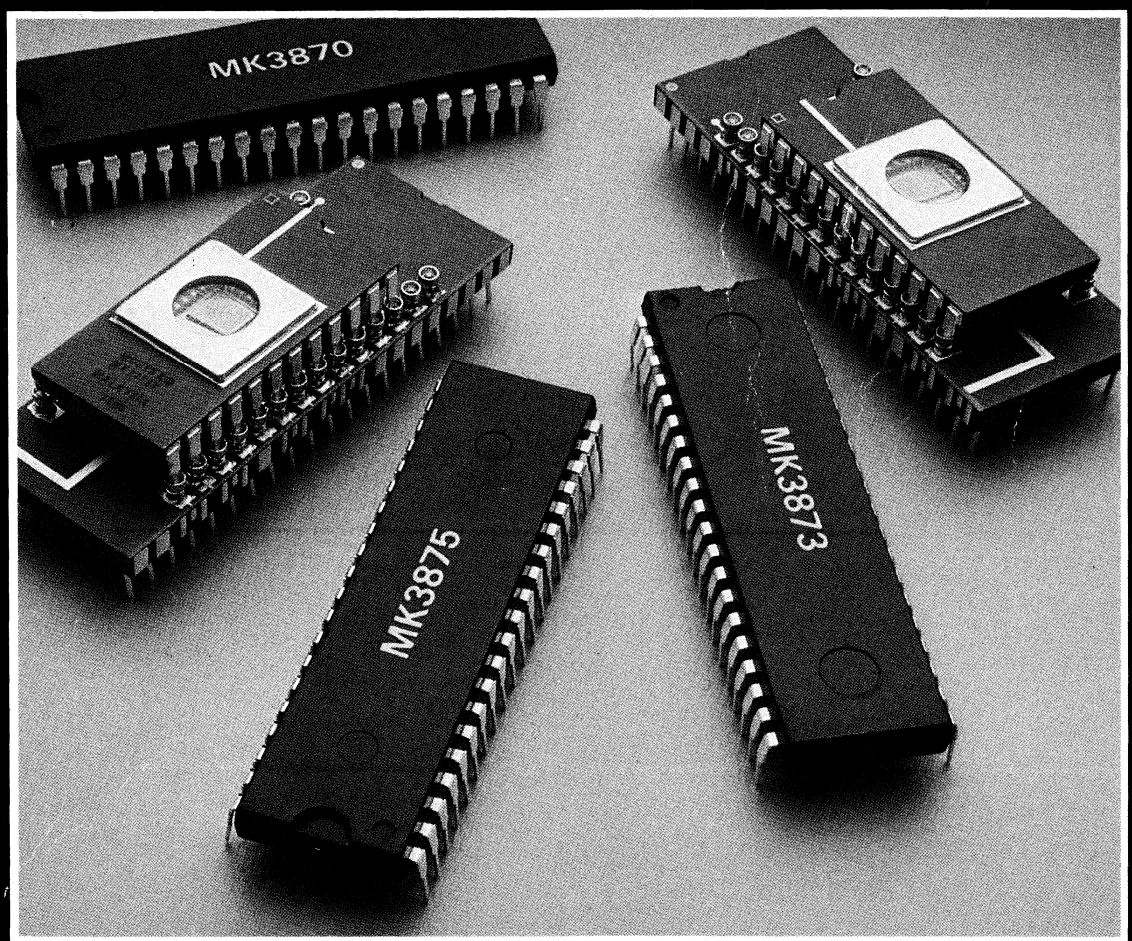
MOSTEK 1981

3870/F8 MICROCOMPUTER DATA BOOK



1981
3870/F8 MICROCOMPUTER
DATA BOOK

PRINTED IN USA February 1981
Publication Number MK79602

3870/F8 MICROCOMPUTER DATA BOOK

1981 MICROCOMPUTER DATA BOOK

TABLE OF CONTENTS

VI - 3870/F8 Microcomputer Application Notes

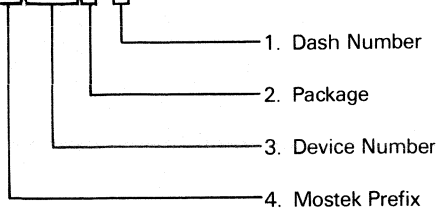
VII - Microcomputer Peripherals

3870/F8 MICROCOMPUTER DATA BOOK

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

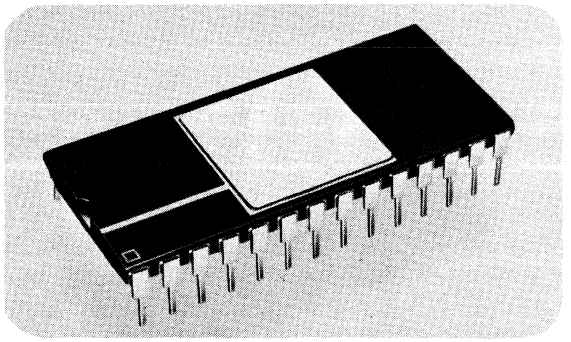
4. Mostek Prefix

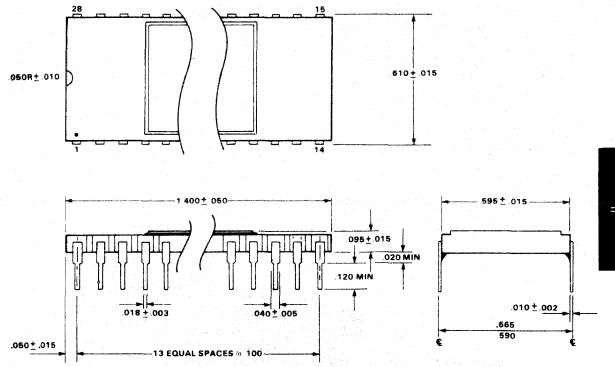
   MK-Standard Prefix

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

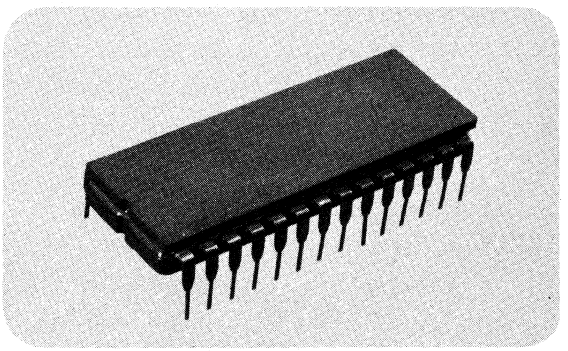
Package Descriptions

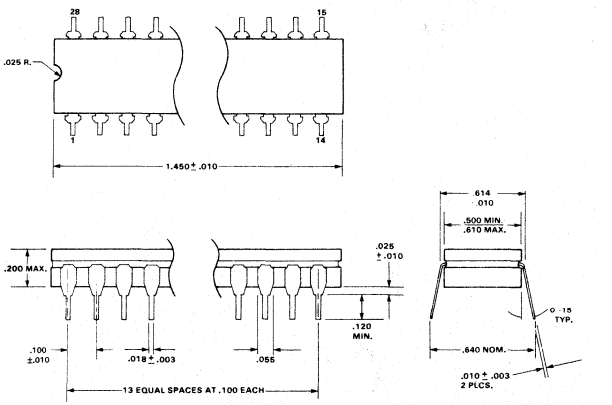
Ceramic Dual-In-Line Package (P)
28 Pin



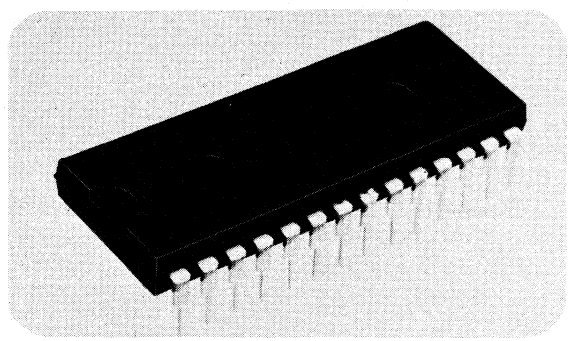


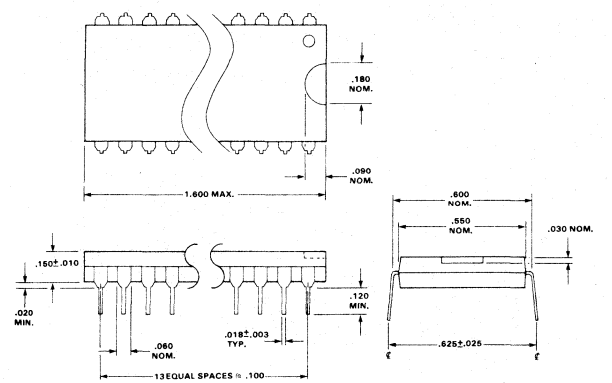
Cerdip Hermetic Packaging (J)
28 Pin



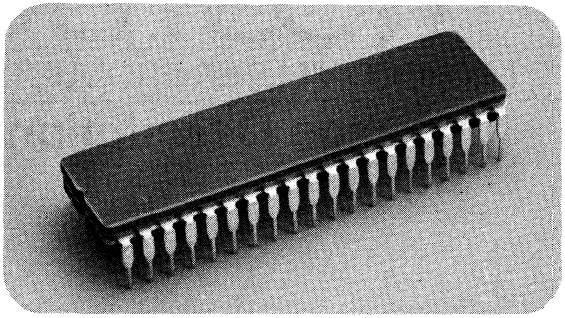


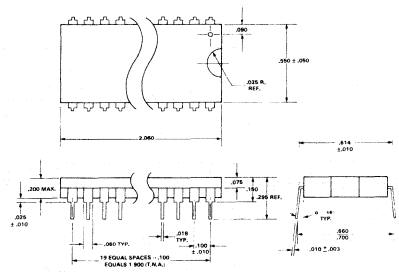
Plastic Dual-In-Line Package (N)
28 Pin



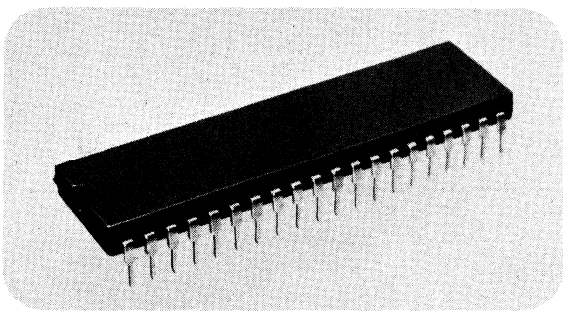


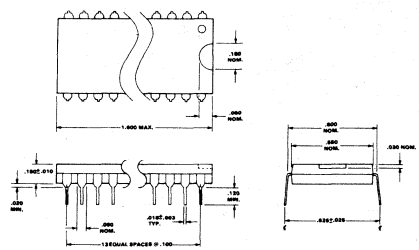
Cerdip Hermetic Packaging (J)
40 Pin





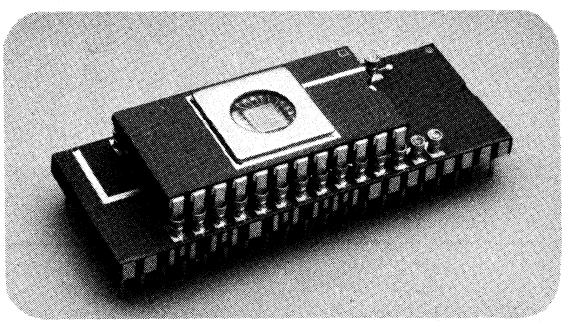
Plastic Dual-In-Line Packaging (N)
40 Pin

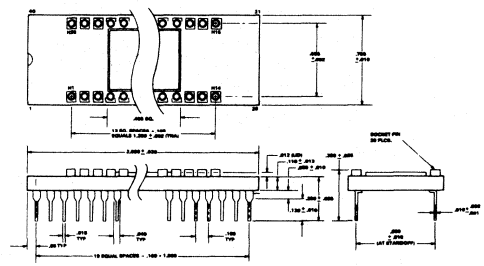




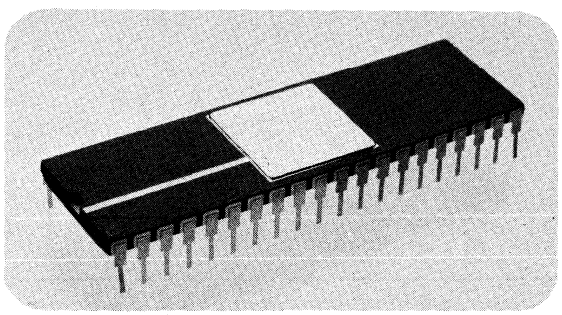
NOTE: Overall length includes .005 flash on either end of package.

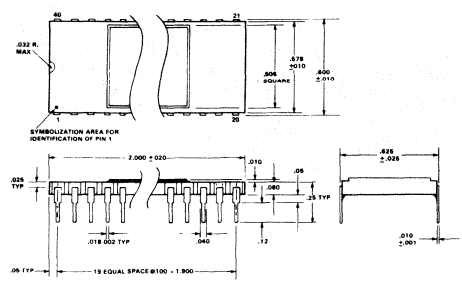
P-PROM Package (R)
40 Pin



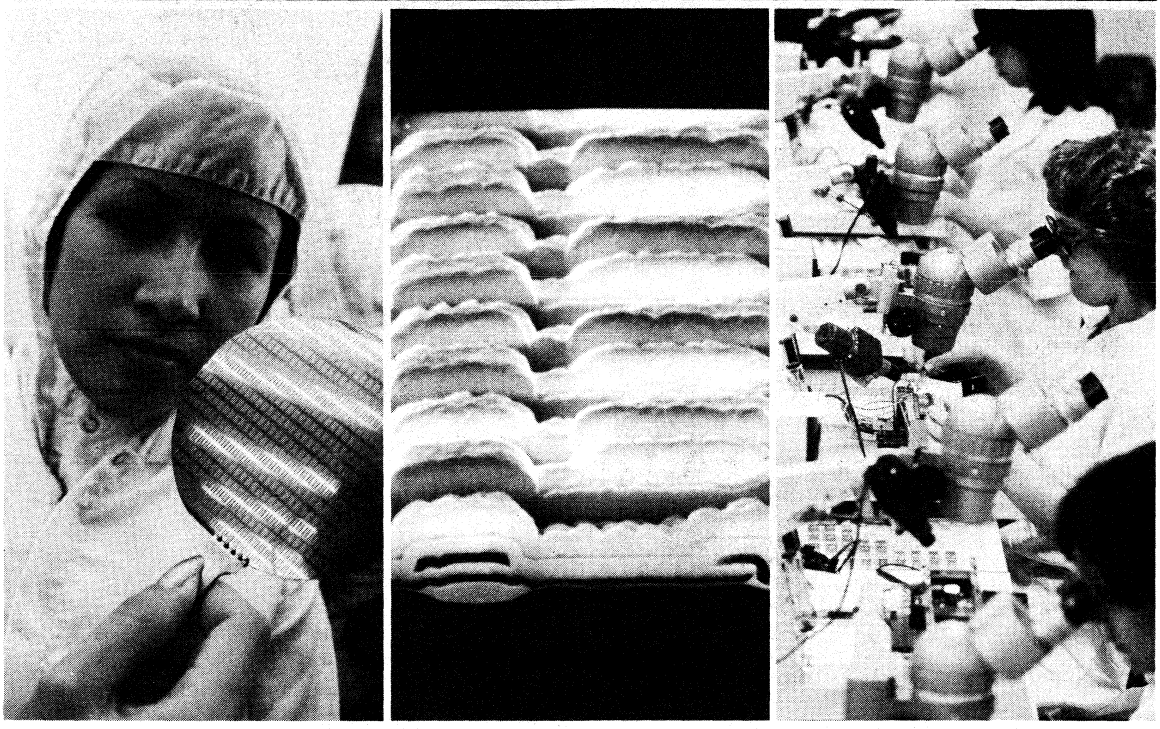


Ceramic Dual-In-Line Package (P)
40 Pin





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.

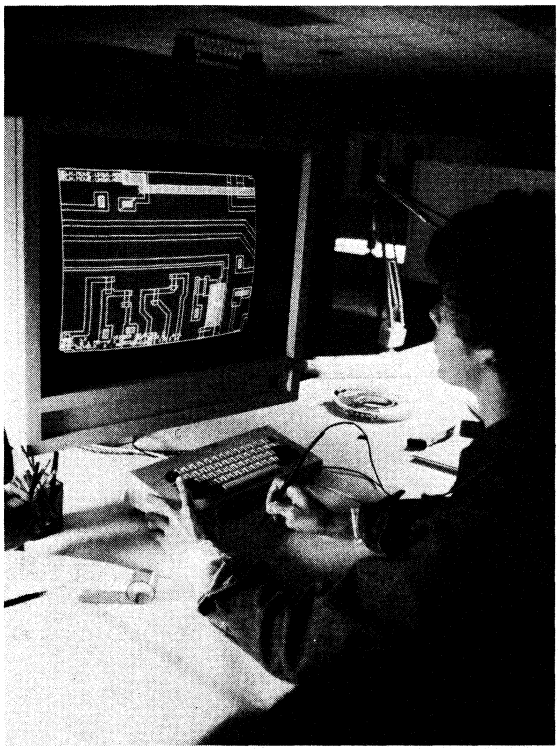
QUALITY

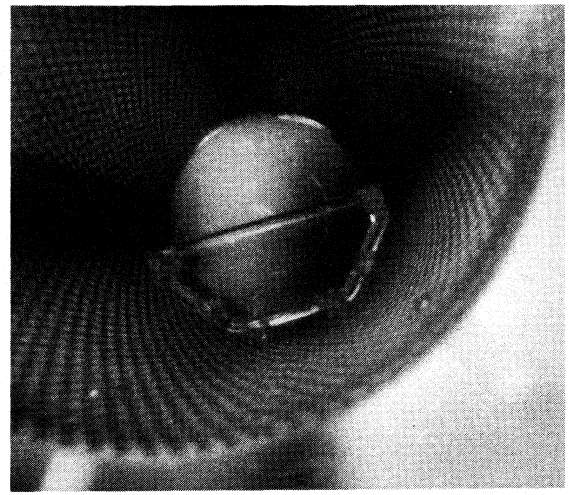
The worth of a product is measured by how well it is designed, manufactured and tested and by how well it works in your system.

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

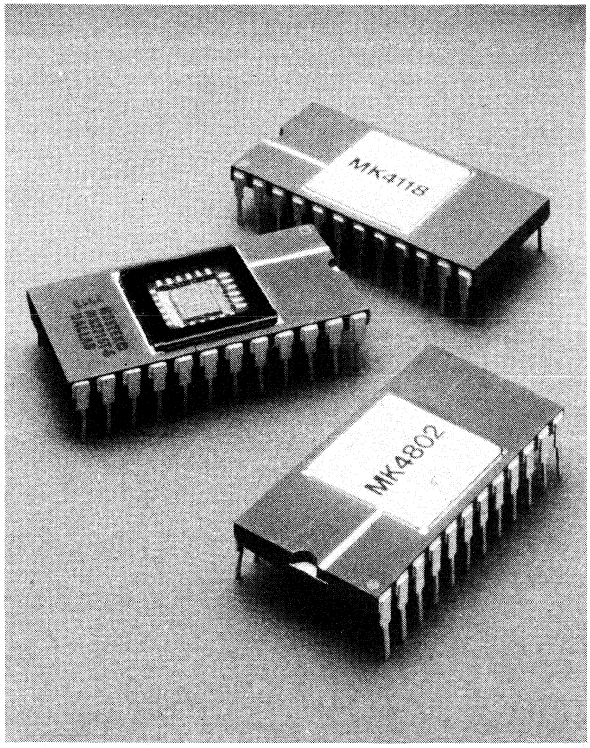
Through innovations in both circuit design, wafer processing and production, Mostek has become the industry's leading supplier of memory products.

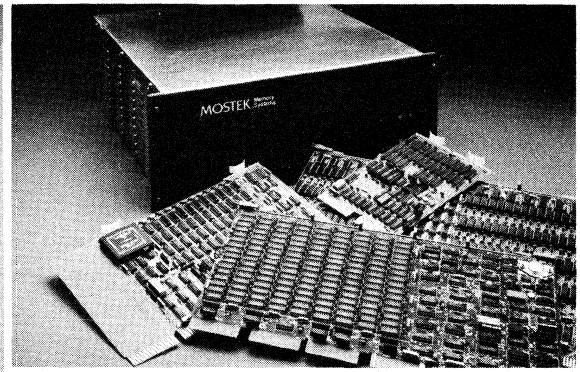
An example of Mostek leadership is our new BYTEWYDE™ family of static RAMs, ROMs, and EPROMs. All provide high performance, N words x 8-bit organization and common pin configurations to allow easy system upgrades in density and performance. Another important product area is fast static RAMs. With major 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.

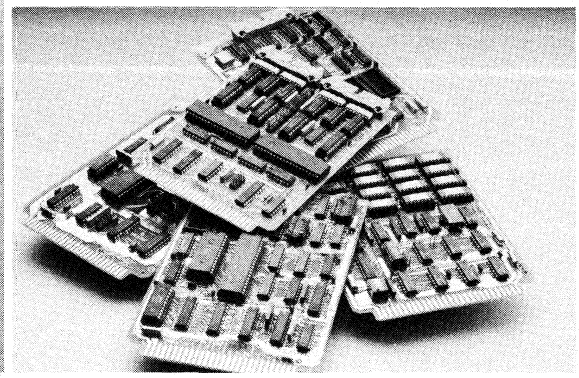
Microcomputer Components

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.

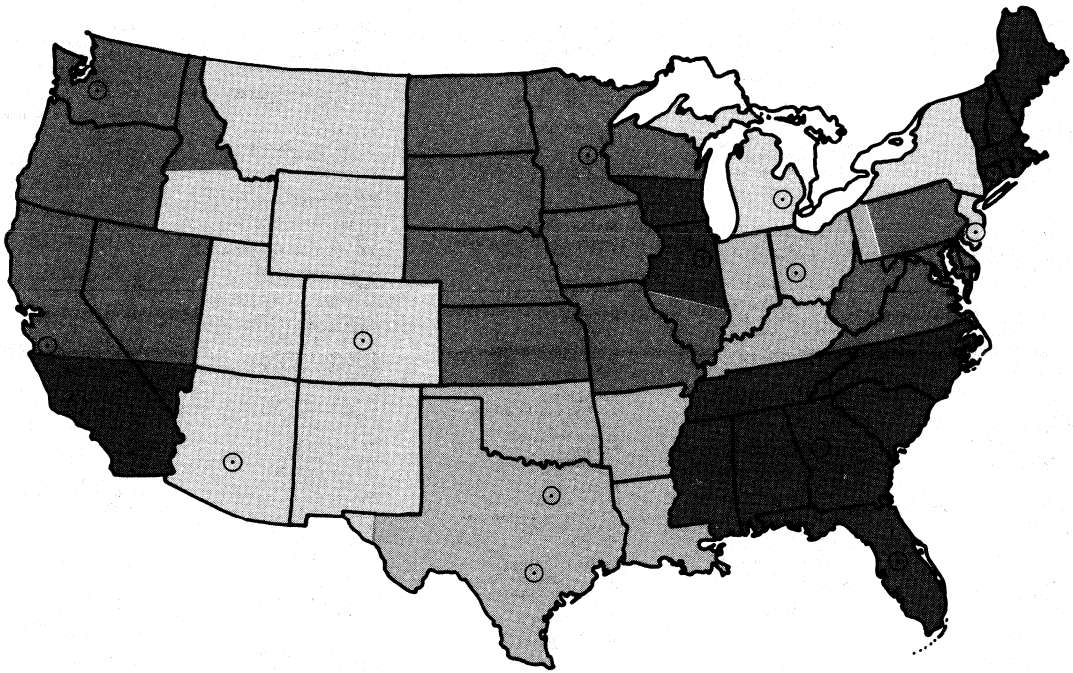
Mostek's MD Series™ features both stand-alone microcomputer boards and expandable microcomputer boards. The expandable boards are modularized by function, reducing system cost because the designer buys only the specific functional modules his system requires. All MDX boards are STD-Z80 BUS compatible.

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

3870/F8 MICROCOMPUTER DATA BOOK

MOSTEK®

MICROCOMPUTER COMPONENTS

Technical Manual

MK3870 FAMILY

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.0 INTRODUCTION

The MK3870 Family of Single Chip Microcomputers are complete, 8 bit microcomputers implemented on a single MOS integrated circuit. These microcomputers are ideal for use as logic replacement elements in a variety of control applications. Features which are common among devices in the MK3870 Family are listed listed below:

- ☐ Common instruction set consisting of over 70 instruction types.
- ☐ Versions with various combinations of ROM and executable RAM.
- ☐ Up to 32 bits (4 ports) TTL compatible I/O.
- ☐ Programmable binary timer
  - Interval Timer Mode
  - Pulse Width Measurement Mode
  - Event Counter Mode
- ☐ External interrupt input.
- ☐ Crystal, LC, RC, or External time base options available
- ☐ EPROM compatible versions available for development, prototyping, and low-volume production.
- ☐ Pinout compatibility (In Socket Expandibility)
- ☐ Low power dissipation.
- ☐ Single +5 volt power supply.

Members in the MK3870 Family require only a single +5V power supply and dissipate very little power. Utilizing ion-implanted, N-channel silicon gate technology and advanced circuit design techiques, MK3870 Family devices offer maximum cost effectiveness in a wide range of applications.

All MK3870 Family microcomputers execute a common set of more than 70 instructions. These devices are available in a wide range of memory sizes and types so that a designer can choose a device with the right combination of ROM and RAM to suit his system requirements. In addition, MK3870 Family devices are available with special types of I/O, such as the 3873 with an on chip, sixteen bit serial I/O port. All devices in the family are pin compatible, a feature which allows easy system upgrade by replacing a MK3870 device in an existing design with another in the family with greater amounts of ROM and/or RAM or special I/O functions.

This ease of system upgrade is a concept known as In Socket Expandibility. In Socket Expandibility provides the designer with a new concept in system expansion. With In Socket Expandability, microcomputer based systems can be enhanced or expanded in many different ways without affecting the printed circuit board, the enclosure, or power supply requirements for the system. The Mostek MK3870 Microcomputer Family implements the concept of In Socket Expandibility to provide low design costs. Manufacturers who have used the MK3870 in product designs can extend the products capability simply by removing the MK3870 microcomputer from its socket and replacing it with another member of the MK3870 Microcomputer Family.

Mostek supplies a complete line of development equipment and associated software support packages which can be used as tools for writing and debugging MK3870 programs. For the user who requires a sophisticated development system, the MATRIX (TM) dual floppy disk based development system is available which is based on the powerful Mostek Z80 chip set. A macro cross-assembler for the MK3870 instruction set, called MACRO-70, is available which runs under FLP-80DOS, the operating system for the MATRIX. MACRO-70 is the most powerful macro assembler on the microcomputer market, and features a number of macro definitions on diskette which can be used to extend the base instruction set of the 3870. Use of MACRO-70 can result in quicker generation of MK3870 programs. The AIM-73E Application

Interface Module is a systems product which is directly interfaced with the MATRIX and provides real time in-circuit-emulation for all devices in the MK3870 Family. The AIM-73E standard features include breakpoint, single step, and display and modification of the contents of any memory location, register, or I/O port. In addition, the AIM-73E has a 1048 x 48 history trace memory which can be used to capture up to 1048 cycles of program execution.

The Mostek MK3870 Family of single chip microcomputers is recognized as an industry standard in logic replacement. The MK3870 has been designed into and successively used in a wide range of applications which require some type of intelligent control. The MK3870 has made possible a whole new technology that can create cost effective system solutions to manufacturers of automobiles, major appliances, industrial controls, computer peripherals, and more. New and more powerful products have been added to the MK3870 Family, making even more applications practical and affordable.

2.0 PART IDENTIFICATION

2.1 USING THE TECHNICAL MANUAL

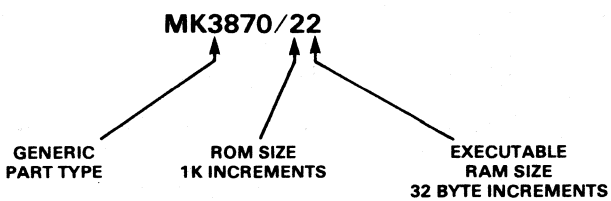
The MK3870 Family Technical Manual is provided as an overall user's guide to the operation and application of MK3870 Family products. It is intended to provide detailed technical information on those features which are common to all (or several) devices in the family. Additional information is provided which covers briefly features that apply to a specific group of devices in the 3870 Family. For example, an overview of the MK3873's serial I/O port logic is given in the technical manual, but the MK3873 data sheet should be consulted for detailed operation and programming of the serial I/O port. In some sections of the technical manual, certain electrical and timing specifications are referred to in the discussion of a subject such as the selection of a crystal. Again, the user should still consult the appropriate device data sheet for exact specifications.

2.2 PART NUMBERING SYSTEM

Since a number of new devices are continually being added to the 3870 Family, a new part numbering system has been implemented by Mostek. This part numbering system uses a generic number to designate a particular pin configuration and technology of a 3870 Family device, followed by a slash number which represents the amount and type of memory implemented on that device. An example of this part numbering system is illustrated in Figure 2-1.

3870 PART NUMBERING EXAMPLE

Figure 2-1



The two digit slash number specifies the size of ROM and RAM on the chip. The left digit represents the amount of ROM on chip in increments of 1K bytes. The right digit represents the amount of executable RAM on chip in increments of 32 bytes. Note that all 3870 devices have 64 bytes of scratchpad RAM which is not included in the slash number designation, since it is not addressed as main memory. Only the RAM memory which is addressed in the same memory space as ROM is indicated by the slash number.

At present, there are three generic types of 3870 Family devices which are available in various combinations of ROM and RAM as specified in the two-digit slash number suffix. These three types of devices are listed below:

MK3870/XX - 32 bits of bidirectional, parallel I/O
MK3873/XX - Serial I/O port microcomputer
MK3875/XX - Battery backup microcomputer

Mostek also offers a full line of EPROM compatible 3870's. A special packaging technology has been developed which allows a standard 24 pin or 28 pin EPROM memory to be plugged directly into the back of the 40 pin package. This special type of package has been termed the Piggyback PROM or P-PROM (TM) package. The Mostek part numbering system specifies a P-PROM device with the designation 38P7X where the "X" represents the generic part type being emulated. For example, the P-PROM device which can be used to emulate a 3870 mask ROM microcomputer would be a 38P70. All P-PROM devices have an identical pinout as the mask ROM device they emulate.

A product which has recently been announced by MOSTEK is a CMOS version of the 3870 which will be designated as the MK38C70. The first CMOS device which is intended to be made available is the P-PROM equivalent which is called the MK38CP70. The 38CP70 will allow exact emulation of the mask ROM device.

Since many users are probably familiar with the old part numbering system of 3870 Family single chip microcomputers, the following cross reference guide shown in Table 2-1 is provided as an aid in determining the new part number from the old designation. Also, the list shown in Table 2-2 is provided as a summary of devices which are now available or soon to be available in the MK3870 single chip microcomputer family.

3870 FAMILY PART NUMBER CROSS REFERENCE

Table 2-1

Old Part Number	New Part Number	ROM	Executable RAM	Parallel I/O	Special I/O	Technology
MK3870	MK3870/20	2K	0 bytes	32 bits	-	NMOS
MK3872	MK3870/42	4K	64 bytes	32 bits	-	NMOS
MK3872 w/standby	MK3875/42	4K	64 bytes	30 bits	V_{SB},V_{BB}	NMOS
MK3873	MK3873/20	2K	0 bytes	29 bits	SI,SO, SRCLK	NMOS
MK3874 MK97400, MK97401	MK38P70/02	Ext.	64 bytes	32 bits	-	NMOS P-PROM pkg.
MK3876	MK3870/22	2K	64 bytes	32 bits	-	NMOS
MK3876 w/standby	MK3875/22	2K	64 bytes	30 bits	V_{SB},V_{BB}	NMOS

3870 SINGLE CHIP MICROCOMPUTER FAMILY

Table 2-2

Device	ROM (bytes)	Scratchpad RAM	Executable RAM	Parallel I/O	Special I/O	Technology
MK3870/10	1K	64 bytes	0 bytes	32 bits	-	NMOS
MK3870/12	1K	64 bytes	64 bytes	32 bits	-	NMOS
MK3870/20	2K	64 bytes	0 bytes	32 bits	-	NMOS
MK3870/22	2K	64 bytes	64 bytes	32 bits	-	NMOS
MK3870/30	3K	64 bytes	0 bytes	32 bits	-	NMOS
MK3870/32	3K	64 bytes	64 bytes	32 bits	-	NMOS
MK3870/40	4K	64 bytes	0 bytes	32 bits	-	NMOS
MK3870/42	4K	64 bytes	64 bytes	32 bits	-	NMOS
MK3873/10	1K	64 bytes	0 bytes	29 bits	SI,SO SRCLK	NMOS
MK3873/12	1K	64 bytes	64 bytes	29 bits	SI,SO SRCLK	NMOS
MK3873/20	2K	64 bytes	0 bytes	29 bits	SI,SO SRCLK	NMOS
MK3873/22	2K	64 bytes	64 bytes	29 bits	SI,SO SRCLK	NMOS
MK3875/22	2K	64 bytes	64 bytes	30 bits	V_{SB},V_{BB}	NMOS
MK3875/42	4K	64 bytes	64 bytes	30 bits	V_{SB},V_{BB}	NMOS
MK38C70/10	1K	64 bytes	0 bytes	32 bits	-	CMOS
MK38C70/20	2K	64 bytes	0 bytes	32 bits	-	CMOS
MK38P70/02	Ext.	64 bytes	64 bytes	32 bits	-	NMOS P-PROM pkg.
MK38P73/02	Ext.	64 bytes	64 bytes	29 bits	SI,SO SRCLK	P-PROM pkg.
MK38CP70/02	Ext.	64 bytes	64 bytes	32 bits	-	CMOS P-PROM pkg.

3.0 FUNCTIONAL PIN DESCRIPTION

The chart shown in Figure 3-1 pictures the pin configuration for the three generic part types currently available in the 3870 Family. MK3870 Family microcomputers exhibit the feature of universal pin compatibility. All devices in the Family are complete, 8-bit microcomputer systems implemented on a single integrated circuit chip so that the majority of pins are dedicated to I/O. Some devices differ slightly in terms of pin functions in cases where special I/O functions have been implemented.

The pin designation shown in the chart for devices with the generic part number "3870" represent those which have a full 32 bits of bidirectional, parallel I/O which are addressed as four 8 bit ports. The "3873" number represents those devices which have 3 lines dedicated to a serial I/O function along with 29 bits of parallel I/O. "3875" devices have two pins which serve as the standby power option lines for saving the contents of executable RAM in a low power standby mode.

3870 FAMILY PIN COMPATIBILITY CHART

Figure 3-1

3873	3875	38C70	3870			3870	38C70	3875	3873
			X1	1	40	V_{CC}			
			X2	2	39	$\overline{RST}$			
	V_{BB}		P0-0	3	38	INT			
	V_{SB}		P0-1	4	37	P1-0			
			P0-2	5	36	P1-1			SRCLK
			P0-3	6	35	P1-2			SI
			$\overline{STB}$	7	34	P1-3			SO
			P4-0	8	33	P5-0			
			P4-1	9	32	P5-1			
			P4-2	10	31	P5-2			
			P4-3	11	30	P5-3			
			P4-4	12	29	P5-4			
			P4-5	13	28	P5-5			
			P4-6	14	27	P5-6			
			P4-7	15	26	P5-7			
			P0-7	16	25	P1-7			
			P0-6	17	24	P1-6			
			P0-5	18	23	P1-5			
			P0-4	19	22	P1-4			
			GND	20	21	TEST			

The following is a description of the function of each pin associated with the MK3870, the base 3870 Family part type:

PIN FUNCTION SUMMARY

Table 3-1

PIN NAME	DESCRIPTION	TYPE
$\overline{P0-0}$ - $\overline{P0-7}$	I/O Port 0	Bidirectional
$\overline{P1-0}$ - $\overline{P1-7}$	I/O Port 1	Bidirectional
$\overline{P4-0}$ - $\overline{P4-7}$	I/O Port 4	Bidirectional
$\overline{P5-0}$ - $\overline{P5-7}$	I/O Port 5	Bidirectional
$\overline{STROBE}$	Ready Strobe	Output
EXT INT	External Interrupt	Input
$\overline{RESET}$	External Reset	Input

PIN FUNCTION SUMMARY

Table 3-1 (Continued)

PIN NAME	DESCRIPTION	TYPE
TEST	Test Line	Input
XTL 1, XTL2	Time Base	Input
SI	Serial Input	Input
SO	Serial Output	Output
SRCLK	Serial Clock	Bidirectional
V_{SB}	Standby Power	Input
V_{BB}	Substrate Decoupling	Input
V_{CC}, GND	Power Supply Lines	Input

$\overline{P0\text{-}0}$—$\overline{P0\text{-}7}$, $\overline{P1\text{-}0}$—$\overline{P1\text{-}7}$, $\overline{P4\text{-}0}$—$\overline{P4\text{-}7}$, and $\overline{P5\text{-}0}$—$\overline{P5\text{-}7}$ are 32 lines which can be individually used as either TTL compatible inputs or as latch outputs.

$\overline{STROBE}$ is a ready strobe associated with I/O Port 4. This pin which is normally high provides a single low pulse after valid data is present on the $\overline{P4\text{-}0}$—$\overline{P4\text{-}7}$ pins during an output instruction.

$\overline{RESET}$ may be used to externally reset the 3870. When pulled low the 3870 will reset. When then allowed to go high the 3870 will begin program execution at program location H '000'.

EXT INT is the external interrupt input. Its active state is software programmable. This input is also used in conjunction with the timer for pulse width measurement and event counting.

XTL 1 and XTL 2 are the time base inputs to which a crystal, LC network, RC network, or an external single-phase clock may be connected.

TEST is an input, used only in testing the 3870. For normal circuit functionality this pin is left unconnected or may be grounded.

SI is a serial input line and exists only on MK3873/XX devices. It is the data input to the serial port of the MK3873.

SO is a serial output line and exists only on MK3873/XX devices. It is the data output of the serial I/O port of the MK3873.

SRCLK is the clock for the serial port and exists only on MK3873/XX devices. It can be programmed by software as an input or an output so that the serial port may be driven from the internal baud rate generator or from an external source.

V_{SB} is the auxilliary power supply input used only on MK3875/XX devices.

V_{BB} is the substrate decoupling pin used only on MK3875/XX devices. A capacitor is required to be tied from V_{BB} to ground.

V_{CC} is the power supply input.

4.0 MK3870 ARCHITECTURE

4.1 INTRODUCTION

All members of the MK3870 Single Chip Microcomputer Family share a common architecture. This section describes the basic functional elements of the 3870 Family architecture. Elements of the architecture which are common to all 3870 Family devices are discussed in sections 4.2-4.9. A block diagram which depicts the MK3870 is shown in Figure 4-1, along with a programming model which is pictured in Figure 4-3. Section 4.10 contains a brief description of the MK3873 serial I/O port, and Section 4.11 briefly describes the battery backup feature of the MK3875.

4.2 MAIN CONTROL LOGIC

The Instruction Register (IR) receives the operation code (OP code) of the instruction to be executed from the program ROM via the data bus. The OP code is loaded into the instruction register from the internal data bus at the end of the execution sequence for the previous instruction.

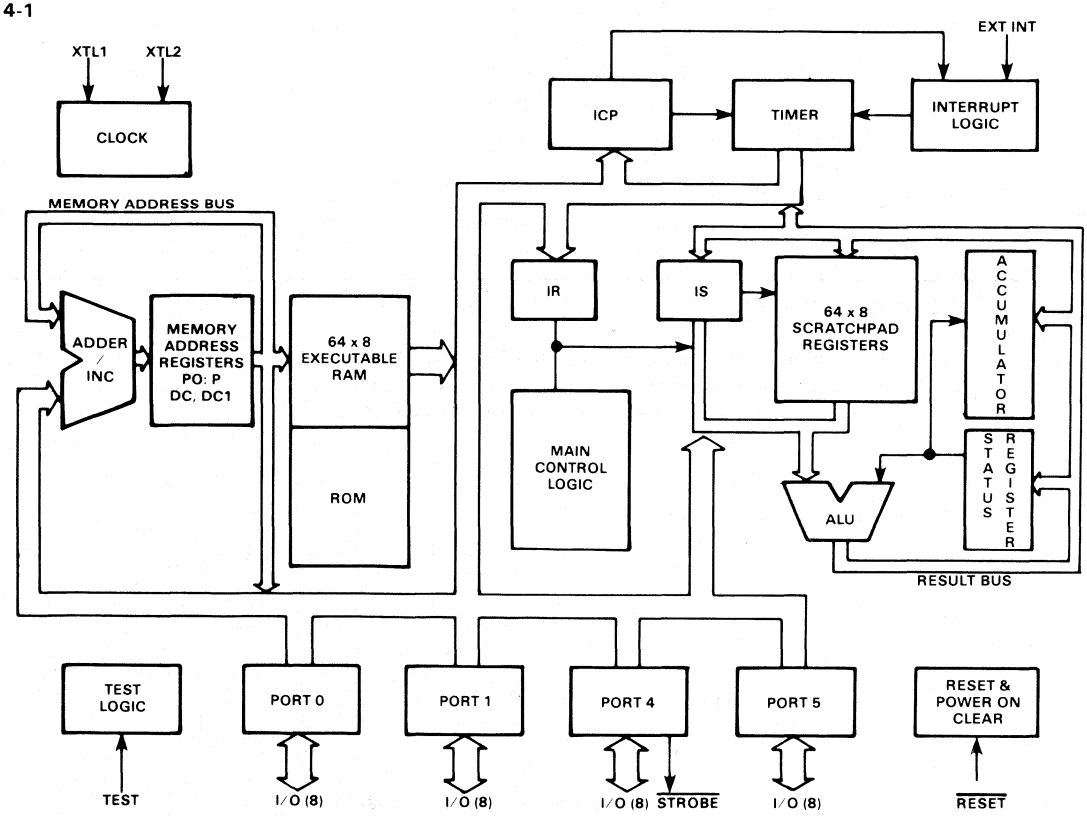
OP codes in the 3870 are either 4 or 8 bits long. In those instructions in which the OP code is 4 bits long, the lower 4 bits are either used as an immediate address to an I/O port or scratchpad location or as an immediate 4 bit operand. Once latched into the IR the main control logic decodes the instruction and provides the necessary gating signals to all circuit elements.

4.3 ARITHMETIC AND LOGIC UNIT (ALU)

After receiving commands from the main control logic, the ALU performs the required arithmetic or logic operations (using the data presented on the two input busses) and provides the result on the result bus. The arithmetic operations that can be performed in the ALU are binary add, decimal

MK3870 FAMILY BLOCK DIAGRAM

Figure 4-1



adjust, add with carry, decrement, and increment. The logic operations that can be performed are AND, OR, EXCLUSIVE OR, 1's complement, shift right, and shift left. Besides providing the result on the result bus, the ALU also provides four signals representing the status of the result. These signals, stored in the Status Register (W), represent CARRY, OVERFLOW, SIGN, and ZERO condition of the result of the operation.

4.4 ACCUMULATOR (A)

The Accumulator (A) is the principal register for data manipulation within the 3870. The A serves as one input to the ALU for arithmetic or logical operations. The result of ALU operations are stored in the Accumulator.

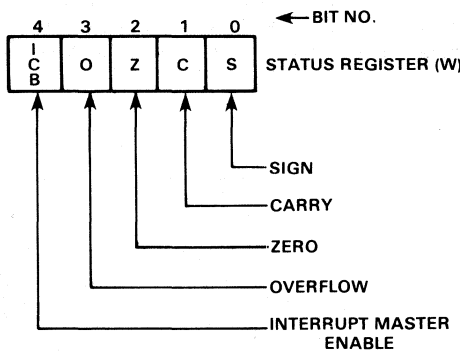
4.5 THE STATUS REGISTER (W)

The Status Register (also called the W register) holds five status flags as shown below in Figure 4-2.

4.5.1 SIGN (S BIT)

When the results of an ALU operation are being interpreted as a signed binary number, (2's complement representation) the high order bit (bit 7) represents the sign of the number. At the conclusion of instructions that may modify the Accumulator most significant bit, (bit 7) the S bit is set to the complement of the result of the operation.

STATUS REGISTER
Figure 4-2



4.5.2 CARRY (C BIT)

The C bit may be visualized as an extension of an 8-bit data unit, i.e., the ninth bit of a 9-bit data unit. When two bytes are added, and the sum is greater than 255, then the carry out of the high order bit appears in the C bit. Here are some examples:

	C	7	6	5	4	3	2	1	0	<- Bit Number
Accumulator contents :		0	1	1	0	0	1	0	1	
Value added :		0	1	1	1	0	1	1	0	
Sum :	0	1	1	0	1	1	0	1	1	

There is no carry, so C is reset to 0.

	C	7	6	5	4	3	2	1	0	<- Bit Number
Accumulator contents :		1	0	0	1	1	1	0	1	
Value added :		1	1	0	1	0	0	0	1	
Sum :	1	0	1	1	0	1	1	1	0	

There is a carry, so C is set to 1.

4.5.3 ZERO (Z BIT)

The Z bit is set whenever an arithmeic or logical operation generates a zero result. The Z bit is reset to 0 when an arithmetic of logical operation could have generated a zero result, but did not.

4.5.4 OVERFLOW (O BIT)

When the results of an ALU operation are being interpreted as a signed binary number (2's complement representation), some method must be provided for indicating carries out of the highest numeric bit (bit 6). This is done using the O bit. After arithmetic operations, the O bit is set to the Exclusive-OR of carries out of bits 6 and 7. The fact that this simplifies signed binary arithmetic is described in the MK3870 Family Programming Manual. Here are some examples:

	7	6	5	4	3	2	1	0	<- Bit Number
Accumulator contents :	1	0	1	1	0	0	1	1	
Value added :	0	1	1	1	0	0	0	1	
Sum :	0	0	1	0	0	1	0	0	
1									

There is a carry out of bit 6 and out of bit 7, so the O bit is reset to 0 (1 + 1 = 0). The C bit is set to 1.

	7	6	5	4	3	2	1	0	<- Bit Number
Accumulator contents :	0	1	1	0	0	1	1	1	
Value added :	0	0	1	0	0	1	0	0	
Sum :	1	0	0	0	1	0	1	1	

There is a carry out of bit 6, but no carry out of bit 7; the O bit is set to 1 (1 + 0 = 1). The C bit is reset to 0.

4.5.5 INTERRUPTS (ICB BIT)

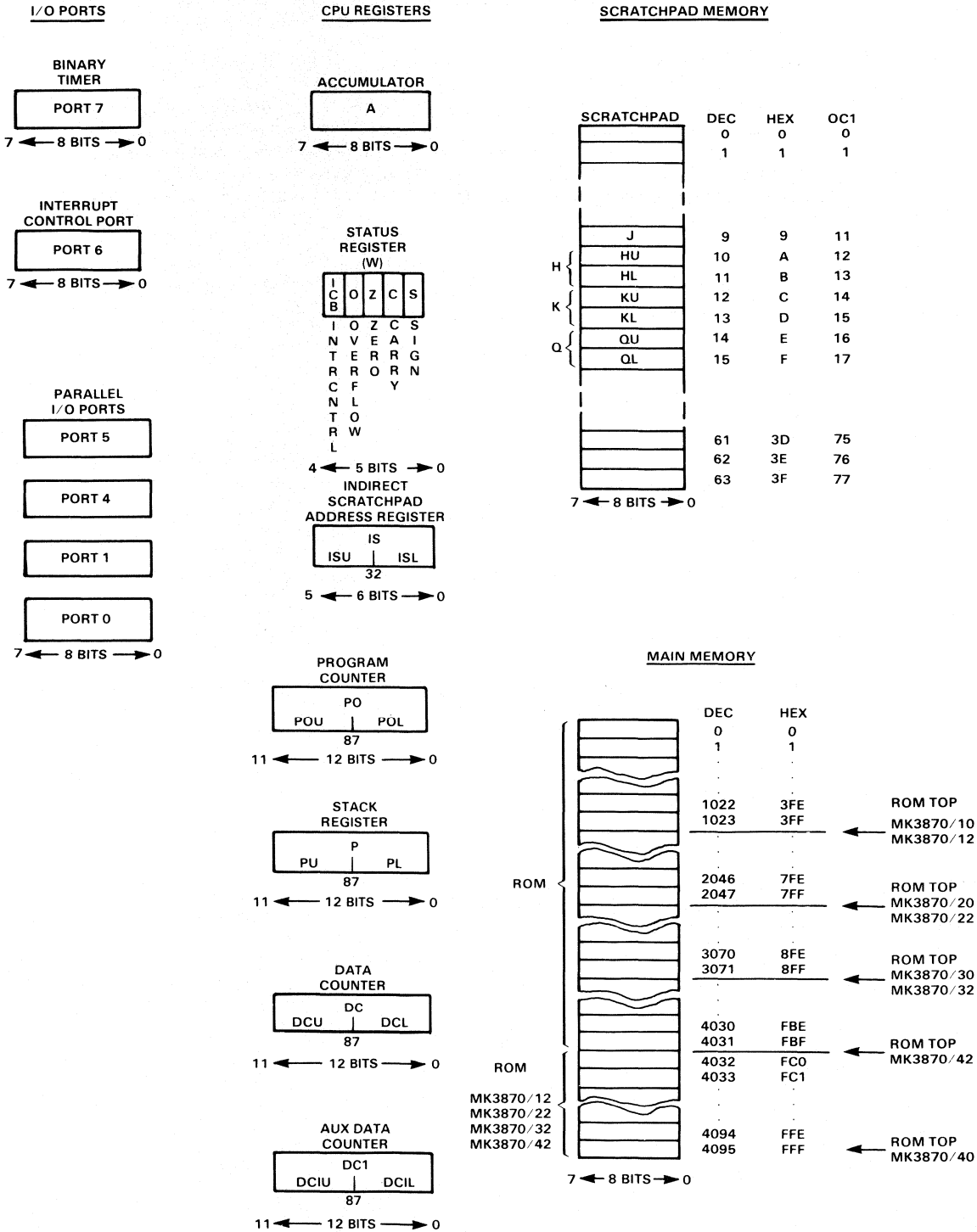
External logic can alter program execution sequence within the CPU by interrupting ongoing operations, as described in Section 5; however, interrupts are allowed only when the ICB bit is set to 1; interrupts are disallowed when the ICB bit is reset to 0.

A SUMMARY OF STATUS BITS
Table 4-1

OVERFLOW = $CARRY + \overline{CARRY}$
ZERO = $\overline{ALU} \oplus \overline{ALU} \oplus \overline{ALU} \oplus \overline{ALU} \oplus \overline{ALU} \oplus \overline{ALU} \oplus \overline{ALU} \oplus \overline{ALU}$
CARRY = $CARRY_7$
SIGN = $\overline{ALU}$

3870 FAMILY PROGRAMMING MODEL

Figure 4-3



4.6 MAIN MEMORY AND MAIN MEMORY ADDRESSING

The Main Memory section of the 3870 consists of a combination of ROM and executable RAM. There are four registers associated with the Main Memory Array. These are the Program Counter (P0), the Stack Register (P), the Data Counter (DC) and the Auxiliary Data Counter (DC1). The Program Counter is used to address instructions during program execution. P is used to save the contents of P0 during an interrupt or subroutine call. Thus, P contains the return address at which processing is to resume upon completion of the subroutine or the interrupt routine.

The Data Counter (DC) is used to address data tables. This register is auto-incrementing. Of the two data counters only DC can access the ROM. However, the XDC instruction allows DC and DC1 to be exchanged.

Associated with the address registers is an Adder/Incrementer. This logic element is used to increment P0 or DC when required and is also used to add displacements to P0 on relative branches or to add the data bus contents to DC in the ADC (Add Data Counter) instruction.

The amount of on chip ROM currently available on 3870 Family devices ranges from 0 bytes on P-PROM emulator parts to 4096 bytes on the MK3870/40. The microcomputer program and data constants may be stored in the program ROM. When a ROM access is required, the appropriate address register (P0 or DC) is gated onto the ROM address bus and the ROM ouput is gated onto the main data bus. On the P-PROM devices, external EPROM memeory is addressed and used as program memory since there is no on chip ROM on these devices.

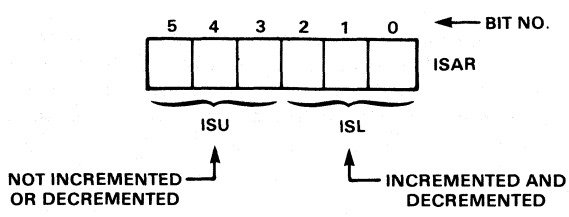
Some 3870 single chip microcomputers, such as the MK3870/22, have RAM which is addressable in the main memory map in addition to the 64 bytes of scratchpad RAM which exists on all 3870 Family devices. This extra RAM can be used for additional storage of variables, or since it is addressed the same as program memory, it is conceivable that software routines can be loaded into this RAM memory and executed. It is for this reason that additional RAM on 3870 microcomputers is termed executable RAM.

4.7 SCRATCHPAD AND IS

The scratchpad provides 64 8-bit registers which may be used as general purpose RAM memory. The Indirect Scratchpad Address Register (IS) is a 6 bit register used to address the 64 registers. All 64 registers may be accessed using IS. In addition, the lower order 12 registers may also be directly addressed.

IS can be visualized as holding two octal digits as illustrated in Figure 4-4. This visualization of IS is important since a number of instructions increment or decrement only the least significant 3 bits of IS when referencing scratchpad bytes via IS. This makes it easy to reference a buffer consisting of contiguous scratchpad bytes. For example, when the low order octal digit is incremented or decremented IS is incremented from octal 27 (0 '27') to 0 '20' or is decremented from 0 '20' to 0 '27'. (The notation O'nn' is used to specify an octal number.)

THE ISAR REGISTER
Figure 4-4



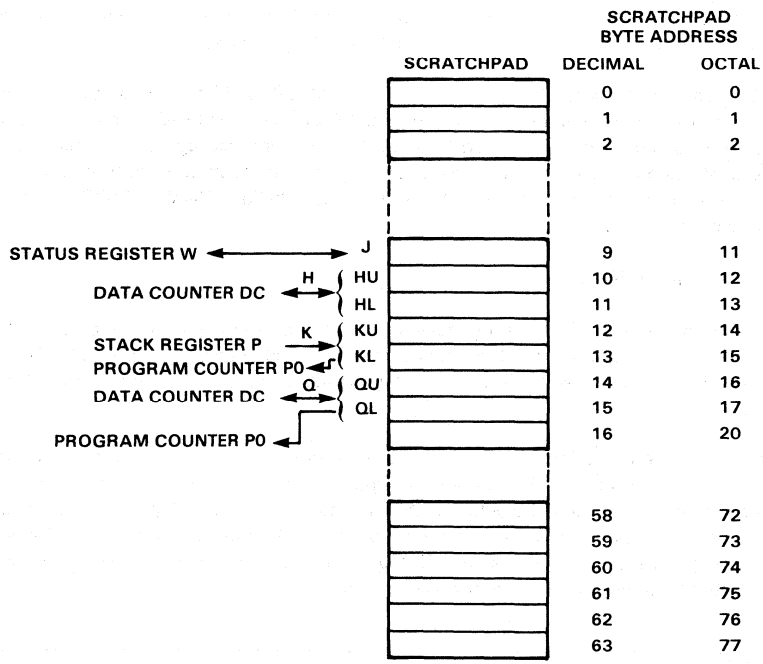
This feature of the IS is very useful in many program sequences. All six bits of IS may be loaded at one time or either half may be loaded independently.

Scratchpad registers 9-15 (decimal) are given mnemonic names (J,H,K, and Q) because of special linkages between these registers and other registers such as the Stack Register. These special linkages facilitate the implementation of multi-level interrupts and subroutine nesting. For example, the instruction LR K,P stores the lower eight bits of the Stack Register into 13 (K lower or KL) and stores the upper three bits of P into register 12 (K upper or KU).

Figure 4-5 identifies the data transfers which may take place by executing a single 3870 instruction. For example, the illustration: W register ◄► J means that a single instruction can move the contents of the W (or status) register to scratchpad register 9 (also called the J register). Another single instruction can move data in the opposite direction. Some linkages exist in only one direction, e.g. P0,◄- Q.

SCRATCHPAD REGISTER MAP

Figure 4-5



4.8 I/O PORTS

The 3870 provices four complete bidirectional Input/Output ports. These are ports 0, 1, 4, and 5. In addition, the Interrupt Control Port is addressed as port 6 and the binary timer is addressed as port 7. An output instruction (OUT or OUTS) causes the contents of A to be latched into the addressed port. An input instruction (IN or INS) transfers the contents of the port to A (port 6 is an exception which is described in Chapter 5).

4.9 EXTERNAL RESET

When $\overline{\text{RESET}}$ is taken low the content of the Program Counter is pushed to the Stack Register and then the Program Counter and the ICB bit of the W Status Register are cleared. The original Stack

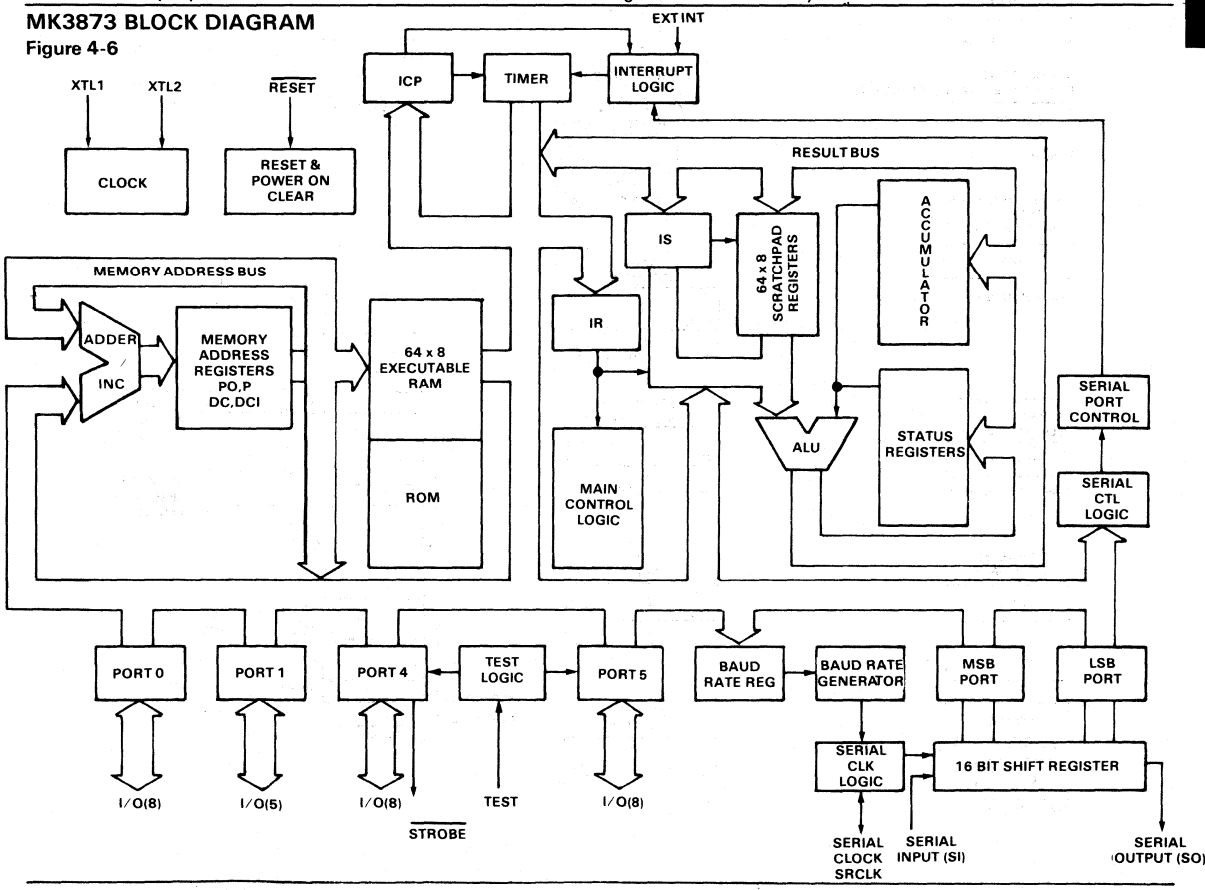
Register content is lost. Ports 4,5,6 and 7 are loaded with H '00'. The contents of all other registers and ports are unchanged or undefined. When RESET is taken high the first program instruction is fetched from ROM location 0000 Hex. When an external reset of the 3870 occurs, P0 is pushed into P and the old contents of P are lost. It must be noted that an external reset is recognized at the start of a machine cycle and not necessarily at the end of an instruction. Thus if the 3870 is executing a multi-cycle instruction, that instruction is not completed and the contents of P upon reset may not necessarily be the address of the instruction that would have been executed next. It may, for example, point to an immediate operand if the reset occurred during the second cycle of a LI or CI instruction. Additionally, several instructions (JMP, PI,PK, LR P0,Q) as well as the interrupt acknowledge sequence modify P0 in parts. That is, they alter P0 by first loading one part and then the other and the entire operation takes more than one cycle. Should reset occur during this modification process the value pushed into P will be part of the old P0 (the as yet unmodified part) and part of the new P0 (already modified part). Thus care should be taken (perhaps by external gating) to insure that reset does not occur at an undesirable time if any significance is to be given to the contents of P after a reset occurs.

4.10 SERIAL I/O

The MK3873 features an on-chip serial I/O port which is addressed as additional I/O within the device. This serial port is capable of either synchronous or asychronous serial data transfers. The heart of the serial port is a 16-bit Shift Register that is double buffered on transmit and receive. The Shift Register clock source may be either the internal baud rate generator or an external clock. An end-of-word vectored interrupt is generated for both the transmit and receive mode so that the CPU overhead is only at the word rate and not at the serial bit rate. This serial channel can be used to provide a low-cost data channel for communicating between 3873 microcomputers or between a 3873 and another host computer. The serial port is also very flexible so that it could be used for other purposes such as an interface to external serial logic or serial memory devices.

MK3873 BLOCK DIAGRAM

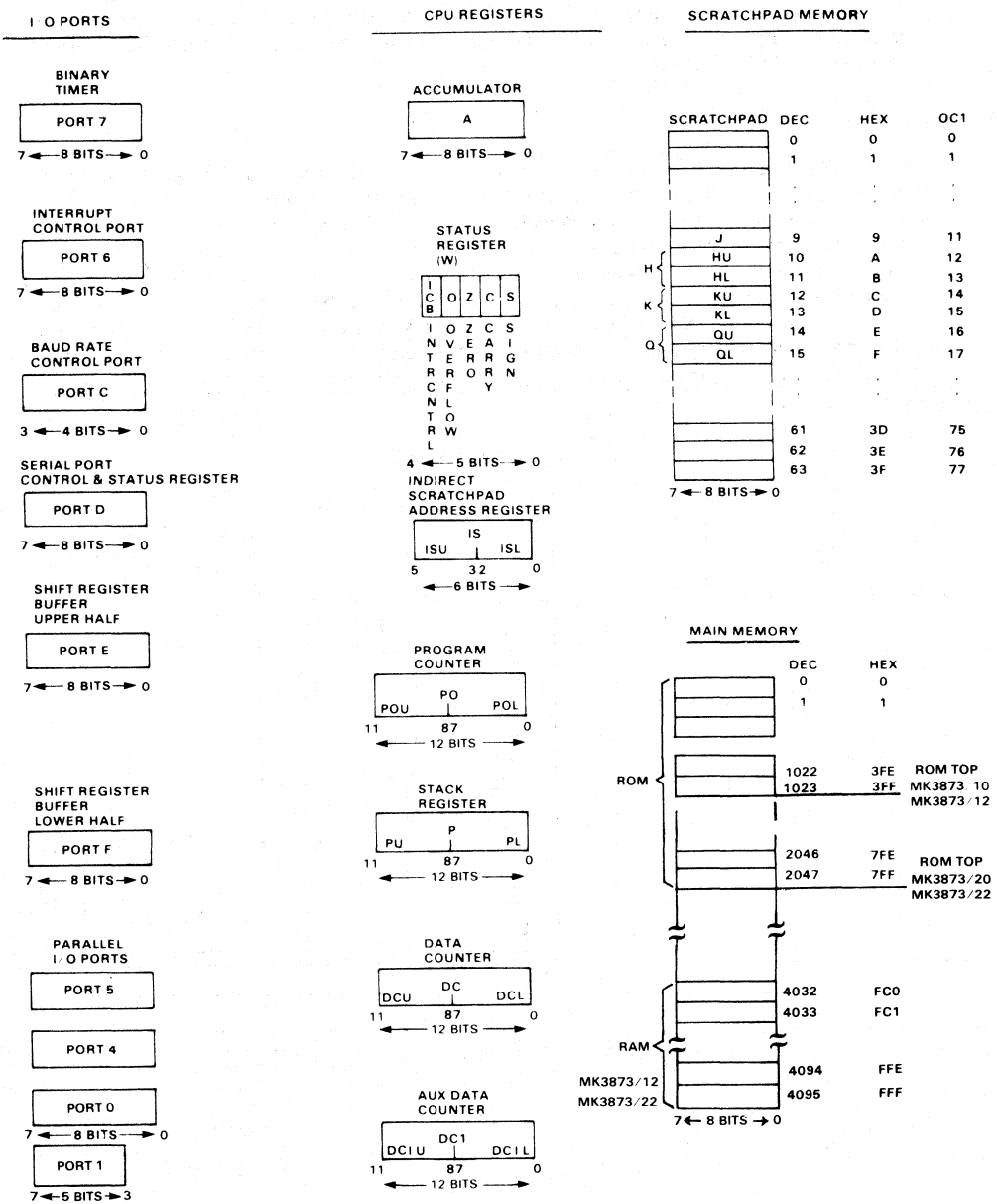
Figure 4-6



The architecture of the MK3873 is identical to that of the rest of the devices in the 3870 Family, with the exception of the serial port logic. The block diagram of the MK3873 is shown in Figure 4-6. Addressing of the serial port logic is accomplished through I/O instructions. A programming model of the MK3873 is shown in Figure 4-7. Operation and programming of the serial port is thoroughly discussed in the MK3873 Data Sheet.

MK3873 PROGRAMMING MODEL

Figure 4-7

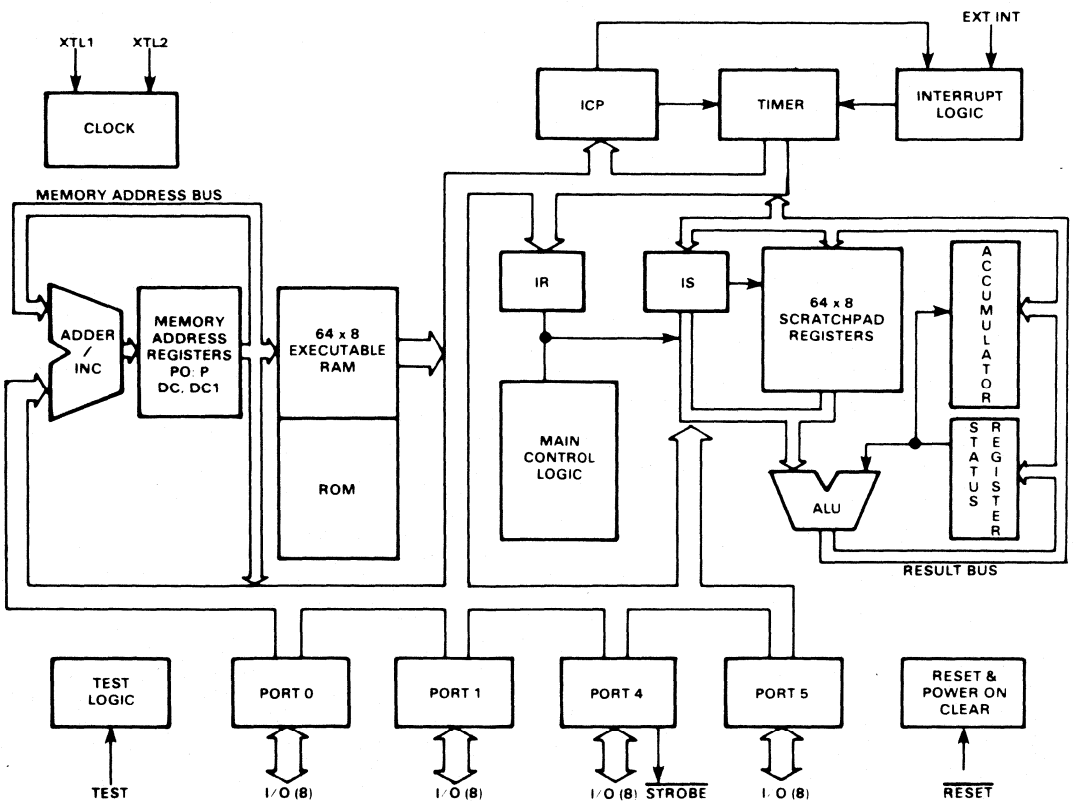


4.11 STANDBY POWER

The MK3875 offers the addition of a Low Power Standby mode of operation on its 64 bytes of Executable RAM. The Low Power Standby feature provides a means of retaining data in the Executable RAM on the MK3875 while the main power supply line (V_{CC}) is at 0 volts and the rest of the MK3875 microcomputer is shut down. The Executable RAM is powered from an auxiliary power supply input (V_{SB}) while operating in the Low Power Standby mode. When V_{SB} is maintained at or above its minimum level, data is retained in the executable RAM memory with a very low power dissipation.

The block diagram of the MK3875 single chip microcomputer is shown in Figure 4-8. The architecture is identical with that of the MK3870 with the exception of the auxiliary power supply, V_{SB}. The user is referred to the MK3875 Data Sheet for a detailed description of the operation of the MK3875 in the Low Power Standby mode.

MK3875 BLOCK DIAGRAM
Figure 4-8



5.0 TIMER AND EXTERNAL INTERRUPT OPERATION

5.1 INTRODUCTION

The Timer is an 8-bit binary counter which is software programmable to operate in one of three modes: the Interval Timer Mode, the Pulse Width Measurement Mode, or the Event Counter Mode. As shown in Figure 5-1, associated with the Timer are an 8-bit register called the Interrupt Control Port, a programmble prescaler, and an 8-bit modulo-N register. A functional logic diagram is shown in Figure 5-2.

5.2 INTERRUPT CONTROL PORT

The desired timer mode, prescale value, starting and stopping the timer, active level of the EXT INT pin, and local enabling or disabling of interrupts are selected by outputting the proper bit configuration from the Accumulator to the Interrupt Control Port (port 6) with an OUT or OUTS instruction. Bits within the Interrupt Control Port are defined as follows:

INTERRUPT CONTROL PORT (PORT 6)

Bit 0 - External Interrupt Enable
Bit 1 - Timer Interrupt Enable
Bit 2 - EXT INT Active Level
Bit 3 - Start/Stop Timer
Bit 4 - Pulse Width/Interval Timer
Bit 5 - ÷2 Prescale
Bit 6 - ÷5 Prescale
Bit 7 - ÷20 Prescale

A special situation exists when reading the Interrupt Control Port (with an IN or INS instruction). The Accumulator is not loaded with the content of the ICP; instead, Accumulator bits 0 through 6 are loaded with 0's while bit 7 is loaded with the logic level being applied to the EXT INT pin, thus allowing the status of EXT INT to be determined without the necessity of servicing an external interrupt request. When reading the Interrupt Control Port (port 6) bit 7 of the Accumulator is loaded with the actual logic level being applied to the EXT INT pin, regardless of the status of ICP bit 2 (the EXT INT Active Level bit); that is, if EXT INT is at +5V, bit 7 of the Accumulator is set to a logic 1, but if EXT INT is at GND then Accumulator bit 7 is reset to logic 0. This capability is useful in establishing a high speed polled handshake procedure or for using EXT INT as an extra input pin if external interrupts are not required and the Timer is used only in the Interval Timer Mode. However, if it is desirable to read the contents of the ICP then one of the 64 scratchpad registers or one byte of RAM may be used to save a copy of whatever is written to the ICP.

The rate at which the timer is clocked in the Interval Timer Mode is determined by the frequency of the internal Φ clock and by the division value selected for the prescaler. (The internal Φ clock operates at one-half the external time base frequency.) If ICP bit 5 is set and bits 6 and 7 are cleared, the prescaler divides Φ by 2. Likewise, if bit 6 or 7 is individually set the prescaler divides Φ by 5 or 20 respectively. Combinations of bits 5, 6, and 7 may also be selected. For example, if bits 5 and 7 are set while 6 is cleared the prescaler will divide by 40. Thus possible prescaler values are divided by 2, 5, 10, 20, 40, 100, and 200.

Any of three conditions will cause the prescaler to be reset: whenever the timer is stopped by clearing ICP bit 3, execution of an output instruction to port 7, (the timer is assigned port address 7), or on the trailing edge transition of the EXT INT pin when in the Pulse Width Measurement Mode. These last two conditions are explained in more detail below.

An OUT or OUTS instruction to Port 7 will load the content of the Accumulator to both the Timer and the 8-bit time constant register, reset the prescaler, and clear any previously stored timer interrupt request. As previously noted, the Timer is an 8-bit down counter which is clocked by the prescaler in the Interval Timer Mode and in the Pulse Width Measurement Mode. The prescaler is not used in the Event Counter Mode. The Modulo-N register is a buffer whose function is to save the value

which was most recently output to Port 7. The Time Constant register is used in all three timer modes.

