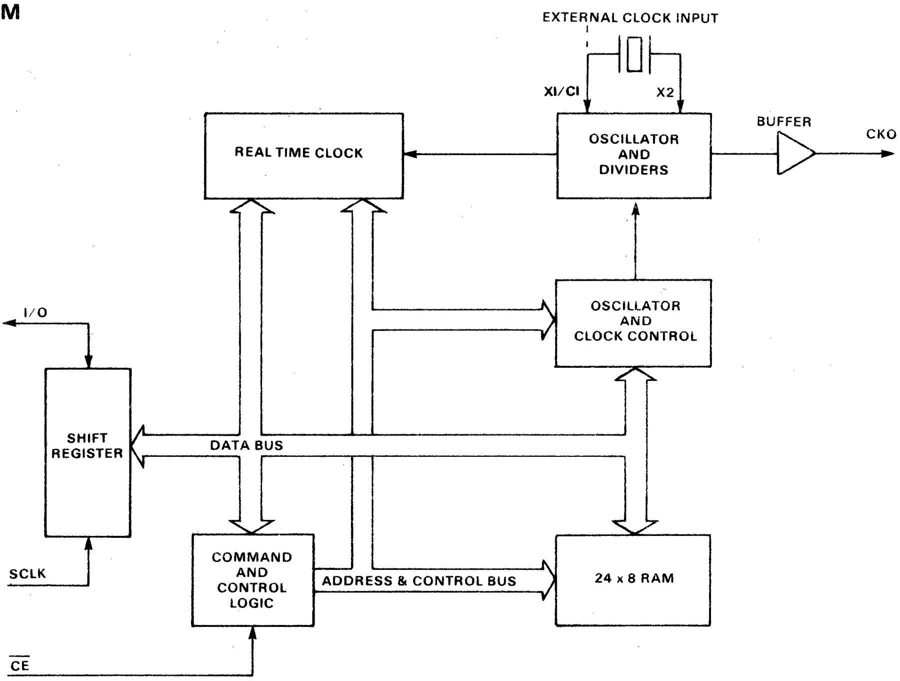
TECHNICAL DESCRIPTION

Figure 1 is a block diagram of the CLOCK/RAM chip. Its main elements are the oscillator and divider circuit, the real-time clock/calendar, static RAM, the serial shift register, and the command and control logic.

The shift register is used to communicate with the outside world. Data on the I/O line is either input or output on each shift register clock pulse when the chip is enabled. If the chip is in the input mode, the data on the I/O line is input to

BLOCK DIAGRAM

Figure 1



the shift register on the rising edge of SCLK. If in the output mode, data is shifted out onto the I/O line on the falling edge of SCLK.

The command and control logic receives the first byte input by the shift register after $\overline{CE}$ goes active. This byte must be the command byte and will direct further operations within the CLOCK/RAM. The command specifies whether subsequent transfers will be data input or data output, and which register or RAM location will be involved.

A control register provides programmable control of the divider for the internal clock signal, the external clock signal, the crystal type and mode, and the write protect function, which is useful during power-up and power-down conditions.

The real-time clock/calendar is accessed via seven registers. These registers control seconds, minutes, hours, day, date, month, and year. Certain bits within these registers also control a run/stop function, 12/24 hour format, and indicate AM or PM (12 hour mode only). These registers can be accessed sequentially in Burst Mode, or randomly in a single byte transfer.

The static RAM is organized as 24 bytes of 8-bits each. They can be accessed either sequentially in burst mode, or randomly in a single byte transfer.

DATA TRANSFER

Data Transfer is accomplished under control of the $\overline{CE}$ and SCLK inputs by an external microcomputer. Each transfer consists of a single byte ADDRESS/COMMAND input followed by a single byte or multiple byte (if Burst Mode is specified) data input or output, as specified by the ADDRESS/COMMAND byte. The serial data transfer occurs with LSB first, MSB last format.

ADDRESS/COMMAND BYTE

The ADDRESS/COMMAND Byte is shown below:

7	6	5	4	3	2	1	0
1	RAM/$\overline{CK}$	A4	A3	A2	A1	A0	Rd/$\overline{W}$

As defined, the MSB (bit 7) must be a logical 1; bit 6 specifies a Clock/Calendar/Control register if logical 0 or a RAM register if logical 1; bits 1-5 specify the designated register(s) to be input or output; and the LSB (bit 0) specifies a WRITE operation (input) if logical 0 or READ operation (output) if logical 1.

BURST MODE

Burst Mode may be specified for either the Clock/Calendar/Control registers or for the RAM registers by addressing location 31 (ADDRESS/COMMAND bits 1-5 = logical 1). As before, bit 6 specifies Clock or RAM and bit 0 specifies read or write.

There is no data storage capability at location 31 in either the Clock/Calendar/Control registers or the RAM registers.

SCLK AND $\overline{CE}$ CONTROL

All data transfers are initiated by $\overline{CE}$ going low. After $\overline{CE}$ goes low, the next 8 SCLK cycles input an ADDRESS/COMMAND byte of the proper format. If bit 7 is not a logical 1, indicating a valid CLOCK/RAM ADDRESS/COMMAND, the ADDRESS/COMMAND byte is ignored as are all SCLK cycles until $\overline{CE}$ goes high and returns low to initiate a new ADDRESS/COMMAND transfer. See Figure 2.

ADDRESS/COMMAND bits and DATA bits are input on the rising edge of SCLK, and DATA bits are output on the falling edge of SCLK.

A data transfer terminates if $\overline{CE}$ goes high, and the transfer must be reinitiated by the proper ADDRESS/ COMMAND when $\overline{CE}$ again goes low. The data I/O pin is high impedance when $\overline{CE}$ is high.

DATA INPUT

Following the 8 SCLK cycles that input the WRITE Mode ADDRESS/COMMAND byte (bit 0 = logical 0), a DATA byte is input on the rising edge of the next 8 SCLK cycles (per byte, if Burst Mode is specified). Additional SCLK cycles are ignored should they inadvertently occur.

DATA OUTPUT

Following the 8 SCLK cycles that input the READ Mode ADDRESS/COMMAND byte (bit 0 = logical 1), a DATA byte is output on the falling edge of the next 8 SCLK cycles (per byte, if Burst Mode is specified). Additional SCLK cycles retransmit the data byte(s) should they inadvertently occur, so long as $\overline{CE}$ remains low. This operation permits continuous Burst Read Mode capability.

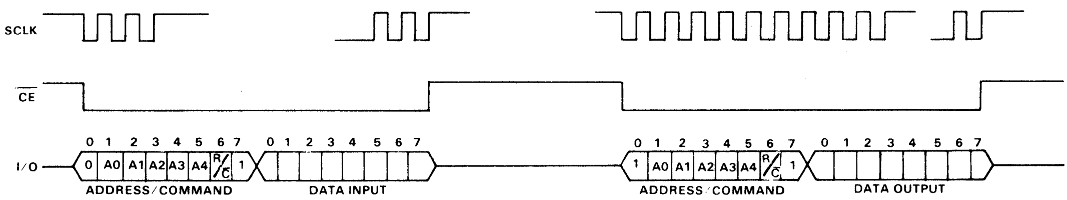
DATA TRANSFER SUMMARY

A data transfer summary is shown in Figure 2.

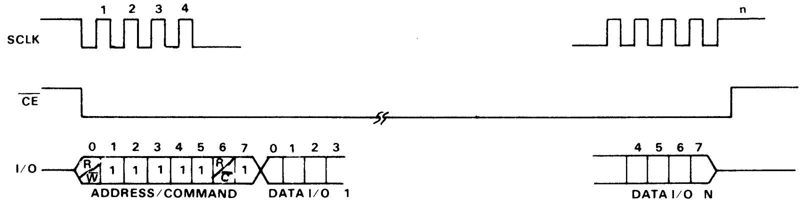
DATA TRANSFER SUMMARY

Figure 2

I. Single Byte Transfer



II. Burst Mode Transfer



FUNCTION	N	n
CLOCK	8	72
RAM	24	200

NOTES

1) Data input sampled on rising edge of clock
2) Data output changes on falling edge of clock
3) Rising edge of $\overline{CE}$ terminates operation and resets address/command

REGISTER DEFINITION

CLOCK/CALENDAR

The Clock/Calendar is contained in 7 addressable/writeable/readable registers, as defined below.

Address	Function	Range (BCD)
0	Seconds+Clock Halt Flag	00-59
1	Minutes	00-59
2	Hours/AM-PM/12-24 Mode	00-23 or 01-12
3	Date	01-28,29, 30,31
4	Month	01-12
5	Day	01-07
6	Year	00-99

Data contained in the Clock/Calendar registers is in binary coded decimal format (BCD).

CLOCK HALT FLAG

Bit 7 of the Seconds Register is defined as the Clock Halt Flag. Bit 7 = logical 1 inhibits the 1 Hz input to the Clock/Calendar. Bit 7 is set to logical 1 on power-up to prevent counting, and it may be set high or low by writing to the seconds register under normal operation of the device.

AM-PM/12-24 MODE

Bit 7 of the Hours Register is defined as the 12 or 24 hour mode select bit. In the 12-hour mode, bit 5 is the AM/PM bit, and in the 24-hour mode, bit 5 is the second 10-hour bit (20-23 hours).

TEST MODE BITS

Bit 7 of the Date Register and Bit 7 of the Day Register are Test Mode Bits utilized in testing the MK3805. These bits should be logic 0 for normal operation.

CONTROL REGISTER

The Control Register specifies the crystal mode/frequency to be used, the system clock output frequency, and the WRITE PROTECT Mode for data protection. The Control Register is located at address 7 in the Clock/Calendar/Control address space.

7	6	5	4	3	2	1	0
WP	C1	C0	X4	X3	X2	X1	X0

CRYSTAL DIVIDER MODE

X4 and X3 specify the Crystal frequency divider mode selected.

X4	X3	Xtal Mode	Primary Frequencies
0	0	Binary	2^{22}, 2^{21}, 2^{20} Hz
0	1	Microprocessor	8, 5, 4, 2.5, 2, 1.25, 1 MHz
1	0	Baud Rate	7.3728, 3.6864, 1.8432 MHz
1	1	Color Burst	3.5795 MHz

CRYSTAL DIVIDER PRESCALER

X2, X1, and X0 specify a particular prescaler divider selection necessary to generate a 1 Hz frequency for the Clock/Calendar. Refer to Figure 4 for complete definition.

SYSTEM CLOCK OUTPUT

C1 and C0 designate the system clock output frequency selected. The options are X, X/2, X/4, and ~2 kHz. When in the Binary Mode, the output frequency is 2048 Hz. In any other mode the output frequency is ~2048 Hz. Refer to Figure 5 for complete definition.

WRITE PROTECT

Bit 7 of the Control Register is the WRITE PROTECT Flag. Bit 7 is set to logical 1 on power-up, and it may be set high or low by writing to the Control Register. When high, the WRITE PROTECT Flag prevents a write operation to any internal register, including the other bits of the Control Register. Further, logic is included such that the WRITE PROTECT bit may be reset to a logic 0 by a Write operation without altering the other bits of the Control Register.

CLOCK/CALENDAR/CONTROL BURST MODE

Address 31 of the Clock/Calendar/Control Address space specifies Burst Mode operation. In this mode, the 7 Clock/Calendar Registers and the Control Register may be consecutively read or written. Addresses above address 7 (Control Register) are non-existent; only addresses 0-7 are accessible.

RAM

The static RAM is contained in 24 addressable/writeable/readable registers, addressed consecutively in the RAM address space beginning at location 0.

RAM BURST MODE

Address 31 of the RAM address space specifies Burst Mode operation. In this mode, the 24 RAM registers may be consecutively read or written. Addresses above the maximum RAM address location are non-existent and are not accessible.

REGISTER SUMMARY

A Register, Data Format summary is shown in Figure 3.

MICROCOMPUTER CLOCK/RAM ADDRESS/COMMAND, REGISTER, DATA FORMAT SUMMARY

Figure 3

I. ADDRESS/COMMAND FORMAT

7	6	5	4	3	2	1	0
1	RAM/$\overline{CK}$	A_4	A_3	A_2	A_1	A_0	RD/$\overline{W}$

II. REGISTER ADDRESS

A. CLOCK

REGISTER DEFINITION

	7	6	5	4	3	2	1	0		7			4	3			0	POWER ON RESET
SEC	1	0	0	0	0	0	0	RD/$\overline{W}$	00-59	CH	10 SEC			SEC				80
MIN	1	0	0	0	0	0	1	RD/$\overline{W}$	00-59	0	10 MIN			MIN				00
HR	1	0	0	0	0	1	0	RD/$\overline{W}$	01-12 00-23	12/ 24	0	10 A/P	HR	HR				00
DATE	1	0	0	0	0	1	1	RD/$\overline{W}$	01-28/29 01-30 01-31	T_1	0	10 DATE		DATE				01
MONTH	1	0	0	0	1	0	0	RD/$\overline{W}$	01-12	0	0	0	10 M	MONTH				01
DAY	1	0	0	0	1	0	1	RD/$\overline{W}$	01-07	T_2	0	0	0	0	DAY			01
YEAR	1	0	0	0	1	1	0	RD/$\overline{W}$	0-99	10 YEAR				YEAR				00
CONTROL	1	0	0	0	1	1	1	RD/$\overline{W}$		WP	C_1	C_0	X_4	X_3	X_2	X_1	X_0	A0
CLOCK BURST	1	0	1	1	1	1	1	RD/$\overline{W}$										

B. RAM

RAM 0	1	1	0	0	0	0	0	RD/$\overline{W}$	RAM DATA 0									XX
					● ● ●									● ● ●				
RAM 23	1	1	1	0	1	1	1	RD/$\overline{W}$	RAM DATA 23									XX
RAM BURST	1	1	1	1	1	1	1	RD/$\overline{W}$										

CRYSTAL FREQUENCY SELECTION TABLE

Figure 4

X4	X3	X2	X1	X0	f_{XTAL} (MHz) Crystal Frequency	Comments
0	0	0	0	0	8.388608	Power on condition
0	0	0	0	1	8.388608	
0	0	0	1	0	4.194304	
0	0	0	1	1	4.194304	
0	0	1	0	0	2.097152	
0	0	1	0	1	2.097152	
0	0	1	1	0	1.048576	
0	0	1	1	1	0.032768	
0	1	0	0	0	8.000000	
0	1	0	0	1	5.000000	
0	1	0	1	0	4.000000	
0	1	0	1	1	2.500000	
0	1	1	0	0	2.000000	
0	1	1	0	1	1.250000	
0	1	1	1	0	1.000000	
0	1	1	1	1	0.031250	
1	0	0	0	0	7.372800	
1	0	0	0	1	7.372800	
1	0	0	1	0	3.686400	
1	0	0	1	1	3.686400	
1	0	1	0	0	1.843200	
1	0	1	0	1	1.843200	
1	0	1	1	0	0.921600	
1	0	1	1	1	0.028800	
1	1	0	0	0	7.159040	
1	1	0	0	1	7.159040	
1	1	0	1	0	3.579520	
1	1	0	1	1	3.579520	
1	1	1	0	0	1.789760	
1	1	1	0	1	1.789760	
1	1	1	1	0	0.894880	
1	1	1	1	1	0.027965	

CLOCK OUTPUT SELECTION TABLE

Figure 5

C1	C0	CKO Output Frequency	Comments
0	0	f_{XTAL}	
0	1	$f_{XTAL} \div 2$	Power on condition
1	0	$f_{XTAL} \div 4$	
1	1	2048 Hz	Binary mode

ELECTRICAL SPECIFICATIONS

ABSOLUTE MAXIMUM RATINGS*

Voltage on any pin relative to V_{SS}	-0.5V to + 12.0V
Operating Temperature, T_A (Ambient)	-40°C to + 85°C
Storage Temperature	-55°C to +125°C

*Stresses greater than those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to Absolute Maximum Rating conditions for extended periods may affect device reliability.

RECOMMENDED DC OPERATING CONDITIONS

$-40°C \leq T_A \leq +85°C$

SYMBOL	PARAMETER	MIN	TYP	MAX	UNIT	NOTES
V_{CC}	Supply Voltage	3.0	5.0	9.5	V	1
V_{SS}	Supply Voltage	0	0	0	V	1

DC ELECTRICAL CHARACTERISTICS

$-40°C \leq T_A \leq +85°C$, $V_{CC} = 5V \pm 10\%$

SYMBOL	PARAMETER	MIN	TYP	MAX	UNIT	NOTES
I_{CC1}	Power Supply Current			2.0	mA	2
I_{CC2}	Power Supply Current			0.1	mA	3
I_{IL}	Input Leakage Current	-1.0		1.0	μA	4
I_{OL}	Output Leakage Current	-10.0		10.0	μA	4
V_{IH}	Logic "1" Voltage, All Inputs	2.0			V	1
V_{IL}	Logic "0" Voltage, All Inputs			0.8	V	1
$V_{I/OH}$	Output Logic "1" Voltage, I/O pin	2.4			V	1(I_{OH}=-100μA)
$V_{I/OL}$	Output Logic "0" Voltage, I/O pin			0.4	V	1(I_{OL}= 1.8 mA)
V_{CKH}	Output Logic "1" Voltage, CKO pin	2.4			V	1(I_{OH} = -400μA)
V_{CKL}	Output Logic "0" Voltage, CKO pin			0.4	V	1(I_{OL} = 4.0 mA)

NOTES

1. All voltages referenced to V_{SS}.
2. Crystal/Clock Input frequency = 8.4 MHz, outputs open.
3. Crystal/Clock Input frequency = 32,768 Hz, outputs open.
4. Measured with V_{CC} = 5.0V, $0 \leq V_I \leq 5.0V$, outputs deselected.

AC ELECTRICAL CHARACTERISTICS

-40°C ≤ T_A + 85°C, V_{CC} 5V ± 10%

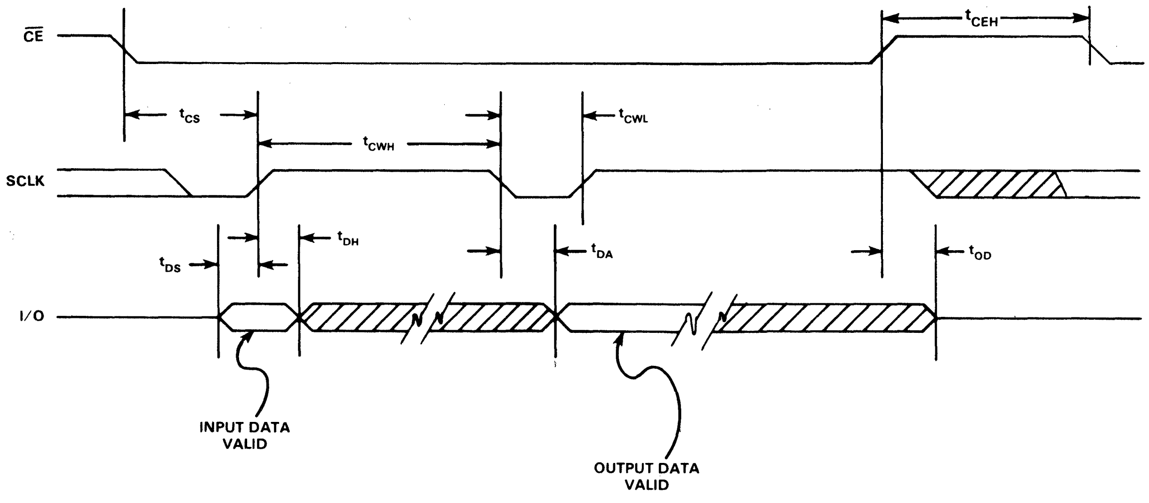
SYMBOL	PARAMETER	MIN	TYP	MAX	UNIT	NOTES
C_I	Capacitance on Input pin		6	10	pF	5
$C_{I/O}$	Capacitance on I/O pin		7	12	pF	5
C_X	Capacitance on XI/CI and X2		7	12	pF	5
f_X	Crystal frequency	27		8400	kHz	
t_{CS}	$\overline{CE}$ to SCLK set up time	1.0			µs	1,6
t_{DS}	Input Data to SCLK set up time	1.0			µs	1,6
t_{DH}	Input Data from SCLK hold time	1.0			µs	1,6
t_{DA}	Output Data from SCLK delay time			1.0	µs	1,6,7,8
t_{OD}	$\overline{CE}$ to I/O high impedance			1.5	µs	1,6,7,8
t_{CWL}	SCLK low time	1.95		∞	µs	
t_{CWH}	SCLK high time	1.95		∞	µs	
f_{SCLK}	SCLK frequency	DC		250	kHz	
t_{SR}, t_{SF}	SCLK Rise and Fall Time			50	ns	9
t_{CR}, t_{CF}	CKO Rise and Fall Time			50	ns	8,9
t_{CEH}	$\overline{CE}$ high time	2.0			µs	

NOTES

5. Measured as $C = \frac{I \Delta t}{\Delta V}$ with ΔV = 3V, and unmeasured pins grounded.
6. Measured at V_{IH} = 2.0 V of V_{IL} = 0.8 V and 5 ns rise and fall times on inputs.
7. Measured at V_{OH} = 2.4V and V_{OL} = 0.4V.
8. Load Capacitance = 100 pF
9. t_r and t_f measured from 0.8V to 2.0V

I/O TIMING DIAGRAM

Figure 6



PRELIMINARY

MOSTEK®

MICRO PERIPHERAL COMPONENTS

MK3807

Programmable CRT Video Control Unit (VCU)

FEATURES

□ Fully Programmable Display Format
Characters per data row (1-200)
Data rows per frame (6-64)
Raster scans per data row (1-16)

□ Programmable Monitor Sync Format
Raster Scans/Frame (256-1023)
"Front Porch"
Sync Width
"Back Porch"
Interlace/Non-Interlace
Vertical Blanking

□ Direct Outputs to CRT Monitor
Horizontal Sync
Vertical Sync
Composite Sync
Blanking
Cursor coincidence

□ Programmed via:
Processor data bus
External PROM

□ Standard or Non-Standard CRT Monitor Compatible

□ Refresh Rate: 60 Hz

□ Scrolling
Single Line
Multi-Line

□ Cursor Position Registers

□ Programmable Character Format

□ Programmable Vertical Data Positioning

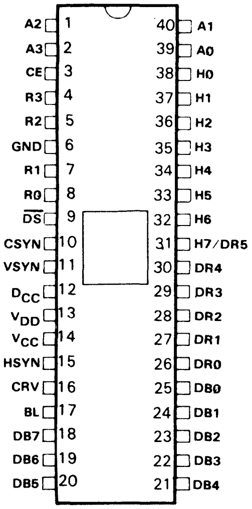
□ Balanced Beam Current Interlace

□ Graphics Compatible

□ Split-Screen Applications
Horizontal
Vertical

□ Interlace or Non-Interlace operation

PIN CONFIGURATION



□ TTL Compatibility

□ BUS Oriented: Compatible with most microprocessors

□ Second source to SMC CRT 5037

□ N-Channel Silicon Gate Technology

GENERAL DESCRIPTION

The Programmable CRT Video Control Unit (VCU) Chip is a user programmable 40-pin n channel MOS/LSI device containing the logic functions required to generate all the timing signals for the presentation and formatting of interlaced and non-interlaced video data on a standard or non-standard CRT monitor. The MK3807 VCU is a second source to SMC CRT 5037.

With the exception of the dot counter, which may be clocked at a video frequency above 25 MHz and therefore not recommended for MOS implementation, all frame formatting, such as horizontal, vertical, and composite sync, characters per data row, data rows per frame, and raster scans per data row and per frame are totally user programmable. The data row counter has been designed to facilitate scrolling. Refer to Table 1 for description of pin functions.

Programming is accomplished by loading seven 8-bit control registers directly off an 8-bit bidirectional data bus. Four register address lines and a chip enable line provide complete microprocessor compatibility for program controlled set up. The device can be "self loaded" via an external PROM tied on the data bus as described in the OPERATION section.

Figure 1 shows a block diagram of the internal functional components of the VCU.

The MK3807 (VCU) may be programmed for an odd or even number of scan lines per data row in both interlaced and non-interlaced modes.

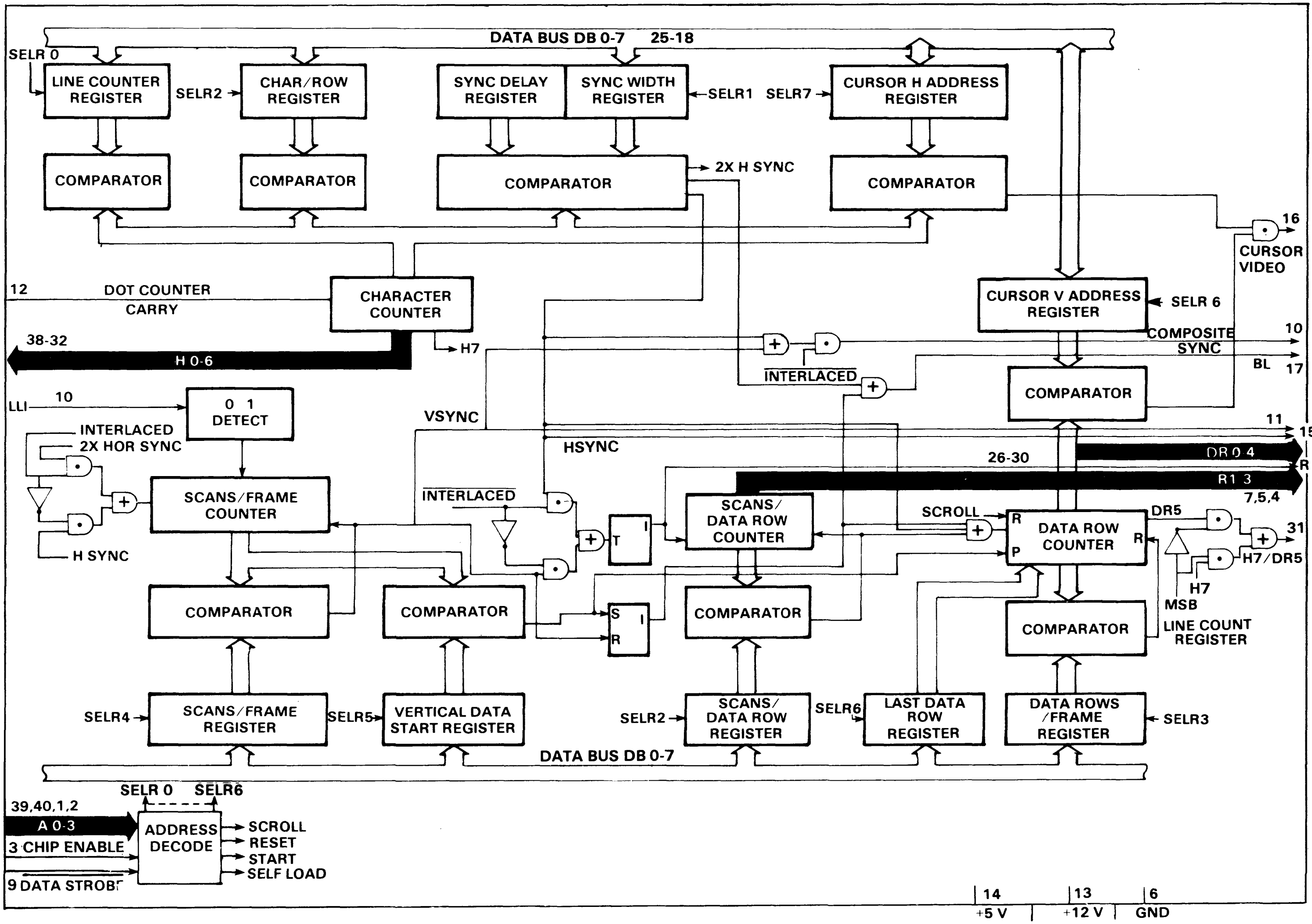
In addition to the seven control registers, two additional registers are provided to store the cursor character and data row addresses for generation of the cursor video signal. The contents of these two registers can also be read out onto the bus for update by the program.

DESCRIPTION OF PIN FUNCTIONS

Table 1

Pin No.	Symbol	Name	Input/ Output	Function
25-18	DB0-7	Data Bus	I/O	Data bus. Input bus for control words from microprocessor or PROM. Bi-directional bus for cursor address.
3	CE	Chip Enable	I	Signals chip that it is being addressed.
39,40,1,2	A0-3	Register Address	I	Register address bits for selecting one of seven control registers or either of the cursor address registers.
9	$\overline{DS}$	Data Strobe	I	Strobes DB0-7 into the appropriate register or outputs the cursor character address or cursor line address onto the data bus.
12	DCC	Dot Counter Carry	I	Carry from off-chip dot counter establishing basic character clock rate. Character clock.
38-32	H0-6	Character Counter Outputs	O	Character counter outputs.
7,5,4	R1-3	Scan Counter Outputs	O	Three most significant bits of the Scan Counter; row select inputs to character generator.
31	H7/DR5	H7/DR5	O	Pin definition is user programmable. Output is MSB of Character Counter if horizontal line counter (REG.0) is ≥ 128; otherwise output is MSB Of Data Row Counter.
8	R0	Scan Counter LSB	O	Least significant bit of the scan counter. In the interlaced mode with an even number of scans per data row, R0 will toggle at the field rate; for an odd number of scans per data row in the interlaced mode, R0 will toggle at the data row rate.
26-30	DR0-4	Data Row Counter Outputs	O	Data Row counter outputs.
17	BL	Blank	O	Defines non-active portion of horizontal and vertical scans.
15	HSYN	Horizontal Sync	O	Initiates horizontal retrace.
11	VSYN	Vertical Sync	O	Initiates vertical retrace.
10	CSYN	Composite Sync Output	O	Composite sync is provided on the MK3807. This output is active in non-interlaced mode only. Provides a true RS-170 composite sync wave form.
16	CRV	Cursor Video	O	Defines cursor location in data field.
14	V_{CC}	Power Supply	PS	+5 volt Power Supply
13	V_{DD}	Power Supply	PS	+12 volt Power Supply

BLOCK DIAGRAM
Figure 1



OPERATION

The design philosophy employed was to allow the MK3807 Programmable CRT Video Control Unit (VCU) to interface effectively with either a microprocessor based or hardwire logic system. The device is programmed by the user in one of two ways; via the processor data bus as part of the system initialization routine, or during power up via a PROM tied on the data bus and addressed directly by the Row Select outputs of the chip (See Figure 2). Seven 8-bit words are required to fully program the chip. Bit assignments for these words are shown in Tables 2, 3 and 4. The information contained in these seven words consists of the following:

Horizontal Formatting:

Characters/Data Row
A 3 bit code providing 8 mask programmable character lengths from 20 to 132. The standard device will be masked for the following character lengths; 20, 32, 40, 64, 72, 80, 96, and 132.

Horizontal Sync Delay
3 bits assigned providing up to 8 character times for generation of "front porch".

Horizontal Sync Width
4 bits assigned providing up to 16 character times for generation of horizontal sync width.

Horizontal Line Count
8 bits assigned providing up to 256 character times for total horizontal formatting.

Skew Bits
A 2 bit code providing from a 0 to 2 character skew (delay) between the horizontal address counter and the blank and sync (horizontal, vertical, composite) signals to allow for retiming of video data prior to generation of composite video signal. The Cursor Video signal is also skewed as a function of this code.

Vertical Formatting:

Interlaced/Non-interlaced
This bit provides for data presentation with odd/even field formatting for interlaced systems. It modifies the vertical timing counters as described below. A logic 1 establishes the interlace mode.

Scans/Frame
8 bits assigned, defined according to the following equations: Let X = value of 8 assigned bits.
1) in interlaced mode—scans/frame = 2X + 513. Therefore for 525 scans, program X = 6 (00000110). Vertical sync will occur precisely every 262.5 scans, thereby producing two interlaced fields.
Range = 513 to 1023 scans/frame, odd counts only.
2) in non-interlaced mode—scans/frame = 2X + 256. Therefore for 262 scans, program X = 3 (00000011).
Range = 256 to 766 scans/frame, even counts only.
In either mode, vertical sync width is fixed at three horizontal scans (≡3H).

Vertical Data Start
8 bits defining the number of raster scans from the leading edge of vertical sync until the start of display data. At this raster scan the data row counter is set to the data row address at the top of the page.

Data Rows/Frame
6 bits assigned providing up to 64 data rows per frame.

Last Data Row
6 bits to allow up or down scrolling via a preload defining the count of the last displayed data row.

Scans/Data Row
4 bits assigned providing up to 16 scan lines per data row.

ADDITIONAL FEATURES

MK3807 VCU Initialization:

Under microprocessor control—The device can be reset under system or program control by presenting a 1010 address on A3-0. The device will remain reset at the top of the even field page until a start command is executed by presenting a 1110 address on A3-0.

Via "Self Loading"—In a non-processor environment, the self loading sequence is effected by presenting and holding the 1111 address on A3-0, and is initiated by the receipt of the strobe pulse ($\overline{DS}$). The 1111 address should be maintained long enough to insure that all seven registers have been loaded (in most applications under one millisecond). The timing sequence will begin one line scan after the 1111 address is removed. In processor based systems, self loading is initiated by presenting the 0111 address to the device. Self loading is terminated by presenting the start command to the device which also initiates the timing chain.

Scrolling—In addition to the Register 6 storage of the last displayed data row a "scroll" command (address 1011) presented to the device will increment the first displayed data row count to facilitate up scrolling in certain applications.

CONTROL REGISTERS PROGRAMMING CHART

Table 2

Horizontal Line Count: Total Characters/Line = N + 1, N = 0 to 255 (DB0 = LSB)

Characters/Data Row:

DB2	DB1	DB0		
0	0	0	= 20	Active Characters/Data Row
0	0	1	= 32	
0	1	0	= 40	
0	1	1	= 64	
1	0	0	= 72	
1	0	1	= 80	
1	1	0	= 96	
1	1	1	= 132	

Horizontal Sync Delay: = N, from 1 to 7 character times (DB0 = LSB, N = 0 Disallowed)

Horizontal Sync Width: = N, from 1 to 15 character times (DB3 = LSB, N = 0 Disallowed)

Skew Bits

DB7	DB8	Sync/Blank Delay	Cursor Delay (Character Times)
0	0	0	0
1	0	1	0
0	1	2	1
1	1	2	2

Scans/Frame
8 bits assigned, defined according to the following equations:
Let X = value of 8 assigned bits. DB0 = LSB)
1) in interlaced mode— scans/frame = 2X + 513. Therefore for 525 scans, program X = 6 (0000110). Vertical sync will occur precisely every 262.5 scans, thereby producing two interlaced fields.
Range = 513 to 1023 scans/frame, odd counts only.
2) in non-interlaced mode—scans/frame = 2X + 256. Therefore for 262 scans, program X = 3 (00000011).
Range = 256 to 766 scans/frame, even counts only.
In either mode, vertical sync width is fixed at three horizontal scans (= 3H)

Vertical Data Start: N = number of raster lines delay after leading edge of vertical sync of vertical start position. (DB0 = LSB)

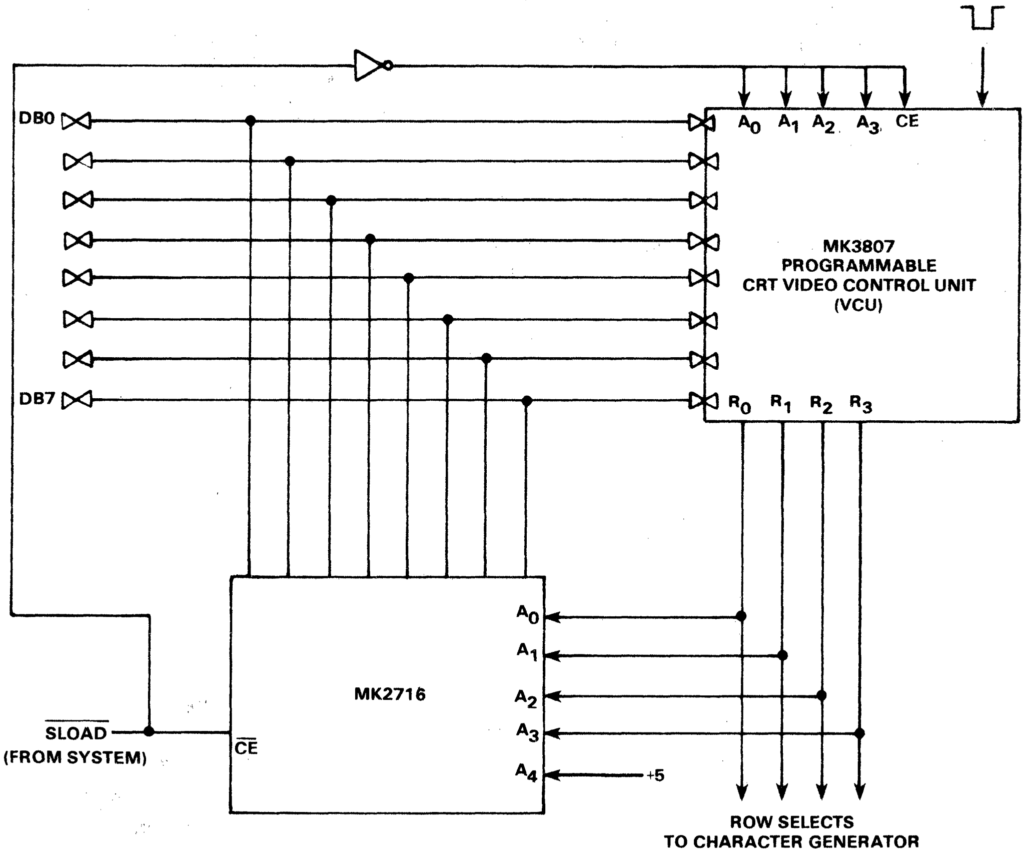
Data Rows/Frame: Number of data rows = N + 1, N = 0 to 63 (DB0 = LSB)

Last Data Row: N = Address of last displayed data row, N = 0 to 63, ie; for 24 data rows, program N = 23. (DB0 = LSB)

Mode: Regster, 1, DB7 = 1 established Interlace.

Scans/Data Row:
Interlace Mode
Scans per data Row = N + 2. N = 0 to 14, odd or even counts.
Non-Interlace Mode
Scans per Data Row = N + 1, odd or even count, N = 0 to 15.

SELF LOADING SCHEME
Figure 2



OPTIONAL START-UP SEQUENCE

When employing microprocessor controlled loading of the MK3807 VCU's registers, the following sequence of instruction may be used optionally:

ADDRESS	COMMAND
1 1 1 0	Start Timing Chain
1 0 1 0	Reset
0 0 0 0	Load Register 0
.	.
.	.
.	.
0 1 1 0	Load Register 6
1 1 1 0	Start Timing Chain

The sequence of START RESET LOAD START is necessary to insure proper initialization of the registers.

This sequence is not required if register loading is via either of the Self Load modes.

REGISTER SELECTS/COMMAND CODES

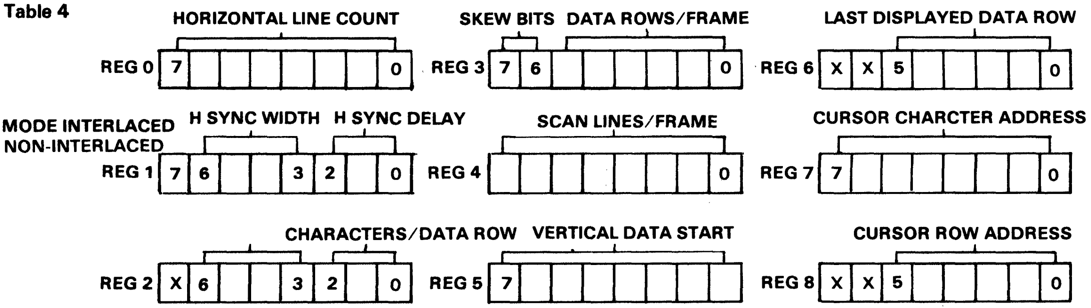
Table 3

A3	A2	A1	A0	Select/Command	Description
0	0	0	0	Load Control Register 0	See Table 4
0	0	0	1	Load Control Register 1	
0	0	1	0	Load Control Register 2	
0	0	1	1	Load Control Register 3	
0	1	0	0	Load Control Register 4	
0	1	0	1	Load Control Register 5	
0	1	1	0	Load Control Register 6	
0	1	1	1	Processor Initiated Self Load	Command from processor instructing MK3807 VCU to enter Self Load Mode (via external PROM)
1	0	0	0	Read Cursor Line Address	
1	0	0	1	Read Cursor Character Address	
1	0	1	0	Reset	Resets timing chain to top left of page. Reset is latched on chip by $\overline{DS}$ and counters are held until released by start command.
1	0	1	1	Up Scroll	Increments address of first displayed data row on page, i.e.; prior to receipt of scroll command—top line = 0, bottom line = 23. After receipt of Scroll Command—top line = 1, bottom line = 0.
1	1	0	0	Load Cursor Character Address[1]	
1	1	0	1	Load Cursor Line Address[1]	
1	1	1	0	Start Timing Chain	Receipt of this command after a Reset or Processor Self Load command will release the timing chain approximately one scan line later. In applications requiring synchronous operation of more than one VCU the dot counter carry should be held low during the $\overline{DS}$ for this command.
1	1	1	1	Non-Processor Self Load	Device will begin self load via PROM when $\overline{DS}$ goes low. The 1111 command should be maintained on A3-0 long enough to guarantee self load. (Scan counter should cycle through at least once). Self load is automatically terminated and timing chain initiated when the all "1's" condition is removed, independent of $\overline{DS}$. For synchronous operation of more than one VCU, the Dot Counter Carry should be held low when the command is removed.

NOTE 1: During Self-Load, the Cursor Character Address Register (REG 7) and the Cursor Row Address Register (REG 8) are enabled during states 0111 and 1000 of the R3-R0 Scan Counter outputs respectively. Therefore, Cursor data in the PROM should be stored at these addresses.

BIT ASSIGNMENT CHART

Table 4



MAXIMUM GUARANTEED RATINGS*

Operating Temperature Range	0°C to + 70°C
Storage Temperature Range	-55°C to + 150°C
Lead Temperature (soldering, 10 sec.)	+ 325°C
Positive Voltage on any Pin, with respect to ground	+ 18.0 V
Negative Voltage on any Pin, with respect to ground	-0.3 V

*Stresses above those listed may cause permanent damage to the device. This is a stress rating only and funcitonal operation of the device at these or at any other condition above those indicated in the operational sections of this specification is not implied.

NOTE: When powering this device from laboratory or system power supplies, it is important that the Absolute Maximum Ratings not be exceeded or device failure can result. Some power supplies exhibit voltage spikes or "glitches" on their outputs when the AC power is switched on and off. In addition, voltage transients on the AC power line may appear on the DC output. For example, the bench power supply programmed to deliver +12 volts may have large voltage transients when the AC power is switched on and off. If this possibility exists it is suggested that a clamp circuit be used.