5.3 INTERVAL TIMER MODE

The operation of the Interval Timer Mode is graphically depicted in Figure 5-3a. When ICP bit 4 is cleared (logic 0) and at least one prescale bit is set the Timer operates in the Interval Timer Mode. When bit 3 of the ICP is set the Timer will start counting down from the time constant value (abbreviated as TC in Figure 5-3a). After counting down to 01 Hex, the Timer returns to the time constant value at the next count. On the transition from 01 Hex to N Hex the Timer sets a timer interrupt request latch. Note that the interrupt request latch is set by the transition to N Hex and not be the presence of N Hex in the Timer, thus allowing a full 256 counts if the Time Constant register is preset to 00 Hex. If bit 1 of the ICP is set, the interrupt request is passed on to the CPU section of the 3870. However, if bit 1 of the ICP is a logic 0 the interrupt request is not passed on to the CPU section but the interrupt request latch remains set. If ICP bit 1 is subsequently set, the interrupt request will then be passed on to the CPU section. (Recall from the discussion of the Status Register's Interrupt Control Bit that the interrupt request will be acknowledged by the CPU section only if ICB is set.) Only two events can reset the timer interrupt request latch; when the timer interrupt request latch is acknowledged by the CPU section, or when a new load of the Time Constant register is performed.

Consider an example in which the Time Constant register is loaded with 064 Hex (decimal 100). The timer interrupt request latch will be set at the 100th count following the timer start and the timer interrupt request latch will repeatedly be set on precise 100 count intervals. If the prescaler is set at divided by 40 the timer interrupt request latch will be set every 4000 Φ clock periods. For a 2MHz Φ clock (4MHz time base frequency) this will produce 2 millisecond intervals.

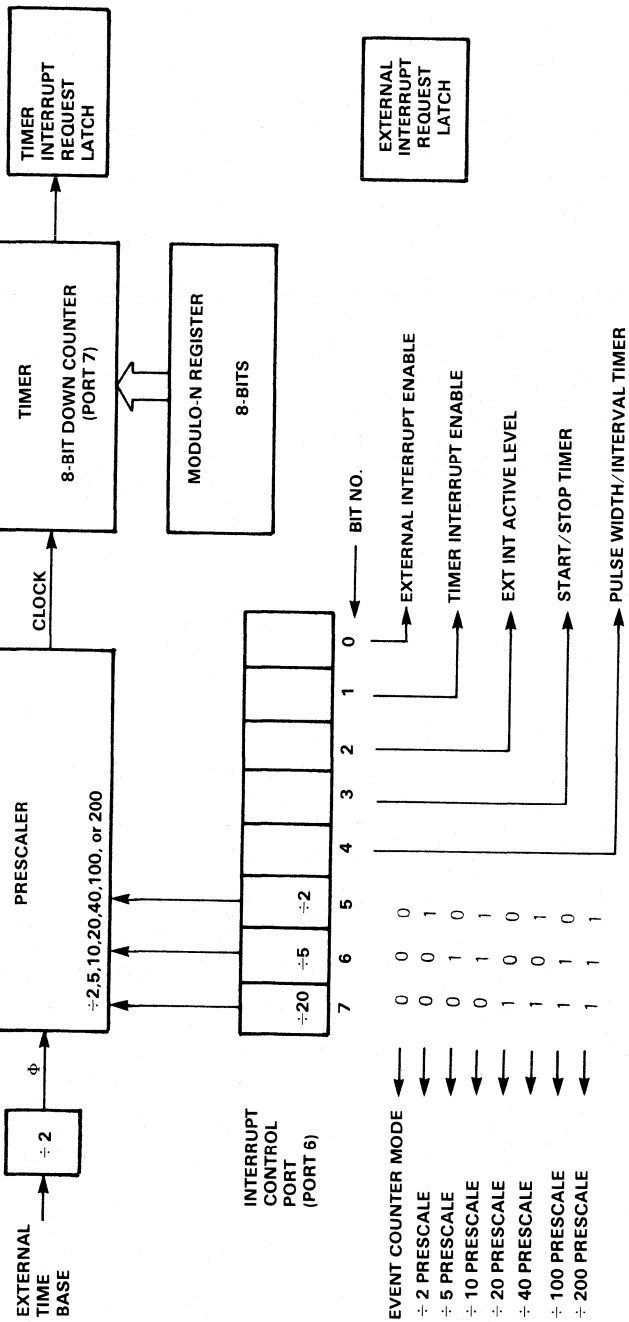
The range of possible intervals is from 2 to 51,200 Φ clock periods (1us to 25.6ms for a 2MHz Φ clock). However, approximately 50 Φ periods is a practical minimum because the time between setting the interrupt request latch and the execution of the first instruction of the interrupt service routine is at least 29 Φ periods (the response time is dependent upon how many priviledged instructions are encountered when the request occurs); 29 is based on the timer interrupt occurring at the beginning of a non-priviledged short instruction. To establish time intervals greater than 51,200 Φ clock periods is a simple matter of using the timer interrupt service routine to count the number of interrupts, saving the result in one or more of the scratchpad registers 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 (IN 7 of INS 7) and may take place "on the fly" without interfering with normal timer operation. Also, the Timer may be stopped at any time by clearing bit 3 of the ICP. The Timer will hold its current contents indefinitely and will resume counting when bit 3 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 a cumulative truncation error.

A summary of other timer errors is given in the timing section of this specification. For a free running timer in the Interval Timer Mode the time interval between any two interrupt requests may be in error by ± 6 Φ clock periods although the cumulative error over many intervals is zero. The prescaler and Timer generate precise intervals for setting the timer interrupt request latch but the time out may occur at any time within a machine cycle. (There are two types of machine cycles; short cycles which consist of 4 Φ clock periods and long cycles which consist of 6 Φ clock periods.) Interrupt requests are synchronized with the start of a machine cycle thus giving rise to the possible ±6 Φ error. Additional errors may arise due to the interrupt request occuring while a priviledged instruction or multicycle instruction is being executed. Nevertheless, for most applications all of the above errors are negligible, especially if the desired time interval is greater than 1 ms.

TIMER AND INTERRUPT CONTROL PORT BLOCK DIAGRAM

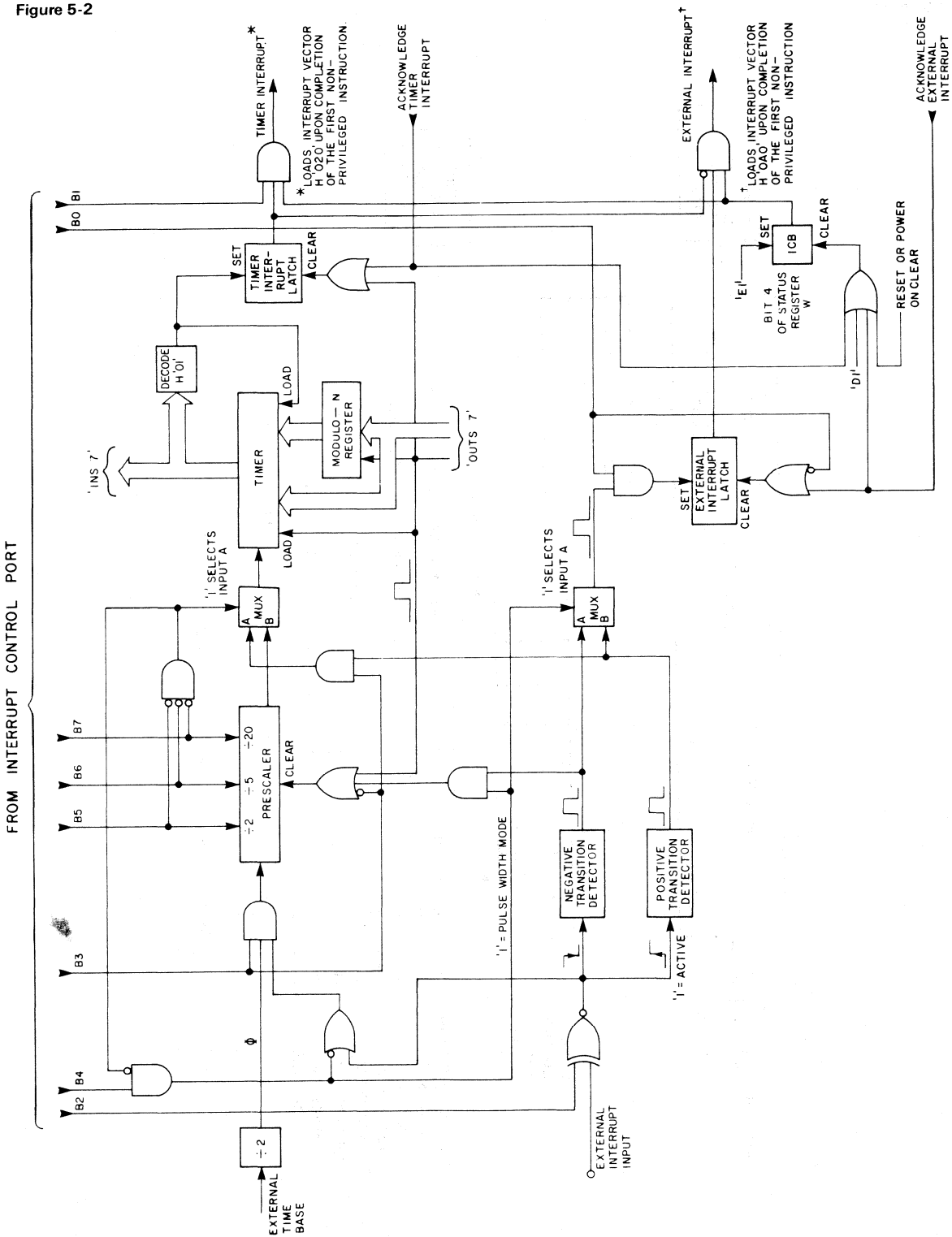
Figure 5-1



NOTE: SEE FIGURE 5 FOR A MORE DETAILED FUNCTIONAL DIAGRAM.

MK3870 TIMER/INTERRUPT FUNCTIONAL DIAGRAM

Figure 5-2



5.4 PULSE WIDTH MEASUREMENT MODE

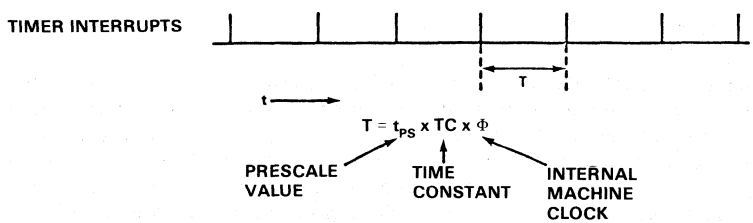
When ICP bit 4 is set (logic 1) and at least one prescale bit is set, the Timer operates in the Pulse Width Measurement Mode. This mode is used for accurately measuring the duration of a pulse applied to the EXT INT pin. The Timer is stopped and the prescaler is reset whenever EXT INT is at its inactive level. The active level of EXT INT is defined by ICP bit 2; if cleared, EXT INT is active low; if set EXT INT is active high. If ICP bit 3 is set, the prescaler and Timer will start counting when EXT INT transitions to the active level. When EXT INT returns to the inactive level the Timer then stops, the prescaler resets, and if ICP bit 0 is set an external interrupt request latch is set. (Unlike timer interrupts, external interrupts are not latched if the ICP Interrupt Enable bit is not set.) The operation of the Pulse Width Measurement Mode of Operation is depicted in Figure 5-3b.

As in the Interval Timer Mode, the Timer may be read at any time, may be stopped at any time by clearing ICP bit 3, the prescaler and ICP bit 1 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 01 Hex to N Hex. Note that the EXT INT 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 time constant value are easily measured by using the timer interrupt service routine to store the number of timer interrupts in one or more scratchpad registers.

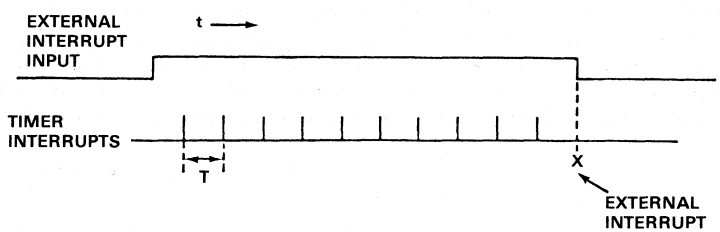
As for accuracy, the actual pulse duration is typically slightly longer than the measured value because the status of the prescaler is not readable and is reset when the Timer is stopped. Thus for maximum accuracy it is advisable to use a small division setting for the pre-scaler.

TIMER OPERATING MODES
Figure 5-3

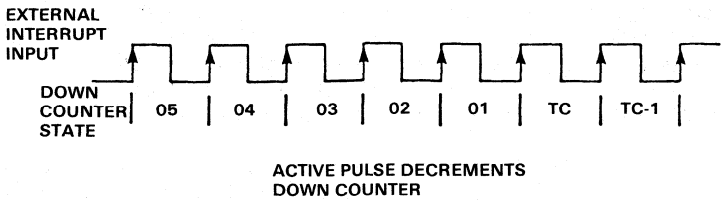
a. INTERVAL TIMER MODE



b. PULSE WIDTH MEASUREMENT MODE



c. EVENT COUNTER MODE



5.5 EVENT COUNTER MODE

When ICP bit 4 is cleared and all prescale bits (ICP 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 EXT INT pin. If ICP bit is set the Timer will decrement on each transition from the inactive level to the active level of the EXT INT pin. The prescaler is not used in this mode; but as in the other two timer modes, the Timer may be read at any time, may be stopped at any time by clearing ICP bit 3 ICP bit 1 functions previously described, and the timer interrupt request latch is set on the Timer's transition from 01 Hex to N Hex.

Normally ICP bit 0 should be kept cleared in the Event Counter Mode otherwise, external interrupts will be generated on the transition from the inactive level to the active level of the EXT INT pin.

For the Event Counter Mode the minimum pulse width required on EXT INT is 2 Φ clock periods and the minimum inactive time is 2 Φ clock periods; therefore, the maximum repetition rate is 500 KHz.

5.6 EXTERNAL INTERRUPTS

When the timer is in the Interval Timer Mode the EXT INT pin is available for non-timer related interrupts. If ICP bit 0 is set, an external interrupt request latch is set when there is a transition from the inactive level to the active level of EXT INT. (EXT INT is an edge-triggered input). The interrupt request is latched until either acknowledged by the CPU section or until ICP bit 0 is cleared (unlike timer interrupt requests which remain latched even when ICP bit 1 is cleared). External interrupts are handled in the same fashion when the Timer is in the Pulse Width Measurement Mode or in the Event Counter Mode, except that only in the Pulse Width Measurement Mode is the external interrupt request latch set on the trailing edge of EXT INT; that is, on the transition from the active level to the inactive level.

5.7 INTERRUPT HANDLING

When either a timer or an external interrupt request is communicated to the CPU section of the 3870, it will be acknowledged and processed at the completion of the first non-priviledged instruction if the Interrupt Control Bit of the Status Register is set. If the Interrupt Control Bit is not set, the interrupt request will continue until either the Interrupt Control Bit is set and the CPU section acknowledges the interrupt or until the interrupt request is cleared as previously described.

If there is both a timer interrupt request and an external interrupt request when the CPU section starts to process the requests, the timer interrupt is handled first.

When an interrupt is allowed the CPU section will request that the interrupting element pass its interrupt vector address to the Program Counter via the data bus. The vector address for a timer interrupt is 020 Hex. The vector address for external interrupts is 0A0 Hex. After the vector address is passed to the Program Counter, the CPU section sends an acknowledge signal to the appropriate

interrupt request latch which clears that latch. The execution of the interrupt service routine will then commence. The return address of the original program is automatically saved in the Stack Register, P.

The Interrupt Control Bit of W (Status Register) is automatically reset when an interrupt request is acknowledged. It is then the programmer's responsibility to determine when ICB will again be set (by executing an EI instruction). This action prevents an interrupt service routine from being interrupted unless the programmer so desires.

The interrupt sequence which occurs within the 3870 when an interrupt is processed is described in Section 6, "TIMING".

6.0 TIMING

6.1 TIMING SIGNALS

There are two principal signals within all 3870 Family devices which are used to generate all the timing signals within the microcomputer. These two timing signals are Φ and WRITE. The execution sequence for each 3870 instruction is timed with these signals. The external time base frequency of the 3870 is divided by two to produce the internal Φ clock. The falling edge of WRITE is used to mark the beginning of a new machine cycle. The WRITE signal has a high-going pulse which is one Φ period in width. A machine cycle is either 4 or 6 Φ periods long, with all instructions requiring between 1 and 5 machine cycles to complete their execution. Figure 6-1 illustrates the timing of the short cycle and the long cycle, along with that of the internal Φ clock.

6.2 INSTRUCTION EXECUTION

The simplest instructions of the 3870 instruction set execute in one short cycle while the most complex instruction (PI) requires two short cycles plus three long cycles. Every instruction execution sequence ends with the next instruction OP code being fetched from memory. The OP code is loaded into the Instruction Register where it is decoded by the Control Logic.

The only instructions which may be executed in a single cycle are those which do not require the use of the Data Bus. This permits the Data Bus to be used to fetch the next instruction OP code simultaneously with the performance of the operation indicated by the current OP code.

Other instructions require more than one cycle to execute. Different operations, specified by the particular instruction, are performed during each of the additionally required long or short cycle(s). In general, CPU operations which do not require the use of the Data Bus can be executed using only a short cycle.

For example, an operation in which the ALU is used to combine two 8-bit words of source data into an 8-bit result may take place without the use of the Data Bus. The source data could come from a scratchpad register and the Accumulator. The last cycle, however, will always be an OP code fetch cycle. In all instructions except the Decrement Scratchpad instruction, the OP code fetch cycle is executed with a short cycle. Following an instruction fetch, CPU logic decodes the fetched instruction code and executes the specified instruction.

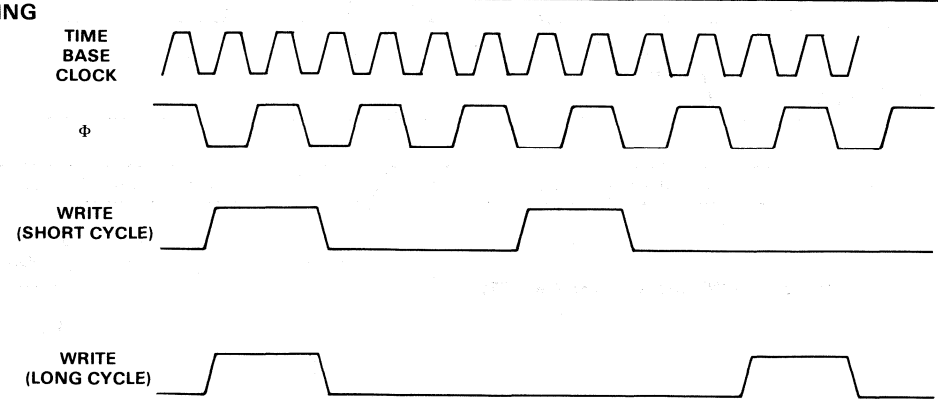
There are many operations which can take place during a machine cycle. Of these various operations, there are three which are of interest to the user:

1) Main Memory Access in P-PROM devices
2) I/O Port Access
3) Interrupt Acknowledge

The timing of these operations is discussed in the following sections.

WRITE CYCLE TIMING

Figure 6-1

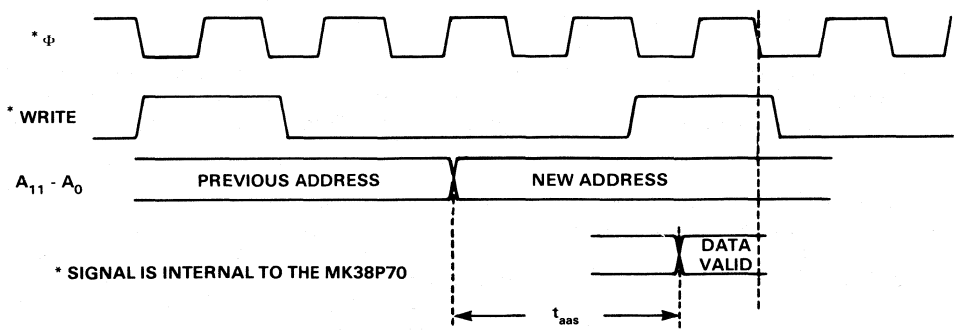


6.3 MAIN MEMORY ACCESS CYCLE

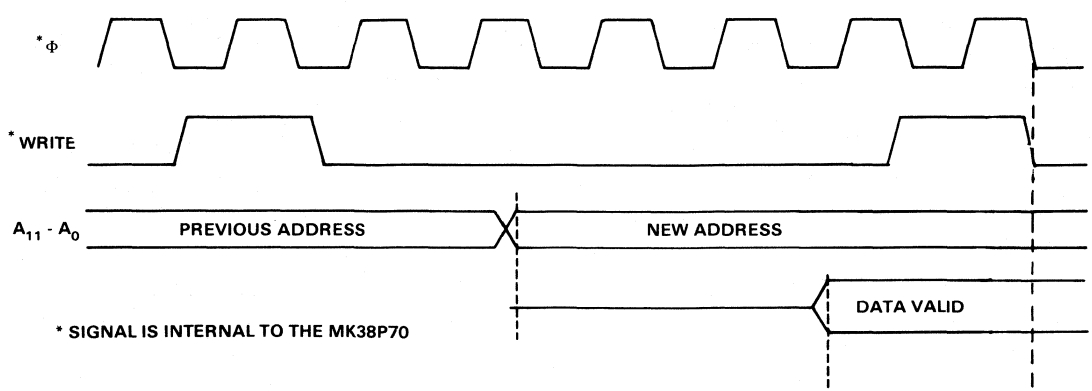
Data may be transferred from one of the 3870's CPU registers and Main Memory. In mask ROM 3870 single chip microcomputers, detailed timing of how a access from Main Memory takes place is of no concern to the user. However, in P-PROM 38P7X devices, program and data memory is contained in an external EPROM. An access of external EPROM memory may take place during a short cycle or a long cycle. The timing diagrams for these two types of machine cycles are shown in Figure 6-2a and b. The worst case memory cycle is the short cycle, during which time an OP code fetch may be performed. After a delay from the falling edge of the WRITE clock, the address lines become stable. Data must be valid at the data output lines of the EPROM memory prior to the next falling edge of the WRITE pulse. The total access time available for the MK38P7X is shown as t_{aas}; or the time when address is stable until data must be valid on the data bus lines. The equation for calculating available memory access time along with some calculated access times based on the listed time base frequencies is also shown in Table 6-1.

38P7X EXTERNAL MEMORY ACCESS CYCLE
Figure 6-2

a. SHORT CYCLE MAIN MEMORY ACCESS (OP CODE FETCH)



b. LONG CYCLE MAIN MEMORY ACCESS



$$t_{aas} = \frac{6}{\text{TIME BASE FREQ.}} - 850 \text{ NS}$$

(FROM ADDRESS STABLE)

	4 MHz	3.58 MHz	3 MHz	2.5 MHz	2 MHz
ACCESS TIME	650 ns	825 ns	1.15 μs	1.55 μs	2.15 μs

6.4 I/O PORT ACCESS CYCLE

Data may be transferred between the Accumulator and one of the 3870's I/O port locations. The timing for this instruction cycle is described in Figure 6-3.

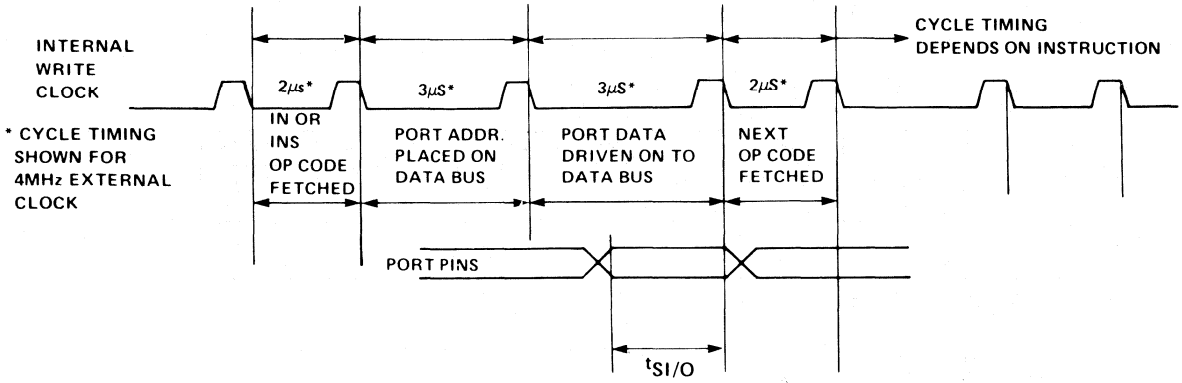
I/O operations involving ports 0 and 1 execute in two machine cycles. During the first cycle, the fetched instruction is decoded and data is either sent from the Accumulator to the I/O latch or enabled from the I/O pin to the Accumulator, depending on whether the instruction is an ouput or an input. At the falling edge of WRITE (marking the end of the first cycle and beginning of the second cycle) the data is strobed into either the latch (OUTS) or the Accumulator (INS) respectively. The second cycle is then used by the CPU for its next instruction fetch. The setup time required for data at the I/O port pin(s) to be stable is shown as $t_{SI/O}$ and the delay time for output data to be stable at the I/O port pins is given as $t_{DI/O}$. Both $t_{SI/O}$ and $t_{D/IO}$ are specified in the appropriate 3870 device data sheet. Figure 6-3c and d illustrate timing for an I/O operation to Ports 0 and 1.

For I/O operations performed at other I/O port locations, three cycles are required to execute the instruction. During the first cycle, the instruction is decoded, and the port address is placed on the internal Data Bus. This first cycle is a long cycle. Then, data is transferred between the Accumulator and the I/O port location during the second (long) cycle. The setup and delay times, $t_{S/IO}$ and $t_{D/IO}$, also apply to these I/O operations. The timing for these I/O operations is pictured in Figures 6-3a and b.

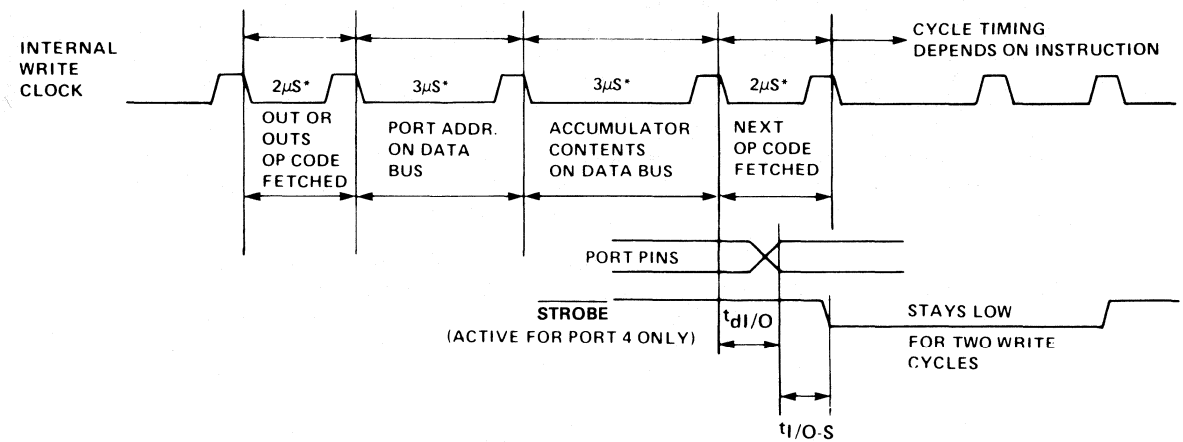
When an output instruction is performed at Port 4, an output ready strobe ($\overline{\text{STROBE}}$) is issued automatically. $\overline{\text{STROBE}}$ provides a single low pulse which stays low for two WRITE cycles following the OP code fetch cycle which terminates the OUT 04 or OUTS 4 instruction. Since $\overline{\text{STROBE}}$ is timed from the WRITE clock, the length of the pulse shown as t_{SL} is dependant on the timing of the instruction following the output operation to Port 4. Thus, t_{SL} will be a minimum of two short cycles and a maximum of two long cycles in length, minus some delay. $\overline{\text{STROBE}}$ pulse length, t_{SL} is specified in the appropriate 3870 device data sheet. Likewise, the delay to falling edge of $\overline{\text{STROBE}}$, $t_{I/O\text{-}S}$, is specified in the appropriate device data sheet.

INPUT/OUTPUT AC TIMING

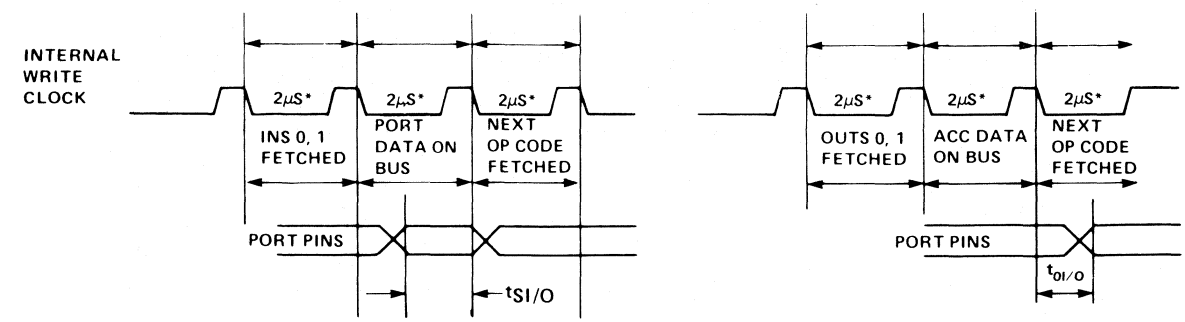
Figure 6-3



A. INPUT ON PORT 4 OR 5



B. OUTPUT ON PORT 4 OR 5



C. INPUT ON PORT 0 OR 1

D. OUTPUT ON PORT 0, 1

6.5 INTERRUPT TIMING

This section describes the timing associated with an interrupt acknowledge cycle occuring in the 3870. Figure 6-4 details the interrupt sequence which occurs whether the interrupt request is from an external source via EXT INT or from the 3870's internal timer. Events are labeled with the letters A through G and are described below. Programming of the Timer and External Interrupt is discussed in Section 5.

EVENT A

An interrupt request must satisfy a set up time requirement prior to the rising edge of the WRITE clock to guarantee that it will be recognized during that machine cycle. Otherwise, it might be one machine cycle later before it is recognized.

EVENT B

Event B represents the instruction being executed when the interrupt occurs. The last cycle of B is normally the instruction fetch for the next cycle. However, if B is not a priviledged instruction and the CPU's Interrupt Control Bit is set, then the last cycle becomes a "freeze" cycle rather than a fetch. At the end of the freeze cycle the interrupt request latches are inhibited from altering the interrupt daisy-chain so that sufficient time will be allowed for the daisy-chain to settle. (If B is a priviledged instruction, the instruction fetch is not replaced by a freeze cycle; instead, the fetch is performed and the next instruction is executed.) Although unlikely to be encountered, a series of priviledged instructions will be sequentially executed without interrupt. One more instruction, called a 'protected' instruction, will always be executed after the last priviledged instruction. The last cycle of the protected instruction then performs the freeze.

The dashed lines on EXT INT illustrate the last opportunity for EXT INT to cause the last cycle of a non-protected instruction to become a freeze cycle.

The freeze cycle is a short cycle (4 Φ clock periods) in all cases except where B is the Decrement Scratchpad instruction, in which case the freeze cycle is a long cycle (6 Φ clock periods).

$\overline{\text{INT REQ}}$ goes low on the next negative edge of WRITE if both PRO IN is low and the appropriate interrupt enable bit of the Interrupt Control Part is set. Both $\overline{\text{INT REQ}}$ and WRITE are internal signals.

EVENT C

A NO-OP long cycle to allow time for the internal priority chain to settle.

EVENT D

The program counter (P0) is pushed to the stack register (P) in order to save the return address. The interrupt circuitry places the lower 8 bits of the interrupt vector address onto the data bus. This is always a long cycle.

EVENT E

A long cycle in which the interrupt circuitry places the upper 8 bits of the interrupt vector address onto the data bus.

EVENT F

A short cycle in which the interrupting interrupt request latch is cleared. Also, the CPU's Interrupt Control Bit is cleared, thus disabling interrupts until an EI instruction is performed. The fetch of the next instruction from the interrupt address.

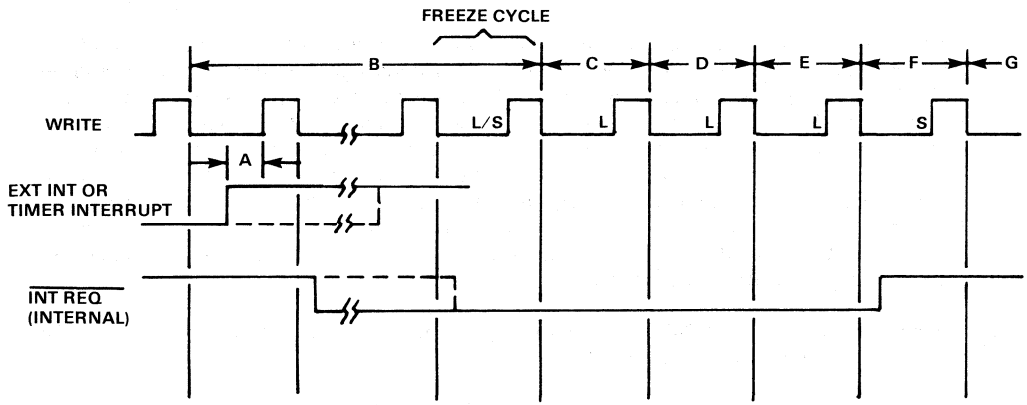
EVENT G

Begin execution of the first instruction of the interrupt service routine.

6.6 SUMMARY OF INTERRUPT SEQUENCE

For the MK3870 the interrupt response time is defined as the time elapsed between the occurence of EXT INT going active (or the Timer transitioning to H 'N') and the beginning of execution of the first instruction of the interrupt service routine. The interrupt response time is a variable dependent upon what the microprocessor is doing when the interrupt request occurs. As shown in Figure 6-4 the minimum interrupt response time is 3 long cycles plus 2 short cycles plus one WRITE clock pulse width plus a setup time of EXT INT prior to the leading edge of the WRITE pulse--a total of 27 Φ clock periods plus the setup time. At a 2 MHz Φ this is 14.25 us. Although the maximum could theoretically be infinite, a practical maximum is 35 μs (based on the interrupt request occuring near the beginning of a PI and LR K, P sequence).

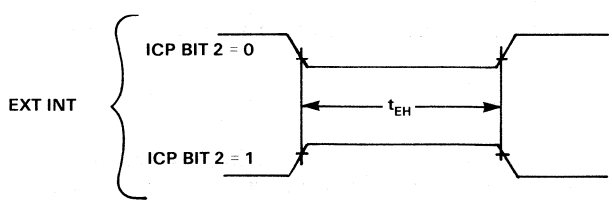
INTERRUPT SEQUENCE
Figure 6-4



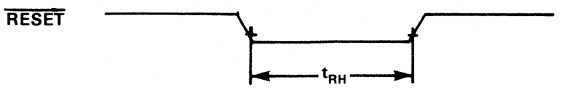
6.7 EXTERNAL INTERRUPT TIMING

The External Interrupt pin is an edge-sensitive interrupt which may be programmed to be active on either a positive going or a negative going edge. In order to guarantee that the internal External interrupt request latch be set, the pulse on the External Interrupt line must meet a minimum hold time which is shown as t_{EH} in Figure 6-5, below. The appropriate 3870 device data sheet specifies the minimum value for t_{EH}.

EXTERNAL INTERRUPT TIMING
Figure 6-5



RESET HOLD TIME
Figure 6-6



6.8 RESET TIMING

In order for a proper reset operation to take place, $\overline{\text{RESET}}$ on the 3870 must be held low for a minimum hold time which is shown as t_{RH} in Figure 6-6 below. The appropriate device data sheet specifies the minimum hold time required for $\overline{\text{RESET}}$.

6.9 TIMER ERRORS

Definitions:

Error = Indicated time value - actual time value

tpsc = t Φ x Prescale Value

Interval Timer Mode:

Single interval error, free running (Note 3)	±6 tΦ
Cumulative interval error, free running (Note 3)	0
Error between two Timer reads (Note 2)	±(tpsc + tΦ)
Start Timer to stop Timer error (Notes 1, 4)	+tΦ to —(tpsc + tΦ)
Start Timer to read Timer error (Notes 1, 2)	—5tΦ to —(tpsc + 7tΦ
Start Timer to interrupt request error (Notes 1, 3)	—2tΦ to —8tΦ
Load Timer to stop Timer error (Note 1)	+tΦ to —(tpsc + 2tΦ)
Load Timer to read Timer error (Notes 1, 2)	—5tΦ to —(tpsc +8tΦ)
Load Timer to interrupt request error (Notes 1, 3)	—2tΦ to —9tΦ

Pulse Width Measurement Mode:

Measurement accuracy (Note 4)	+5Φ to —(tpsc + 2tΦ)
Minimum pulse width of EXT INT pin	2tΦ

Event Counter Mode:

Minimum active time of EXT INT pin	2tΦ
Minimum inactive time of EXT INT pin	2tΦ

Notes:

1. All times which entail loading, starting , or stopping the Timer are referenced from the end of the last machine cycle of the OUT or OUTS instruction.
2. All times which entail reading the Timer are referenced from the end of the last machine cycle of the IN or INS instruction.
3. All times which entail the generation of an interrupt request are referenced from the start of the machine cycle in which the appropriate interrupt request latch is set. Additional time may elapse if the interrupt request occurs during a privileged or multicycle instruction.
4. Error may be cumulative if operation is repetitively performed.

7.0 MK3870 HARDWARE IMPLEMENTATION

7.1 INTRODUCTION

This section discusses some of the external hardware which is required in a system using an MK3870 Family Device. The following subjects are covered:

1) Power On Clear circuitry
2) V_{CC} Decoupling
3) Test Logic
4) 3870 Time Base Options

7.2 POWER-ON CLEAR

The intent of the Power-On-Reset circuitry on the 3870 is to automatically reset the device following a typical power-up situation, thus saving external reset circuitry in many applications. This circuitry is not guaranteed to sense a "Brown Out" (low voltage) condition nor is it guaranteed to operate under all possible power-on situations.

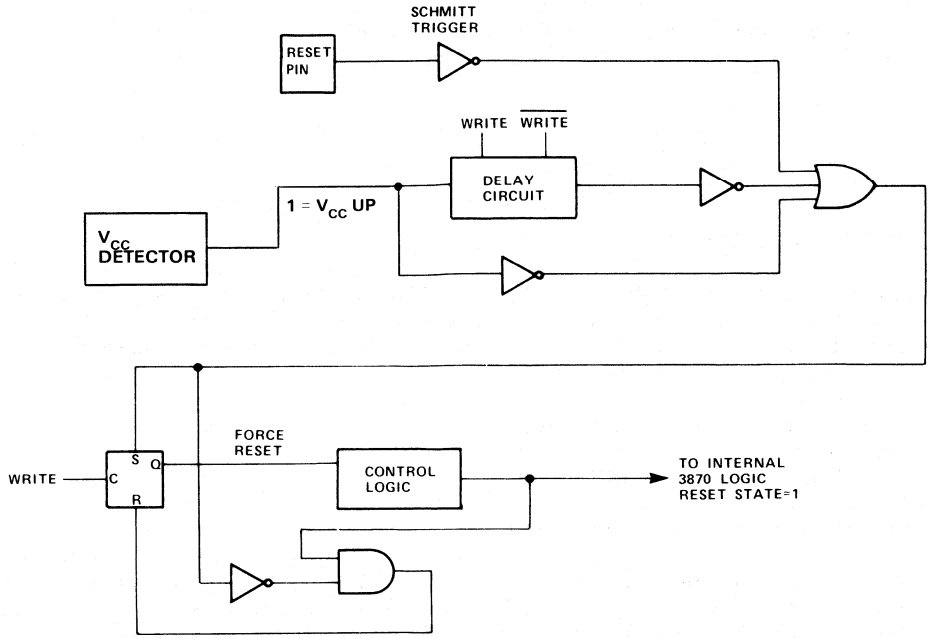
Two conditions are required before the 3870 will leave the reset state and begin operation. Refer to Figure 7-1 as an aid to the following description. The On-Chip V_{CC} detector senses a minimum value of V_{CC} before it will allow the 3870 to operate. The threshold of this detector is set by analog circuitry. Because a stable voltage reference is not available with MOS processing, processing variations will cause this threshold to vary from a low of 3.0 volts to a high near 4.3 volts with 3.5 volts being typical.

The second condition required is that the clocks of the 3870 must be functioning. Typically the clocks will start to function at V_{CC} equal to 3 to 3.5 volts but Mostek cannot guarantee any operation below V_{CC} minimum. The output of the delay circuit in Figure 7-1 will stay low until the clocks start to function. If the input to the delay circuit is high, typically after 100 cycles of the WRITE clock (800 cycles of the external clock) the output of the delay circuit will go high allowing the 3870 to begin execution.

If V_{CC} falls to ground for at least a few hundred nanoseconds the output of the delay circuit will go low immediately and the 3870 will reset.

MK3870 POWER ON CLEAR BLOCK DIAGRAM

Figure 7-1



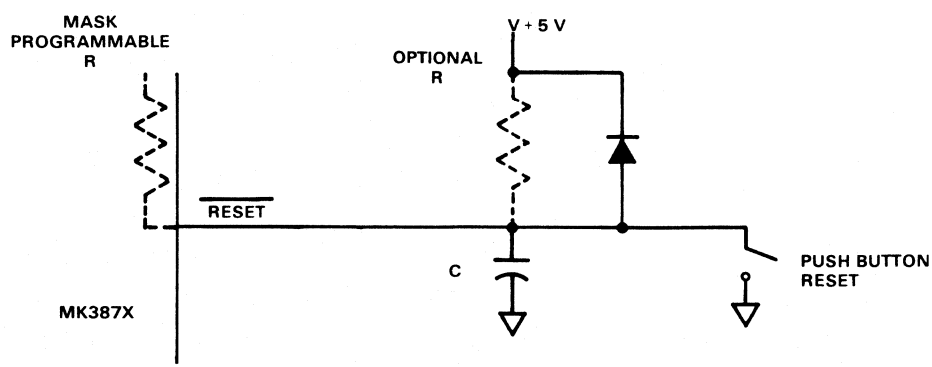
The internal logic may detect a valid V_{CC} and clocks at V_{CC} less than V_{CC} minimum and allow the 3870 to start execution after the time delay. With a slowly rising power supply the part may start running before V_{CC} is within the guaranteed range. When power-on-clear is required with a slowly rising power supply, an external capacitor must be used on the $\overline{\text{RESET}}$ pin to hold it below V_{ILR} until V_{CC} above V_{CC} minimum. (Note: The option to disconnect the internal pull-up resistor on $\overline{\text{RESET}}$ is available which allows the use of a larger external pull-up resistor and a small capacitor on $\overline{\text{RESET}}$).

In many applications, it is desirable if the unit does an automatic power-on-clear, but not mandatory. The unit will have a $\overline{\text{RESET}}$ push button and if the unit does not power-up correctly or malfunctions because of some disturbance on the V_{CC} line, the operator will simply press $\overline{\text{RESET}}$ and restore normal operation. If is for these applications that the internal power-on-clear circuitry was designed.

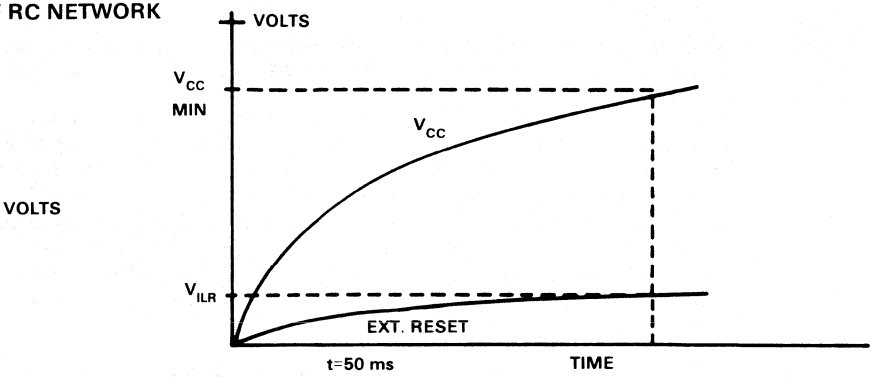
Figure 7-2 shows a schematic of an RC network which can be tied to the $\overline{\text{RESET}}$ pin of the 3870. Figure 7-3 shows the desired response of the RC network.

RECOMMENDED RC NETWORK FOR RESET

Figure 7-2



DESIRED RESPONSE OF RC NETWORK
Figure 7-3



In some applications it is required that the microcomputer continue to run properly without operator intervention after brown-outs, power line disturbances, electrical noise, computer malfunction due to a programming bug or any other disturbance except a catastrophic failure of some component.

One concept used to keep computers running is that of the "WATCHDOG TIMER". The computer is programmed to periodically reset the watchdog timer during the normal execution of its program (this is easily done in the 3870 as its normal application is in some control function which is typically periodic). As long as the computer continues to execute its program the watchdog timer is continually reset and never times out. Should the computer stop executing its program for whatever reason, the watchdog timer will time out producing a $\overline{\text{RESET}}$ pulse to the CPU restarting execution. This is a very positive way to assure that the computer is doing its job, i.e., executing the program. It is important that the software driving the watchdog timer test as many functional blocks (timer, ALU, scratchpad RAM, and Ports) of the 3870 as possible before reseting the watchdog timer. This is because operation of the 3870 with an out of spec power supply may allow some of the functions to operate correctly while other functions are not operable.

Mostek can guarantee correct operation of the 3870 only while the V_{CC} voltage remains within its specified limits. If proper operation of the 3870 must be guaranteed after a distrubance on the V_{CC} line, then an external circuit must be used to monitor the V_{CC} line and produce a $\overline{\text{RESET}}$ to the 3870 whenever V_{CC} is out of the specified limits.

A related characteristic to power-on-clear is the Startup time of the basic timing element. When using the LC and RC time base modes, the time base network begins to function almost immediately once V_{CC} is high enough to allow the on-chip oscillator to operate (V_{CC} typically $< V_{CC}$ min.) Operation with a crystal is partly mechanical and some start time is required to get the mass of the crystal into vibrational motion. This time is basically dependent on the frequency (mass) of the crystal. Crystals within the frequency range used by the 3870 typically require about 2 to 50 msec to start oscillating. Of course, this time may vary greatly from crystal to crystal and is also a function of the power supply rise time characteristic. However, higher frequency crystals typically start faster than lower frequencies.

The condition of the port pins during the power-on-clear sequence is often asked. The port pins or the $\overline{\text{STROBE}}$ line cannot be specified until V_{CC} reaches minimum operating voltage and the 3870 enters the $\overline{\text{RESET}}$ state. Before this, the port pins may stay at V_{SS}, may track V_{CC} as it rises, or they may track V_{CC} part way up then return to V_{SS} (Ports 4 & 5 will go to V_{CC} once the clocks are tunning and the 3870 has sufficient V_{CC} to properly operate the internal control logic and I/O ports).

7.3 VCC DECOUPLING

The 3870 family devices have dynamic circuitry internally which requires a good high frequency decoupling capacitor to surpress noise on the V_{CC} line. A .01 μF or .1 μF ceramic capacitor should be placed between V_{CC} and ground, located physically close to the 3870 device. This will reduce noise generated by the 3870 to about 70-100 millivolts on the V_{CC} line.

7.4 TEST LOGIC

Special test logic is implemented to allow access to the internal main data bus for test purposes.

In normal operation, the TEST pin may be left unconnected, but it is recommended that TEST be grounded. When TEST is placed at about $V_{CC}/2$, port 4 becomes an output of the internal data bus and port 5 becomes a wired-OR input to the internal data bus. The data appearing on the port 4 pins is logically true whereas input data forced on port 5 must be logically false. When TEST is placed at a high voltage, the ports act as above and additionally the 2K x 8 program ROM is prevented from driving the data bus. In this mode operands and instructions may be forced externally through port 5 instead of being accessed from the program ROM. When TEST is either the first state or the high state, $\overline{\text{STROBE}}$ ceases its normal function and becomes a machine cycle clock (identical to the inverse of the internal WRITE clock).

Timing complexities render the capabilities associated with the TEST pin impractical for use in a user's application, but these capabilities are thoroughly sufficient to provide a rapid method for thoroughly testing the 3870.

7.5 3870 TIME BASE OPTIONS

This section describes the selection of a time base for NMOS MK3870 Family single chip microcomputers. The 3870 contains an on-chip oscillator circuit which provides an internal clock. The frequency of the oscillator circuit is set from the external time base network. The time base for the 3870 may originate from one of four sources:

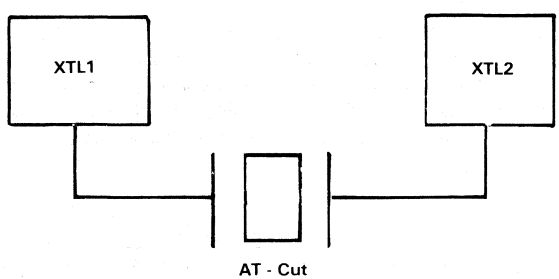
1) Crystal
2) LC Network
3) RC Network
4) External Clock

The four configurations are described in the following sections. There is an internal capacitor between XTL 1 and GND and an internal capacitor between XTL 2 and GND. Thus external capacitors are not neccessarily required. In all external clock modes the external time base frequently is divided by two to form the internal Φ clock.

7.5.1 CRYSTAL SELECTION

The use of a crystal as the time base is highly recommended as the frequency stability and reproducability from system to system is unsurpassed. The Crystal Mode time base configuration is shown in Figure 7-4.

CRYSTAL MODE CONNECTION
Figure 7-4



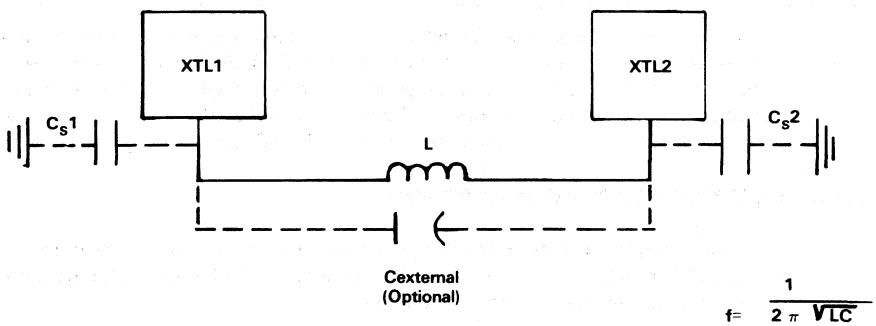
Through careful buffering of the XTL1 pin, it may be possible to amplify this waveform and distribute it to other devices. However, Mostek recommends that a separate active device (such as a 7400 series TTL gate) be used to oscillate the crystal and the waveform from that ocillator be buffered and supplied to all devices, including the 3870, in the event that a single crystal is to provide the time base for more than just a single 3870.

While a ceramic resonator may work with the 3870 crystal oscillator, it was not designed specifically to support the use of this component. Thus, Mostek does not support the use of a ceramic resonator either through proper testing, parametric specification, or applications support.

7.5.2 LC NETWORK

The LC time base configuration can be used to provide a less expensive time base for the 3870 than can be provided with a crystal. However, the LC configuration is much less accurate than is the crystal configuration. The LC time base configuration is shown in Figure 7-5. Also shown in the figure are the specified parameters for the LC components, along with the formula for calculating the resulting time base frequency.

LC MODE CONNECTION
Figure 7-5

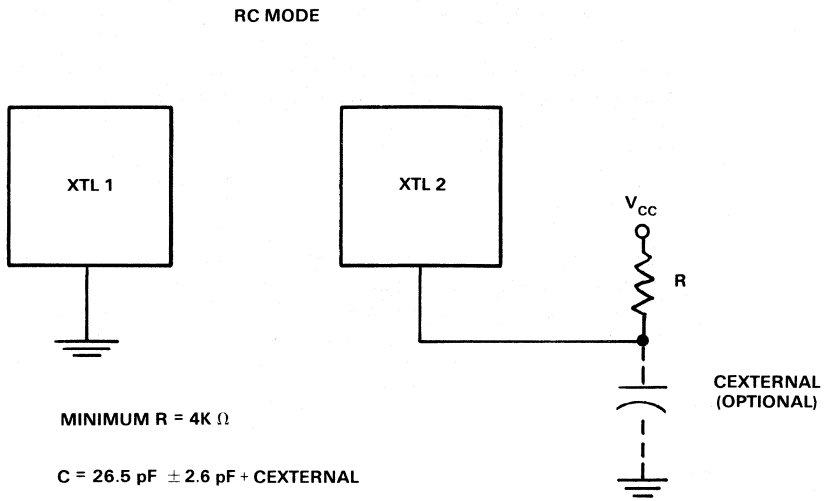


Variation in time base frequency with the LC network can arise from one of four sources: 1) Variation in the value of the inductor. 2) Variation in the value of the external capacitor. 3) Variation in the value of the internal capacitance of the 3870 at XTL1 and XTL2 and 4) Variation in the amount of stray capacitance which exists in the circuit. Therefore, the actual frequency which is generated by the LC circuit is within a range of possible frequencies, where the range of frequencies is determined by the worst case variation in circuit parameters. The designer must select component values such that the range of possible frequencies with the LC mode does not go outside of the specified operating frequency range for the appropriate device.

7.5.3 RC CLOCK CONFIGURATION

The time base for the 3870 may be provided from an RC network tied to the XTL 2 pin, when XTL 1 is grounded. A schematic picturing the RC clock configuration is shown in Figure 7-6. The RC time base configuration is intended to provide an inexpensive time base source for applications in which timing is not critical. Some users have elected to tune each unit using a variable resistor or external capacitor thus reducing the variation in frequency. However, for increased time base accuracy Mostek recommends the use of the Crystal or LC time base configuration.

RC MODE CONNECTION
Figure 7-6



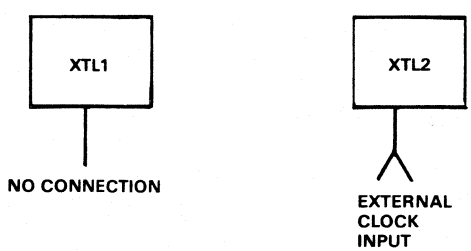
The designer must select the RC product such that a frequency of less than the minimum time base freuqency specified for the appropriate MK3870 Family device. Also, the RC product must not allow a frequency greater than the maximum time base frequency specified for the appropriate device. Temperature induced variations in the external components should be considered in calculating the RC product.

7.5.4 EXTERNAL CLOCK CONFIGURATION

The connection for the external clock time base configuration is shown in Figure 7-7. Refer to the DC Characteristics section in the appropriate 3870 Family device data sheet for proper input levels and current requirements.

Refer to the Capacitance section of the appropriate 3870 Family device data sheet for input capacitance.

EXTERNAL MODE CONNECTION
Figure 7-7



8.0 MK3870 INSTRUCTION SET

8.1 INTRODUCTION

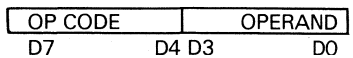
This section describes the execution and timing of the MK3870 Family instruction set. The 3870 instruction set is compatible with that of the multi-chip F8 Family.

8.2 3870 ADDRESSING MODES

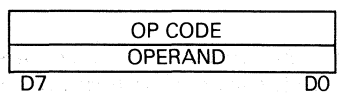
Most of the 3870 instructions operate on data stored in internal CPU registers, in main memory, in scratchpad memory, or in the I/O ports. The term "addressing mode" refers to how the address of this data is generated. This section gives a summary of the types of addressing used in the 3870.

8.2.1 IMMEDIATE ADDRESSING

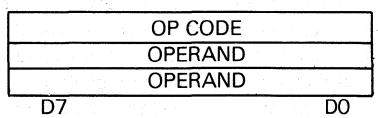
In this mode of addressing, the operand is contained in memory following the OP code of the instruction. Immediate addressing can be used in one byte, two byte, and three byte instructions. For a one byte instruction which uses immediate addressing the 3870 uses a special short form instruction format. The OP code is contained in the most significant four bits (D7-D4) of a byte in memory while a four bit operand is specified in the least significant four bits as shown below:



A two byte instruction which uses immediate addressing contains an 8-bit OP code which is specified in the first byte of the instruction and an 8-bit immediate operand in the second byte.

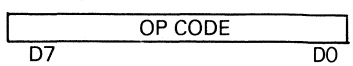


For certain instructions requiring greater than an 8 bit immediate operand, two bytes are required to specify an immediate operand. An example would be the DCI instruction, which loads an immediate value into the Data Counter. These instructions are three bytes long as shown below:



8.2.2 IMPLIED ADDRESSING

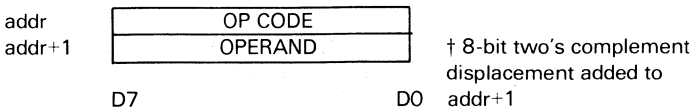
Instructions which use Implied addressing are instructions which are one byte long in which the operand or operands are implicitly specified in the instruction OP code itself. For example, an instruction which uses Implied addressing would be the 'LR Q,DC' instruction. The result of this instruction would be the contents of the Data Counter register loaded into the Q Linkage register.



8.2.3 RELATIVE ADDRESSING

Relative addressing is used exclusively by the Branch instructions in the 3870. Relative addressing uses one byte of data following the OP code to specify a displacement from the current Program Counter location to which a program jump can occur. This displacement is

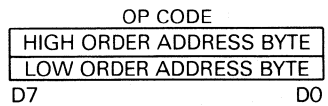
a signed two's complement number that is added to the address of the byte containing the displacement:



Relative addressing allows program jumps to locations within the range from -127 bytes backward to +128 bytes forward relative to the memory location containing the branch instruction op code. Relative addressing requires only two bytes of memory space to implement. For most programs relative branches 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.

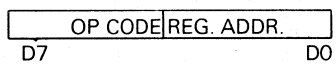
8.2.4 EXTENDED ADDRESSING

Extended addressing provides for up to 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 a subroutine may be called:



8.2.5 SCRATCHPAD ADDRESSING

Scratchpad addressing is utilized by instructions which access the Scratchpad Register Array. These instructions utilize the 3870's short form instruction format and are therefore only one byte long, with the instruction OP code specified in the most significant four bits:



The least sigificant four bits are used as an address to designate the scratchpad register whose contents are to be used as the operand in the instruction. The Scratchpad Register array contains 64 bytes of RAM. When the value of the four bit operand field is in the range of 0 through 0B (Hex) then the corresponding scratchpad location is accessed directly. When the operand value is 0C, 0D, or 0E (Hex), then the Indirect Scratchpad Register (IS) is used to point to the scratchpad location which is to be accessed. Note that the IS register can be used to point to any location in the scratchpad register array. Scratchpad locations 0CH, 0DH, 0EH, and 0FH are actually Linkage Registers K and Q, and, as such, may be directly accessed through instructions which use implied addressing.

The IS register is six bits wide which is divided into two halves, called IS Upper (ISU) and IS Lower (ISL). The Scratchpad registers may be thought of as an 8 x 8 array, with ISU pointing to a row of registers in the array and ISL pointing to a column of registers. This is illustrated below in Figure 8-1.

OCTAL REPRESENTATION OF SCRATCHPAD REGISTER ARRAY
Figure 8-1

		IS 0	upper 1	-> 2	3	4	5	6	7
	0	00	01	02	03	04	05	06	07
IS	1	10	11	12	13	14	15	16	17
lower	2	20	21	22	23	24	25	26	27
	3	30	31	32	33	34	35	36	37
	4	40	41	42	43	44	45	46	47
	5	50	51	52	53	54	55	56	57
	6	60	61	62	63	64	65	66	67
	7	70	71	72	73	74	75	76	77

This figure illustrates how the 2 octal digits of the IS register can be visualized as pointing to a scratchpad register in an 8 x 8 matrix.

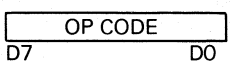
When the IS register is used to point to a register during the execution of an instruction which uses Scratchpad Addressing, the lower three bits of IS are modified according to the value specified in the operand as shown below:

OPERAND	IS LOWER
0CH	Unmodified
0DH	Incremented
0EH	Decremented
0FH	Illegal OP code

Thus, the lower half of the IS register can be automatically incremented, decremented, or left unmodified. When the IS is incremented or decremented during the execution of an instruction, only the lower half of the IS register is modified. As an example, suppose that the IS contains the octal value 27. If an instruction is executed which automatically increments the IS, the value will be changed to an octal 20 so that IS Upper is left unmodified.

8.2.6 INDIRECT MEMORY ADDRESSING

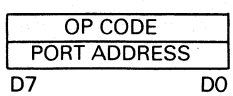
Instructions which operate on data contained in a location in Main Memory must access that data through the Data Counter Register (DC). These instructions are 1 byte in length. The one byte OP code specifies the implicit use of the Data Counter to perform an indirect access of main memory to fetch the 8-bit operand:



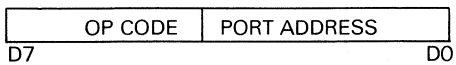
The value of the Data Counter is automatically incremented by one after the access is complete.

8.2.7 I/O PORT ADDRESSING

The 3870 can use one of two forms of the I/O Port Address mode to transfer data between the Accumulator and an I/O port location. Instructions using this type of addressing may be two bytes long with the port address specified in the 8-bit operand field following the OP code, as shown below:



Alternatively, the instruction may use the short form to specify one of the first 16 I/O port locations with a single byte:



8.3 MK3870 INSTRUCTION TYPES

The 3870 instruction set consists of 76 different types of instructions. They can be classified into seven separate groups by function:

1) Arithmetic and Logical Group
2) Branch and Jump Group
3) Address Register Group
4) Accumulator Data Movement Group
5) Input/Output Group
6) CPU Control Group

The operation of the executable instructions in the 3870 is summarized in Table 8-3. An overview of the instruction set operation by group accompanies the summary. Notation used in this table is described in "Notes" at the end of this section.

8.3.1 ARITHMETIC AND LOGICAL GROUP

The Arithmetic and Logical Group of instructions allow the 3870 to perform various operations on data contained in the Accumulator and/or data from one of four sources within the device as listed below:

1) A Scratchpad Register
2) A location in Main Memory
3) An immediate value
4) The Data Counter Register

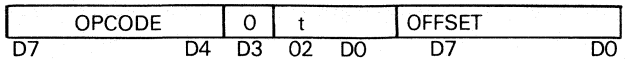
Data in a Scratchpad location may be added, added with decimal adjust, logically AND'ed, or Exclusive OR'ed with data contained in the Accumulator. Additionally, a scratchpad register may be decremented. Instructions in the Arithmetic and Logical group which operate on the Scratchpad utilize Scratchpad Addressing. The contents of the Accumulator may be added, added with decimal adjust, AND'ed, compared, OR'ed or exclusive OR'ed with a location in Main Memory. A location in Main Memory is accessed indirectly through the use of the Data Counter (Indirect Memory Addressing). The contents of the Accumulator may be complemented, incremented, shifted left or right, or AND'ed, OR'ed, Exclusive ORed, or compared with an immediate value. These types of instructions use either Implied or Immediate addressing. Finally, the contents of the Data Counter Register may be added with the contents of the Accumulator, which is treated as a signed binary (two's complement) number, with the result in the Data Counter. This instruction also uses Implied addressing.

8.3.2 BRANCH, JUMP, CALL, AND RETURN GROUP

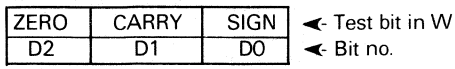
The Branch, Jump, Call, and Return Group include all of those instructions which can be used to transfer program control in the 3870.

Branch instructions which are conditional on the state of one or more of the bits in the Status register (W) are actually one of two basic instructions: Branch on True and Branch on False. The Branch on True instruction is a two byte instruction, with the branch condition contained in a field in the first byte of the instruction along with the OP code, and the second byte containing the relative offset pointing to the branch destination. The form of the Branch

on True instruction is shown below:



The three bit field "t" is actually a mask field for three of the bits which are to be tested in the Status Register. This three bit field is shown in detail below:



When a bit in the "t" field is set to a "1", the corresponding bit in the Status Register is tested during the execution of the Branch on True instruction. When a bit in the "t" field is set to a "0", the state of the corresponding bit in the Status Register is ignored. The operation of the Branch on True instruction is such that the branch to the branch destination address is taken if any of the unmasked bits in the Status Register are true. There are a total of 8 combinations of conditions which can be specified with the Branch on True instruction. Four of these conditions are often used in programming and given unique mnemonic names which are recognized by all 3870 cross assembler programs. These four forms of the Branch on True instruction are Branch on Carry (BC), Branch on Positive (BP) and Branch on Zero (BC). All of the possible branch conditions which exist for the Branch if True instructions are summarized in Table 8-1.

BRANCH CONDITIONS FOR BT INSTRUCTION

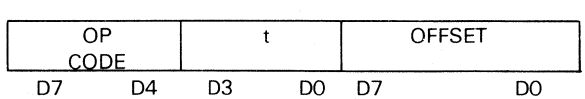
Table 8-1

OPERAND	STATUS FLAGS TESTED				
t	ZERO	CARRY	SIGN	DEFINITION	COMMENTS
0	0	0	0	Non-Functional	An effective 3 cycle NOP
1	0	0	1	Branch if Positive	Same as BP
2	0	1	0	Branch on Carry	Same as BC
3	0	1	1	Branch if Positive or on Carry	
4	1	0	0	Branch if Zero	Same as BZ
5	1	0	1	Branch if Positive	Same as t=1
6	1	1	0	Branch if Zero or on Carry	
7	1	1	1	Branch if Positive or on Carry	Same as t=3

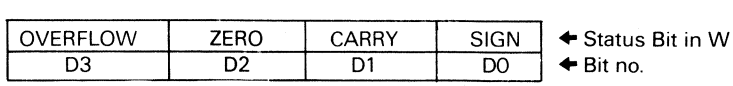
BRANCH CONDITIONS FOR BF INSTRUCTION
Table 8-2

OPERAND	STATUS FLAGS TESTED					
t	OVF	ZERO	CARRY	SIGN	DEFINITION	COMMENTS
0	0	0	0	0	Unconditional Branch Relative	
1	0	0	0	1	Branch on Negative	Same as BM
2	0	0	1	0	Branch if no Carry	Same as BNC
3	0	0	1	1	Branch if no Carry and Negative	
4	0	1	0	0	Branch if not Zero	Same as BZ
5	0	1	0	1		Same as t=1
6	0	1	1	0	Branch if no Carry and result is no Zero	
7	0	1	1	1		Same as t=3
8	1	0	0	0	Branch if there is no Overflow	Same as BNO
9	1	0	0	1	Branch if Negative and no Overflow	
A	1	0	1	0	Branch if no Overflow and no Carry	
B	1	0	1	1	Branch if no Overflow, no Carry and Negative	
C	1	1	0	0	Branch if no Overflow and not Zero	
D	1	1	0	1		Same as t=9
E	1	1	1	1	Branch if no Overflow, no Carry, and not Zero	
F	1	1	1	1		Same as t=B

The Branch if IS not = 7 instruction is used indicate whether or not Similarly, the Branch on False instruction consists of two bytes: The first byte specifies the OP code along with a four bit mask field for the OVERFLOW, ZERO, CARRY, AND SIGN bits in the Status Register. The form of the Branch on False instruction is illustrated below:



The mask field for the Branch on False instruction is:



The mask word selectively enables or disables the test on the bits in the Status Register as in the case of the Branch on True instruction. The operation of the Branch on False instruction is such that the branch to the destination address is taken if all of the unmasked bits in the Status Register are false (logic 0). Special forms of the Branch on False instruction which are given special mnemonic names include the Branch on Minus (BM), Branch on No Carry (BNC), Branch on Not Zero (BNZ), Branch if No Overflow (BNO), and Branch Relative (BR) instructions. All of the possible branch conditions which exist for the Branch if False instructions are summarized in Table 8-2. The branch condition for the BR7 instruction is true when ISC $\neq$ 7. The branch is not not taken when ISC = 7, or when the lower half of the IS register is pointing to the end byte of an 8 byte block in the Scratchpad Register array. It is useful in many program sequences since only the lower half of the IS register can be auto-incremented or auto-decremented.

The Jump instruction allows program control to be transferred to any location within the 3870's internal Main Memory map. This is accomplished through the use of Extended Addressing. The user should take note of the fact that the contents of the Accumulator are modified during the execution of the Jump instruction. The Accumulator is used as a holding register for the most significant 8 bits of the destination address specified in the two byte operand field following the Jump OP code. Thus, this value results in the Accumulator when the Jump instruction is completed. The Jump Indirect (LR P0,Q) instruction can be used to transfer program control to the address in Main Memory which is pointed to by the contents of the Q Linkage Register.

Subroutines may be called either directly using the Call to Subroutine (PI) instruction or indirectly using the Call to Subroutine Indirect (PK) instruction. In either case the return address is placed in the Program Counter Stack Register during the execution of the instruction. The P register facilitates one level of subroutine calls or interrupts. Additional levels of subroutine and/or interrupts are possible by utilizing those instructions which transfer data between the P0 and P registers and the Linkage registers in the Scratchpad Array. These instructions are discussed in the Address Register Instruction description. For a detailed description of multiple level subroutine and interrupt implementation in the 3870, the user should refer to the Mostek application note "Multi-Level Subroutine and Interrupt Handling in the 3870". The user should note that the PI instruction modifies the contents of the Accumulator during the course of its execution. Like the Jump instruction, the Accumulator is used the hold the most significant portion of the destination address, or that portion of the address which is greater than the least significant 8 bits. Thus, this is the value which results in the Accumulator on completion of the instruction.

8.3.3 ACCUMULATOR DATA MOVEMENT GROUP

Instructions in the Accumulator Data Movement Group allow data to be moved between the Accumulator and either a scratchpad register or a location in Main Memory. An immediate 8 bit value may be moved into the Accumulator from the location in Main Memory following the instruction OP code. Data may be transferred between the Accumulator and any location in Main Memory by using the Load Memory (LM) and Store Memory (ST) instructions. The LM and ST instruction utilize Indirect Memory Addressing by pointing to the source/destination memory location with the Data Counter Register.

Data may also be transferred between the Accumulator and any Scratchpad location with instructions in the Accumulator Data Movement Group which use Scratchpad Addressing.

8.3.4 ADDRESS REGISTER GROUP

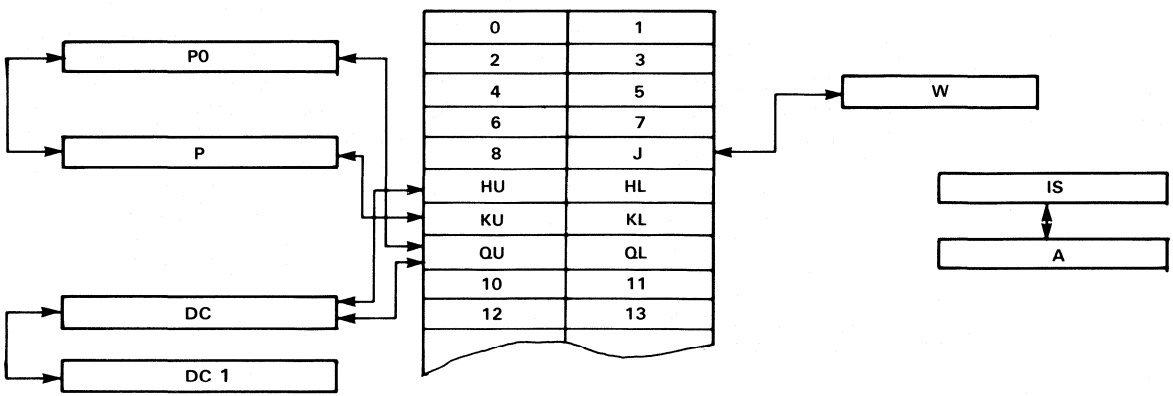
Intructions in the Address Register Group are used to manipulate the contents of registers within the 3870 which are used to address either the Scratchpad registers or Main Memory. The Indirect Scratchpad Address Register (IS) is used to address a location in the Scratchpad. The Program Counter Stack Register (P) is associated with the Program Counter and is primarily used for saving the return address during subroutine calls. The Data Counter (DC) and Auxiliary Data Counter (DC1) are used in address data constants in

Main Memory.

The Load IS Upper (LISU) and Load IS Lower (LISL) instructions can be used to selectively load either the upper three bits or the lower three bits of the IS Register. These instructions cause an immediate three bit value which is contained in the same byte as the OP code to be transferred into the appropriate half of the IS register. The contents of the IS register may be transferred to the Accumutor, or the contents of the Accumulator may be transferred into the IS register, by using the LR A,IS or LR IS,A instructions, respectively.

Data may be transferred between the P, DC, and DC1 registers and special register pairs which reside in the Scratchpad Register Array and are known as Linkage Registers. The Linkage registers are given special mnemonic names: H designates the register pair at locations 10 and 11, K designates the register pair at locations 12 and 13, and Q designates the register pair at locations 14 and 1. Figure 8-2 summarizes the data transfers which are possible between the Linkage Registers and the Address registers. For every such transfer shown in the diagram, there exists a corresponding instruction in the Address Register Group which facilitates that transfer, except for the LR P0,Q instruction, which is discussed with the Branch, Jump, Call, and Return instructions.

3870 ADDRESS REGISTER LINKAGES
Figure 8-2



8.3.5 INPUT/OUTPUT GROUP

There are four types of instructions in the Input/Output group which can access any I/O port location within the 3870. They are the Input (IN), Output (OUT), Input Short (INS), and Output Short (OUTS) instructions. The short form instructions can access any of the lower sixteen I/O port locations in the 3870 I/O port map and require only one byte. The long form instructions require two bytes; one for the OP code and the other to specify a port address from 0-255. In all existing 3870 devices, the short form instructions are sufficient to access all I/O port locations.

8.3.6 CPU CONTROL GROUP

The Enable Interrupts and Disable Interrupts instructions are included in the CPU Control Group. Also included are the instructions which transfer data between the Status Register W and Linkage Register J. These are the LR W,J and LR J,W instructions, which are primarily used to save and restore the status of the 3870 during interrupt subroutines. A No Operation instruction is included in the CPU Control Group.