DC CHARACTERISTICS

(T_A = 0°C to 70°C, V_{CC} = +5V ± 5%, V_{DD} = +12V ± 5%, unless otherwise noted)

PARAMETER	MIN	TYP	MAX	UNIT	COMMENTS
INPUT VOLTAGE LEVELS					
Low Level, V_{IL}			0.8	V	
High Level, V_{IH}	V_{CC}-1.5		V_{CC}	V	
OUTPUT VOLTAGE LEVELS					
Low Level - V_{OL} for R0-3			0.4	V	I_{OL}=3.2 ma
Low Level - V_{OL}, all others			0.4	V	I_{OL}=1.6 ma
High Level - V_{OH} for R0-3, DB0-7	2.4				I_{OH}=80μa
High Level - V_{OH} all others	2.4				I_{OH}=40μa
INPUT CURRENT					
Low Level, I_{IL} (Address, CE only)			250	μA	V_{IN}=0.4 V
Leakage, I_{IL} (All inputs except Address, CE)			10	μA	$0 \leq V_{IN} \leq V_{CC}$
INPUT CAPACITANCE					
Data Bus, C_{IN}		10	15	pF	
$\overline{DS}$, Clock, C_{IN}		25	40	pF	
All other, C_{IN}		10	15	pF	
DATA BUS LEAKAGE in INPUT MODE I_{DB}			10	μA	$0.4 \leq V_{IN} \leq 5.25$ V
POWER SUPPLY CURRENT					
I_{CC}		80	100	mA	
I_{DD}		40	60	mA	

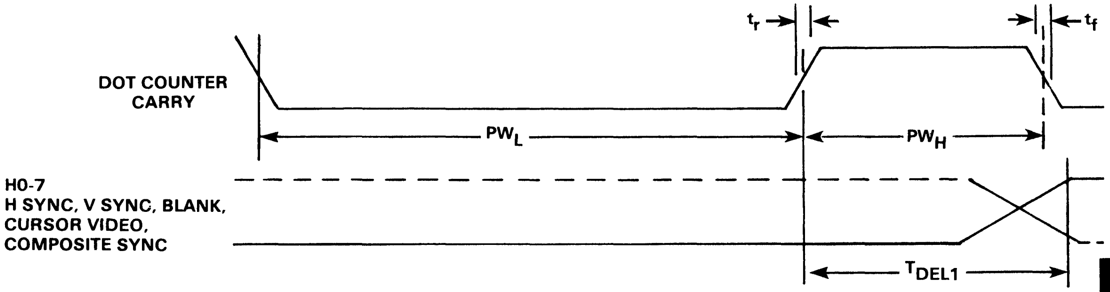
AC CHARACTERISTICS

(T_A = 25°C)

PARAMETER	MIN	TYP	MAX	UNIT	COMMENTS
DOT COUNTER CARRY					
frequency	0.5		4.0	MHz	Figure 3
PW_H	35			ns	Figure 3
PW_L	215			ns	Figure 3
t_r, t_f		10	50	ns	Figure 3
DATA STROBE					
PW_{DS}	150ns		10μs		Figure 4
ADDRESS, CHIP ENABLE					
Set-up time	125			ns	Figure 4
Hold time	50			ns	Figure 4
DATA BUS - LOADING					
Set-up time	125			ns	Figure 4
Hold time	75			ns	Figure 4
DATA BUS - READING					
T_{DEL2}			125	ns	Figure 4, CL =50pF
T_{DEL4}	5		60	ns	Figure 4, CL =50pF
OUTPUTS, H0-7, HS, VS, BL, CRV					
CE-T_{DEL1}			125	ns	Figure 3, CL =20pF
OUTPUTS: R0-3, DR0-5					
T_{DEL3}	*		500	ns	Figure 5, CL =20pF

AC TIMING DIAGRAMS
VIDEO TIMING

Figure 3



VII Z80 MICRO-COMPUTER PERIPHERALS

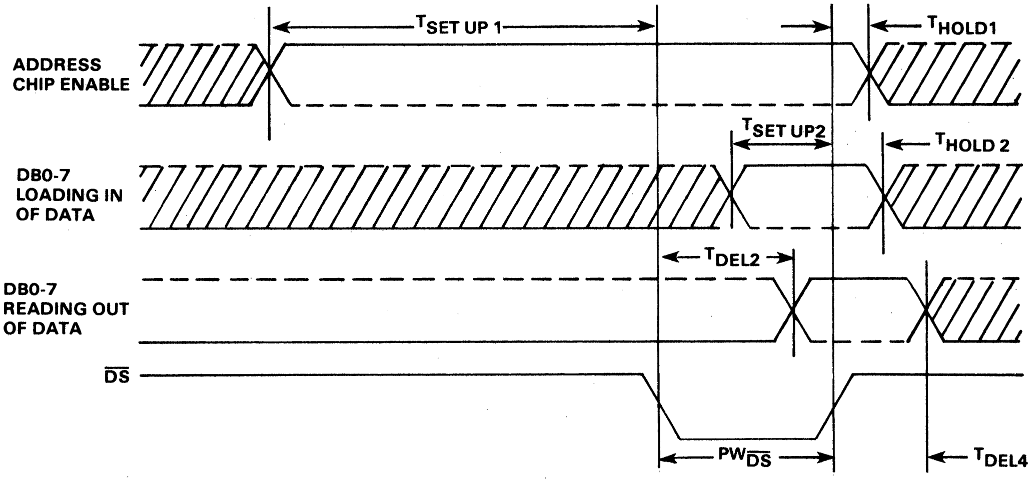
RESTRICTIONS

1. Only one pin is available for strobing data into the device via the data bus. The cursor X and Y coordinates are loaded into the chip by presenting one set of addresses and outputed by presenting a different set of addresses. Therefore, the standard WRITE and READ control signals from most microprocessors must be "NORed" externally to present a single strobe ($\overline{DS}$) signal to the device.

2. In interlaced mode the total number of character slots assigned to the horizontal scan must be even to insure that vertical sync occurs precisely between horizontal sync pulses.

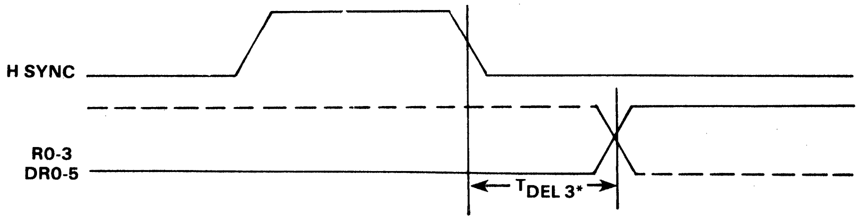
LOAD/READ TIMING

Figure 4



SCAN AND DATA ROW COUNTER TIMING

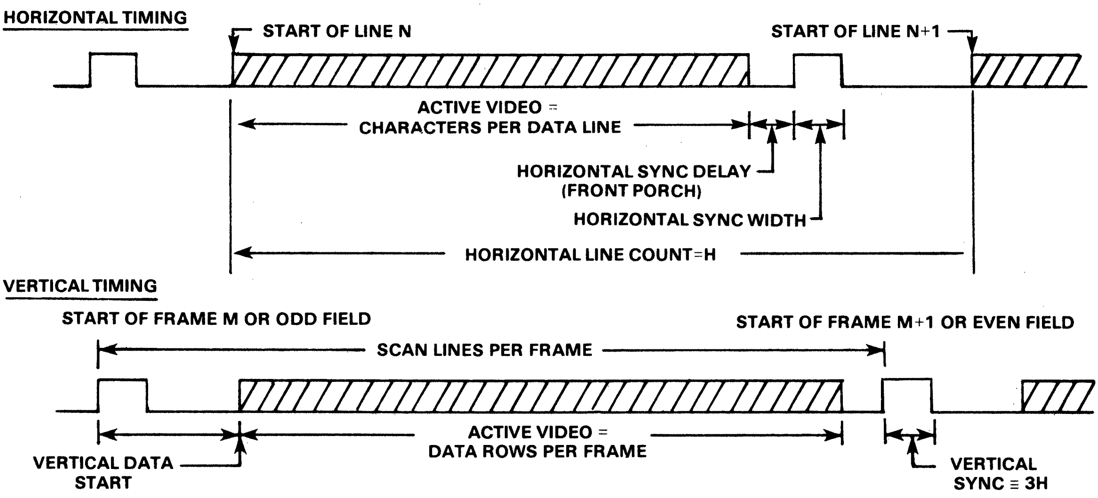
Figure 5



*R0-3 and DR0-5 may change prior to the falling edge of H sync

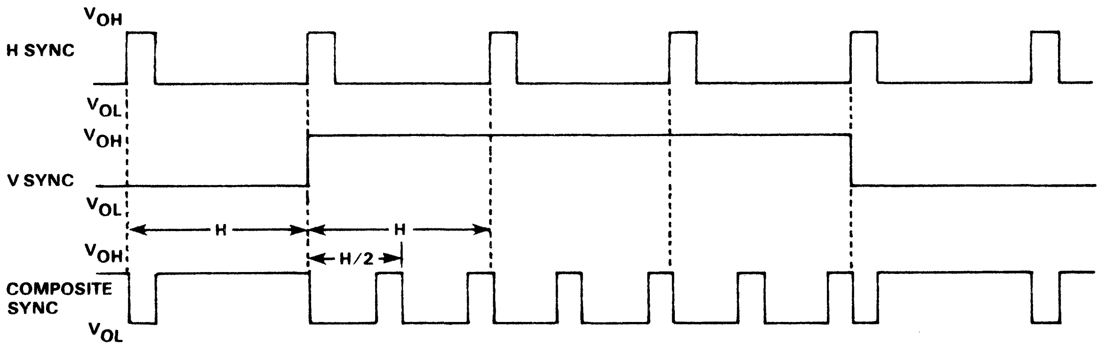
GENERAL TIMING

Figure 6



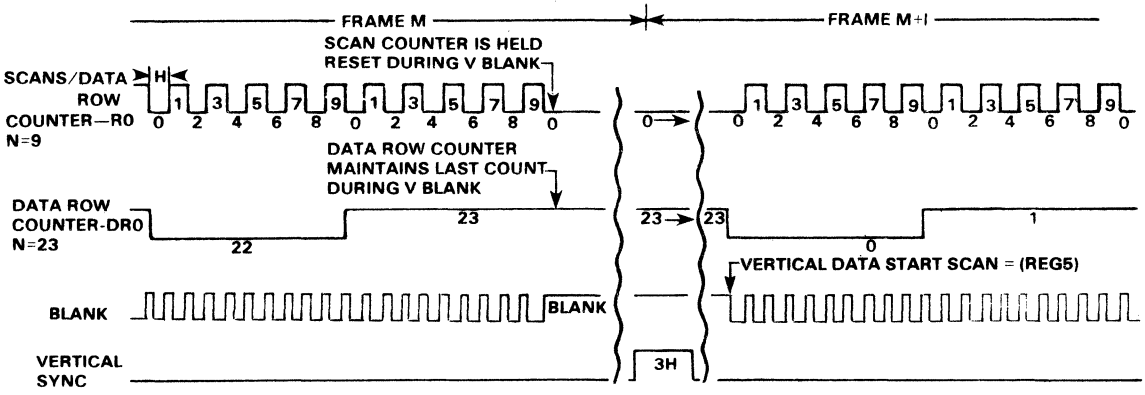
COMPOSITE SYNC TIMING

Figure 7



VERTICAL SYNC TIMING

Figure 8



FRAME M
FRAME M+I
SCAN COUNTER IS HELD RESET DURING V BLANK
SCANS/DATA ROW COUNTER—R0 N=9
H
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
0
DATA ROW COUNTER MAINTAINS LAST COUNT DURING V BLANK
DATA ROW COUNTER-DR0 N=23
22
23
23
23
0
1
VERTICAL DATA START SCAN = (REG5)
BLANK
BLANK
VERTICAL SYNC
3H

EXAMPLE BASED ON NON-INTERLACED (REG 1, BIT 7 = 0), 24 DATA ROWS, 10 SCANS/DATA ROW

ADVANCE INFORMATION

MOSTEK®

Z80 MICROCOMPUTER PERIPHERALS

STI-Z80 Version

MK3801

FEATURES

- □ Full Duplex USART
- □ Two Binary Delay Timers
- □ Two Full Feature Timers with
  - Delay to Interrupt Mode
  - Pulse Width Measurement Mode
  - Event Counter Mode
- □ Eight General Purpose Lines with
  - Full Bi-Directional I/O Capability
  - Edge Triggered Interrupts on Either Edge
- □ Full Control of Each Interrupt Channel
  - Enable/Disable
  - Maskable
  - Automatic End-of-Interrupt Mode
  - Software End-of-Interrupt Mode
  - Automatic End-of-Service Mode

INTRODUCTION

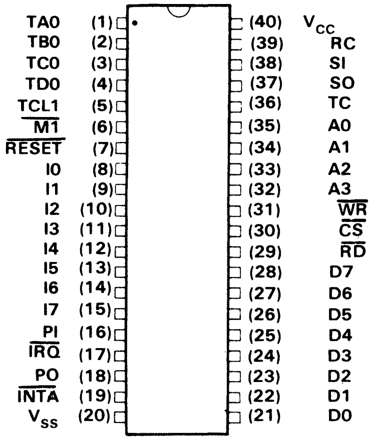
The MK3801 is a multifunctional peripheral device for use in Z80 based systems. It provides a USART, four timers (two binary and two full function), and eight bi-directional I/O lines with individually programmable interrupts. The interrupt structure of the device is fully programmable for all interrupts, provides for interrupt vector generation, conforms to the Z80 daisy chain interrupt priority scheme, and supports automatic end-of-interrupt and end-of-service functions for the Z80.

PIN DESCRIPTION

Figure 1 illustrates the pinout of the MK3801. The functions of these individual pins are described below.

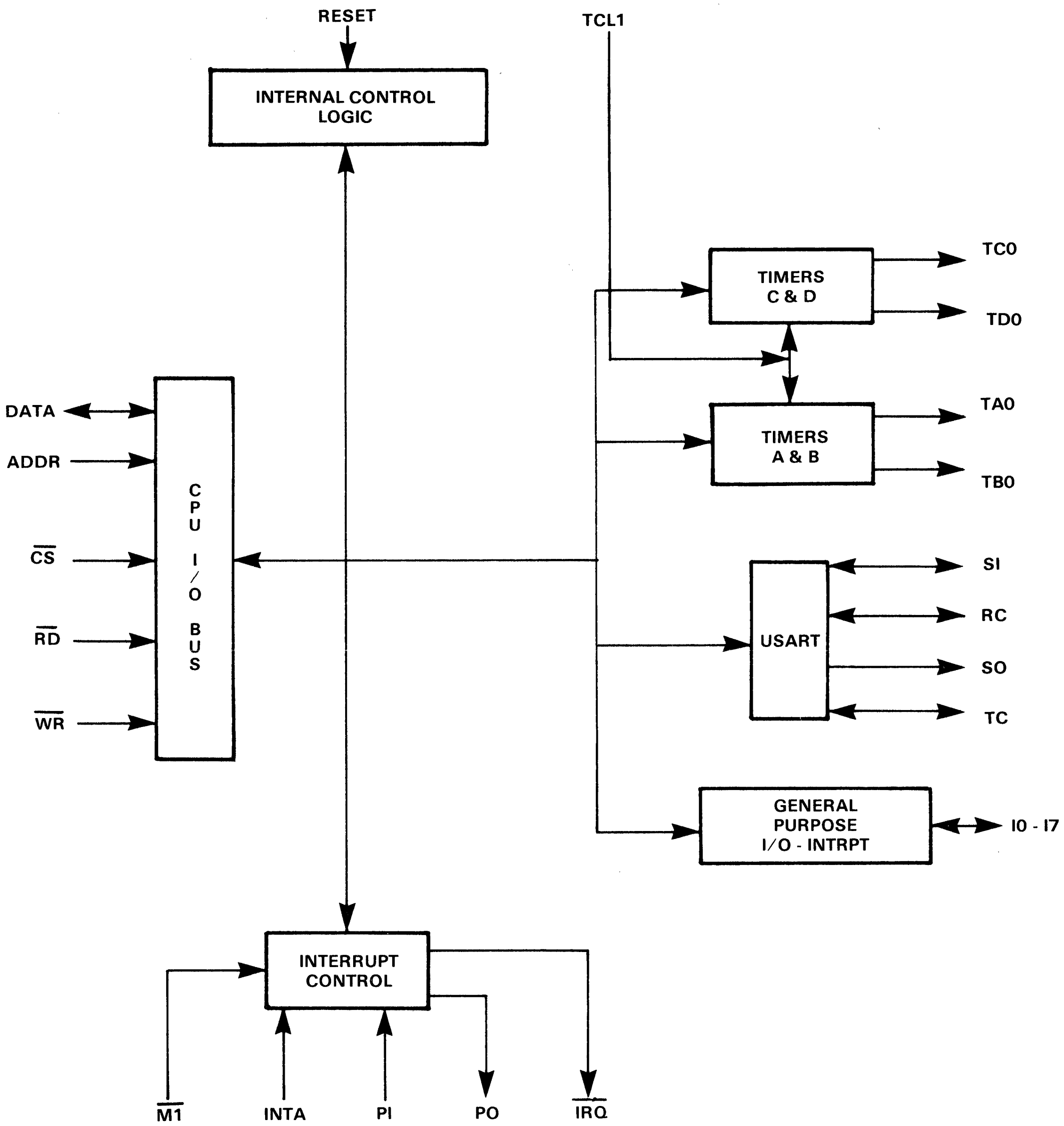
DEVICE PINOUT

Figure 1



SIGNAL NAME	DESCRIPTION
V_{SS}	Ground.
V_{CC}	+5 volts (± 5 or 10 percent)
$\overline{\text{CS}}$	Chip Select (negative true)
$\overline{\text{RD}}$	Read Enable (negative true)
$\overline{\text{WR}}$	Write Enable (negative true)
A0 - A3	Address Inputs. Used to address one of the internal registers during a read or write operation.
D0 - D7	Data Bus (bi-directional).
$\overline{\text{RESET}}$	Device Reset (negative true). When activated, all internal registers (except for Timer or USART Data Registers) will be cleared, all timers stopped, USART turned off, all interrupts disabled and all pending interrupts cleared, and all I/O lines placed in tri-state mode.
I0 - I7	General purpose I/O and interrupt lines.
$\overline{\text{IRQ}}$	Interrupt Request Output (active low open drain).
$\overline{\text{INTA}}$	Interrupt Acknowledge. Used to signal the MK3801 that the CPU is acknowledging its interrupt.
PI	Priority Input.
PO	Priority Output.
SO	Serial Output.
SI	Serial Input.
RC	Receiver Clock Input.
TC	Transmit Clock Input.
TA0 - TD0	Timer Outputs.
TCL1	Timer Clock Input.
$\overline{\text{M1}}$	Z80 Machine Cycle One (negative true)

INTERNAL ORGANIZATION
Figure 2



INTERNAL ORGANIZATION

Figure 2 illustrates the MK3801 internal organization, supporting the full set of timing, communications, parallel I/O, and interrupt processing functions available in the device.

CPU I/O BUS

The CPU I/O Bus provides the means of communications between the system and the MK3801. Data, Status, and Control Registers in the MK3801 are accessed by the bus in order to establish device parameters, assert control, and transfer status and data between the system and the MK3801.

Each register in the MK3801 is addressed over the address bus and with Chip Select (/CS), while data is transferred over the eight bit Data bus under control of Read (/RD) and Write (/WR) signals.

INTERRUPT CONTROL

The Interrupt Control section provides full vectored interrupt control processing for all I/O facilities of the MK3801. Each individual function is provided with a unique interrupt vector that is presented to the system during the interrupt acknowledge cycle. All interrupts are fully maskable, with individual enable and disable controls. In addition, the interrupts associated with each of the General Purpose I/O lines may be programmed to occur on either the rising edge or the falling edge of the incoming signal. Optional End-of-Interrupt modes and End-of-Service modes are available under software control.

TIMERS

Four timers are available on the MK3801. Two provide full service features including delay timer operation, event counter operation, pulse width measurement operation, and pulse generation. The other two timers provide delay timer features only, and may be used for baud rate generators for use with the USART.

All timers are prescaler/counter timers, with a common independent clock input, and are not required to be operated from the system clock. In addition, all timers have a time-out output function that toggles each time the timer times out.

USART

Serial Communication is provided by the USART. This section is capable of either asynchronous or synchronous operation. Variable word width and start and stop bit configurations are available under software control for asynchronous operation. For synchronous operation, a Sync Word is provided to establish synchronization during receive operations. The Sync Word will also be repeatedly transmitted when no other data is available for transmission.

Separate receive and transmit clocks are available, and separate receive and transmit status and data bytes allow independent operation of the transmit and receive sections.

GENERAL PURPOSE I/O - INTERRUPT PORT

The General Purpose I/O - Interrupt Port provides eight I/O lines that may be operated either as inputs or outputs under control of software. In addition, each line may generate an interrupt on either a positive going edge or a negative going edge of the input signal.

Two of the lines in this port provide auxiliary input functions for the timers in the pulse width measurement mode and the event counter mode. Two others serve as auxiliary output lines for the USART, and reflect the status of the receive and transmit buffers, for control of DMA operations.

MOSTEK®

8-BIT A/D CONVERTER/8-CHANNEL ANALOG MULTIPLEXER

MK50808(N/P)

FEATURES

□ Single 5-Volt Supply (± 5%)

□ Low Power Dissipation - 6.825mW(max) at 640kHz

□ Total Unadjusted Error < ± ½ LSB

□ Linearity Error < ± ½ LSB

□ No Missing Codes

□ Guaranteed Monotonicity

□ No Zero Adjust Required

□ No Full-Scale Adjust Required

□ 108μs Conversion Time (Typically)

□ Easy Microprocessor Interface

□ Latched TTL-Compatible Three-State Output with True Bus-Driving Capability

□ 8-channel Analog Multiplexer

□ Latched Address Input

□ Fixed Reference or Ratiometric Conversion

□ Continuous or Controlled Conversion

□ On-Chip Chopper-Stabilized Comparator

□ Low Reference-Voltage Current Drain

DESCRIPTION

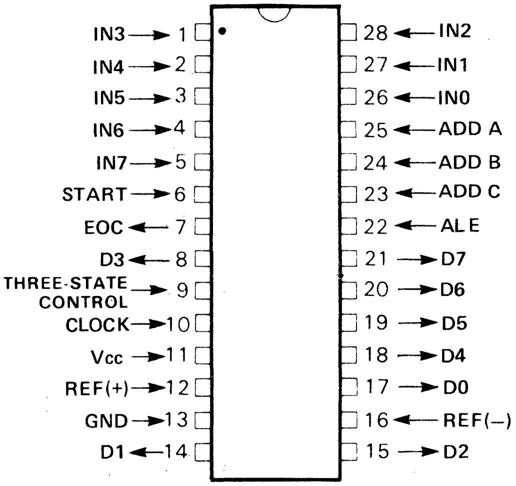
The MK50808 is a monolithic CMOS device with an 8-bit successive approximation A/D converter, an 8-channel analog multiplexer and microprocessor-compatible control logic. The 8-channel multiplexer can directly access any one of 8 single-ended analog channels. The 8-bit A/D converter consists of 256 series resistors with an analog switch array, a chopper-stabilized comparator and a successive approximation register. The series resistor approach guarantees monotonicity and no missing codes as well as allowing both ratiometric and fixed-reference measurements. The need for external zero and full-scale adjustments has been eliminated and an absolute accuracy of ≤ 1 LSB, including quantizing error, is provided. A block diagram of the MK50808 is shown in Figure 2.

All digital outputs are TTL-compatible, all digital inputs are TTL-compatible with a pull-up resistor, and all digital inputs and outputs are CMOS-compatible; this makes it easy to interface with most microprocessors. The output latch is three-state and provides true bus-driving capability (300ns from Three-State Control to Q Logic State with 200pF load). A Start signal initiates the conversion process, and, upon completion, an End-Of-Conversion signal is generated. Continuous conversion is possible by tying the Start-Convert pin to the End-of-Conversion pin.

The pin configuration of the MK50808 is shown in Figure 1.

PIN CONNECTIONS

Figure 1



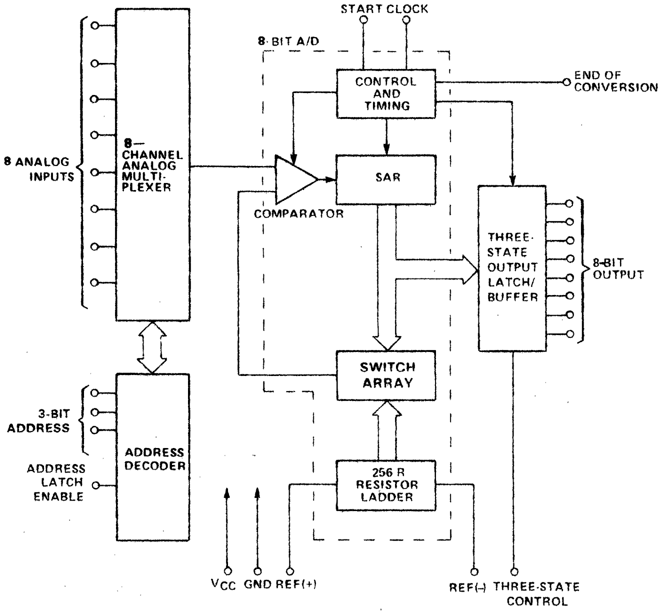
VII Z80 MICRO-COMPUTER PERIPHERALS

The MK50808 features low power, high accuracy, minimal temperature dependence, and excellent long-term accuracy and repeatability. These characteristics make this device ideally suited to machine and industrial controls.

A block diagram of a microprocessor control system using the MK50808 is shown in Figure 3.

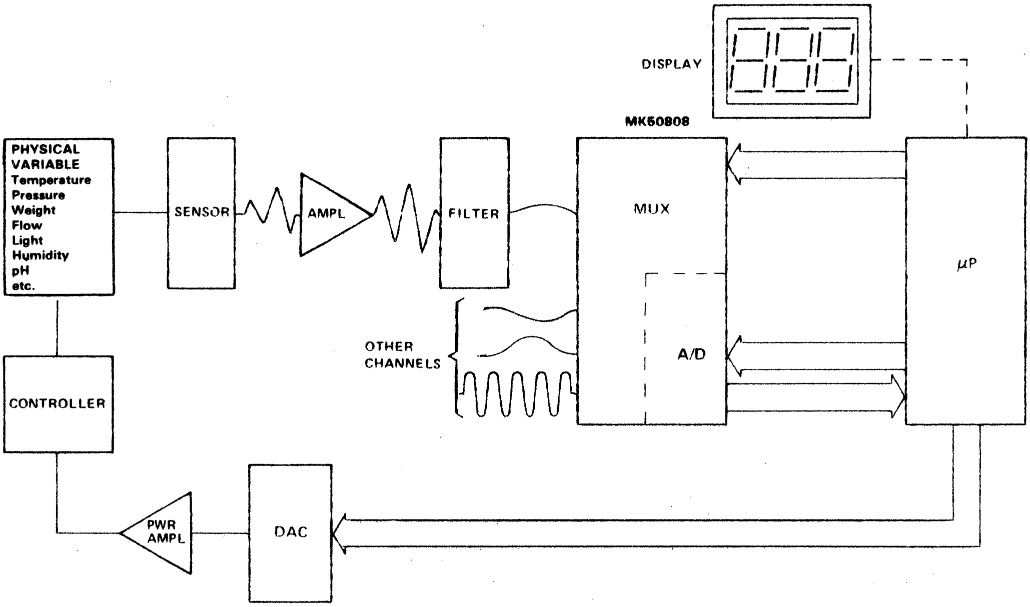
MK50808 BLOCK DIAGRAM

Figure 2



TYPICAL MICROPROCESSOR CONTROL SYSTEM

Figure 3



FUNCTIONAL DESCRIPTION (Refer To Figure 2 for a Block Diagram)

ADDRESS, Pins 23-25

The address decoder allows the 8-input analog multiplexer to select any one of 8 single-ended analog input channels. Table 1 shows the required address inputs to select any analog input channel.

ADDRESS LATCH ENABLE, Pin 22

A positive transition applied to the Address Latch Enable (ALE) input latches a 3-bit address into the address decoder. ALE can be tied to Start with parameter t_D being satisfied.

CLOCK INPUT, Pin 10

This Clock Input will accept an external clock input from 100kHz to 1.2MHz

POSITIVE AND NEGATIVE REFERENCE VOLTAGES [REF (+) and REF (-)], Pins 12 and 16

These inputs supply voltage references for the analog-to-digital converter. Internal voltage references are derived from REF (+) and REF (–) by a 256-R ladder network, Figure 4.

This approach was chosen because of its inherent monotonicity, which is extremely important in closed-loop feedback control systems. A non-monotonic transfer characteristic can cause catastrophic oscillations within a system.

The top and bottom resistors of the ladder network in Figure 4 are not the same value as the rest of the resistors in the ladder. They are chosen so that the output characteristic will be symmetrical about its full-scale and zero points. The first output transition occurs when the analog signal reaches +½ LSB and succeeding transitions occur every 1 LSB until the output reaches full scale.

ANALOG INPUTS, Pins 1-5, 26-28

These inputs are multiplexing analog switches which accept analog inputs from 0V to V_{CC}.

The comparator is the most important section of the A/D converter because this section determines the ultimate accuracy of the entire converter. It is the DC drift of the comparator which determines the repeatability of the device. A chopper-stabilized comparator was chosen because it best satisfies all the converter requirements.

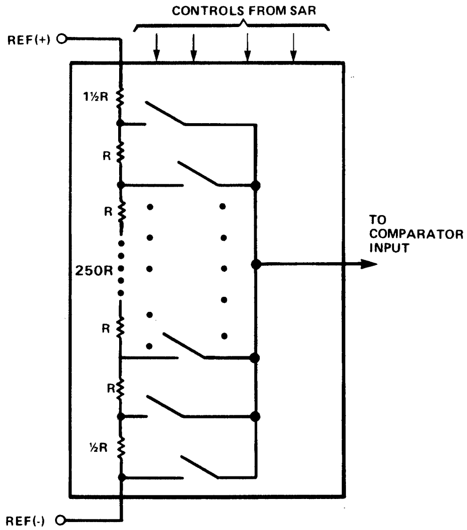
The chopper-stabilized comparator converts the DC input signal into an AC signal. This signal is amplified by a high-gain AC amplifier and the DC level is restored. This technique limits the drift component of the comparator because the drift is a DC component which is not passed by the AC amplifier.

Since drift is virtually eliminated, the entire A/D converter is extremely insensitive to temperature and exhibits very little long-term drift and input offset error.

ANALOG CHANNEL SELECTION
Table 1

SELECTED ANALOG CHANNEL	ADDRESS LINE			
		C	B	A
IN0		L	L	L
IN1		L	L	H
IN2		L	H	L
IN3		L	H	H
IN4		H	L	L
IN5		H	L	H
IN6		H	H	L
IN7		H	H	H

RESISTOR LADDER AND SWITCH ARRAY
Figure 4



START, Pin 6

The A/D converter's successive approximation register (SAR) is reset by the positive edge of the Start pulse. Conversion begins on the falling edge of the Start pulse.

A conversion in progress will be interrupted if a new Start pulse is received and a new conversion will begin.

END OF CONVERSION, Pin 7

The End-Of-Conversion (EOC) output goes high when the conversion process has been completed. The positive edge of the EOC output indicates a valid digital output. Continuous conversion can be accomplished by tying the EOC output to the Start input. If the A/D converter is used in this mode, an external Start pulse should be applied after power up. End of Conversion will go low within 2 clock periods after the positive edge of Start.

8-BIT DIGITAL OUTPUT, Pins 8, 14, 15, 17-21

These pins supply the binary digital output code which corresponds to the analog input voltage. D0 is the least significant bit (LSB) and D7 is the most significant bit (MSB). This output is stored in a TTL-compatible three-state output latch which can drive a 200pF bus from high impedance to either logic state in 300ns. Each pin can drive one standard TTL load.

THREE-STATE CONTROL, Pin 9

The Three-State Control allows the converter to be connected to an 8-bit data bus. A low level applied to this input causes the digital output to go to a high impedance state and a high level causes the output to go to a Q logic state.

ABSOLUTE MAXIMUM RATINGS* (Note 1)

Absolute Maximum V_{CC}	6.5V
Operating Temperature Range	MK50808 0°C to +70°C MK50808-1 -40° to +85°C
Storage Temperature Range	-65° to +150°C
Power Dissipation at 25°C	500mW
Voltage at any Pin except Digital Inputs	-0.3 to V_{CC} + 0.3V
Voltage at Digital Inputs	-0.3 to +15V

*Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL OPERATING CHARACTERISTICS

MK50808, MK50808-1 (Note 1)

SYM	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
V_{CC}	Power Supply Voltage	Measured at V_{CC} Pin	4.75	5.00	5.25	V	
V_{LADDER}	Voltage Across Ladder	From REF(+) to REF(-)	0.512	5.12	5.25	V	2
$V_{REF}(+)$	Voltage at Top of Ladder	Measured at REF(+)		V_{CC}	$V_{CC}+0.1$	V	
$\frac{\left(V_{REF}(+)+V_{REF}(-)\right)}{2}$	Voltage at Center of Ladder	Measured at $R_{LADDER}/2$	$\frac{V_{CC}}{2}-0.1$	$\frac{V_{CC}}{2}$	$\frac{V_{CC}}{2}+0.1$	V	
$V_{REF}(-)$	Voltage at Bottom of Ladder	Measured at REF(-)	-0.1	0		V	

DC CHARACTERISTICS

All parameters are 100% tested at 25°C. Device parameters are characterized at high and low temperature limits to assure conformance with the specification.

MK50808, MK50808-1

$4.75 \leqslant V_{CC} \leqslant 5.25V$, $-40 \leqslant T_A \leqslant +85°C$ unless otherwise noted

SYMBOL	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
V_{INHIGH}	Logic Input High Voltage	V_{CC} = 5V	3.5			V	
V_{INLOW}	Logic Input Low Voltage	V_{CC} = 5V			1.5	V	
$V_{OUTHIGH}$	Logic Output High Voltage	I_{OUT} = -360μA	V_{CC} - 0.4			V	
V_{OUTLOW}	Logic Output Low Voltage	I_{OUT} = 1.6mA			0.4	V	
I_{INHIGH}	Logic Input High Current	V_{IN} = 15V			1.0	μA	
I_{INLOW}	Logic Input Low Current	V_{IN} = OV	-1.0			μA	
I_{CC}	Supply Current	Clk. Freq=500kHz Clk. Freq=640kHz		300	1000 1300	μA μA	
I_{OUT}	Three-State Output Current	V_{OUT} = V_{CC} V_{OUT} = OV	 -3		3 	μA μA	

DC CHARACTERISTICS

MK50808-1, $-40 \leqslant T_A \leqslant +85°C$; MK50808, $0° \leqslant T_A \leqslant +70°C$

SYMBOL	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
P_{SR}	Power Supply Rejection	$4.75 \leqslant V_{CC} = V_{REF}(+) \leqslant 5.25V$; $V_{REF}(-)$=GND		0.05	0.15	%/V	10
R_{LADDER}	Ladder Resistance	From REF(+) to REF (-)	3.8	7		kΩ	

ANALOG MULTIPLEXER

MK50808, MK50808-1

$-40° \leqslant T_A \leqslant +85°C$ unless otherwise noteo

SYMBOL	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
I_{ON}	On-Channel Input Current	f_C = 640kHz During Conversion	-2	± .05	+2	μA	11
$I_{OFF}(+)$	Off - Channel Leakage Current	V_{CC}=5V, V_{IN}=5V, T_A=25°C		10	200	nA	
$I_{OFF}(-)$	Off - Channel Leakage Current	V_{CC}=5V, V_{IN}=OV, T_A = 25°C	-200	-10		nA	

CONVERTER SECTION

$V_{CC} = V_{REF}(+) = 5V, V_{REF}(-) = GND, V_{IN} = V_{COMPARATOR\ IN}, f_C = 640kHz$

MK50808-1, $-40 \leq T_A \leq +85°C$ unless otherwise noted

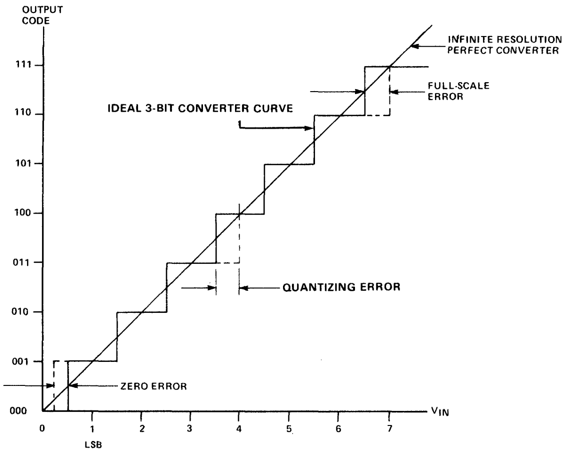
PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
Resolution				8	Bits	
Non-Linearity Error			± ¼	± ½	LSB	3
Zero Error			± ¼	± ½	LSB	5
Full-Scale Error			± ¼	± ½	LSB	6
Total Unadjusted Error	$T_A = 25°C$		± ¼ ± ¼	± ½ ± ¾	LSB LSB	7
Quantizing Error				±½	LSB	8
Absolute Accuracy	$T_A = 25°C$		± ¾ ± ¾	±1 ± 1¼	LSB LSB	9

MK50808, $0° \leq T_A \leq +70°C$

PARAMETER	MIN	TYP	MAX	UNITS	NOTES
Resolution			8	Bits	
Non-Linearity Error		± ½	± 1	LSB	3
Zero Error		± ¼	± ½	LSB	5
Full-Scale Error		± ¼	± ½	LSB	6
Total Unadjusted Error		± ½	± 1	LSB	7
Quantizing Error			± ½	LSB	8
Absolute Accuracy		± 1	± 1½	LSB	9

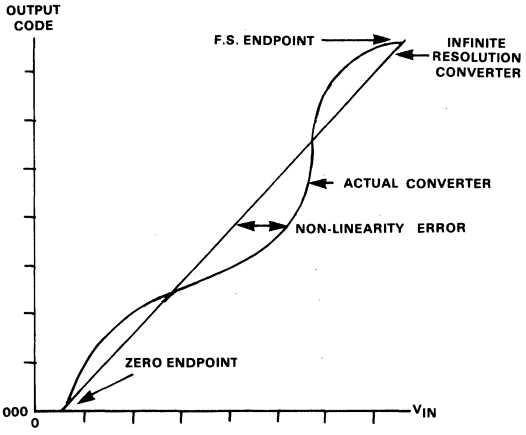
FULL-SCALE, QUANTIZING AND ZERO ERROR

Figure 5



NON-LINEARITY ERROR

Figure 6



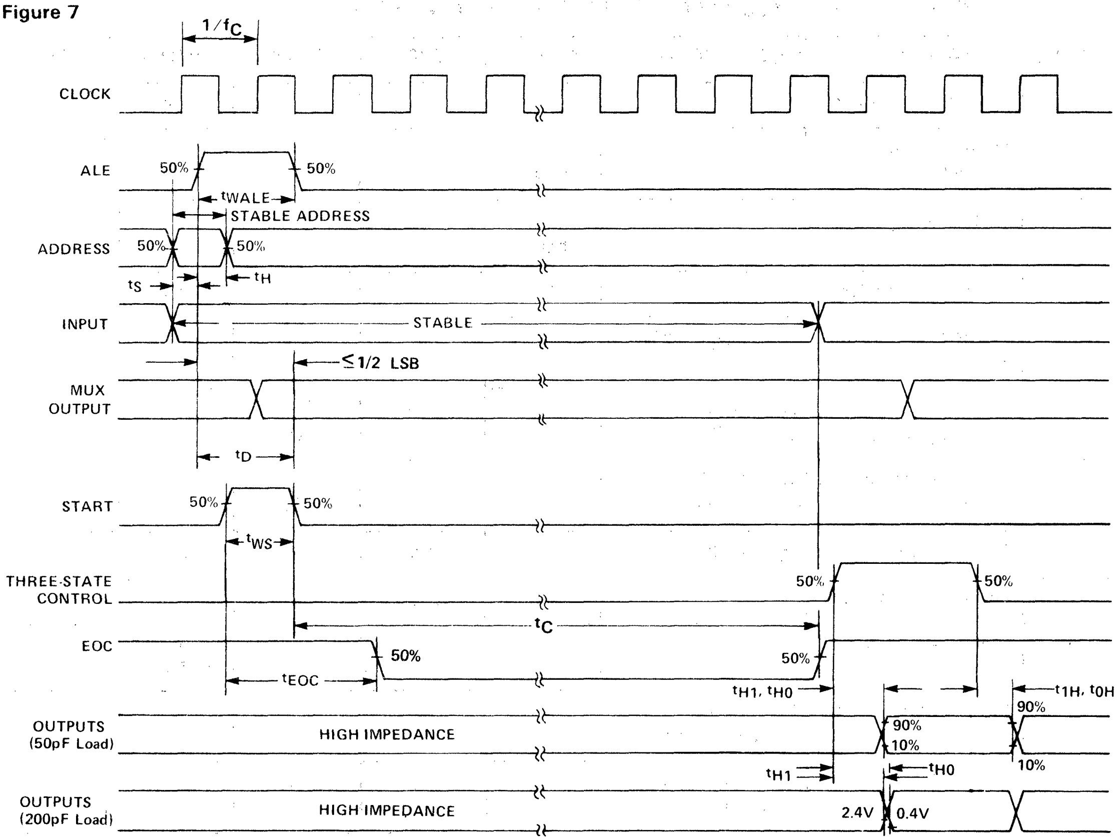
AC CHARACTERISTICS (Figure 7)

MK50808, MK50808-1, T_A = 25°C, $V_{CC} = V_{REF}(+)$ = 5V or 5.12V, $V_{REF}(-)$ = GND

SYMBOL	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
t_{WS}	Start Pulse Width		200			ns	
t_{WALE}	Minimum ALE Pulse Width		200			ns	
t_S	Address Set-Up Time		50			ns	
t_H	Address Hold Time		50			ns	
t_D	Analog MUX Delay Time from ALE	$R_S + R_{ON} \leqslant 5k\Omega$		1	2.5	μs	12
t_{H1}, t_{H0}	Three-State Control to Q Logic State	C_L = 50pF C_L = 200pF		125	250 300	ns ns	
t_{1H}, t_{0H}	Three-State Control to Hi-Z	C_L = 10pF, R_L = 10kΩ		125	250	ns	
t_C	Conversion Time	f_C= 640kHz	106	108	110	μs	
f_C	External Clock Freq.		100	640	1200	kHz	
t_{EOC}	EOC Delay Time		0		2	Clock Periods	4
C_{IN}	Input Capacitance	At Logic Inputs At MUX Inputs		10 5	15 7.5	pF pF	
C_{OUT}	Three-State Output Capacitance	At Three-State Outputs		5	7.5	pF	

TIMING DIAGRAM

Figure 7



NOTES:

1. All voltages are measured with respect to GND.
2. The minimum value for V_{LADDER} will give 2mV resolution. However, the guaranteed accuracy is only that which is specified under "DC Characteristics"
3. Non-linearity error is the maximum deviation from a straight line through the end points of the A/D transfer characteristics, Figure 6.
4. When EOC is tied to START, EOC delay is 1 clock period.
5. Zero Error is the difference between the actual input voltage and the design input voltage which produces a zero output code, Figure 5.
6. Full-Scale Error is the difference between the actual input voltage and the design input voltage which produces a full-scale output code, Figure 5.
7. Total Unadjusted Error is the true measure of accuracy the converter can provide less any quantizing effects.
8. Quantizing Error is the ±½ LSB uncertainty caused by the converter's finite resolution, Figure 5.
9. Absolute Accuracy is the difference between the actual input voltage and the full-scale weighted equivalent of the binary output code. This includes quantizing and all other errors.
10. Power Supply Rejection is the ability of an ADC to maintain accuracy as the power supply voltage varies. The power supply and $V_{REF}(+)$ are varied together and the change in accuracy is measured with respect to full-scale.
11. Input Current is the time average current into or out of the chopper-stabilized comparator. This current varies directly with clock frequency and has little temperature dependence.
12. This is the time required for the output of the analog multiplexer to settle within ±½ LSB of the selected analog input signal.