MK3870 INSTRUCTION SET SUMMARY

Table 8-3

ARITHMETIC AND LOGICAL GROUP

OPERATION	MNEMONIC-OPERAND	ADDR. MODE	DESCRIPTION
Add Carry	LNK	IMP	Add the Carry bit to the contents of the Accumulator
Add Immediate	AI ii	IMM	Add 8-bit immediate operand to the
Add to Data Counter	ADC	IND	Add the contents of the Accumulator to the Data Counter DC; result is in the Data Counter the contents of the Accumulator
Add Memory	AM	IND	Add the contents of memory location [DC] to the contents of the Accumulator
Add Memory Decimal	AMD	IND	Add the contents of memory location [DC] to the Accumulator with decimal adjust
Add Scratchpad	AS r	SCR	The contents of Scratchpad Register 'r' are added to the Accumulator
Add Scratchpad Decimal	ASD r	SCR	The contents of Scratchpad Register 'r' are added to the Accumulator with decimal adjust
And Immediate	NI ii	IMM	Perform the logical AND of the 8-bit immediate operand and the contents of the Accumulator
And Memory	NM	IND	Perform the logical AND between memory location [DC] and the Accumulator
And Scratchpad	NS r	SCR	Scratchpad location 'r' is logically ANDed with the contents of the Accumulator
Compare Immediate	CI ii	IMM	Non-destructive subtraction of the Accumulator from the immediate operand
Compare Memory	CM	INC	Non-destructive subtraction of the Accumulator from the contents of memory location [DC]
Complement	COM	IMP	Performs a one's complement operation on the contents of the Accumulator
Decrement Scratchpad	DS r	SCR	The contents of Scratchpad Register are decremented by 1
Exclusive OR Immediate	XI ii	IMM	Performs a logical Exclusive OR operation of the 8 bit immediate operand and the Accumulator
Exclusive OR Memory	XM	IND	Perform a logical Exclusive OR operation between the Accumulator and memory location [DC]

ARITHMETIC AND LOGICAL GROUP (Continued)

OPERATION	MNEMONIC-OPERAND	ADDR. MODE	DESCRIPTION
Exclusive OR Scratchpad	XS r	SCR	Scratchpad location 'r' is logically exclusive OR'ed with the contents of the Accumulator
Increment	INC	IMP	Add the value of 1 to the Accumulator
OR Immediate	OI ii	IMM	Perform a logical OR of the 8-bit immediate operand and the Accumulator
OR Memory	OM	IND	Perform a logical OR operation between the Accumulator and memory location [DC]
Shift Left 1	SL 1	IMP	Shift the contents of the Accumulator left by one
Shift Left 4	SL 4	IMP	Shift the contents of the Accumulator left by four
Shift Right 1	SR 1	IMP	Shift the contents of the Accumulator right by one
Shift Right 4	SR 4	IMP	Shift the contents of the Accumulator right by four

BRANCH, JUMP, CALL, AND RETURN GROUP

OPERATION	MNEMONIC-OPERAND	ADDR. MODE	DESCRIPTION
Branch Relative	BR aa	REL	Branch unconditionally
Branch if False	BF taa	REL	Branch if all of the unmasked Status bits = 0
Branch if Minus	BM aa	REL	Branch if the Sign bit = 0
Branch if No Carry	BNC aa	REL	Branch if the Carry bit = 0
Branch if No Overflow	BNO aa	REL	Branch if the Overflow bit = 0
Branch if Not Zero	BNZ aa	REL	Branch if the Zero bit = 0
Branch if ISL not = 7	BR7 aa	REL	Branch if the value of the lower half of the IS register is not = 7
Branch on Carry	BC aa	REL	Branch if Carry bit = 1
Branch on Positive	BP aa	REL	Branch if the Sign bit = 1
Branch on True	BT taa	REL	Branch if any unmasked Status bit = 1
Branch on Zero	BZ aa	REL	Branch if the Zero bit = 1
Jump	JMP aaaa	EXT	Program Counter is loaded with the immediate value 'aaaa', the Accumulator is loaded with upper half of aaaa

BRANCH, JUMP, CALL, AND RETURN GROUP

OPERATION	MNEMONIC-OPERAND	ADDR. MODE	DESCRIPTION
Jump Indirect	LR P0,Q	IMP	The Program Counter is loaded with the contents of Linkage Register Q
Call to Subroutine	PI iiii	EXT	The Program Counter is loaded with the value 'iiii'; the return address is saved in P; A is destroyed
Call to Subroutine Indirect	PK	IND	The Program Counter is loaded with the contents of Linkage Register K; the return address is saved in P
Return from Subroutine	POP	IMP	Swap the contents of the Program Counter with the Stack Register P

ACCUMULATOR DATA MOVEMENT GROUP

OPERATION	MNEMONIC-OPERAND	ADDR. MODE	DESCRIPTION
Clear	CLR	IMM	Load Accumulator immediate with zero
Load	LR A,KU	IMP	The Accumulator is loaded with the contents of the upper half of Linkage Register K
Load	LR A,KL	IMP	The Accumulator is loaded with the contents of the lower half of Linkage Register K
Load	LR KU,A	IMP	The upper half of Linkage register K is loaded with the contents of the Accumulator
Load	LR KL,A	IMP	The lower half of Linkage register K is loaded with the contents of the Accumulator
Load	LR A,QU	IMP	The Accumulator is loaded with the contents of the upper half of Linkage Register Q
Load	LR A,QL	IMP	The Accumulator is loaded with the contents of the lower half of Linkage Register Q
Load	LR QU,A	IMP	The upper half of Linkage register Q is loaded with the contents of the Accumulator
Load	LR QL,A	IMP	The upper half of Linkage register Q is loaded with the contents of the Accumulator
Load	LR A,r	SCR	The Accumulator is loaded with the contents of Scratchpad Register 'r'
Load	LR r,A	SCR	Scratchpad Register 'r' is loaded with the contents of the Accumulator
Load Immediate	LI ii	IMM	Load the value of the 8-bit immediate operand to the Accumulator

ACCUMULATOR DATA MOVEMENT GROUP (Continued)

OPERATION	MNEMONIC-OPERAND	ADDR. MODE	DESCRIPTION
Load Immediate Short	LIS 0i	IMM	Load the value of the 8-bit immediate operand to the Accumulator
Load Memory	LM	IND	Load Accumulator with the contents of memory location [DC]
Store Memory	ST	IND	Store the value of the Accumulator in memory location [DC]

ADDRESS REGISTER GROUP

OPERATION	MNEMONIC-OPERAND	ADDR. MODE	DESCRIPTION
Load Data Counter Immediate	DCI iiii	IMM	The Data Counter is loaded with the immediate value 'iiii'
Exchange DC	XDC	IMP	Swap the contents of DC with DC1
Load Data Counter	LR DC,Q	IMP	The Data Counter is loaded with the contents of linkage register Q
Store Data Counter	LR Q,DC	IMP	Linkage Register Q is loaded with the contents of the Data Counter
Load Data Counter	LR DC,H	IMP	The Data Counter is loaded with the contents of Linkage Register H
Store Data Counter	LR H,DC	IMP	Linkage Register H is loaded with the contents of the Data Counter
Load IS Lower	LISL bbb	IMM	Load the lower half of the IS register with the immediate value 'bbb'
Load IS Upper	LISU bbb	IMM	Load the upper half of the IS register with the immediate value 'bbb'
Load IS	LR IS,A	IMP	Load the IS register with the contents of the Accumulator
Store IS	LR A,IS	IMP	The Accumulator is loaded with the contents of the IS register
Load Stack Register	LR P,K	IMP	The Stack Register P is loaded with the contents of Linkage Register K
Store Stack Register	LR K,P	IMP	Linkage Register K is loaded with the contents of the Stack Register

INPUT/OUTPUT GROUP

OPERATION	MNEMONIC-OPERAND	ADDR. MODE	DESCRIPTION
Input	IN aa	IOP	The contents of Port 'aa' are loaded (04-FF) into the Accumulator
Input Short	INS a (0-F)	IOP	The contents of Port 'a' are loaded into the Accumulator
Output	OUT aa (04-FF)	IOP	The contents of the Accumulator are output to Port 'aa'
Output Short	OUTS a (00-04)	IOP	The contents of the Accumulator are output to Port 'a'

CPU CONTROL GROUP

OPERATION	MNEMONIC-OPERAND	ADDR. MODE	DESCRIPTION
Disable Interrupts	DI	IMP	Interrupts to the 3870 are disabled; ICB is reset to 0
Enable Interrupts	EI	IMP	Interrupts are enabled to the 3870; ICB is set to 1
Load Status Register	LR W,J	IMP	The Status Register is loaded with the contents of Linkage Register J
No Operation	NOP	IMP	The contents of the Program Counter are incremented
Store Status Register	LR J,W	IMP	Linkage Register J is loaded with the contents of the Status Register

8.4 INSTRUCTION EXECUTION AND TIMING

A detailed summary of the timing and execution of the 3870 instruction set is included in Table 8-4. This table provides OP codes, instruction cycle sequence, effect on Status flags, and operation information for each and every 3870 instruction. The information contained in each of the columns contained in the table is described below;

OP CODE: The mnemonic name for the instruction is contained in this column.

OPER: The symbolic name for the operand(s) used in the instruction appear in this column.

OBJECT CODE: The hexadecimal equivalent of the machine code to which the instruction translates.

CYCLE: The sequence of machine cycles which occur during the course of execution of the instruction. A Long cycle is symbolized with an "L" and a Short cycle is symbolized with an "S". The order in which these cycles occur during instruction execution appear downward in this column. e.g: The instruction LM is executed with a Long cycle (L), then a Short cycle (S).

μS: The execution time of the instruction is indicated in this column in units of microseconds. This execution time is based on using a time base frequency of 4 MHz, yielding an internal Φ clock frequency of 2 MHz. Execution time which results from a different time base frequency may be calculated by multiplying this time by the value of (time base freq.)/4 MHz.

STATUS FLAGS: The effect which the instruction has on the four flags in the Status register is shown in this column. (See notes for terminology explanation.)

INT: If the instruction is privileged, it is denoted with the letter "p" in this column.

FUNCTION: The operation of the instruction is described symbollicaly in this column. (See Notes for explanation of terminology).

INSTRUCTION TIMING AND EXECUTION
Table 8-4

ARITHMETIC AND LOGICAL GROUP

OP CODE	OPER.	OBJ. CODE	CYC.	μS	STATUS FLAGS O	Z	C	S	INT	FUNCTION
ADC		8E	L S	5	-	-	-	-		C ← DC + A
AI	ii	24	L S	5	↕	↕	↕	↕		A ← A + ii
AM		88	L S	5	↕	↕	↕	↕		A ← A + [DC]
AMD		89	L S	5	↕	↕	↕	↕		A ← A + [DC] w/BCD adjust
AS	r	Cr	S	2	↕	↕	↕	↕		A ← A + r
ASD	r	Dr	S S	4	↕	↕	↕	↕		A ← A + r w/BCD adjust
CI	ii	25ii	L S	5	↕	↕	↕	↕		ii - A

ARITHMETIC AND LOGICAL GROUP (Continued)

OP CODE	OPER.	OBJ. CODE	CYC.	μS	STATUS FLAGS O	Z	C	S	INT	FUNCTION
CM		8D	L S	5	↕	↕	↕	↕		[DC] - A
COM		18	S	2	0		0			A ← A*
DS	r	3r	L	3	↕	↕	↕	↕		r ← r - 1
INC		1F	S	2	↕	↕	↕	↕		A ← A + 1
LNK		19	S	2	↕	↕	↕	↕		A ← A + CRY
NI	ii	21ii	L S	5	0	↕	0	↕		A ← A © ii
NM		8A	L S	5	0	↕	0	↕		A ← A © [DC]
NS	r	Fr	S	2	0	↕	0	↕		A ← A © r
OI	ii	22ii	L S	5	0	↕	0	↕		A ← A v ii
OM		8B	L S	5	0	↕	0	↕		A ← A v [DC]
SL	1	13	S	2	0	↕	0	↕		Shift 'A' left by 1 LSB ← 0
SL	4	15	S	2	0	↕	0	↕		Shift 'A' left by 4 LS nibble ← 0000
SR	1	12	S	2	0	↕	0	1		Shift 'A' right by 1 MSB ← 0
SR	4	14	S	2	0	↕	0	1		Shift 'A' right by 4 MS nibble ← 0000
XI	ii	23ii	L S	5	0	↕	0	↕		A ← A (+) ii
XM		8C	L S	5	0	↕	0	↕		A ← A (+) [DC]
XS	r	Er	S	2	0	↕	0	↕		A ← A (+) r

BRANCH, JUMP, CALL, AND RETURN GROUP

OP CODE	OPER.	OBJ. CODE	CYC.	μS	STATUS FLAGS O	Z	C	S	INT	FUNCTION
BC	aa	82aa	S S/L S	6/7	-	-	-	-		Branch if C = 1
BF	aa	9taa	S S/L S	6/7	-	-	-	-		Branch if unmasked status it is false
BM	aa	91aa	S S/L S	6/7	-	-	-	-		Branch if S = 0
BNC	aa	92aa	S S/L S	6/7	-	-	-	-		Branch if C = 0
BNO	aa	98aa	S S/L S	6/7	-	-	-	-		Branch if O = 0
BNZ	aa	94aa	S S/L S	6/7	-	-	-	-		Branch if Z = 0
BP	aa	81aa	S S/L S	6/7	-	-	-	-		Branch if S = 1
BR	aa	90aa	S L	7	-	-	-	-		Branch always

BRANCH, JUMP, CALL, AND RETURN GROUP (Continued)

OP CODE	OPER.	OBJ. CODE	CYC.	μS	STATUS FLAGS O	Z	C	S	INT	FUNCTION
			S							
			S							
BT	aa	8taa	S	6/7	-	-	-	-		Branch if any unmasked status it is true
			S/L							
			S							
BZ	aa	84aa	S	6/7	-	-	-	-		Branch if Z = 1
			S/L							
			S							
JMP	aaaa	29aaaa	L	11	-	-	-	-	p	P0 ← aaaa;
			L							A ← P0 upper
			L							
			S							
LR	P0,Q	0D	L	8	-	-	-	-		P0 ← Q
			L							
			S							
PI	iiii	28iiii	L	13	-	-	-	-	p	P ← P0;
			S							P0 ← iiii
			L							
			L							
			S							
PK		0C	L	8	-	-	-	-	p	P ← P0;
			L							P0 ← K
			S							
POP		1C	S	4	-	-	-	-	p	P0 <=> P
			S							

NOTE:

In all conditional branch instructions, two possible execution times are given. The shorter time corresponds to the execution time which results when the branch is not taken. This corresponds to the fact that a short cycle is executed in this case. When the branch is taken, a long cycle is executed, and the execution time is longer.

ACCUMULATOR DATA MOVEMENT GROUP

OP CODE	OPER.	OBJ. CODE	CYC.	μS	STATUS FLAGS O	Z	C	S	INT	FUNCTION
CLR	70	70	S	2	-	-	-	-		A ← 00
LR	A,KU	00	S	2	-	-	-	-		A ← KU
LR	A,KL	01	S	2	-	-	-	-		A ← KL
LR	KU,A	04	S	2	-	-	-	-		KU ← A
LR	KL,A	05	S	2	-	-	-	-		KL ← A
LR	A,QU	02	S	2	-	-	-	-		A ← QU
LR	A,QL	03	S	2	-	-	-	-		A ← QL
LR	QU,A	06	S	2	-	-	-	-		QU ← A
LR	QL,A	07	S	2	-	-	-	-		QL ← A
LR	A,r	4r	S	2	-	-	-	-		A ← r

ACCUMULATOR DATA MOVEMENT GROUP (Continued)

LR	r,A	5r	S	2	-	-	-	-		r ← A
LI	ii	20ii	L S	5	-	-	-	-		A ← ii
LIS	i	7i	S	2	-	-	-	-		A ← 7i
LM		16	L S	5	-	-	-	-		A ← [DC]
ST		17	L S	5	-	-	-	-		[DC] ← A

ADDRESS REGISTER GROUP

OP CODE	OPER.	OBJ. CODE	CYC.	μS	STATUS FLAGS O	Z	C	S	INT	FUNCTION
DCI	iiii	2Aiiii	L S L S S	12	-	-	-	-		DC ← iiii
XDC		2C	S S	4	-	-	-	-		DC <=> DC1
LR	DC,Q	0F	L L S	8	-	-	-	-		DC ← Q
LR	Q,DC	0E	L L S	8	-	-	-	-		Q ← DC
LR	DC,H	10	L L S	8	-	-	-	-		DC ← H
LR	H,DC	11	L L S	8	-	-	-	-		H ← DC
LISL		6(0bbb)	S	2	-	-	-	-		ISL ← bbb
LISU		6(1bbb)	S	2	-	-	-	-		ISU ← bbb
LR	A,IS	0A	S	2	-	-	-	-		A ← IS
LR	IS,A	0B	S	2	-	-	-	-		IS ← A
LR	K,P	0 8	L L S	8	-	-	-	-		K ← P
LR	P,K	09	L L S	8	-	-	-	-		P ← K

INPUT/OUTPUT GROUP

OP CODE	OPER.	OBJ. CODE	CYC.	μS	STATUS FLAGS O	Z	C	S	INT	FUNCTION
IN	pp	26pp	L L S	8	0		0			A ← pp
INS	0,1	A0,A1	S S	4	0		0			A ← [0,1]
INS	p	Ap	L L S	8	0		0			A ← [p] p > 1

INPUT/OUTPUT GROUP (Continued)

OUT	pp	27pp	L L S	8	-	-	-	-	p	pp ← A
OUTS	0,1	B0,B1	S S	4	-	-	-	-	p	p0,p1 ← A
OUTS	2-15	B2-BF	L L S	8	-	-	-	-	p	p0-pF ← A

CPU CONTROL GROUP

OP CODE	OPER.	OBJ. CODE	CYC.	μS	STATUS FLAGS O	Z	C	S	INT	FUNCTION
DI		1A	S	2	-	-	-	-		Clear ICB
EI		1B	S	2	-	-	-	-	p	Set ICB
LR	J,W	1E	S	2	-	-	-	-		J ← W
LR	W,J	1D	S	4	-	-	-	-	p	W ← J
NOP		2B	S	2	-	-	-	-		P0 ← P0 + 1

NOTES

p Denotes privileges instruction

Status legend:

m = test and modify according to result
? = unknown
- = not altered
0 = reset to 0
1 = set to 1

Function symbology

← is loaded with
<-> is exchanged with
[ ] the contents of the location pointed to by
© logical AND
v logical OR
a
a Address variable (four bits)
A Accumulator
b One bit immediate operand
DC Data Counter (Indirect Memory Address Register)
DC1 Auxiliary Data Counter
H Scratchpad register pair 10 and 11 (Linkage Register)
i Immediate operand (four bits)
ICB Interrupt Control Bit
IS Indirect Scratchpad Address Register
ISL Least significant 3 bits of IS
ISU Most significant 3 bits of IS
J Scratchpad Register 9
K Scratchpad Register pair 12 and 13 (Linkage Register)
KL Scratchpad Register 13 (K Lower)
KU Scratchpad Register 12 (K Upper)
P0 Program Counter
P Program Counter Stack Register
p I/O Port location
Q Scratchpad Register pair 14 and 15 (Linkage Register)

QL	Scratchpad Register 15 (Q Lower)
QU	Scratchpad Register 14 (Q Upper)
r	Scratchpad Register (any address 0 thru b; see below)
W	Status Register

3870 Addressing Modes (Abbreviations)

IMM	Immediate Addressing
IMP	Implied Addressing
REL	Relative Addressing
EXT	Extended Addressing
SCR	Scratchpad Addressing (see below)
INM	Indirect Memory Addressing
IOP	I/O Port Addressing

Scratchpad Addressing Using IS (r not = 0 thru B)

r = C (Hex)	Register Addressed by IS (IS is unmodified)
r = D	Register Addressed by IS (IS is incremented)
r = E	Register Addressed by IS (IS is decremented)
r = F	Register Legal OP Code

9.0 PROGRAMMING EXAMPLES

9.1 INTRODUCTION

This section contains a number of programming examples which are useful in almost all applications. They can be used in programming any 3870 device, since all 3870 Family devices share a common instruction set. The programming examples are shown in the form of an assembly listing output produced by the Mostek MACRO-70 3870 macro cross assembler. Please refer to the MACRO-70 Operations Manual for explanation of the assembly listing output.

9.1.1 SCRATCHPAD OPERATIONS

As described in Section 8, the internal Scratchpad Register array may be thought of as an array which contains 8 rows of registers, where each row contains 8 registers. This is a convienient visualization of the Scratchpad since the IS register, which is used as an indirect address pointer to the Scratchpad, is split into two 3-bit halves. The upper half of the IS register (ISU) may be thought of as selecting a particular row of Scratchpad registers, while the lower half of the IS register may be thought of as selecting a column of Scratchpad registers.

A simple example which illustrates manipulating a row of Scratchpad registers is shown in the following sequence, which sets all 8 bytes in a row of Scratchpad registers to zero:

CLEAR REGISTER ROUTINE
Figure 9-1

```
                                F8/3870 MACRO CROSS ASSM. V2.2
LOC   OBJ.CODE    STMT-NR SOURCE-STMT  PASS2 INIT        INIT  INIT  REL
                  1                    NAME  INIT
                     * THIS ROUTINE CLEARS A ROW OF SCRATCHPAD REGISTERS
0000  70           3 START              CLR                     * CLEAR THE ACCUMULATOR
0001  62           4                    LISU  2                 * POINT TO ROW 2 (R16 - R23)
0002  6F           5                    LISL  7                 * POINT TO LAST REG IN ROW
0003  5E           6 LOOP               LR    D,A               * CLEAR AND DECREMENT
0004  8FFE         7                    BR7   LOOP              * LOOP UNTIL ISL = 7
```

For register locations with addresses greater than 0B Hex, the IS register must be used to perform any operation on those locations. In the routine called "INIT" shown above, the Accumulator is first cleared with the CLR instruction, and then the IS is loaded with the address of the first Scratchpad location to be cleared in the loop. Each half of the IS register is loaded with the individual instructions LISU and LISL. The LR D,A instruction loads the Scratchpad register with the contents of the Accumulator, and then automatically decrements the lower half of the IS register. The BR7 instruction will branch back to the program location signified by the label "LOOP" until the value of the lower half of the IS register again equals 7. Note that when this operation is complete, the value of the IS register will again equal 27 Octal since only ISL is auto-decremented.

9.1.2 DOUBLE PRECISION BINARY ADDITION

This example illustrates how two double precision, or sixteen bit, numbers can be added together in the 3870. The contents of register R0 and R1 are treated as a sixteen bit value with R0 representing the most significant eight bits and R1 representing the least significant eight bits. A similar value is contained in registers R2 and R3. Figure 9-2 shows the source listing for this routine.

DOUBLE PRECISION BINARY ADD ROUTINE
Figure 9-2

```
                                F8/3870 MACRO CROSS ASSM. V2.2
LOC OBJ.CODE    STMT-NR SOURCE-STMT PASS2 ADD                  ADD  ADD  REL
                1                   NAME     ADD
                * THIS ROUTINE ILLUSTRATES A DOUBLE PRECISION BINARY
                * ADDITION OPERATION. THE CONTENTS OF R0,R1 ARE ADDED
                * R2,R3 WITH THE RESULT IN R0,R1.
0000'43         5 START             LR       A,3     * GET LS BYTE OF R2,R3 (R3)
0001 C1         6                   AS       1       * ADD TO LS BYTE OF R0,R1
0002 51         7                   LR       1,A     * SAVE RESULT IN R1
0003 42         8                   LR       A,2     * GET MS BYTE OF R2,R3 (R2)
0004 19         9                   LNK              * ADD CARRY FROM LS BYTE ADD
0005 C0         10                  AS       0       * ADD TO LS BYTE OF R0,R1
0006 50         11                  LR       0,A     * SAVE RESULT IN R0
0007            12                  END
```

First, the least significant bytes (LS byte) of each number are added together by loading R3 into the Accumulator and adding it to R1. The result of this operation is then saved in R1. Note that the Carry flag represents any carry bit which may have propagated through bit 7 during this operation. The contents of R2 (MS byte) are loaded into the Accumulator, and the Carry flag resulting from the addition of the two least significant bytes is added to the Accumulator by using the LNK, or Add Carry, instruction. Note the instructions which simply move data between the Accumulator and the Scratchpad (LR 1,A and LR A,2) do not affect any of the bits in the Status Register (W). Finally, the value in the Accumulator, (R2 + Carry) is added to R0 and the result is placed in R0.

9.1.3 DOUBLE PRECISION BINARY NEGATE

Since there are no subtraction instructions in the 3870 instruction set, a two's complement subtraction operation must be accomplished by negating the minuend and performing a binary addition with the subtrahend. An 8 bit binary minuend may be negated by performing a one's complement operation and then adding the value of "1" to the complemented value. If the minuend was in the Accumulator this would be equivalent to performing a COM, INC instruction sequence. The following example illustrates how a double precision, or sixteen bit number, contained in Scratchpad RAM, may be negated. A double precision subtraction could then be performed by adding the negated value to the double precision subtrahend, using the double precision addition routine described above.

DOUBLE PRECISION NEGATE ROUTINE
Figure 9-3

```
                                F8/3870 MACRO CROSS ASSM. V2.2
LOC  OBJ.CODE   STMT-NR SOURCE-STMT PASS2 NEGD                 NEGD  NEGD  REL
                1                   NAME     NEGD
                * A DOUBLE PRECISION NEGATE OPERATION IS PERFORMED
                * ON THE CONTENTS OF THE SIXTEEN BIT VALUE CONTAINED
                * IN R0 AND R1.
          =0000' 5 NEGD             EQU      $
0000 40         6                   LR       A,0     * GET LS BYTE
0001 18         7                   COM              * NEGATE IT
0002 1F         8                   INC              * NEGATE IT
0003 50         9                   LR       0,A     * RESTORE LS BYTE NEGATED
0004 1E         10                  LR       J,W     * SAVE CARRY
0005 41         11                  LR       A,1     * GET MS BYTE
0006 18         12                  COM              * COMPLEMENT IT (CARRY <- 0)
0007 1D         13                  LR       W,J     * RESTORE CARRY
```

Figure 9-3 Cont.

```
0008 19          14          LNK             * ADD IT TO MS BYTE
0009 51          15          LR      1,A     * RESTORE MS BYTE NEGATED
000A             16          END
```

In this example, the double precision value is contained in registers R0 and R1. Register R1, the least significant byte is negated by using the COM, INC instruction sequence. The value of the Carry flag resulting from this operation must be saved so that it may be added to the complemented value of the most significant byte.

The Carry flag must be saved prior to the subsequent complement operation of the contents of R1 since the COM instruction clears the Carry flag in the Status Register W. In this example, the LR J,W instruction is used to save the contents of W in the J Linkage Register while the most significant byte is complemented. Once the complement operation is performed the Carry flag is restored in the Status Register by executing the LR W,J instruction, and it is added to the one's complement of the most significant byte with the LNK, or Add Carry, instruction.

9.1.4 SHIFT LEFT DOUBLE

This routine illustrates how a shift left of a double precision number in the Scratchpad RAM may be accomplished. The most significant bit of the sixteen bit value is shifted into the carry flag and a zero is shifted into the least significant bit. The assembly listing for this routine is given below:

SHIFT LEFT DOUBLE ROUTINE
Figure 9-4

```
                                F8/3870 MACRO CROSS ASSM. V2.2
LOC  OBJ.CODE       STMT-NR SOURCE-STMT PASS2 SHLD          SHLD  SHLD  REL
                    1                   NAME    SHLD
                    * A DOUBLE SHIFT LEFT OPERATION IS PERFORMED ON THE
                    * CONTENTS OF R6,R7. A ZERO IS SHIFTED INTO THE LSB A
                          ND
                    * THE MSB IS LEFT IN THE CARRY BIT.
                    * CARRY <- R6,R7 <- 0
                    *
             =0000' 7 SHLD              EQU     $
0000 46             8                   LR      A,7     * GET THE CONTENTS OF R6
0001 C6             9                   AS      7       * SHIFT LS BYTE BY ADDING
0002 56             10                  LR      7,A     * R6 <- RESULT
0003 47             11                  LR      A,6     * GET MS BYTE FROM R7
0004 19             12                  LNK             * IF CARRY SET ADD TO MS BYTE
0005 C7             13                  AS      6
0006 57             14                  LR      6,A     * R7 <- R7 + R7 + R6 MSB
0007                15                  END
```

In this operation, bit 7 of the least significant byte (R6) must be shifted into the least significant bit of R7. Therefore the Shift Left 1 (SL 1) instruction cannot be used since the most significant bit of the shifted value is lost. An alternative way is to add the value of R6 to itself, which effectively shifts the value of R6 to the left in the Accumulator, with the most significant bit resulting in the carry flag. Similarly, the most significant bit of R7 is to be shifted into the Carry flag. However, adding the value of R7 to itself would destroy the value of the Carry flag resulting from shifting R6 left by one. The Carry flag can be added to the value of R7 in the Accumulator and then the value of R7 can be added to the result. The net result is the same since the order of addition for R7 doesn't matter:

$$R6 + R6 + Carry = R6 + Carry + R7\ MSB$$

9.1.5 LOOP COUNTERS

There are several methods which can be used to create loop counters within the 3870. Usually the goal in using loop counters is to keep the overhead on the resources of the machine at a minimum. Three examples are discussed below which require little overhead in terms of program memory and execution time.

LOOP COUNTER ROUTINES
Figure 9-5

```
                                        F8/3870 MACRO CROSS ASSM. V2.2
LOC  OBJ.CODE       STMT-NR SOURCE-STMT PASS2 LOOP                  LOOP  LOOP  REL
                    1                   NAME      LOOP
                    *
                    * THIS ROUTINE DEMONSTRATES SOME
                    TECHNIQUES FOR
                    * HANDLING LOOP COUNTERS IN THE 3870.
                =0010 5 COUNT           EQU       10H
                =0000' 6 LOOP           EQU       $
                    *
                    * LOOP USING ISL
0000 6F             9                   LISL      7         * LOOP COUNT FOR EIGHT TIMES
                =0001' 10 LOOP1         EQU       $         * LOOP CODE CONTAINED HERE
                    *                                       * LOOP CODE CONTAINED HERE
                    *                                       * LOOP CODE CONTAINED HERE
0001 4E             13                  LR        A,D       * DECREMENT ISAR
0002 8FFE           14                  BR7       LOOP1     * IF ISL NOT = 7 CONT. LOOP
                    *
                    * LOOP USING 'DS' INSTRUCTION
                    * TRANSVERSE LOOP 'COUNT' TIMES
0004 2010           18                  LI        COUNT     * GET LOOP COUNT INITIAL VAL
0006 50             19                  LR        0,A       * INITIALZE COUNTER REGISTER
                =0007' 20 LOOP2         EQU       $         * LOOP CODE CONTAINED HERE
                    *                                       * LOOP CODE CONTAINED HERE
                    *                                       * LOOP CODE CONTAINED HERE
0007 30             23                  DS        0         * CONTINUE LOOPING
0008 94FE           24                  BNZ       LOOP2     * UNTIL R0 = 0.
                    *
                    * LOOP USING 'DS' INSTRUCTION
                    * TRANSVERSE LOOP 'COUNT + 1' TIMES
000A 2010           28                  LI        COUNT     * GET LOOP COUNT INITIAL VAL
000C 50             29                  LR        0,A       * INITIALIZE COUNTER REGISTER
                =000D' 30 LOOP3         EQU       $         * LOOP CODE CONTAINED HERE
                    *                                       * LOOP CODE CONTAINED HERE
                    *                                       * LOOP CODE CONTAINED HERE
000D 30             33                  DS        0         * CONTINUE LOOPING
000E 82FE           34                  BC        LOOP3     * UNTIL R0 = -1
0010                35                  END
```

For loop counters which require transversing a block of code no more than eight times, the lower half of the IS register provides a convienient loop counter. This is because of the auto-decrementing or auto-incrementing feature of the ISL register. In the part of the example which is labeled "LOOP1", the ISL register is loaded with an initial value of 7 with the single byte LISL instruction. ISL is decremented each time the loop is executed by the LR A,D instruction which does a dummy read of the Scratchpad RAM. When the loop has been transversed eight times, the ISL register rolls over to the value of 7 and the program will fall through the BR7 instruction. A restriction in using this approach is that the code contained within the loop cannot modify ISL.

"LOOP2" and "LOOP3" illustrate the implementation of loops which require executing a block of code more than eight times. These routines use a register in the Scratchpad RAM as a loop counter. "LOOP2" executes a block of code 'COUNT' number of times. The Decrement Scratchpad, or DS, instruction is used to decrement the loop counter register (R0) without the use of the Accumulator. "LOOP3" executes a block of code 'COUNT + 1' number of times since the DS instruction does essentially an add with 0FF Hex; hence the Carry flag is always set unless the loop counter register (R0) equals zero already.

9.1.6 SINGLE PRECISION MULTIPLICATION ROUTINE

The routine shown in Figure 9-7 performs a single precision (8 unsigned bit) binary multiplication operation. The contents of R0 and R1 are multiplied together and the sixteen bit result is placed in the register pair R6,R7. The general algorithm which is used is shown below:

1) Initialize R6 and R7 to 0 and set the loop count = 8.
2) Shift the partial product R6,R7 left by one.
3) Shift the multiplicand left by one (Carry <- MSB).
4) If the Carry flag = 1, then add the multiplier to the partial product R6,R7.
5) Decrement the loop counter and go to step 2 if not zero.

The effect of this algorithm can be visualized in Figure 9-6 shown below:

MULTIPLICATION ALGORITHM EXAMPLE
Figure 9-6

Decimal	Binary	
54	0 0 1 1 0 1 1 0	<- Multiplier
165	1 0 1 0 0 1 0 1	<- Multiplicand
8910	0 0 1 1 0 1 1 0	<- Partial Product #8
	0 0 0 0 0 0 0 0	<- Partial Product #7
	0 0 1 1 0 1 1 0	<- Partial Product #6
	0 0 0 0 0 0 0 0	<- Partial Product #5
	0 0 0 0 0 0 0 0	<- Partial Product #4
	0 0 1 1 0 1 1 0	<- Partial Product #3
	0 0 0 0 0 0 0 0	<- Partial Product #2
	0 0 1 1 0 1 1 0	<- Partial Product #1
	0 0 1 0 0 0 1 0 1 1 0 0 1 1 1 0	<- Product = 022CE Hex

Each partial product is calculated and added to the value of R6,R7. Since the double precision shift left of R6,R7 takes place prior to the addition of the next partial product, the partial products are calculated starting with the most significant bit of the multiplicand. Partial Product #1 as shown in Figure 9-6 is then effectively shifted to the left the proper number of times, or eight times.

The execution time of this routine is:

Maximum = 379 μS (Multiplicand = 0FFH)
Average = 331 μS (Multiplicand = 0AAH)

MULTIPLICATION ROUTINE
Figure 9-7

```
                                F8/3870 MACRO CROSS ASSM. V2.2
LOC  OBJ.CODE      STMT-NR SOURCE-STMT PASS2 MULT          MULT   MULT   REL
                   1                   NAME   MULT
                   * THIS ROUTINE PERFORMS A SINGLE PRECISION (8 BIT)
                   * MULTIPLICATION OF R0 AND R1 AND PLACES THE RESULT
                   * IN REGISTER PAIR R6,R7.
          =0000'   5 MULT              EQU    $
0000 6F            6                   LISL   7        * USE ISL AS A LOOP COUNTER
0001 70            7                   LIS    0        * INIT R6,R7 WITH 0
0002 56            8                   LR     6,A      * INIT R6, R7 WITH 0
0003 57            9                   LR     7,A      * INIT R6, R7 WITH 0
                   *
                   * SHIFT PRODUCT LEFT (R6,R7)
0004'47            12 MULT10           LR     A,7      * GET PRODUCT LS BYTE
0005 C7            13                  AS     7        * SHIFT LEFT* CARRY <- MSB
0006 57            14                  LR     7,A      * REPLACE R7 WITH SHIFT VAL
0007 46            15                  LR     A,6      * GET PRODUCT MS BYTE
0008 19            16                  LNK             * ADD CARRY FROM R7
0009 C6            17                  AS     6        * SHIFT LEFT
000A 56            18                  LR     6,A      * REPLACE R6 WITH SHIFT VAL
                   *
                   * SHIFT MULTIPLIER LEFT (R0)
000B 40            21                  LR     A,0      * GET MULTIPLIER
000C C0            22                  AS     0        * SHIFT MULTIPLIER LEFT
000D 50            23                  LR     0,A      * REPLACE R0 WITH SHIFT VALU
000E 9207          24                  BNC    MULT50   * IF NEXT MULTIPLIER BIT = 0
                   *                                    DO NOT ADD MULTIPLICAND
                   *
                   * ADD MULTIPLICAND TO PRODUCT
0010 41            28                  LR     A,1      * GET MULTIPLICAND
0011 C7            29                  AS     7        * ADD TO PARTIAL PRODUCT
0012 57            30                  LR     7,A      * UPDATE PARTIAL PROD LS BYTE
0013 46            31                  LR     A,6      * GET PARTIAL PRODUCT MS BYTE
0014 19            32                  LNK             * ADD CARRY FROM R1 + R7 -
0015 56            33                  LR     6,A      * UPDATE PARTIAL PROD MS BYTE
                   *
                   * CHECK FOR COMPLETION
0016'4E            36 MULT50           LR     A,D      * DECREMENT ISAR COUNTER
0017 8FEC          37                  BR7    MULT10   * REPEAT IF NOT DONE
0019               38                  END
```

9.1.7 MAGNITUDE COMPARISONS

By testing the appropriate status bit(s) of the MK3870's Status register, you can make magnitude comparisons without altering the contents of the device's Accumulator or memory.

The MK3870's instruction set provides two comparison instructions which do not store a result: CI (compare immediate) and CM (compare memory). These instructions add the two's complement value of the Accumulator to the immediate byte (CI) or to the memory byte (CM) referenced by the data counter. Although a comparison's result is discarded, the operation alters the status register according to the rules of two's complement addition.

For two numbers, A and B (signed or unsigned numbers), Table 9-1 indicates the status conditions necessary for each comparison. The routines shown in Figures 9-8 and 9-9 test for each condition and perform a branch if the relation is true. Although these routines use CI, CM can be substituted for memory comparison.

UNSIGNED MAGNITUDE COMPARISON EXAMPLES
Figure 9-8

```
                                F8/3870 MACRO CROSS ASSM. V2.2
LOC OBJ. CODE       STMT-NR SOURCE-STMT PASS2 UMAG          UMAG    UMAG    REL
                    1                   NAME    UMAG
                    * UNSIGNED MAGNITUDE COMPARISONS
                    * THESE EXAMPLES COMPARE THE CONTENTS OF THE
                    * ACCUMULATOR AGAINST A CONSTANT 'B', AND BRANCH
                    * ACCORDING TO THE STATUS SET BY THE RESULT
                    *
                =0080 7 B           EQU     80TH    * ASSIGN B TO SOME VALUE

                    *
                    * EXAMPLE (A) A = B
0000 2580           11              CI      B
0002 8405           12              BZ      EQUA    * Z FLAG SET IF A = B
                =0004' 13 NEQU      EQU     $       * ELSE NOT EQUAL
                    *
                    * EXAMPLE (B) A<->B
0004 2580           16              CI      B
0006 94FD           17              BNZ     NEQU    * Z FLAG CLEAR IF A<->B

                =0008' 19 EQUA      EQU     $       * ELSE EQUAL
                    *
                    * EXAMPLE (C)
0008 2580           22              CI      B
000A 9205           23              BNC     AGTB    * CARRY CLEAR IF A >B
                =000C' 24 ALEB      EQU     $       * ELSE A <= B
                    *
                    * EXAMPLE (D) A <= B
000C 2580           27              CI      B
000E 82FD           28              BC      ALEB    * CARRY SET IF A >= B
                =0010' 29 AGTB      EQU$    * ELSE A > B
                    *
                    * EXAMPLE (E) A < B
0010 2580           32              CI      B
0012 9203           33              BNC     AGEB    * CARRY CLEAR IF A >= B
0014 9405           34              BNZ     ALTB    * Z CLEAR AND C SET IF A < B
                =0016' 35 AGEB      EQU     $       * ELSE A > B
                    *
                    * EXAMPLE (F) A >= B
0016 92FF           38              BNC     AGEB    * CARRY CLEAR IF A > B
0018 84FD           39              BZ      AGEB    * Z SET AND C SET IF A = B
                =001A' 40 ALTB      EQU     $       * ELSE A < B
                    *
                    *
001A                43              END
```

SIGNED MAGNITUDE COMPARISON EXAMPLES
Figure 9-9

```
                                        F8/3870 MACRO CROSS ASSM. V2.2
LOC OBJ. CODE       STMT-NR SOURCE-STMT PASS2 SMAG          SMAG    SMAG    REL
                    1                   NAME    SMAG
                    * THIS SET OF EXAMPLES DEMONSTRATE THE USE
                    * OF THE STATUS FLAGS SET DURING THE EXECUTION
                    * OF THE CI INSTRUCTION IN DETERMINING
                    * MAGNITUDE COMPARISONS FOR SIGNED BINARY NUMBERS.
                    *
          =0080     7 B               EQU     80H     * ASSIGN B TO SOME VALUE
                    *
                    * EXAMPLE (A) A = B
                    *                 SAME AS EXAMPLE (A) IN UMAG
                    *
                    * EXAMPLE (B) A◄►B
                                      SAME AS EXAMPLE (B) IN UMAG
                    *
                    * EXAMPLE (C) A > B
0000 2580           16                CI      B       * COMPARE VALUES
0002 990D           17                BF      9,AGTB  * BRANCH IF OVF=S=0
004 9803            18                BNO     ALEB    * S=1 IF OVF=0
0006 8109           19                BP      AGTB    * BRANCH IF OVF=S=1
          =0008'    20 ALEB           EQU     $       * A <= B IF OVF =1, S=0
                    *
                    * EXAMPLE (D) A <= B
0008 2580           23                CI      B
000A 9905           24                BF      9,AGTB  * BRANCH IF OVF=S=0
000C 98FB           25                BNC     ALEB    * S=1 IF OVF=0
000E 91F9           26                BM      ALEB    * BRANCH IF S◄►0
          =0010'    27 AGTB           EQU     $
                    *
                    * EXAMPLE (E) A < B
0010 2580           30                CI      B
0012 9905           31                BF      9,AGEB  * BRANCH IF OVF=S=0
0014 9C0B           32                BF      12,ALTB * BRANCH IF OVF=Z=0
0016 9109           33                BM      ALTB    * BRANCH IF S=Z=0
          =0018'    34 AGEB           EQU     $       * CONTINUE HERE IF OVF=S OR
                    *                                 Z = 1
                    *
                    * EXAMPLE (F) A >= B
0018 2580           38                CI      B
001A 99FB           39                BF      9,AGEB  * BRANCH IF OVF=S=0
001C 9C03           40                BF      12,ALTB * BRANCH IF OVF=Z=0
001E 81F9           41                BP      AGEB    * BRANCH IF S=1
          =0020'    42 ALTB           EQU     $       * CONTINUE IF 0◄►S
                    *                                 AND Z=0
                    *
                    *
0020                46                END
```

STATUS BIT RELATIONS FOR VARIOUS CONDITIONS
Table 9-1

CONDITION	UNSIGNED					SIGNED				
		0	Z	C		S		0	Z	C	S
A = B		-	1	-	-		-	1	-	-	
A ◄► B		-	0	-	-		-	0	-	-	
A > B		-	-	0	-		0	-	-	0	
						or	1	-	-	1	
A <= B		-	-	1	-		0	-	-	1	
						or	1	-	-	0	
A < B		-		1	-		0	-	-	1	
						or	0	0	-	1	
A >= B		0	0	-	0		0	-	-	0	
	or	-	1	-	-	or	1	-	-	1	
						or	-	1	-	-	

9.1.8 ADDITIONAL PROGRAMMING EXAMPLES

For additional programming examples, the user is referred to the MACRO-70 Operations Manual (Publication No. MK79658). This document is the operations manual for the macro cross assembler program which runs on the Mostek MATRIX (TM) Development System under FLP-80DOS V2.2. A number of macro definitions are included in this manual which implement a higher level instruction set for the 3870. Listings of these definitions are included in this operations manual and can be used as examples for performing various programming operations within the 3870. Examples of these operations include: stack manipulation, rotates through carry, bit manipulation, logical OR with the scratchpad, and arithmetic shifts.

MOSTEK®

3870 SINGLE CHIP MICRO FAMILY

MK3870 and MK38P70

MK3870 FEATURES

- ☐ Available with 1K, 2K, 3K, or 4K bytes of mask programmable ROM memory
- ☐ 64 bytes scratchpad RAM
- ☐ Available with 64 bytes executable RAM
- ☐ 32 bits (4 ports) TTL compatible I/O
- ☐ Programmable binary timer
  Interval timer mode
  Pulse width measurement mode
  Event counter mode
- ☐ External interrupt input
- ☐ Crystal, LC, RC, or external time base options
- ☐ Low power (275 mW typ.)
- ☐ Single +5 volt supply

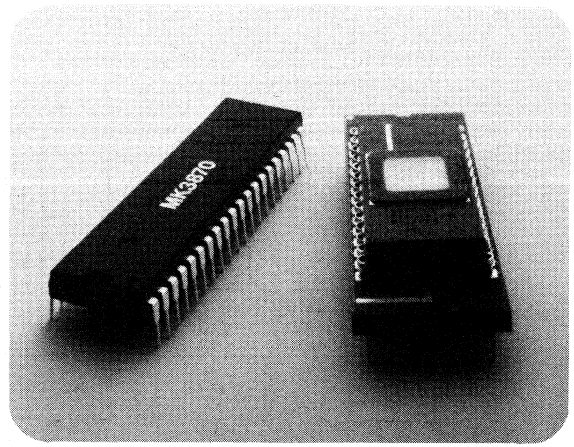
MK38P70 FEATURES

- ☐ EPROM version of MK3870
- ☐ Piggyback PROM (P-PROM)™ package
- ☐ Accepts 24 pin or 28 pin EPROM memories
- ☐ Identical pinout as MK3870
- ☐ In-Socket emulation of MK3870

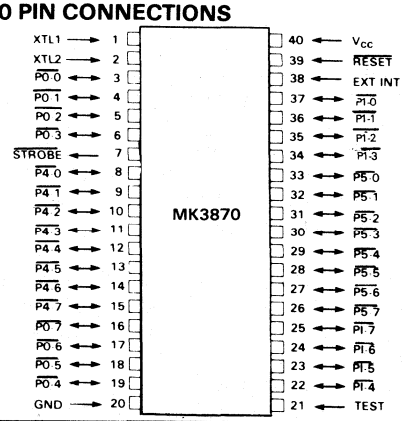
GENERAL DESCRIPTION

The MK3870 is a complete 8-bit microcomputer on a single MOS integrated circuit. The MK3870 can execute a set of more than 70 instructions, and is completely software compatible with the rest of the devices in the 3870 family. The MK3870 features 1-4K bytes of ROM and optional additional executable RAM depending on the specific part type designated by a slash number suffix. The MK3870 also features 64 bytes of scratchpad RAM, a programmable binary timer, and 32 bits of I/O.

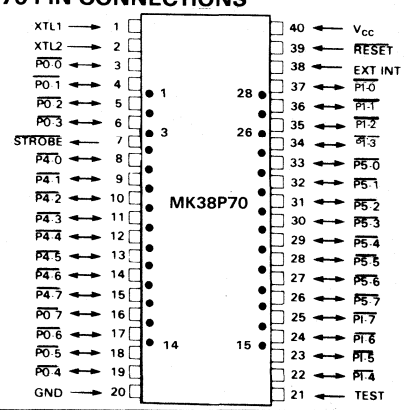
The programmable binary timer operates by itself in the interval timer mode or in conjunction with the external interrupt input in the pulse width measurement and the



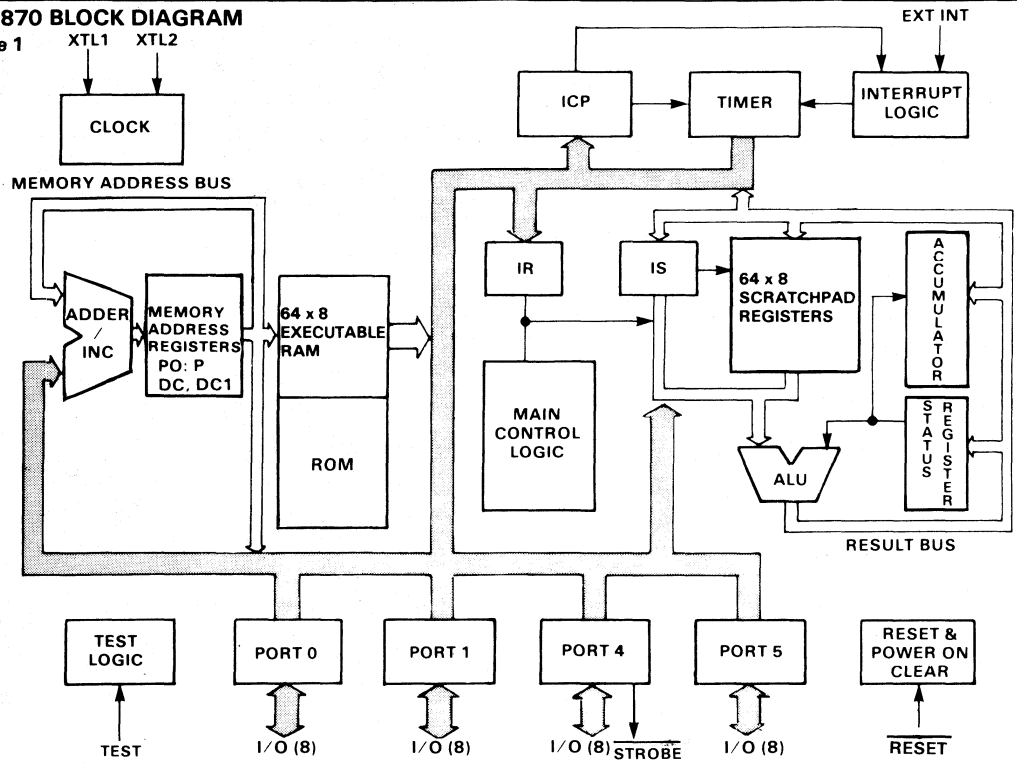
MK3870 PIN CONNECTIONS



MK38P70 PIN CONNECTIONS



MK3870 BLOCK DIAGRAM
Figure 1



event counter modes of operation. Two sources of vectored, prioritized interrupt are provided with the binary timer and the external interrupt input. The user has the option of specifying one of four clock sources for the MK3870 and MK38P70: Crystal, LC, RC, or external clock. In addition, the user can specify either a ±10% power supply tolerance or a ±5% power supply tolerance.

The MK38P70 microcomputer is the PROM based version of the MK3870. It is called the piggyback PROM (P-PROM)™ because of its packaging concept. This concept allows a standard 24-pin or 28 pin EPROM to be mounted directly on top of the microcomputer itself. The EPROM can be removed and reprogrammed as required with a standard PROM programmer. The MK38P70 retains exactly the same pinout and architectural features as other members of the 3870 family. The MK38P70 is discussed in more detail in a later section.

PIN NAME	DESCRIPTION	TYPE
$\overline{P0\text{-}0}$ -- $\overline{P0\text{-}7}$	I/O Port 0	Bidirectional
$\overline{P1\text{-}0}$ -- $\overline{P1\text{-}7}$	I/O Port 1	Bidirectional
$\overline{P4\text{-}0}$ -- $\overline{P4\text{-}7}$	I/O Port 4	Bidirectional
$\overline{P5\text{-}0}$ -- $\overline{P5\text{-}7}$	I/O Port 5	Bidirectional
$\overline{STROBE}$	Ready Strobe	Output
EXT INT	External Interrupt	Input
$\overline{RESET}$	External Reset	Input
TEST	Test Line	Input
XTL 1, XTL 2	Time Base	Input
V_{CC}, GND	Power Supply Lines	Input

FUNCTIONAL PIN DESCRIPTION

$\overline{P0\text{-}0}$-- $\overline{P0\text{-}7}$, $\overline{P1\text{-}0}$--$\overline{P1\text{-}7}$, $\overline{P4\text{-}0}$--$\overline{P4\text{-}7}$, and $\overline{P5\text{-}0}$--$\overline{P5\text{-}7}$ are 32 lines which can be individually used as either TTL compatible inputs or as latched outputs.

$\overline{STROBE}$ is a ready strobe associated with I/O Port 4. This pin, which is normally high, provides a single low pulse after valid data is present on the $\overline{P4\text{-}0}$--$\overline{P4\text{-}7}$ pins during an output instruction.

$\overline{RESET}$ may be used to externally reset the MK3870. When pulled low the MK3870 will reset. When then allowed to go high the MK3870 will begin program execution at program location H '000'.

EXT INT is the external interrupt input. Its active state is software programmable. This input is also used in conjunction with the timer for pulse width measurement and event counting.

XTL 1 and XTL 2 are the time base inputs to which a crystal, LC network, RC network, or an external single-phase clock may be connected. The time base network must be specified when ordering a mask ROM MK3870. The MK38P70 will operate with any of the four configurations.

TEST is an input, used only in testing the MK3870. For normal circuit functionality this pin may be left

unconnected, but it is recommended that TEST be grounded.

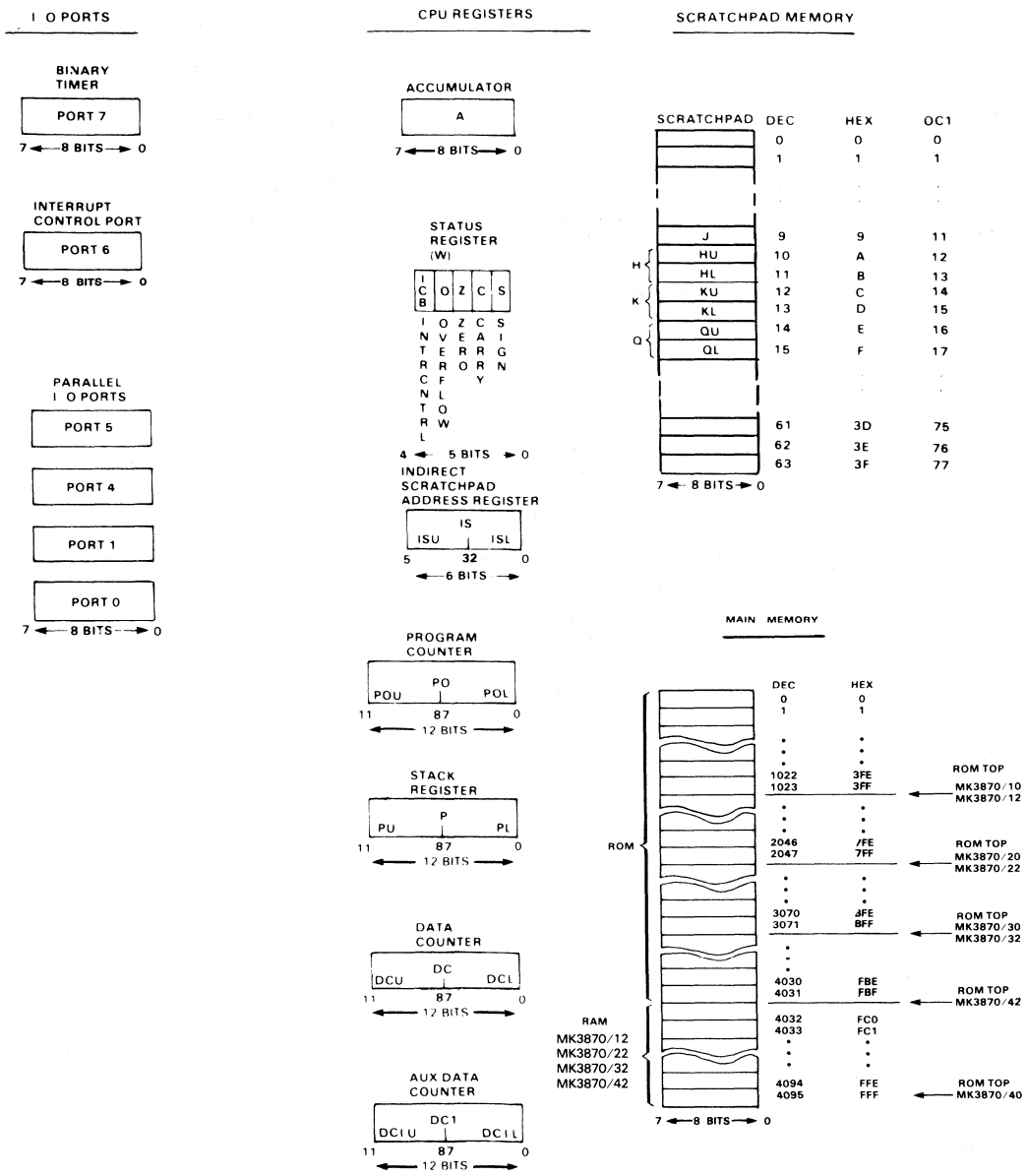
V_{CC} is the power supply input (single +5v).

MK3870 ARCHITECTURE

The basic functional elements of the MK3870 are shown in Figure 1. A programming model is shown in Figure 2. The architecture is common to all members of the 3870 family. All 3870 devices are instruction set compatible and differ only in amount and type of ROM, RAM, and I/O. The unique features of the MK3870 are discussed in the following sections. The user is referred to the 3870 Family Technical Manual for a thorough discussion of the architecture, instruction set, and other features which are common to all 3870 family devices.

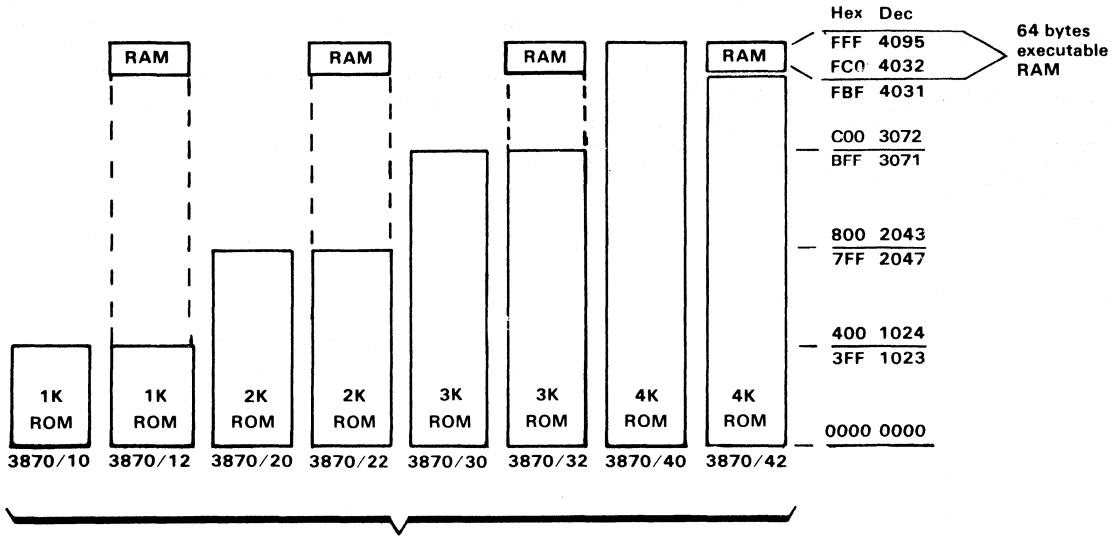
MK3870 PROGRAMMABLE REGISTERS, PORTS, AND MEMORY MAP

Figure 2



MK3870 MAIN MEMORY
SIZES AND TYPES BY SLASH NUMBERS

Figure 3



All devices contain 64 bytes of scratchpad RAM.

NOTE:

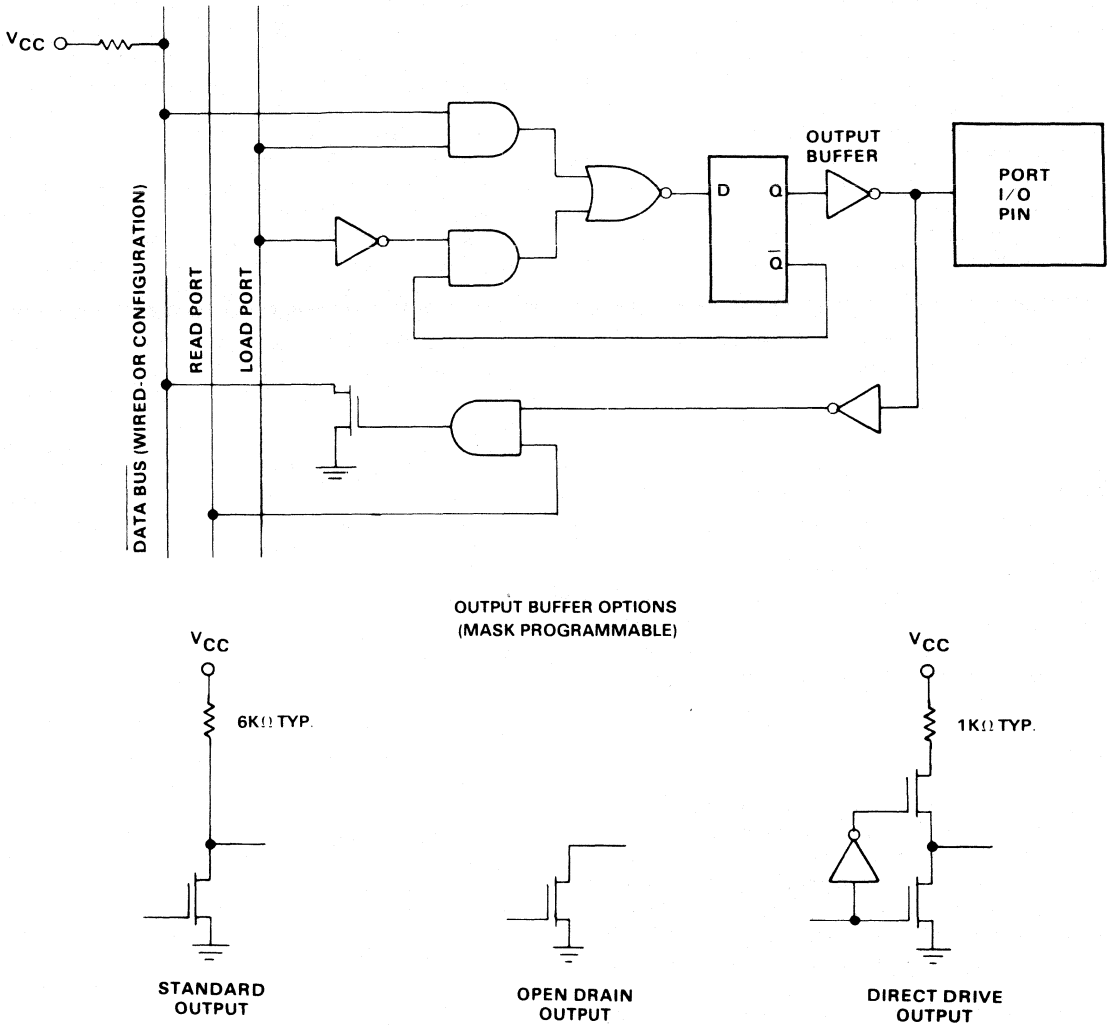
Data derived from addressing any locations other than those within a part's specified ROM space or RAM space (if any) is not tested nor is it guaranteed. Users should refrain from entering this area of the memory map.

Device	Scratchpad RAM Size (Decimal)	Address Register Size (P0, P, DC, DC1)	ROM Size (Decimal)	Executable RAM Size
MK3870/10	64 bytes	12 bits	1024 bytes	0 bytes
MK3870/12	64 bytes	12 bits	1024 bytes	64 bytes
MK3870/20*	64 bytes	12 bits	2048 bytes	0 bytes
MK3870/22	64 bytes	12 bits	2048 bytes	64 bytes
MK3870/30	64 bytes	12 bits	3072 bytes	0 bytes
MK3870/32	64 bytes	12 bits	3072 bytes	64 bytes
MK3870/40	64 bytes	12 bits	4096 bytes	0 bytes
MK3870/42	64 bytes	12 bits	4032 bytes	64 bytes

*The MK3870/20 is equivalent to the original 3870 device in memory size; however, the original 3870 had an 11-bit Address Register. The original 3870 with 11-bit Address Register is available where required. Consult the section describing ROM Code Ordering Information for additional information.

I/O PIN CONCEPTUAL DIAGRAM WITH OUTPUT BUFFER OPTIONS

Figure 4



Ports 0 and 1 are Standard Output type only.

Ports 4 and 5 may both be any of the three output options (mask programmable bit by bit)

The $\overline{\text{STROBE}}$ output is always configured similar to a Direct Drive Output except that it is capable of driving 3 TTL loads.

$\overline{\text{RESET}}$ and EXT INT may have standard 6KΩ (typical) pull-up or may have no pull-up (mask programmable).

MK3870 MAIN MEMORY

There are four address registers used to access main memory. These are the Program Counter (PO), the Stack Register (P), the Data Counter (DC), and the Auxiliary Data Counter (DC1). The Program Counter is used to address instructions or immediate operands. The Stack Register is used to save the contents of the Program Counter during an interrupt or subroutine call. Thus, the Stack Register contains the return address at which processing is to resume upon completion of the subroutine or interrupt routine. The Data Counter is used to address data tables. This register is auto-incrementing. Of the two data counters, only Data Counter (DC), can access the ROM. However, the XDC instruction allows the Data Counter and Auxiliary Data Counter to be exchanged.

The graph in Figure 3 shows the amounts of ROM and executable RAM for every available slash number in the MK3870 pin configuration.

EXECUTABLE RAM

The upper bytes of the address space in some of the MK3870 devices is RAM memory. As with the ROM memory, the RAM may be addressed by the PO and the DC address registers. The executable RAM may be addressed by all MK3870 instructions which address Main Memory. Additionally, the MK3870 may execute an instruction sequence which resides in the executable RAM. Note this cannot be done with the scratchpad RAM memory, which is the reason the term "executable RAM" is given to this additional memory.

I/O PORTS

The MK3870 provides four, 8 bit bidirectional Input/Output ports. These are ports 0, 1, 4, 5. In addition, the Interrupt Control Port is addressed as Port 6 and the binary timer is addressed as Port 7. The programming of Ports 6 and 7 and the bidirectional I/O pin are covered in the 3870 Family Technical Manual. The schematic of an I/O pin and available output drive options are shown in Figure 4.

An output ready strobe is associated with Port 4. This flag may be used to signal a peripheral device that the MK3870 has just completed an output of new data to Port 4. The strobe provides a single low pulse shortly after the output operation is completely finished, so either edge may be used to signal the peripheral. $\overline{\text{STROBE}}$ may also be used as an input strobe to Port 4 after completing the input operation.

MK38P70 GENERAL DESCRIPTION

The MK38P70 is the EPROM version of the MK3870. It retains an identical pinout with the MK3870, which is documented in the section of this data sheet entitled "FUNCTIONAL PIN DESCRIPTION". The MK38P70 is housed in the "R" package which incorporates a 28-pin socket located directly on top of the package. A number of standard EPROMs may be plugged into this socket.

The MK38P70 can act as an emulator for the purpose of exact verification of user code prior to the ordering of mask ROM MK3870 devices. Thus, the MK38P70 eliminates the need for emulator board products. In addition, several MK38P70s can be used in prototype systems in order to test design concepts in field service before committing to high-volume production with mask ROM MK3870s. The compact size of the MK38P70/EPROM combination allows the packaging of such prototype systems to be the same as that used in production. Finally, in low-volume applications, the MK38P70 can be used as the actual production device.

Most of the material which has been presented for the MK3870 in this document applies to the MK38P70. This includes the description of the pin configuration, architecture, programming model, and I/O ports. Additional information is presented in the following sections.

MK38P70 MAIN MEMORY

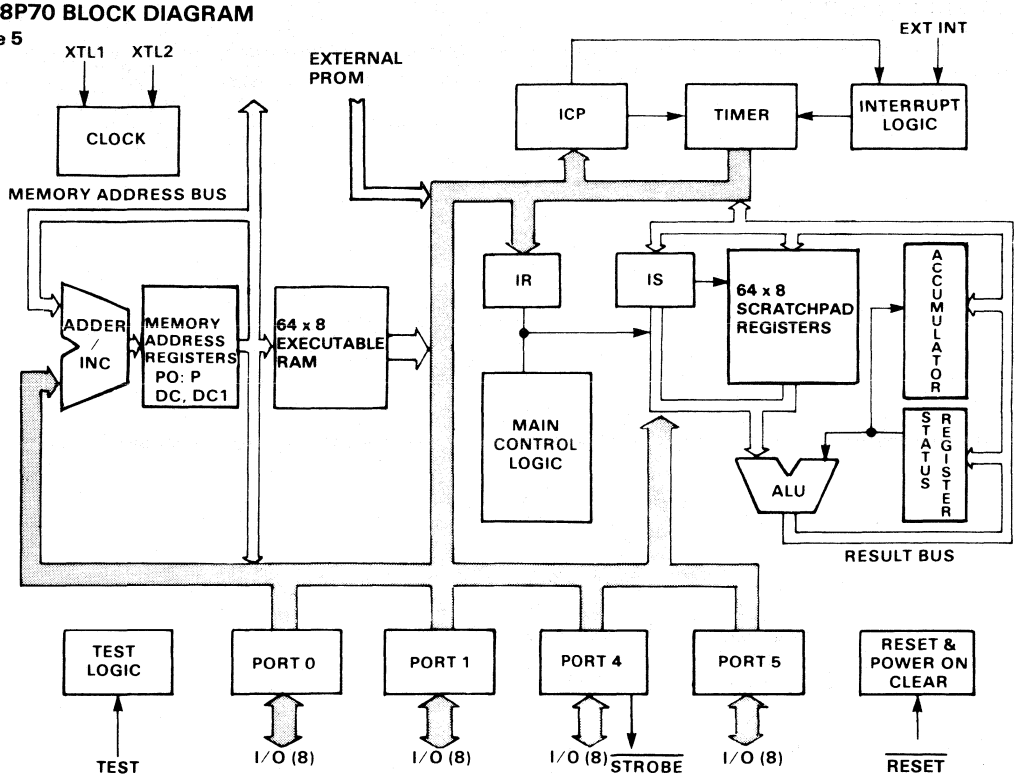
As can be seen from the block diagram in Figure 5, the MK38P70 contains executable RAM in the main memory map. The MK38P70 contains no on-chip ROM. Instead, the memory address lines are brought out to the 28 pin socket located directly on top of the 40 pin package, so the external EPROM memory is addressed as main memory.

There is one memory version of the MK38P70 and it is designated as the MK38P70/02. The MK38P70/02 contains 64 bytes of on-chip executable RAM. The MK38P70/02 can emulate the following devices.

MK3870/10
MK3870/12
MK3870/20
MK3870/22
MK3870/30
MK3870/32
MK3870/42

MK38P70 BLOCK DIAGRAM

Figure 5



The MK38P70/02 cannot exactly emulate the MK3870/40 because of the 64 bytes of executable RAM which are mapped into the upper 64 bytes of addressable main memory space. The MK3870/40 contains ROM memory in these locations.

Addressing of main memory on the MK38P70 is accomplished in the same way as it is for the MK3870. See Figure 6 for main memory addresses and for address register size in the MK38P70.

MK38P70 EPROM SOCKET

A 28 pin EPROM socket is located on top of the MK38P70 "R" package. The socket and compatible EPROM memories are shown in Figure 7. When 24 pin memories are used in the 28 pin socket, they should be inserted so that pin 1 of the memory device is plugged into pin 3 of the socket (the 24 pin memory should be lower justified in the 28-pin socket).

The 28-pin socket has been provided to allow use of both 24-pin and 28-pin memory devices. Minor pin-out differences in the memory devices must be accommodated by providing different versions of the MK38P70.

Initially, the MK38P70 that is compatible with the MK2716 is available. The MK38P70 designed to accomodate the 28-pin memory devices will be available at a later date.

MK38P70 I/O PORTS

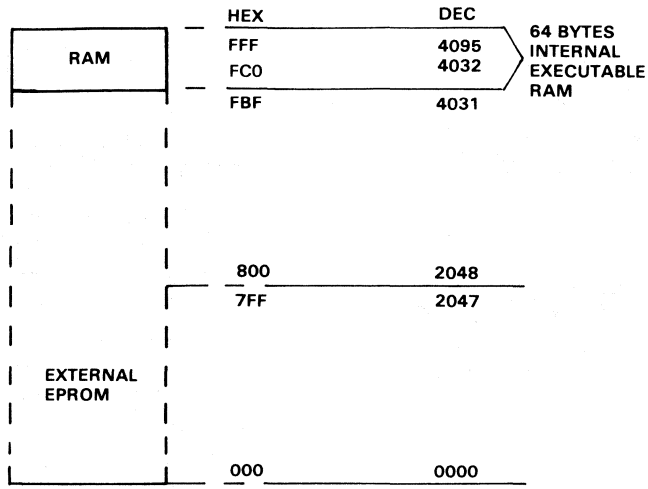
The MK38P70 is offered with two types of output buffer options on Ports 4 and 5. These are the open drain output buffer and the standard output buffer which are pictured in Figure 4. The open drain version of the MK38P70 is provided so that user-selected open drain port pins on the MK3870 can be emulated prior to ordering those mask ROM devices. Figure 7 lists which version(s) of the MK38P70 has open drain output buffers and which has standard output buffers in parentheses following the specified MK38P70 part ordering number (MK97XXX).