PRELIMINARY

MOSTEK®

INDUSTRIAL PRODUCTS

μP-Compatible A/D Converter

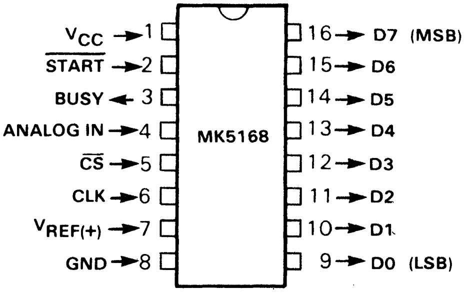
MK5168(N)-1

FEATURES

- □ Complete system in 16-pin package operates stand-alone or μP-driven
- □ Optional Clocks - external signal or internal oscillator
- □ Easy microprocessor interface
- □ Bus-compatible, 3-state data outputs
- □ Single 5 volt supply
- □ Low power - 1.5 mW typical
- □ ≤ ± ½ LSB total unadjusted error
- □ No full-scale or zero adjust required
- □ Guaranteed monotonicity
- □ No missing codes

PIN CONNECTIONS

Figure 1



DESCRIPTION

The MK5168 is an 8-bit, μP-compatible A/D Converter using the successive-approximations technique. CMOS construction provides low-power operation and the 16-pin package saves valuable board space, keeping system costs low.

The MK5168 A/D Converter is designed to interface with microprocessors or operate as a stand-alone subsystem. The A/D converter consists of 256 series resistors with an analog switch array, a chopper-stabilized comparator, and a successive approximation register. The series resistor approach guarantees monotonicity and no missing codes. The need for external zero and full-scale adjustments has been eliminated and an absolute accuracy of ≤ 1 LSB, including quantizing error, is provided.

All digital inputs are CMOS compatible. The data outputs D0 to D7 are 3-state latches providing true bus-driving capability (250 ns from $\overline{\text{CS}}$ to a valid logic level with 56 pF load). A $\overline{\text{START}}$ signal starts the conversion process and, upon completion, BUSY is driven to logic 0. Continuous conversion is possible by tying the $\overline{\text{START}}$ pin to the BUSY pin. The CLK pin may be connected to an external signal or tied to ground to enable the on-chip oscillator.

The MK5168 features high accuracy, minimal temperature dependence, and excellent long-term accuracy and repeatability, characteristics which make this device ideally suited to machine and industrial controls. A block diagram of a microprocessor control system using the MK5168 is shown in Figure 3.

FUNCTIONAL DESCRIPTION (Refer to Figure 2 for Block Diagram)

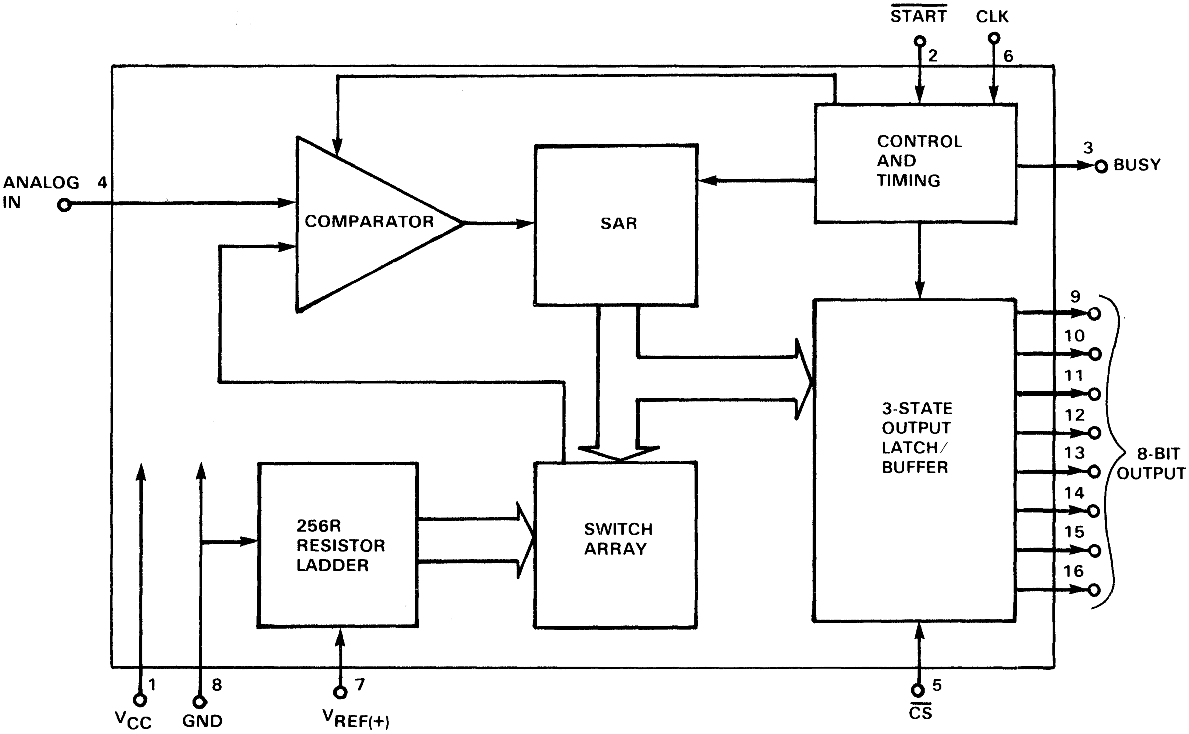
V_{CC}, Pin 1

V_{CC} must be connected to +5 Vdc ±5%.

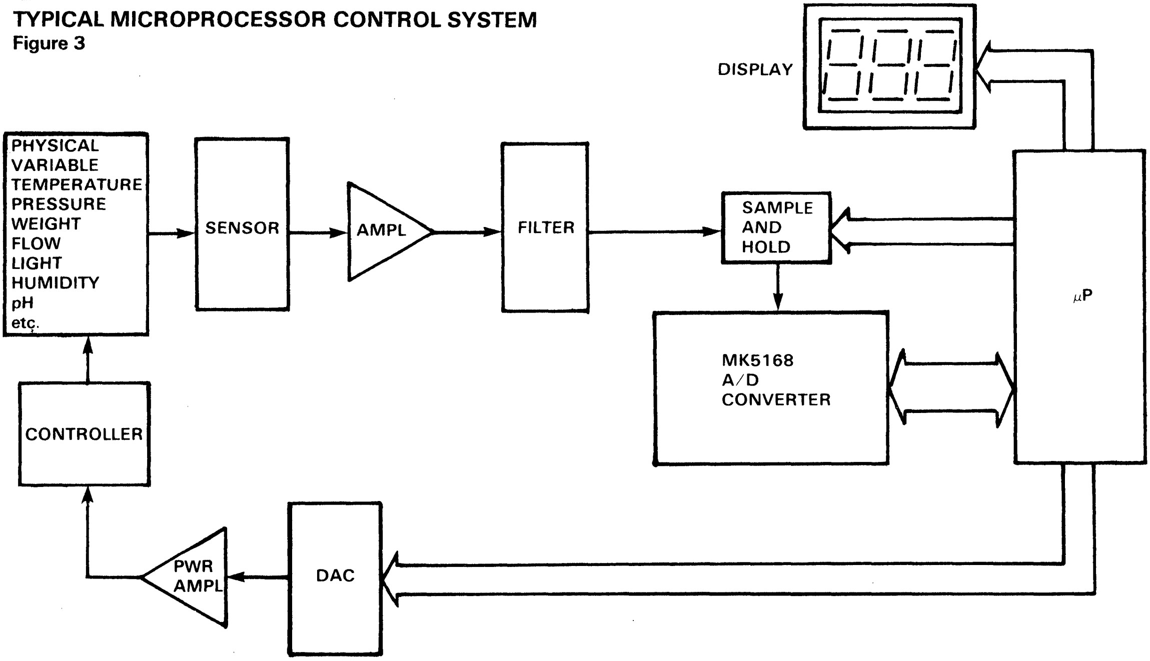
$\overline{\text{START}}$, Pin 2

The A/D Converter's successive approximation register (SAR) is reset by the falling edge of the $\overline{\text{START}}$ pulse. Conversion begins on the rising edge of the $\overline{\text{START}}$ pulse. A conversion in progress will be interrupted if a new $\overline{\text{START}}$ pulse is received and a new conversion will begin.

MK5168 BLOCK DIAGRAM
Figure 2



TYPICAL MICROPROCESSOR CONTROL SYSTEM
Figure 3



BUSY, Pin 3

The BUSY output goes low when the conversion process has been completed. The falling edge of the BUSY output indicates a valid digital output. Continuous conversion can be accomplished by tying the BUSY output to the $\overline{\text{START}}$ input. If the A/D Converter is used in this mode, an external $\overline{\text{START}}$ conversion pulse should be applied after power up. BUSY will go high within two clock periods after the positive edge of the $\overline{\text{START}}$ pulse.

ANALOG IN, Pin 4

The ANALOG INPUT accepts an analog signal from 0 V to V_{CC}.

The comparator is the most important section of the A/D Converter because this section determines the ultimate accuracy of the entire converter. It is the dc drift of the comparator which determines the repeatability of the device. A chopper-stabilized comparator was chosen because it best satisfies all the converter requirements.

The chopper-stabilized comparator converts the dc input signal into an ac signal. This signal is amplified by a high-gain ac amplifier and the dc level is restored. This technique limits the drift component of the comparator because the drift is a dc component which is not passed by the ac amplifier.

Since drift is virtually eliminated, the entire A/D Converter is insensitive to temperature and exhibits little long-term drift and input offset error.

$\overline{\text{CS}}$, Pin 5

The $\overline{\text{CHIP SELECT}}$ ($\overline{\text{CS}}$) allows the converter to be connected to an 8-bit data bus. A high level applied to this input causes the digital outputs to go to a high impedance state and a low level applied causes the digital outputs to go to valid logic levels.

CLK, Pin 6

The CLOCK input (CLK) will accept an external clock input from 100 kHz to 1.2 MHz. For an external clock signal to be recognized by the MK5168, the signal must have a duty cycle from 20% to 80%.

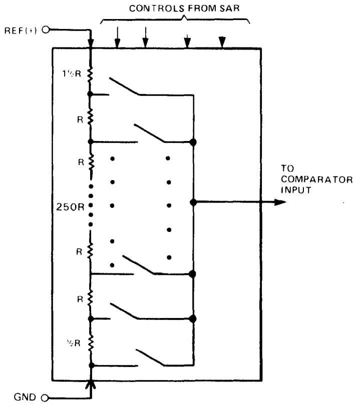
If CLK is grounded, the conversion process will be controlled by an on-chip oscillator, resulting in a typical conversion time of 150 μs.

$V_{REF(+)}$, Pin 7

This input supplies the voltage reference for the A/D Converter. Internal voltage references are derived from $V_{REF(+)}$ and GND by a 256 resistor ladder network, as shown in Figure 4. $V_{REF(+)}$ may be tied to V_{CC} or to a higher precision 5 V source for greater noise immunity.

RESISTOR LADDER AND SWITCH ARRAY

Figure 4



This approach was chosen because of its inherent monotonicity. A non-monotonic transfer characteristic can cause oscillations within a closed-loop feedback system.

The top and bottom resistors of the ladder network in Figure 4 are not the same value as the rest of the resistors in the ladder. They are chosen so that the output characteristic will be symmetrical about the full-scale and zero points. The first output transition occurs when the analog signal reaches +½ LSB and succeeding transitions occur every 1 LSB until the output reaches full-scale.

GND, Pin 8

All inputs and outputs are referenced to GROUND (GND), which is defined as 0 V and 0 logic level.

DIGITAL OUTPUT, Pins 9-16
D0-D7

These pins supply the digital output code which corresponds to the analog input voltage. D0 is the least significant bit (LSB) and D7 is the most significant bit (MSB). This output is stored in a TTL-compatible, 3-state output latch which can drive a 56 pF bus from high impedance to either logic state in 250 ns. Each pin can drive one standard TTL load directly without a pull-up resistor.

ABSOLUTE MAXIMUM RATINGS* (Note 1)

Absolute Maximum V_{CC}	6.5 V
Operating Temperature Range	-40° to +85°C
Storage Temperature Range	-65° to +150°C
Power Dissipation at 25°C Ambient	500 mW
Voltage at any pin except Digital Inputs	-0.3 to V_{CC} + 0.3 V
Voltage at Digital Inputs	-0.3 to +15 V

*Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL OPERATING CHARACTERISTICS

MK5168-1 (Note 1)

SYMBOL	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
V_{CC}	Power Supply Voltage	Measured at V_{CC} pin	4.75	5.00	5.25	V	
$V_{REF(+)}$	Voltage Across Ladder	From $V_{REF(+)}$ to GND	V_{CC}-0.12		V_{CC}+0.12	V	

DC CHARACTERISTICS

MK5168-1

$4.75 \leq V_{CC} \leq 5.25$ V, $-40 \leq T_A \leq +85$°C unless otherwise noted

SYMBOL	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
V_{INHIGH}	Logic Input High Voltage	V_{CC} = 5 V	3.5			V	
V_{INLOW}	Logic Input Low Voltage	V_{CC} = 5 V			1.5	V	
$V_{OUTHIGH}$	Logic Output High Voltage	I_{OUT} = -360 μA	V_{CC} - 0.4			V	
V_{OUTLOW}	Logic Output Low Voltage	I_{OUT} = 1.6 mA			0.5	V	
I_{INHIGH}	Logic Input High Current	V_{IN} = 15 V			1.0	μA	
I_{INLOW}	Logic Input Low Current	V_{IN} = 0 V	-1.0			μA	
I_{CC}	Supply Current	Clk Freq=500 kHz Clk Freq=640 kHz		300	1000 1300	μA μA	
I_{LEAK}	High Impedance Output Current	V_{OUT} = V_{CC} V_{OUT} = 0 V	 -3		3	μA μA	

DC CHARACTERISTICS

MK5168-1 -40 ≤ T_A ≤ +85°C,

SYMBOL	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
R_{PS}	Power Supply Rejection	4.75 ≤ V_{CC} ≤ 5.25 $V_{REF(+)} = V_{CC}$		0.05	0.15	%/V	9
$I_{COMP\ IN}$	Comparator Input Current	During Conversion f_c = 640 kHz	-2	±0.5	2	μA	10
R_{LADDER}	Ladder Resistance	From $V_{REF(+)}$ to GND	3.3	7		kΩ	

CONVERTER SECTION

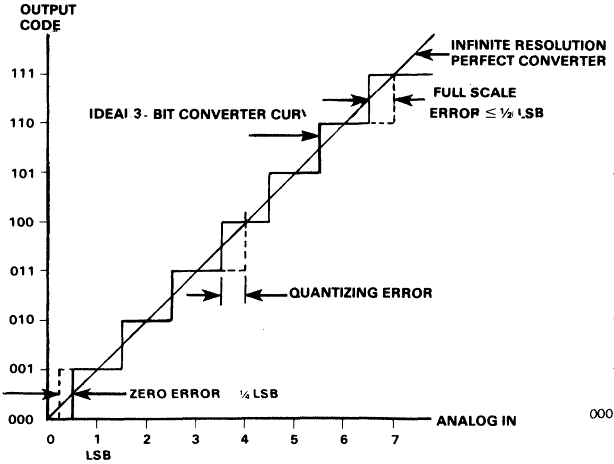
$V_{CC} = V_{REF(+)}$ = 5 V
f_c = 640 kHz
MK5168-1 -40 ≤ T_A ≤ + 85°C unless otherwise noted

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
Resolution				8	Bits	
Non-Linearity Error			± ¼	± ½	LSB	2
Zero Error			± ¼	± ½	LSB	4
Full-Scale Error			± ¼	± ½	LSB	5
Total Unadjusted Error	 T_A = 25°C		± ¼ ± ¼	± ¾ ± ½	LSB LSB	6 6
Quantizing Error				± ½	LSB	7
Absolute Accuracy	 T_A = 25°C		± ¾ ± ¾	± 1¼ ± 1	LSB LSB	8 8

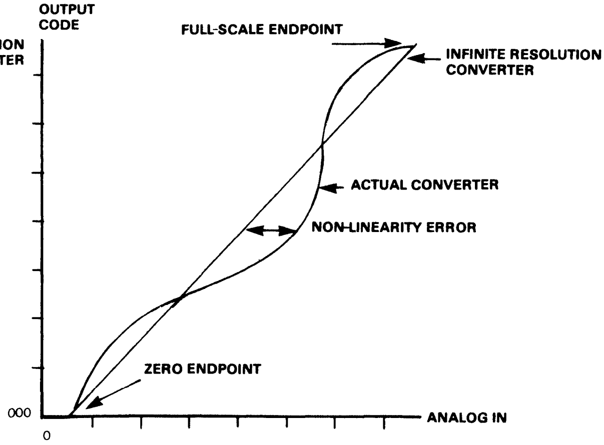
AC CHARACTERISTICS (Reference Figure 7)
MK5168-1 T_A = 25°C, V_{CC} = V_{REF} (+) = 5 V or 5.12 V

SYMBOL	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
t_{START}	START Pulse Width		200			ns	
t_{CSQ}	Chip Select Time to Valid Logic Levels On Digital Outputs	C_L =56 pF C_L =200 pF		125 300	250	ns ns	
t_{CSO}	Time to HI-Z From $\overline{CS}$ = V_{CC}	C_L = 10 pF R_L = 10 kΩ		125	250	ns	
t_c	Conversion Time	f_c = 640 kHz f_c = $f_{Internal\ Clock}$ f_c = 1200 kHz	106 57	108 150 58	110 59	μs μs μs	
f_c	External Clock Freq.		100	640	1200	kHz	11
t_{BUSY}	BUSY Delay Time		0		2	Clock Periods	3
C_{IN}	Input Capacitance	At Logic Inputs		10	15	pF	
C_{OUT}	Output Capacitance	At Digital Outputs $\overline{CS}$=V_{CC}		5	7.5	pF	

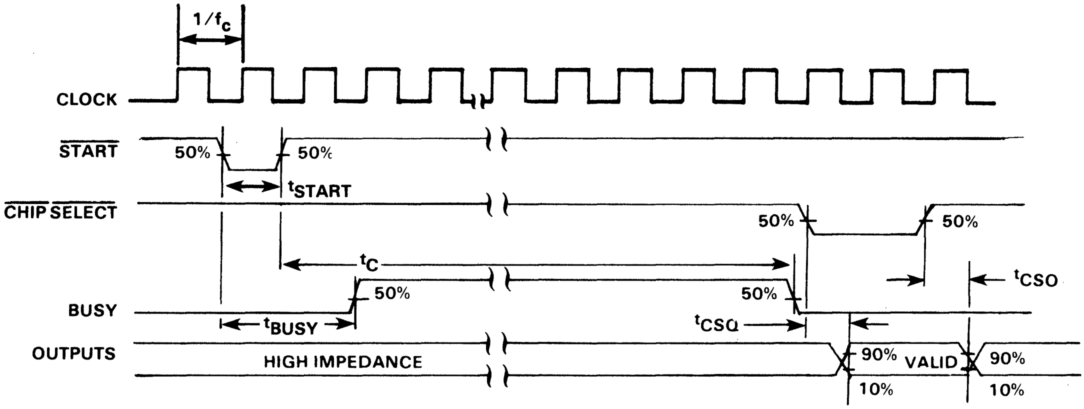
FULL-SCALE, QUANTIZING AND ZERO ERROR
Figure 5



NON-LINEARITY ERROR
Figure 6



TIMING DIAGRAM
Figure 7



NOTES:

1. All voltages are measured with respect to GND.
2. Non-linearity error is the maximum deviation from a straight line through the end-points of the A/D transfer characteristic. (Figure 6)
3. When BUSY is tied to START, BUSY delay is 1 clock period.
4. Zero Error is the difference between the actual input voltage and the design input voltage which produces a zero output code. (Figure 5)
5. Full-Scale Error is the difference between the actual input voltage and the design input voltage which produces a full-scale output code. (Figure 5)
6. Total Unadjusted Error is the true measure of accuracy the converter can provide less any quantizing effects.
7. Quantizing Error is the ±½ LSB uncertainty caused by the converter's finite resolution. (Figure 5)
8. Absolute Accuracy is the difference between the actual input voltage and the full-scale weighted equivalent of the binary output code. This includes quantizing and all other errors.
9. Power Supply Rejection is the ability of an ADC to maintain accuracy as the power supply voltage varies. The power supply and $V_{REF(+)}$ are varied together and the change in accuracy is measured with respect to full-scale.
10. Comparator Input Current is the time average current into or out of the chopper stabilized comparator. This current varies directly with clock frequency and has little temperature dependence.
11. A minimum duty cycle of 20% is required at the clock input.