MEMORY ACCESS TIMING

A timing diagram depicting the memory access timing of the MK38P70 is shown in the next table. The Φ clock signal is derived internally in the MK38P70 by dividing the time base frequency by two and is used to establish all timing frequencies. The WRITE signal is another internal signal to the MK38P70 which corresponds to a machine cycle, during which time a memory access may be performed. Each machine cycle is either 4 Φ clock periods or 6 Φ clock periods long. These machine cycles are termed short cycles and long cycles, respectively. The worst case memory cycle is the short cycle, during which time an op code fetch is performed. This is the cycle which is pictured in the timing

MK38P70 MAIN MEMORY MAP

Figure 6



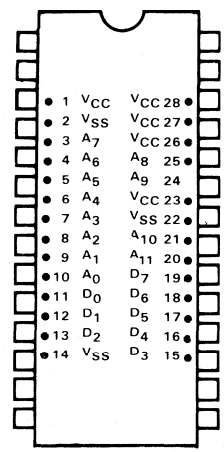
MK38P70/02

Device	Scratchpad RAM Size (Decimal)	Address Register Size (P0, P, DC, DC1)	ROM Size (Decimal)	Executable RAM Size
MK38P70/02	64 bytes	12 bits	0 bytes	64 bytes

diagram. After a delay from the falling edge of the WRITE clock, the address lines become stable. Data must be valid at the data out lines of the PROM for a setup time prior to the next falling edge of the WRITE pulse. The total access time available for the MK38P70 version is shown as t_{aas} or the time when address is stable until data must be valid on the data bus lines. The equation for calculating available memory access time along with some calculated access times based on the listed time base frequencies is shown in Figure 8.

MK38P70 "R" PACKAGE SOCKET PINOUT

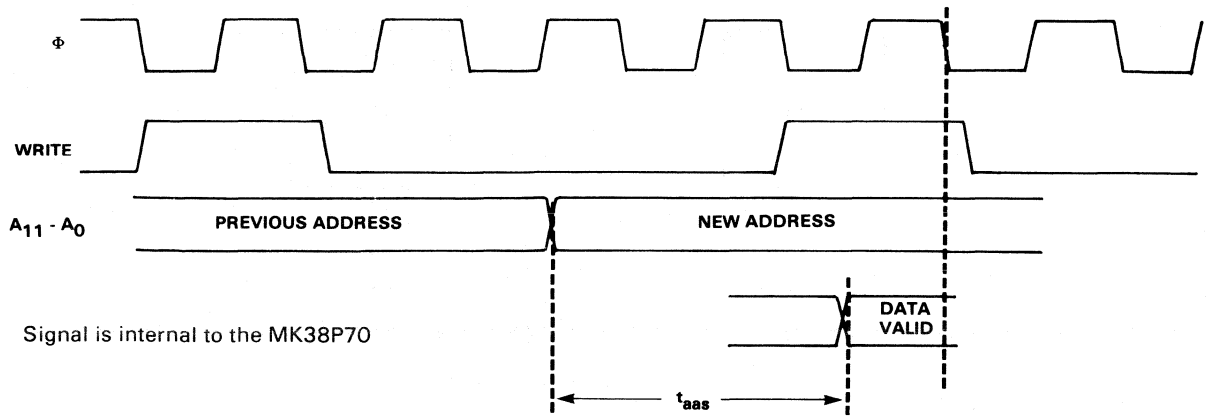
Figure 7



MK97400 (Standard Outputs) Compatible Memories	MK97410 (Open Drain) Compatible Memories
2758	2758
MK2716	MK2716
2516	2516
2532	2532

MEMORY ACCESS SHORT CYCLE OP CODE FETCH MK38P70

Figure 8



Signal is internal to the MK38P70

$$t_{aas} = \frac{6}{\text{time base freq.}} - 850ns$$

(FROM ADDRESS STABLE)

	4MHz	3.58MHz	3MHz	2.5MHz	2MHz
ACCESS TIME	650ns	825ns	1.15μs	1.55μs	2.15μs

3870 TIME BASE OPTIONS

The 3870 contains an on-chip oscillator circuit which provides an internal clock. The frequency of the oscillator circuit is set from the external time base network. The time base for the 3870 may originate from one of four sources:

1) Crystal
2) LC Network
3) RC Network
4) External Clock

The type of network which is to be used with the mask ROM MK3870 must be specified at the time when mask ROM devices are ordered. However, the MK38P70 may operate with any of the four configurations so that it may emulate any configuration used with a mask ROM device.

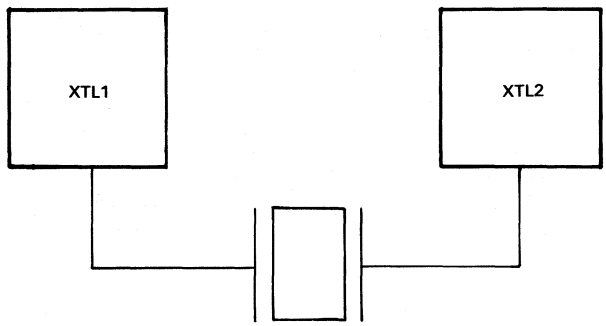
The specifications for the four configurations are given in the following text. There is an internal 26pF capacitor between XTL 1 and GND and an internal 26pF capacitor between XTL 2 and GND. Thus, external capacitors are not necessarily required. In all external clock modes the external time base frequently is divided by two to form the internal PHI clock.

CRYSTAL SELECTION

The use of a crystal as the time base is highly recommended as the frequency stability and reproducability from system to system is unsurpassed. The 3870 has an internal divide by two to allow the use of inexpensive and widely available TV Color Burst Crystals (3.58MHz). Figure 10 lists the required crystal parameters for use with the 3870. The Crystal Mode time base configuration is shown in Figure 9.

CRYSTAL MODE CONNECTION

Figure 9



CRYSTAL PARAMETERS

Figure 10

a) Parallel resonsnace, fundamental mode AT-Cut
b) Shunt capacitance (C_0) = 7 pf max.
c) Series resistance (R_S) = See table
d) Holder = See table below.

Frequency	Series Resistance	Holder
f = 2-2.7 MHz	Rs = 300 ohms max	HC-6 HC-33
f = 2.8-4 MHz	Rs = 150 ohms max	HC-6 HC-18* HC-25* HC-33

*This holder may not be available at frequencies near the lower end of this range.

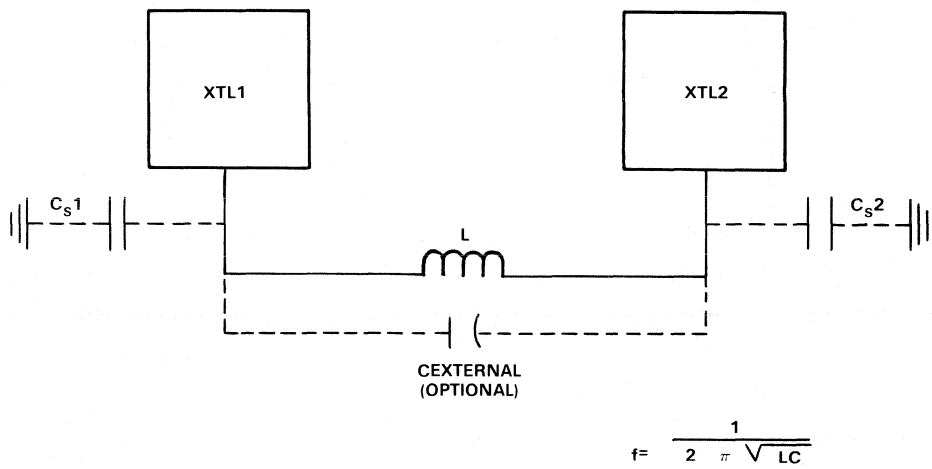
Through careful buffering of the XTL1 pin it may be possible to amplify this waveform and distribute it to other devices. However, Mostek recommends that a separate active device (such as a 7400 series TTL gate) be used to oscillate the crystal and the waveform from that oscillator be buffered and supplied to all devices, including the 3870, in the event that a single crystal is to provide the time base for more than just a single 3870.

While a ceramic resonator may work with the 3870 crystal oscillator, it was designed specifically to support the use of this component. Thus, Mostek does not support the use of a ceramic resonator either through proper testing, parametric specification, or applications support.

LC NETWORK

The LC time base configuration can be used to provide a less expensive time base for the 3870 than can be provided with a crystal. However, the LC configuration is much less accurate than is the crystal configuration. The LC time base configuration is shown in Figure 11. Also shown in the figure are the specified parameters for the LC components, along with the formula for calculating the resulting time base frequency. The minimum value of the inductor which is required for proper operation of the LC time base network is 0.1 millihenries. The inductor must have a Q factor which is no less than 40. The value of C is derived from C external, the internal capacitance of the 3870, C_{XTL}, and the stray

LC MODE CONNECTION
Figure 11



capacitances, C_{S1} and C_{S2}. C_{XTL} is the capacitance looking into the internal two port network at XTL1 and XTL2. C_{XTL} is listed under the "Capacitance" section of the Electrical Specifications. C_{S1} and C_{S2} are stray capacitances from XTL1 to ground and from XTL2 to ground, respectively. C external should also include the stray shunt capacitance across the inductor. This is typically in the 3 to 5 pf range and significant error can result if it is not included in the frequency calculation.

Variation in time base frequency with the LC network can arise from one of four sources: 1) Variation in the value of the inductor. 2) Variation in the value of the external capacitor. 3) Variation in the value of the internal capacitance of the 3870 at XTL1 and XTL2 and 4) Variation in the amount of stray capacitance which exists in the circuit. Therefore, the actual frequency which is generated by the LC circuit is within a range of possible frequencies, where the range of frequencies is determined by the worst case variation in circuit parameters. The designer must select component values such that the range of possible frequencies with the LC mode does not go outside of the specified operating frequency range for the 3870.

RC CLOCK CONFIGURATION

The time base for the 3870 may be provided from an RC network tied to the XTL2 pin, when XTL1 is grounded. A schematic picturing the RC clock configuration is shown in Figure 12. The RC time base configuration is intended to provide an inexpensive time base source for applications in which timing is not critical. Some users have elected to tune each unit using a variable resistor or external capacitor thus reducing the variation in frequency. However, for increased time base accuracy Mostek recommends the use of the Crystal or LC time base configuration. Figure 13 illustrates a curve which gives the resulting operating frequency for a particular RC value. The x-axis represents the product of the value of the resistor times the value of the capacitor. Note that three curves are actually shown. The curve in the middle represents the nominal frequency obtained for a given value of RC. A maximum curve and a minimum curve for different types of 3870 devices are also shown in the diagram.

The designer must select the RC product such that a frequency of less than 2 MHz is not possible taking into account the maximum possible RC product and using the minimum curve shown in Figure 13 below. Also, the RC product must not allow a frequency of more than 4 MHz taking into account the minimum possible R and C and using the Maximum curve shown below. Temperature induced variations in the external components should be considered in calculating the RC product.

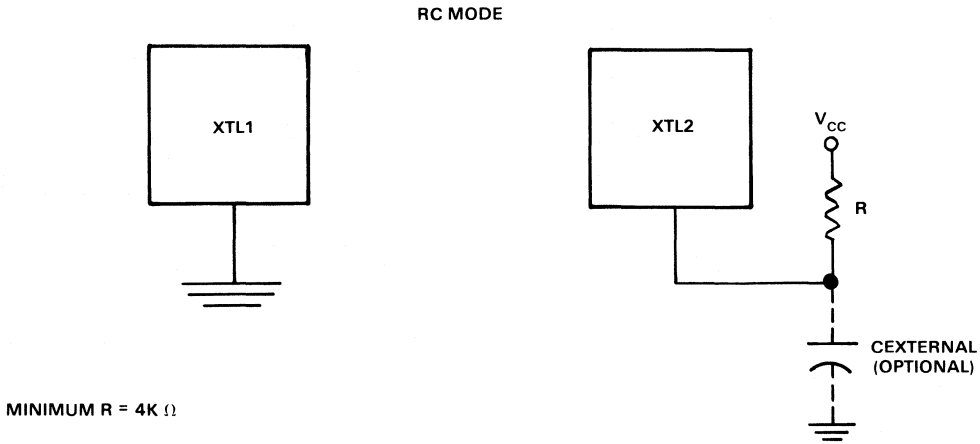
Frequency variation from unit to unit due to switching speed and level at constant temperature and V_{CC} = + or - 5 percent.

Frequency variation due to V_{CC} with all other parameters constant with respect to +5V = +7 percent to -4 percent on all devices.

Frequency variation due to temperature with respect to 25 C (all other parameters constant) is as follows:

PART #	VARIATION
387X-00, -05	+6 percent to - 9 percent
387X-10, -15	+9 percent to -12 percent

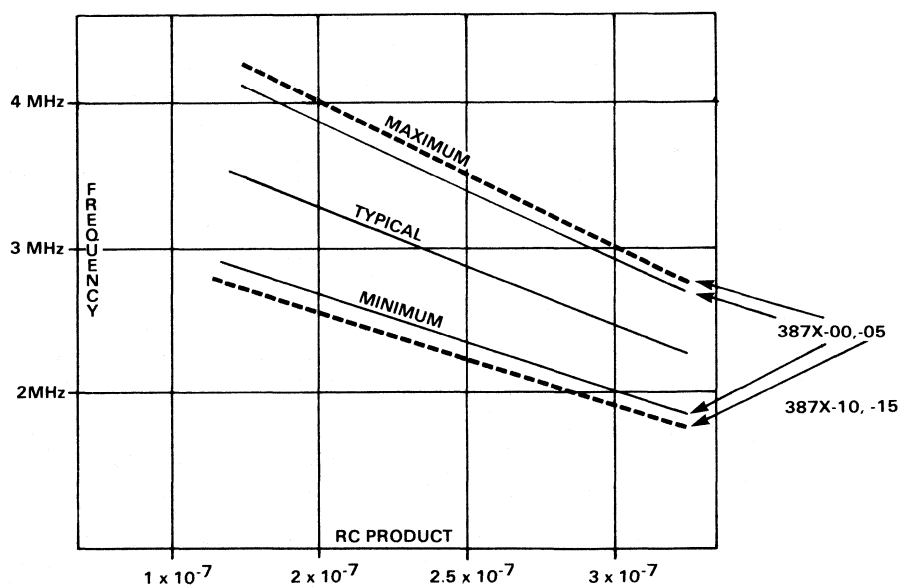
RC MODE CONNECTION
Figure 12



MINIMUM R = 4K Ω

C = 26.5 pF ± 2.6pF + Cexternal

FREQUENCY VS. RC
Figure 13



Variations in frequency due to variations in RC components may be calculated as follows:

Maximum RC = (R max) (C external max + C_{XTL} max)

Minimum RC = (R min) (C external min + C_{XTL} min)

Typical RC = (R typ) (C external typ + $\frac{\{C_{XTL} \text{ max} + C_{XTL} \text{ min}\}}{2}$)

Positive Freq. Variation = $\frac{\text{RC typical - RC minimum}}{\text{RC typical}}$

Negative Freq. Variation due to RC Components = $\frac{\text{RC maximum - RC typical}}{\text{RC typical}}$

Total frequency variation due to all factors:

387X-00, -05
= +18 percent plus positive frequency variation due to RC components

387X-10, -15
= +21 percent plus positive frequency variation due to RC components

= -18 percent minus negative frequency variation due to RC components

= -21 percent minus negative frequency variation due to RC components

Total frequency variation due to V_{CC} and temperature of a unit tuned to frequency at +5V V_{CC}, 25 C

387X-00, -05
= + 13 percent

387X-10, -15
= + 16 percent

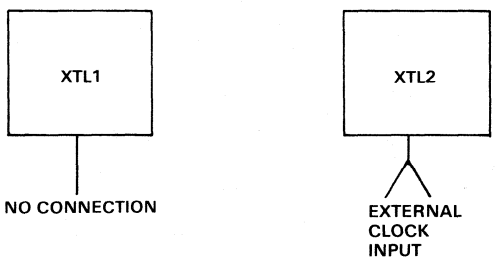
EXTERNAL CLOCK CONFIGURATION

The connection for the external clock time base configuration is shown in Figure 14. Refer to the DC Characteristics section for proper input levels and current requirements.

Refer to the Capacitance section of the appropriate 3870 Family device data sheet for input capacitance.

EXTERNAL MODE CONNECTION

Figure 14



OPERATING VOLTAGES AND TEMPERATURES

Dash Number Suffix	Operating Voltage V_{CC}	Operating Temprature T_A
— 00	+5V ± 10%	0°C - 70°C
— 05	+5V ± 5%	0°C - 70°C
— 10	+5V ± 10%	-40°C - +85°C
— 15	+5V ± 5%	-40°C - +85°C

See Ordering Information for explanation of part numbers.

ABSOLUTE MAXIMUM RATINGS*

	-00, -05	-10, -15
Temperature Under Bias	-20°C to +85°C	-50°C to +100°C
Storage Temperature	-65°C to +150°C	-65°C to +150°C
Voltage on any Pin With Respect to Ground (Except open drain pins and TEST)	-1.0V to +7V	-1.0V to +7V
Voltage on TEST with Respect to Ground	-1.0V to +9V	-1.0V to +9V
Voltage on Open Drain Pins With Respect to Ground	-1.0V to +13.5V	-1.0V to +13.5V
Power Dissipation	1.5W	1.5W
Power Dissipation by any one I/O pin	60mW	60mW
Power Dissipation by all I/O pins	600mW	600mW

*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.

AC CHARACTERISTICS

T_A, V_{CC} within specified operating range.
I/O power dissipation ≤ 100mW (Note 2)

SIGNAL	SYM	PARAMETER	-00,-05 MIN	-00,-05 MAX	-10,-15 MIN	-10,-15 MAX	UNIT	NOTES
XTL1 XTL2	t_o	Time Base Period, all clock modes	250	500	250	500	ns	4MHz-2MHz
	$t_{ex(H)}$	External clock pulse width high	90	400	100	390	ns	
	$t_{ex(L)}$	External clock pulse width low	100	400	110	390	ns	
Φ	t_Φ	Internal Φ clock	$2t_O$		$2t_O$			
WRITE	t_W	Internal WRITE Clock period	4tΦ 6tΦ		4tΦ 6tΦ			Short Cycle Long Cycle
I/O	$t_{dI/O}$	Output delay from internal WRITE clock	0	1000	0	1200	ns	50pF plus one TTL load
	$t_{sI/O}$	Input setup time to internal WRITE clock	1000		1200		ns	
$\overline{STROBE}$	$t_{I/O-s}$	Output valid to $\overline{STROBE}$ delay	3tΦ -1000	3tΦ +250	3tΦ -1200	3tΦ +300	ns	I/O load = 50pF + 1 TTL load
	t_{sL}	$\overline{STROBE}$ low time	8tΦ -250	12tΦ +250	8tΦ -300	12tΦ +300	ns	$\overline{STROBE}$ load= 50pF + 3TTL loads
$\overline{RESET}$	t_{RH}	$\overline{RESET}$ hold time, low	6tΦ +750		6tΦ +1000		ns	
	t_{RPOC}	$\overline{RESET}$ hold time, low for power clear	power supply rise time · 0.1		power supply rise time · 1.0		ms	
EXT INT	t_{EH}	EXT INT hold time in active and inactive state	6tΦ +750		6tΦ +1000		ns	To trigger interrupt
			2tΦ		2tΦ		ns	To trigger timer

AC CHARACTERISTICS FOR MK38P70

(Signals brought out at socket)
T_A, V_{CC} within specified operating range.
I/O Power Dissipation ≤ 100 mW. (Note 2)

SYMBOL	PARAMETER	-00, -05		-10, -15		UNIT	CONDITION
		MIN	MAX	MIN	MAX		
t_{aas}*	Access time from Address A_{11} - A_0[1] stable until data must be valid at D_7-D_0	650		650		ns	Φ = 2.0 MHz

*See Table in Figure 8.

CAPACITANCE

T_A = 25°C
All Part Numbers

SYM	PARAMETER	MIN	MAX	UNIT	NOTES
C_{IN}	Input capacitance		10	pF	Unmeasured Pins Grounded
C_{XTL}	Input capacitance; XTL1, XTL2	23.5	29.5	pF	

DC CHARACTERISTICS

T_A, V_{CC} within specified operating range
I/O power dissipation ≤ 100mW (Note 2)

SYMBOL	PARAMETER	-00, -05		-10, -15		UNIT	DEVICE
		MIN	MAX	MIN	MAX		
I_{CC}	Average Power Supply Current		85		110	mA	MK3870/10 Outputs Open
			94		125	mA	MK3870/12 Outputs Open
			85		110	mA	MK3870/20 Outputs Open
			94		125	mA	MK3870/22 Outputs Open
			100		130	mA	MK3870/30 Outputs Open
			100		130	mA	MK3870/32 Outputs Open
			100		130	mA	MK3870/40 Outputs Open
			100		130	mA	MK3870/42 Outputs Open

DC CHARACTERISTICS (cont.)

SYMBOL	PARAMETER	-00, -05		-10, -15		UNIT	DEVICE
		MIN	MAX	MIN	MAX		
I_{CC}	Average Power Supply Current		125		150	mA	MK38P70/02 No EPROM, Outputs Open
P_D	Power Dissipation		400		525	mW	MK3870/10 Outputs Open
			440		575	mW	MK3870/12 Outputs Open
			400		525	mW	MK3870/20 Outputs Open
			440		575	mW	MK3870/22 Outputs Open
			475		620	mW	MK3870/30 Outputs Open
			475		620	mW	MK3870/32 Outputs Open
			475		620	mW	MK3870/40 Outputs Open
			475		620	mW	MK3870/42 Outputs Open
			600		750	mW	MK38P70/02 No EPROM, Outputs Open

DC CHARACTERISTICS (cont.)

T_A, V_{CC} within specified operating range, I/O power dissipation ≤ 100mW (Note 2)

SYM	PARAMETER	-00,-05		-10,-15		UNIT	CONDITIONS
		MIN	MAX	MIN	MAX		
V_{IHEX}	External Clock input high level	2.4	5.8	2.4	5.8	V	
V_{ILEX}	External Clock input low level	-.3	.6	-.3	.6	V	
I_{IHEX}	External Clock input high current		100		130	μA	$V_{IHEX}=V_{CC}$
I_{ILEX}	External Clock input low current		-100		-130	μA	$V_{ILEX}=V_{SS}$
$V_{IHI/O}$	Input high level, I/O pins	2.0	5.8	2.0	5.8	V	Standard pull-up
		2.0	13.2	2.0	13.2	V	Open drain (1)
V_{IHR}	Input high level, $\overline{RESET}$	2.0	5.8	2.2	5.8	V	Standard pull-up
		2.0	13.2	2.2	13.2	V	No Pull-up
V_{IHEI}	Input high level, EXT INT	2.0	5.8	2.2	5.8	V	Standard pull-up
		2.0	13.2	2.2	13.2	V	No Pull-up
V_{IL}	Input low level	-.3	.8	-.3	.7	V	(1)
I_{IL}	Input low current, all pins with standard pull-up resistor		-1.6		-1.9	mA	$V_{IN}=0.4V$
I_L	Input leakage current, open drain pins, and inputs with no pull-up resistor		+10 -5		+18 -8	μA μA	$V_{IN}=13.2V$ $V_{IN}=0.0V$
I_{OH}	Output high current pins with standard pull-up resistor	-100 -30		-89 -25		μA μA	$V_{OH}=2.4V$ $V_{OH}=3.9V$
I_{OHDD}	Output high current, direct drive pins	-100 -1.5 -8.5		-80 -1.3 -11		μA mA mA	$V_{OH}=2.4V$ $V_{OH}=1.5V$ $V_{OH}=0.7V$
I_{OHS}	$\overline{STROBE}$ Output High current	-300		-270		μA	V_{OH} = 2.4V
I_{OL}	Output low current	1.8		1.65		mA	$V_{OL}=0.4V$
I_{OLS}	$\overline{STROBE}$ Output Low current	5.0		4.5		mA	$V_{OL}=0.4V$

DC CHARACTERISTICS FOR MK38P70

(Signals brought out at socket)

T_A, V_{CC} within specified operating range, I/O power dissipation ≤ 100mW (Note 2)

SYM	PARAMETER	-00, -05		-10, -15		UNIT	CONDITION
		MIN	MAX	MIN	MAX		
I_{CCE}	Power Supply Current for EPROM		-185		-185	mA	
V_{IL}	Input Low Level Data bus in	-0.3	0.8	-0.3	0.8	V	
V_{IH}	Input High Level Data bus in	2.0	5.8	2.0	5.8	V	
I_{OH}	Output High Current	-100		-90		μA	V_{OH}=2.4V
		-30		-25		μA	V_{OH}=3.9V
I_{OL}	Output Low Current	1.8		1.65		mA	V_{OL}=0.4V
I_{IL}	Input Leakage Current		10		10	μA	Data Bus in Float

1. $\overline{\text{RESET}}$ and EXT INT have internal Schmit triggers giving minimum .2V hysteresis.
2. Power dissipation for I/O pins is calculated by $\Sigma (V_{CC} - V_{IL})(|I_{IL}|) = \Sigma (V_{CC} - V_{OH})(|I_{OH}|) = \Sigma (V_{OL})(I_{OL})$

TIMER AC CHARACTERISTICS

Definitions:
Error = Indicated time value - actual time value

tpsc = tΦ x Prescale Value

Interval Timer Mode:

Single interval error, free running (Note 3) ..... ±6tΦ
Cumulative interval error, free running (Note 3) 0
Error between two Timer reads (Note 2) ..... ± (tpsc + tΦ)
Start Timer to stop Timer error (Notes 1, 4) ..... + tΦ to - (tpsc + tΦ)
Start Timer to read Timer error (Notes 1, 2) ..... -5tΦ to - (tpsc + 7tΦ)
Start Timer to interrupt request error (Notes 1, 3) ..... -2tΦ to -8tΦ
Load Timer to stop Timer error (Note 1) ..... + tΦ to - (tpsc + 2tΦ)
Load Timer to read Timer error (Notes 1, 2) ..... -5tΦ ± to - (tpsc + 8tΦ)
Load Timer to interrupt request error (Notes 1, 3) ..... -2tΦ to -9tΦ

Pulse Width Measurement Mode:

Measurement accuracy (Note 4) ..... + tΦ to -(tpsc + 2tΦ)
Minimum pulse width of EXT INT pin 2tΦ

Event Counter Mode:

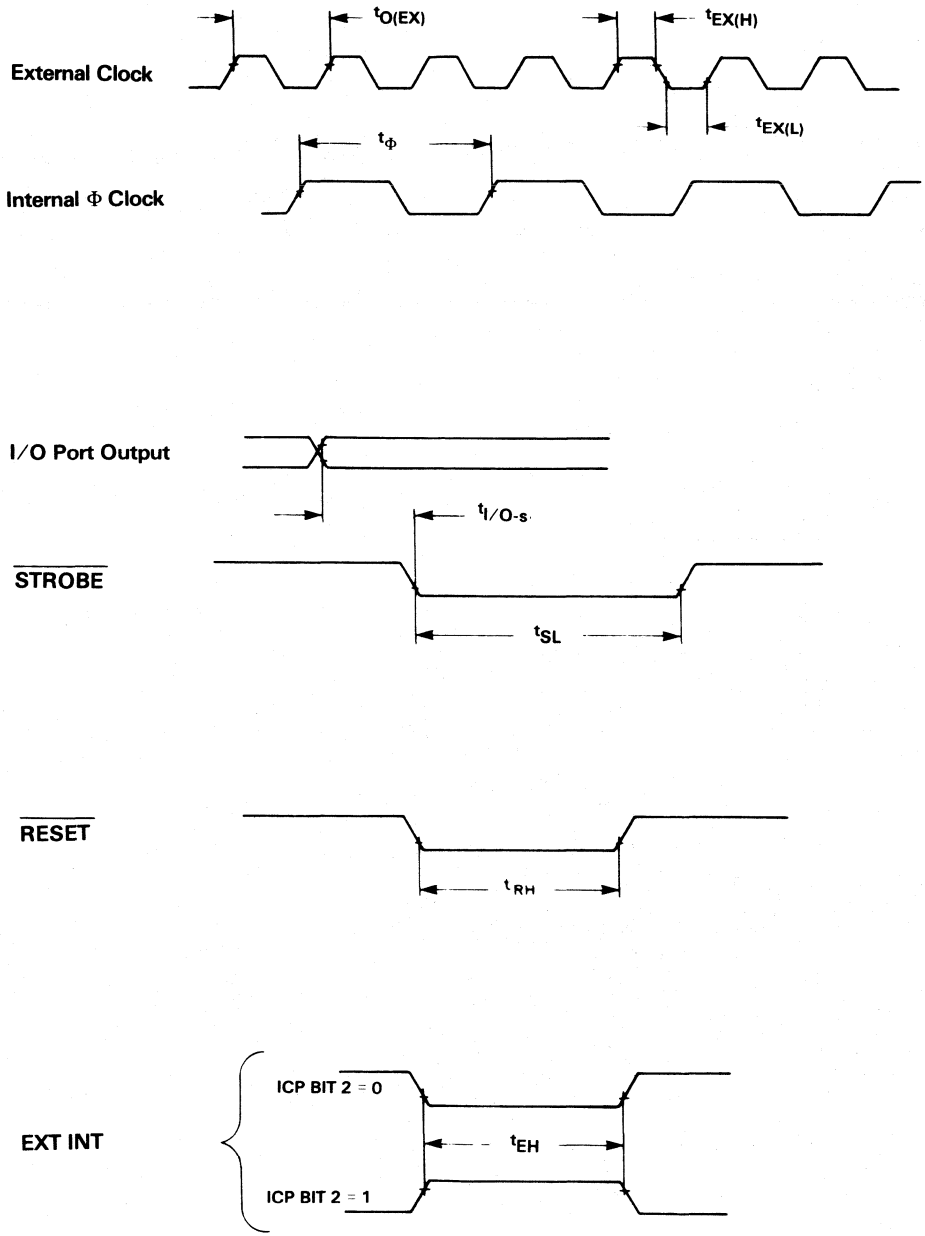
Minimum active time of EXT INT pin 2tΦ
Minimum inactive time of EXT INT pin 2tΦ

Notes:

1. All times which entail loading, starting, or stopping the Timer are referenced from the end of the last machine cycle of the OUT or OUTS instruction.
2. All times which entail reading the Timer are referenced from the end of the last machine cycle of the IN or INS instruction.
3. All times which entail the generation of an interrupt request are referenced from the start of the machine cycle in which the appropriate interrupt request latch is set. Additional time may elapse if the interrupt request occurs during a privileged or multicycle instruction.
4. Error may be cumulative if operation is repetitively performed.

AC TIMING DIAGRAM

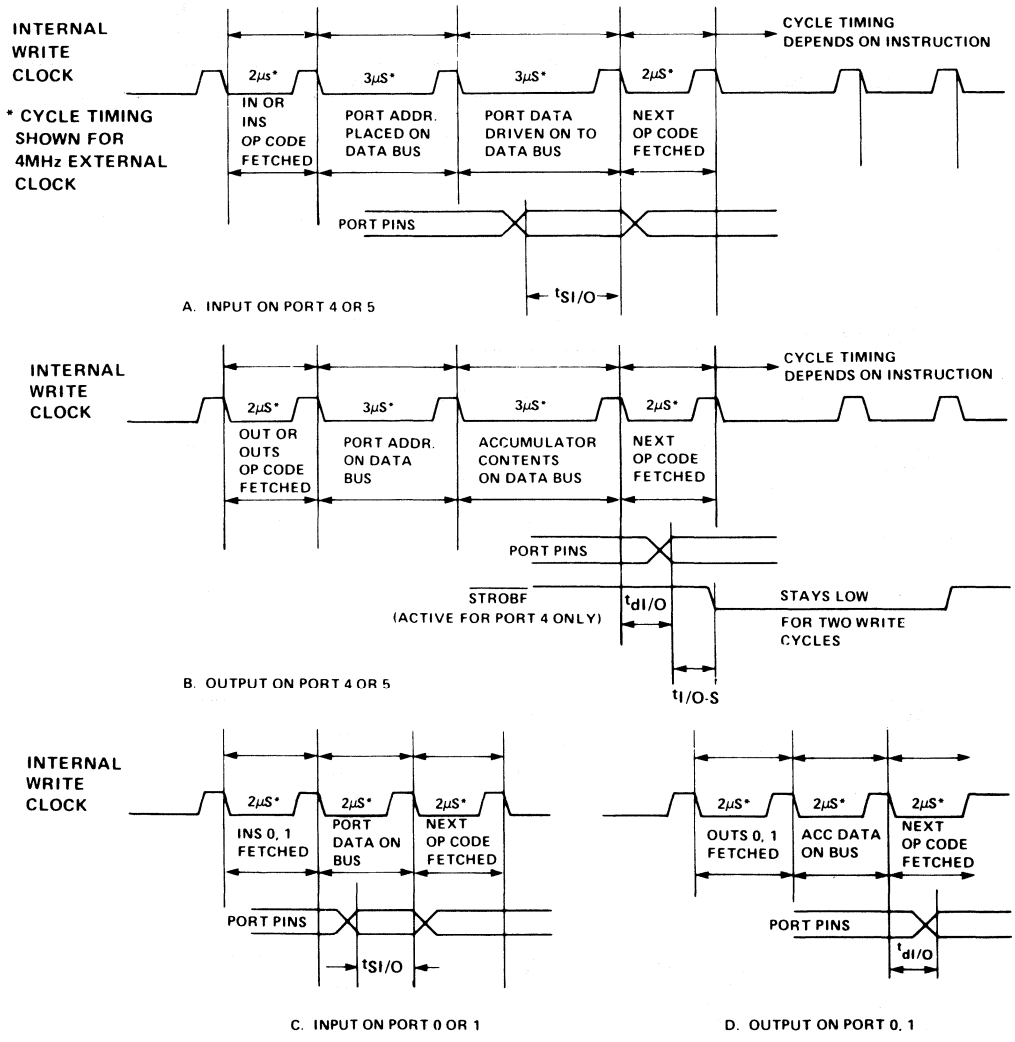
Figure 15



Note: All AC measurements are referenced to V_{IL} max., V_{IH} min., V_{OL} (.8v), or V_{OH} (2.0v).

INPUT/OUTPUT AC TIMING

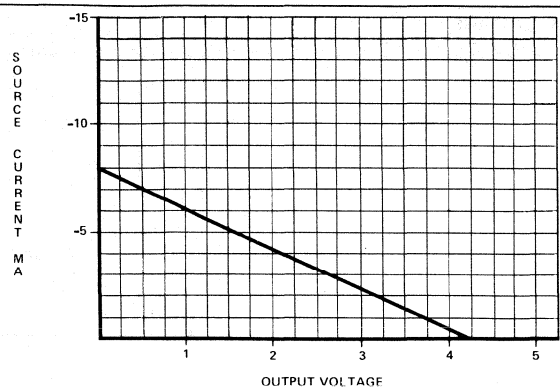
Figure 16



STROBE SOURCE CAPABILITY

(TYPICAL AT V_{CC} = 5V, T_A = 25°C)

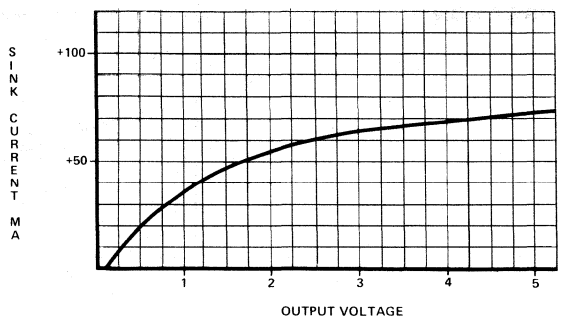
Figure 17



STROBE SINK CAPABILITY

(TYPICAL AT V_{CC} = 5V, T_A = 25°C)

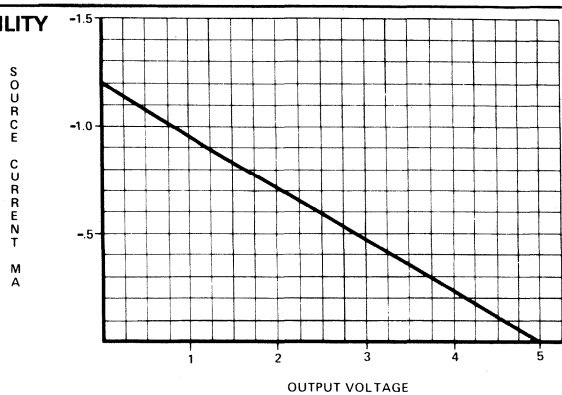
Figure 18



STANDARD I/O PORT SOURCE CAPABILITY

(TYPICAL AT V_{CC} = 5V, T_A = 25°C)

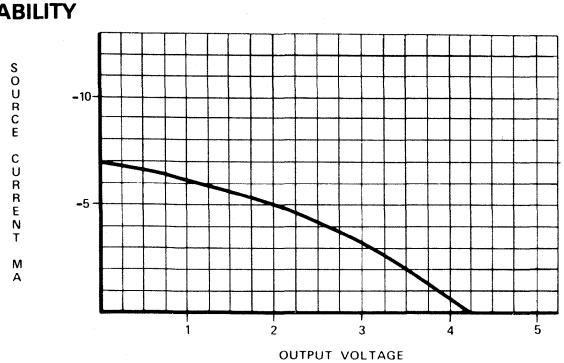
Figure 19



DIRECT DRIVE I/O PORT SOURCE CAPABILITY

(TYPICAL AT V_{CC} = 5V, T_A = 25°C)

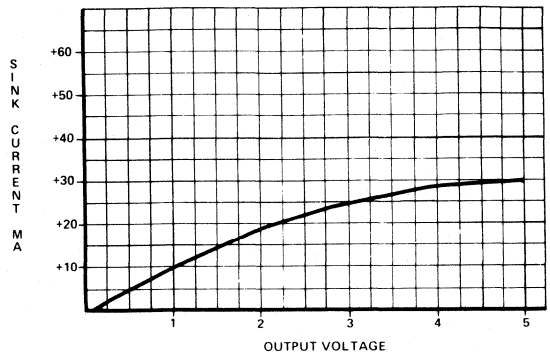
Figure 20



I/O PORT SINK CAPABILITY

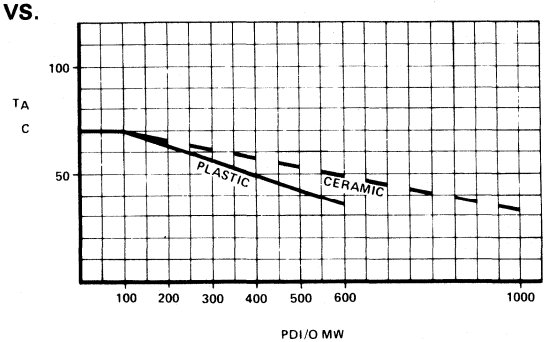
(TYPICAL AT V_{CC} = 5V, T_A = 25°C)

Figure 21



MAXIMUM OPERATING TEMPERATURE VS. I/O POWER DISSIPATION

Figure 22

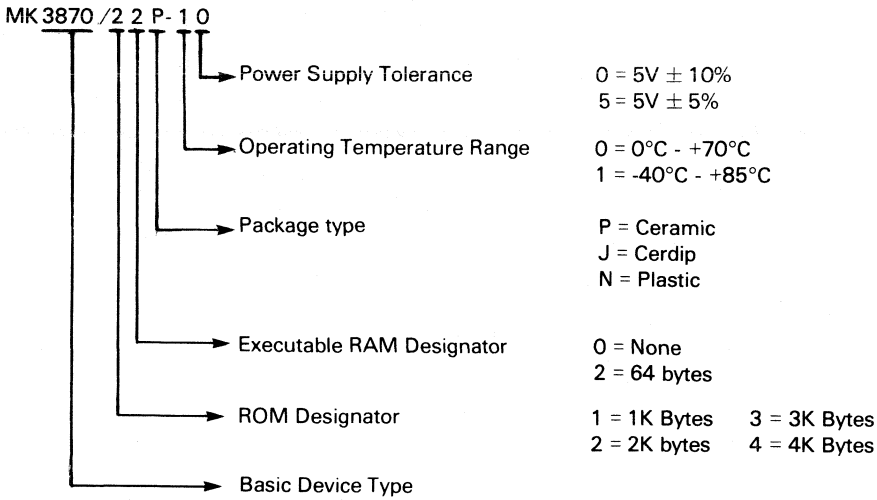


ORDERING INFORMATION

There are two types of part numbers for the 3870 family of devices. The generic part number describes the basic device type, the amount of ROM and Executable RAM, the desired package type, temperature range, and power supply tolerance. For each customer specific code, additional information defining I/O options and oscillator options will be combined with the information described in the generic part number to define a customer/code specific device order number. Note: the specific device order number will be used to differentiate between the MK3870/20 with 12-bit Address Registers and the original 3870 with 11-bit Address Register, as mentioned in an earlier section.

GENERIC PART NUMBER

An example of the generic part number is shown below.

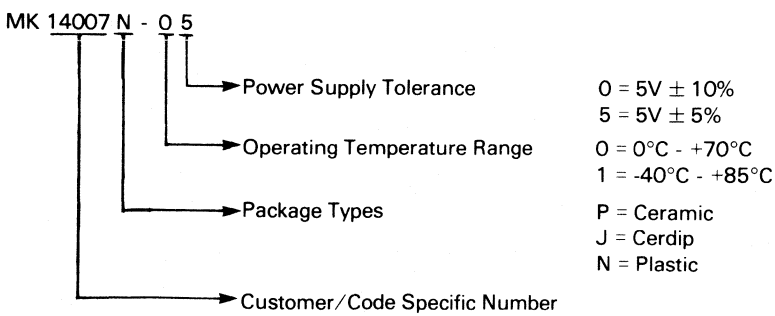


An example of the generic part number for the EPROM device is shown below.

MK38P70/02 R-05

DEVICE ORDER NUMBER

An example of the device order number is shown below.



The Customer/Code specific number defines the ROM bit pattern, I/O configuration, oscillator type, and generic part type to be used to satisfy the requirements of a particular customer purchase order. For further information on the ordering of mask ROM devices, the customer should refer to the 3870 Family Technical Manual.

PRELIMINARY

MOSTEK®

3870 SINGLE CHIP MICRO FAMILY

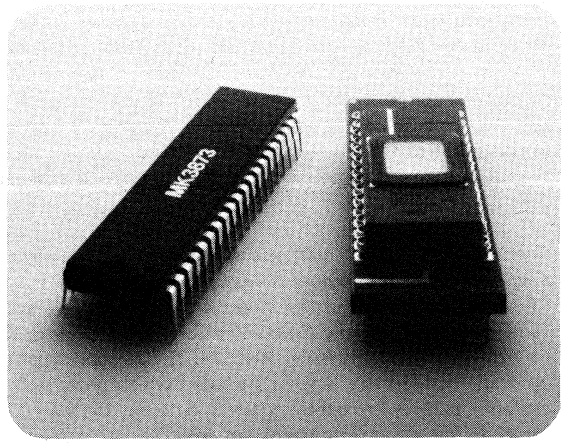
MK3873 and MK38P73

MK3873 FEATURES

- ☐ Available with 1K or 2K byte mask programmable ROM
- ☐ Software compatible with 3870 instruction set
- ☐ 64 byte scratchpad RAM
- ☐ Available with 64 byte Executable RAM
- ☐ 29 bits (4 ports) TTL compatible parallel I/O
- ☐ Serial Input/Output port
  - External or Internal Serial Port Clock
  - Transmit and Receive registers double buffered
  - Internal Baud rate generator
  - Synchronous or Asynchronous serial I/O
  - Data rates to 9600 bits per second (ASYNC)
  - I/O pins dedicated as SERIAL IN, SERIAL OUT, and SERIAL CLOCK
  - Variable duty cycle waveform generation
- ☐ Vectored interrupts
- ☐ Programmable binary timer
  - Internal timer mode
  - Pulse width measurement mode
  - Event counter mode
- ☐ External Interrupt
- ☐ Crystal, LC, RC or external time base options available
- ☐ Low power (325 mW typ.)
- ☐ Single +5V power supply
- ☐ Pinout compatible with the 3870 Family members

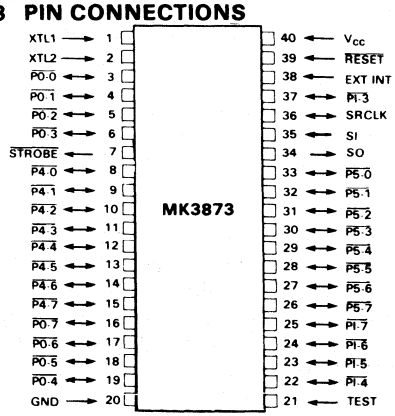
MK38P73 FEATURES

- ☐ EPROM version of MK3873
- ☐ Piggyback PROM (P-PROM)™ package
- ☐ Accepts 24 pin or 28 pin EPROM memories
- ☐ Identical pinout as MK3873
- ☐ In-Socket emulation of MK3873

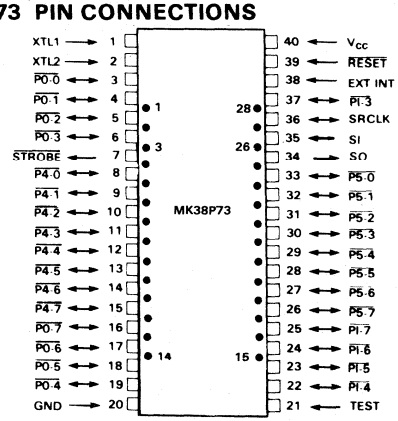


III 3870 SINGLE CHIP MICROCOMPUTER FAMILY

MK3873 PIN CONNECTIONS



MK38P73 PIN CONNECTIONS



GENERAL DESCRIPTION

The MK3873 single chip microcomputer introduces a major addition to the 3870 microcomputer family, a serial input/output port. This serial port is capable of either synchronous or asynchronous serial data transfers. The heart of the serial port is a 16-bit Shift Register that is double-buffered on transmit and receive. The Shift Register clock source can be either the internal baud rate generator or an external clock. An end-of-word vectored interrupt is generated in either transmit or receive mode so that the CPU overhead is only at the word rate and not at the serial bit rate. This serial channel can be used to provide a low-cost data channel for communicating between 3873 microcomputers or between a 3873 and another host computer. The serial port is also very flexible so that it could be used for other purposes such as an interface to external serial logic or serial memory devices.

The MK3873 retains commonality with the 3870 family of single chip microcomputers. It has up to 2048 bytes of mask ROM for program storage, and 64 bytes of scratchpad random-access memory. Certain versions also include up to 64 bytes of Executable RAM. Also, the 3870's sophisticated programmable binary timer is included which provides for system flexibility by operating in 3 different modes. The MK3873 has a large number of parallel I/O lines available to the user. Twenty nine pins of the MK3873 are dedicated to parallel I/O. In addition, three pins are dedicated to the serial I/O port. These pins provide input, output, and clock for the serial port. The serial clock pin can be driven externally or programmed to provide a 50% duty cycle TTL compatible serial clock. No additional CPU instructions are necessary for use with the serial port. Thus, the MK3873 is instruction set compatible with the rest of the 3870 family.

The MK38P73 microcomputer is the PROM based version of the MK3873 single-chip microcomputer. The MK38P73 is called the Piggyback PROM (P-PROM)™ microcomputer because of a new packaging concept. This concept allows a 24 or 28 pin EPROM to be mounted directly on top of the microcomputer itself. The EPROM can then be removed and reprogrammed as required with a standard PROM programmer. The MK38P73 retains exactly the same pinout and architectural features as other members of the MK3873 Family. The MK38P73 is discussed in more detail in a later section of this document.

FUNCTIONAL PIN DESCRIPTION

$\overline{P0\text{-}0}$ - $\overline{P07}$, $\overline{P1\text{-}3}$ - $\overline{P1\text{-}7}$, $\overline{P4\text{-}0}$ - $\overline{P4\text{-}7}$, $\overline{P5\text{-}0}$ - $\overline{P5\text{-}7}$ are 29 bidirectional I/O lines which can either be used as TTL compatible inputs or latch outputs.

SI - SERIAL IN is a TTL compatible Schmitt Trigger input pin for either serial synchronous or asynchronous data.

SO - SERIAL OUT is an output line for either serial synchronous or asynchronous data.

SRCLK is the clock for the serial port operations. It can be configured by software to be an input or output depending upon whether an internal baud rate or external clock is desired. It has a Schmitt trigger input and can be used to drive up to 3 TTL loads.

$\overline{STROBE}$ is a ready strobe associated with I/O Port 4. This pin which is normally high provides a single low pulse after valid data is present on the $\overline{P4\text{-}0}$ - $\overline{P4\text{-}7}$ pins during an output instruction. $\overline{STROBE}$ can be used to drive up to 3 TTL loads.

$\overline{RESET}$ may be used to externally reset the MK3873. When pulled low the MK3873 will reset. When allowed to go high the MK3873 will begin program execution at program location H'000'.

PIN NAME	DESCRIPTION	TYPE
$\overline{P0\text{-}0}$, $\overline{P0\text{-}7}$	I/O Port 0	Bidirectional
$\overline{P1\text{-}3}$ - $\overline{P1\text{-}7}$	I/O Port 1	Bidirectional
$\overline{P4\text{-}0}$ - $\overline{P4\text{-}7}$	I/O Port 4	Bidirectional
$\overline{P5\text{-}0}$ - $\overline{P5\text{-}7}$	I/O Port 5	Bidirectional
$\overline{STROBE}$	Ready Strobe	Output
$\overline{EXT\ INT}$	External Interrupt	Input
$\overline{RESET}$	External Reset	Input
SI	Serial Input	Input
SO	Serial Output	Output
SRCLK	Serial Clock	Bidirectional
TEST	Test Line	Input
XTL 1, XTL 2	Time Base	Input
V_{CC}, GND	Power Supply Lines	Input

EXT INT is the external interrupt input. Its active state is software programmable as described in the 3870 Family Technical Manual. This input is also used in conjunction with the timer for pulse width measurement and event counting.

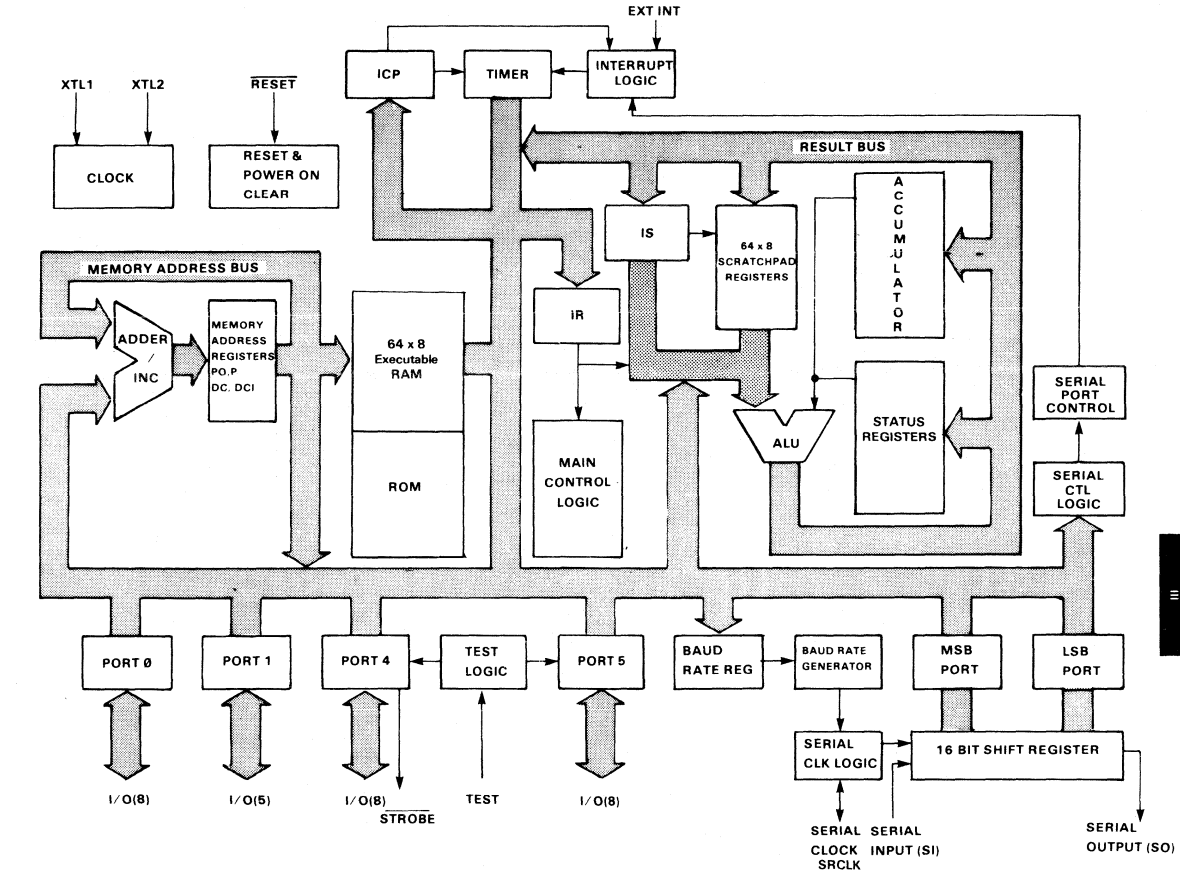
XTL 1 and XTL 2 are the time base inputs (2 MHz to 4 MHz) to which a crystal, LC network, RC network, or an external single-phase clock may be connected. The time base mode must be specified when submitting an order for a mask ROM MK3873. The MK38P73 will operate with any of the four configurations.

MK3873 ARCHITECTURE

The architecture of the MK3873 is identical to that of the rest of the devices in the 3870 family, with the exception of the serial port logic. The serial port logic is shown in the block diagram of the MK3873 (Figure 1). Addressing of the serial port logic is accomplished through I/O instructions. Operation and programming of the serial port is thoroughly discusssed below. A programming-model of the MK3873 is shown in Figure 2. For a more complete discussion of the 3870 family architecture, the user is referred to the 3870 Family Technical Manual.

MK3873 BLOCK DIAGRAM

Figure 1



MAIN MEMORY

The main memory section on the MK3873 consists of a combination of ROM and executable RAM.

There are four registers associated with the main memory section. These are the Program Counter (PO), the Stack Register (P), the Data Counter (DC), and the Auxiliary Data Counter (DC1). The Program Counter is used to address instructions during program execution. P is used to save the contents of PO during an interrupt or subroutine call. Thus, P contains the return address at which processing is to resume upon completion of the subroutine or the interrupt routine. The Data Counter (DC) is used to address data tables. This register is auto-incrementing. Of the two data counters, only DC can access memory directly. However, the XDC instruction allows DC and DC1 to be exchanged.

The length of the PO, P, DC, and DC1 registers for all MK3873 devices is listed in the table shown in Figure 3. The graph and table in Figure 3 also shows the amounts of ROM and executable RAM for the different members of the MK3873 family.

EXECUTABLE RAM

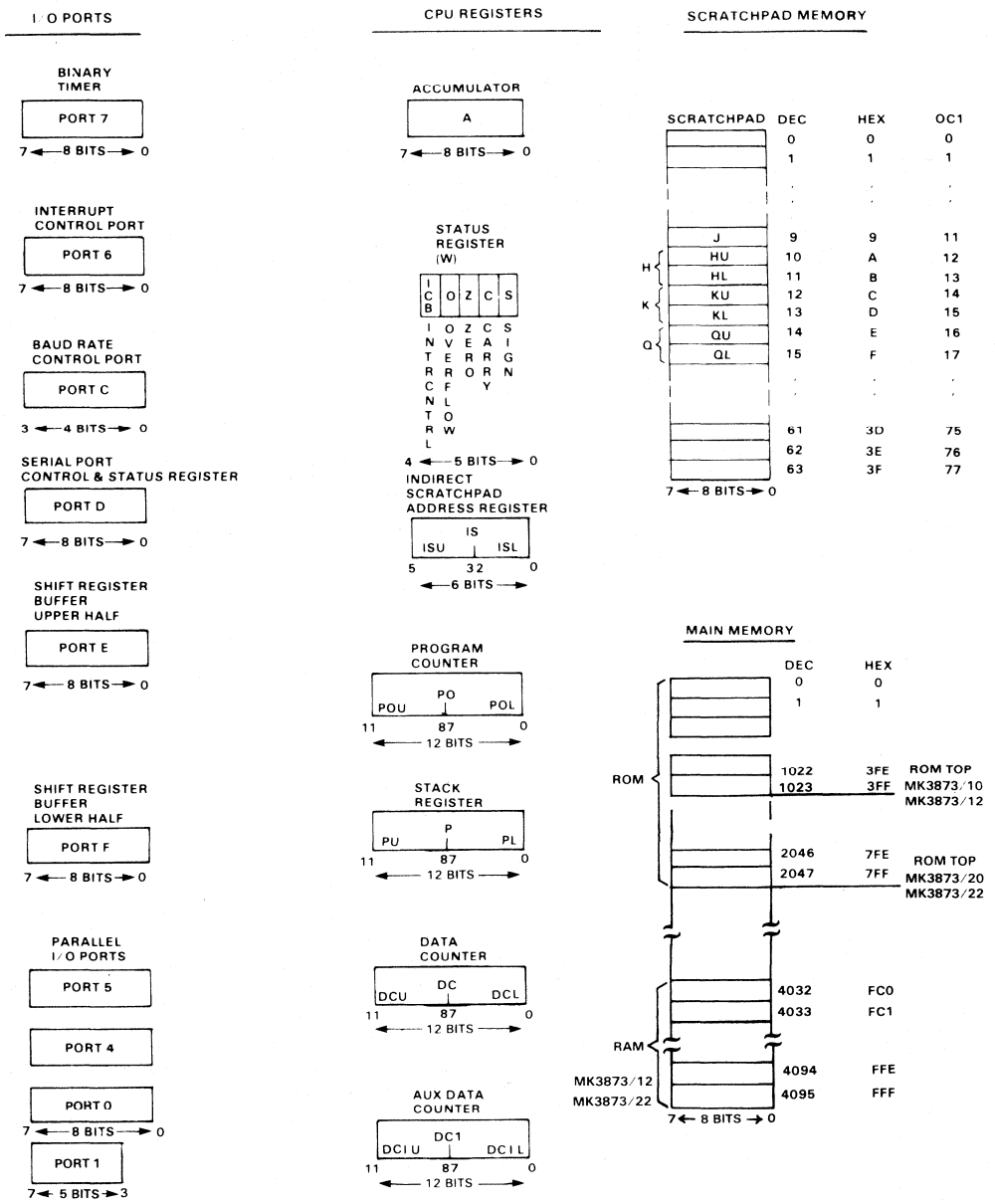
The upper bytes of the total address space in certain MK3873 devices is RAM memory. As with the ROM memory the RAM may be addressed by the PO and DC address registers. The executable RAM may be accessed by all 3870 instructions which address main memory indirectly through the Data Counter (DC) register. Additionally, the MK3873 may execute an instruction sequence which resides in the Executable RAM. Note that this cannot be done with the scratchpad RAM memory, which is the reason the term "Executable RAM" is given to this additional memory.

I/O PORTS

On the MK3873, 29 lines are provided for bidirectional, parallel I/O. These lines are addressable as four parallel I/O ports at locations 0, 1, 4, and 5. Note that Ports 0, 4, and 5 are 8 bits wide, while Port 1 contains only 5 bits of I/O in bit positions 3, 4, 5, 6, and 7. Bits 0-2 on Port 1 are not available for use as I/O port pins or as storage elements. The remaining three pins are used to provide the serial I/O

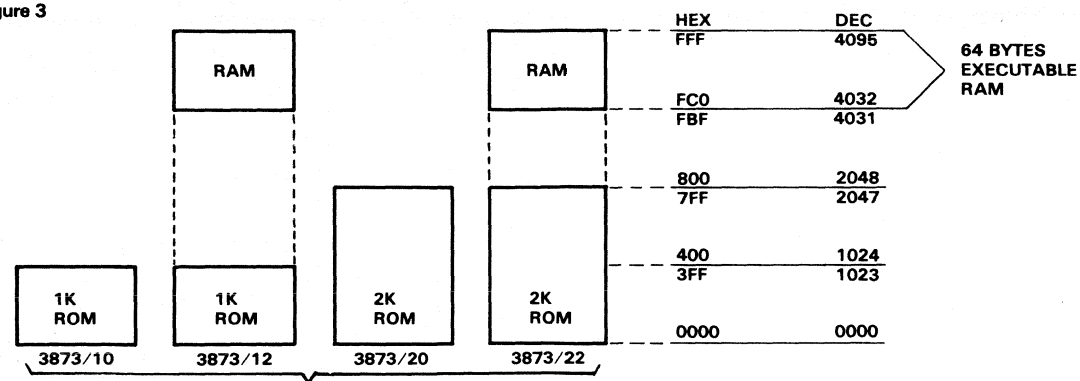
MK3873 PROGRAMMABLE REGISTERS, PORTS, AND MEMORY MAP

Figure 2



MK3873 MAIN MEMORY SIZES AND TYPES

Figure 3



All devices contain 64 bytes of Scratchpad RAM

Data derived from addressing any locations other than within the specified ROM and RAM space is not tested nor is it guaranteed. Users should refrain from entering this area of the memory map.

Device	Scratchpad RAM Size (Decimal)	Address Register Size P0, P, DC, DC1	ROM Size (Decimal)	Executable RAM Size
MK3873/10	64 bytes	12 bits	1024 bytes	0 bytes
MK3873/12*	64 bytes	12 bits	1024 bytes	64 bytes
MK3873/20	64 bytes	12 bits	2048 bytes	0 bytes
MK3873/22*	64 bytes	12 bits	2048 bytes	64 bytes

*The 3873/12 and 3873/22 will be available as future products.

function. A conceptual schematic of a bidirectional I/O port pin and available output drive options are shown in Figure 4.

As in all other 3870 family devices, an output ready strobe is associated with Port 4. This flag may be used to signal a peripheral device that the 3873 has just completed an output of new data to port 4. The strobe provides a single low pulse shortly after the output operation is completely finished, so either edge may be used to signal the peripheral. $\overline{\text{STROBE}}$ may also be used as an input strobe by doing a dummy output of H '00' to port 4 after completing the input operation.

SERIAL I/O OPERATION

The Serial Input/Output Port consists of a serial Shift Register, baud rate generator and control logic as shown in Figure 1. Together these elements provide the MK3873 with a half duplex asynchronous, or a full duplex synchronous, variable bit length serial port. Data is shifted into or out of the shift register at a rate determined by the internal baud rate generator or external clock. An end-of-word interrupt is generated in transmit or receive mode so that the CPU overhead is only at the word rate and not the serial bit rate.

SHIFT CLOCK

The internal clock is used to clock data transfers into and out of the 16 bit Shift Register. It is also used as an input to an internal counter which keeps track of the number of bits which have been shifted into or out of the Shift Register. Input data is sampled on the SERIAL INPUT, (SI), line on the rising edge of the $\overline{\text{SHIFT}}$ clock and is clocked into the most significant bit of the shift register. Output data is gated to the SERIAL OUTPUT line on the falling edge of the internal SHIFT clock.

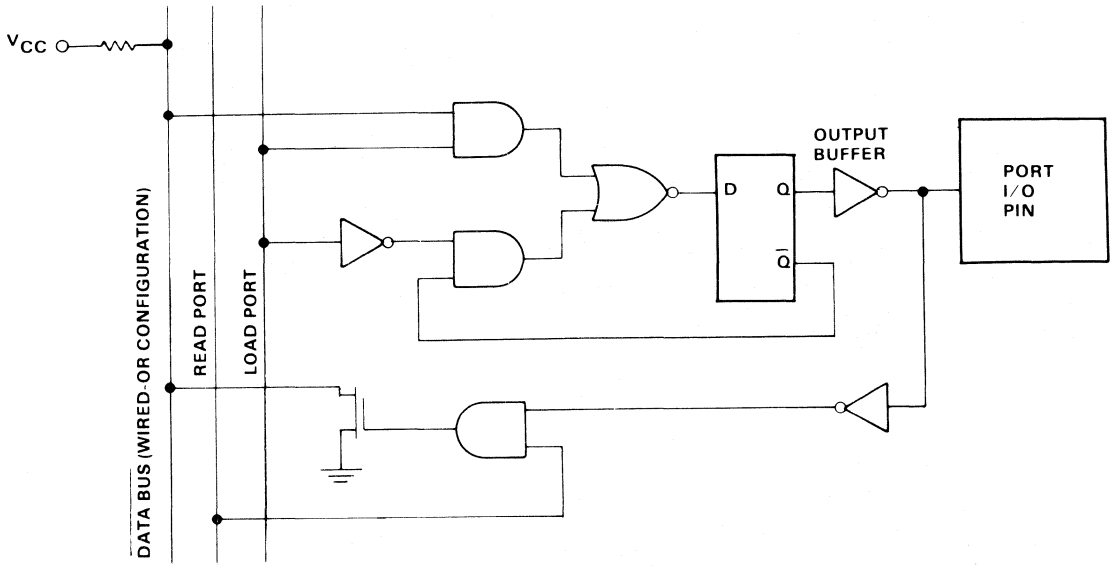
The clock is derived from the SRCLK pulse. The SRCLK pulse may be generated from the internal baud rate generator or it may be programmed as an input. The internal SHIFT clock operates at the same frequency as the SRCLK pulse when the Sync mode is selected, and at a rate which is divided by 16 (÷16) from the SRCLK pulse when the Async mode is selected.

SHIFT REGISTER

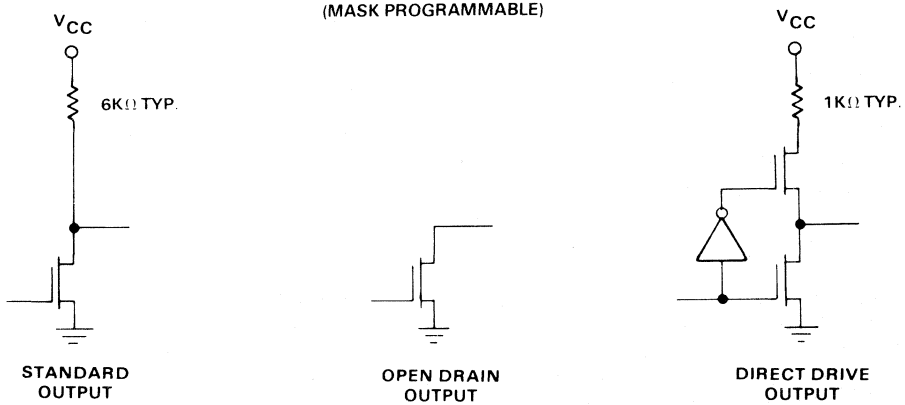
The Serial Port Shift Register is a 16-bit serial to parallel, parallel to serial shift register. This register is addressed and double-buffered by ports E and F as shown in Figure 5A.

I/O PIN CONCEPTUAL DIAGRAM WITH OUTPUT BUFFER OPTIONS

Figure 4



OUTPUT BUFFER OPTIONS
(MASK PROGRAMMABLE)



STANDARD OUTPUT

OPEN DRAIN OUTPUT

DIRECT DRIVE OUTPUT

Ports 0 and 1 are Standard Output type only.

Ports 4 and 5 may both be any of the three output options (mask programmable bit by bit)

The $\overline{\text{STROBE}}$ output is always configured similar to a Direct Drive Output except that it is capable of driving 3 TTL loads.

$\overline{\text{RESET}}$ and EXT INT may have standard 6KΩ (typical) pull-up or may have no pull-up (mask programmable). These two inputs have Schmitt trigger inputs with a minimum of 0.2 volts of hysteresis.

Serial In is a Schmitt trigger input with a minimum of 0.2V hysterisis.

Serial Out (SO) is the Standard Output type.

SRCLK output is capable of driving 3 TTL loads.

PORT D SERIAL PORT CONTROL REGISTER

The Serial Port Control register is write only and is addressed as Port D. The bit assignment is pictured in Figure 5C. The function of each bit is described below.

N2, N1, N0 - WORD LENGTH SELECT

These bits select one of the eight possible word lengths which are available with the MK3873 serial port. The serial port will shift the programmed number of bits through the Shift Register. If the Transmit mode is selected, data will be shifted out of the least significant bit (SR0) of the Shift Register to the Serial Out line (SO) while data is simultaneosly sampled at the Serial Input (SI) line and shifted into the most significant bit (SR15) of the Shift Register. When the Receive mode is selected, data will be sampled at SI and shifted in, but the SO line will be disabled such that it remains in a marking condition (Logic "1"). After the programmed number of bits have been shifted, the serial port logic will generate an end-of-word condition. This end-of-word condition will cause an interrupt if the serial port INTERRUPT ENABLE bit has been set.

It should be noted that the word values have been chosen so that the MK3873 can be programmed to send and receive a wide variety of asynchronous serial codes with various combinations of start and stop bits. Shown in Figure 6 is a table which gives the word length.

Values which would be programmed into the MK3873 Serial Port Register for Baudot, ASCII and 8 bit binary codes in an asynchronous word format are shown in the table of Figure 6. Shown in the table are word length values for various combinations of data bits, start and stop bits, and parity. It can be seen that the MK3873 serial port can accommodate many different word lengths of asynchronous or synchronous data.

SERIAL PORT REGISTERS

Figure 5A

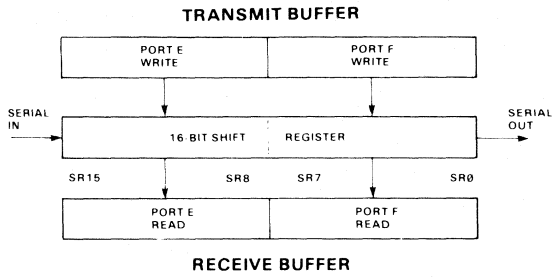


Figure 5B

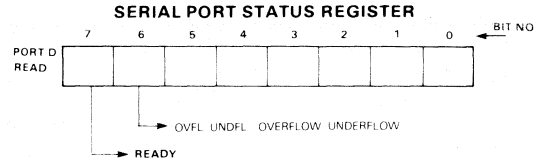
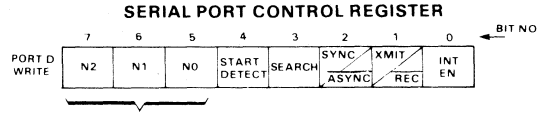


Figure 5C



WORD LENGTH SELECT

N2	N1	N0	WORD LENGTH
0	0	0	4 Bits
0	0	1	7 Bits
0	1	0	8 Bits
0	1	1	9 Bits
1	0	0	10 Bits
1	0	1	11 Bits
1	1	0	12 Bits
1	1	1	16 Bits

ASYNCHRONOUS WORD LENGTHS

Figure 6

DATA WORD	# OF BITS	START BITS	STOP BITS	PARITY	WORD LENGTH (BITS)
BAUDOT	5	1	1	No	7
	5	1	2	No	8
	5	1	1	Yes	8
	5	1	2	Yes	9
ASCI I	7	1	1	No	9
	7	1	2	No	10
	7	1	1	Yes	10
	7	1	2	Yes	11
8 Bit Binary	8	1	1	No	10
	8	1	2	No	11
	8	1	1	Yes	11
	8	1	2	Yes	12

START DETECT

When the START DETECT bit is enabled the serial port will not shift data through the Shift Register until a valid start bit is detected at the SI input pin. The Start Detect mode is operative only when the Async mode has been selected by programming bit 2 of the Serial Port Control Register to a logic "0". By selecting the Async mode, the internal SHIFT clock frequency is divided by 16 from the clock frequency at the SRCLK pin. (Recall that SRCLK can be an input or an output depending on whether the internal baud rate generator or the external clock is selected). When the START DETECT bit is set, the serial port logic looks for a high to low transition on the SI input. Until this transition occurs, the internal SHIFT clock is held low and no data is shifted in through the shift register. Once the transition is sensed, the SI input will be sampled on every SRCLK pulse for seven clock periods. If the logic level remains at zero on the SI input for each of the seven clock periods, the serial port logic will begin shifting data into the Shift Register on the eighth SRCLK pulse. Data will be shifted in at the ÷16 or SHIFT clock rate until the number of bits which have been programmed into the word length select have been shifted in. Once the programmed number of bits have been shifted in, the start detect circuitry will be rearmed and will begin searching for the next high-to-low transition on SI. This operation is pictured in the example shown in Figure 7.

When the START DETECT bit is disabled, data is continuously shifted through the Shift Register. An end-of-word condition will be generated each time the programmed number of bits has been shifted into or out of the Shift Register. A serial port interrupt will be generated when the end of word condition occurs if it has been enabled.

SEARCH

The SEARCH bit is enabled by programming it to a logic "1". When enabled, the SEARCH bit causes the serial port logic to generate an interrupt at every bit time if the serial port interrupt has been enabled. This interrupt will occur regardless of whether the Transmit or Receive mode has been selected and whether the Synchronous or Asynchronous mode has been selected. The Search mode is usually used for recognition of a sync character in synchronous serial data transmission. The MK3873 serial port does not automatically detect sync characters.

SYNC/ASYNC

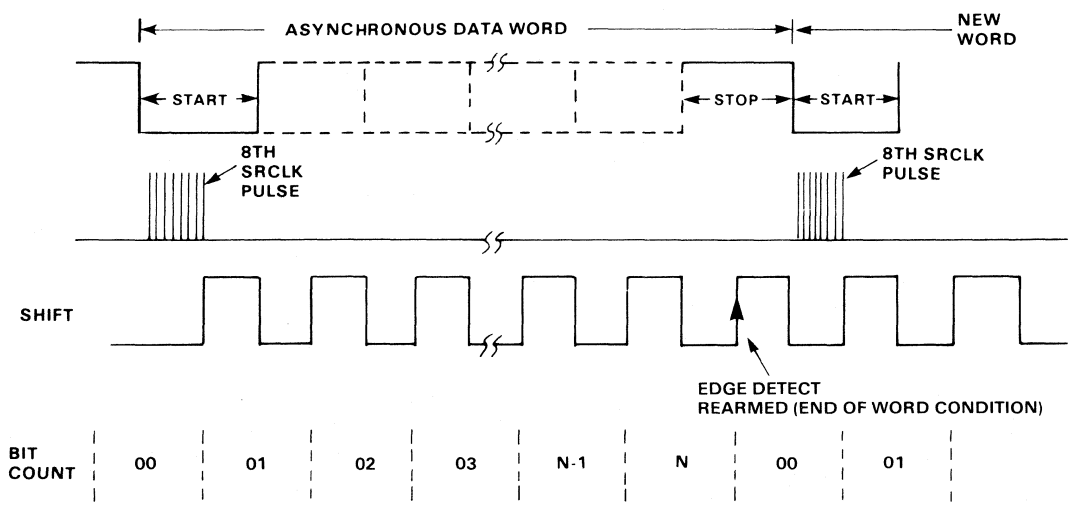
The SYNC/$\overline{\text{ASYNC}}$ bit is used to select either the Synchronous mode of operation or the Asynchronous mode of operation. In the Synchronous mode of operation data is shifted through the Shift Register at a rate which is ÷1 the rate of SRCLK. When the Synchronous mode is selected, the start bit detect circuitry cannot be enabled, even if the START DETECT bit is programmed to a "1". In the Asynchronous mode (SYNC/$\overline{\text{ASYNC}}$ = 0) the internal SHIFT clock operates at a rate which is ÷16 the rate of SRCLK.

XMIT/$\overline{\text{REC}}$

The XMIT/$\overline{\text{REC}}$ bit is used to select either the Transmit or Receive modes of operation. When programmed to a "1" XMIT is selected and the serial port will shift data on the SO line as well as shift data into the SI input. Transmitted data will be enabled on the SO output on the falling edge of the internal SHIFT clock. When the Receive mode is selected (by programming XMIT/$\overline{\text{REC}}$ = 0), data will be clocked into the Shift Register on the rising edge of SHIFT, as it is when the

MK3873 SERIAL PORT START BIT DETECTION

Figure 7



where N is the word length value selected by programming bits N2-N0 in the serial port control register

Transmit mode is enabled, but data will be disabled from being shifted out on Serial Out. Serial Out will be held at a marking, or logic "1", condition.

SERIAL PORT INTERRUPT ENABLE

By programming this bit to a "1", the serial port interrupt will be enabled. A serial port interrupt may then occur when an end-of-word condition is generated. Program control will be vectored to one of two locations upon a serial port interrupt, depending on the way the XMIT/$\overline{\text{REC}}$ bit has been programmed. If the Transmit mode has been selected by programming XMIT/$\overline{\text{REC}}$ bit to a "1", then program control will be vectored to location E0 (Hex). For the Receive mode (XMIT/$\overline{\text{REC}}$ = 0) program control will be vectored to 60 (Hex) when the serial port interrupt occurs. With the addition of the Serial Port Interrupt, the MK3873 has three sources of interrupt. If these three interrupts were to occur simultaneously, priority between them would be such that they would be serviced in the following order:

1) Serial Port
2) Timer
3) External Interrupt

STATUS REGISTER

Reading port D of the MK3873 by performing an Input or Input Short (IN or INS) instruction will load the contents of the Serial Port Status Register into the Accumulator. The two bits which make up the Status Register are shown in Figure 5B. The operation of these two bits is described below:

READY - The meaning of the READY flag depends on whether the Transmit or Receive mode is selected. When the Transmit mode has been selected, the READY flag is set when a Transmit Buffer empty condition occurs. This means that any previous data which may have been loaded into the Transmit Buffer register pair has been transferred into the Shift Register. Loading either byte of the Transmit Buffer will clear the READY flag until the time that the Transmit Buffer register pair is loaded into the Shift Register during an end-of-word condition

In the Receive mode (XMIT/$\overline{\text{REC}}$ = 0), the READY flag is used to indicate a Receive Buffer full condition. This means that a word of the programmed length has been shifted in and has been loaded into the receive buffer register pair. Reading one of the ports E or F which make up the receive buffer register pair will clear the READY flag. The READY flag will remain a 0 until the next word is completely shifted in and loaded into the receive buffer.

OVFL/UNDFL is like the READY flag; the meaning of OVFL/UNDFL depends on the programming of the XMIT/$\overline{\text{REC}}$ bit in the Serial Port Control Register. When the Transmit mode has been selected OVFL/UNDFL is used to indicate a transmitter underflow condition.

A transmitter underflow condition can occur as follows: Assume that the Transmit mode is selected. Suppose that a word is loaded into the Transmit Buffer register. The serial port logic will load the contents of the Transmit Buffer into the Shift Register and will begin to shift the word out on the SO pin. When the contents of the Transmit Buffer are loaded into the Shift Register, the serial port logic will signal the Transmit Buffer empty condition by setting the READY flag to a "1". When the word in the Shift Register is completely shifted out, an end-of-word condition will be generated. The serial port logic will then check to see if new data has been loaded into the Transmit Buffer. If it has not, the OVFL/UNDFL flag will be set, indicating that the serial port logic has run out of data to send. The OVFL/UNDFL flag can be used to signal an error condition to the firmware, or it can be used to signal that all data has been cleared out of the Shift Register for the purposes of line turnaround.

The OVFL/UNDFL flag which, in this case, represents a transmitter underflow condition, is reset by reading the Status Register.

When the Receive mode is programmed, OVFL/UNDFL is used to signal that the Receive Buffer has overflowed. This overflow condition can occur as follows: Suppose that a serial word is shifted in, generating an end-of-word condition. The serial port logic will load the contents of the Shift Register into the Receive Buffer, and will set the READY flag to a "1" to indicate that the Receive Buffer is full. When the next word being received is completely shifted in, generating the next end-of-word condition, the serial port logic will check to see if the Receive Buffer has been read by examining the state of the READY flag. If the READY flag = 0, then the previous word has already been read from the Receive Buffer by the software and the serial port logic will load the current word into the Receive Buffer and will again set the READY flag. If the READY flag = 1, then the previous word has not been read from the Receive Buffer. The serial port logic will load the new word into the Receive Buffer, destroying the previous word. This action is signalled by the serial port logic setting the OVFL/UNDFL to a "1" signalling a receive buffer overflow condition. In this case reading the status register also clears the OVFL/UNDFL flag.

BAUD RATE CONTROL REGISTER

Port C is designated as the Baud Rate Control register. Four bits, 0-3, are used to select nine different internal baud rates or an external clock. When an internal baud rate is programmed, the SRCLK output is generated at a frequency which is divided from the MK3873's time base frequency. The SRCLK frequency can be calculated by dividing the time base frequency by the divide factor shown in Figure 8 for the bit pattern which is programmed into bits C3-C0. Also shown in Figure 7 is the programming of bits C3-C0 to obtain a set of standard baud rates when a 3.6864MHz crystal is used as a time base.

BAUD RATE CONTROL PORT
PORT C WRITE ONLY
Figure 8

PORT C WRITE 7	6	5	4	3	2	1	0	Bit No.	Shift Clock Rate @ 3.6864 MHz time base	
X	X	X	X	C3	C2	C1	C0	SRCLK Divide Factor	SYNC	ASYNC
				1	0	1	1	÷24	153.6 kbs	9600 bps
				1	0	1	0	÷48	76.8 kbs	4800 bps
				1	0	0	1	÷96	38.4 kbs	2400 bps
				1	0	0	0	÷192	19.2 kbs	1200 bps
				0	1	1	1	÷384	9600 bps	600 bps
				0	1	1	0	÷768	4800 bps	300 bps
				0	1	0	1	÷1536	2400 bps	150 bps
				0	1	0	0	÷2096	1758.8 bps	110 bps
				0	0	1	1	÷3072	1200 bps	75 bps
				0	0	0	0	External Clock Mode		

When any of the internal baud rates are selected, pin 36 becomes an output port pin. This pin is capable of driving three standard TTL inputs and provides a square wave output from the frequency selected in port C. The SYNC/$\overline{\text{ASYNC}}$ bit in the Serial I/O Control register has no effect on the output clock rate. The output will always be ÷1 directly from the baud rate generator.

If all zeros are loaded into this port, the External Clock mode is selected. Pin 36 becomes an input. Any TTL compatible square wave input can be used to generate the clock for the serial port.

TRANSMIT AND RECEIVE BUFFERS

The Receive Buffer registers are two eight bit registers which are addressed as ports E and F (Hex) and are read only. The Receive Buffer registers may be read at any time. The Transmit Buffer registers are also two 8-bit registers which are write only and addressed as ports E and F (Hex).

In the Receive mode, the contents of the 16 bit Shift Register are transferred to the Receive Buffer Register pair when a complete word has been shifted in. Bits SR15-SR8 of the Shift Register are loaded into bits 7-0 of port E while bits SR7-SR0 are loaded into bits 7-0 of Port F.

When entering the Transmit mode, the first data transfer from the Transmit Buffer to the 16 bit Shift Register won't occur until a 1 word time delay after entering Transmit Mode.

In the Receive mode, no transfers between the Transmit Buffer and the 16 bit Shift Register can occur.

The serial port does not automatically right justify incoming data, nor does it insert or strip start and stop bits from an asynchronous data word. Therefore, it is usually necessary to right justify incoming data read from the Receive Buffer registers in software through shift instructions, as well as strip start and stop bits if an asynchronous data format is being used. Likewise, in transmitting an asynchronous data word, it is usually necessary to insert start and stop bits in software into the 16 bit word which is to be loaded in two halves into the Transmit Buffer register.

RESET

The reset circuit on the MK3873 is used to initialize the device to a known condition either during the course of program execution or on a power on condition. This section discusses the effect of $\overline{\text{RESET}}$ on the serial port logic. A more complete description of $\overline{\text{RESET}}$ may be found in the 3870 Family Technical Manual.

Upon reset, both the serial port control register (port D) and the Baud Rate Control register (port C) are loaded with zeroes. This action sets the serial port control logic in the following state:

N2, N1, N0 (word length) = 4 bits
START DETECT disabled
SEARCH disabled
Asynchronous Receive mode
Serial port interrupt disabled
External Clock mode (SRCLK = 1).
Ports E and F are undefined

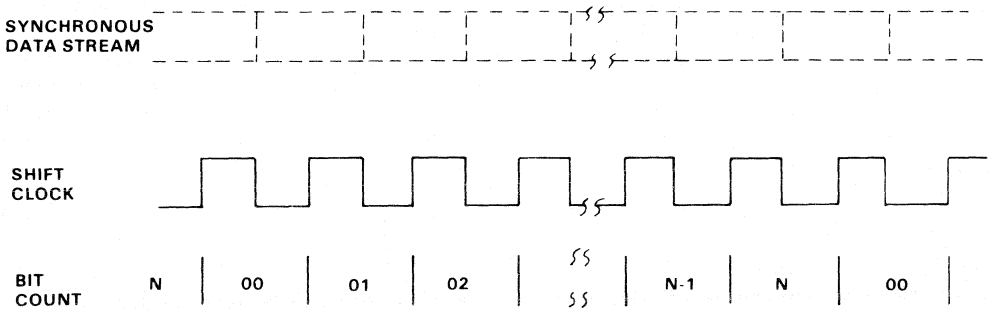
After the first control word is written to the Serial Port Control Register which selects an internal clock mode, the SRCLK will become an output and will remain high for one-half of a clock period as programmed into port C. It will then go low and produce a clock output waveform with the selected frequency.

ASYNCHRONOUS RECEIVE OPERATION

Figure 7 illustrates the timing for an example using the serial port in the Asynchronous mode. When operating in this mode, the Serial Port Control Register should be programmed for receive (XMIT/$\overline{\text{REC}}$ = 0) and the START DETECT bit should be enabled. Also, the Async mode should be selected, which allows the start detect circuitry to operate and sets the internal SHIFT clock at a rate which is divided by 16 (÷16) from the SRCLK rate. Upon selecting the Async mode and the START DETECT bit, the internal SHIFT clock is held low until a negative transition occurs on the SI pin. After a valid edge has been detected (see the START DETECT bit operation section) the SHIFT clock will go high and data will be shifted in at the middle of each bit time. When the programmed number of bits have been shifted in, an end-of-word condition is generated and a serial port receive interrupt will occur if it has been enabled.

After the falling edge of SHIFT following the end-of-word interrupt, the start detect circuitry will be enabled in preparation for the next word. Thus, if a start bit is present immediately following the time when the start detect circuitry is enabled, SHIFT Clock will again go high approximately one bit-time after the rising edge of SHIFT which clocked in the last bit of the preceeding word and caused the end-of-word interrupt. In other words, SHIFT

SYNCHRONOUS TRANSMIT OR RECEIVE TIMING
Figure 9



can go high again on the eighth SRCLK pulse as soon as the start detect circuitry is rearmed.

The Shift Register may be read before the next end-of-word condition; otherwise, a receiver overrun error will occur. For a 9600 bps data rate, this would require reading the Receive Buffer within N x 104 μs from the time that the end-of-word condition is generated, where N is the number of bits in the data word.

The example in Figure 7 shows the timing required for asynchronous data reception from a device such as a teletype. Within this data stream are start, data and stop bits. A typical format requires 1 start bit, 8 data bits and 2 stop bits for a total of 11 bits. All of 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.

SYNCHRONOUS RECEIVE OPERATION

For synchronous operation, the START DETECT bit should not be enabled and the XMIT/$\overline{\text{REC}}$ bit should be programmed to a zero. Also the Sync mode should be enabled so that the internal SHIFT clock is divided by 1, or is equivalent to, SRCLK. Once a control word is written to port D specifying START DETECT = 0, Receive mode, and Sync mode, then the Serial Port will continuously shift data into the MSB of the upper half of the Shift Register at the SRCLK rate and will generate an end-of-word condition when the programmed number of bits have been shifted in.

An illustration of synchronous receive timing is shown in Figure 9. This diagram is a synchronous receive sequence for a word which is N bits in length, where N corresponds to the number of bits which have been programmed into the Serial Port Control Register. Note the relationship of SHIFT clock, the synchronous data stream, and the bit count. Since the START DETECT bit is not enabled, the serial port logic will continuously shift data in and generate end-of-word conditions at regular intervals. When the end-of-word condition occurs, a serial port receive interrupt occurs if it has been enabled, and the contents of the Shift Register will be loaded into the Receive Buffer. The serial port logic will set the READY flag in the Serial Port Status Register, indicating that the receive buffer is full. Since the serial port is double-buffered on receive, the program has entire word time to read the Receive Buffer. At 9600 bps this corresponds to a word time of N x 104μs, where N is the number of bits in a word.

Note that if a new control word is written to port D during the time that a serial word is shifted in, the bit count will be reset.

When using the Synchronous Receive mode on the MK3873, it is usually necessary to establish word synchronization in the data stream. The SEARCH bit, when enabled, causes the serial port logic to interrupt on each rising edge of SHIFT so that the data stream can be examined on a bit by bit basis. When the last bit of a sync word is found, the Search mode can be disabled and the serial port logic will shift in data and interrupt at the word rate.

ASYNCHRONOUS TRANSMIT OPERATION

The Asynchronous Transmit mode of operation is initiated by setting the XMIT/$\overline{\text{REC}}$ bit to a "1", and by programming the SYNC/$\overline{\text{ASYNC}}$ bit to a "0". Also, there must be an SRCLK pulse by selecting an internal or external source for SRCLK by programming port C. Upon setting XMIT/$\overline{\text{REC}}$ to a "1", there will be a 1 word length delay prior to the actual transfer of the first word from the Transmit Buffer to the 16 bit Shift Register. Serial data will then be shifted to the right on each rising edge of the internal SHIFT clock, and each new bit in the data stream will be enabled onto the SERIAL OUTPUT pin (SO) at the time of the falling edge of the

internal SHIFT clock.

As mentioned, one word time delay is generated between the time that the Transmit mode is initiated by programming XMIT/$\overline{\text{REC}}$ = 1 and the time that the contents of the Transmit Buffer are transferred into the Shift Register. This word time delay is generated internally to the MK3873 by counting the number of SHIFT clock pulses which correspond to the number of bits programmed into the word length select section of the Serial Port Control Register (N2, N1, N0). Therefore, the word time delay is equivalent to the time it takes to shift a complete serial data word out of the Shift Register. The same word time delay will result if data had been loaded prior to programming the XMIT/$\overline{\text{REC}}$ bit to a "1". As mentioned in the "START DETECT" bit description, the internal SHIFT clock is disabled when this bit is programmed to a "1". Since the serial port logic counts SHIFT clock pulses to generate the word time delay, the Transmit Buffer contents will never be transferred to the Shift Register and shifted out when the START DETECT bit is enabled. Also, the Transmit Buffer contents cannot be loaded into the Shift Register when XMIT/$\overline{\text{REC}}$ bit = 0.

When the initial serial data word has been transferred into the Shift Register, the READY flag is set in the Serial Port Status Register which is used to indicate the Transmit Buffer is empty. A transmit interrupt will be generated if the INTERRUPT ENABLE bit has been set in the Serial Port Control Register, and program control will be vectored to location E0 (hex). When operating the serial port in a polled environment with the serial port interrupt disabled, the READY bit can be used as a flag which indicates that new data may be loaded into the Transmit Buffer. In an interrupt driven software configuration, new data may be loaded into the Transmit Buffer at the beginning of the serial port interrupt service routine.

During the operation of the Transmit Mode the SERIAL INPUT pin (SI) is sampled and shifted into the Shift Register. However, since the START DETECT bit must be disabled during a transmit sequence, there is no way of establishing bit synchronization on any incoming serial data. Therefore, in the Asynchronous mode, the serial port can only be used in a half-duplex configuration.

After a block of data has been sent, it is sometimes useful for the program to know when the last serial word has been shifted out of the shift register. This is especially useful when operating the MK3873 with a bidirectional half-duplex transmission line. Once the block of serial data has been completely shifted out of the port, then it is usually desirable to reverse the direction of the line so that data may be received.

One way of determining when the last word has been shifted out of the Shift Register is through the use of the OVFL/UNDFL status bit in the Serial Port Status Register. The sequence would take place as follows: The program loads the Transmit Buffer with the last serial data word which is to be sent out either when the "READY" bit is set or during a transmit interrupt service routine. Loading the Transmit Buffer clears the READY flag. At the next end-of-word condition, the last serial data word is transferred from the Transmit Buffer into the Shift Register, which sets the READY flag once again. At this point the program would not load any more data into the Transmit Buffer and the READY flag will remain set. When the last word is completely shifted out of the Shift Register, the serial port logic will check to see if any new data has been loaded into the Transmit Buffer register pair. When it determines that there is no new data in the Transmit Buffer, the serial port logic will set the OVFL/UNDFL bit in the serial port status register and will return the SERIAL OUTPUT pin (SO) to a marking condition (logic "1"). The SERIAL OUTPUT pin (SO) is always returned to a marking condition on transmitter underflow when the ASYNC mode is selected. Since the OVFL/UNDFL bit is set when the last serial data word has completely been sent out, it can be used as a signal to indicate the end of transmission and that the direction of the transmission line may be set for receive.

SYNCHRONOUS TRANSMIT OPERATION

The Synchronous Transmit mode of operation is selected by programming bit 2 (XMIT/$\overline{\text{REC}}$) of the Serial Port Control register to a "1" and setting the SYNC/$\overline{\text{ASYNC}}$ bit to a "1".

Figure 9 illustrates serial output timing relationships in the Synchronous mode. Data is shifted to the right on each rising edge of the internal SHIFT clock. Output data is not enabled to the SERIAL OUTPUT pin (SO) until the falling edge of the SHIFT clock. In a 16 bit data word, SR0, the least significant bit of the Shift Register is shifted out first, and SR15, the most significant bit of the Shift Register, is shifted out last. While the Shift Register contents are being output on a bit by bit basis, data is simultaneously shifted in to the Shift Register through the SI pin.

As discussed in the "ASYNCHRONOUS TRANSMIT OPERATION" section, a word time delay is generated between the time that data is written to the Transmit Buffer and the time that the contents of the Transmit Buffer are loaded into the Shift Register once the XMIT/$\overline{\text{REC}}$ bit has been programmed to a one (1).

Another way of loading the initial data word into the Transmit Buffer requires the word synchronization having been achieved through recognition of a received sync character. Recall that in the Transmit mode, data is sampled at SI and shifted into the Shift Register at the same time that data is shifted out through SO. Upon power up or reset, a control word may be written to Port D which specifies Transmit and Synchronous modes. Word synchronization can then be achieved through the use of the SEARCH bit as described in the section which covers Synchronous Receive mode. Once word synchronization is achieved, the SEARCH bit is disabled and the serial port shifts in data and generates an end-of-word condition at the word rate.

Each time the end of word condition is reached, receive data

is transferred from the shift register into the Receive Buffer. At the same time, data is transferred from the Transmit Buffer into the Shift Register.

Therefore, in the Synchronous Transmit mode, the serial port may be used in a full duplex mode if word synchronization is established. At each end of word condition, output data is transferred to the Shift Register from the Transmit Buffer. At the same time, an incoming data word is transferred from the Shift register to the Receive Buffer register pair. In this case, the End-of-Word transmit routine would be used for sending data by loading the Transmit Buffer register, and for receiving data by reading the Receive Buffer register. Note that once word synchronization is established, an amount of time which is equal to one word time is available following the end-of-word interrupt for loading data into the Transmit Buffer.

The serial port operates differently in the Transmit mode for Synchronous operation than it does for Asynchronous operation. In the Asynchronous mode, after a word has been shifted out, the SO line is returned to a marking condition if no new data has been loaded into the Transmit Buffer.

In the Synchronous mode, after a word has been shifted out, the contents of the Transmit Buffer are loaded into the Shift Register regardless of whether or not new data was loaded into the Transmit Buffer. If new data was not loaded since the last time the transmit buffer was read, the OVFL/UNDFL flag is set which signals a transmitter underflow condition. This feature of always reloading the Shift Register with the contents of the Transmit Buffer when an end-of-word condition occurs allows a sync word to be continuously generated without CPU intervention when the transmitter is idle. This feature also allows variable duty cycle, variable frequency waveforms to be generated on the Serial Output line.

MK38P73 GENERAL DESCRIPTION

The MK38P73 is the EPROM version of the MK3873. It retains an identical pinout with the MK3873. The MK38P73 is housed in the "R" package which incorporates a 28 pin socket located directly on top of the package.

The MK38P73 can act as an emulator for the purpose of verification of user code prior to the ordering of mask ROM MK3870 devices. Thus, the MK38P73 eliminates the need for emulator board products. In addition, several MK38P73s can be used in prototype systems in order to test design concepts in field service before committing to high-volume production with mask ROM MK3873s. The compact size of the MK38P73/EPROM combination allows the packaging of such prototype systems to be the same as that used in production.

Finally, in low-volume applications the MK38P73 can be used as the actual production device.

Most of the material which has been presented for the MK3873 applies to the MK38P73. The MK38P73 has the same architecture and pinout as the MK3873. Additional information is presented in the following sections.

MK38P73 MAIN MEMORY

As can be seen from the block diagram in Figure 10, the MK38P73 contains no on-chip ROM. The memory address and data lines are brought out to the 28 pin socket located directly on top of the 40 pin package. The MK38P73 will address up to 4096 bytes of external EPROM memory.

There is one memory version of the MK38P73 and it is designated as the MK38P73/02. The MK38P73/02 contains 64 bytes of on-chip executable RAM. The MK38P73/02 can emulate the following mask ROM MK3873 devices:

MK3873/10
MK3873/12
MK3873/20
MK3873/22

Addressing of main memory on the MK38P73 is accomplished in the same way as it is for the MK3873. See Figure 12 for Main Memory addresses and for address register size in the MK38P73.

MK38P73 EPROM SOCKET

A 28 pin EPROM socket is located on top of the MK38P73 "R" package. The socket and compatible EPROM memories is shown in Figure 11. When 24 pin memories are used in the 28 pin socket, they should be inserted so that pin 1 of the memory device is plugged into pin 3 of the socket. (The memory should be lower justified in the 28 pin socket.)

The 28-pin socket has been provided to allow use of both 24-pin and 28-pin memory devices. Minor pin-out differences in the memory devices must be accommodated by providing different versions of the MK38P73.

Initially, the MK38P73 that is compatible with the MK2716 is available. The MK38P73 designed to accommodate the 28-pin memory devices will be available at a later date.

MK38P73 I/O PORTS

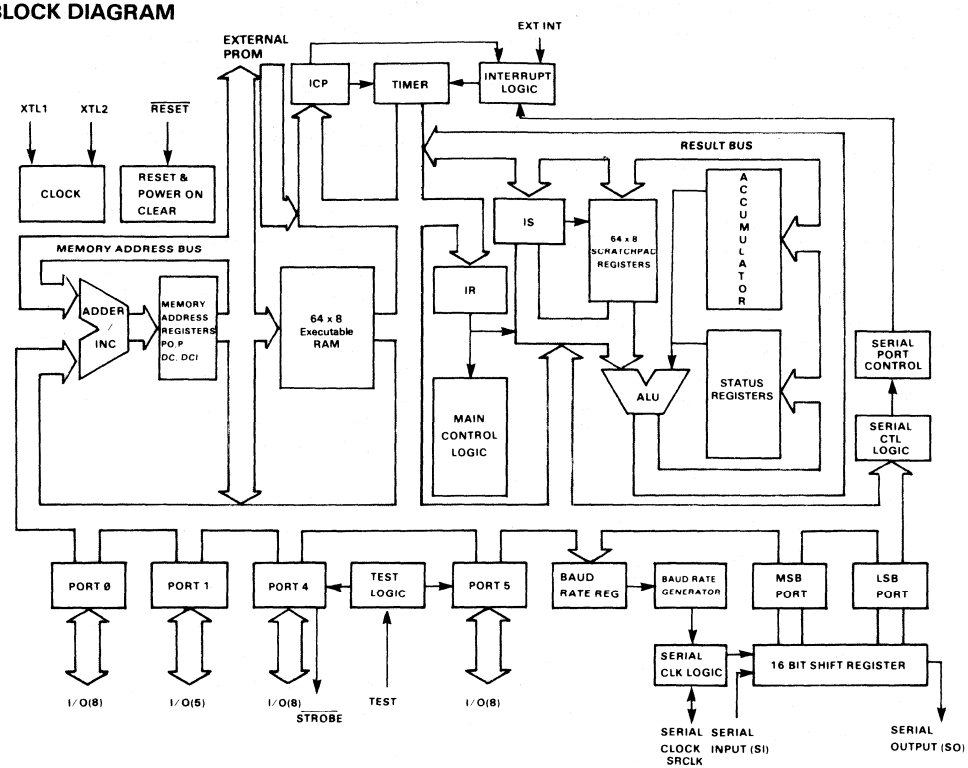
The MK38P73 is offered with open drain type output buffers on Ports 4 and 5. This open drain version is provided so that user-selected open drain port pins on the mask ROM MK38P73 can be emulated prior to ordering those mask ROM parts. Figure 11 lists the part ordering number for an MK38P73/02.

MEMORY ACCESS TIMING

A timing diagram depicting the memory access timing of the MK38P73 is shown in Figure 13. The Φ clock signal is derived

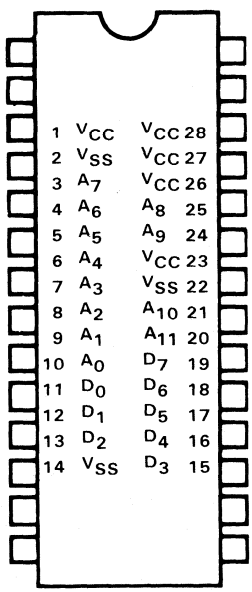
MK38P73 BLOCK DIAGRAM

Figure 10



MK38P73 "R" PACKAGE PINOUT

Figure 11



MK97310 (Open Drain)
Compatible Memories
2758
MK2716
2516 2532

internally in the MK38P73 by dividing the time base frequency by two and is used to establish all timing frequencies. The WRITE signal is another internal signal to the MK38P73 which corresponds to a machine cycle during which time a memory access may be performed. Each machine cycle is either 4 Φ clock periods or 6 Φ clock periods long. These machine cycles are termed short cycles and long cycles respectively. The worst case memory cycle is the short cycle, during which time an op code fetch is performed. This is the cycle which is pictured in the timing diagram. After a delay from the falling edge of the WRITE clock, the address lines A_{11} - A_0 become stable. Data must be valid at the data out lines of the PROM for a setup time prior to the next falling edge of the WRITE pulse. The total access time available for the MK38P73 is shown as t_{aas}, or the time when address is stable until data must be valid on the data bus lines.

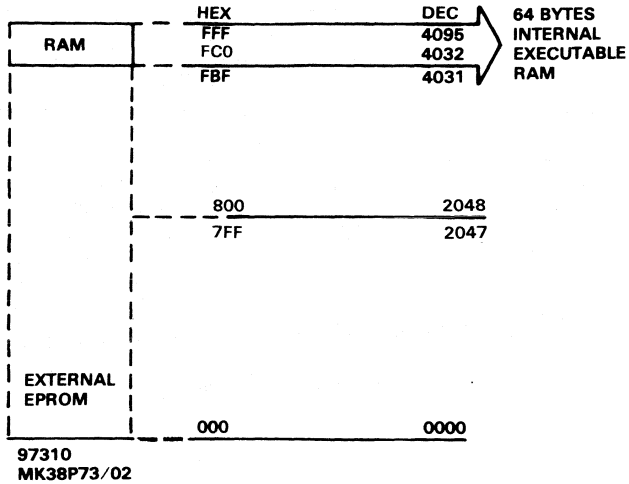
An equation for calculating available memory access time along with some calculated access times based on the listed time base frequencies is also shown in Figure 13.

3873 TIME BASE OPTIONS

The 3873 contains an on-chip oscillator circuit which provides an internal clock. The frequency of the oscillator circuit is set from the external time base network. The time

MK38P73 MAIN MEMORY MAP

Figure 12



Device	Scratchpad RAM Size (Decimal)	Address Register Size P0, P, DC, DC1)	ROM Size (Decimal)	Executable RAM Size
MK38P73/02 97310	64 bytes	12 bits	0 bytes	64 bytes

base for the 3873 may originate from one of four sources:

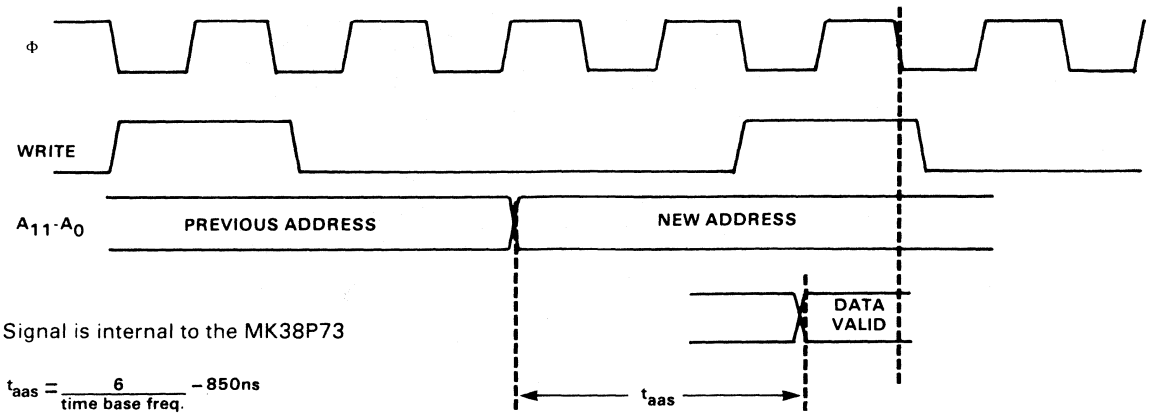
1) Crystal
2) LC Network
3) RC Network
4) External Clock

The type of network which is to be used with the mask ROM MK3873 must be specified at the time when mask ROM devices are ordered. However, the MK38P73 may operate with any of the four configurations so that it may emulate any configuration used with a mask ROM device.

The specifications for the four configurations are given in the following text. There is an internal 26pF capacitor between XTL 1 and GND and an internal 26pF capacitor between XTL 2 and GND. Thus, external capacitors are not necessarily required. In all external clock modes the external time base frequently is divided by two to form the internal PHI clock.

MEMORY ACCESS SHORT CYCLE OP CODE FETCH MK38P73

Figure 13



Signal is internal to the MK38P73

$$t_{aas} = \frac{6}{\text{time base freq.}} - 850\text{ns}$$

(FROM ADDRESS STABLE)

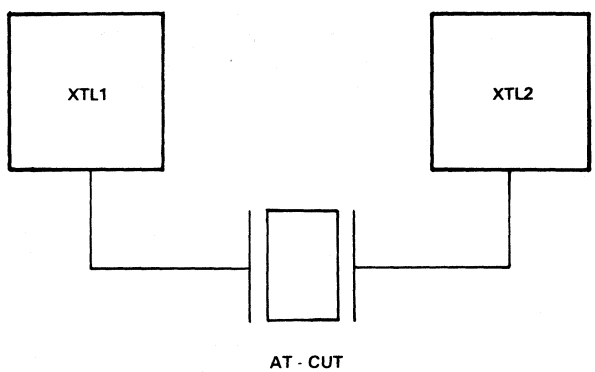
	4MHz	3.58MHz	3MHz	2.5MHz	2MHz
ACCESS TIME	650ns	825ns	1.15μs	1.55μs	2.15μs

CRYSTAL SELECTION

The use of a crystal as the time base is highly recommended as the frequency stability and reproducability from system to system is unsurpassed. The 3873 has an internal divide by two to allow the use of inexpensive and widely available TV Color Burst Crystals (3.58MHz). Figure 15 lists the required crystal parameters for use with the 3873. The Crystal Mode time base configuration is shown in Figure 14.

CRYSTAL MODE CONNECTION

Figure 14



CRYSTAL PARAMETERS

Figure 15

a) Parallel resonsnace, fundamental mode AT-Cut
b) Shunt capacitance (C_0) = 7 pf max.
c) Series resistance (R_S) = See table
d) Holder = See table below.

Frequency	Series Resistance	Holder
f = 2-2.7 MHz	Rs = 300 ohms max	HC-6 HC-33
f = 2.8-4 MHz	Rs = 150 ohms max	HC-6 HC-18* HC-25* HC-33

*This holder may not be available at frequencies near the lower end of this range.

Through careful buffering of the XTL1 pin it may be possible to amplify this waveform and distribute it to other devices. However, Mostek recommends that a separate active device (such as a 7400 series TTL gate) be used to oscillate the crystal and the waveform from that oscillator be buffered and supplied to all devices, including the 3873, in the event that a single crystal is to provide the time base for more than just a single 3873.

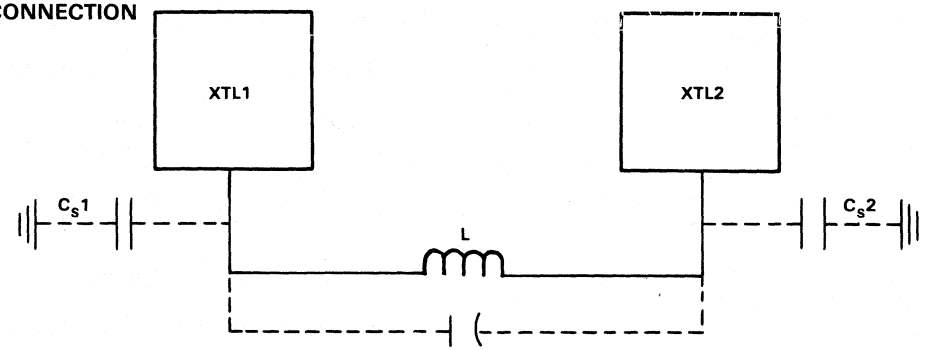
While a ceramic resonator may work with the 3873 crystal oscillator, it was designed specifically to support the use of this component. Thus, Mostek does not support the use of a ceramic resonator either through proper testing, parametric specification, or applications support.

LC NETWORK

The LC time base configuration can be used to provide a less expensive time base for the 3873 than can be provided with a crystal. However, the LC configuration is much less accurate than is the crystal configuration. The LC time base configuration is shown in Figure 16. Also shown in the figure are the specified parameters for the LC components, along with the formula for calculating the resulting time base frequency. The minimum value of the inductor which is required for proper operation of the LC time base network is 0.1 millihenries. The inductor must have a Q factor which is no less than 40. The value of C is derived from C external, the internal capacitance of the 3873, C_{XTL}, and the stray capacitances, C_{S1} and C_{S2}. C_{XTL} is the capacitance looking into the internal two port network at XTL1 and XTL2. C_{XTL} is listed under the "Capacitance" section of the Electrical Specifications. C_{S1} and C_{S2} are stray capacitances from XTL1 to ground and from XTL2 to ground, respectively. C external should also include the stray shunt capacitance across the inductor. This is typically in the 3 to 5 pf range and significant error can result if it is not included in the frequency calculation.

LC MODE CONNECTION

Figure 16

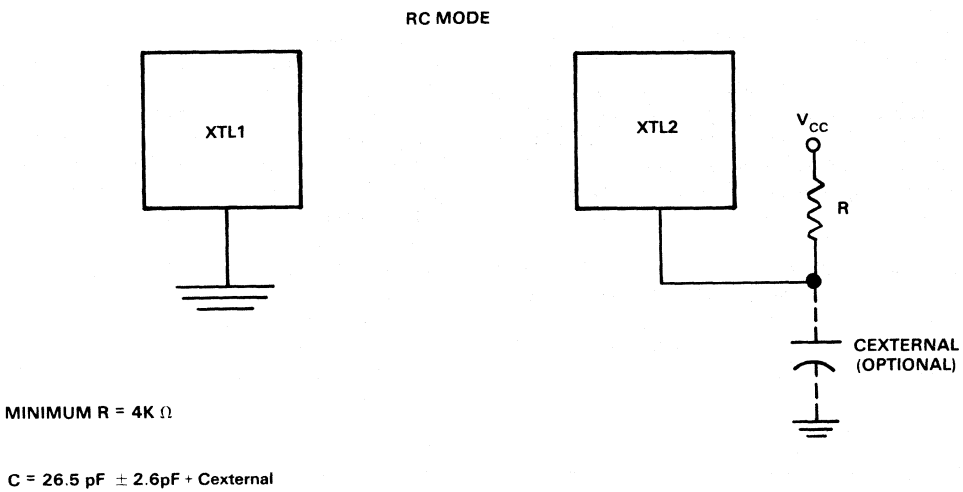


Variation in time base frequency with the LC network can arise from one of four sources: 1) Variation in the value of the inductor. 2) Variation in the value of the external capacitor. 3) Variation in the value of the internal capacitance of the 3873 at XTL1 and XTL2 and 4) Variation in the amount of stray capacitance which exists in the circuit. Therefore, the actual frequency which is generated by the LC circuit is within a range of possible frequencies, where the range of frequencies is determined by the worst case variation in circuit parameters. The designer must select component values such that the range of possible frequencies with the LC mode does not go outside of the specified operating frequency range for the 3873.

RC CLOCK CONFIGURATION

The time base for the 3873 may be provided from an RC network tied to the XTL2 pin, when XTL1 is grounded. A schematic picturing the RC clock configuration is shown in Figure 17. The RC time base configuration is intended to provide an inexpensive time base source for applications in which timing is not critical. Some users have elected to tune each unit using a variable resistor or external capacitor thus reducing the variation in frequency. However, for increased time base accuracy Mostek recommends the use of the Crystal or LC time base configuration. Figure 18 illustrates a curve which gives the resulting operating frequency for a particular RC value. The x-axis represents the product of the value of the resistor times the value of the capacitor. Note that three curves are actually shown. The curve in the middle represents the nominal frequency obtained for a given value of RC. A maximum curve and a minimum curve for different types of 3873 devices are also shown in the diagram.

RC MODE CONNECTION
Figure 17



The designer must select the RC product such that a frequency of less than 2 MHz is not possible taking into account the maximum possible RC product and using the minimum curve shown in Figure 18. Also, the RC product must not allow a frequency of more than 4 MHz taking into account the minimum possible R and C and using the Maximum curve shown . Temperature induced variations in the external components should be considered in calculating the RC product.

Frequency variation from unit to unit due to switching speed and level at constant temperature and V_{CC} = + or - 5 percent.

Frequency variation due to V_{CC} with all other parameters constant with respect to +5V = +7 percent to -4 percent on all devices.

Frequency variation due to temperature with respect to 25 C (all other parameters constant) is as follows:

PART #	VARIATION
387X-00, -05	+6 percent to - 9 percent
387X-10, -15	+9 percent to -12 percent

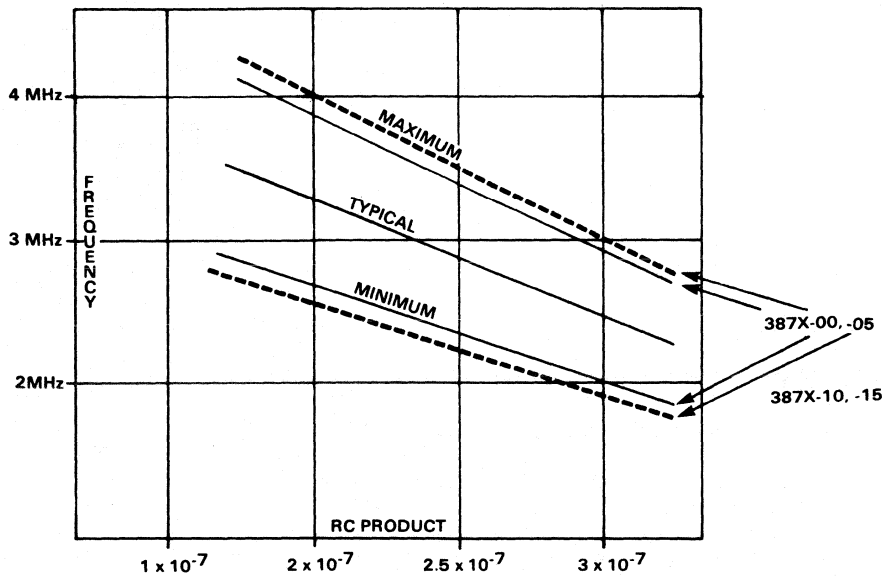
Variations in frequency due to variations in RC components may be calculated as follows:

Maximum RC = (R max) (C external max + C_{XTL} max)

Minimum RC = (R min) (C external min + C_{XTL} min)

$$\text{Typical RC} = (\text{R typ})\left(\text{C external typ} + \frac{\{C_{XTL}\ \text{max} + C_{XTL}\ \text{min}\}}{2}\right)$$

FREQUENCY VS. RC
Figure 18



Positive Freq. Variation = RC typical - RC minimum / RC typical

Negative Freq. Variation = RC maximum - RC typical / RC typical
due to RC Components

Total frequency variation due to all factors:

387X-00, -05
= +18 percent plus positive frequency variation due to RC components

= -18 percent minus negative frequency variation due to RC components

387X-10, -15
= +21 percent plus positive frequency variation due to RC components

= -21 percent minus negative frequency variation due to RC components

Total frequency variation due to V_{CC} and temperature of a unit tuned to frequency at +5V V_{CC}, 25 C

387X-00, -05
= + 13 percent

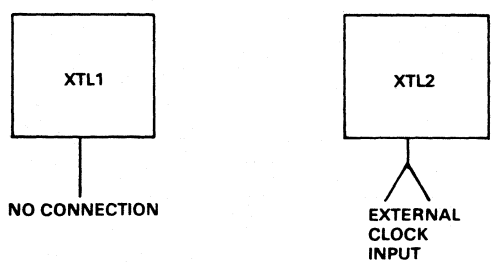
387X-10, -15
= + 16 percent

EXTERNAL CLOCK CONFIGURATION

The connection for the external clock time base configuration is shown in Figure 19. Refer to the DC Characteristics section for proper input levels and current requirements.

Refer to the Capacitance section of the appropriate 3873 Family device data sheet for input capacitance.

EXTERNAL MODE CONNECTION
Figure 19



OPERATING VOLTAGES AND TEMPERATURES

Dash Number Suffix	Operating Voltage V_{CC}	Operating Temperature T_A
—00	+5V ± 10%	0°C - 70°C
—05	+5V ± 5%	0°C - 70°C
—10	+5V ± 10%	-40°C - +85°C
—15	+5V ± 5%	-40°C - +85°C

ABSOLUTE MAXIMUM RATINGS*

	-00,-05	-10,-15
Temperature Under Bias	-20°C to +85°C	-50°C to +100°C
Storage Temperature	-65°C to +150°C	-65°C to +150°C
Voltage on any Pin With Respect to Ground (Except open drain pins and TEST)	-1.0V to +7V	-1.0V to +7V
Voltage on TEST with Respect to Ground	-1.0V to +9V	-1.0V to +9V
Voltage on Open Drain Pins With Respect to Ground	-1.0V to +13.5V	-1.0V to + 13.5V
Power Dissipation	1.5W	1.5W
Power Dissipation by any one I/O pin[2]	60mW	60mW
Power Dissipation by all I/O pins[2]	600mW	600mW

*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.

AC CHARACTERISTICS

T_A, V_{CC} within specified operating range.
I/O Power Dissipation ≤ 100mW (Note 2)

SIGNAL	SYM	PARAMETER	-00,-05 MIN	-00,-05 MAX	-10,-15 MIN	-10,-15 MAX	UNIT	NOTES
XTL1 XTL2	t_O	Time Base Period, all clock modes	250	500	250	500	ns	4MHz-2MHz
	$t_{ex(H)}$	External clock pulse width high	90	400	100	390	ns	
	$t_{ex(L)}$	External clock pulse width low	100	400	110	390	ns	
Φ	t_Φ	Internal Φ clock	$2t_O$		$2t_O$			
WRITE	t_W	Internal WRITE Clock period	4tΦ 6tΦ		4tΦ 6tΦ			Short Cycle Long Cycle
I/O	$t_{dI/O}$	Output delay from internal WRITE clock	0	1000	0	1200	ns	50pF plus one TTL load
	$t_{sI/O}$	Input setup time to internal WRITE clock	1000		1200		ns	
$\overline{STROBE}$	$t_{I/O-s}$	Output valid to $\overline{STROBE}$ delay	3tΦ -1000	3tΦ +250	3tΦ -1200	3tΦ +300	ns	I/O load = 50pF + 1 TTL load
	t_{sL}	$\overline{STROBE}$ low time	8tΦ -250	12tΦ +250	8tΦ -300	12tΦ +300	ns	$\overline{STROBE}$ load = 50pF+3TTL loads
$\overline{RESET}$	t_{RH}	$\overline{RESET}$ hold time, low	6tΦ +750		6tΦ +1000		ns	
	t_{RPOC}	$\overline{RESET}$ hold time, low for power clear	power supply rise time -0.1		power supply rise time -0.15		ms	
EXT INT	t_{EH}	EXT INT hold time in active and inactive state	6tΦ +750		6tΦ +1000		ns	To trigger interrupt
			2tΦ		2tΦ		ns	To trigger timer

CAPACITANCE
T_A = 25°C
All Part Numbers

SYM	PARAMETER	MIN	MAX	UNIT	NOTES
C_{IN}	Input capacitance; I/O, $\overline{\text{RESET}}$, EXT INT, TEST		10	pF	unmeasured pins grounded
C_{XTL}	Input capacitance; XTL1, XTL2	23.5	29.5	pF	

AC CHARACTERISTICS FOR SERIAL I/O PINS
T_A, V_{CC} within specified operating range.
I/O Power Dissipation ≤ 100mW (Note 2)

SIGNAL	SYM	PARAMETER	-00, -05		-10, -15		UNIT	CONDITIONS
			MIN	MAX	MIN	MAX		
SRCLK	$t_{C(SRCLK)}$	Serial Clock Period in External Clock Mode	3.25	∞	3.25	∞	μs	
	$t_{W(SRCLKH)}$	Serial Clock Pulse Width, High. External Clock Mode	1.3	∞	1.3	∞	μs	
	$t_{W(SRCLKL)}$	Serial Clock Pulse Width, Low. External Clock Mode	1.3	∞	1.3	∞	μs	
	$t_{r(SRCLK)}$	Serial Clock Rise Time Internal Clock Mode		60		100	ns	0.8V –2.0V C_L = 100pf
	$t_{f(SRCLK)}$	Serial Clock Fall Time Internal Clock Mode		30		50	ns	2.4V –0.4V C_L = 100pf
SI	$t_{S(SI)}$	Setup Time To Rising Edge of SRCLK (SYNC Mode)	0		0		ns	
	$t_{H(SI)}$	Hold Time From Rising Edge of SRCLK (SYNC Mode)	1500		1500		ns	
SO	$t_{D(SO)}$	Data Output Delay From Falling Edge of SRCLK (SYNC Mode)		1190		1190	ns	

AC CHARACTERISTICS FOR MK38P73

(Signals brought out at socket)
T_A, V_{CC} within specified operating range.
I/O Power Dissipation ≤ 100mW (Note 2)

SYMBOL	PARAMETER	-00, -05		-10, -15		UNIT	CONDITION
		MIN	MAX	MIN	MAX		
t_{aas}*	Access time from Address A_{11}-A_{01} stable until data must be valid at D_7-D_0	650		650		ns	Φ = 2.0MHz

*See Table in Figure 13.

DC CHARACTERISTICS

T_A, V_{CC} within specified operating range
I/O Power Dissipation ≤ 100mW (Note 2)

SYM	PARAMETER	-00,-05		-10,-15		UNIT	DEVICE
		MIN	MAX	MIN	MAX		
I_{CC}	Average Power Supply Current		93.5		121	mA	MK3873/10 Outputs Open
			103		138	mA	MK3873/12 Outputs Open
			93.5		121	mA	MK3873/20 Outputs Open
			103		138	mA	MK3873/22 Outputs Open
			138		165	mA	MK38P73/02 No EPROM, Outputs Open
P_D	Power Dissipation		440		570	mW	MK3873/10 Outputs Open
			485		645	mW	MK3873/12 Outputs Open
			440		570	mW	MK3873/20 Outputs Open
			485		645	mW	MK3873/22 Outputs Open
			646		775	mW	MK38P73/02 No EPROM, Outputs Open

DC CHARACTERISTICS

T_A, V_{CC} within specified operating range
I/O Power Dissipation ≤ 100mW (Note 2)

SYM	PARAMETER	-00,-05		-10,-15		UNIT	CONDITIONS
		MIN	MAX	MIN	MAX		
V_{IHEX}	External Clock input high level	2.4	5.8	2.4	5.8	V	
V_{ILEX}	External Clock input low level	-.3	.6	-.3	.6	V	
I_{IHEX}	External Clock input high current		100		130	μA	$V_{IHEX}=V_{CC}$
I_{ILEX}	External Clock input low current		-100		-130	μA	$V_{ILEX}=V_{SS}$
$V_{IHI/O}$	I/O input high level	2.0	5.8	2.0	5.8	V	standard pull-up (1)
		2.0	13.2	2.0	13.2	V	open drain (1)
V_{IHR}	Input high level, $\overline{RESET}$	2.0	5.8	2.2	5.8	V	standard pull-up (1)
		2.0	13.2	2.2	13.2	V	No pull-up
V_{IHEI}	Input high level, EXT INT	2.0	5.8	2.2	5.8	V	standard pull-up (1)
		2.0	13.2	2.2	13.2	V	No pull-up
V_{IL}	I/O ports, $\overline{RESET}$[1], EXT INT[1] input low level	-.3	.8	-.3	0.7	V	(1)
I_{IL}	Input low current, standard pull-up pins		-1.6		-1.9	mA	$V_{IN}=0.4V$
I_L	Input leakage current, open drain pins $\overline{RESET}$ and EXT INT inputs With no pull-up resistor		+10 -5		+18 -8	μA μA	$V_{IN}=13.2V$ $V_{IN}=0.0V$
I_{OH}	Output high current, standard	-100		-89		μA	$V_{OH}=2.4V$
	pull-up pins	-30		-25		μA	$V_{OH}=3.9V$
I_{OHDD}	Output high current,	-100		-80		μA	$V_{OH}=2.4V$
	direct drive pins	-1.5	 -8.5	-1.3	 -11	mA mA	$V_{OH}=1.5V$ $V_{OH}=0.7V$
I_{OL}	Output low current, I/O ports	1.8		1.65		mA	$V_{OL}=0.4V$
I_{OHS}	$\overline{STROBE}$ Output High current	-300		-270		μA	$V_{OH}=2.4V$
I_{OLS}	$\overline{STROBE}$ output low current	5.0		4.5		mA	$V_{OL}=0.4V$

DC CHARACTERISTICS FOR MK38P73

(Signals brought out at socket)

T_A, V_{CC} within specifiec operating range, I/O Power Dissipation ≤ 100mW. (Note 2)

SYM	PARAMETER	-00, -05		-10, -15		UNIT	CONDITION
		MIN	MAX	MIN	MAX		
I_{CCE}	Power Supply Current for EPROM		-185		-185	mA	
V_{IL}	Input Low Level Data bus in	-0.3	0.8	-0.3	0.7	V	
V_{IH}	Input High Level Data bus in	2.0	5.8	2.0	5.8	V	
I_{OH}	Output High Current	-100 -30		-90 -25		μA μA	V_{OH}=2.4V V_{OH}=3.9V
I_{OL}	Output Low Current	1.8		1.65		mA	V_{OL}=0.4V
I_{IL}	Input Leakage Current		10		10	μA	Data Bus in Float

DC CHARACTERISTICS FOR SERIAL PORT I/O PINS

T_A, V_{CC} within specified operating range

I/O Power Dissipation ≤ 100mW (Note 2)

SYM	PARAMETER	-00, -05		-10, -15		UNIT	TEST CONDITIONS
		MIN	MAX	MIN	MAX		
V_{IHS}	Input High for SI, SRCLK	2.0	5.8	2.0	5.8	V	
V_{ILS}	Input Low level for SI, SRCLK	-.3	.8	-.3	0.7	V	
I_{ILS}	Input low current for SI, SRCLK		-1.6		-1.9	mA	V_{IL} = 0.4V
I_{OHSO}	Output High Current SO	-100 -30		-90 -25		μA μA	V_{OH} = 2.4V V_{OL} = 3.9V
I_{OLSO}	Output Low Current SO	1.8		1.65		mA	V_{OL} = 0.4V
I_{OHSRC}	Output High Current SRCLK	-300		-270		μA	V_{OH} = 2.4V
I_{OLSRC}	Output Low Current	5.0		4.5		mA	V_{OL} = 0.4V

1. $\overline{\text{RESET}}$ and EXT INT have internal Schmit triggers giving minimum .2V hysteresis.
2. Power dissipation for I/O pins is calculated by $\Sigma (V_{CC} - V_{IL})(|I_{IL}|) + \Sigma (V_{CC} - V_{OH})(|I_{OH}|) + \Sigma (V_{OL})(I_{OL})$

TIMER AC CHARACTERISTICS

Definitions:

Error = Indicated time value - actual time value

tpsc = t Φ x Prescale Value

Interval Timer Mode:

Single interval error, free running (Note 3) ..... ±6tΦ
Cumulative interval error, free running (Note 3) 0
Error between two Timer reads (Note 2) ..... ± (tpsc + tΦ
Start Timer to stop Timer error (Notes 1,4) ..... +tΦ to —(tpsc +tΦ)
Start Timer to read Timer error (Notes 1,2) ..... —5tΦ to —(tpsc + 7tΦ)
Start Timer to interrupt request error (Notes 1,3) ..... —2tΦ to —8tΦ)
Load Timer to stop Timer error (Note 1) ..... +tΦ to —(tpsc + 2tΦ)
Load Timer to read Timer error (Notes 1,2) ..... —5tΦ to —(tpsc + 8tΦ)
Load Timer to interrupt request error (Notes 1,3) ..... —2tΦ to —9tΦ)

Pulse Width Measurement Mode:

Measurement accuracy (Note 4)	+t Φ to —(tpsc +2t Φ)
Minimum pulse width of EXT INT pin	2tΦ

Event Counter Mode:

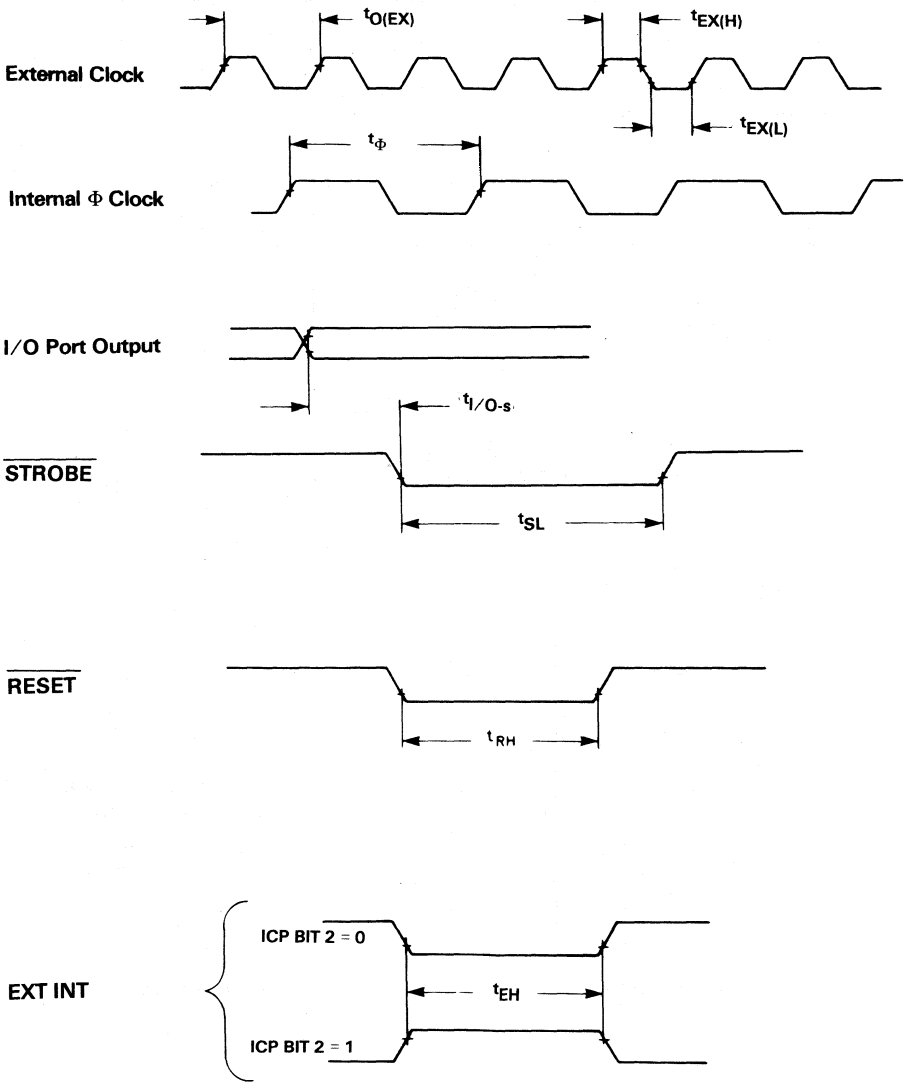
Minimum active time of EXT INT pin	2tΦ
Minimum inactive time of EXT INT pin	2tΦ

Notes:

1. All times which entail loading, starting, or stopping the Timer are referenced from the end of the last machine cycle of the OUT or OUTS instruction.
2. All times which entail reading the Timer are referenced from the end of the last machine cycle of the IN or INS instruction.
3. All times which entail the generation of an interrupt request are referenced from the start of the machine cycle in which the appropriate interrupt request latch is set. Additional time may elapse if the interrupt request occurs during a privileged or multicycle instruction.
4. Error may be cumulative if operation is repetitively performed.

AC TIMING DIAGRAM

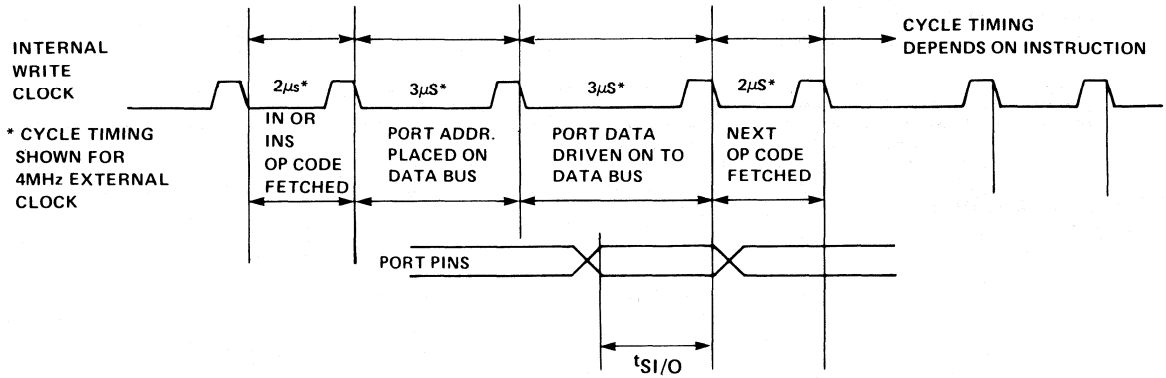
Figure 20



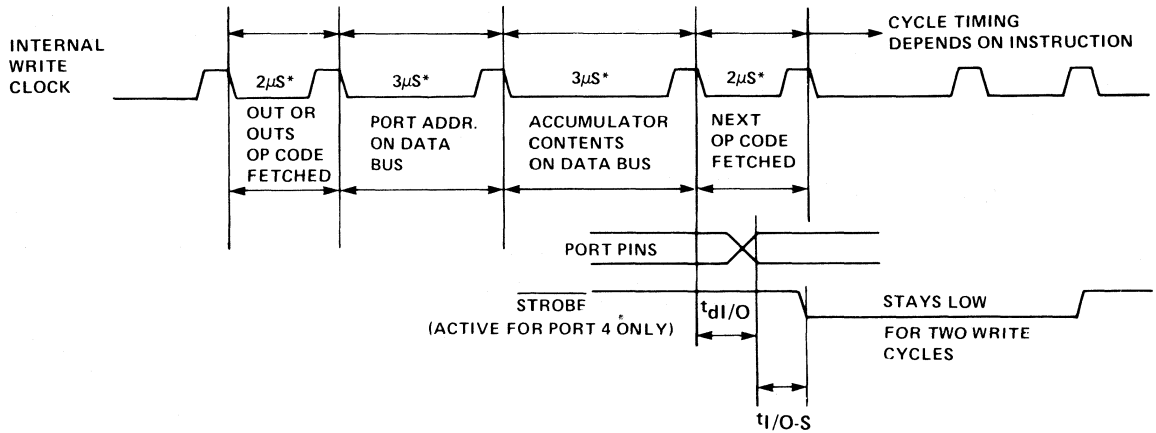
Note: All AC measurements are referenced to V_{IL} max., V_{IH} min., V_{OL} (.8v), or V_{OH} (2.0v).

INPUT/OUTPUT AC TIMING

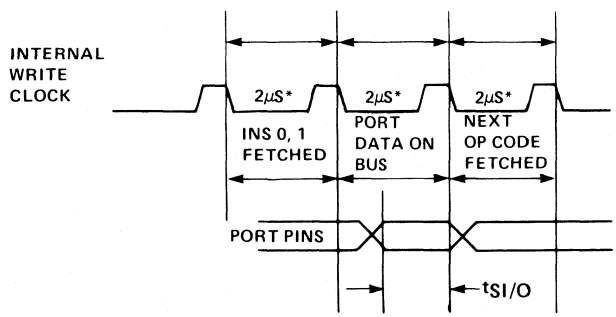
Figure 21



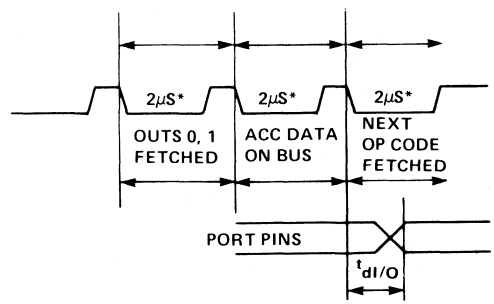
A. INPUT ON PORT 4 OR 5



B. OUTPUT ON PORT 4 OR 5



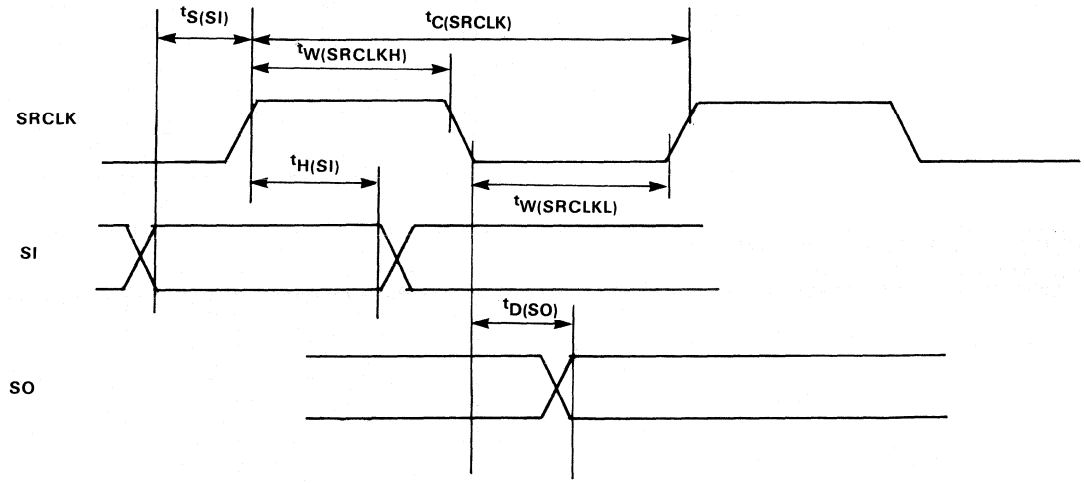
C. INPUT ON PORT 0 OR 1



D. OUTPUT ON PORT 0, 1

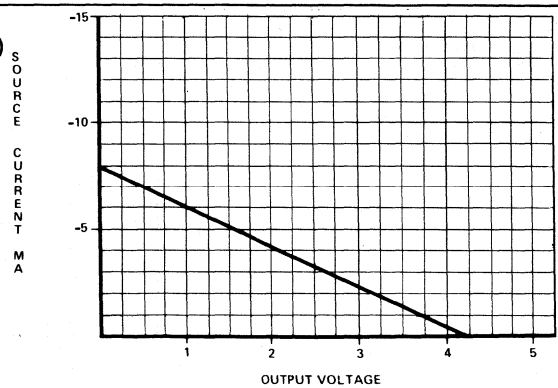
AC TIMING DIAGRAM FOR SERIAL I/O PINS.

Figure 22



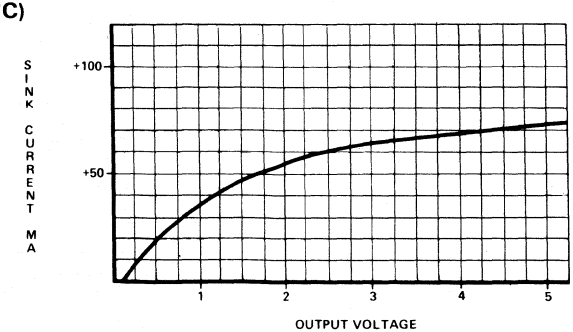
STROBE SOURCE CAPABILITY (TYPICAL AT V_{CC} = 5V, T_A = 25°C)

Figure 23



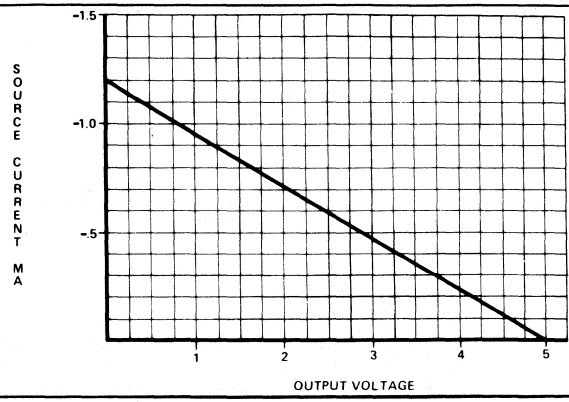
STROBE SINK CAPABILITY (TYPICAL AT V_{CC} = 5V, T_A = 25°C)

Figure 24



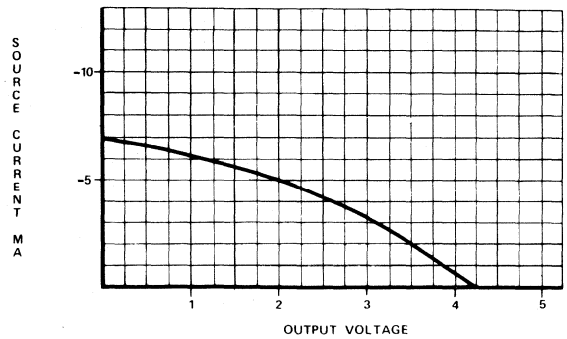
STANDARD I/O PORT SOURCE CAPABILITY (TYPICAL AT V_{CC} = 5V, T_A = 25°C)

Figure 25



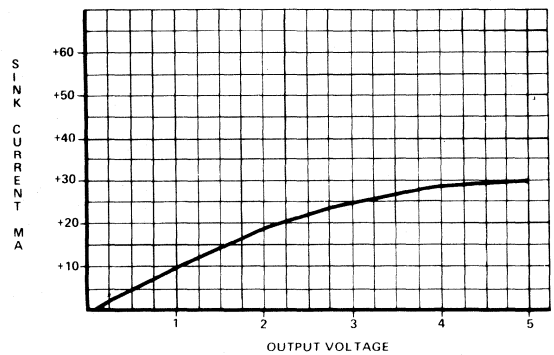
DIRECT DRIVE I/O PORT SOURCE CAPABILITY (TYPICAL AT V_{CC} = 5V, T_A = 25°C)

Figure 26



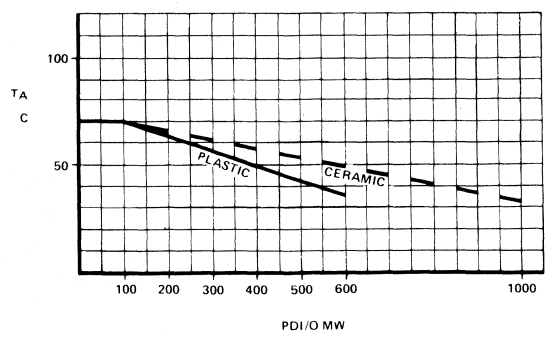
I/O PORT SINK CAPABILITY (TYPICAL AT V_{CC} = 5V, T_A = 25°C)

Figure 27



MAXIMUM OPERATING TEMPERATURE VS. I/O POWER DISSIPATION

Figure 28

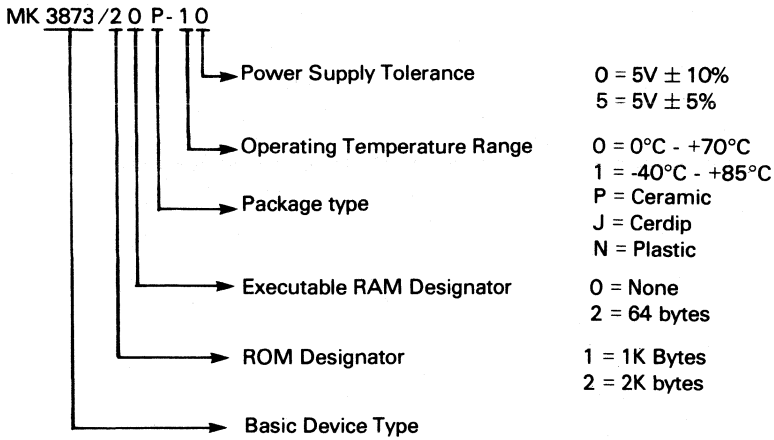


ORDERING INFORMATION

There are two types of part numbers for the 3870 family of devices. The generic part number describes the basic device type, the amount of ROM and Executable RAM, the desired package type, temperature range, and power supply tolerance. For each customer specific code, additional information defining I/O options and oscillator options will be combined with the information described in the generic part number to define a customer/code specific device order number.

GENERIC PART NUMBER

An example of the generic part number is shown below.

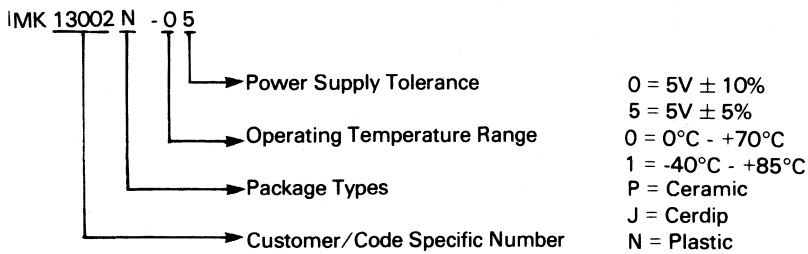


An example of the generic part number for the PPROM device is shown below.

MK38P73/02 R-05

DEVICE ORDER NUMBER

An example of the device order number is shown below.



The Customer/Code specific number defines the ROM bit pattern, I/O configuration, oscillator type, and generic part type to be used to satisfy the requirements of a particular customer purchase order. For further information on the ordering of mask ROM devices, the customer should refer to the 3870 Family Technical Manual.

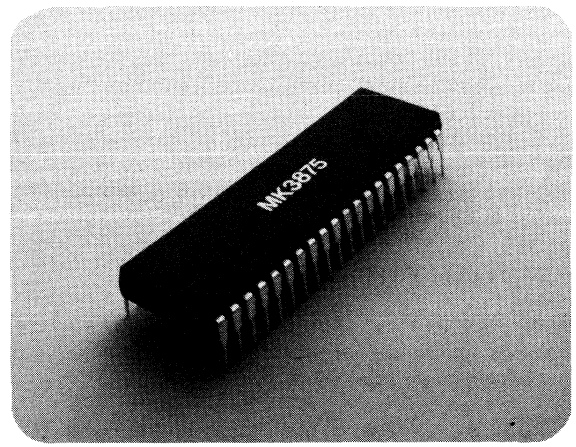
MOSTEK

3870 SINGLE CHIP MICRO FAMILY

MK3875

MK3875 FEATURES

- ☐ Available with 2K or 4K bytes of mask programmable ROM memory.
- ☐ 64 bytes scratchpad RAM
- ☐ 64 bytes of Executable RAM
- ☐ Standby feature for low power data retention of executable RAM including:
  Low standby power,
  Low standby supply voltage
  No external components required to trickle charge battery.
- ☐ Software compatible with 3870 family.
- ☐ 30 bits (4 ports) TTL compatible I/O
- ☐ Programmable Binary Timer
  Interval Timer Mode
  Pulse Width Measurement Mode
  Event Counter Mode
- ☐ External Interrupt Input
- ☐ Crystal, LC, RC, or external time base options available.
- ☐ Low power under normal operation (285 mW typ.)
- ☐ +5 volt main power supply.
- ☐ Pinout compatible with 3870 family.