MOSTEK®

8-BIT A/D CONVERTER/16-CHANNEL ANALOG MULTIPLEXER

MK50816(N/P)

FEATURES

□ Single 5 Volt Supply (± 5%)

□ Low Power Dissipation - 6.825mW(max) at 640kHz

□ Total Unadjusted Error < ± ½ LSB

□ Linerarity Error < ± ½ LSB

□ No Missing Codes

□ Guaranteed Monotonicity

□ No Zero Adjust Required

□ No Full-Scale Adjust Required

□ 108μs Conversion Time (Typically)

□ Easy Microprocessor Interface

□ Latched TTL Compatible Three-State Output with True Bus-Driving Capability

□ Expandable 16-channel Analog Multiplexer

□ Latched Address Input

□ Fixed Reference or Ratiometric Conversion

□ Continuous or Controlled Conversion

□ On-Chip or External Clock

□ On-Chip Chopper-Stabilized Comparator

□ Low Reference-Voltage Current Drain

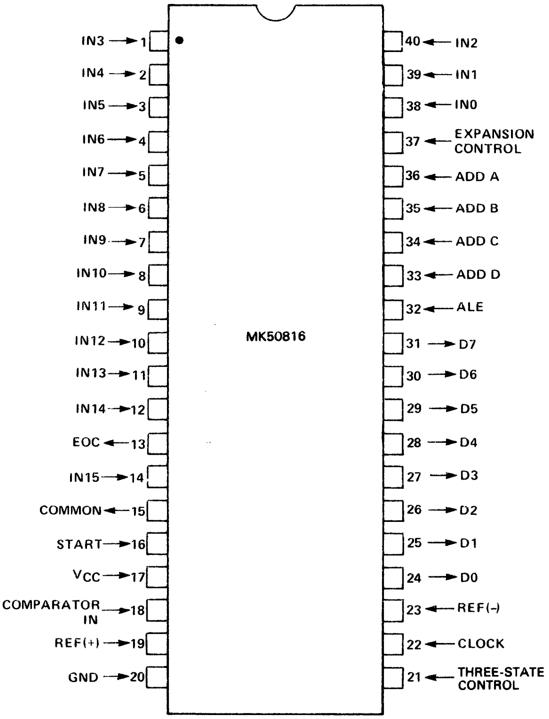
DESCRIPTION

The MK50816 is a monolithic CMOS device with an 8-bit successive approximation A/D converter, a 16-channel analog multiplexer and microprocessor-compatible control logic. The 16-channel multiplexer can directly access any one of 16 single-ended analog channels and provides logic for additional channel expansion. The 8-bit A/D converter consists of 256 series resistors with an analog switch array, a chopper-stabilized comparator and a successive approximation register. The series resistor approach guarantees monotonicity and no missing codes as well as allowing both ratiometric and fixed-reference measurements. The need for zero and full-scale adjustments has been eliminated and an absolute accuracy of ≤ 1 LSB, including quantizing error, is provided.

The pin configuration of the MK50816 is shown in Figure 1 below:

PIN CONNECTIONS

Figure 1

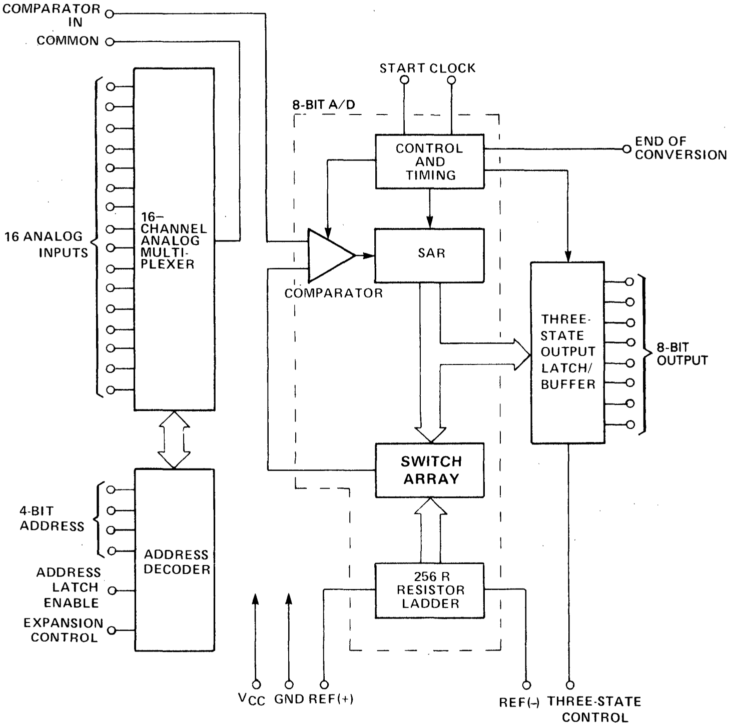


All digital outputs are TTL-compatible, all digital inputs are TTL-compatible with a pull-up resistor, and all digital inputs and outputs are CMOS-compatible; this makes it easy to interface with most microprocessors. The output latch is three-state and provides true bus-driving capability (300ns from Three-State Control to Q Logic State with 200pF load). A Start Convert signal initiates the conversion process, and, upon completion, an End Of Conversion signal is generated. Continuous conversion is possible by tying the Start-Convert pin to the End-of-Conversion pin. The clock pin may be connected to an external oscillator or tied to ground to enable an on-chip oscillator.

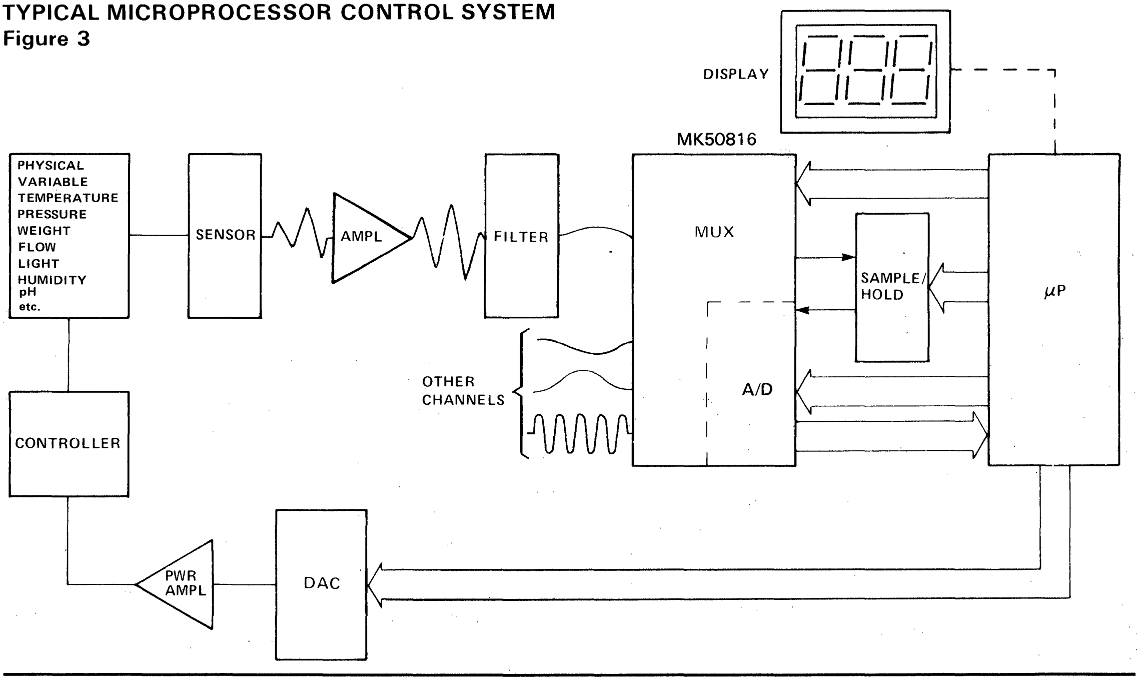
The MK50816 features low power, high accuracy, minimal temperature dependence, and excellent long-term accuracy and repeatability. These characteristics make this device ideally suited to machine and industrial controls.

A block diagram of a microprocessor control system using the MK50816 is shown in Figure 3.

MK50816 BLOCK DIAGRAM
Figure 2



TYPICAL MICROPROCESSOR CONTROL SYSTEM
Figure 3



FUNCTIONAL DESCRIPTION (Refer To Figure 2 for a Block Diagram)

ADDRESS, Pins 33-36

The address decoder allows the 16-input analog multiplexer to select any one of 16 single-ended analog input channels. Table 1 shows the required address and expansion control inputs to select any analog input channel.

ADDRESS LATCH ENABLE, Pin 32

A positive transition applied to the Address Latch Enable (ALE) input latches a 4-bit address into the address decoder. ALE can be tied to Start with parameter t_D being satisfied.

COMMON OUTPUT, Pin 15

This is the output of the 16-channel analog multiplexer. The maximum ON resistance is 3kΩ.

EXPANSION CONTROL, Pin 37

Additional single-ended analog signals can be multiplexed to the A/D converter by holding the Expansion Control low, disabling the multiplexer. These additional externally-multiplexed signals are to be connected to the Comparator Input and the device ground. Additional signal conditioning such as sample-and-hold or instrumentation amplification can be added between the analog signal and the Comparator Input.

CLOCK INPUT, Pin 22

The Clock Input will accept an external clock input from 100kHz to 1.2MHz. A minimum duty cycle of 20% is required for the Clock Input to detect the presence of an external clock signal.

If the Clock pin is grounded, the conversion process will be controlled by an on-chip oscillator.

POSITIVE AND NEGATIVE REFERENCE VOLTAGES [REF (+) and REF (-)], Pins 19 and 23

These inputs supply voltage references for the analog-to-digital converter. Internal voltage references are derived from REF (+) and REF (-) by a 256-R ladder network, Figure 4.

This approach was chosen because of its inherent monotonicity, which is extremely important in closed-loop feedback control systems. A non-monotonic transfer characteristic can cause catastrophic oscillations within a system.

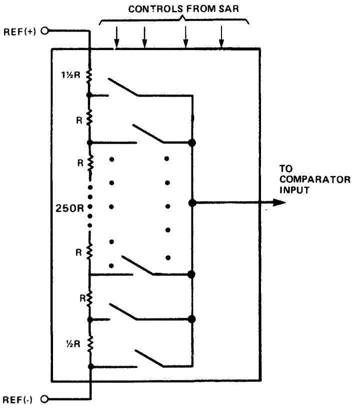
The top and bottom resistors of the ladder network in Figure 4 are not the same value as the rest of the resistors in the ladder. They are chosen so that the output characteristic will be symmetrical about its full-scale and zero points. The first output transition occurs when the analog signal reaches +½ LSB and succeeding transitions occur every 1 LSB until the output reaches full scale.

ANALOG CHANNEL SELECTION
Table 1

SELECTED ANALOG CHANNEL	ADDRESS LINE				EXPANSION CONTROL
	D	C	B	A	
IN0	L	L	L	L	H
IN1	L	L	L	H	H
IN2	L	L	H	L	H
IN3	L	L	H	H	H
IN4	L	H	L	L	H
IN5	L	H	L	H	H
IN6	L	H	H	L	H
IN7	L	H	H	H	H
IN8	H	L	L	L	H
IN9	H	L	L	H	H
IN10	H	L	H	L	H
IN11	H	L	H	H	H
IN12	H	H	L	L	H
IN13	H	H	L	H	H
IN14	H	H	H	L	H
IN15	H	H	H	H	H
All Channels OFF	X	X	X	X	L

X = don't care

RESISTOR LADDER AND SWITCH ARRAY
Figure 4



ANALOG INPUTS, PINS 1-12, 14, 38-40

These inputs are multiplexing analog switches which accept analog inputs from 0V to V_{CC}.

COMPARATOR INPUT, Pin 18

The comparator is the most important section of the A/D converter because this section determines the ultimate accuracy of the entire converter. It is the DC drift of the comparator which determines the repeatability of the device. A chopper-stabilized comparator was chosen because it best satisfies all the converter requirements.

The chopper-stabilized comparator converts the DC input signal into an AC signal. This signal is amplified by a high-gain AC amplifier and the DC level is restored. This technique limits the drift component of the comparator because the drift is a DC component which is not passed by the AC amplifier.

Since drift is virtually eliminated, the entire A/D converter is extremely insensitive to temperature and exhibits very little long-term drift and input offset error.

START, Pin 16

The A/D converter's successive approximation register (SAR) is reset by the positive edge of the Start pulse. Conversion begins on the falling edge of the Start pulse. A conversion in progress will be interrupted if a new start conversion pulse is received and a new conversion will begin.

END OF CONVERSION, Pin 13

The End Of Conversion (EOC) output goes high when the conversion process has been completed. The positive edge of the EOC output indicates a valid digital output. Continuous conversion can be accomplished by tying the EOC output to the Start input. If the A/D converter is used in this mode, an external start conversion pulse should be applied after power up. End of Conversion will go low within 2 clock periods after the positive edge of Start.

8-BIT DIGITAL OUTPUT, Pins 24-31

These pins supply the digital output code which corresponds to the analog input voltage. D0 is the least significant bit (LSB) and D7 is the most significant bit (MSB). This output is stored in a TTL-compatible three-state output latch which can drive a 200pF bus from high impedance to either logic state in 300ns. Each pin can drive one standard TTL load.

THREE-STATE CONTROL, Pin 21

The Three-State Control allows the converter to be connected to an 8-bit data bus. A low level applied to this input causes the digital output to go to a high impedance state and a high level causes the output to go to a Q logic state.

ABSOLUTE MAXIMUM RATINGS* (Note 1)

Absolute Maximum V_{CC}	6.5V
Operating Temperature Range	MK50816 0° to +70°C
	MK50816-1 -40°C to +85°C
Storage Temperature Range	-65°C to +150°C
Power Dissipation at 25°C	500mW
Voltage at any Pin except Digital Inputs	-0.3 to V_{CC} + 0.3V
Voltage at Digital Inputs	-0.3 to +15V

*Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL OPERATING CHARACTERISTICS
MK50816, MK50816-1 (Note 1)

SYM	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
V_{CC}	Power Supply Voltage	Measured at V_{CC} Pin	4.75	5.00	5.25	V	
V_{LADDER}	Voltage Across Ladder	From REF(+) to REF (-)	0.512	5.12	5.25	V	2
$V_{REF}(+)$	Voltage at Top of Ladder	Measured at REF (+)		V_{CC}	$V_{CC}+0.1$	V	
$\frac{\left(V_{REF}(+) + V_{REF}(-)\right)}{2}$	Voltage at Center of Ladder	Measured at $R_{LADDER}/2$	$\frac{V_{CC}}{2} - 0.1$	$\frac{V_{CC}}{2}$	$\frac{V_{CC}}{2} + 0.1$	V	
$V_{REF}(-)$	Voltage at Bottom of Ladder	Measured at REF(-)	-0.1	0		V	

DC CHARACTERISTICS

All parameters are 100% tested at 25°C. Device parameters are characterized at low and high temperature limits to assure conformance with the specification.

MK50816, MK50816-1

$4.75 \leq V_{CC} \leq 5.25V$, $-40 \leq T_A \leq +85°C$ unless otherwise noted

SYM	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
V_{INHIGH}	Logic Input High Voltage	$V_{CC} = 5V$	3.5			V	
V_{INLOW}	Logic Input Low Voltage	$V_{CC} = 5V$			1.5	V	
$V_{OUTHIGH}$	Logic Output High Voltage	$I_{OUT} = -360\mu A$	$V_{CC} - 0.4$			V	
V_{OUTLOW}	Logic Output Low Voltage	$I_{OUT} = 1.6mA$			0.4	V	
I_{INHIGH}	Logic Input High Current	$V_{IN} = 15V$			1.0	μA	
I_{INLOW}	Logic Input Low Current	$V_{IN} = 0V$	-1.0			μA	
I_{CC}	Supply Current	Clk Freq=500kHz Clk Freq=640kHz		300	1000 1300	μA μA	
I_{OUT}	Three-State Output Current	$V_{OUT}=V_{CC}$ $V_{OUT}=0V$	 -3		3 	μA μA	

DC CHARACTERISTICS

MK50816-1 $-40 \leq T_A \leq +85°C$, MK50816 $0° \leq T_A \leq +70°C$

SYM	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
R_{PS}	Power Supply Rejection	$4.75 \leq V_{CC} \leq 5.25$ $V_{REF}(+) = V_{CC}$ $V_{REF}(-) = GND$		0.05	0.15	%/V	10
$I_{COMP\ IN}$	Comparator Input Current	$f_C = 640kHz$ During Convs.	-2	± 0.5	2	μA	11
R_{LADDER}	Ladder Resistance	From REF(+) to REF (-)	3.8	7		kΩ	

ANALOG MULTIPLEXER
MK50816, MK50816-1

-40° ≤ T_A ≤ +85°C unless otherwise noted

SYM	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
R_{ON}	Analog Multiplexer ON Resistance	(Any Selected Channel) T_A = 25°C, R_L = 10k		1.5	3	kΩ	
ΔR_{ON}	Δ ON Resistance Between Any 2 Channels	(Any Selected Channel) R_L = 10k		75		Ω	
$I_{OFF}(+)$	OFF Channel Leakage Current	V_{CC}=5V, V_{IN}=5V, T_A=25°C		10	200	nA	
$I_{OFF}(-)$	OFF Channel Leakage Current	V_{CC}=5V, V_{IN}=0V, T_A=25°C	-200	-10		nA	

CONVERTER SECTION

V_{CC} = $V_{REF}(+)$ = 5V, $V_{REF}(-)$ = GND, V_{IN} = $V_{COMPARATOR\ IN}$,
f_C = 640kHz
MK50816-1 -40 ≤ T_A ≤ +85°C unless otherwise noted

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
Resolution				8	Bits	
Non-Linearity Error			± ¼	± ½	LSB	3
Zero Error			± ¼	± ½	LSB	5
Full-Scale Error			± ¼	± ½	LSB	6
Total Unadjusted Error	T_A = 25°C		± ¼ ± ¼	± ½ ± ¾	LSB LSB	7
Quantizing Error				± ½	LSB	8
Absolute Accuracy	T_A = 25°C		± ¾ ± ¾	± 1 ± 1¼	LSB LSB	9

MK50816 0° ≤ T_A ≤ +70°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
Resolution				8	Bits	
Non-Linearity Error			± ½	± 1	LSB	3
Zero Error			± ¼	± ½	LSB	5
Full-Scale Error			± ¼	± ½	LSB	6
Total Unadjusted Error			± ½	± 1	LSB	7
Quantizing Error				± ½	LSB	8
Absolute Accuracy			± 1	± 1½	LSB	9

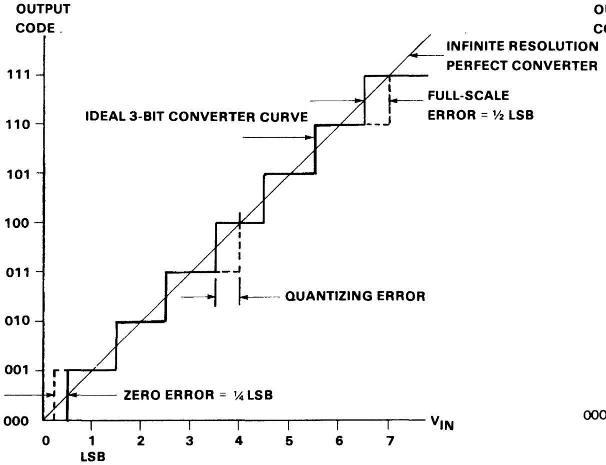
AC CHARACTERISTICS (Figure 7)

MK50816, MK50816-1 T_A = 25°C, V_{CC} = $V_{REF}(+)$ = 5V or 5.12V, $V_{REF}(-)$ = GND

SYM	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
t_{WS}	Start Pulse Width		200			ns	
t_{WALE}	Minimum ALE Pulse Width		200			ns	
t_S	Address Set-Up Time		50			ns	
t_H	Address Hold Time		50			ns	
t_D	Analog MUX Delay Time from ALE	Common Tied to Comparator In, $R_S + R_{ON} \leq 5k\Omega$, C_L = 10pF		1	2.5	µs	12
t_{H1}, t_{H0}	Three-State Control to Q Logic State	C_L = 50pF C_L = 200pF		125 300	250	ns ns	
t_{1H}, t_{0H}	Three-State Control to Hi-Z	C_L = 10pF, R_L = 10kΩ		125	250	ns	
t_C	Conversion Time	f_C = 640kHz f_C = $f_{INTERNAL\ CLOCK}$	106	108 150	110	µs µs	
f_C	External Clock Freq		100	640	1200	kHz	13
t_{EOC}	EOC Delay Time		0		2	Clock Periods	4
C_{IN}	Input Capacitance	At Logic Inputs At MUX Inputs		10 5	15 7.5	pF pF	
C_{OUT}	Three-State Output Capacitance	At Three-State Outputs		5	7.5	pF	