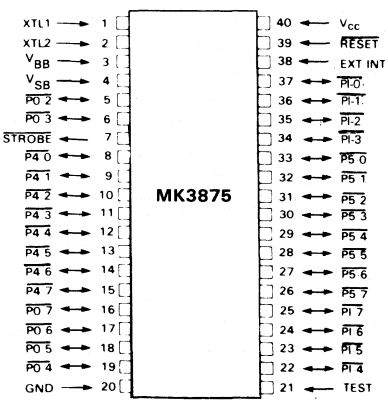


III
3870 SINGLE CHIP MICROCOMPUTER FAMILY

GENERAL DESCRIPTION

The MK3875 Single Chip Microcomputer offers a Low Power Standby mode of operation as an addition to the 3870 Family. The Low Power Standby feature provides a means of retaining data in the executable RAM on the MK3875 while the main power supply line (V_{CC}) is at 0 volts and the rest of the MK3875 microcomputer is shut down. The executable RAM is powered from an auxiliary power supply input (V_{SB}) while operating in the Lower Power Standby mode. When V_{SB} is maintained at or above

PIN CONNECTIONS
MK3875



PIN NAME	DESCRIPTION	TYPE
$\overline{P0\text{-}2}$ - $\overline{P0\text{-}7}$	I/O Port 0	Bidirectional
$\overline{P1\text{-}0}$ - $\overline{P1\text{-}7}$	I/O Port 1	Bidirectional
$\overline{P4\text{-}0}$ - $\overline{P4\text{-}7}$	I/O Port 4	Bidirectional
$\overline{P5\text{-}0}$ - $\overline{P5\text{-}7}$	I/O Port 5	Bidirectional
$\overline{STROBE}$	Ready Strobe	Output
EXT INT	External Interrupt	Input
$\overline{RESET}$	External Reset	Input
TEST	Test Line	Input
XTL 1, XTL 2	Time Base	Input
V_{CC}, GND	Power Supply Lines	Input
V_{SB}	Standby Power	Input
V_{BB}	Substrate Decoupling	Input

its minimum level, data is retained in the executable RAM memory with a very low power dissipation.

The MK3875 retains commonality with the rest of the industry standard 3870 family of single chip microcomputers. It has the same central processing unit, oscillator and clock circuits, and 64 byte scratchpad memory array. Also, the 3870's sophisticated programmable binary timer is included which provides three different operating modes. Two pins on the MK3875 are dedicated to the Low Power Standby mode and are designated as V_{SB} and V_{BB}. The $\overline{\text{RESET}}$ line serves to reset the MK3875 and place it in a protected state so that the contents of the Executable RAM will remain unchanged when V_{CC} is being powered down to 0 volts. All other pins on the MK3875 are identical in function to corresponding pins on the MK3870, so that pin compatibility is maintained. The MK3875 executes the entire 3870 instruction set.

FUNCTIONAL PIN DESCRIPTION

$\overline{\text{P0-2}}$ - $\overline{\text{P0-7}}$, $\overline{\text{P1-0}}$ - $\overline{\text{P1-7}}$, $\overline{\text{P4-0}}$ - $\overline{\text{P4-7}}$, and $\overline{\text{P5-0}}$ - $\overline{\text{P5-7}}$ are 30 lines which can be individually used as either TTL compatible inputs or as latched outputs.

$\overline{\text{STROBE}}$ is a ready strobe associated with I/O Port 4. This pin, which is normally high, provides a single low pulse after valid data is present on the $\overline{\text{P4-0}}$ - $\overline{\text{P4-7}}$ pins during an output instruction.

$\overline{\text{RESET}}$ - may be used to externally reset the MK3875. When pulled low, the MK3875 will reset. When allowed to go high the MK3875 will begin program execution at program location H '000'. Additionally, when $\overline{\text{RESET}}$ is brought low all accesses of the executable RAM are prevented and the RAM is placed in a protected state for powering down V_{CC} without loss of data.

EXT INT is the external interrupt input. Its active state is software programmable. This input is also used in conjunction with the timer for pulse width measurement and event counting.

XTL 1 and XTL 2 are the time base inputs to which a crystal (2 to 4MHz), LC network, RC network, or an external single-phase clock may be connected. The time base network must be specified when ordering an MK3875.

TEST is an input, used only in testing the MK3875. For normal circuit functionality this pin may be left unconnected but it is recommended that TEST be grounded.

V_{CC} is the power supply input +5V.

V_{SB} is the RAM standby power supply input.

V_{BB} is the substrate decoupling pin. A .01 micro-Farad capacitor is required which is tied between V_{BB} and GND.

MK3875 ARCHITECTURE

The basic functional elements of the mask ROM MK3875 single chip microcomputer are shown in the block diagram in Figure 1. A programming model is shown in Figure 2. Much of the MK3875 architecture is identical with the rest of the devices in the 3870 family. The significant features of the MK3875 are discussed in the following sections. The user is referred to the 3870 Family Technical Manual for a thorough discussion of the architecture, instruction set, and other features which are common to the 3870 family.

MAIN MEMORY

The main memory section on the MK3875 consists of a combination of ROM and executable RAM.

There are four registers associated with the main memory section. These are the Program Counter (P0), the Stack Register (P), the Data Counter (DC) and the Auxiliary Data Counter (DC1). The Program Counter is used to address instructions during program execution. P is used to save the contents of P0 during an interrupt or subroutine call. Thus, P contains the return address at which processing is to resume upon completion of the subroutine or the interrupt routine.

The Data Counter (DC) is used to address data tables. This register is auto-incrementing. Of the two data counters only DC can access the memory. However, the XDC instruction allows DC and DC1 to be exchanged.

The length of the P0, P, DC, and DC1 registers for all MK3875 devices is 12 bits. Figure 3 shows the amounts of ROM and Executable RAM for each device in the MK3875 family.

EXECUTABLE RAM

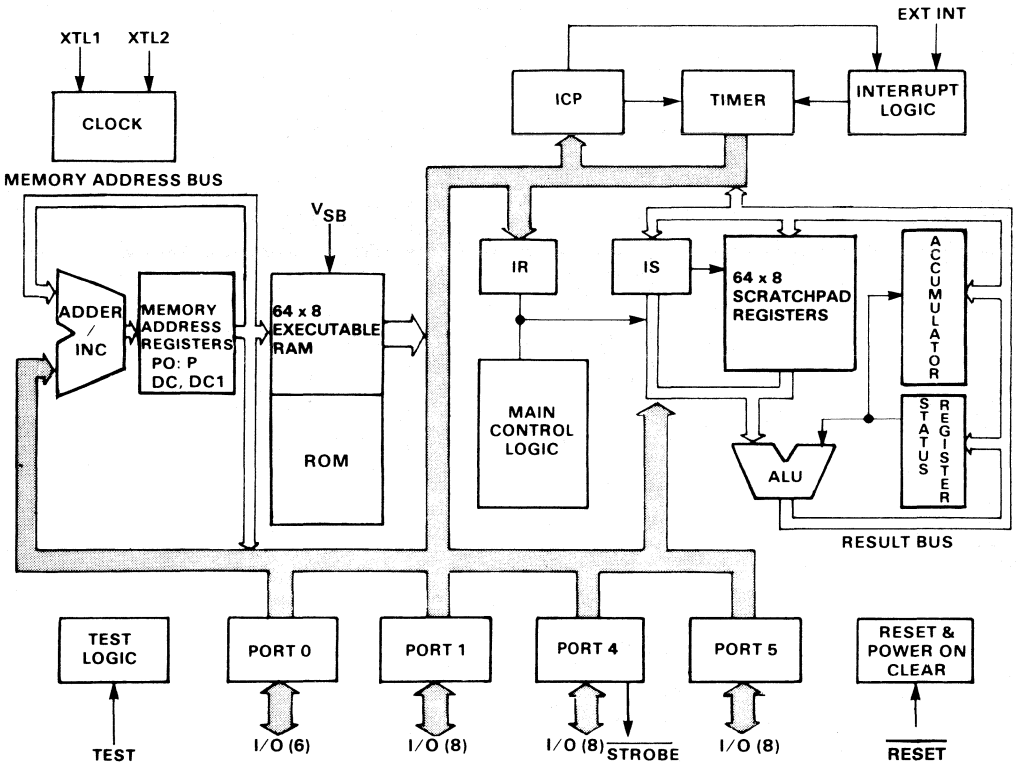
The upper bytes of the total address space in all MK3875 devices is RAM memory. As with the ROM memory, the RAM may be addressed by the P0 and DC address registers. The executable RAM may be accessed by all 3870 instructions which address main memory indirectly through the Data Counter (DC) register. Additionally, the MK3875 may execute an instruction sequence which resides in the executable RAM. Note that this cannot be done with the scratchpad RAM memory, which is the reason the term "executable RAM" is given to this additional memory. The contents of the executable RAM memory are preserved when the Low Power Standby mode is in operation.

I/O PORTS

The MK3875 provides 30 bits of bidirectional parallel I/O. These lines are addressed as Ports 0, 1, 4, and 5. In addition, the Interrupt Control Port is addressed as Port 6 and the binary timer is addressed as Port 7. The programming of Ports 6 and 7 and the bidirectional I/O pins are covered in the 3870 Family Technical Manual.

MK3875 BLOCK DIAGRAM

Figure 1



Since two pins are dedicated to serve the Standby Power mode (V_{SB} and V_{BB}), port 0 has only the upper 6 bits, $\overline{P0\text{-}2}$ -$\overline{P0\text{-}7}$, available for use as general purpose I/O pins. Ports 1, 4, and 5 are all a full 8 bits wide.

The schematic of an I/O pin and available output drive options are shown in Figure 4.

An output ready strobe is associated with Port 4. This flag may be used to signal a peripheral device that the MK3875 has just completed an output of new data to Port 4. The strobe provides a single low pulse shortly after the output operation is completely finished, so either edge may be used to signal the peripheral. $\overline{STROBE}$ may be used as an input strobe simply by doing a dummy output of H '00' to Port 4 after completing the input operation.

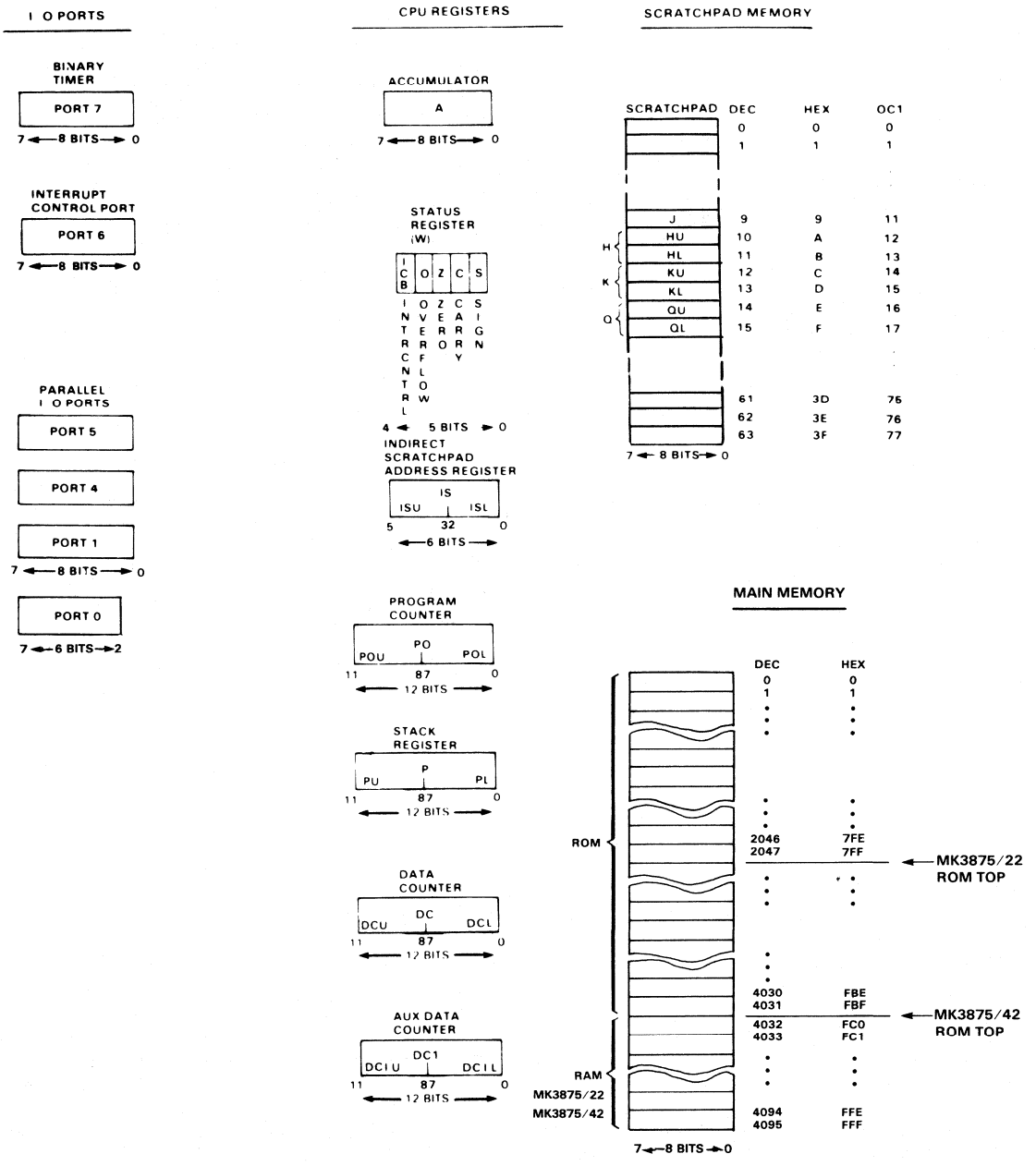
STANDBY POWER MODE

On the MK3875, the contents of the on-chip executable RAM can be saved when the Standby Power mode is operative. The Standby Power mode allows the MK3875's main power supply to drop all the way down to 0 volts while the on-chip executable RAM is powered from the auxilliary low power supply input, V_{SB}. Thus, key variables may be maintained within the MK3875 executable RAM during the time that the rest of the microcomputer is powered down.

On the MK3875, two of the pins which are used as bidirectional port pins on the MK3870 are used for the Standby Power feature. Port 0, Bit 0 (P0-0) remains readable and writeable although it is not connected to a package pin. The logic level being applied to the auxiliary

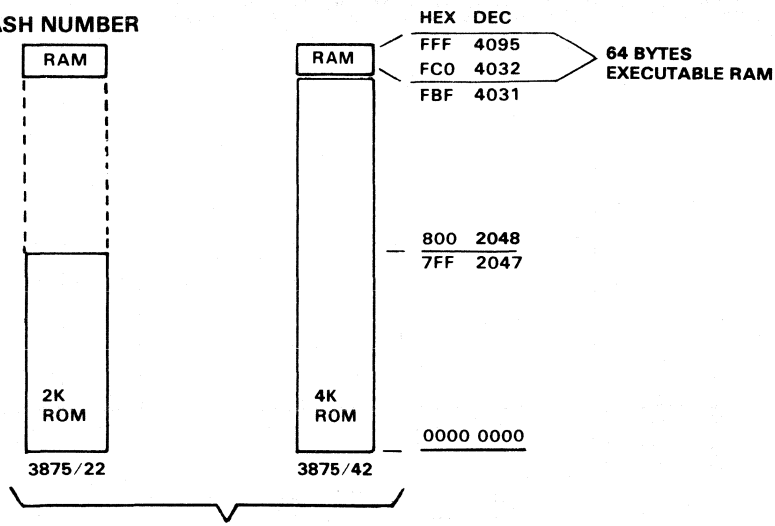
3875 PROGRAMMABLE REGISTERS, PORTS AND MEMORY MAP

Figure 2



MK3875 MAIN MEMORY SIZES AND TYPES BY SLASH NUMBER

Figure 3



All devices contain 64 bytes of scratchpad RAM

Data derived from addressing any locations other than within the specified ROM or RAM space is not tested nor is it guaranteed. Users should refrain from entering this area of the memory map.

Device	Scratchpad RAM Size (Decimal)	Address Register Size (P0,P,DC,DC1)	ROM Size (Decimal)	Executable RAM Size
MK3875/22	64 bytes	12 bits	2048 bytes	64 bytes
MK3875/42	64 bytes	12 bits	4032 bytes	64 bytes

power supply input (V_{SB}) can be read at Port 0, Bit 1 ($\overline{PO\text{-}1}$). Writing to $\overline{PO\text{-}1}$ has no effect.

A capacitor (.01 microfarads) must be connected between pin 3 (V_{BB}) and ground. V_{BB} is bonded directly to the substrate of the MK3875. The purpose of the capacitor is to decouple noise on the substrate of the circuit when V_{CC} is switched on and off.

It is recommended that Nickel Cadmium batteries (typical voltage of 3 series cells = 3.6V) be used for standby power, since the MK3875 can automatically trickle charge the three NiCads. If more than three cells in series are used, the charging circuit must be provided outside the MK3875.

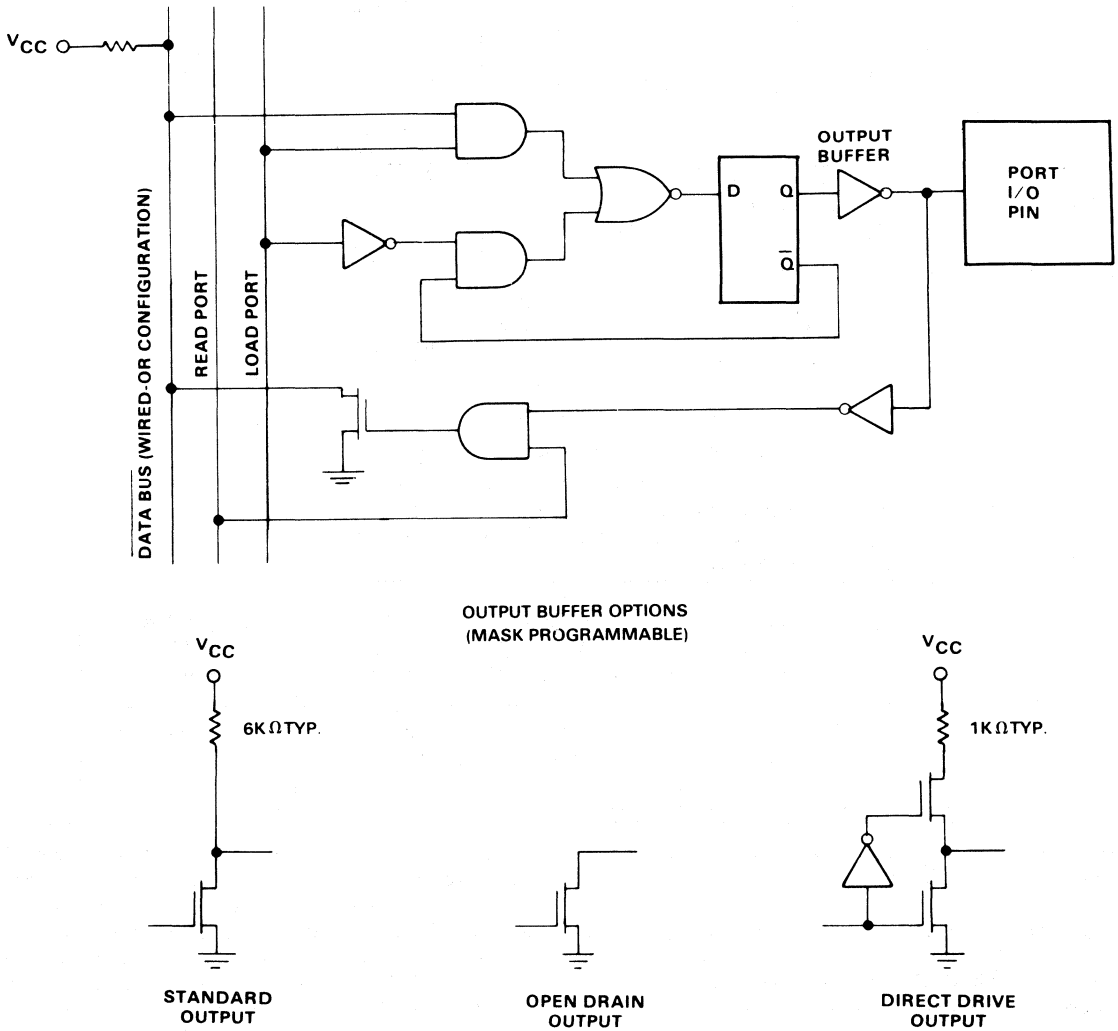
Whenever $\overline{RESET}$ is brought low, the executable RAM is placed in a protected state. Also the RAM is switched from V_{CC} power to the V_{SB} power. When powering down, it may be desirable to interrupt the MK3875 when an impending power down condition is detected, so that the necessary data can be saved before V_{CC} falls below the minimum level. After the save is completed, $\overline{RESET}$ can fall, which prevents any further access of the RAM. The timing for this power down sequence is illustrated in Figure 5A.

A second power down sequence is illustrated in Figure 5B, and may be used if a special save data routine is not needed. The EXT INT line need not be used. Note that for both cases shown in Figures 5A and 5B, $\overline{RESET}$ must be low before V_{CC} drops below the minimum specified operating voltage for the MK3875. This is to insure that the contents of the executable RAM is not altered during the power down sequence.

There may be a set of variables stored in the RAM memory which is continually updated during the time when the MK3875 is in its normal operating mode. If a particular variable occupies more than one byte of RAM, there can be a problem if a reset occurs in response to an impending power down condition during the time that the multi-byte variable was being modified. If such a reset occurs, then only part of the variable may contain the updated value, while the rest contains the old value. An example of this case would be when a double precision (2 byte) binary number is being saved in the executable RAM. Suppose that a new value of the number has been calculated in the program, and that this new value is to replace the old value contained in the executable RAM. Note that a reset could occur just after the program wrote one byte of the new value

I/O PIN CONCEPTUAL DIAGRAM WITH OUTPUT BUFFER OPTIONS

Figure 4



Ports 0 and 1 are Standard Output type only.

Ports 4 and 5 may both be any of the three output options (mask programmable bit by bit)

The $\overline{\text{STROBE}}$ output is always configured similar to a Direct Drive Output except that it is capable of driving 3 TTL loads.

$\overline{\text{RESET}}$ and EXT INT may have standard 6KΩ (typical) pull-up or may have no pull-up (mask programmable). These two inputs have Schmitt trigger inputs with a minimum of 0.2 volts of hysteresis.

into the RAM. When power is restored following the Standby Power mode, the double precision variable would contain an erroneous value.

This problem can be avoided if the external interrupt is used to signal the MK3875 of an impending power down condition. The user's system should be designed so that the MK3875 can properly save all variables between the time that the external interrupt occurs and $\overline{\text{RESET}}$ falls. If multi-byte variables must be saved during the Standby Power mode and it is not desirable to use the external interrupt in the manner described above, then each byte of a multi-byte variable may be kept with an associated flag. The method of updating a two byte variable would be as follows:

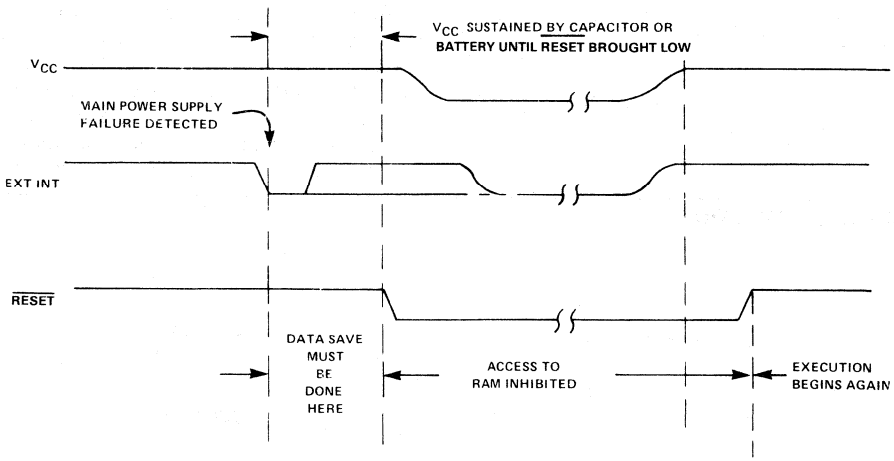
- Clear Flag Word 1
- Update Byte 1
- Set Flag Word 1
- Clear Flag Word 2
- Update Byte 2
- Set Flag Word 2

Now if $\overline{\text{RESET}}$ goes low during the update of a byte of a variable, the flag word associated with that byte of data will be reset. Any byte of the variable where the flag word is "set" is a good byte of data. While this method significantly encumbers the data storage process, it eliminates the need for a power fail interrupt which both reduces external circuitry and leaves the external interrupt pin completely free for other use.

Often it is necessary to distinguish between an initial power-on condition wherein there is not valid data stored in the RAM (or where V_{SB} has dropped below the minimum required stand-by level) and a re-application of power wherein valid RAM data has been maintained during the power outage. One method of distinguishing between these two conditions is to reserve several memory locations for key words and checksums. When V_{CC} is applied and processor operation begins, these locations can be checked for proper contents. However, this method may not be perfectly accurate as those locations holding key codes may be maintained even though V_{SB} drops below its minimum required level while other RAM locations may lose data, or they could power up with the exact data required to match the key codes. Also, a checksum may be matched on occasion even though RAM data has been corrupted. The accuracy of this method is greatly increased by increasing the number of memory locations used and the variety of key codes and or checksums used.

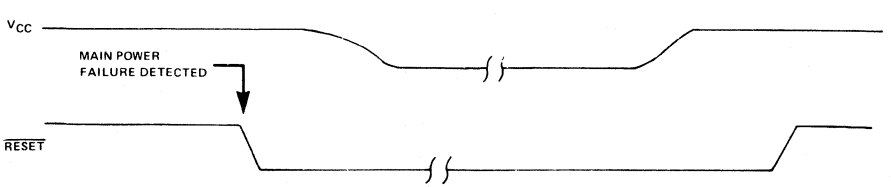
SAVE ROUTINE REQUIRED, $V_{SB} > 3.2$ VOLTS

Figure 5a



NO SAVE ROUTINE REQUIRED, $V_{SB} > 3.2$ VOLTS

Figure 5b

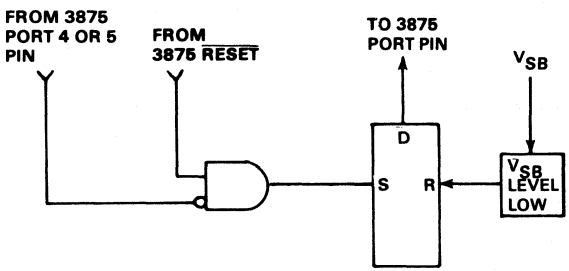


A more reliable method is the external V_{SB} flip-flop. The flip-flop is designed to power up in a known first state and hold that first state until forced into a second state. As long as V_{SB} is above the minimum operating level, the flip-flop can hold the second state but if V_{SB} drops below the minimum level, the flip-flop will flip back to the first state. Thus when power is initially applied or if V_{SB} drops below the minimum level during a V_{CC} outage, the flip-flop will be in the first state. The flip-flop output can be read through a port pin by the processor when processor operation begins to determine whether the RAM data is valid (second state) or invalid (first state). If the flip-flop is found to be in the first state it can be forced to the second state by the processor. If it holds the second state that indicates that V_{SB} is above the minimum level (batteries are charged).

A conceptual diagram is shown in Figure 6.

CONCEPTUAL DIAGRAM

Figure 6



3875 TIME BASE OPTIONS

The 3875 contains an on-chip oscillator circuit which provides an internal clock. The frequency of the oscillator circuit is set from the external time base network. The time base for the 3875 may originate from one of four sources:

1) Crystal
2) LC Network
3) RC Network
4) External Clock

The type of network which is to be used with the mask ROM MK3875 must be specified at the time when mask ROM devices are ordered. However, the MK38P75 may operate with any of the four configurations so that it may emulate any configuration used with a mask ROM device.

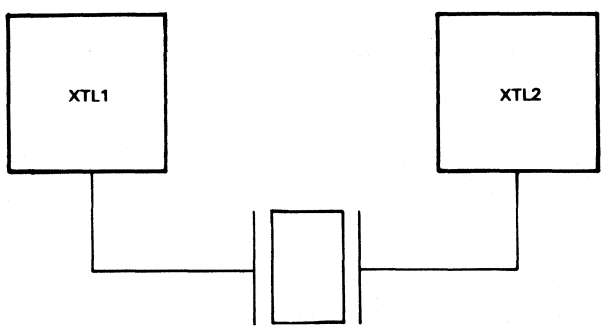
The specifications for the four configurations are given in the following text. There is an internal 26pF capacitor between XTL 1 and GND and an internal 26pF capacitor between XTL 2 and GND. Thus, external capacitors are not necessarily required. In all external clock modes the external time base frequently is divided by two to form the internal PHI clock.

CRYSTAL SELECTION

The use of a crystal as the time base is highly recommended as the frequency stability and reproducability from system to system is unsurpassed. The 3875 has an internal divide by two to allow the use of inexpensive and widely available TV Color Burst Crystals (3.58MHz). Figure 8 lists the required crystal parameters for use with the 3875. The Crystal Mode time base configuration is shown in Figure 7.

CRYSTAL MODE CONNECTION

Figure 7



CRYSTAL PARAMETERS

Figure 8

a) Parallel resonsnace, fundamental mode AT-Cut
b) Shunt capacitance (C_0) = 7 pf max.
c) Series resistance (R_S) = See table
d) Holder = See table below.

Frequency	Series Resistance	Holder
f = 2-2.7 MHz	Rs = 300 ohms max	HC-6 HC-33
f = 2.8-4 MHz	Rs = 150 ohms max	HC-6 HC-18* HC-25* HC-33

*This holder may not be available at frequencies near the lower end of this range.

Through careful buffering of the XTL1 pin it may be possible to amplify this waveform and distribute it to other devices. However, Mostek recommends that a separate active device (such as a 7400 series TTL gate) be used to oscillate the crystal and the waveform from that oscillator be buffered and supplied to all devices, including the 3875, in the event that a single crystal is to provide the time base for more than just a single 3875.

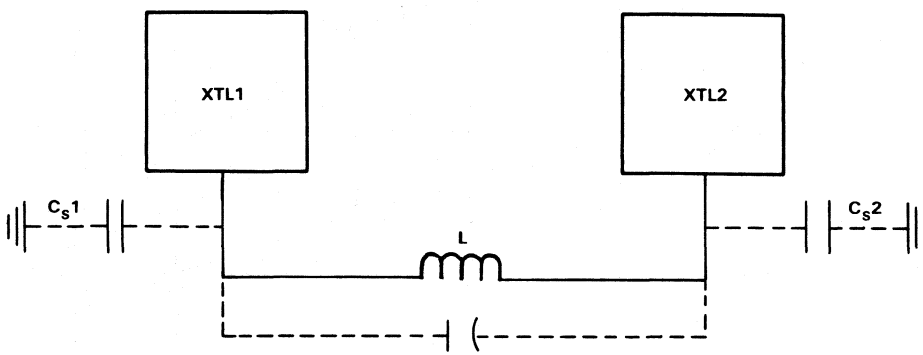
While a ceramic resonator may work with the 3875 crystal oscillator, it was designed specifically to support the use of this component. Thus, Mostek does not support the use of a ceramic resonator either through proper testing, parametric specification, or applications support.

LC NETWORK

The LC time base configuration can be used to provide a less expensive time base for the 3875 than can be provided with a crystal. However, the LC configuration is much less accurate than is the crystal configuration. The LC time base configuration is shown in Figure 9. Also shown in the figure are the specified parameters for the LC components, along with the formula for calculating the resulting time base frequency. The minimum value of the inductor which is required for proper operation of the LC time base network is 0.1 millihenries. The inductor must have a Q factor which is no less than 40. The value of C is derived from C external, the internal capacitance of the 3875, C_{XTL}, and the stray capacitances, C_{S1} and C_{S2}. C_{XTL} is the at XTL1 and capacitance looking into the internal two port network XTL2. C_{XTL} is listed under the "Capacitance" section of the Electrical Specifications. C_{S1} and C_{S2} are stray capacitances from XTL1 to ground and from XTL2 to ground, respectively. C external should also include the stray shunt capacitance across the inductor. This is typically in the 3 to 5 pf range and significant error can result if it is not included in the frequency calculation.

LC MODE CONNECTION

Figure 9

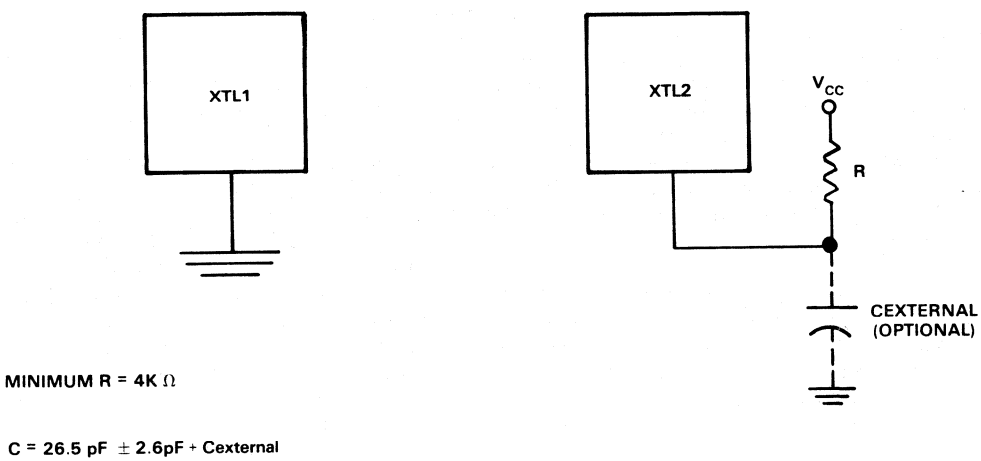


Variation in time base frequency with the LC network can arise from one of four sources: 1) Variation in the value of the inductor. 2) Variation in the value of the external capacitor. 3) Variation in the value of the internal capacitance of the 3875 at XTL1 and XTL2 and 4) Variation in the amount of stray capacitance which exists in the circuit. Therefore, the actual frequency which is generated by the LC circuit is within a range of possible frequencies, where the range of frequencies is determined by the worst case variation in circuit parameters. The designer must select component values such that the range of possible frequencies with the LC mode does not go outside of the specified operating frequency range for the 3875.

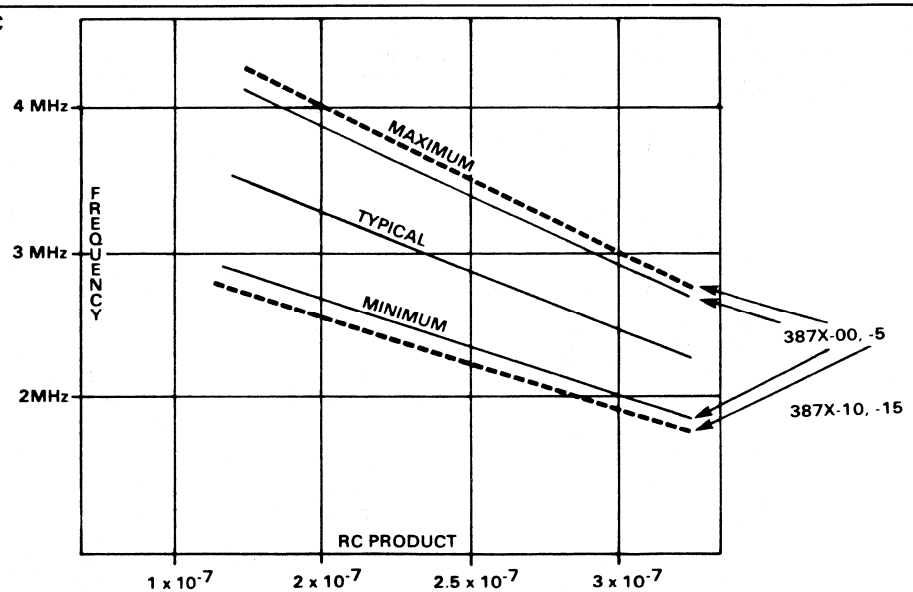
RC CLOCK CONFIGURATION

The time base for the 3875 may be provided from an RC network tied to the XTL2 pin, when XTL1 is grounded. A schematic picturing the RC clock configuration is shown in Figure 10. The RC time base configuration is intended to provide an inexpensive time base source for applications in which timing is not critical. Some users have elected to tune each unit using a variable resistor or external capacitor thus reducing the variation in frequency. However, for increased time base accuracy Mostek recommends the use of the Crystal or LC time base configuration. Figure 11 illustrates a curve which gives the resulting operating frequency for a particular RC value. The x-axis represents the product of the value of the resistor times the value of the capacitor. Note that three curves are actually shown. The curve in the middle represents the nominal frequency obtained for a given value of RC. A maximum curve and a minimum curve for different types of 3875 devices are also shown in the diagram.

RC MODE CONNECTION
Figure 10



FREQUENCY VS. RC
Figure 11



The designer must select the RC product such that a frequency of less than 2 MHz is not possible taking into account the maximum possible RC product and using the minimum curve shown in Figure 11 . Also, the RC product must not allow a frequency of more than 4 MHz taking into account the minimum possible R and C and using the Maximum curve shown. Temperature induced variations in the external components should be considered in calculating the RC product.

Frequency variation from unit to unit due to switching speed and level at constant temperature and V_{CC} = + or - 5 percent.

Frequency variation due to V_{CC} with all other parameters constant with respect to +5V = +7 percent to -4 percent on all devices.

Frequency variation due to temperature with respect to 25 C (all other parameters constant) is as follows:

PART #	VARIATION
387X-00, -05	+6 percent to - 9 percent
387X-10, -15	+9 percent to -12 percent

Variations in frequency due to variations in RC components may be calculated as follows:

Maximum RC = (R max) (C external max + C_{XTL} max)

Minimum RC = (R min) (C external min + C_{XTL} min)

$$\text{Typical RC} = (\text{R typ})\left(\text{C external typ} + \frac{\{C_{XTL}\ \text{max} + C_{XTL}\ \text{min}\}}{2}\right)$$

Positive Freq. Variation = $\frac{\text{RC typical - RC minimum}}{\text{RC typical}}$

Negative Freq. Variation due to RC Components = $\frac{\text{RC maximum - RC typical}}{\text{RC typical}}$

Total frequency variation due to all factors:

387X-00, -05	387X-10, -15
= +18 percent plus positive frequency variation due to RC components	= +21 percent plus positive frequency variation due to RC components
= -18 percent minus negative frequency variation due to RC components	= -21 percent minus negative frequency variation due to RC components

Total frequency variation due to V_{CC} and temperature of a unit tuned to frequency at +5V V_{CC}, 25 C

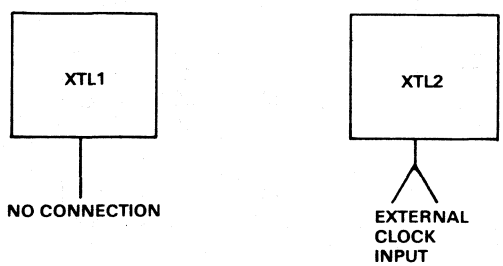
387X-00, -05	387X-10, -15
= + 13 percent	= + 16 percent

EXTERNAL CLOCK CONFIGURATION

The connection for the external clock time base configuration is shown in Figure 12. Refer to the DC Characteristics section for proper input levels and current requirements.

Refer to the Capacitance section of the appropriate 3875 Family device data sheet for input capacitance.

EXTERNAL MODE CONNECTION
Figure 12



MK3875
ELECTRICAL SPECIFICATIONS

OPERATING VOLTAGES AND TEMPERATURES

Dash Number Suffix	Operating Voltage V_{CC}	Operating Temperature T_A
- 00	+5V ± 10%	0°C - 70°C
- 05	+5V ± 5%	0°C - 70°C
- 10	+5V ± 10%	-40°C - +85°C
- 15	+5V ± 5%	-40°C - +85°C

See order information for explanation of part numbers.

ABSOLUTE MAXIMUM RATINGS*

	-00, -05	-10, -15
Temperature Under Bias	-20°C +85°C	-50°C to 100°C
Storage Temperature	-65°C +150°C	-65°C to +150°C
Voltage on any Pin With Respect to Ground (Except open drain pins and TEST)	-1.0V to +7V	-1.0V to +7V
Voltage on TEST with Respect to Ground	-1.0V to +9V	-1.0V to +9V
Voltage on Open Drain Pins with Respect to Ground	-1.0V to +13.5V	-1.0V to 13.5V
Power Dissipation	1.5W	1.5W
Power Dissipation by any one I/O pin	60mW	60mW
Power Dissipation by all I/O pins	600mW	600mW

*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 and conditions for extended periods may affect device reliability.

AC CHARACTERISTICS

T_A, V_{CC} within specified operating range
I/O Power Dissipation < 100mW (Note 4)

SIGNAL	SYM	PARAMETER	-00,-05 MIN	-00,-05 MAX	-10,-15 MIN	-10,-15 MAX	UNIT	NOTES
XTL1 XTL2	t_o	Time Base Period, all clock modes	250	500	250	500	ns	4MHz-2MHz
	$t_{ex(H)}$	External clock pulse width high	90	400	100	390	ns	
	$t_{ex(L)}$	External clock pulse width low	100	400	110	390	ns	
Φ	t_Φ	Internal Φ clock	$2t_0$		$2t_0$			
WRITE	t_W	Internal WRITE Clock period	4 t Φ 6 t Φ		4tΦ 6tΦ			Short Cycle Long Cycle
I/O	$t_{dI/O}$	Output delay from internal WRITE clock	0	1000	0	1200	ns	50pF plus one TTL load
	$t_{sI/O}$	Input setup time to internal WRITE clock	1000		1200		ns	
$\overline{STROBE}$	$t_{I/O-s}$	Output valid to $\overline{STROBE}$ delay	3tΦ -1000	3tΦ +250	3tΦ -1200	3tΦ +300	ns	I/O load = 50fF + 1 TTL load
	t_{sL}	$\overline{STROBE}$ low time	8tΦ -250	12tΦ +250	8tΦ -300	125Φ +300	ns	$\overline{STROBE}$ load = 50pF + 3TTL loads
$\overline{RESET}$	t_{RH}	$\overline{RESET}$ hold time, low	6tΦ +750		6tΦ +1000		ns	
	t_{RPOC}	$\overline{RESET}$ hold time, low for power clear	power supply rise time +5.0		power supply rise time +5.5		ms	
EXT INT	t_{EH}	EXT INT hold time in active and inactive state	6tΦ +750		6tΦ +1000		ns	To trigger interrupt
			2tΦ		2tΦ		ns	To trigger timer

CAPACITANCE

T_A = 25°C All Part Numbers

SYM	PARAMETER	MIN	MAX	UNIT	NOTES
C_{IN}	Input capacitance; I/O $\overline{\text{RESET}}$, EXT INT, TEST		10	pF	unmeasured pins grounded
C_{XTL}	Input capacitance; XTL1, XTL2	23.5	29.5	pF	

DC CHARACTERISTICS

T_A, V_{CC} within specified operating range

I/O Power Dissipation ≤ 100mW (Note 4)

		-00,-05		-10,-15			
SYM	PARAMETER	MIN	MAX	MIN	MAX	UNIT	NOTES
I_{CC}	Average Power Supply Current		94		125	mA	Outputs Open (5)
P_D	Average Power Dissipation		440		575	mW	Outputs Open (6)
V_{IHEX}	External Clock input high level	2.4	5.8	2.4	5.8	V	
V_{ILEX}	External Clock input low level	-.3	.6	-.3	.6	V	
I_{IHEX}	External Clock input high current		100		130	μA	$V_{IHEX}=V_{CC}$
I_{ILEX}	External Clock input low current		-100		-130	μA	$V_{ILEX}=V_{SS}$
$V_{IHI/O}$	Input high level, I/O pins	2.0	5.8	2.0	5.8	V	Standard Pull-Up (1,2)
		2.0	13.2	2.0	13.2	V	Open Drain (1,3)
V_{IHR}	Input high level, $\overline{\text{RESET}}$	2.0	5.8	2.2	5.8	V	Standard Pull-Up(1, 2)
		2.0	13.2	2.2	13.2	V	No Pull-Up (1,3)
V_{IHEI}	Input high level, EXT INT	2.0	5.8	2.2	3.8	V	Standard Pull-Up(1, 2)
		2.0	13.2	2.2	13.2	V	No Pull-Up (1,3)
V_{IL}	I/O ports, $\overline{\text{RESET}}$, EXT INT input low level	-.3	.8	-.3	.7	V	
V_{ILRPT}	$\overline{\text{RESET}}$ input low level to protect RAM during loss at V_{CC}	-.3	.4	-.3	.4	V	
I_{IL}	Input low current, standard pull-up pins		-1.6		-1.9	mA	V_{IN}=0.4V (2)
I_L	Input leakage current, open drain pins		+10		+18	μA	V_{IN}=13.2V
	$\overline{\text{RESET}}$ and EXT INT inputs With no pull-up resistor		-5		-8	μA	V_{IN}=0.0V
I_{OH}	Output high current, standard	-100		-89		μA	V_{OH}=2.4V
	Pull-Up pins	-30		-25		μA	V_{OH}=3.9V
I_{OHDD}	Output high current	-100		-80		μA	V_{OH}=2.4V
	Direct Drive pins	-1.5		-1.3		mA	V_{OH}=1.5V
			-8.5		-11	mA	V_{OH}=0.7V
I_{OL}	Output low current, I/O ports	1.8		1.65		mA	V_{OL}=0.4V
I_{OHS}	$\overline{\text{STROBE}}$ Output High current	-300		-270		μA	V_{OH}=2.4V
I_{OLS}	$\overline{\text{STROBE}}$ output low current	5.0		4.5		mA	V_{OL}=0.4V

DC CHARACTERISTICS FOR STANDBY POWER PINS

V_{CC}, T_A within operating range I/O Power Dissipation ≤ 100 mW (Note 4)

SYMBOL	PARAMETER	-00, -05		-10,-15		UNIT	NOTES
		MIN	MAX	MIN	MAX		
V_{SB}	Standby V_{CC} for RAM	3.2	V_{CC} MAX	3.2	V_{CC} MAX	V	
I_{SB}	Standby Current		6		7.5	mA	$V_{SB} = V_{SB}$ MAX
			3.7		5.0	mA	$V_{SB} = V_{SB}$ MIN
I_{CHARGE}	Trickle charge available on V_{SB} with V_{CC} in operating range.	-.8		-.7		mA	V_{SB} = 3.8V
			-15		-19	mA	V_{SB} = 3.2V

1. $\overline{\text{RESET}}$ and ET INT have internal Schmit triggers giving minimum .2V hysteresis.
2. $\overline{\text{RESET}}$ and EXT INT prgrammed with standard pull-up
3. $\overline{\text{RESET}}$ or EXT INT programmed without standard pull-up
4. Power dissipation for I/O pins is calculated by $\Sigma (V_{CC} - V_{IL}) (|I_{IL}|) + \Sigma (V_{CC} - V_{OH}) (|I_{OH}|) + \Sigma (V_{OL}) (I_{OL})$
5. I_{CC} exclusive of I_{charge}.
6. P_D exclusive of battery charging power. Battery charging power dissipated inside the MK3875 = $(V_{CC} - V_{SB}) (I_{charge})$.

TIMER AC CHARACTERISTICS

Definitions:

Error = Indicated time value - actual time value

tpsc = tΦ x Prescale Value

Interval Timer Mode

Single interval error, free running (Note 3) ..... ±6tΦ
Cumulative interval error, free running (Note 3) 0
Error between two Timer reads (Note 2) ..... ±(tpsc + tΦ)
Start timer to stop Timer error (Notes 1, 4) ..... +tΦ to - (tpsc + tΦ)
Start Timer to read Timer error (Notes 1, 2) ..... -5tΦ to -(tpsc + 7tΦ)
Start Timer to interrupt request error (Notes 1, 3) ..... -2tΦ to -8tΦ
Load Timer to stop Timer error (Note 1) ..... +tΦ to -(tpsc + 2tΦ)
Load Timer to read Timer error (Notes 1, 2) ..... -5tΦ to -(tpsc + 8tΦ)
Load Timer to interrupt request error (Notes 1, 3) ..... -2tΦ to -9tΦ

Pulse Width Measurement Mode

Measurement accuracy (Note 4) ..... +t Φ to -(tpsc +2t Φ
Minimum pulse width of EXT INT pin ..... 2t Φ

Event Counter Mode

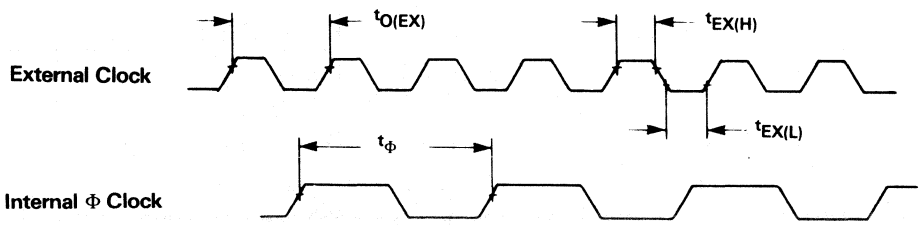
Minimum active time of EXT INT pin ..... 2t Φ
Minimum inactive time of EXT INT pin ..... 2t Φ

Notes:

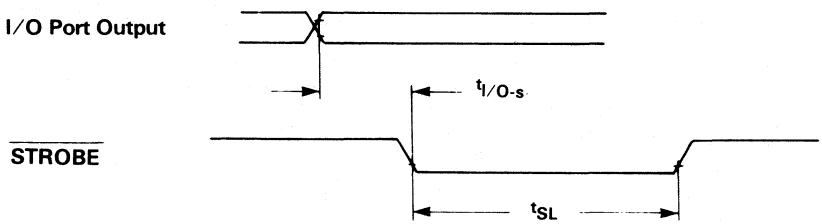
1. All times which entail loading, starting, or stopping the Timer are referenced from the end of the last machine cycle of the OUT or OUTS instruction.
2. All times which entail reading the Timer are referenced from the end of the last machine cycle of the IN or INS instruction.
3. All times which entail the generation of an interrupt request are referenced from the start of the machine cycle in which the appropriate interrupt request latch is set. Additional time may elapse if the interrupt request occurs during a privileged or multicycle instruction.
4. Error may be cumulative if operation is repetitively performed.

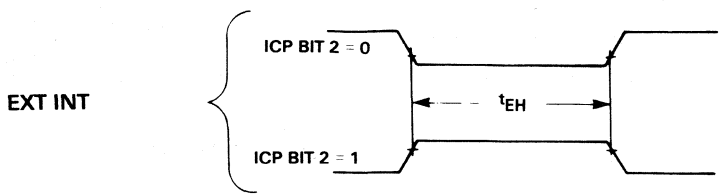
AC TIMING DIAGRAM

Figure 13



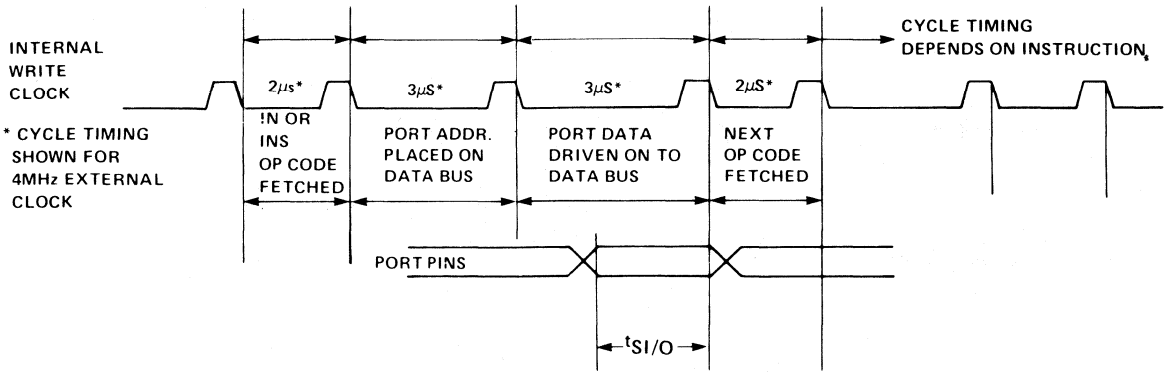
Input capacitance; I/O, $\overline{\text{RESET}}$, EXT INT,



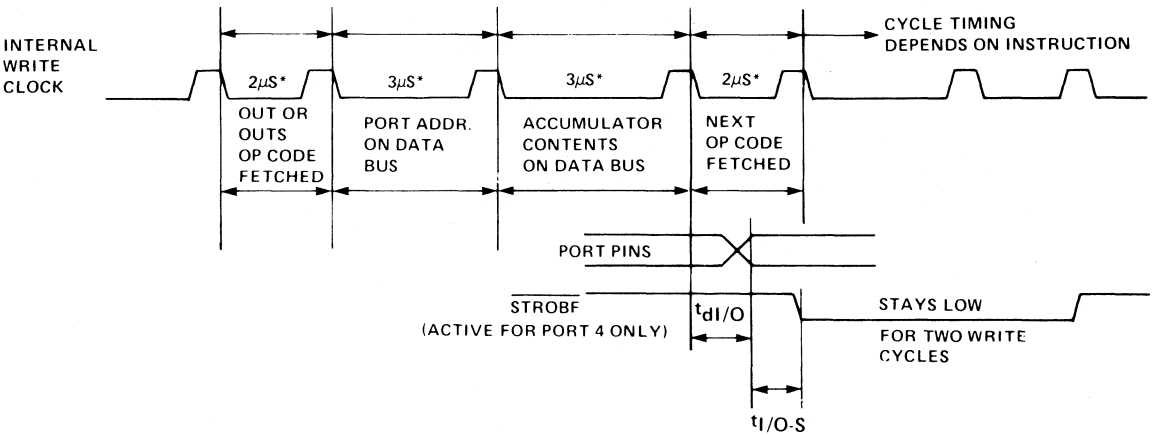


Note: All AC measurements are referenced to V_{IL} max., V_{IH} min., V_{OL} (.8v), or V_{OH} (2.0v).

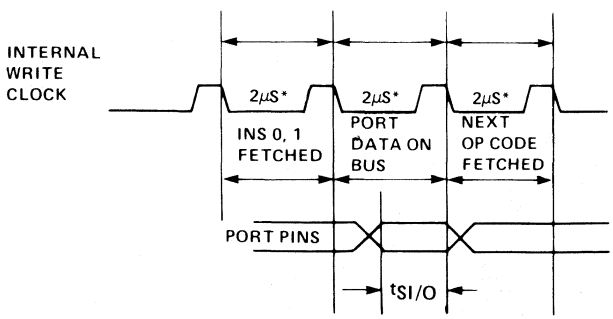
INPUT/OUTPUT AC TIMING
Figure 14



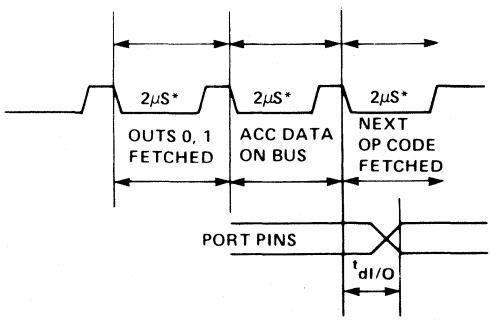
A. INPUT ON PORT 4 OR 5



B. OUTPUT ON PORT 4 OR 5



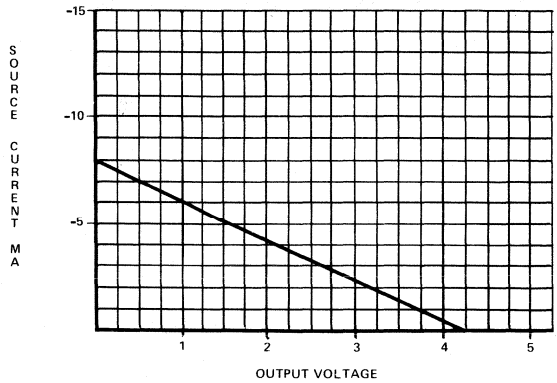
C. INPUT ON PORT 0 OR 1



D. OUTPUT ON PORT 0, 1

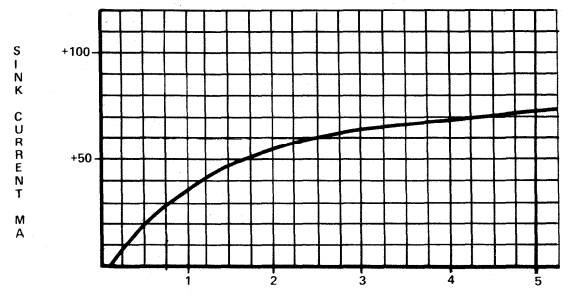
STROBE SOURCE CAPABILITY (TYPICAL AT V_{CC} = 5V, T_A = 25°C)

Figure 15



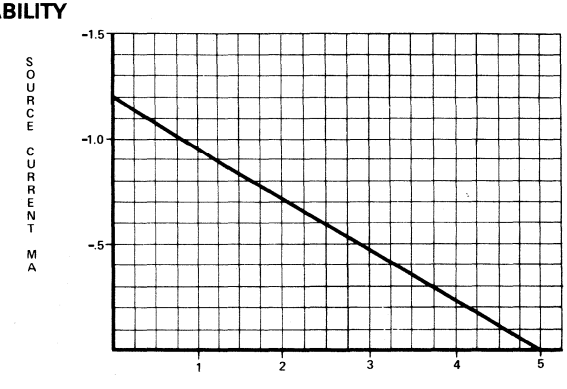
STROBE SINK CAPABILITY (TYPICAL AT V_{CC} = 5V, T_A = 25°C)

Figure 16



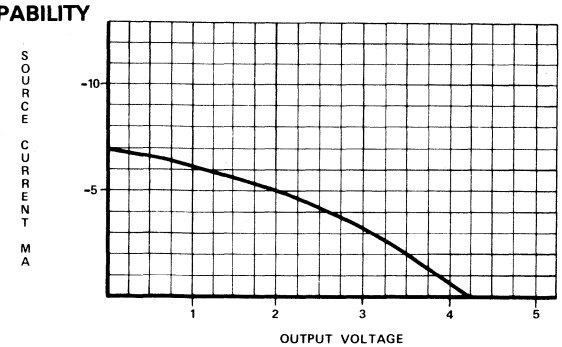
STANDARD I/O PORT SOURCE CAPABILITY (TYPICAL AT V_{CC} = 5V, T_A = 25°C)

Figure 17



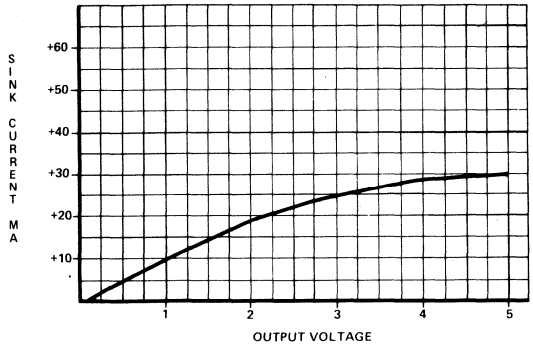
DIRECT DRIVE I/O PORT SOURCE CAPABILITY (TYPICAL AT V_{CC} = 5V, T_A = 25°C)

Figure 18



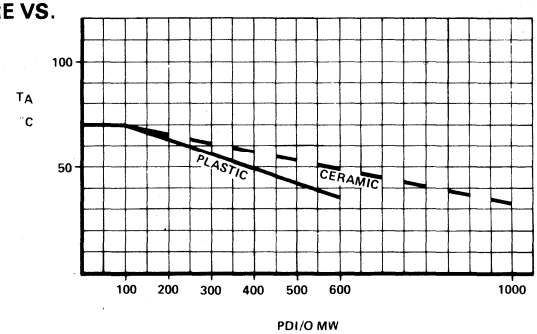
I/O PORT SINK CAPABILITY (TYPICAL AT V_{CC} = 5V, T_A = 25°C)

Figure 19



MAXIMUM OPERATING TEMPERATURE VS. I/O POWER DISSIPATION

Figure 20

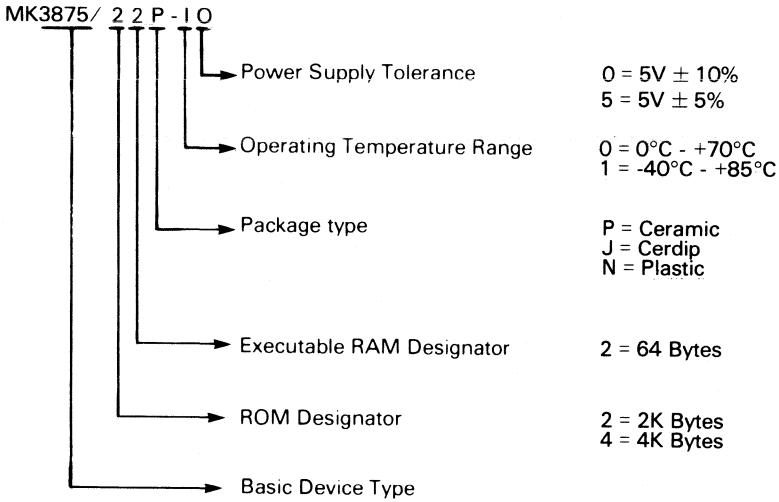


ORDERING INFORMATION

There are two types of part numbers for the 3870 family of devices. The generic part number describes the basic device type, the amount of ROM and executable RAM, the desired package type, temperature range and power supply tolerance. For each customer specific code, additional information defining I/O options and oscillator options will be combined with the information described in the generic part number to define a customer/code specific device order number.

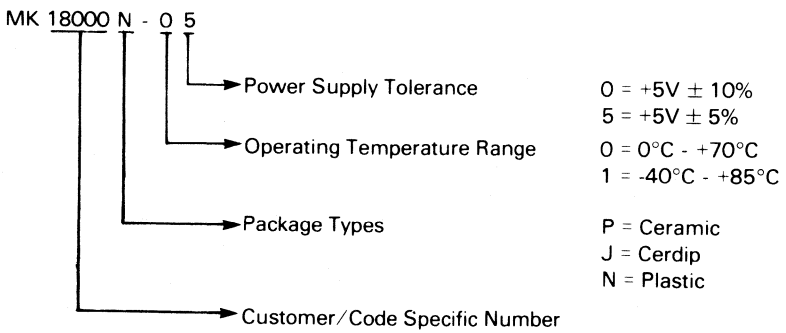
GENERIC PART NUMBER

An example of the generic part number is shown below.



DEVICE ORDER NUMBER

An example of the device order number is shown below.



The customer/code specific number defines the ROM bit pattern, I/O configuration, oscillator type, and generic part type to be used to satisfy the requirement of a particular customer purchase order. For further information on the ordering of mask ROM devices, the customer should refer to the 3870 Family Technical Manual.

ADVANCE INFORMATION

MOSTEK®

3870 SINGLE CHIP MICRO FAMILY

MK38C70 and MK38PC70

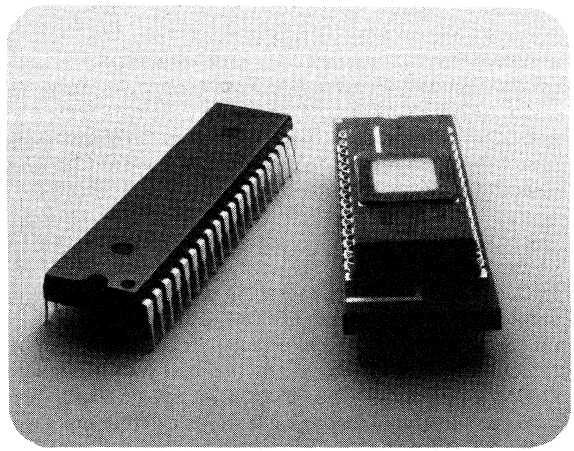
MK38C70 FEATURES

- ☐ Available with 2K bytes of mask programmable ROM memory
- ☐ 64 bytes scratchpad RAM
- ☐ 32 bits (4 ports) TTL compatible I/O
- ☐ Programmable binary timer
  - Interval timer mode
  - Pulse width measurement mode
  - Event counter mode
- ☐ External interrupt input
- ☐ Crystal, LC, RC, or external time base options
- ☐ Low power (50 mW typ.)
- ☐ Two Standby Halt Modes
  - Halt except timers (5mW)
  - Full Halt (500μW)
- ☐ Single +5 volt supply

MK38PC70 FEATURES

- ☐ EPROM version of MK38C70
- ☐ Piggyback PROM package
- ☐ Accepts 24 pin or 28 pin EPROM memories

PHOTO

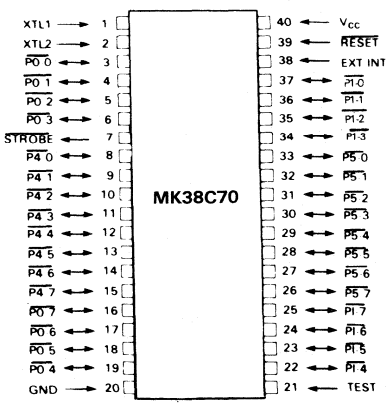


- ☐ Identical pinout as MK38C70
- ☐ In-Socket emulation of MK38C70

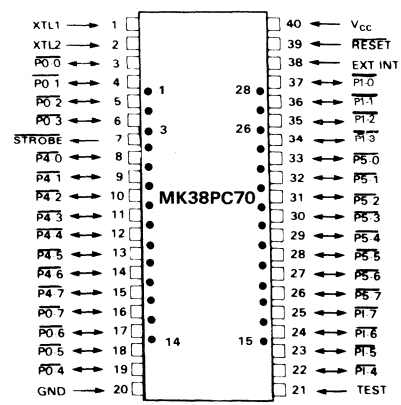
GENERAL DESCRIPTION

The MK38C70 is a complete 8-bit microcomputer on a single CMOS integrated circuit. The MK38C70 can execute more than 70 instructions and is completely software

MK38C70 PIN CONNECTIONS



MK38PC70 PIN CONNECTIONS



compatible with the 3870 family. The MK38C70 features 2K bytes of ROM with 64 bytes of scratchpad RAM, a programmable binary timer, and 32 bits of I/O.

The CMOS process used in manufacturing the MK38C70 gives this part exceptional power characteristics. Typical operating power is 50mW at 5 volts. In addition, two power-down sleep mode instructions give the MK38C70 the ability to reduce power consumption until reactivated by external interrupt. One of these modes allows the binary timer to continue to operate while the processor sleeps.

The programmable binary timer operates by itself in the interval timer mode or in conjunction with the external interrupt input in pulse width measurement and the event counter modes of operation. Two sources of vectored, prioritized interrupt are provided with the binary timer and the external interrupt. The user has the option of specifying one of four clock sources for the MK38C70 and MK38PC70: Crystal, LC, RC, or external clock.

The MK38PC70 microcomputer is the PROM based version of the MK38C70. It is called the piggyback PROM (P-PROM)™ because of its packaging concept. This allows a standard 24-pin or 28-pin EPROM to be mounted directly on top of the microcomputer itself. The EPROM can be removed and reprogrammed as required with a standard PROM programmer. The MK38PC70 retains exactly the same pinout and architectural features as other members of the 3870 family.

INSTRUCTION SET

The MK38C70 and the MK38PC70 are completely instruction set compatible with the other 3870 family members, with the addition of two special sleep mode instructions described below.

SLEEP MODES

Two special sleep modes are provided on the MK38C70 and the MK38PC70 in order to provide a way to reduce power consumption during times of processor inactivity. These two instructions are in addition to the full 3870 instruction set supported by the MK38C70 and the MK38PC70.

HALT EXCEPT TIMER (HET) - This halt instruction causes the MK38C70 to halt execution and enter a low power sleep mode. The clocks and timer continue to operate in normal fashion. The processor resumes execution upon receipt of an interrupt. Execution begins at the address of the interrupt vector if interrupts are enabled, or at the instruction immediately following the HET if interrupts are not enabled. Power consumption during HET sleep mode is approximately 5mW.

HALT (HAL) - This halt instruction causes the MK38C70 to halt execution and enter a low power sleep mode. In this mode, both the processor and the clocks and timer stop. Execution resumes upon receipt of an interrupt, either at the address of the interrupt vector if interrupts are enabled, or at the next instruction immediately following the HAL if interrupts are not enabled. HAL is the lowest power mode, consuming approximately 500μW.

MOSTEK

MICROCOMPUTER COMPONENTS

Display Terminal Controller DTC 14004

FEATURES

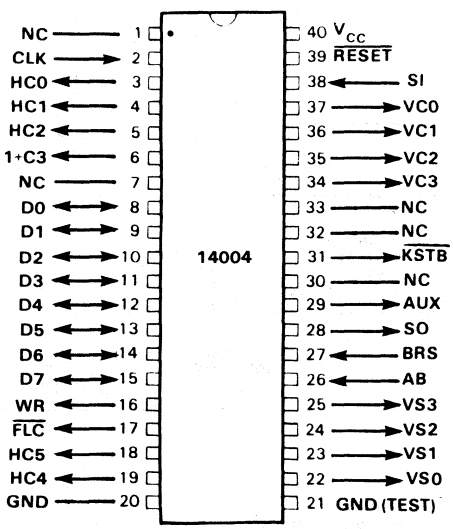
- 16 lines of 64 characters
- 5 x 8 dot matrix character format; upper and lower case ASCII
- Page Mode from top of screen
- Autoscroll after 16th (bottom) line
- Up, down, right, left and home cursor control
- Carriage return
- Screen clear
- Erase to end of line and end of screen
- Direct cursor address - absolute or relative
- Data transfer up to 300 baud
- Serial data line interface
- ASCII/BAUDOT conversion
- Single +5 V; TTL compatible
- MK3870 based

The DTC 14004 is a preprogrammed MK3870, an n channel MOS LSI chip in a 40 pin package. The DTC 14004 is a highly cost effective display terminal controller requiring minimal chip count (see Figure 1) for a fully integrated display terminal capable of up to 300 baud data transfer rate. The chip performs serial transmission and reception, keyboard scanning and cursor control functions.

GENERAL DESCRIPTION

Figure 1 shows the fundamental building blocks of a typical terminal. DTC 14004 is outlined i the dotted box. The keyboard scanner monitors the keyboard and when a key is pressed, passes the corresponding code to the serial transmitter. The transmitter converts the parallel data from the keyboard into the appropriate serial format for the data link. The serial receiver converts incoming serially formatted data into parallel data for processing. Cursor control function decodes incoming characters into two groups: printable and special. Printable characters are written into the screen RAM and special characters are further decoded and their functions executed.

PIN FUNCTIONS

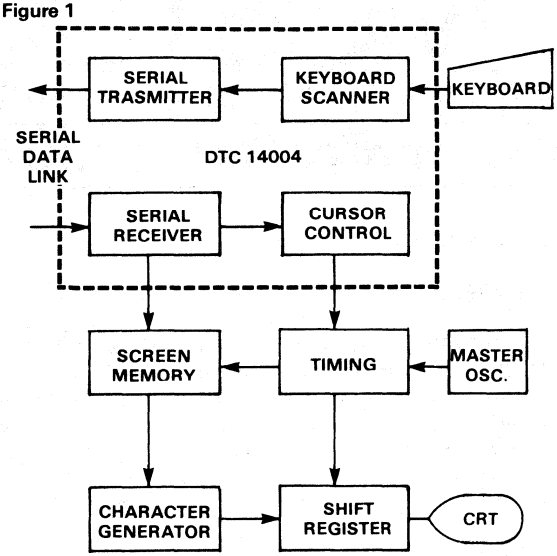


PIN DESCRIPTION

NAME	DESCRIPTION
D0-D7	DATA
HC0-HC5	HORIZONTAL CURSOR
VC0-VC3	VERTICAL CURSOR
VS0-VS3	SCROLL
AB	ASCII/BAUDOT
BRS	BAUD RATE SELECT
AUX	AUXILIARY OUTPUT
SO/SI	SERIAL OUT/SERIAL IN
KSTB	KEYBOARD STROBE
$\overline{FLC}$	LINE CLEAR
WR	WRITE

BLOCK DIAGRAM

Figure 1



DTC 14004 replaces the SSI and MSI CRT terminal functions outlined in the dotted box

Master oscillator and timing block generate ROM and RAM addresses, composite sync, cursor compare and other control signals. The character generator converts each displayed character into a dot matrix, and the shift register converts the matrices into a serial video, see Figure 2.

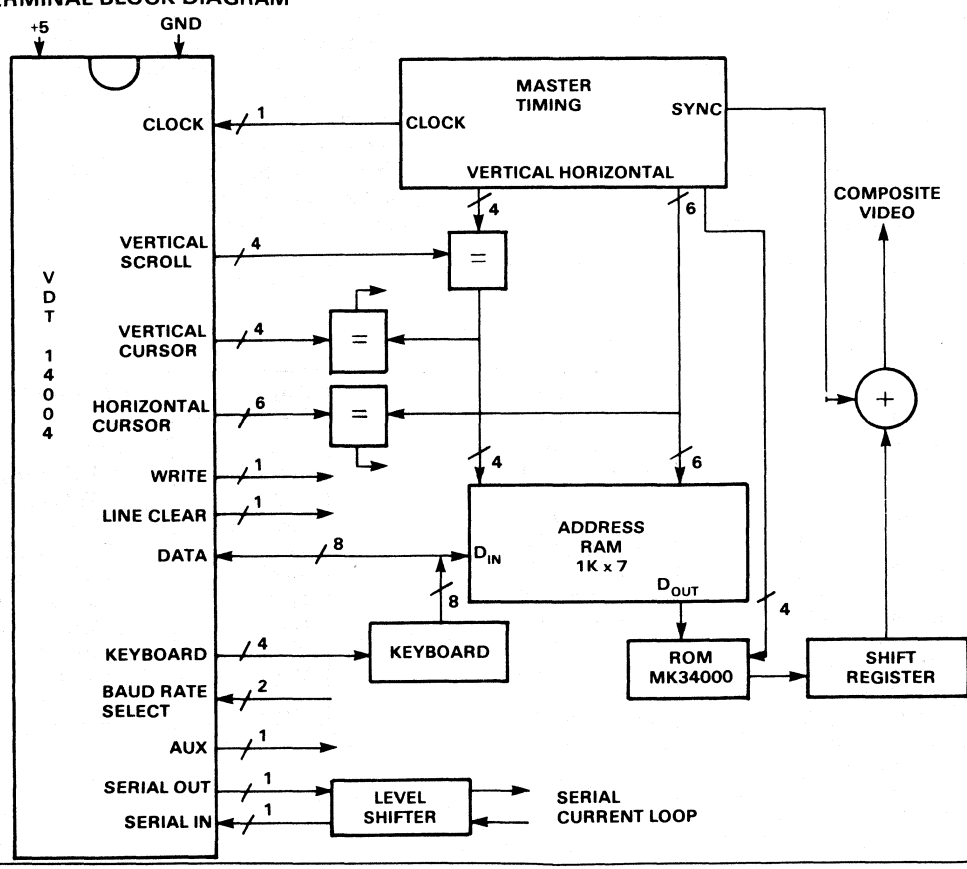
Addresses from the timing chain add to the vertical scroll address to generate the physical RAM vertical address. This allows the displaying of any physical line of the RAM as the top line on the screen.