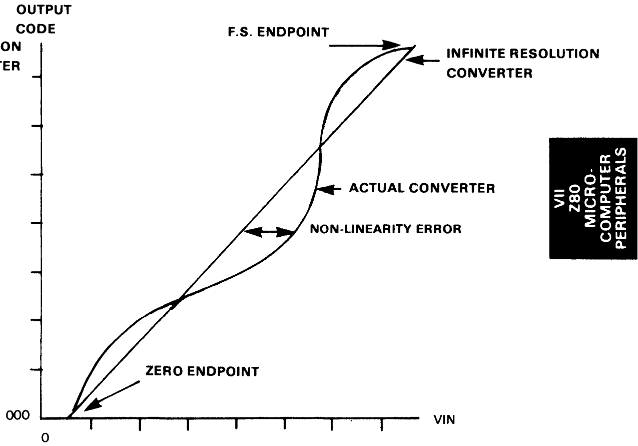
FULL SCALE, QUANTIZING AND ZERO ERROR

Figure 5



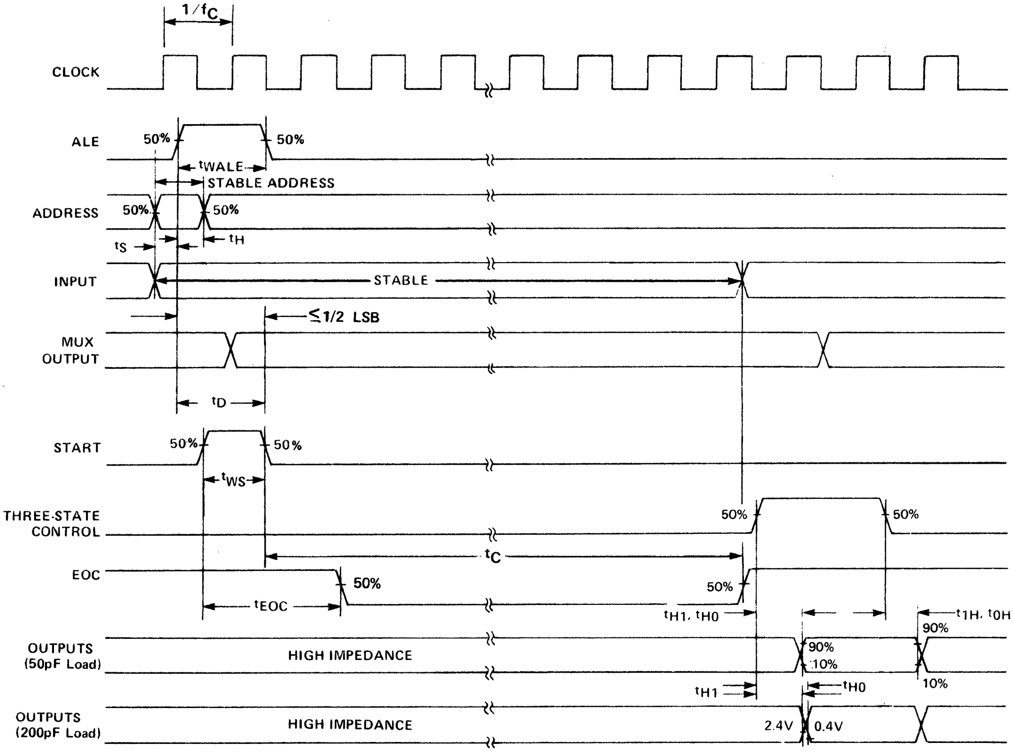
NON-LINEARITY ERROR

Figure 6



TIMING DIAGRAM

Figure 7



NOTES:

1. All voltages are measured with respect to GND.
2. The minimum value for V_{LADDER} will give 2mV resolution. However, the guaranteed accuracy is only that which is specified under "DC Characteristics".
3. Non-linearity error is the maximum deviation from a straight line through the end points of the A/D transfer characteristic, Figure 6.
4. When EOC is tied to START, EOC delay is 1 clock period.
5. Zero Error is the difference between the actual input voltage and the design input voltage which produces a zero output code, Figure 5.
6. Full-Scale Error is the difference between the actual input voltage and the design input voltage which produces a full-scale output code, Figure 5.
7. Total Unadjusted Error is the true measure of accuracy the converter can provide less any quantizing effects.
8. Quantizing Error is the ± ½ LSB uncertainty caused by the converter's finite resolution, Figure 5.
9. Absolute Accuracy is the difference between the actual input voltage and the full-scale weighted equivalent of the binary output code. This includes quantizing and all other errors.
10. Power Supply Rejection is the ability of an ADC to maintain accuracy as the power supply voltage varies. The power supply and $V_{REF}(+)$ are varied together and the change in accuracy is measured with respect to full-scale.
11. Comparator Input Current is the time average current into or out of the chopper-stabilized comparator. This current varies directly with clock frequency and has little temperature dependence.
12. This is the time required for the output of the analog multiplexer to settle within ± ½ LSB of the selected analog input signal.
13. A minimum duty cycle of 20% is required at the clock input.

1981 Z80 MICROCOMPUTER DATA BOOK

I Table of Contents

I TABLE OF CONTENTS

II General Information

II GENERAL INFORMA TION

III Z80 Family Technical Manuals

III Z80 FAMILY TECHNICAL MANUALS

IV Z80 Family Data Sheets

IV Z80 FAMILY DATA SHEETS

V Z80 Development Equipment

V Z80 DEVELOP MENT EQUIPMENT

VI Z80 Microcomputer Application Notes

VI Z80 MICRO COMPUTER APPLICATION

VII Z80 Microcomputer Peripherals

VII Z80 MICRO COMPUTER PERIPHERALS

VIII Z80 Military/ Hi-Rel

VIII Z80 MILITARY/ HI-REL

MILITARY/HI-RELIABILITY PRODUCTS

Table of Contents

Military and High Reliability Products

INTRODUCTION

Overview

Mostek's Military/Hi-Rel Products Department serves the special needs of the Defense, Aerospace and Commercial Hi-Rel markets. The organization's principal objective is to provide Mostek's state-of-the-art products screened to MIL-STD 883, Methods 5004 and 5005.

Traditional Military IC manufacturers have met stringent Military reliability requirements at a cost of being several years behind the state-of-the-art in commercial products. Mostek Military brings the leading edge in high reliability RAM, ROM, EPROM and microprocessor devices to the Military systems designer today. As MIL-M-38510 slash sheets are announced, the Military Products Department will qualify Mostek's products in the JAN 38510 program. Mostek has already received QPL listing of its 4116 dynamic RAM. Designated JM-38510/240, this device is one of the most advanced MOS circuits to receive QPL listing to date.

The Military Products Department is also heavily engaged in the development of high density leadless chip carrier packaging technology. Several circuits are currently offered in carriers with more planned for the near future.

Product offerings are broken into two categories. Devices prefixed "MKB" are screened to the full requirements of MIL-STD-883 Class B. "MKI" (Industrial grade) prefixed devices are screened to a subset of 883B requirements and offer high reliability at significantly reduced cost (see the following sections for more detail concerning MKI products).

Quality Conformance/Reliability

Mostek has been providing MKB versions of selected memory products since 1978. Quality conformance inspections in accordance with Method 5005 of Standard 883 are a central part of MKB screening procedures. Group A inspections are performed on a 100% basis to the requirements contained in the detail specification (Mostek data sheet). Group B, C and D inspections are performed periodically per the requirements of Standard 883.

A data report documenting the results of Group B, C and D testing through the present is available at no charge.

SCD Program

Support Customer Documentation

The Military/Hi-Rel Products Department has instituted a program to provide customers with source control drawings for nonstandard parts requirements, patterned after DESC "mini-spec" documents. This new documentation from Mostek is an effort to provide a reliable, detailed spec where no DESC drawing or slash sheet exists. This benefits the customer's procurement effort and should help hasten the development of a standard, government-approved, industry accepted specification.

All new military devices will have a Mostek SCD control document generated describing them. They will be written around DDL103 Guidelines for DESC Selected Item Drawings. Their usage in programs should be restricted to that period prior to the introduction of an appropriate DESC prepared Selected Item Drawing.

In sum, we feel the Mostek SCD offers customers a ready-made source control document customizable with his name and part number and acceptable to his ultimate customer - the U.S. Government.

SCD's AVAILABLE: MKB4118A

SCD's FORTHCOMING: MKB4801, 4802, 4164, 2764

For more information contact:

Mostek Corporation
Military Products Department
Mail Station 1100
1215 West Crosby Road
Carrollton, Texas 75006

Telephone - (214) 323-6250/7718
TWX - 910-860-5856
Telex - 730423

Applications Guide

APPLICATION	TYPE OF SYSTEM	MOSTEK TYPE
Military/Aerospace	Ground Airborne Tactical Missile Space	MKI/MKB MKB/JAN MKB/JAN Class S Equiv.
Industrial	Process Control Instrumentation Telecom	MKI
Automotive	Engine Control Instrumentation	MKI
Commercial	FAA Airborne	MKB/MKI
Medical	Instrumentation	MKI

"MKB" SCREENING: 100% tested to the detail procedures of MIL-STD-883, Method 5004 Class B and qualification and quality conformance procedures of Method 5005, Class B.

"MKI" SCREENING: Tested to a cost-effective hi-rel subset of 883B including extended burn-in. AQL levels are tightened.

MOSTEK®

MILITARY/HI-REL PRODUCTS

MKB Military Products Listing

Product	Device	Organization	Packages	Temp. Range	Access Times/Freq.	Active Power	Standby Power
JAN DYNAMIC RAMs (4116)	JM-38510/24001 BEC	16K x 1	P	-55°C/+110°C	200 ns	462 mw	30 mw
	JM-38510/24002 BEC	16K x 1	P	-55°C/+110°C	250 ns	462 mw	30 mw
DYNAMIC RAMs	MKB4116	16K x 1	E,F,J	-55°C/+110°C	150/200/250ns	462 mw	30 mw
	MKB4164†	64K x 1	E,P	-55°C/+85°C	150/200 ns	—	—
	MKM4332	32K x 1	D	-55°C/+110°C	200/250 ns	495 mw	60 mw
STATIC RAMs	MKB4104	4K x 1	E,J	-55°C/+125°C	250/300/350ns	150 mw	53 mw
	MKB2147H†	4K x 1	E,P	-55°C/+125°C	70/90/120 ns	—	—
	MKB4167	16K x 1	E,P	-55°C/+125°C	120 ns	—	—
	MKB4118	1K x 8	*E,P	-55°C/+125°C	150/200 ns	500 mw	400 mw
	MKB4801†	1K x 8	*E,P	-55°C/+125°C	90/120 ns	—	—
	MKB4802†	2K x 8	*E,P	-55°C/+125°C	90/120 ns	—	—
ROMs	MKB36000	8K x 8	P	-55°C/+125°C	250/300 ns	220 mw	55 mw
	MKB37000†	8K x 8	*E,P	-55°C/+125°C	300 ns	—	—
EPROMs	MKB2716	2K x 8	*E,J	-55°C/+100°C	390/450 ns	633 mw	165 mw
	MKB2764†	8K x 8	*E,T	-55°C/+100°C	450 ns	—	—
MICRO-PROCESSORS	MKB3880	Z80 CPU	P	-55°C/+125°C	2.5/4.0 MHz	—	—

*NOTE: Bytewyde™ Products in leadless chip carrier (E package) will become available beginning 2nd half, 1981.

†1981 Introduction

MKI Industrial Products Listing

Product	Device	Organi- zation	Packages	Temp. Range	Access Time Or Frequency
DYNAMIC RAMs	MKI 4116	16K x 1	J	-40°C/+85°C	150/200/250 ns
	MKI 4164†	64K x 1	J	-40°C/+85°C	150 ns
STATIC RAMs	MKI 4118	1K x 8	J	-40°C/+85°C	150/200 ns
	MKI 4802†	2K x 8	J	-40°C/+85°C	70/90/120 ns
ROMs	MKI 37000†	8K x 8	J	-40°C/+85°C	300 ns
EPROMs	MKI 2716	2K x 8	J	-40°C/+85°C	390/450 ns
	MKI 2764†	8K x 8	J	-40°C/+85°C	450 ns
MICROPROCESSORS	MKI 3880	Z80 CPU	P	-40°C/+85°C	2.5/4.0 MHz

†1981 Introduction

PRELIMINARY

MOSTEK®

Z80 CENTRAL PROCESSING UNIT

Processed to MIL-STD 883, Method 5004, Class B

MKB3880(P)-80/84

FEATURES

□ Screened per MIL-STD-883, Method 5004 Class B

□ -55°C to 125°C temperature range

□ Two speeds
- 2.5 MHz MKB3880)P)-80
- 4.0 MHz MKB3880(P)-84

□ Single 5-Volt supply and single-phase clock required

□ Z80 CPU and Z80 A CPU

□ Software compatible with 8080A CPU

□ Complete development and OEM system product support

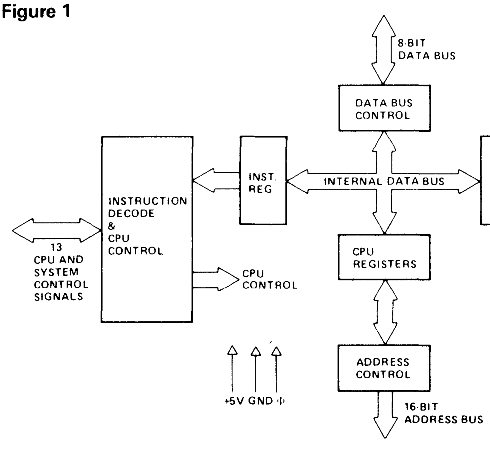
□ Industrial MKI version available (-40°C to 85°C)

DESCRIPTION

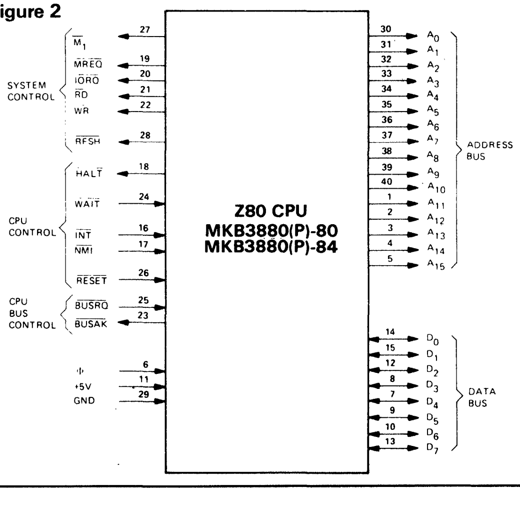
The Mostek Z80 family of components is a significant advancement in the state-of-art of microcomputers. These components can be configured with any type of standard semiconductor memory to generate computer systems with an extremely wide range of capabilities. For example, as few as two LSI circuits and three standard TTL MSI packages can be combined to form a simple controller. With additional memory and I/O devices, a computer can be constructed with capabilities that only a minicomputer could deliver previously. This wide range of computational power allows standard modules to be constructed by a user that can satisfy the requirements of an extremely wide range of applications.

The CPU is the heart of the system. Its function is to obtain instructions from the memory and perform the desired operations. The memory is used to contain instructions and, in most cases, data that is to be processed. For example, a typical instruction sequence may be to read data from a specific peripheral device, store it in a location in memory, check the parity, and write it out to another peripheral device. Note that the Mostek component set includes the CPU and various general purpose I/O device controllers, as well as a wide range of memory devices. Thus, all required components can be connected together in a very simple manner with virtually no other external logic. The usei s effort then becomes primarily one of the software development. That is, the user can concentrate on describing his problem and translating it into a series of instructions that can be loaded into the microcomputer memory. Mostek is dedicated to making this step of software generation as simple as possible. A good example of this is our assembly language in which a simple mnemonic is used to represent every instruction that the CPU can perform. This language is self-documenting in such a way that from the mnemonic the user can understand exactly what the instruction is doing without constantly checking back to a complex cross listing.

Z80-CPU BLOCK DIAGRAM
Figure 1



Z80 PIN CONFIGURATION
Figure 2



ELECTRICAL SPECIFICATIONS

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias .... -55°C to +125°C
Storage Temperature .... -65°C to +150°C
Voltage on Any Pin with Respect to Ground .... -0.3V to +7V
Power Dissipation 1.5W

*Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC CHARACTERISTICS

(T_A = -55°C to 125°C, V_{CC} = 5 V ± 5% unless otherwise specified)

SYM	PARAMETER	MIN	TYP	MAX	UNIT	TEST CONDITION
V_{ILC}	Clock Input Low Voltage	-0.3		0.8	V	
V_{IHC}	Clock Input High Voltage	V_{CC}-.6		V_{CC}+.3	V	
V_{IL}	Input Low Voltage	-0.3		0.8	V	
V_{IH}	Input High Voltage All inputs except NMI	2.4		V_{CC}	V	
$V_{IH(NMI)}$	Input High Voltage (NMI)	2.7		V_{CC}	V	
V_{OL}	Output Low Voltage			0.4	V	I_{OL} = 1.8mA
V_{OH}	Output High Voltage	2.4			V	I_{OH} = -250 μA
I_{CC}	Power Supply Current			200	mA	
I_{LI}	Input Leakage Current			10	μA	V_{IN} = 0 to V_{CC}
I_{LOH}	Tri-State Output Leakage Current in Float			10	μA	V_{OUT} = 2.4 to V_{CC}
I_{LOL}	Tri-State Output Leakage Current in Float			-10	μA	V_{OUT} = 0.4V
I_{LD}	Data Bus Leakage Current In Input Mode			±10	μA	$0 < V_{IN} < V_{CC}$

CAPACITANCE

T_A = 25°C, f = 1 MHz

SYM	PARAMETER	MAX	UNIT	TEST CONDITIONS
CΦ	Clock Capacitance	35	pF	Unmeasured Pins
C_{IN}	Input Capacitance	5	pF	Returned to Ground
C_{OUT}	Output Capacitance	10	pF	

AC CHARACTERISTICS

MKB3880(P)-80 Z80-CPU

(T_A = −55°C to 125°C, V_{CC} = +5V, ±5%, Unless Otherwise Noted)

SIGNAL	SYM	PARAMETER	MIN	MAX	UNIT	TEST CONDITION
Φ	t_c	Clock Period	.4	[12]	μsec	
	$t_w(\Phi H)$	Clock Pulse Width, Clock High	180	(D)	nsec	
	$t_w(\Phi L)$	Clock Pulse Width, Clock Low	180	2000	nsec	
	$t_{r,}f$	Clock Rise and Fall Time		30	nsec	
$A_{0\text{-}15}$	$t_{D(AD)}$	Address Output Delay		145	nsec	C_L = 50pF
	$t_{F(AD)}$	Delay to Float		110	nsec	
	t_{acm}	Address Stable Prior to $\overline{\text{MREQ}}$ (Memory Cycle)	[1]		nsec	
	t_{aci}	Address Stable Prior to $\overline{\text{IORQ}}$, $\overline{\text{RD}}$ or $\overline{\text{WR}}$ (I/O Cycle)	[2]		nsec	
	t_{ca}	Address Stable From $\overline{\text{RD}}$, $\overline{\text{WR}}$, $\overline{\text{IORQ}}$ or $\overline{\text{MREQ}}$	[3]		nsec	Except T3-M1
	t_{caf}	Address Stable From $\overline{\text{RD}}$ or $\overline{\text{WR}}$ During Float	[4]		nsec	
$D_{0\text{-}7}$	$t_{D(D)}$	Data Output Delay		250	nsec	C_L = 50pF
	$t_{F(D)}$	Delay to Float During Write Cycle		90	nsec	
	$t_{S\Phi(D)}$	Data Setup Time to Rising Edge of Clock During M1 Cycle	50		nsec	
	$t_{S\overline{\Phi}(D)}$	Data Setup Time to Falling Edge at Clock During M2 to M5	60		nsec	
	t_{dcm}	Data Stable Prior to $\overline{\text{WR}}$ (Memory Cycle)	[5]		nsec	
	t_{dci}	Data Stable Prior to $\overline{\text{WR}}$ (I/O Cycle)	[6]		nsec	
	t_{cdf}	Data Stable From $\overline{\text{WR}}$	[7]		nsec	
	t_H	Input Hold Time	0		nsec	
$\overline{\text{MREQ}}$	$t_{DL\overline{\Phi}(MR)}$	$\overline{\text{MREQ}}$ Delay From Falling Edge of Clock, $\overline{\text{MREQ}}$ Low		100	nsec	C_L = 50pF
	$t_{DH\Phi(MR)}$	$\overline{\text{MREQ}}$ Delay From Rising Edge of Clock, $\overline{\text{MREQ}}$ High		100	nsec	
	$t_{DH\overline{\Phi}(MR)}$	$\overline{\text{MREQ}}$ Delay From Falling Edge of Clock, $\overline{\text{MREQ}}$ High		100	nsec	
	$t_{w(\overline{MRL})}$	Pulse Width, $\overline{\text{MREQ}}$ Low	[8]		nsec	
	$t_{w(\overline{MRH})}$	Pulse Width, $\overline{\text{MREQ}}$ High	[9]		nsec	
$\overline{\text{IORQ}}$	$t_{DL\Phi(IR)}$	$\overline{\text{IORQ}}$ Delay From Rising Edge of Clock, $\overline{\text{IORQ}}$ Low		90	nsec	C_L = 50pF
	$t_{DL\overline{\Phi}(IR)}$	$\overline{\text{IORQ}}$ Delay From Falling Edge of Clock, $\overline{\text{IORQ}}$ Low		110	nsec	
	$t_{DH\Phi(IR)}$	$\overline{\text{IORQ}}$ Delay From Rising Edge of Clock, $\overline{\text{IORQ}}$ High		100	nsec	
	$t_{DH\overline{\Phi}(IR)}$	$\overline{\text{IORQ}}$ Delay From Falling Edge of Clock, $\overline{\text{IORQ}}$ High		110	nsec	
$\overline{\text{RD}}$	$t_{DL\Phi(RD)}$	$\overline{\text{RD}}$ Delay From Rising Edge of Clock, $\overline{\text{RD}}$ Low		100	nsec	C_L = 50pF
	$t_{DL\overline{\Phi}(RD)}$	$\overline{\text{RD}}$ Delay From Falling Edge of Clock, $\overline{\text{RD}}$ Low		130	nsec	
	$t_{DH\Phi(RD)}$	$\overline{\text{RD}}$ Delay From Rising Edge of Clock, $\overline{\text{RD}}$ High		100	nsec	
	$t_{DH\overline{\Phi}(RD)}$	$\overline{\text{RD}}$ Delay From Falling Edge of Clock, $\overline{\text{RD}}$ High		110	nsec	

AC CHARACTERISTICS (Cont.)

SIGNAL	SYM	PARAMETER	MIN	MAX	UNIT	TEST CONDITION
$\overline{\text{WR}}$	$t_{DL\Phi(WR)}$	$\overline{\text{WR}}$ Delay From Rising Edge of Clock, $\overline{\text{WR}}$ Low		80	nsec	C_L = 50pF
	$t_{DL\bar{\Phi}(WR)}$	$\overline{\text{WR}}$ Delay From Falling Edge of Clock, $\overline{\text{WR}}$ Low		90	nsec	
	$t_{DH\Phi(WR)}$	$\overline{\text{WR}}$ Delay From Falling Edge of Clock, $\overline{\text{WR}}$ High		100	nsec	
	$t_{w(\overline{WRL})}$	Pulse Width, $\overline{\text{WR}}$ Low	[10]		nsec	
$\overline{\text{M1}}$	$t_{DL(M1)}$	$\overline{\text{M1}}$ Delay From Rising Edge of Clock $\overline{\text{M1}}$ Low		130	nsec	C_L = 50pF
	$t_{DH(M1)}$	$\overline{\text{M1}}$ Delay From Rising Edge of Clock, $\overline{\text{M1}}$ High		130	nsec	
$\overline{\text{RFSH}}$	$t_{DL(RF)}$	$\overline{\text{RFSH}}$ Delay From Rising Edge of Clock, $\overline{\text{RFSH}}$ Low		180	nsec	C_L = 30pF
	$t_{DH(RF)}$	$\overline{\text{RFSH}}$ Delay From Rising Edge of Clock $\overline{\text{RFSH}}$ High		150	nsec	
$\overline{\text{WAIT}}$	$t_{S(WT)}$	$\overline{\text{WAIT}}$ Setup Time to Falling Edge of Clock	70		nsec	
$\overline{\text{HALT}}$	$t_{D(HT)}$	$\overline{\text{HALT}}$ Delay Time From Falling Edge of Clock		300	nsec	C_L = 50pF
$\overline{\text{INT}}$	$t_{s(IT)}$	$\overline{\text{INT}}$ Setup Time to Rising Edge of Clock	80		nsec	
$\overline{\text{NMI}}$	$t_{w(\overline{NML})}$	Pulse Width, $\overline{\text{NMI}}$ Low	80		nsec	
$\overline{\text{BUSRQ}}$	$t_{s(BQ)}$	$\overline{\text{BUSRQ}}$ Setup Time to Rising Edge of Clock	80		nsec	
$\overline{\text{BUSAK}}$	$t_{DL(BA)}$	$\overline{\text{BUSAK}}$ Delay From Rising Edge of Clock, $\overline{\text{BUSAK}}$ Low		120	nsec	C_L = 50pF
	$t_{DH(BA)}$	$\overline{\text{BUSAK}}$ Delay From Falling Edge of Clock, $\overline{\text{BUSAK}}$ High		110	nsec	
$\overline{\text{RESET}}$	$t_{s(RS)}$	$\overline{\text{RESET}}$ Setup Time to Rising Edge of Clock	90		nsec	
	$t_{F(C)}$	Delay to/from Float ($\overline{\text{MREQ}}$, $\overline{\text{IORQ}}$, $\overline{\text{RD}}$ and $\overline{\text{WRI}}$)		100	nsec	
	t_{mr}	$\overline{\text{M1}}$ Stable Prior to IORQ (Interrupt Ack.)	[11]		nsec	

NOTES

1. Data should be enabled onto the CPU data bus when RD is active. During interrupt acknowledge data should be enabled when $\overline{\text{M1}}$ and $\overline{\text{IORQ}}$ are both active.
2. The $\overline{\text{RESET}}$ signal must be active for a minimum of 3 clock cycles.
3. Output Delay vs. Load Capacitance
   T_A = 125°C V_{CC} = 5 V ± 5%
   Add 10 nsec delay for each 50pF increase in load up to a maximum of 200pF for the data bus and 100pF for address and control lines.
4. Although static by design, testing guarantees t_W (ΦH) of 200 μsec maximum.

[1] $t_{acm} = t_W(\Phi H) + t_f - 75$

[2] $t_{aci} = t_c - 80$

[3] $t_{ca} = t_W(\Phi L) + t_r - 40$

[4] $t_{caf} = t_W(\Phi L) + t_r - 60$

[5] $t_{dcm} = t_c - 210$

[6] $t_{dci} = t_W(\Phi L) + t_r - 210$

[7] $t_{cdf} = t_W(\Phi L) + t_r - 80$

[8] $t_W(\overline{MRL}) = t_c - 40$

[9] $t_W(\overline{MRH}) = t_W(\Phi H) + t_f - 70$

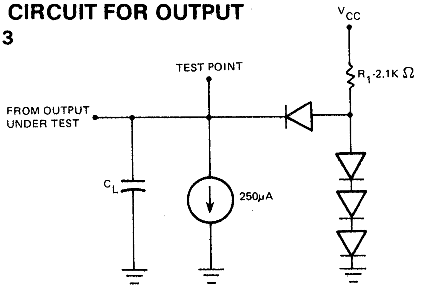
[10] $t_W(\overline{WR}) = t_c - 40$

[11] $t_{mr} = 2t_c + t_W(\Phi H) + t_f - 80$

[12] $t_c = t_W(\Phi H) + t_W(\Phi L) + t_r + t_f$

LOAD CIRCUIT FOR OUTPUT

Figure 3



AC CHARACTERISTICS
MKB3880(P)-84 Z80A-CPU

(T_A = −55°C to 125°C, V_{CC} = +5V, ±5%, Unless Otherwise Noted)

SIGNAL	SYM	PARAMETER	MIN	MAX	UNIT	TEST CONDITION
Φ	t_c	Clock Period	.25	[12]	μsec	
	$t_w(\Phi H)$	Clock Pulse Width, Clock High	110	(D)	nsec	
	$t_w(\Phi L)$	Clock Pulse Width, Clock Low	110	2000	nsec	
	$t_{r,}f$	Clock Rise and Fall Time		30	nsec	
$A_{0\text{-}15}$	$t_{D(AD)}$	Address Output Delay		110	nsec	
	$t_{F(AD)}$	Delay to Float		90	nsec	
	t_{acm}	Address Stable Prior to $\overline{\mathrm{MREQ}}$ (Memory Cycle)	[1]		nsec	C_L = 50pF
	t_{aci}	Address Stable Prior to $\overline{\mathrm{IORQ}}$, $\overline{\mathrm{RD}}$ or $\overline{\mathrm{WR}}$ (I/O Cycle)	[2]		nsec	
	t_{ca}	Address Stable From $\overline{\mathrm{RD}}$, $\overline{\mathrm{WR}}$, $\overline{\mathrm{IORQ}}$ or $\overline{\mathrm{MREQ}}$	[3]		nsec	Except T3-M1
	t_{caf}	Address Stable From $\overline{\mathrm{RD}}$ or $\overline{\mathrm{WR}}$ During Float	[4]		nsec	
$D_{0\text{-}7}$	$t_{D(D)}$	Data Output Delay		170	nsec	
	$t_{F(D)}$	Delay to Float During Write Cycle		90	nsec	
	$t_{S\Phi(D)}$	Data Setup Time to Rising Edge of Clock During M1 Cycle	50		nsec	
	$t_{S\overline{\Phi}(D)}$	Data Setup Time to Falling Edge at Clock During M2 to M5	60		nsec	C_L = 50pF
	t_{dcm}	Data Stable Prior to $\overline{\mathrm{WR}}$ (Memory Cycle)	[5]		nsec	
	t_{dci}	Data Stable Prior to $\overline{\mathrm{WR}}$ (I/O Cycle)	[6]		nsec	
	t_{cdf}	Data Stable From $\overline{\mathrm{WR}}$	[7]		nsec	
	t_H	Input Hold Time	0		nsec	
$\overline{\mathrm{MREQ}}$	$t_{DL\overline{\Phi}(MR)}$	$\overline{\mathrm{MREQ}}$ Delay From Falling Edge of Clock, $\overline{\mathrm{MREQ}}$ Low	20	85	nsec	
	$t_{DH\Phi(MR)}$	$\overline{\mathrm{MREQ}}$ Delay From Rising Edge of Clock, $\overline{\mathrm{MREQ}}$ High		85	nsec	
	$t_{DH\overline{\Phi}(MR)}$	$\overline{\mathrm{MREQ}}$ Delay From Falling Edge of Clock, $\overline{\mathrm{MREQ}}$ High		85	nsec	C_L = 50pF
	$t_{w(\overline{MRL})}$	Pulse Width, $\overline{\mathrm{MREQ}}$ Low	[8]		nsec	
	$t_{w(\overline{MRH})}$	Pulse Width, $\overline{\mathrm{MREQ}}$ High	[9]		nsec	
$\overline{\mathrm{IORQ}}$	$t_{DL\Phi(IR)}$	$\overline{\mathrm{IORQ}}$ Delay From Rising Edge of Clock, $\overline{\mathrm{IORQ}}$ Low		75	nsec	
	$t_{DL\overline{\Phi}(IR)}$	$\overline{\mathrm{IORQ}}$ Delay From Falling Edge of Clock, $\overline{\mathrm{IORQ}}$ Low		85	nsec	C_L = 50pF
	$t_{DH\overline{\Phi}(IR)}$	$\overline{\mathrm{IORQ}}$ Delay From Rising Edge of Clock, $\overline{\mathrm{IORQ}}$ High		85	nsec	
	$t_{DH\Phi(IR)}$	$\overline{\mathrm{IORQ}}$ Delay From Falling Edge of Clock, $\overline{\mathrm{IORQ}}$ High		85	nsec	
$\overline{\mathrm{RD}}$	$t_{DL\Phi(RD)}$	$\overline{\mathrm{RD}}$ Delay From Rising Edge of Clock, $\overline{\mathrm{RD}}$ Low		85	nsec	
	$t_{DL\overline{\Phi}(RD)}$	$\overline{\mathrm{RD}}$ Delay From Falling Edge of Clock, $\overline{\mathrm{RD}}$ Low		95	nsec	C_L = 50pF
	$t_{DH\Phi(RD)}$	$\overline{\mathrm{RD}}$ Delay From Rising Edge of Clock, $\overline{\mathrm{RD}}$ High		85	nsec	
	$t_{DH\overline{\Phi}(RD)}$	$\overline{\mathrm{RD}}$ Delay From Falling Edge of Clock, $\overline{\mathrm{RD}}$ High		85	nsec	

AC CHARACTERISTICS (Cont.)

SIGNAL	SYM	PARAMETER	MIN	MAX	UNIT	TEST CONDITION
$\overline{\text{WR}}$	$t_{DL\Phi(WR)}$	$\overline{\text{WR}}$ Delay From Rising Edge of Clock, $\overline{\text{WR}}$ Low		65	nsec	C_L = 50pF
	$t_{DL\bar{\Phi}(WR)}$	$\overline{\text{WR}}$ Delay From Falling Edge of Clock, $\overline{\text{WR}}$ Low		80	nsec	
	$t_{DH\Phi(WR)}$	$\overline{\text{WR}}$ Delay From Falling Edge of Clock, $\overline{\text{WR}}$ High		80	nsec	
	$t_{w(\overline{WRL})}$	Pulse Width, $\overline{\text{WR}}$ Low	[10]		nsec	
$\overline{\text{M1}}$	$t_{DL(M1)}$	$\overline{\text{M1}}$ Delay From Rising Edge of Clock $\overline{\text{M1}}$ Low		100	nsec	C_L = 50pF
	$t_{DH(M1)}$	$\overline{\text{M1}}$ Delay From Rising Edge of Clock, $\overline{\text{M1}}$ High		100	nsec	
$\overline{\text{RFSH}}$	$t_{DL(RF)}$	$\overline{\text{RFSH}}$ Delay From Rising Edge of Clock, $\overline{\text{RFSH}}$ Low		130	nsec	C_L = 50pF
	$t_{DH(RF)}$	$\overline{\text{RFSH}}$ Delay From Rising Edge of Clock $\overline{\text{RFSH}}$ High		120	nsec	
$\overline{\text{WAIT}}$	$t_{S(WT)}$	$\overline{\text{WAIT}}$ Setup Time to Falling Edge of Clock	70		nsec	
$\overline{\text{HALT}}$	$t_{D(HT)}$	$\overline{\text{HALT}}$ Delay Time From Falling Edge of Clock		300	nsec	C_L = 50pF
$\overline{\text{INT}}$	$t_{s(IT)}$	$\overline{\text{INT}}$ Setup Time to Rising Edge of Clock	80		nsec	
$\overline{\text{NMI}}$	$t_{w(\overline{NML})}$	Pulse Width, $\overline{\text{NMI}}$ Low	80		nsec	
$\overline{\text{BUSRQ}}$	$t_{s(BQ)}$	$\overline{\text{BUSRQ}}$ Setup Time to Rising Edge of Clock	50		nsec	
$\overline{\text{BUSAK}}$	$t_{DL(BA)}$	$\overline{\text{BUSAK}}$ Delay From Rising Edge of Clock, $\overline{\text{BUSAK}}$ Low		100	nsec	C_L = 50pF
	$t_{DH(BA)}$	$\overline{\text{BUSAK}}$ Delay From Falling Edge of Clock, $\overline{\text{BUSAK}}$ High		100	nsec	
$\overline{\text{RESET}}$	$t_{s(RS)}$	$\overline{\text{RESET}}$ Setup Time to Rising Edge of Clock	60		nsec	
	$t_{F(C)}$	Delay to/From Float ($\overline{\text{MREQ}}$, $\overline{\text{IORQ}}$, $\overline{\text{RD}}$ and $\overline{\text{WR}}$)		80	nsec	
	t_{mr}	$\overline{\text{M1}}$ Stable Prior to $\overline{\text{IORQ}}$ (Interrupt Ack.)	[11]		nsec	

NOTES

1. Data should be enabled onto the CPU data bus when $\overline{\text{RD}}$ is active. During interrupt acknowledge data should be enabled when M1 and IORQ are both active.
2. The $\overline{\text{RESET}}$ signal must be active for a minimum of 3 clock cycles.
3. Output Delay vs. Load Capacitance
   T_A = 125°C V_{CC} = 5 V ± 5%
   Add 10 nsec delay for each 50pF increase in load up to a maximum of 200pF for the data bus and 100pF for address and control lines.
4. Although static by design, testing guarantees t_W (ΦH) of 200 μsec maximum.

[1] $t_{acm} = t_w (\Phi H) + t_f$ -65

[2] $t_{aci} = t_c$ -70

[3] $t_{ca} = t_w (\Phi L) + t_r$ -50

[4] $t_{caf} = t_w (\Phi L) + t_r$ -45

[5] $t_{dcm} = t_c$ -170

[6] $t_{dci} = t_w (\Phi L) + t_r$ -170

[7] $t_{cdf} = t_w (\Phi L) + t_r$ -70

[8] $t_w (\overline{MRL}) = t_c$ -30

[9] $t_w (\overline{MRH}) = t_w (\Phi H) + t_f$ -40

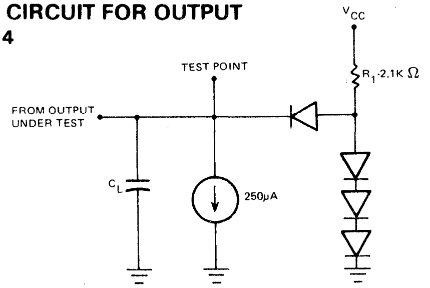
[10] $t_w (\overline{WR}) = t_c$ -30

[11] $t_{mr} = 2t_c + t_w (\Phi H) + t_f$ -65

[12] $t_c = 6_w (\Phi H) + t_w(\Phi L) + t_r + t_f$

LOAD CIRCUIT FOR OUTPUT

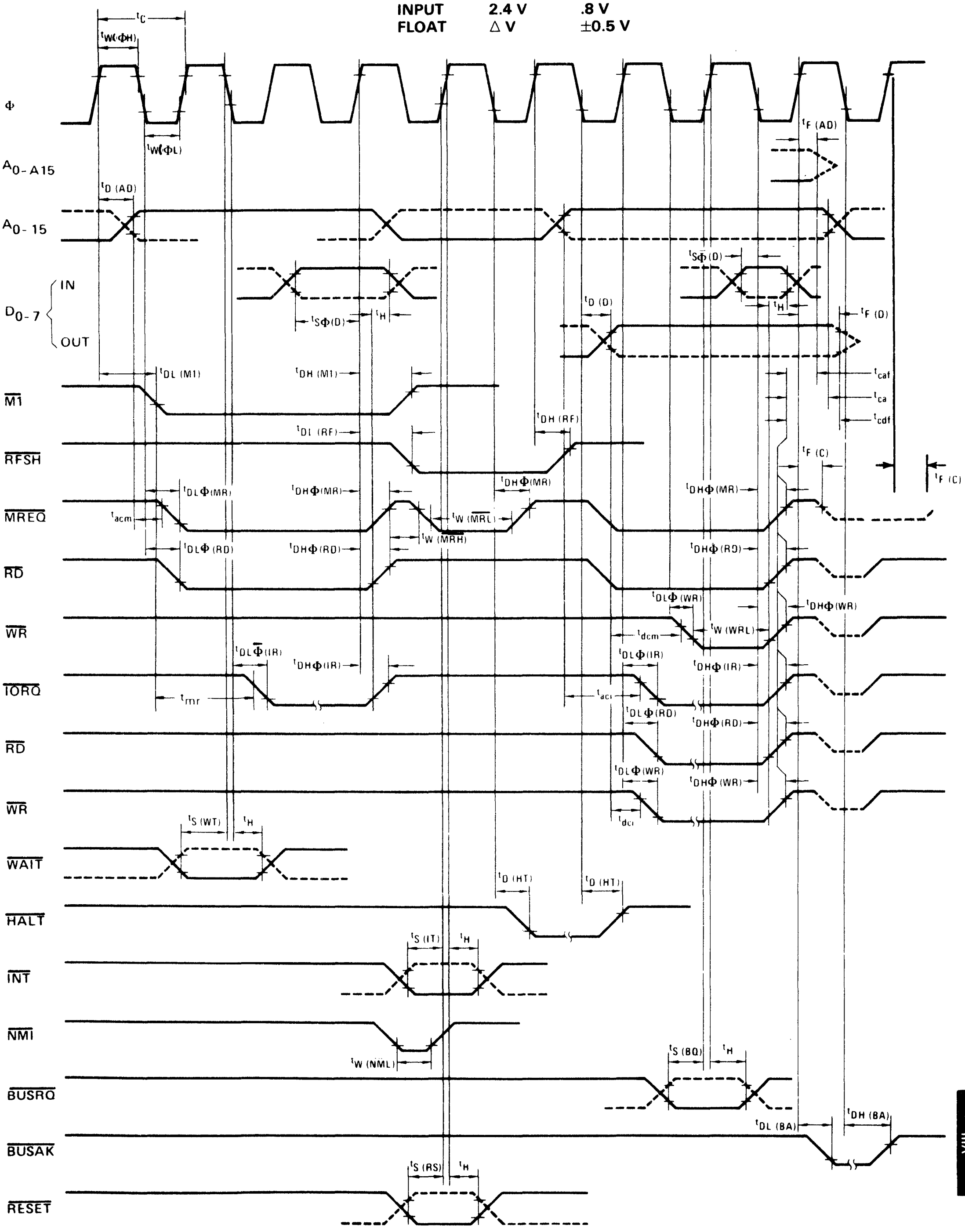
Figure 4



A.C. TIMING DIAGRAM

Timing measurements are made at the following voltages, unless otherwise specified.

	"1"	"0"
CLOCK	V_{CC}—.6	.8 V
OUTPUT	2.4 V	.8 V
INPUT	2.4 V	.8 V
FLOAT	Δ V	±0.5 V



MOSTEK®

1215 W. Crosby Rd. • Carrollton, Texas 75006 • 214/323-6000
In Europe, Contact: MOSTEK Brussels
270-272 Avenue de Tervuren (BTE21), B-1150 Brussels, Belgium;
Telephone: 762.18.80

PRINTED IN USA March 1981
Publication No MK79602