HARDWARE INTERFACE DESCRIPTION

DTC 14004 is TTL compatible. The UART interface (S0, S1) is application dependent and may be driven with differential or single ended, isolated or non-isolated current loop or RS-232-C line drivers/receivers. For reliable reception, a 'slice factor' of 16, typically used in hardware UARTS (also called 16 x clock) can be used in full duplex application however the data rate will be slowed. For half duplex or one-way-only reception a slice factor of 1 may be adequate.

DETAILED TERMINAL BLOCK DIAGRAM

Figure 2



CLOCK RATES

Factors involved in clock rate consderations are:

1. DTC clock rate (f)
2. Data link Baud rate (B)
3. Slice factor (s)
4. Timer divisor factor (d)

1. DTC Clock Rate is often determined by factors not relating to the software UART. For example, a 3.58 MHz color TV crystal may be selected for availability and low cost. Or an existing clock source may be used. Generally, flexibility in the timer will allow great latitude in DTC clock rate selection.

2. Baud Rate is often determined by the application. Factors such as hardware limitations may influence maximum practical Baud rate.

3. Slice Factor refers to the number of samples per bit time. Hardware UARTs typically use a Slice factor of 16 (also referred to as 16 X clock). A large slice factor improves receiver reliability at the cost of more CPU time spent in the UART routine. A slice factor of 1 is entirely adequate for transmitting, and for one-way-only or half-duplex reception. Full duplex reception will generally require a slice factor of 8 or greater.

4. Timer Divisor Factor is calculated from the preceding factors according to the formula

$$d = \frac{f}{sB} \qquad (1)$$

where f is the effective DTC cycle rate (or input to timer prescaler). The value computed for d will seldom be an integer value directly attainable from a timer (unless the DTC clock rate is selected for the UART).

For example:

A given application requires a DTC cycle rate of 1.52064 MHz. A slice factor of 8 is selected.

Table 2 shows computed values of d for four baud rates, along with nearest divisor attainable with the DTC and the resultant baud rate error. Since a slice factor of 8 provides for several percent of error margin, the results of Table 2 are entirely adequate.

DIVISOR COMPUTATIONS
Table 1

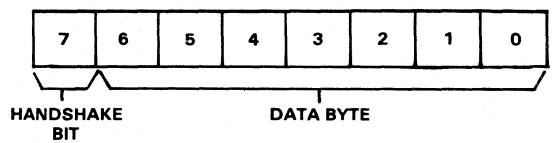
f	s	B	d	nearest integer	error
1.52064 M	8	300	633.6	635	-0.22%
		110	1728.0	1730	0.12%
		74.2	2561.7	2560	+0.07%
		45.45	4181.8	4180	+0.04%

FUNCTIONAL DESCRIPTION

UART

The Serial Transmitter and receiver functional blocks shown in Figure 1 compose the software UART. The UART implements asynchronous serial data format, which is widely used in low to medium speed data communication. Once the software UART has been started, the overhead software to pass data to and from it is actually less than if a hardware UART had been used.

RXD/TXD FORMAT
Figure 3



STARTING THE UART

Initialization of the UART involves setting up the port bits, timer, and UART registers.

Port bits: The serial out pin is initialized to a "MARK" state. The other pins are cleared for use as inputs.

Timer presets are determined by the baud select pins. In the example, one of the four baud rates listed in Table 1 is selected by the 2 bit code, pins 26 and 27. From Table 2 the prescaler and divisor values are selected and output to the timer control and timer count ports, respectively.

All UART registers are set to FF_{16}. RXD and TXD are thus initialized to a "no data available" state, and the slice counters are set to the idle state.

TIMER SETUP
Table 2

d	prescale	divisor	Port 6	Port 7
635	5	127	4A	7F
1730	10	173	6A	AD
2560	40	64	AA	40
4180	20	209	8A	D1

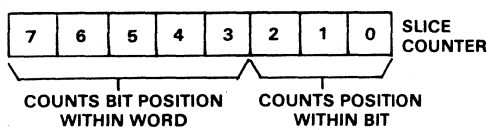
THE RECEIVER

When no data is being received, the receiver is idle. The receive counter contains FF_{16}. Figure 4 shows the format of the slice counters. At each slice time, the receiver detects the idle FF_{16} and tests for a start bit. If a start bit is not detected, the slice count is reset to FF_{16} and the receiver is exited. When a start bit is received, the slice count is set to $B4_{16}$ (see Figure 5). At each subsequent slice time, the count is incremented. Four slices later, when the count becomes $B8_{16}$, the receiver again checks for a start bit. If the start bit is no longer there, the receiver assumes a false start bit and resets the counter to idle. If the start bit is still good, the receive process is allowed to continue.

At subsequent slice times the counter is incremented. Each time the low 3 bits are all 0, the serial input line is sampled and the new bit merged into the input byte register (IB). The last bit is sampled when the counter equals $F8_{16}$. Any code conversion required is done at this time, and then the received (converted) data byte is placed into RXD with MSB=0. The counter is incremented toward FF_{16} at subsequent slice times, and after FF_{16} is reached the receiver becomes idle, ready for a new start bit. Note that only slightly more than one half stop bit is required for proper reception. This helps guarantee that the receiver can keep up with the transmitter even if the transmitter is operated a slightly faster than nominal baud rate, or in the presence of data link distortion.

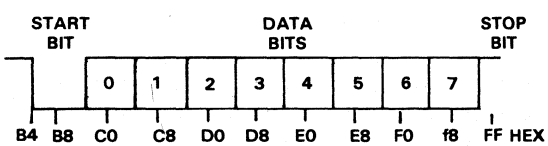
SLICE COUNTER FORMAT

Figure 4



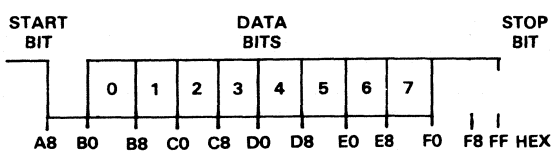
RECEIVE

Figure 5



TRANSMIT

Figure 6



THE TRANSMITTER

When no data is being transmitted, the transmitter is idle, and the slice counter contains FF_{16}. At each slice time, the transmitter checks TXD for data to send. If none is available, the transmitter is exited. If data is available in TXD, any code conversion required is completed, the data is copied into the output byte register and the MSB of TXD is set to 1. A start bit is output to the serial output port bit, and the slice counter is set to $A8_{16}$ (see Figure 6). At subsequent slice times the counter is incremented. Whenever the low three bits equal zero (see also Figure 4) the LSB of the output byte is output to the serial output. The byte is shifted toward the LSB, and 1 is placed into the upper bit.

After all eight data bits have been shifted out at bit times $B0_{16}$ through $E8_{16}$, two stop bits are output at $F0_{16}$ and $F8_{16}$. Those stop bits are simply the 1's shifted into the MSB of the output bit. At the end of the last stop the transmitter becomes idle, ready for a new character. Note that two full stop bits are transmitted. If the receiving end requires less than two, the extra time is interpreted as idle time.

INSTRUCTION SET

The DTC 14004 performs the following tasks executed by instructions stored in the on board ROM.

POWER UP ROUTINE

On power up the program automatically clears the screen and places the cursor in the upper left corner. It then transfers control to the basic READ Loop which reads one character from the receiver and decodes it.

Steps

- Set receiver and transmitter idle and no data ready
- Set up timer ports for baud rate selected
- Screen clear automatic on power up
- Set cursor to bottom line
- Set scroll to zero
- Home cursor to upper left corner
- Set cursor to top line
- Carriage return
- Set cursor to left margin
- Read - outputs new cursor position and inputs one character from serial receiver and decode it.

Decode:
- Printable
- From feed screen clear
- Home
- Carriage return
- Line feed
- Vertical tab
- Horizontal tab
- Backspace
- DC1 (set aux)
- DC3 (clear aux)
- Erase to end of screen

Erase to end of line
Start of cursor sequence

SCAN ROUTINE

The scan routine alternately tests for receive data ready and transmitter ready.

Steps

- Test receiver for new character
- Test transmitter for readiness
- Test keyboard for key depressed
- Test for first time down
- Start transmission then loop

WRITE SCREEN

All printable characters are written and line clear performed on the screen by doing the following.

Steps

- Enable write on horizontal and vertical cursor match
- Wait for one full screen scan period (1/60 sec)
- Place code for blank on RAM data lines
- Enable write on vertical cursor match
- Turn off write and line clear

LINE FEED

Steps

- Increment vertical cursor (MOD 16)
- If result zero cursor was on bottom line, so reset cursor to bottom line
- Increment scroll
- Clear new bottom line on screen

VERTICAL TAB AND HORIZONTAL TAB

Steps

- Decrement vertical cursor (MOD 16)
- If result = 15, cursor was on top line, so reset cursor to top line
- Decrement scroll
- Clear new top line on screen

- If horizontal cursor = 63 (right margin) do nothing otherwise increment horizontal cursor

BACKSPACE

Steps

- If horizontal cursor = 0 (left margin) do nothing otherwise decrement horizontal cursor

SPECIAL CHARACTERS

Special characters, erase to end of screen, starts at the bottom of the screen and clears up to but not including the current line. Erase to end of line starts at the right margin and clears up to but not including the current character position.

Steps

- Erase to end of screen
- Put code for "blank" on data in lines to RAM
- Save old cursor value
- Set cursr to bottom line
- If old cursor not euqal to new cursor, clear current line
- Decrement new cursor and loop

- Erase to end of line
- Place code for "blank" on data in lines to RAM
- Save old cursor value
- Set cursor to right margin
- If old cursor not equal to new cursor, write current character position
- Decrement new cursor and loop

DC1, DC3

Since the DC1 and DC3 set and clear AUX pin on DTC 14004, this pin can be used to control a tape drive, change character sets and indicate status.

CURSOR CONTROL

Cursor control sequences provide for direct cursor movement. For example, ESC = A, B moves the cursor to the second line (ASCII value of A MOD 16 = 1, top line = 0), third column (B MOD 64 = 2, left column = 0). Similarly, ESC +C, D moves cursor down three and to the right four places.

SCU 1

FEATURES

- ☐ Provides programmable remote I/O functions as well as network communications in a single 40 pin package
- ☐ Performs a specific preprogrammed function on command including:
  - Bit input or bit output
  - Byte input or byte output
  - Set, clear or toggle selected pins
  - Interface to A/D converter D/A converter or 3½ digit DPM
  - Monitor input pins for a specific bit pattern
- ☐ Over 20 preprogrammed functions
- ☐ Allows a user to communicate with multiple SCU's over a single communications channel
- ☐ Asynchronous serial data transmission
- ☐ 300 or 1200 Baud Data Transmission Rate with 3.6864 MHz Crystal Time Base
- ☐ Secure, error resistant data link protocol
  - Parity generate and check
  - LRC generate and check
  - Efficient message format
- ☐ Modem Control Signals Provided
- ☐ Requires single +5 volt supply
- ☐ Low power (275 mW typ)

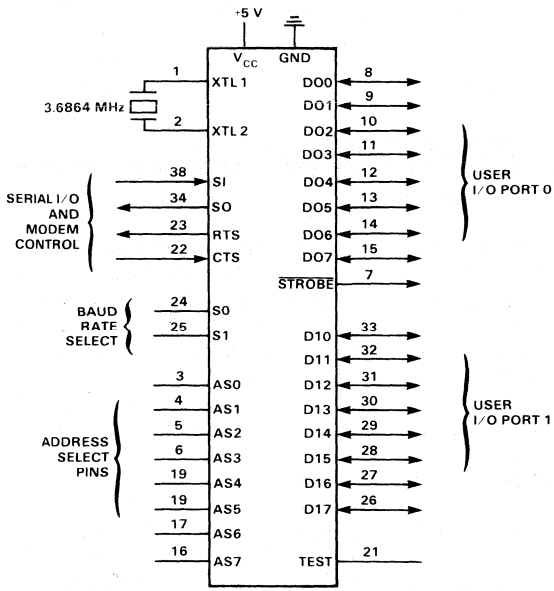
FUNCTIONAL PIN DESCRIPTION

XTL1,2: Time base inputs for a 3.6864MHz crystal.

AS0 - AS7: SCU 1 Address (input). These 8 pins determine the address of the SCU 1, using positive logic. Internal pullup resistors allow unconnected inputs to be a logic 1. Grounded inputs are a logic 0. Address 'FF' hex is not allowed.

DO0 - DO7: SCU 1 port 0. These 8 pins may be TTL compatible inputs or latched outputs. All have internal pullup resistors.

PIN CONNECTIONS



D10 - D17: SCU 1 port 1. These 8 pins may be TTL compatible inputs or latched outputs. All have internal pullup resistors.

STROBE: (Output, active low). This pin provided a single low pulse after valid data has been output to port 0. It is capable of driving 3 TTL loads.

SI: Serial Input. This Schmit trigger input receives serial, asynchronous data from the host computer.

SO: Serial Output. The SCU 1 command response is serially output through this pin, least significant bit first. Between transmissions, SO remains at a logic 1 (marking or idle line).

RTS: Request To Send (output, active high). Prior to responding to a command, RTS becomes active indicating the serial line should direct information from the SCU 1 to the host.

CTS: Clear to Send (input, active high). An active input indicates the SCU 1 may begin its response; thus it is an indication that the serial link is now ready to transmit information from the SCU 1 to the host.

RESET: External Reset (input, active low). This input may be used to guarantee the SCU 1 is held in a reset state until V_{CC} reaches the minimum operating voltage or to reset the device after a disturbance on the V_{CC} line.

SO, SI: Baud Rate Select (inputs). These inputs are strapped to specify the serial Baud rate for transmit and receive. Baud rate selection is made according to the following chart.

SI	SO	Baud Rate
0	0	300
0	1	1200
1	0	1200
1	1	1200

EXT STATUS: External Status (input, active high). Used for certain commands, EXT STATUS signals when an operation is done.

TEST: (input, active high). Test is used by Mostek to test the SCU 1. For normal circuit functionality, this pin is left unconnected or may be grounded.

NC: No Connect. These lines are undefined and should not be connected to anything.

V_{CC}: Power supply, +5V.

GND: Power supply ground.

GENERAL DESCRIPTION

The Serial Control Unit (SCU 1) is a pre-programmed MK3870 single chip microcomputer. It acts as a complete, remote input/output controller, recognizing over 20 commands received from a host processor via a half-duplex asynchronous serial link. After performing the specified command, the SCU generally echoes the message back to the host for acknowledgement and verification. The SCU 1 may be used for both monitoring and control systems where remote intelligence is required as its 16 I/O lines may be configured to provide many different monitoring and input/output functions. The device contains a complete communications processor capable of both generating and receiving asynchronous serial messages.

A flexible and expandable protocol has been designed to allow up to 255 SCU 1's to be on a single communications channel. Included in the message protocol are parity and check sum redundancy to provide a highly error resistant message interchange. The SCU 1 has modem signals allowing easy interfacing with various serial line drivers and receivers. All input and output pins are TTL compatible.

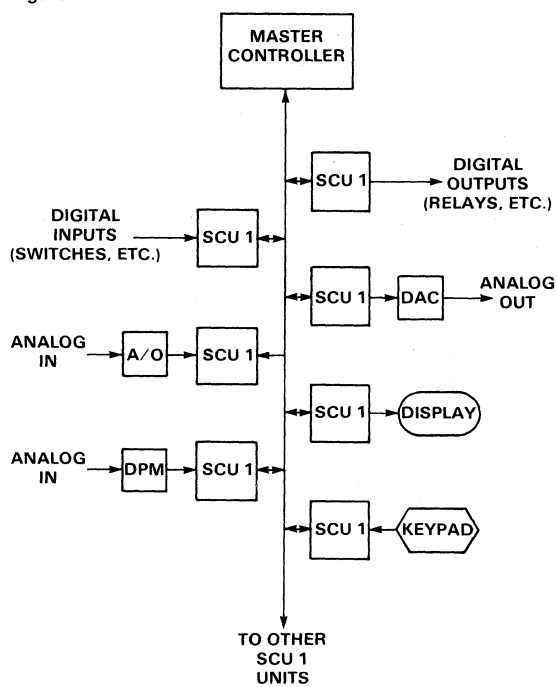
The SCU 1 is designed to communicate with a master controller (host computer) via a half-duplex (or simplex) multidrop serial link. The SCU 1 exists as a slave unit to the master controller, and each SCU 1 on a serial link must have a unique address, which identifies which SCU 1 is to respond to a particular command or inquiry from the master.

To prevent line contention, only the host may initiate communication. The particular SCU 1 addressed by the host will respond only after receiving a valid command. After the particular SCU 1 receives a valid message (address + command), it will perform the appropriate task and in most cases, transmit a response to the host. While transmitting, the SCU 1 will ignore any serial input.

A system configured as described above is generally referred to as a serial polled communications network (or a polled network). As indicated, a particular SCU 1 must be polled by the host via a valid message before it may respond. A typical system configuration is shown in Figure 1.

TYPICAL SYSTEM CONFIGURATION

Figure 1



The SCU 1 operates with a data communications protocol defined for ease of use, high throughput, and data integrity. The protocol, though bit serial in nature, is character oriented, with 5 characters comprising a typical message. Communications is comprised of a command or inquiry message transmitted from the host to a particular SCU 1, followed, in general, by a response message from the SCU 1 to the host.

The network communications protocol is character oriented, with the character format as shown in Figure 2. The format is consistent with industry standard.

Network communications is comprised of command or

inquiry messages transmitted from the host to a particular SCU 1 to the host. Figure 2 shows a typical message format. In order of transmission, the characters are (1) Address, (2) Command, (3) Data 1, (4) Data 2, and (5) Logitudinal Redundancy Check Character.

Any SCU 1 must receive a complete message from the host before it may respond. The SCU 1 will then perform the appropriate task and, in most cases, transmit a response to the host. There are 2 allowable message formats for polling (transmitting a command or inquiry from the host to the SCU 1) and 3 allowable message formats for SCU 1 response to the host.

*For additional information on the SCU 1, refer to the SCU 1 Operations Manual.

POWER ON RESET

The SCU 1 contains Power-On-Reset circuitry to automatically reset the device following a typical power-up situation, thus saving external reset circuitry in many applications. The internal Power-on-reset circuitry is designed to keep the SCU 1 in a reset condition until the crystal oscillator and internal clock circuitry is operational.

EXTERNAL RESET

In some applications it may be desirable to provide external reset circuitry to accommodate particular power-on conditions or to provide operator reset capability, such as with a RESET push button. Figure 3 shows a possible external reset circuit that may be used to control the power-on reset and/or provide operator reset capability. Figure 4 shows the desired operation of the SCU 1 RESET input.

SCU 1 CLOCKS

The suggested time base for the SCU 1 is a 3.6864MHz crystal. This frequency was chosen to provide proper serial timing. Any frequency from 2 MHz to 4 MHz may be used, however the serial Baud rates will be proportionally affected. The following crystal parameters are recommended for the SCU 1:

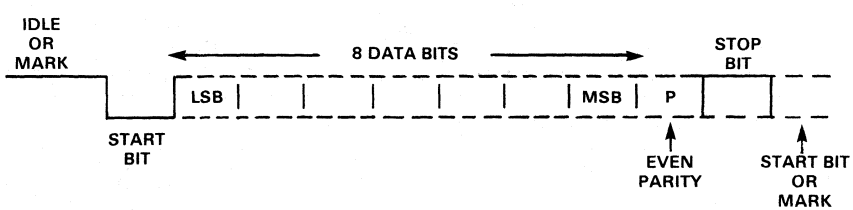
a) Parallel resonance, Fundamental Mode At-Cut
b) Shunt capacitance (CO) = 7pF maximum
c) Series resistance (Rs) = See table
d) Holder = See table

FREQUENCY	Rs	HOLDER
2-2.7 MHz	300 Ohms max	HC-6 HC-33
2.8-4 MHz	150 Ohms max	HC-6 HC-18* HC-25* HC-33

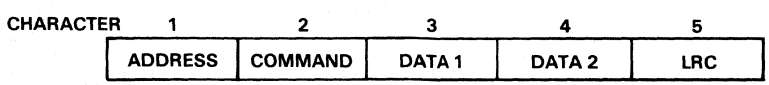
*This holder may not be available at frequencies near the lower end of this range.

Figure 2

SCU 1 CHARACTER FORMAT

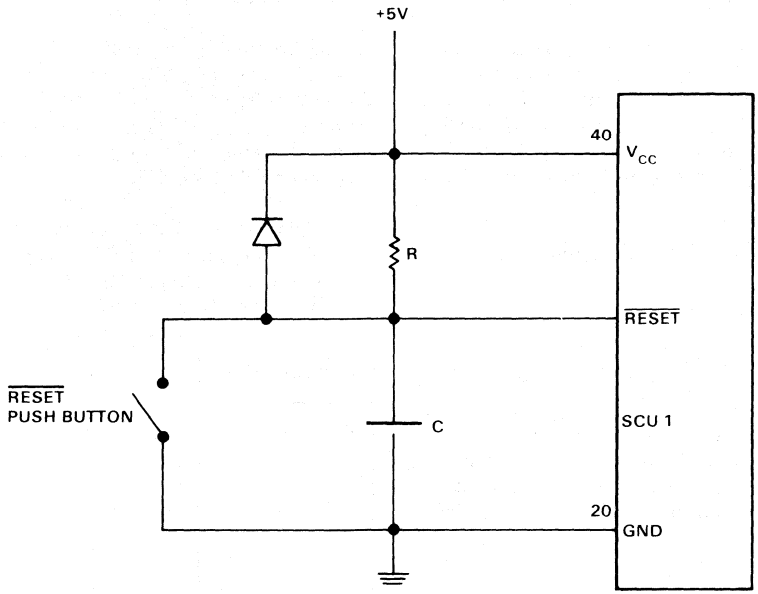


TYPICAL MESSAGE FORMAT



EXTERNAL RESET CIRCUITRY

Figure 3



I/O GROUP COMMAND/RESPONSE SUMMARY

Table 1

	COMMAND DESCRIPTION			CORRECT RESPONSE				ERROR RESPONSE	
HEX COMMAND	DESCRIPTION	DATA 1	DATA 2	HEX COMMAND	DATA 1	DATA 2	HEX COMMAND	DATA 1	DATA 2
00-79	USER DEFINED								
80	Byte Output to Port	Port Select	Data Out	80	Port Selected	Verified Data	FB	Port Selected	Data Out
81	Byte Input From Port	Port Select	And Mask	81	Port Selected	Input Data	FB	Port Selected	And Mask
82	Set Bit In Port	Port Select	Or Mask	82	Port Selected	Verified Output	FB	Port Selected	Or Mask
83	Clear Bit In Port	Port Select	And Mask	83	Port Selected	Verified Output	FB	Port Selected	And Mask
84	Toggle Bit(s) in Port	Port Select	XOR Mask	84	Port Selected	Verified Output	FB	Port Selected	XOR Mask
85	Output Two Bytes	Port 1 Data	Port Ø Data	85	Port 1 Data	Port Ø Data	FB	Port Ø Data	Port 2 Data
86	Input Two Bytes	Port 1 And Mask	Port Ø And Mask	86	Port 1 Input	Port Ø Input	FB	Port 1 And Mask	Port Ø And Mask

MONITOR AND A/D GROUP COMMAND/RESPONSE SUMMARY

Table 2

COMMAND DESCRIPTION				CORRECT RESPONSE				ERROR RESPONSE	
HEX COMMAND	DESCRIPTION	DATA 1	DATA 2	HEX COMMAND	DATA 1	DATA 2		HEX COMMAND	DATA 1	DATA 2
87	12 Bit A/D Conversion	MPX Channel	FF	87	MPX & Data	Data	FB	MPX	FF
88	Monitor Port Ø (=)	And Mask	Pattern	Reponds on Poll:			N/A	N/A	N/A
89	Monitor Port 1 (=)	And Mask	Pattern	Responds on Poll:			N/A	N/A	N/A
8A	Monitor Port Ø (≠)	And Mask	Pattern	Responds on Poll:			N/A	N/A	N/A
8B	Monitor Port 1 (≠)	And Mask	Pattern	Responds on Poll			N/A	N/A	N/A
8C ~ F8	Reserved for Future Mostek Commands								

N/A = Not Applicable

SUPERVISORY GROUP COMMAND/RESPONSE SUMMARY

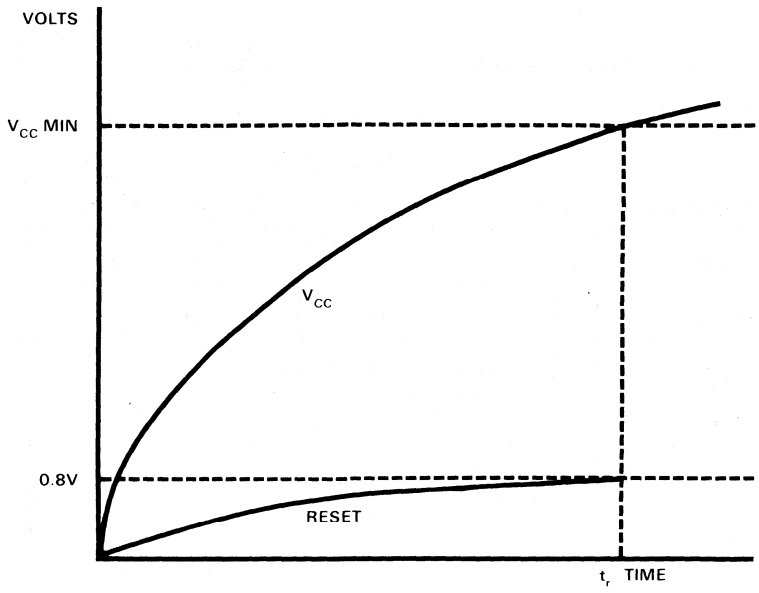
Table 3

				ACTIVE RESPONSE				INACTIVE RESPONSE	
HEX COMMAND	DESCRIPTION	DATA 1	DATA 2	HEX COMMAND	DATA 1	DATA 2		HEX COMMAND	DATA 1	DATA 2
F9	Short Poll Disable	XX	XX	F9	XX	XX	N/A	N/A	N/A
FA	Short Poll Enable	XX	XX	FA	XX	XX	N/A	N/A	N/A
FB	Loop (Echo)	XX	XX	FB	XX	XX	N/A	N/A	N/A
FC	Reset Report/Task	XX	XX	FC	XX	XX	N/A	N/A	N/A
FD	Long Poll	XX	XX	FD	Mask	Pattern	FD	XX	XX
FE	Short Poll	None	None	FE	Mask	Pattern	FE	None	None
FF	Initialization	FF	FF	← No Response →					

N/A = Not Applicable
X = Don't Care

RESET CIRCUIT OPERATION

Figure 4



t_r + 50 msec typical

OPERATING VOLTAGES AND TEMPERATURES

Dash Number Suffix	Operating Voltage V_{CC}	Operating Temperature T_A
— 00	+5V ± 10%	0°C - 70°C
— 05	+5V ± 5%	0°C - 70°C
— 10	+5V ± 10%	-40°C - +85°C
— 15	+5V ± 5%	-40°C - +85°C

ABSOLUTE MAXIMUM RATINGS*

	-00, -05	-10, -15
Temperature Under Bias	-20°C to +85°C	-50°C to +100°C
Storage Temperature	-65°C to +150°C	-65°C to +150°C
Voltage on any Pin With Respect to Ground (Except open drain pins and TEST)	-1.0V to +7V	-1.0V to +7V
Voltage on TEST with Respect to Ground	-1.0V to +9V	-1.0V to +9V
Voltage on Open Drain Pins With Respect to Ground	-1.0V to +13.5V	-1.0V to +13.5V
Power Dissipation 1.5W	1.5W	
Power Dissipation by any one I/O pin	60mW	60mW
Power Dissipation by all I/O pins	600mW	600mW

*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.

AC CHARACTERISTICS

T_A, V_{CC} within specified operating range. I/O power dissipation ≤ 100mW (Note 2)

SIGNAL	SYM	PARAMETER	-00, -05 MIN	-00, -05 MAX	-10, -15 MIN	-10, -15 MAX	UNIT	NOTES
XTL1 XTL2	t_o	Time Base Period, all clock modes	250	500	250	500	ns	4MHz-2MHz f = 3.6864MHz
	$t_{ex(H)}$ $t_{ex(L)}$	External clock pulse width high External clock pulse width low	90 100	400 400	100 110	390 390	ns ns	for standard Baud rates
$\overline{STROBE}$	$t_{I/O-s}$	Output valid to $\overline{STROBE}$ delay	3tΦ -1000	tΦ3tΦ +250	3tΦ -1200	3tΦ +300	ns	I/O load = 50pF + 1 TTL load
	t_{sL}	$\overline{STROBE}$ low time	8tΦ -250	12tΦ +250	8tΦ -300	12tΦ +300	ns	$\overline{STROBE}$ load= 50pF + 3TTL loads
$\overline{RESET}$	t_{RH}	$\overline{RESET}$ hold time, low	6tΦ +750		6tΦ +1000		ns	
	t_{RPOC}	$\overline{RESET}$ hold time, low for power clear	power supply rise time +0.1		power supply rise time +.15		ms	

DC CHARACTERISTICS

T_A, V_{CC} within specified operating range I/O power dissipation ≤ 100mW (Note 2)

SYMBOL	PARAMETER	-00, -05 MIN	-00, -05 MAX	-10, -15 MIN	-10, -15 MAX	UNIT	DEVICE
I_{CC}	Average Power Supply Current		85		110	mA	Outputs Open
P_D	Power Dissipation		400		525	mW	Outputs Open
V_{IHEX}	External Clock input high level	2.4	5.8	2.4	5.8	V	
V_{ILEX}	External Clock input low level	-.3	.6	-.3	.6	V	
I_{IHEX}	External Clock input high current		100		130	μA	$V_{IHEX}=V_{CC}$
I_{ILEX}	External Clock input low current		-100		-130	μA	$V_{ILEX}=V_{SS}$
$V_{IHI/O}$	Input high level, I/O pins	2.0	5.8	2.0	5.8	V	
V_{IHR}	Input high level, $\overline{RESET}$	2.0	5.8	2.2	5.8	V	
V_{IHEI}	Input high level SI	2.0	5.8	2.2	5.8	V	
V_{IL}	Input low level	-.3	.8	-.3	.7	V	(1)
I_{IL}	Input low current, all pin with standard pull-up resistor		-1.6		-1.9	mA	V_{IN}=0.4V
I_{OH}	Output high current	-100 -30		-89 -25		μA μA	V_{OH}=2.4V V_{OH}=3.9V
I_{OHS}	$\overline{STROBE}$ Output High current	-300		-270		μA	V_{OL}=2.4V
I_{OL}	Output low current	1.8		1.65		mA	V_{OL}=0.4V
I_{OLS}	$\overline{STROBE}$ Output Low current	5.0		4.5		mA	V_{OL}=0.4V

1. $\overline{RESET}$ and SI have internal Schmit triggers giving minimum .2V hysteresis.
2. Power dissipation for I/O pins is calculated by $\Sigma (V_{CC} - V_{IL})(I_{IL}) = \Sigma (V_{CC} - V_{OH})(I_{OH}) = \Sigma (V_{OL})(I_{OL})$

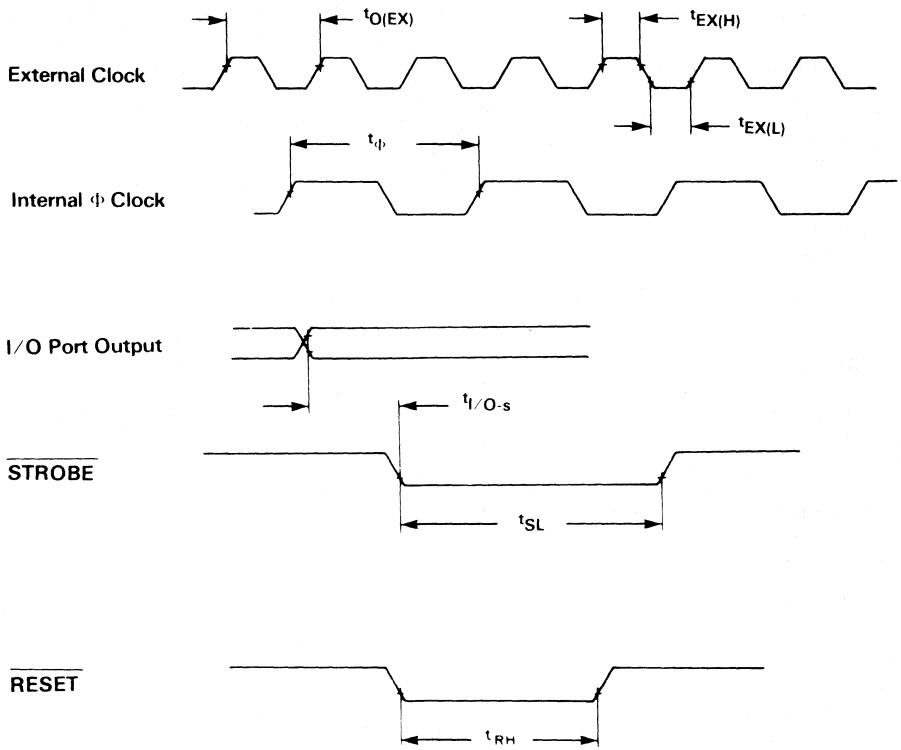
CAPACITANCE

T_A = 25°C
All Part Numbers

SYM	PARAMETER	MIN	MAX	UNIT	NOTES
C_{IN}	Input capacitance		10	pF	Unmeasured Pins Grounded
C_{XTL}	Input capacitance;XTL1, XTL2	23.5	29.5	pF	

AC TIMING DIAGRAM

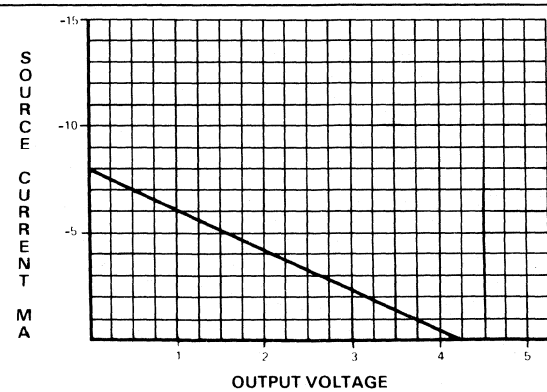
Figure 5



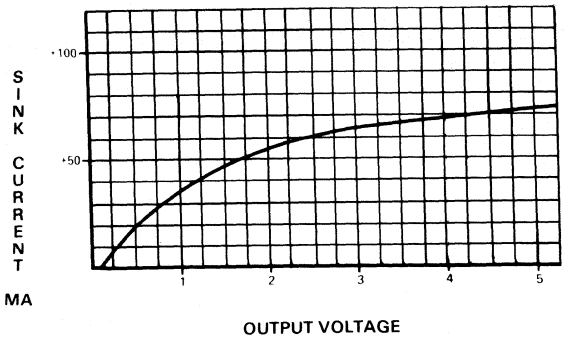
Note: All AC measurements are referenced to V_{IL} max., V_{IH} min., V_{OL} (.8 V), or V_{OH} (2.0 V).

STROBE SOURCE CAPABILITY (TYPICAL TO V_{CC} = 5 V, T_A = 25°C)

Figure 6

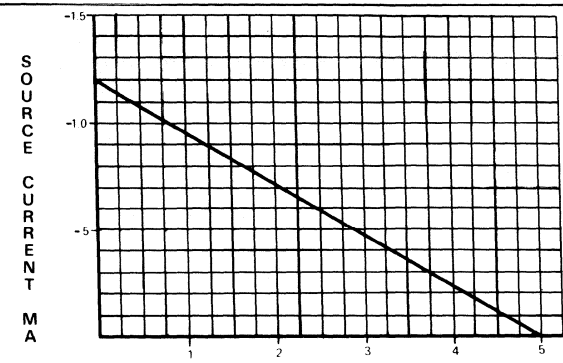


STROBE SINK CAPABILITY
(TYPICAL AT V_{CC} = 5 V, T_A = 25°C)
Figure 7

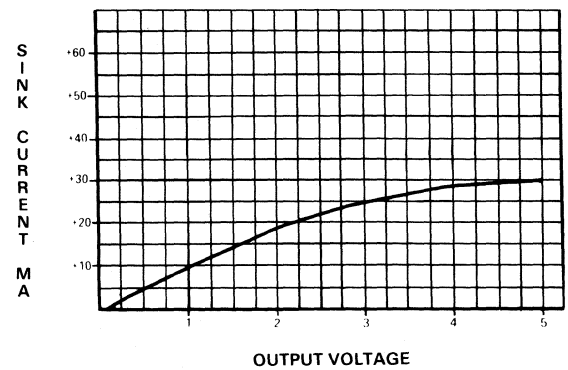


I/O PORT SOURCE CAPABILITY

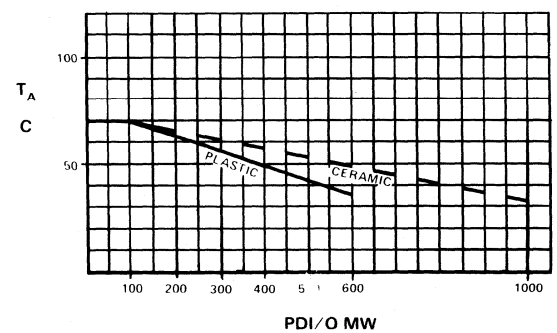
I/O PORT SOURCE CAPABILITY
(TYPICAL AT V_{CC} = 5 V, T_A = 25°C)
Figure 8



I/O PORT SINK CAPABILITY
(TYPICAL AT V_{CC} = 5 V, T_A = 25°C)
Figure 9



MAXIMUM OPERATING TEMPERATURE VS. I/O POWER DISSIPATION
Figure 10



ADVANCE INFORMATION

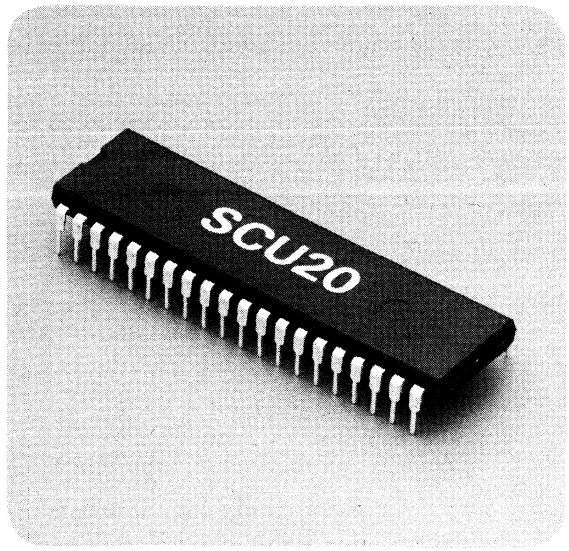
MOSTEK

3870 MICROCOMPUTER COMPONENTS

Serial Control Unit SCU20

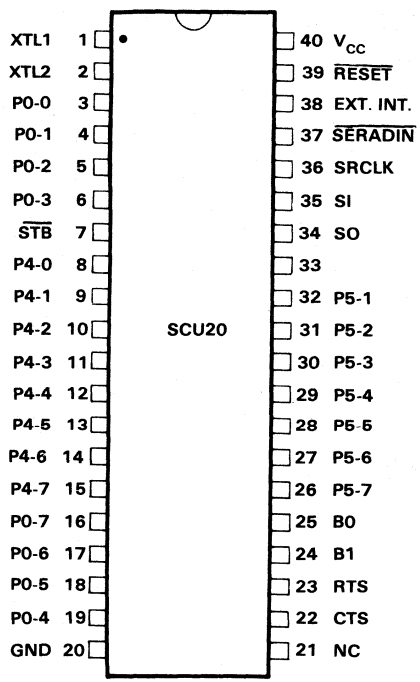
FEATURES

- ☐ Provides programmable remote I/O functions, real time operational capabilities, and standardized network communications on a 40 pin chip.
- ☐ Performs preprogrammed functions on command, including:
  - Byte input and output
  - Bit input and output
  - Set, clear, and toggle selected pins
  - Data access from real time functions
- ☐ Performs real time preprogrammed functions, including:
  - Data log on external interrupt, timer, or host control, up to 63 bytes of data
  - Five Event Counters driven from external interrupt, timer or host control
- ☐ Up to 24 programmable I/O pins
- ☐ Allows user to network up to 254 SCUs on a single communications channel
- ☐ Asynchronous serial data transmission
- ☐ Selectable baud rate (300, 1200, 2400, or 9600 Baud)
- ☐ Secure, Error resistant data link protocol
- ☐ Requires single +5 volt supply
- ☐ Low power (275mW typ)



SCU20 PINOUT

Figure 1



INTRODUCTION

The SCU20 serial control unit is a preprogrammed MK3873 single chip microcomputer. It is a general purpose remote control/data acquisition unit, with 38 preprogrammed functions available to the user.

Communications with the SCU20 takes place over an asynchronous half duplex communications channel at 300, 1200, 2400, or 9600 Baud. The communications protocol is efficient and error resistant, and yet easy to implement on the host system.

The SCU20 can be used for both monitoring and control systems where remote intelligence is required. It can be configured to provide many different input/output and data acquisition functions through its 24 I/O pins. Such intelligent functions as Data Log and Event Counters allow many different applications that will not burden the host system with constant update requirements.
requirements.

FUNCTIONAL PIN DISCRIPTION

The SCU20 is housed in a plastic 40 pin dual in-line package.

Figure 1 shows the location of each pin on the SCU20. The following describes the function of each pin.

SCU20 PINOUT DEFINITION

XTL1, XTL2 - Time base inputs for 3.6864 MHz crystal.

P0-0 - P0-7 - SCU port 0. SCU address input or general purpose data port (see SCU Address section).

$\overline{STB}$ - Data available strobe for port 4.

P4-0 - P4-7 - SCU port 4. General purpose data port.

P5-0 - P5-7 - SCU port 5. General purpose data port.

SRCLK - Clock signal generated by internal Baud rate generator.

SI - Serial input. Receives serial asynchronous data from the host.

SO - Serial output. Transmits serial asynchronous data to the host.

RTS - Request to send.

CTS - Clear to send.

$\overline{RESET}$ - External reset.

EXT. INT. - External interrupt.

$\overline{SERADIN}$ - Serial address input/address mode (see SCU Address section).

B0, B1 - Baud rate select.

V_{CC} - Power supply, 5 volts.

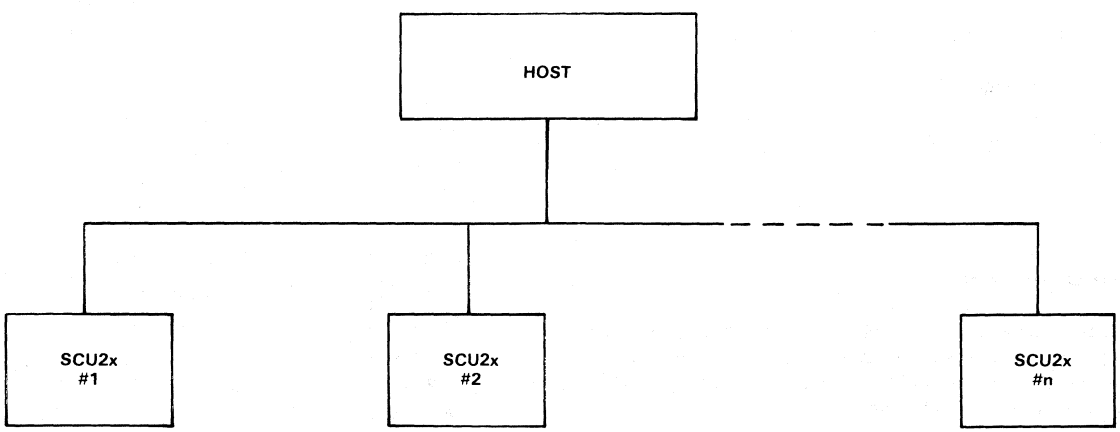
GND - Power supply ground.

SCU2 NETWORK

The SCU2 Network is a serial linked network of devices in the SCU2 family. All communications are via a common serial link using the SCU2 family communications protocol. In this way, a distributed control facility may be easily implemented from standard parts, and controlled by the host computer via the serial link.

Figure 2 illustrates the SCU2 Network.

SCU2 NETWORK
Figure 2



Each SCU2x in the network has an individual address to which it will respond. All SCU2x devices in the network are slave processors to the host, and are unable to initiate communications except in response to the host.

The operation of the SCU2 network procedes along the following lines.

When the system is initialized, all SCU2x devices are in the listen mode, and are performing no functions. The host will issue an enquiry command to each of the SCUs, each of which will respond. Once all SCUs have been queried, the host will issue commands to each SCU to set up the particular operational parameters required of it. When this has been done, the host may then use the SCUs to control equipment, measure values, etc., by issuing commands and receiving responses.

Unless issuing a response, the SCU2x is always in the listen mode. If a command has been sent to an SCU2x, a response is expected within a specific time period. If none is forthcoming, it means that the command transmitted was not successfully accumulated by the SCU. In this case, the host must take steps to either notify the operator or to retransmit the command to the SCU.

If a system error occurs in the host, it may suspend operation of the entire network by outputing the network reset command which causes all SCUs to be reset. This is the only command that does not require a specific SCU address as part of the command. It uses the system reset address which is recognized by each SCU.

SCU ADDRESS

The address to which the SCU responds may be established in one of two ways.

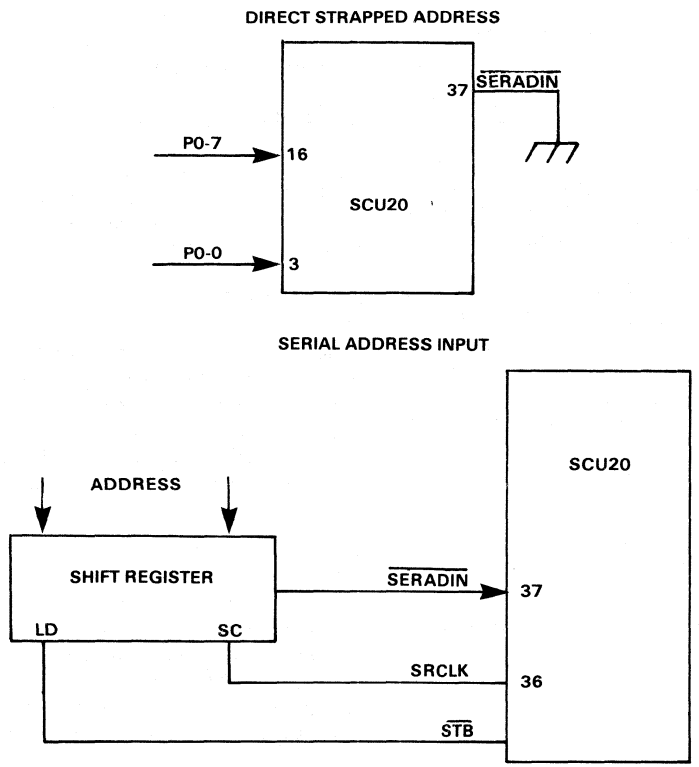
The first mode is the Direct Strapped Address mode, and is enabled by tying the $\overline{\text{SERADIN}}$ pin directly to ground. In this mode, the SCU address is strapped at port 0. Because of this, port 0 is not available as a general purpose I/O port.

The second mode is the Serial Address Input mode. The $\overline{\text{SERADIN}}$ pin is used to input the address as a serial 8-bit stream from a shift register. SRCLK is used as a shift clock for this operation. The $\overline{\text{STB}}$ signal is used at initialization time to cause the address to be loaded into the shift register before shifting begins. In the Serial Address mode, port 0 becomes available for use as a general purpose data I/O port.

Figure 3 illustrates both methods of establishing the SCU address.

SCU20 ADDRESS ESTABLISHMENT

Figure 3



SCU COMMUNICATIONS

The SCU20 communicates with the host computer over a half duplex asynchronous serial link. The communications protocol is simple, yet error resistant.

The general form of the communication message is as follows:

HDR	ADDR	CMD	DATA	DATA	• • •	LRC

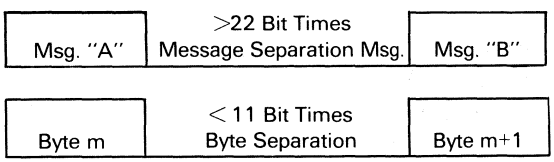
HDR - Message header. Hex '01' indicates a command message from the host; Hex '02' indicates a response from the SCU.

ADDR - SCU Address. Indicates which SCU the message is for, or originates from.

CMD - Command. Indicates the function to be performed.

DATA - Any data that may be required by the particular command.

LRC - Linear Redundancy Check.



Messages are to be transmitted in block mode, with a message separation of at least 22 bit times. Interbyte separations should be no more than 11 bit times.

A message from the host to the SCU will generate a response if there is no transmission error. If any transmission error is detected, no response will be made.

Possible transmission errors are LRC errors, parity errors, interbyte separation errors, or intermessage separation errors.

BAUD RATE SELECTION

The serial Baud rate is selected by a strapped option on the SCU. Those options are listed below:

BAUD RATE	B0 (Pin 25)	B1 (Pin 24)
300	0	0
1200	0	1
2400	1	0
9600	1	1

MODEM SIGNALS

RTS and CTS are provided to facilitate handshaking with modems. Just prior to responding to a valid command, RTS will go to logic 1, indicating that the SCU is ready to send data back to the host. CTS is an input to the SCU that is tested after RTS goes active to determine if the SCU may begin transmitting data.

PARALLEL I/O PORTS

The SCU has a minimum of 2 parallel I/O ports and a maximum of 3 available for general use, depending on the address selection mode chosen. For each of these ports, there exist 2 registers that control and modify the I/O to and from the ports. These are the Data Direction Register (DDR) and the Mask Register(MR).

The Data Direction Register defines the usage of each pin in the port. If a bit is set to 0, then the corresponding pin is used as input. If a bit is set to 1, then the corresponding bit is used as an output. When a port is read, all bits are sampled for input whether or not they are marked for input. When a port is written to, however, only those pins declared as output will be modified.

The Mask Register provides a data mask that may be applied to the input data before transmission to the master. The mask is established once and may be used repeatedly before being changed by establishing a new mask value. If a pin is to be available upon read, the corresponding bit in the mask register is set to 1, while a pin that is to be masked out will have its mask bit set to 0.

SCU PREPROGRAMMED FUNCTIONS

The SCU20 has a variety of preprogrammed functions available to the user. Each of these functions addresses a different general area of application such that the SCU20 is truly a general purpose device.

PORT COMMANDS

There are several commands which allow the host to manipulate the 8-bit general purpose I/O ports. The host may load data into any one or all of the ports, may read any or all of the ports with or without a mask, may read with a new mask, or may read using the last defined mask. When data is loaded, the resulting port state is returned in the response message.

LOGIC COMMANDS

In addition to performing data I/O with the ports, the host may perform logical operations with the ports and data from the host. These commands allow the host to AND, OR, or Exclusive OR (XOR) data with any or all of the ports, and output the result to the ports. The resultant output is returned in the command response message.

BIT COMMANDS

These commands allow the host to SET, CLEAR, TEST, or TOGGLE bits in the ports by specifying bit number (0 - 24). Any pin that is declared as an input will not be changed.

EVENT COUNTERS

There are 5 Event Counters defined in SCU20. They are 16 bit up counters, and are driven by the timer, the external interrupt, or by host command. They may be used as simple event counters, or may be used in conjunction with the Data Log, and Pulse functions.

DATA LOG

The Data Log function allows the user to command the SCU20 to log data from the ports specified in the command, and store the data in the on-board RAM. Up to 63 bytes of data may be accumulated in the log, and may be captured on external interrupt, timer, or host command through use of an Event Counter.

Data from the Log is transmitted back to the host in a single read command burst.

CONTROL COMMANDS

There are several commands to control the SCU20 as well as the entire SCU2 network. These commands provide the host the ability to query each individual SCU on the network for its type, the last message it sent, and for detailed error codes. In addition, there are commands that allow the host to reset an individual SCU, or to cause the entire SCU network to reset with a single command.

ERROR PROCESSING

The SCU does not provide a "negative acknowledge" response to command stream errors. Those errors are parity errors, LRC errors, unidentifiable commands, overrun, or violation of the separation specifications as described earlier.

In some cases, the SCU will provide error response to functional errors in commands that have been recognized. This response will be either a "NAK0" or a "NAK3" as specified for the command. "NAK0" is the hex value H'FB', and "NAK3" is the hex value H'FE'.

H'2'	ADDR	H'FB' or H'FE'	LRC

SCU COMMANDS

Figure 4 gives a complete list of the commands and functions available to the SCU20.

SCU20 COMMANDS
Figure 4

FUNCTION	COMMAND CODES	# DATA BYTES (CMD)	# DATA BYTES (RESP)	ERR COD RET
** PORT COMMANDS **				
Load Data Direction Registers	1E	3	0	-
Load Port (0, 4, 5)	00,01,02	1	1	-
Load All Ports	03	3	3	-
Read Port (0, 4, 5)	04,05,06	0	1	-
Read All Ports	07	0	3	-
Read Port Masked, Mask Provided	08,09,0A	1	1	-
Read All Ports, Masks Provided	0B	3	3	-
Read Port using Previous Mask	0C,0D,0E	0	1	-
Read All Ports using Previous Masks	0F	0	3	-

**** PORT LOGIC COMMANDS ****				
AND Data to Port	10,11,12	1	1	-
AND Data to All Ports	13	3	3	-
OR Data to Port	14,15,16	1	1	-
OR Data to All Ports	17	3	3	-
XOR Data to Port	18,19,1A	1	1	-
XOR Data to All Ports	1B	3	3	-
**** BIT COMMANDS ****				
Set Bit in Port	1F	1	0	-
Clear Bit in Port	20	1	0	-
Toggle Bit in Port	21	1	1	-
Test Bit in Port	22	1	1	-
**** EVENT COUNTERS ****				
Start Event Counter	80	1	0	-
Read Event Counter	81	1	2	NAK0
Clear Event Counter	82	1	0	NAK0
Stop Event Counter	83	1	0	NAK0
Step Event Counter	84	1	0	NAK0
**** DATA LOG COMMANDS ****				
Start Data Log	85	3	0	NAK0
Stop and Read Data Log	86	0	var.	-
Read Data Log Count	87	0	1	-
**** SCU CONTROL COMMANDS ****				
Enquiry	1C	0	var.	NAK3
Return SCU Type	1D	0	1	-
Read Error Code	F7	0	1	-
Reset SCU (2 Command Sequence)	F8/F9	0	0	-
General Reset (SCU Network)	FF	0	-	-

PRELIMINARY

MOSTEK®

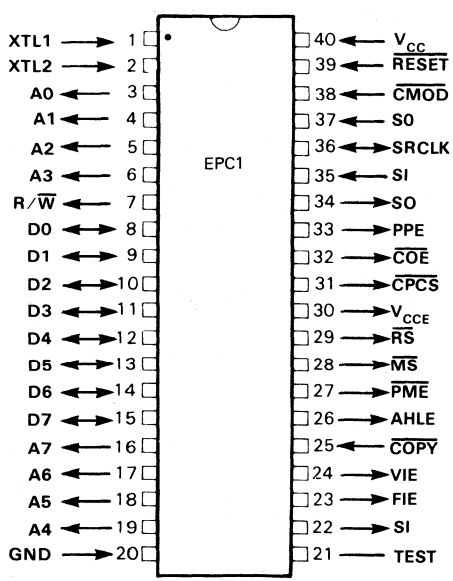
MICROCOMPUTER COMPONENTS

EPROM Programming Controller EPC1

FEATURES

- ☐ Preprogrammed MK3873 single chip microcomputer for use as a controller in a PROM programmer
- ☐ Controls programming sequence for 2758, 2716, 2516, 2532, 2732, and 2764 type EPROMs directly
- ☐ Operates in three modes of operation
  - Stand alone EPROM Programmer
  - Peripheral EPROM Programmer
  - EPROM copier
- ☐ Performs the following functions:
  - Program
  - Duplicate
  - Verify
  - Edit
- ☐ Direct interface to ASCII terminal or host computer via RS232 compatible serial link
- ☐ Personality Module recognition

PIN FUNCTIONS



PINOUT CROSS REFERENCE

3873	EPC1
XTL1	XTL1
XTL2	XTL2
$\overline{P0\text{-}0}$	A0
$\overline{P0\text{-}1}$	A1
$\overline{P0\text{-}2}$	A2
$\overline{P0\text{-}3}$	A3
$\overline{STROBE}$	R/$\overline{W}$
$\overline{P4\text{-}0}$	D0
$\overline{P4\text{-}1}$	D1
$\overline{P4\text{-}2}$	D2
$\overline{P4\text{-}3}$	D3
$\overline{P4\text{-}4}$	D4
$\overline{P4\text{-}5}$	D5
$\overline{P4\text{-}6}$	D6
$\overline{P4\text{-}7}$	D7
$\overline{P0\text{-}7}$	A7
$\overline{P0\text{-}6}$	A6
$\overline{P0\text{-}5}$	A5
$\overline{P0\text{-}4}$	A4
GND	GND
V_{CC}	V_{CC}
$\overline{RESET}$	$\overline{RESET}$
EXT INT	CMOD
$\overline{P1\text{-}3}$	SO
SRCLK	SRCLK
SI	SI
SO	SO
$\overline{P5\text{-}0}$	PPE
$\overline{P5\text{-}1}$	$\overline{COE}$
$\overline{P5\text{-}2}$	$\overline{CPCS}$
$\overline{P5\text{-}3}$	V_{CCE}
$\overline{P5\text{-}4}$	$\overline{RS}$
$\overline{P5\text{-}5}$	$\overline{MS}$
$\overline{P5\text{-}6}$	$\overline{PME}$
$\overline{P5\text{-}7}$	AHLE
$\overline{P1\text{-}7}$	$\overline{COPY}$
$\overline{P1\text{-}6}$	VIE
$\overline{P1\text{-}5}$	FIE
$\overline{P1\text{-}4}$	SI
TEST	TEST

All parallel port pins are TTL compatible and have standard type output buffers as documented in the 3873 Data Sheet. RESET has no pull-up.

PIN DESCRIPTION

A_7 - A_0 Multiplexed Address Bus Lines outputs, active high. These eight pins are used to output a multiplexed 16 bit address bus. The state of the signals determine whether a high order address (A_{15} - A_7) or a low order address (A_6 - A_0) is present on these lines.

D_7 - D_0 Data Bus Lines; bidirectional active high. The data bus is used for exchanges between ROM, ERPOM or RAM memory.

$\overline{AHLE}$ Address High Latch Enable output, active low. This line is used to indicate that a valid high order memory address is valid on the A_7 - A_0 multiplexed address output lines. This signal is normally used to latch this high order address.

$\overline{MS}$ Master Select output, active low. This line is used to signal that a valid memory address has been formed for a memory read operation of the Master EPROM.

$\overline{CS}$ Copy ROM Chip Select output, active low. When low, the COE output is used to enable the tri-state bus drivers on the memory device so that data will be gated onto the bidirectional data bus.

PPE Programming Power Enable output active high. This signal is used to enable the high voltage power supply during the programming sequence.

$\overline{COPY}$ COPY input; active low. The copy signal is a debounced pushbutton switch input which is used to initiate a copy operation of the contents of the Master EPROM onto a blank Copy EPROM.

SI - Serial Input input, active high. Serial Input is used to receive asynchronous serial ASCII data.

SO - Serial Output output, active high. Serial Output is used to transmit asynchronous serial ASCII data.

$\overline{CMOD}$ Copy Mode Input with internal pullup; active low. When held low this pin is used to select the EPROM copying mode of EPC1.

S1, S0 Baud rate select Input. Strapping of these pins determines the baud rate for the EPC1.

XTL1, XTL2 Crystal Input Pins. These lines are tied directly to an on board crystal oscillator circuit. A time base is established by connecting a crystal directly to these pins. A 3.6864 MHz crystal must be used in order for standard Baud rates to be achieved at the serial port.

VIE - Verify Indicator Enable, output, active high. This signal is active when the PROM Programming Controller is in the process of a PRO verification.

FIE - Failure Indicator Enable, output, active high. This signal is active when the PROM Programming Controller has encountered a programming failure during the verification process.

$\overline{RS}$ - RAM buffer select; output, active low. RAM select is used to signal that a valid memory address has been formed for a memory read or memory write operation of the RAM buffer.

R/$\overline{W}$ - Read/$\overline{Write}$ select; output, active high or low. Read/$\overline{Write}$ is used to designate either a read or a write operation to the RAM buffer control circuitry.

V_{CC} and GND. Power Supply LIne. V_{CC} = +5 V ± 10%.

TEST - Test Input. Test is in input reserved for use by Mostek for final testing of the PROM Programming Controller. In normal circuit operation it should be left open or grounded.

SRCCK - Serial Clock, output. SRCCK is the clock used to shift data into or out of the PROM Programming Controller.

V_{CCE} - V_{CC} Enable, Output, active high. This signal is used to enable V_{CC} power (+5 V) to the V_{PP} input of the COPY PROM.

$\overline{RESET}$ - Reset line Input with no pull-up, active low. $\overline{RESET}$ may be used to externally reset the PROM programming Controller. When pulled low, the PPC will reset. When then allowed to go high, the EPC 1 will begin its initialization sequence.

GENERAL DESCRIPTION

The EPC1 is a pre-programmed MK3873 single chip microcomputer. It has the ability to read and program the +5V only MOS EPROMs which are listed below in Table 1. The on-chip serial I/O port of the MK3873 has been used to implement a communications channel which allows the EPC1 to interface directly to an RS232 driver/receiver pair so that commands and data can be received from a standard ASCII terminal or from a host computer. A block diagram of an EPROM programmer utilizing the EPC1 is shown in Figure 1. The EPC1 operates in three different operating modes:

1) Stand Alone Mode
2) Peripheral Mode
3) EPROM Copier Mode

The system configuration for each of these modes is described pictorially in Figure 2.

EPROMs WHICH MAY BE PROGRAMMED BY THE EPC1

Table 1

EPROM	ORGANIZATION	# OF PINS
2758	1K x 8	24
MK2716	2K x 8	24
2516	2K x 8	24
2532	4K x 8	24
2732	4K x 8	24
MK2764	8K x 8	28

A multiplexed 16 bit address bus appears on Port 4 of the 3873, and a bi-directional 8 bit data bus is implemented on Port 5. The address and data bus are used to perform memory access operations on the Master EPROM, the copy EPROM, the RAM buffer, and the personality module ROM. The 16 bit address facilitates access from memory device up to 64K bytes long.

Six different control signals have been implemented on the 3873 to control memory access operations. Master Select ($\overline{MS}$) is used as a chip enable signal to the Master EPROM. Copy Select ($\overline{CPCS}$) and Copy Output Enable ($\overline{COE}$) are used as a chip enable signal and tri-state output buffer enable signal, respectively, for the Copy EPROM. RAM Select ($\overline{RS}$) is used as a chip enable signal for the RAM buffer. Read/Write is used to distinguish between read and write operations to the RAM buffer.

On the EPC1, control signals are implemented for use during a programming operation on the blank Copy ROM. Programming Power is turned on by Programming Power Enable (PPE). Programming Power Enable is used to turn on the high voltage programming power supply at the beginning of the programming sequence. $\overline{CPCS}$ or $\overline{OE}$ are used to latch data into the Copy ROM.

The EPC1 provides a control signal which can be used as a chip enable for an external personality module ROM. This ROM contains encoded information which is used to instruct the EPC 1 how to read and program an EPROM which is not directly accommodated by the EPC 1. A 82S123 256-bit bipolar PROM (32 x 8) may be used to serve this function.

An on-chip serial I/O port can be used to provide an RS232 interface between an ASCII terminal or a remote host computer. The Baud rate can be selected by strapping of pins S1 and S0 as shown below:

BAUD RATE	S1	S0
300	0	0
1200	0	1
2400	1	0
9600	1	1

STAND ALONE MODE

In this operating mode, the EPC1 accepts commands directly from an ASCII terminal. A command received from an ASCII terminal will cause the EPC1 to perform one of the following functions:

Load - A block of locations - the Master EPROM are loaded into the RAM buffer.

Duplicate - The contents of the copy EPROM are programmed with the exact contents of the Master EPROM.

Edit - The contents of the RAM buffer may be examined and/or changed.

Program - A block of locations in the Copy EPROM are programmed with the contents of the RAM buffer. A blank location check is performed prior to programming.

Verify - A block of locations in the Copy EPROM are checked against those of the RAM buffer.

PERIPHERAL MODE

In this mode of operation, the EPC1 acts as an intelligent peripheral to a central or host computer. Commands and data may be sent down from the host computer via the RS232 interface. In the Peripheral Mode, the EPC1 can perform all of the functions which were described in the stand alone mode. Data which is to be programmed into an EPROM may be transmitted to the EPC1 in Intel Hex compatible object output format.

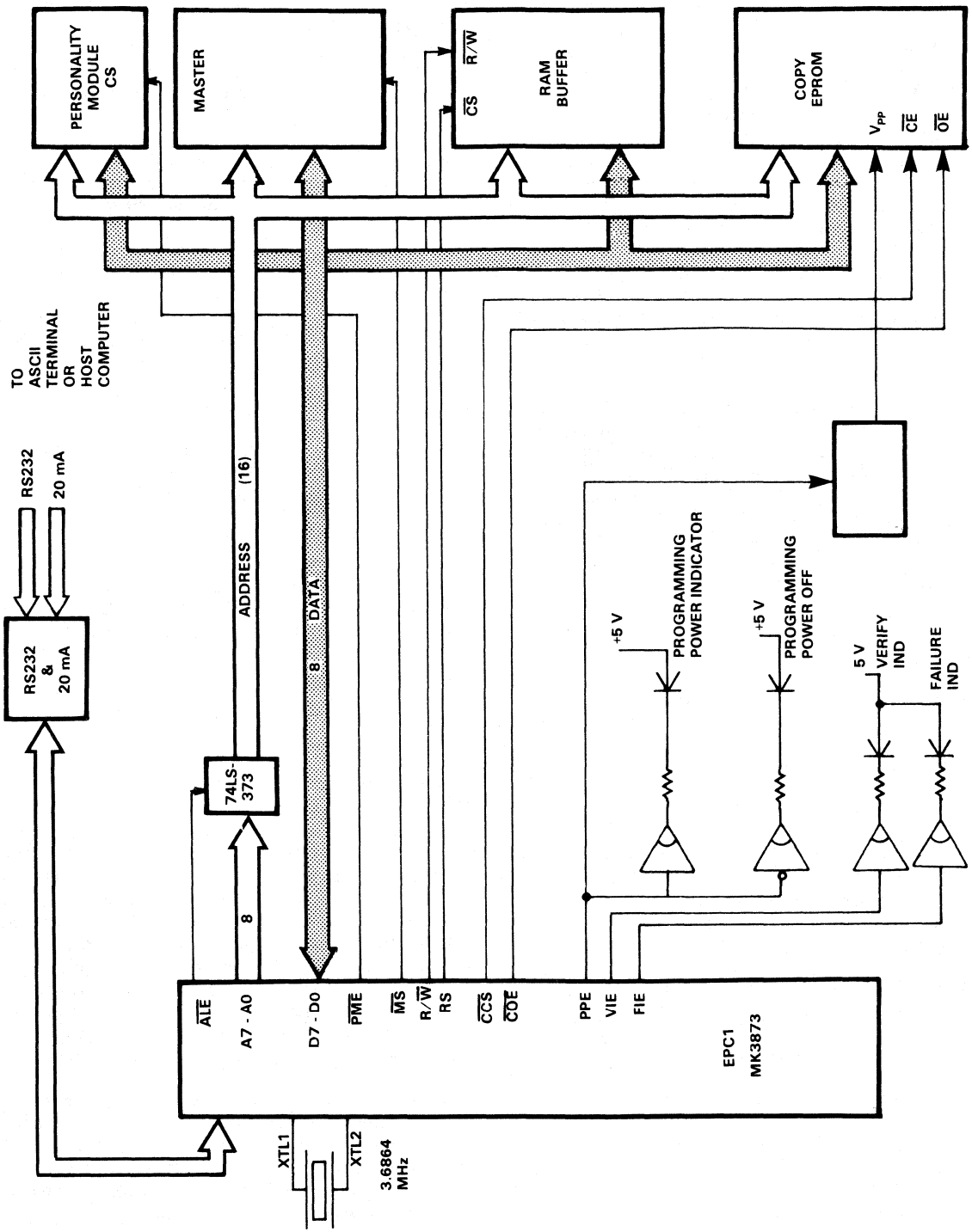
EPROM COPIER MODE

The EPC1 can also be used strictly as an EPROM copier in an environment where one or more copies of a Master EPROM need to be made as quickly as possible. An example of this type of environment would be a system production assembly area. The same hardware configuration as shown in the block diagram in Figure 1 can be used to implement the EPC1 in the EPROM copier mode. The presence of the Personality Module ROM is required in this operating mode. The copying sequence is started by depressing a pushbutton switch tied directly to the COPY input line. This switch input is automatically debounced in software.

There are four indicator lamps which can be driven by the PROM Programming Controller. These are used to indicate when the PROM Programming Power is applied, when PROM Programming Power is off, when a verification operation is in progress, and when a programming failure has been detected.

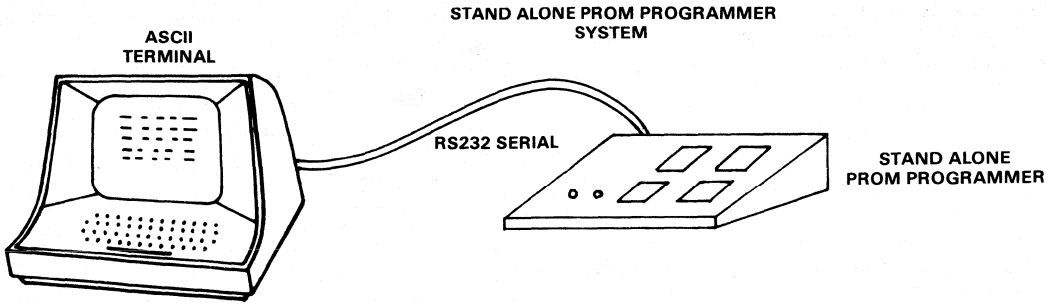
EPC1 BASED EPROM PROGRAMMER

Figure 1



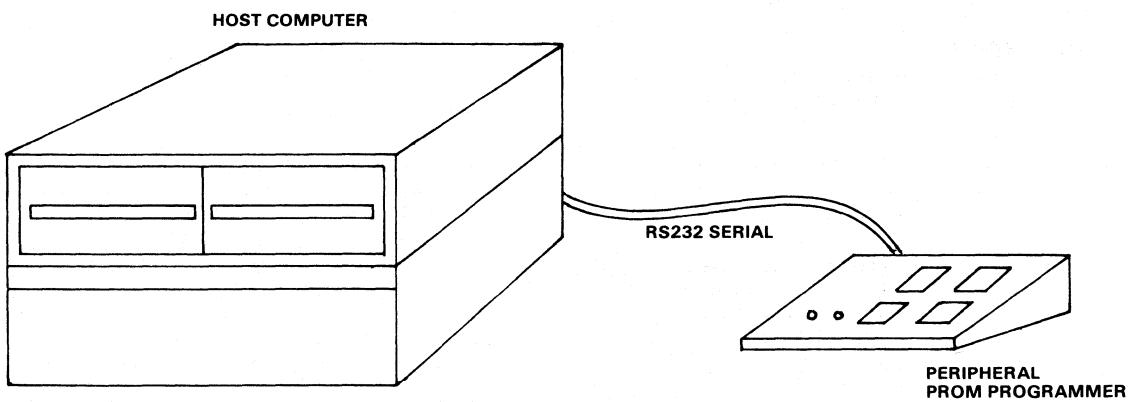
EPC1 SYSTEM CONFIGURATIONS

Figure 2



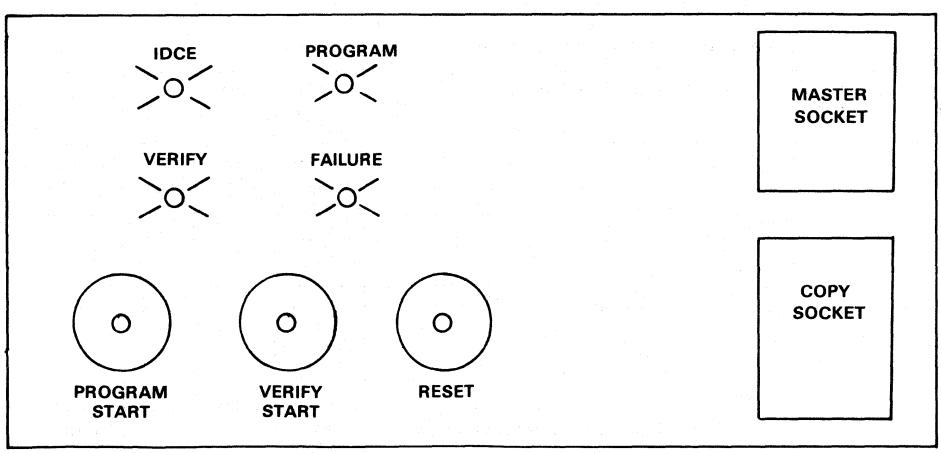
PERIPHERAL PROM PROGRAMMER SYSTEM

Figure 2b



PROM COPIER SYSTEM

Figure 3



ELECTRICAL SPECIFICATIONS

The EPC1 is a pre-programmed MK3873. Control signals on EPC1 are functions assigned to parallel port pins, serial port pins, and EXT INT line. Therefore, electrical specs for EPC1 are identical with the MK3873.

OPERATING VOLTAGES AND TEMPERATURES

Operating Voltage V_{CC}	Operating Temperature T_A
+5 V ± 10%	0° - 70°C

ABSOLUTE MAXIMUM RATINGS -00

Temperature Under Bias ..... -20°C to +85°C
Storage Temperature Voltage on any Pin With Respect to Ground
(Except open drain pins and TEST) ..... -1.0 V to +7 V
Voltage on TEST with Respect to Ground ..... -1.0 V to + 9 V
Voltage on Open Drain Pins With Respect to Ground ..... -1.0 V to +13.5 V
Power Dissipation ..... 1.5 W
Power Dissipation by any one I/O pin[2] ..... 60 mW
Power Dissipation by all I/O pins[2] ..... 600 mW

*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.

AC CHARACTERISTICS

T_A, V_{CC} within specified operating range.
I/O Power Dissipation ≤ 100 mW (Note 2)

SIGNAL	SYM	PARAMETER	MIN	MAX	UNIT	NOTES
XTL1 XTL2	t_o	Time Base Period, all clock modes	250	500	ns	4 MHz - 2 MHz 3.6864 required for
	$t_{ex(H)}$ $t_{ex(L)}$	External clock pulse width high External clock pulse width low	90 100	400 400	ns ns	standard baud frequencies
Φ	t_Φ	Internal Φ clock	$2t_0$			
WRITE	t_w	Internal WRITE Clock period	4tΦ 6tΦ			Short Cycle Long Cycle
I/O	$t_{dI/O}$	Output delay from internal WRITE clock	0	1000	ns	50 pF plus one TTL load
	$t_{sI/O}$	Input setup time to internal WRITE clock	1000		ns	
$\overline{STROBE}$	$t_{I/O-s}$	output valid to $\overline{STROBE}$ delay	3tΦ -1000	3tΦ +250	ns	I/O load = 50 pF + 1 TTL load
	t_{sL}	$\overline{STROBE}$ low time	8tΦ -250	12tΦ +250	ns	STROBE load = 50 pF + 3TTL loads
$\overline{RESET}$	t_{RH}	$\overline{RESET}$ hold time, low	6tΦ +750		ns	
	t_{RPOC}	$\overline{RESET}$ hold time, low for power clear	power supply rise time +0.1		ms	
EXT INT	t_{EH}	EXT INT hold time in active and inactive state	6tΦ +750 2tΦ		ns ns	To trigger interrupt To trigger timer

CAPACITANCE

T_A = 25°C
All Part Numbers

SYM	PARAMETER	MIN	MAX	UNIT	NOTES
C_{IN}	Input capacitance; I/O, RESET, EXT INT, TEST		10	pF	unmeasured pins grounded
C_{XTL}	Input capacitance; XTL1, XTL2	23.5	29.5	pF	

AC CHARACTERISTICS FOR SERIAL I/O PINS

T_A, V_{CC} within specified operating range.
I/O Power Dissipation ≤ 100 mW (Note 2)

SIGNAL	SYM	PARAMETER	MIN	MAX	UNIT	CONDITIONS
SRCLK	$t_{C(SRCLK)}$	Serial Clock Period in External Clock Mode	3.3	∞	μs	
	$t_{W(SRCLKH)}$	Serial Clock Pulse Width, High. External Clock Mode	1.3	∞	μs	
	$t_{W(SRCLKL)}$	Serial Clock Pulse Width, Low. External Clock Mode	1.3	∞	μs	
	$t_{r(SRCLK)}$	Serial Clock Fall Time Internal Clock Mode		60	ns	0.8 V - 2.0 V C_L = 100 pf
	$t_{f(SRCLK)}$	Serial Clock Fall Time Internal Clock Mode		30	ns	2.4 V - 0.4 V C_L = 100 pf
SI	$t_{S(SI)}$	Setup Time To Rising Edge of SRCLK (SYNC Mode)	0		ns	
	$t_{H(SI)}$	Hold Time From Rising Edge of SRCLK (SYNC Mode)	1500		ns	
SO	$t_{D(SO)}$	Data Output Delay From Falling Edge of SRCLK (SYNC Mode)		1190	ns	

AC CHARACTERISTICS FOR MK38P73

(Signals brought out at socket)
T_A, V_{CC} within specified operating range.
I/O Power Dissipation ≤ 100 mW (Note 2)

SYMBOL	PARAMETER	MIN	MAX	UNIT	CONDITION
t_{aas}*	Access time from Address A_{11} - A_0, stable until data must be valid at D_7 - D_0	650		ns	Φ = 2.0 MHz

*See Table in Figure 13.

DC CHARACTERISTICS

T_A, V_{CC} within specified operating range.
I/O Power Dissipation ≤ 100 mW (Note 2)

SYM	PARAMETER	MIN	MAX	UNIT	DEVICE
I_{CC}	Average Power Supply Current		138	mA	MK38P73/02 No EPROM, Outputs Open
P_D	Power Dissipation		646	mW	MK38P73/02 No EPROM, Outputs Open

DC CHARACTERISTICS

T_A, V_{CC} within specified operating range
I/O Power Dissipation ≤ 100 mW (Note 2)

SYM	PARAMETER	MIN	MAX	UNIT	CONDITIONS
V_{IHEX}	External Clock input high level	2.4	5.8	V	
V_{ILEX}	External Clock input low level	-.3	.6	V	
I_{IHEX}	External Clock input high current		100	μA	$V_{IHEX} = V_{CC}$
I_{ILEX}	External Clock input low current		-100	μA	$V_{ILEX} = V_{SS}$
$V_{IHI/O}$	I/O input high level	2.0	5.8	V	standard pull-up (1)
		2.0	13.2	V	open drain (1)
V_{IHR}	Input high level, $\overline{RESET}$	2.0	5.8	V	standard pull-up (1)
		2.0	13.2	V	No pull-up
V_{IHEI}	Input high level, EXT INT	2.0	5.8	V	standard pull-up (1)
		2.0	13.2	V	No pull-up
V_{IL}	I/O ports, $\overline{RESET}$[1], EXT INT[1] input low level	-.3	.8	V	(1)
I_{IL}	Input low current, standard pull-up pins		-1.6	mA	$V_{IN} = 0.4$ V
I_L	Input leakage current, open drain pins $\overline{RESET}$ and EXT INT inputs with no pull-up resistor		+10 -5	μA μA	$V_{IN} = 13.2$ V $V_{IN} = 0.0$V
I_{OH}	Output high current, standard	-100		μA	$V_{OH} = 2.4$ V
	pull-up pins	-30		μA	$V_{OH} = 3.9$ V
I_{OHDD}	Output high current,	-100		μA	$V_{OH} = 2.4$ V
	direct drive pins	-1.5	 -8.5	mA mA	$V_{OH} = 1.5$ V $V_{OH} = 0.7$ V
I_{OL}	Output low current, I/O ports	1.8		mA	$V_{OL} = 0.4$ V
I_{OHS}	$\overline{STROBE}$ Output High current	-300		μA	$V_{OL} = 2.4$ V
I_{OLS}	$\overline{STROBE}$ output low current	5.0		mA	$V_{OL} = 0.4$ V

DC CHARACTERISTICS FOR MK38P73

(Signals brought out at socket)

T_A, V_{CC} within specific operating range, I/O Power Dissipation $\leq$ 100 mW. (Note 2)

SYM	PARAMETER	MIN	MAX	UNIT	CONDITION
I_{CCE}	Power Supply Current for EPROM		-185	mA	
V_{IL}	Input Low Level Data bus in	-0.3	0.8	V	
V_{IH}	Input High Level Data bus in	2.0	5.8	V	
I_{OH}	Output High Current	-100 -30		μA μA	V_{OH} = 2.4 V V_{OH} = 3.9 V
I_{OL}	Output Low Current	1.8		mA	V_{OL} = 0.4 V
I_{IL}	Input Leakage Current		10	μA	Data Bus in Float

DC CHARACTERISTICS FOR SERIAL PORT I/O PINS

T_A, V_{CC} within specified operating range

I/O Power Dissipation $\leq$ 100 mW (Note 2)

SYM	PARAMETER	MIN	MAX	UNIT	TEST CONDITIONS
V_{IHS}	Inut High for SI, SRCLK	2.0	5.8	V	
V_{ILS}	Input Low level for SI, SRCLK	-.3	.8	V	
I_{ILS}	Input low current for SI, SRCLK		-1.6	mA	V_{IL} = 0.4 V
I_{OHSO}	Output High Current SO	-100 -30		μA μA	V_{OH} = 2.4 V V_{OL} = 3.9 V
I_{OLSO}	Output Low Current SO	1.8		mA	V_{OL} = 0.4 V
I_{OHSRC}	Output High Current SRCLK	-300		μA	V_{OH} = 2.4 V
I_{OLSRC}	Output Low Current	5.0		mA	V_{OL} = 0.4 V

NOTES

1. $\overline{\text{RESET}}$ and EXT INT have internal Schmitt triggers giving minimum .2 V hysteresis.
2. Power dissipation for I/O pins is calculated by $\Sigma(V_{CC} - V_{IL})(I_{IL}) + \Sigma(V_{CC} - V_{OH})(I_{OH}) + \Sigma(V_{OL})(I_{OL})$

TIMER AC CHARACTERISTICS

Definitions:

Error = Indicated time value - actual time value

tpsc = t Φ x Prescale Value

Interval Timer Mode:

Single interval error, free running (Note 3) ..... $\pm 6t\Phi$
Cumulative interval error, free running (Note 3) 0
Error between two Timer reads (Note 2) ..... $\pm$(tpsc + tΦ)
Start Timer to stop Timer error (Notes 1,4) ..... +tΦ to —(tpsc +tΦ)
Start Timer to read Timer error (Notes 1,2) ..... —5tΦ to —(tpsc + 7tΦ)
Start Timer to interrupt request error (Notes 1,3) ..... —2tΦ to —8tΦ
Load Timer to stop Timer error (Note 1) ..... +tΦ to —(tpsc + 2tΦ)
Load Timer to read Timer error (Notes 1,2) ..... —5tΦ to —(tpsc + 8tΦ)
Load Timer to interrupt request error (Notes 1,3) ..... —2tΦ to —9tΦ

Pulse Width Measurement Mode:

Measurement accuracy (Note 4) ..... +tΦ to —(tpsc + 2tΦ)
Minimum pulse width of EXT INT pin 2tΦ

Event Counter Mode:

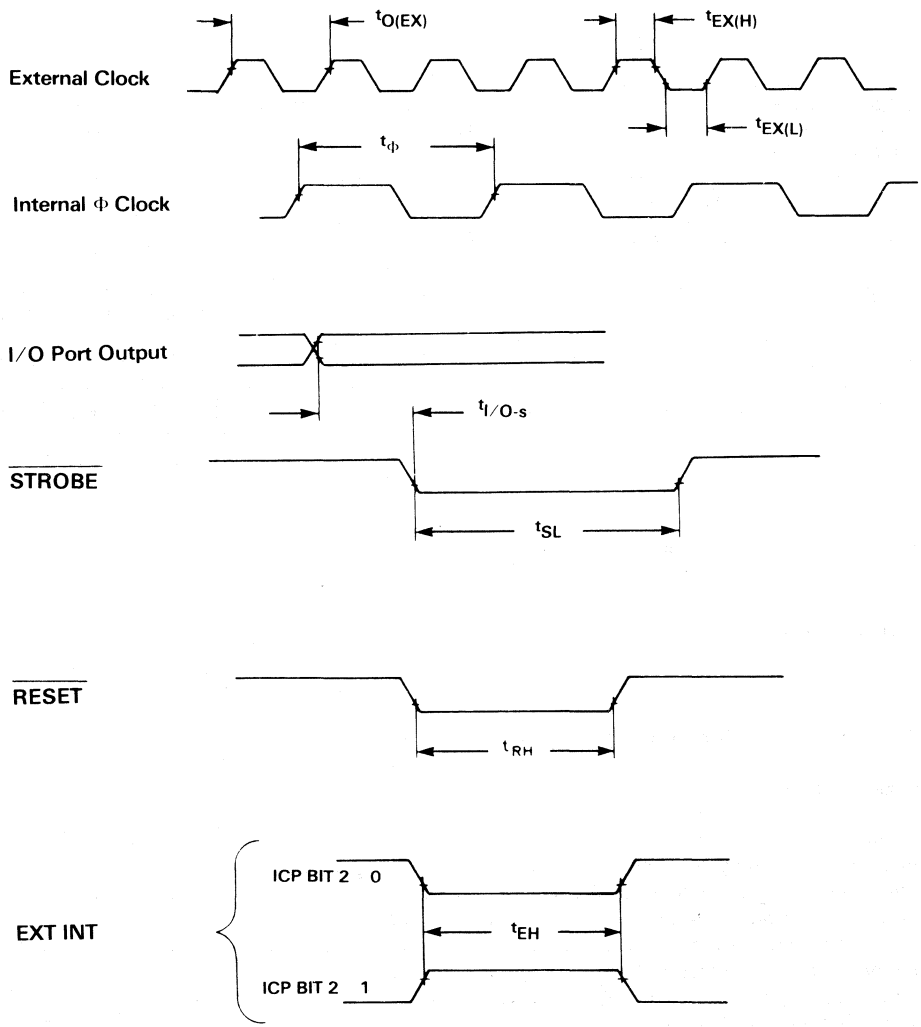
Minimum active time of EXT INT pin 2tΦ
Minimum inactive time of EXT INT pin 2tΦ

NOTES

1. All times which entail loading, starting, or stopping the Timer are referenced from the end of the last machine cycle of the OUT or OUTS instruction.
2. All times which entail reading the Timer are referenced from the end of the last machine cycle of the IN or INS instruction.
3. All times which entail the generation of an interrupt request are referenced from the start of the machine cycle in which the appropriate interrupt request latch is set. Additional time may elapse if the interrupt request occurs during a privileged or multicycle instruction.
4. Error may be cumulative if operation is repetitively performed.

AC TIMING DIAGRAM

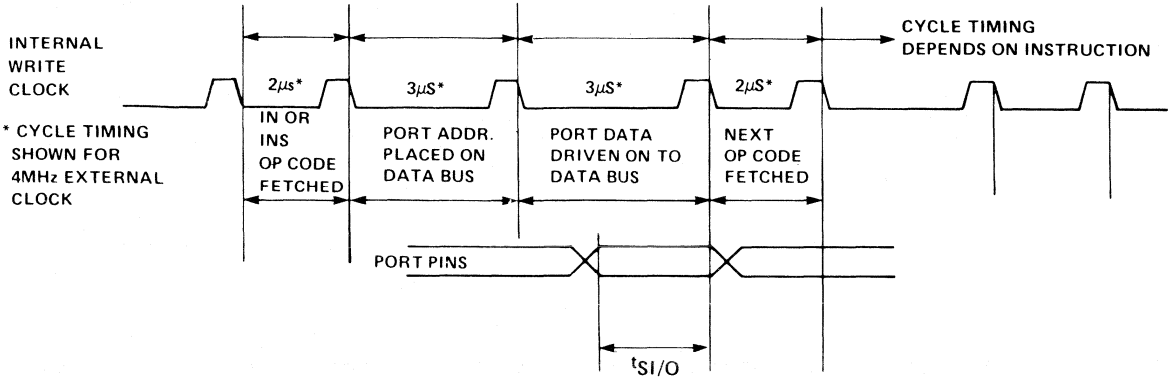
Figure 4



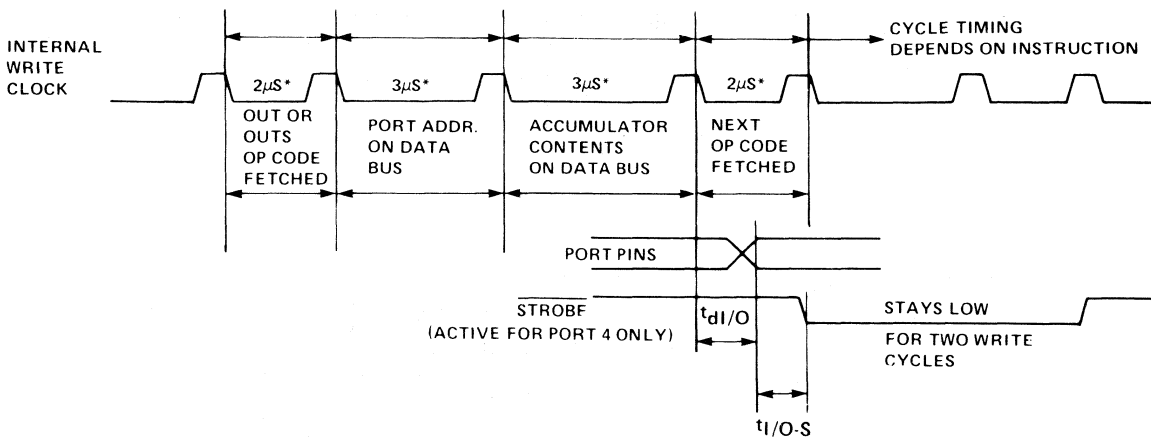
Note: All AC measurements are referenced to V_{IL} max., V_{IH} min., V_{OL} (.8 V), or V_{OH} (2.0 V).

INPUT/OUTPUT AC TIMING

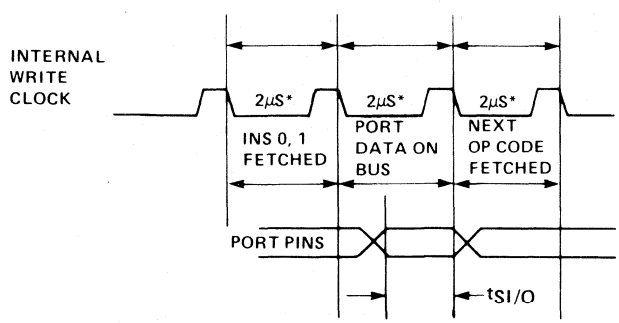
Figure 5



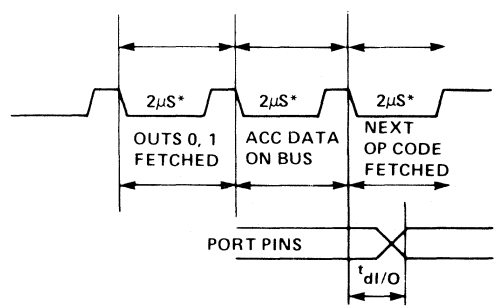
A. INPUT ON PORT 4 OR 5



B. OUTPUT ON PORT 4 OR 5



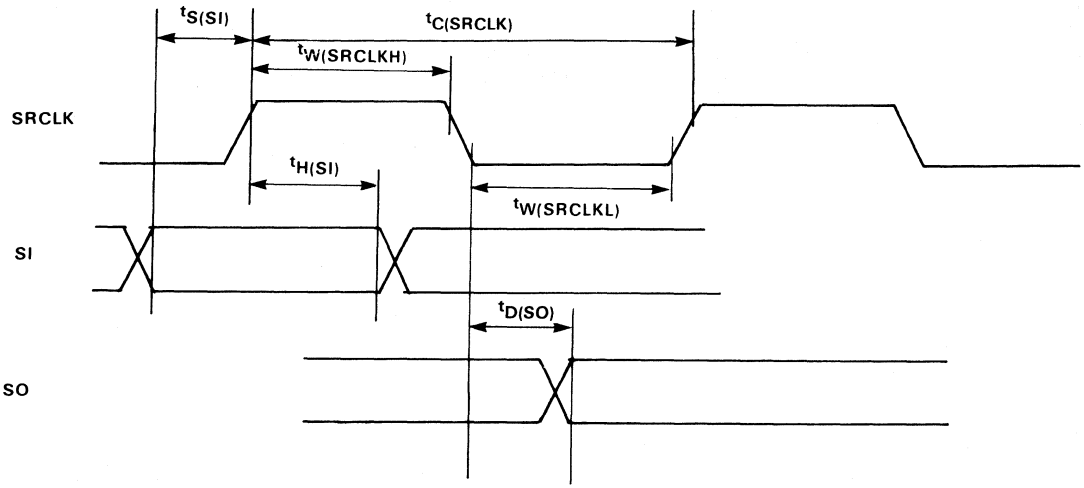
C. INPUT ON PORT 0 OR 1



D. OUTPUT ON PORT 0, 1

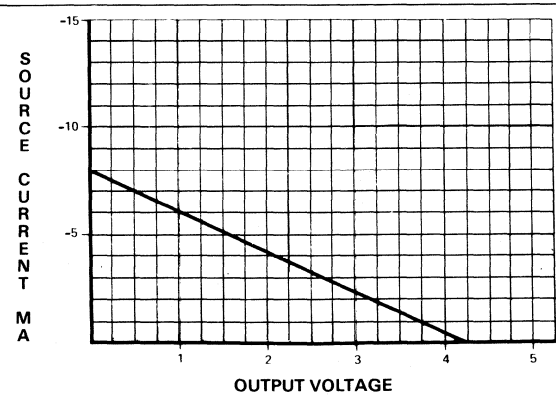
AC TIMING DIAGRAM FOR SERIAL I/O PINS

Figure 6



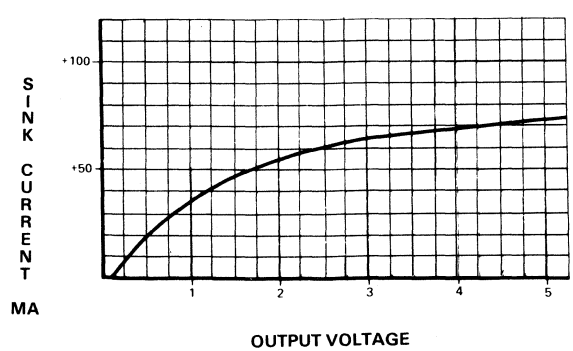
STROBE SOURCE CAPABILITY (TYPICAL TO V_{CC} = 5 V, T_A = 25°C)

Figure 7



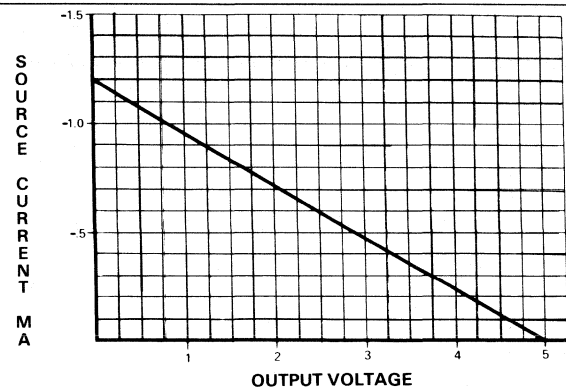
STROBE SINK CAPABILITY (TYPICAL AT V_{CC} = 5 V, T_A = 25°C)

Figure 8



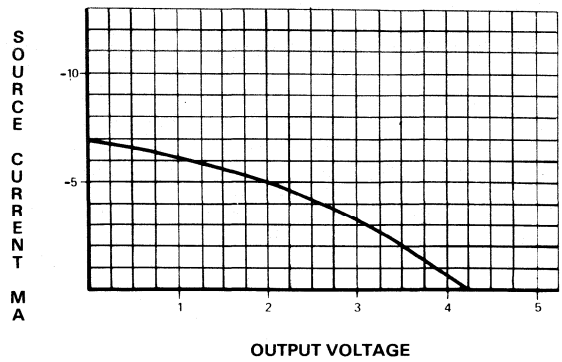
STANDARD I/O PORT SOURCE CAPABILITY (TYPICAL AT V_{CC} = 5 V, T_A = 25°C)

Figure 9



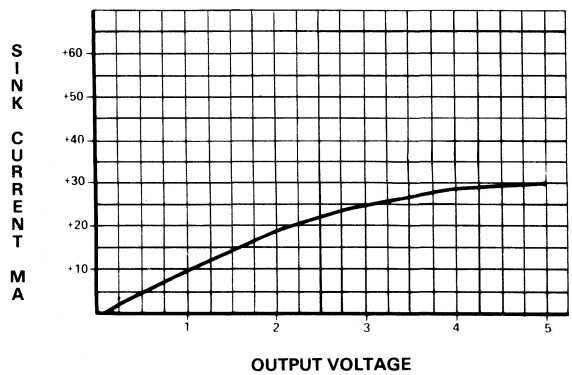
DIRECT DRIVE I/ O PORT SOURCE CAPABILITY (TYPICAL AT V_{CC} = 5 V, T_A = 25°C)

Figure 10



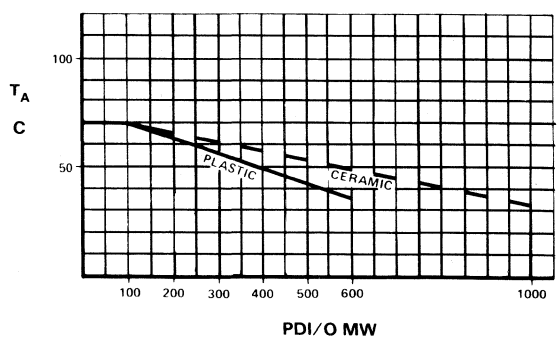
I/O PORT SINK CAPABILITY (TYPICAL AT V_{CC} = 5 V, T_A = 25°C)

Figure 11



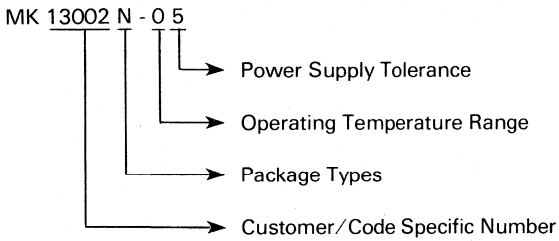
MAXIMUM OPERATING TEMPERATURE VS. I/O POWER DISSIPATION

Figure 12



DEVICE ORDER NUMBER

An example of the device order number is shown below.



The Customer/Code specific number defines the ROM bit pattern, I/O configuration, oscillator type, and generic part type to be used to satisfy the requirements of a particular customer purchase order. For further information on the ordering of mask ROM devices, the customer should refer to the 3870 Family Technical Manual.

0 = 5 V ± 10%
5 = 5 V ± 5%
0 = 0°C - +70°C
1 = -40°C - +85°C
P = Ceramic
J = Cerdip
N = Plastic

ORDERING INFORMATION

NAME	DESCRIPTION	PART NO.
EPC1	EPC1 pre-programmed MK3873 single chip microcomputer	
EPC1 OPS MANUAL	EPC1 Operations Manual	
EPC1 FIRMWARE DOCUMENTATION	Written description of EPC1 firmware with flowcharts and source listing.	

3870 FAMILY

INSTRUCTION SET SUMMARY

ACCUMULATOR GROUP INSTRUCTIONS

OPERATION	MNEMONIC	OPERAND	DESCRIPTION
Add Carry	LNK		Add the Carry bit to the contents of the Accumulator.
Add Immediate	AI	ii	Add the 8 bit immediate operand to the contents of the Accumulator.
AND Immediate	NI	ii	Perform logical AND of the 8 bit immediate operand and Accumulator.
Clear	CLR		Load Accumulator immediate with zero.
Compare	CI	ii	Non destructive subtraction of the Accumulator from the immediate operand.
Complement	COM		Performs a one's complement on the contents of the Accumulator.
Exclusive OR	XI	ii	Performs a logical Exclusive OR operation of the 8 bit immediate operand and the Accumulator.
Increment	INC		Add the value of one to the Accumulator.
Load	LI	ii	Load the value of the 8 bit immediate operand into the Accumulator.
Load	LIS	i	Load the Accumulator with the value of the 8 bit immediate operand.
OR Immediate	OI	ii	Performs a logical OR of the 8 bit immediate operand and the Accumulator.
Shift Left	SL	1	Shift the contents of the Accumulator left by one.
Shift Left	SL	4	Shift the contents of the Accumulator left by four.
Shift Right	SR	1	Shift the contents of the Accumulator right by one.
Shift Right	SR	4	Shift the contents of the Accumulator right by four.

SCRATCHPAD REGISTER INSTRUCTIONS

OPERATION	MNEMONIC OP CODE	OPERAND	DESCRIPTION
Add Scratchpad	AS	r	The contents of Scratchpad register r are added to the Accumulator.
Add Scratchpad with Decimal adjust	ASD	r	The contents of Scratchpad register r are added to the Accumulator with decimal adjust.
Decrement Scratchpad	DS	r	The contents of Scratchpad register r are decremented by one.
Load	LR	A,r	The Accumulator is loaded with the contents of Scratchpad register r.
Load	LR	A,KU	The Accumulator is loaded with the upper half of the K linkage register.
Load	LR	A,KL	The Accumulator is loaded with the lower half of the K linkage register.
Load	LR	A,QU	The Accumulator is loaded with the upper half of the Q linkage register.
Load	LR	A,QL	The Accumulator is loaded with the lower half of the Q linkage register.
Load	LR	r,A	Scratchpad register r is loaded with the contents of the Accumulator.
Load	LR	KU,A	The upper half of linkage register K is loaded with the contents of the Accumulator.
Load	LR	KL,A	The lower half of linkage register K is loaded with the contents of the Accumulator.
Load	LR	QU,A	The upper half of linkage register Q is loaded with the contents of the Accumulator.
Load	LR	QL,A	The lower half of linkage register Q is loaded with the contents of the Accumulator.
AND Scratchpad	NS	r	Scratchpad location r is logically ANDed with the contents of the Accumulator.
Exclusive OR Scratchpad	XS	r	Scratchpad location r is logically Exclusive OR'ed with the contents Accumulator.

MISCELLANEOUS INSTRUCTIONS

OPERATION	MNEMONIC OP CODE	OPERAND	DESCRIPTION
Disable Interrupts	DI		Interrupts are disabled to the 3870 CPU. ICB is reset to 0.
Enable Interrupts	EI		Interrupts are enabled to the 3870 CPU. ICB is set to 1.
Input	IN	04-FF	The contents of Port 'aa' are loaded into the Accumulator.
Input Short	INS	0-F	The contents of Port 0 or 1 are loaded into the Accumulator.
Load ISAR	LR	IS,A	The ISAR register is loaded with the contents of the Acccumulator.
Load ISAR Lower	LISL	bbb	The lower half of the ISAR register is loaded with the 3 bit immediate operand 'bbb'.
Load ISAR Upper	LISU	bbb	The upper half of the ISAR register is loaded with the 3 bit immediate operand 'bbb'.
Load Status Register *	LR	W,J	The Status Register is loaded with the contents of linkage register J.
No Operation	NOP		No Operation
Output *	OUT	04-FF	The contents of the Accumulator are output to Port 'aa'.
Output Short	OUTS	0,1	The contents of the Accumulator are output to Port 0 or 1.
Output Short *	OUTS	4,5,6,7	The contents of the Accumulator are output to Ports 0-F.
Store ISAR	LR	A,IS	The Accumulator is loaded with the contents of the ISAR register.
Store Status Register	LR	J,W	Linkage register J is loaded with the contents of the Status Register.

* Privileged Instruction.

ADDRESS REGISTER INSTRUCTION GROUP

OPERATION	MNEMONIC OP CODE	OPERAND	DESCRIPTION
Add to Data Counter	ADC		Add the contents of the Accumulator to the Data Counter DC. Result is in the Data Counter.
Call to Subroutine*	PK		The Program Counter is loaded with the value contained in linkage register K. The previous contents of P0 are saved in Stack Register P.
Call to Subroutine* Immediate	PI	aaaa	The Program Counter is loaded with the immediate operand. The previous contents of P0 are saved in the Stack Register P. The Accumulator is loaded with the upper byte of "aaaa".
Exchange DC	XDC		The contents of DC are swapped with those of DC1.
Load Data Counter	LR	DC,Q	The Data Counter is loaded with the contents of linkage register Q.
Load Data Counter	LR	DC,H	The Data Counter is loaded with the contents of linkage register H.
Load Data Counter Immediate	DCI	aaaa	The Data Counter is loaded with the immediate value "aaaa".
Load Program Counter	LR	P0,Q	The Program Counter is loaded with the contents of the Q linkage register
Load Stack Register	LR	P,K	The Stack Register is loaded with the contents of the K linkage register.
Return from Subroutine*	POP		The contents of the Program Counter are swapped with those of the Stack Register.
Store Data Counter	LR	Q,DC	Linkage register Q is loaded with the contents of the Data Counter.
Store Data Counter	LR	H,DC	Linkage register H is loaded with the contents of the Data Counter.
Store Data Counter	LR	K,P	Linkage register K is loaded with the contents of the Data Counter.

Important: The contents of the Accumulator are modified during the execution of the PI instruction.

* Privileged Instruction

JUMP AND BRANCH INSTRUCTION GROUP

OPERATION	MNEMONIC OP CODE	OPERAND	DESCRIPTION
Branch on Carry	BC	aa	Branch if Carry bit = 1.
Branch on Positive	BP	aa	Branch if the Sign bit = 1.
Branch on Zero	BZ	aa	Branch if the Zero bit = 1.
Branch on True**	BT	taa	Branch if any unmasked Status bit is true
Branch if Minus	BM	aa	Branch if the Sign bit = 0.
Branch if No Carry	BNC	aa	Branch if the Carry bit = 0.
Branch if No Overflow	BNO	aa	Branch if the Overflow bit = 0.
Branch if Not Zero	BNZ	aa	Branch if the Zero bit = 0.
Branch if False**	BF	taa	Branch if all of the unmasked Status bits are false.
Branch if ISARL ≠7	BR7	aa	Branch if the value of the lower half of the ISAR register ≠ 7.
Branch Relative	BR	aa	Branch unconditionally
Jump*	JMP	aaa	Program Counter is loaded with the immediate value 'aaaa'. The Accumulator is loaded with most significant 8 bits of aaaa.

IMPORTANT! Please be sure to take note of the above description of the execution of the JUMP instruction. The contents of the Accumulator are lost during its execution. The reason for this is that the Accumulator is used to transfer the high order byte of the destination address into the Program Counter.
*Privileged Instruction
**See description of Status bit mask under "Notes".

MEMORY REFERENCE INSTRUCTIONS

OPERATION	MNEMONIC OP CODE	OPERAND	DESCRIPTION
Add Memory	AM		Add the contents of memory location [DC] to the Accumulator.
Add Memory	AMD		Add the contents of memory location [DC] to the Accumulator with decimal adjust.
AND Memory	NM		Performs the logical AND between memory location [DC] and the Accumulator.
Compare	CM		Non-destructive subtraction of the Accumulator from the contents of memory location [DC].
Exclusive OR Memory	XM		Performs an Exclusive OR of the contents of memory location [DC] and the Accumulator.
Load Memory	LM		Load the Accumulator with the contents of memory location [DC].
OR Memory	OM		Performs a logical OR of the contents of memory location [DC] and the Accumulator.
Store	ST		Store the value of the Accumulator in memory location [DC].

Note: In all Memory Reference Instructions, the Data Counter is incremented by one.

NOTES

Lower case denotes variables specified by programmer

Function Definitions

( )	the contents of
a	Address Variable (four bits)
A	Accumulator
b	One bit immedaite operand
DC	Data Counter (Indirect Address Register)
DC1	Data Counter 1 (Auxiliary Data Counter)
DCL	Least significant 8 bits of Data Counter Addressed
DCU	Most significant 8 bits of Data Counter Addressed
H	Scratchpad Register 10 and 11 (Linkage Register)
i	Immediate oprand (four bits)
ICB	Interrupt Control Bit
IS	Indirect Scratchpad Address Register
ISL	Least Significant 3 bits of ISAR
ISU	Most Significant 3 bits of ISAR
J	Scratchpad Register 9
K	Registers 12 and 13 (Linkage Register)
KL	Register 13
KU	Register 12
P0	Program Couinter
POL	Least Significant 8 bits of Program Counter
POU	Most Significant 8 bits of Program Counter
P	Stack Register
PL	Least Significant 8 bits of Program Counter
PU	Most Significant 8 bits of Active Stack Register
Q	Registers 14 and 15 (Linkage Register)
QL	Register 15
QU	Register 14
r	Scratchpad Register (any address 0 thru B) (See Below)
W	Status Register

Scratchpad Addressing Modes Using IS. (R ≠ 0 thru B)

r=H'C'	Register Addressed by IS is (Unmodified)
r=H'D'	Register Addressed by IS is Incremented
r=H'E'	Register Addressed by IS is Decremented
r=H'F'	Illegal OP Code

Test Condition for "BT" and "BF" Instructions

BT - Branch or True
mask word (t):

3	2	1	0	← Bit #
0	X	X	X	← Mask Word 't'
OVF	ZERO	CRY	SIGN	← Status Bit

BF - Branch if False
mask word (t):

3	2	1	0	← Bit #
X		X	X	← Mask Word 't'
OVF	ZERO	CRY	SIGN	← Status Bit

MOSTEK®

MICROCOMPUTER COMPONENTS

Operations Manual

SCU1
SERIAL CONTROL UNIT 1

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

SECTION 1

GENERAL DESCRIPTION

1.1 INTRODUCTION

1.1.1 The Serial Control Unit (SCU 1) is a pre-programmed MK3870 single chip microcomputer. It acts as a complete, remote input/output controller, recognizing over 20 commands received from a host processor via a half-duplex asynchronous serial link. After performing the specified command, the SCU generally echoes the message back to the host for acknowledgement and verification. The SCU 1 may be used for both monitoring and control systems where remote intelligence is required as its 16 I/O lines may be configured to provide many different monitoring and input/output functions. The device contains a complete communications processor capable of both generating and receiving asynchronous messages.

1.1.2 A flexible and expandable protocol has been designed to allow up to 255 SCU 1's to be on a single communications channel. Included in the message protocol are parity and checksum redundancy to provide a highly error resistant message interchange. The SCU 1 has modem signals allowing easy interfacing with various serial line drivers and receivers. All input and output pins are TTL compatible.

1.2 FEATURE SUMMARY

1.2.1 A summary of the SCU 1 features is given in Table 1-1.

1.3 TYPICAL SYSTEM CONFIGURATION

1.3.1 A conceptual drawing of what might be a typical system configuration is shown in Figure 1-1.

TABLE 1-1 SCU 1 FEATURES

Provides programmable remote I/O functions as well as network communications on a single 40 pin chip.

Performs a specific preprogrammed function on command, including:

- .Bit input and bit output
- .Byte input and byte output
- .Set, clear and toggle selected pins
- .Interface to A/D converter, D/A converter or 3 1/2 digit DPM
- .Monitor input pins for a specific bit pattern

Over 20 preprogrammed functions

Allows a user to communicate with multiple SCU's over a single communications channel

Asynchronous serial data transmission

300 or 1200 Baud Data Transmission Rate with 3.6864 MHz Crystal Time Base

Secure, error resistant data link protocol

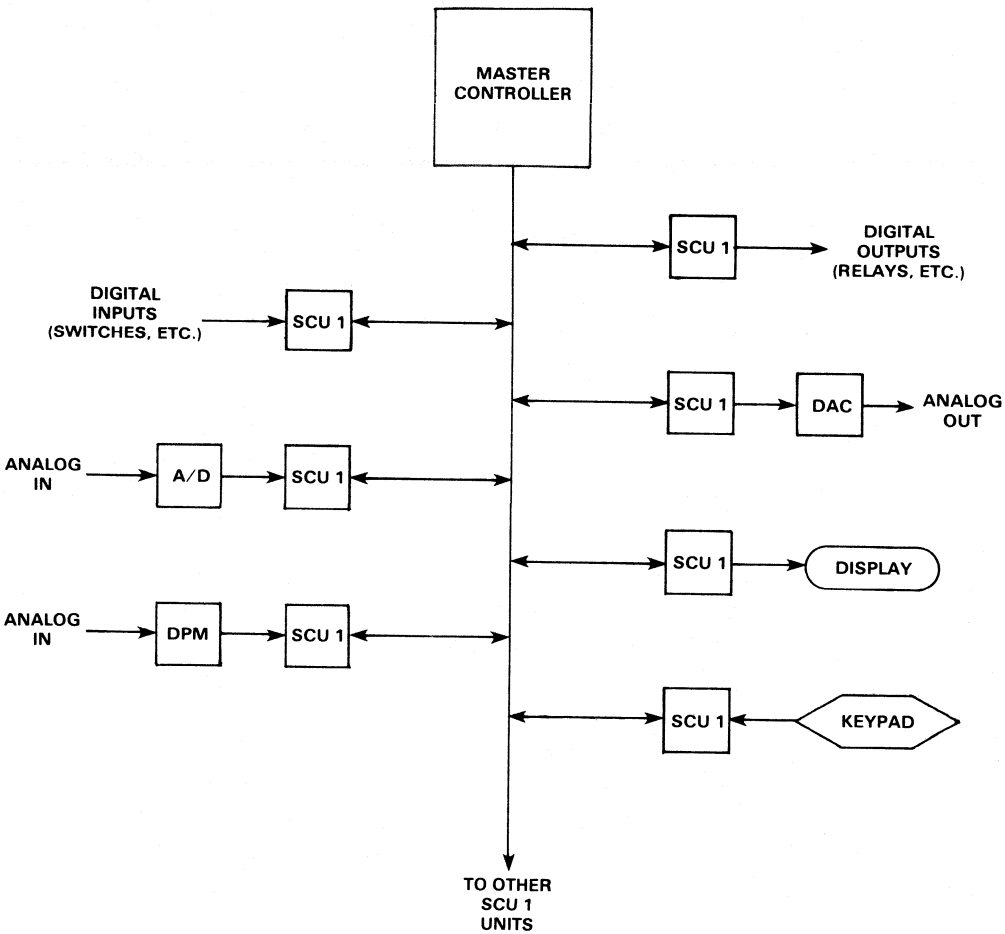
- .Parity generate and check
- .LRC generate and check
- .Efficient message format

Modem Control Signals Provided

Requires single +5 volt supply

Low power (275mW typ)

FIGURE 1-1 TYPICAL SYSTEM CONFIGURATION



SECTION 2

FUNCTIONAL PIN DESCRIPTION

2.1 SCU 1 PIN DEFINITION

2.1.1. SCU 1 is packaged in a 40-pin dual-in-line package. Figure 2-1 shows the location of each pin on SCU 1.

2.2 FUNCTIONAL PIN DESCRIPTION

2.2.1 Table 2-1 describes the function of each pin on SCU 1. Figure 2-2 shows a pin-out of SCU 1 with the pins grouped by function.

FIGURE 2-1 SCU 1 PIN DEFINITION

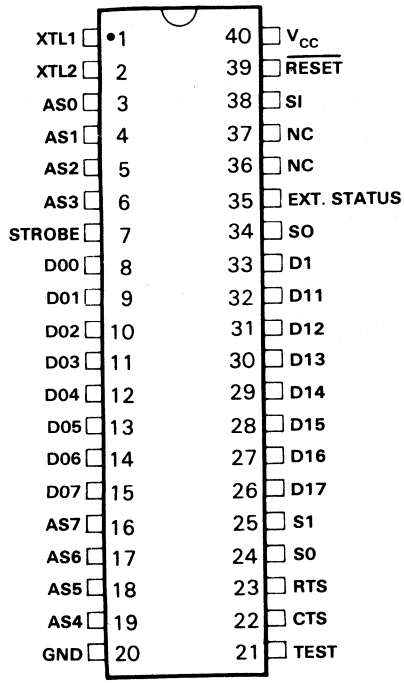


TABLE 2-1 FUNCTIONAL PIN DESCRIPTION

XTL1,2: Time base inputs for a 3.6864MHz crystal.

AS0 - AS7: SCU 1 Address (input). These 8 pins determine the address of the SCU 1, using positive logic. Internal pullup resistors allow unconnected inputs to be a logic 1. Grounded inputs are a logic 0. Address 'FF' hex is not allowed.

D00 - D07: SCU 1 port 0. These 8 pins may be TTL compatible inputs or latched outputs. All have internal pullup resistors.

D10 - D17: SCU 1 port 1. These 8 pins may be TTL compatible inputs or latched outputs. All have internal pullup resistors.

STROBE: (Output, active low). This pin provides a single low pulse after valid data has been output to port 0. It is capable of driving 3 TTL loads.

SI: Serial Input. This Schmitt trigger input receives serial, asynchronous data from the host computer.

SO: Serial Output. The SCU 1 command response is serially output through this pin, least significant bit first. Between transmissions, SO remains at a logic 1 (marking or idle line).

RTS: Request To Send (output, active high). Prior to responding to a command, RTS becomes active indicating the serial line should direct information from the SCU 1 to the host.

CTS: Clear to Send (input, active high). An active input indicates the SCU 1 may begin its response; thus it is an indication that the serial link is now ready to transmit information from the SCU 1 to the host.

RESET: External Reset (input, active low). This input may be used to guarantee the SCU 1 is held in a reset state until Vcc reaches the minimum operating voltage or to reset the device after a disturbance on the Vcc line.

S0,S1: Baud Rate Select (inputs). These inputs are strapped to specify the serial baud rate for transmit and receive. Baud rate selection is made according to the following chart.

S1	S0	Baud Rate
0	0	300
0	1	1200
1	0	1200
1	1	1200

EXT STATUS: External Status (input, active high). Used for certain commands, EXT STATUS signals when an operation is done.

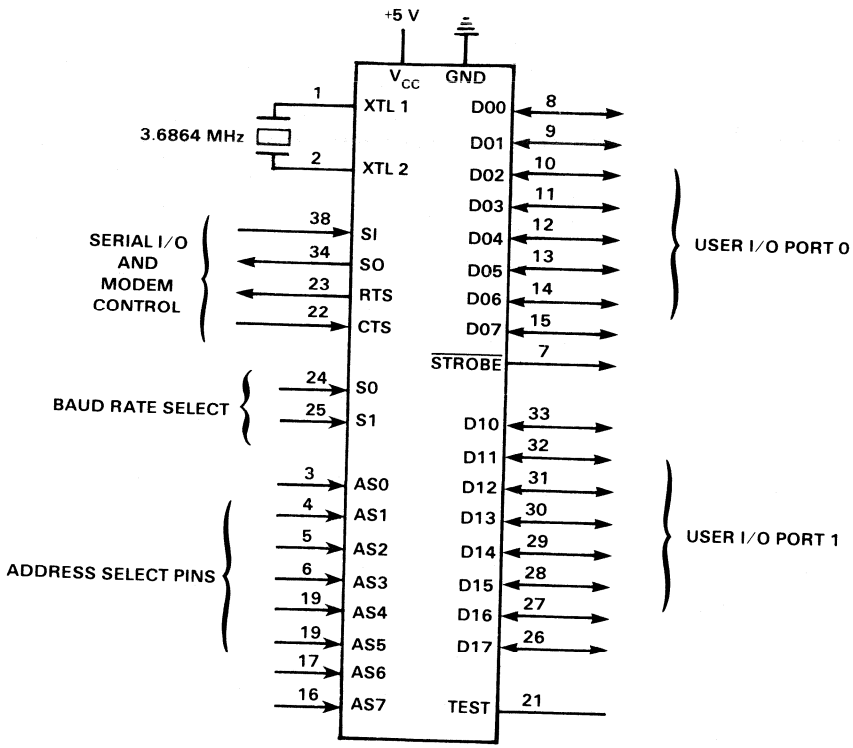
TEST: (input, active high). Test is used by Mostek to test the SCU 1. For normal circuit functionality, this pin is left unconnected or may be grounded.

NC: No Connect. These lines are undefined and should not be connected to anything.

Vcc: Power supply, +5V.

GND: Power supply ground.

FIGURE 2-2 SCU 1 FUNCTIONAL PIN DEFINITION



SECTION 3

SERIAL COMMUNICATIONS

3.1 SERIAL COMMUNICATIONS NETWORK

3.1.1 The SCU 1 is designed to communicate with a master controller (host computer) via a half-duplex (or simplex) multidrop serial link. The SCU 1 exists as a slave unit to the master controller, and each SCU 1 on a serial link must have a unique address, which identifies which SCU 1 is to respond to a particular command or inquiry from the master.

3.1.2 To prevent line connection, only the host may initiate communication. The particular SCU 1 addressed by the host will respond only after receiving a valid command. After the particular SCU 1 receives a valid message (address + command), it will perform the appropriate task and, in most cases, transmit a response to the host. While transmitting, the SCU 1 will ignore any serial input.

3.1.3 A system configured as described above is generally referred to as a serial polled communications network (or a polled network). As indicated, a particular SCU 1 must be polled by the host via a valid message before it may respond.

3.2 NETWORK PROTOCOL

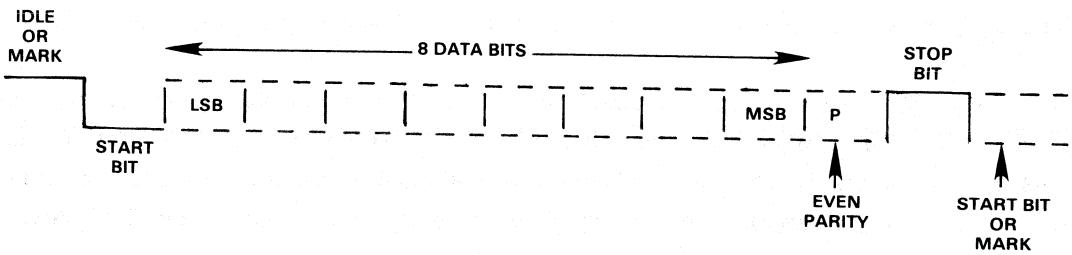
3.2.1 The SCU 1 operates with a data communications protocol defined for ease of use, high throughput, and data integrity. The protocol, though bit serial in nature, is character oriented, with 5 characters comprising a typical message. Communications is comprised of a command or inquiry message transmitted from the host to a particular SCU 1 followed, in general, by a response message from the SCU 1 to the host.

3.3 CHARACTER FORMAT

3.3.1 The network communications protocol is character oriented, with the character format as shown in Figure 3-1. The format is consistent with industry stan-

dard convention for serial asynchronous data transmission, with each character composed of 1 start bit, 8 data bits (transmitted LSB first), 1 even parity bit, and 1 stop bit. The stop bit may be followed by a start bit or by a continuous mark, or idle line, condition that exists until the start bit of the next character.

FIGURE 3-1 SCU 1 CHARACTER FORMAT

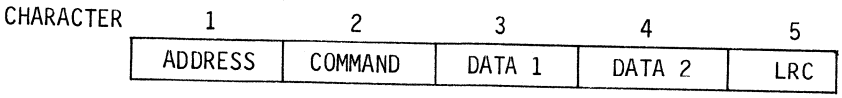


3.4 MESSAGE FORMAT

3.4.1 Network communications is comprised of command or inquiry messages transmitted from the host to a particular SCU 1 followed, in general, by a response message transmitted from the SCU 1 to the host. Figure 3-2 shows a typical message format. In order of transmission, the characters are (1) Address, (2) Command, (3) Data 1, (4) Data 2, and (5) Logitudinal Redundancy Check Character.

3.4.2 Any SCU 1 must receive a complete message from the host before it may respond. The SCU 1 will then perform the appropriate task and, in most cases, transmit a response to the host. There are 2 allowable message formats for polling (transmitting a command or inquiry from the host to the SCU 1) and 3 allowable message formats for SCU 1 response to the host.

FIGURE 3-2 TYPICAL MESSAGE FORMAT



SECTION 4

MESSAGE CHARACTER DESCRIPTION

4.1 ADDRESS CHARACTER

4.1.1 The address character is used to specify which particular SCU 1 in a network is being polled by the host. The address character is repeated by the addressed SCU 1 in the response message. Each SCU 1 in a network should have a unique address. The particular address is defined by the Address Select input pins, AS0-AS7. Up to 255 unique addresses may be specified, from '00' to FE'.

4.1.2 Address 'FF' is the universal address, and it is recognized by all SCU 1 devices for system synchronization only. No SCU 1 should be wired for address 'FF'.

4.2 COMMAND CHARACTER

The command character is defined for system flexibility and future expandability. The command allocations that presently exist are as follows.

1) '00' to '7F' - user defined commands
2) '80' to 'BF' - memory and I/O group
3) 'C0' to 'FF' - supervisory and timer group

4.2.1 The SCU 1 is a factory mask programmed device, so new commands cannot be added to the SCU 1 by the user. The provision for user defined commands allows a user to add SCU 1 - like devices to the network. Any command that is not defined within a particular addressed SCU 1 will result in a "loop" response, 'FB', being issued.

4.3 DATA CHARACTER

4.3.1 The characters DATA 1 and DATA 2 are multiple purpose characters used to designate port or memory addresses, port or memory data, or logical data masks

associated with particular commands. The exact function of the data characters in a particular message is determined by the particular poll command in the message.

4.4 LRC CHARACTER

4.4.1 The LRC character is a vertical error detection character. Each bit of the LRC is an even parity check of the four corresponding bits of the four preceding characters. The LRC may be calculated by the host by performing a logical "exclusive or" function on the four preceding characters. This vertical parity check, combined with the parity of each character, provides a highly error resistant interchange between the SCU 1 and host processor.

Figure 4-1 shows an example of the calculation of the LRC character in a typical message.

FIGURE 4-1 EXAMPLE OF LRC CHARACTER

	HEX	BINARY	PARITY
Address	B2	1011 0010	0
Command	85	1000 0101	1
Data 1	C4	1100 0100	1
Data 2	0A	0000 1010	0
LRC	F9	1111 1001	0

SECTION 5

MESSAGE FORMAT DESCRIPTION

5.1 POLL MESSAGE FORMAT

5.1.1 There are 2 allowable poll message formats for messages transmitted from the host to a particular SCU 1. The 2 poll message formats are designated as (1) normal poll message and (2) short poll message.

5.2 NORMAL POLL MESSAGE

5.2.1 The normal poll message is used for most commands. It is composed of 5 characters which are, in order of transmission, (1) Address, (2) Command, (3) Data 1, (4) Data 2, and (5) LRC.

5.3 SHORT POLL MESSAGE

5.3.1 The short poll message may be used under certain situations in order to reduce the number of characters that are transmitted in a poll and/or response message. The short poll may be used in conjunction with the port monitor functions, but it may be used only after the SCU 1 has been commanded to respond to a short poll inquiry.

5.3.2 The short poll message is composed of 2 characters which are, in order of transmission, (1) address and (2) command. Since no LRC character is included, only the even parity bit of each character is available to provide data integrity.

5.4 RESPONSE MESSAGE FORMAT

5.4.1 There are 3 allowable response message formats for messages transmitted from the SCU 1 to the host. The 3 response message formats are designated as (1) normal response message, (2) short response message, and (3) no response.

5.5 NORMAL RESPONSE MESSAGE

5.5.1 The normal response message is the response for most host poll messages. It is composed of 5 characters which are, in order of transmission, (1) repeat address, (2) repeat command/"loop", (3) data 1, (4) data 2, and (5) LRC.

5.5.2 The command word will be repeated unless an invalid command or an invalid port address is received in the poll message. In this case, the "loop" command, 'FB', will be issued as the response.

5.5.3 The data 1 and data 2 characters will reflect the appropriate response as determined by the command character in the host poll message. The LRC is again generated according to the 4 preceding characters.

5.6 SHORT RESPONSE MESSAGE

5.6.1 The short response message will be issued only in reply to a short poll message when no activity has occurred since the previous poll. If an activity has occurred, the SCU 1 will respond to the short poll message with a normal response message.

5.6.2 The short response message is composed of 2 characters which are, in order of transmission, (1) address and (2) command.

5.7 NO RESPONSE

5.7.1 It is possible for the SCU 1 to echo no response. This condition will occur after a system synchronization command or if the SCU 1 detects a parity or LRC error. Thus, when expecting a reply, the host should always perform a check for no response after a short time delay, and if this is the case, resynchronize and retransmit the poll.

SECTION 6

SYSTEM SYNCHRONIZATION

6.1 SYSTEM SYNCHRONIZATION

6.1.1 In order that each SCU 1 in a network determine which of the received serial bytes are address, command, data and LRC, a means of synchronization is required. A special address and command is reserved for this function. A message of four bytes of 'FF' and the LRC character '00' perform the function.

6.1.2 After an SCU 1 powers up, it will be in the offline condition. While offline, the SCU 1 will not respond to any command but will wait for a system synchronization message. Once synchronized, the SCU 1 will be online. Sending a synchronization message will not interfere with any SCU 1 already online or reset any task in progress, such as a monitor port task.

6.1.3 An SCU 1 will remain online until a parity or LRC error is detected. This is done to prevent an SCU 1 from responding to what may be a false address, which otherwise could cause a bus contention problem on the serial link. An SCU 1 may also go offline if the power supply (Vcc) falls to ground temporarily or if the external reset pin is taken low temporarily.

6.1.4 If an SCU 1 is commanded to perform a task such as monitor port 1 and it then goes offline, the task will still be performed; however, system synchronization must occur before the SCU 1 can respond to a poll command.

SECTION 7

MODEM SIGNALS

7.1 MODEM SIGNALS

7.1.1 Handshaking with line drivers/receivers, modems or other transceivers may be accomplished with the modem signals, RTS (Request-To-Send) and CTS (Clear-To-Send). These signals allow interfacing the SCU 1 to half-duplex serial links with slow turnaround times. Just prior to responding to a valid command, RTS will go to a logic 1, indicating the SCU 1 is ready to send data back to the host. For half-duplex transmission lines, this signal should be used to direct the flow of information from the SCU 1 to the host (i.e., "turning the line around").

7.1.2 CTS is an input tested by the SCU 1 after RTS goes active, to determine when the SCU 1 may begin sending data. This signal should be an acknowledgement that the serial link is ready for data from the SCU 1. CTS must go active within approximately 2.3 seconds following RTS going active. Otherwise, the message will be cancelled, the task will be aborted and the SCU 1 will go offline, requiring resynchronization.

7.1.3 For systems that use neither handshaking signal (full-duplex systems) or only use RTS, CTS should be directly tied to RTS. When so configured, the serial link will have approximately 128 microseconds to turn around following RTS going active.

SECTION 8

SCU 1 CLOCKS

8.1 SCU 1 CLOCKS

8.1.1 The suggested time base for the SCU 1 is a 3.6864 MHz crystal. This frequency was chosen to provide proper serial timing. Any frequency from 2 MHz to 4 MHz may be used, however, the serial Baud rates will be proportionally affected. The following crystal parameters are recommended for the SCU 1:

a) Parallel resonance, Fundamental Mode At-Cut
b) Shunt capacitance (CO) = 7pF maximum
c) Series resistance (Rs) = See table
c) Holder = See table

FREQUENCY	Rs	HOLDER
2-2.7 MHz	300 Ohms max	HC-6 HC-33
2.8-4 MHz	150 Ohms max	HC-6 HC-18* HC-25* HC-33

*This holder may not be available at frequencies near the lower end of this range.

SECTION 9

SCU 1 RESET

9.1 POWER ON RESET

9.1.1 The SCU 1 contains Power-On-Reset circuitry to automatically reset the device following a typical power-up situation, thus saving external reset circuitry in many applications. The internal Power-on-reset is designed to keep the SCU 1 in a reset condition until the proper level of Vcc min is achieved and until the crystal oscillator and internal clock circuitry is operational.

9.2 EXTERNAL RESET

9.2.1 In some applications it may be desirable to provide external reset circuitry to accommodate particular power-on conditions or to provide operator reset capability, such as with a $\overline{\text{RESET}}$ push button. Figure 9-1 shows a possible external reset circuit that may be used to control the power-on reset and/or provide operator reset capability. Figure 9-2 shows the desired operation of the SCU 1 $\overline{\text{RESET}}$ input.

User Ports 0 and 1 on the SCU 1 are set to a logic "1" condition at the time when $\overline{\text{RESET}}$ is pulled low. The state of AS7-AS0, SI, SO, RTS, CTS, S0, S1, and EXT.STATUS is unchanged or undefined when $\overline{\text{RESET}}$ is held low. These signals are I/O lines on the 3870 which are not initialized at the time of a $\overline{\text{RESET}}$. Once $\overline{\text{RESET}}$ is released and allowed to go high the SCU 1 firmware initializes these remaining signals to their proper state so that their intended function can be performed.

FIGURE 9-1 EXTERNAL RESET CIRCUITRY

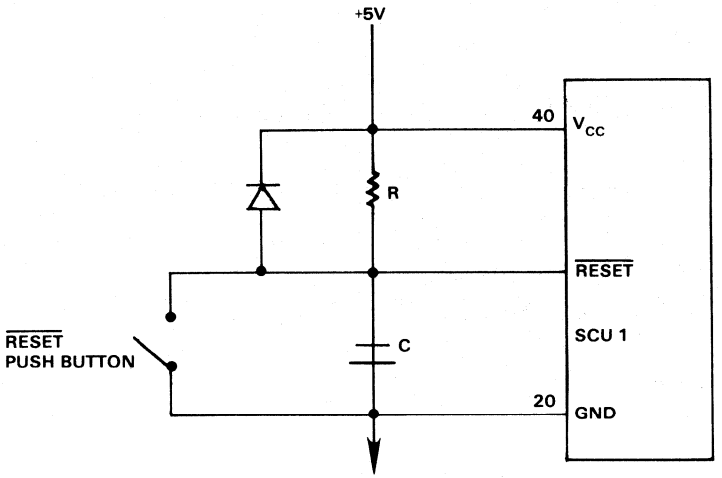
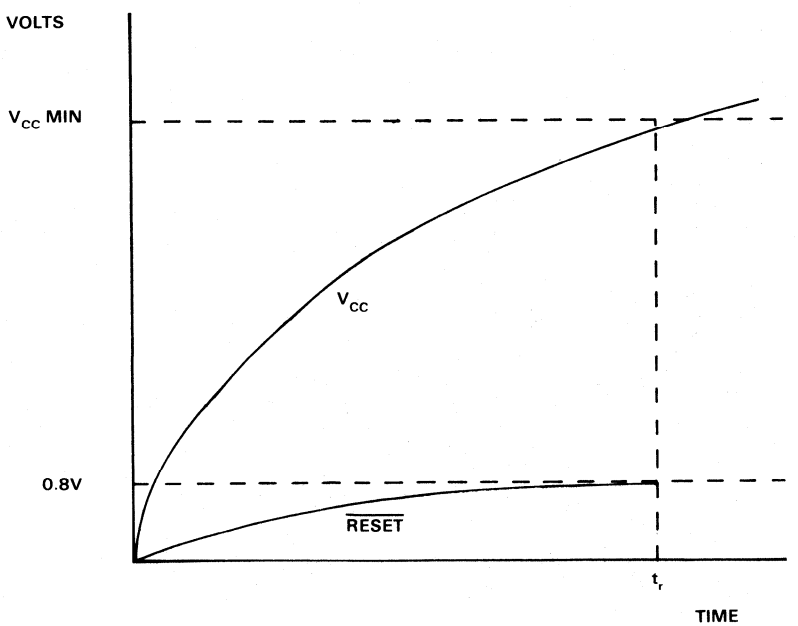


FIGURE 9-2 RESET CIRCUIT OPERATION



t_r = 50 msec typical

SECTION 10

Vcc DECOUPLING

10.1 Vcc DECOUPLING

10.1.1 It is recommended that a good high-frequency decoupling capacitor be used to suppress noise on the Vcc line. A 0.01uF or 0.1uF ceramic capacitor should be placed between Vcc and Ground and physically close to the SCU 1.

SECTION 11

COMMAND/RESPONSE DEFINITION

11.1 COMMAND/RESPONSE DEFINITION

11.1.1 A summary of the command (poll) messages generated by the host and of the response messages generated by the SCU 1 is provided in Tables 11-1, 11-2, and 11-3. A detailed description of each command/response is provided below.

11.2 I/O GROUP

11.2.1 '80' Single Byte Output to Port

Command '80' outputs a byte specified in the Data Field to a selected port. For the SCU 1 only ports 0 and 1 are valid, corresponding to D00-D07 and D10-D17 respectively. The data is latched in the output port. The port is then read and the contents returned in the Data Register as part of the SCU response. If an invalid port is selected, the SCU will respond with an 'FB' Command ("Loop" Command).

Host: Address, '80', Port Select, Verified Data, LRC
SCU 1 Response
Correct: Address, '80', Port Select, Verified Data, LRC
Invalid Port: Address, 'FB', Port Select, Output Data, LRC

11.2.2 '81' Input Byte from Port

Command '81' inputs data from either Port 0 or Port 1. Additionally, a logical AND Mask can be specified to allow selectively masking data input bits at the SCU 1. If no mask is desired, the bit should be a logic one; if a mask is desired, the bit should be a logic zero. Therefore, if no mask is desired, the AND mask field should contain 'FF'.

If an incorrect port is selected, the SCU will respond with an 'FB' Command.

Host: Address, '81', Port Select, AND Mask, LRC
Response
Correct: Address, '81', Port Select, Input Data, LRC
Invalid Port: Address, 'FB', Port Select, AND Mask, LRC

11.2.3 '82' Set Bit(s) in Port

Command '82' sets the individual output bits in the selected port. The OR mask field is logically OR'ed with the present data in the selected port. Only Ports 0 or 1 are valid. If an invalid port is selected, the SCU 1 will respond with the 'FB' Command.

Host: Address, '82', Port Select, OR Mask, LRC
Response
Correct: Address, '82', Port Selected, Verified Output, LRC
Invalid Port: Address, 'FB', Port Selected, OR Mask, LRC

11.2.4 '83' Clear Bit(s) in Port

Command '83' clears the individual output bits in the selected port. The AND mask field is logically AND'ed with the present data in the selected port in order to clear individually selected bits. Only Ports 0 or 1 are valid. If an invalid port is selected, the SCU 1 will respond with the 'FB' Command.

Host: Address, '83', Port Selected, AND Mask, LRC
Response
Correct: Address, '83', Port Selected, Verified Output, LRC
Invalid Port: Address, 'FB', Port Selected, AND Mask, LRC

11.2.5 '84' Toggle Bit(s) in Port

Command '84' toggles the individual output bits in the selected ports. The XOR Mask is Exclusive OR'ed with the present data in the selected port in order to toggle individually selected bits. Only Ports 0 or 1 are valid. If an invalid

port is selected, the SCU 1 will respond with the 'FB' Command.

Host: Address, '84', Port Select, XOR Mask, LRC
Response
Correct: Address, '84', Port Select, Verified Output, LRC
Invalid Port: Address, 'FB', Port Select, XOR Mask, LRC

11.2.6 '85' Output 2 Bytes

Command '85' outputs 2 bytes to the SCU ports. The Port 1 byte is output first then the Port 0 byte. The port is read and the contents returned in the Data Address and Data Fields in the SCU 1 response. This Command is useful in controlling multiple D/A converters.

Host: Address, '85', Port 1 Byte, Port 0 Byte LRC
Response
Address, '85', Port 1 Byte, Port 0 Byte, LRC

11.2.7 '86' Input 2 Bytes

Command '86' inputs data from Port 0 and Port 1. An AND Mask is provided for both ports that allow selectively masking data input bits at the SCU 1. If no mask is desired, both mask values should be all ones.

Host: Address, '86', Port 1 AND Mask, Port 0 AND Mask, LRC
Response
Address: '86', Port 1 Input, Port 0 Input, LRC

11.3 MONITOR AND A/D GROUP

11.3.1 '87' A/D Conversion

Command '87' configures the SCU 1 to work with a 16 channel 12-bit Analog to Digital Converter or 3-1/2 digit panel meters with BCD outputs. This is possible through the use of D00-D07, D10-D17, STROBE and EXT. IN. If an A/D is used the

upper 4 bits of Port 1 (D17-D14) specifies the selected channel. Bits D13-D10 must be loaded with all ones ('F'). The Data Field must contain all ones ('FF') in order to condition Port 0 to accept the A/D transfer.

When the SCU 1 accepts the A/D Command, the upper four bits of Port 1 specifying the multiplex channel are output and the $\overline{\text{STROBE}}$ is issued.

The EXT. IN pin is then monitored for a DONE flag (logic 1 equals done). When the DONE flag is received the SCU 1 will respond back to the host.

Host: Address, '87', MPX Channel, 'FF', LRC
Response
Address, '87', MPX Channel and A/D Data, A/D Data, LRC

If a digital panel meter input is required, the upper four bits of Port 1 used to specify the Multiplex Channel should be made all ones ('F'). Therefore, the $\overline{\text{STROBE}}$ and EXT. IN signals are used for handshaking lines with the DPM. Up to 3-1/2 digits of BCD data can be returned in the Data Address and Data Fields.

11.3.2 '88' Monitor Port 0 (Equality)

Command '88' configures SCU 1 Port 0 to monitor the input port pins for a desired pattern. Only when this pattern occurs (equality) will the SCU 1 notify the host that the monitored condition has occurred. The SCU 1 can only respond when the Poll Command (long or short) is executed. Port 1 can be configured to perform other activities while Port 0 is in the Monitor mode.

The Data Address field contains an AND mask. The mask should have a one or zero in each bit position in order to enable or disable each respective input pin. The desired pattern is selected in the data field. The SCU 1 is continually searching for a match between the masked Port 0 inputs and the desired pattern. When the equality condition is met, it is latched into an internal SCU 1 register which can only be reset by the 'FC' Supervisory Command or reinitialization of the Port.

SCU 1 Port Initialization

```
Host:  Address, '88', AND Mask, Desired Pattern, LRC
Response
Address, '88', AND Mask, Desired Pattern, LRC
Polling Sequence
Host:  Address, 'FE'                                  (short poll)
Host:  Address, 'FD', XX, XX, LRC                     (long poll)
```

SCU 1 Response

```
No Activity:  Address, 'FE'                           (short poll)
No Activity:  Address, 'FD', XX, XX, LRC              (long poll)
Activity:     Address, '88', AND Mask, Pattern, LRC   (long and
                                                       short poll)
```


11.3.3 '89' Monitor Port 1 (Equality)

Command '89' configures SCU 1 Port 1 to monitor the input port pins for a desired pattern. Only when this pattern occurs will the SCU notify the host that the monitored condition has occurred. The SCU 1 can only respond when the Poll Command (Long or Short) is executed. Port 0 can be configured to perform other activities while Port 1 is in the Monitor mode.

The Data Address field contains an AND Mask. The Mask should have a one or zero in each bit position in order to enable or disable each respective input pin. The desired pattern is selected in the Data field. The SCU is continually searching for a match between the masked Port 1 inputs and the desired pattern. When the equality condition is met, it is latched into an internal SCU 1 register which can only be reset by the 'FC' Supervisory Command or reinitialization of the port.

SCU 1 Port Initialization

```
Host:  Address, '89', AND Mask, Desired Pattern, LRC
Response
Address, '89', AND Mask, Desired Pattern, LRC
Polling Sequence
```

```
Host:  Address, 'FE'                                    (short poll)
Host:  Address, 'FD', XX, XX, LRC                       (long poll)
```

SCU 1 Response

```
No Activity:  Address, 'FE'                             (short poll)
No Activity:  Address, 'FD', XX, XX, LRC                (long poll)
Activity:     Address, '89', AND Mask, Pattern, LRC     (long or
                                                         short poll)
```

11.3.4 '8A' Monitor Port 0 (Inequality)

Comand '8A' configures SCU 1 Port 0 to monitor the input port pins for a desired pattern. Only when this pattern does not occur (inequality) will the SCU 1 notify the host that the monitored condition has been sustained. The SCU 1 can only respond when the Poll Command (long or short) is executed. Port 1 can be configured to perform other activities while Port 0 is in the Monitor mode.

The Data Address field contains an AND Mask. The mask should have a one or a zero in each bit position in order to enable or disable each respective input pin. The desired pattern is selected in the Data field. The SCU 1 continually searches for an inequality between the masked Port 0 inputs and the desired pattern. When the inequality condition is met, it is latched into an internal SCU 1 register which can only be reset by the 'FC' Supervisory Command or reinitialization of the port.

SCU 1 Port Initialization

```
Host:  Address, '8A', AND Mask, Desired Pattern, LRC
Response
Address:  '8A', AND Mask, Desired Pattern, LRC
Polling Sequence
Host:  Address, 'FE'                                    (short poll)
Host:  Address, 'FD', XX, XX, LRC                       (long poll)
```

SCU 1 Response

No Activity:	Address, 'FE'	(short poll)
No Activity:	Address, 'FD', XX, XX, LRC	(long poll)
Activity:	Address, '8A', AND Mask, Pattern, LRC	(long or short poll)

11.3.5 '8B' Monitor Port 1 (Inequality)

Command '8B' configures SCU 1 Port 1 to monitor the input port pins for a desired pattern. Only when the pattern does not occur (inequality) will the SCU 1 notify the host that the monitored condition has been sustained. The SCU 1 can only respond when the Poll Command (long or short) is executed. Port 0 can be configured to perform other activities while Port 1 is in the Monitor mode.

The Data Address field contains an AND Mask. The mask should have a one or zero in each bit position in order to enable or disable each respective input pin. The desired pattern is selected in the Data field. The SCU 1 continually searches for an inequality between the masked Port 1 inputs and the desired pattern. When the inequality condition is met, it is latched into an internal SCU 1 register which can only be reset by the 'FC' Supervisory Command or reinitialization of the Port.

SCU 1 Port Initialization

Host: Address, '8B', AND Mask, Desired Pattern, LRC
Response
Address, '8B', AND Mask, Desired Pattern, LRC
Polling Sequence

Host:	Address, 'FE'	(short poll)
Host:	Address, 'FD', XX, XX, LRC	(long poll)

SCU Response

No Activity:	Address, 'FE'	(short poll)
No Activity:	Address, 'FD', XX, XX, LRC	(long poll)
Activity:	Address, '8B', AND Mask Pattern, LRC	(long or short poll)

11.4 SUPERVISORY GROUP

11.4.1 'F9' Short Poll Disable

Command 'F9', disables the circuitry that allows the SCU to recognize a Short Poll. The Data Address and Data field bits are designated as "Don't Care" since they are not used or interpreted for any function.

Host: Address, 'F9', Don't Care, Don't Care, LRC
Response
Address, 'F9', Don't Care, Don't Care, LRC

11.4.2 'FA' Short Poll Enable

Command 'FA' sets the Short Poll Enable circuitry within the SCU 1. This Command is necessary to enable the Short Poll feature whenever an SCU 1 is first initialized or reset. The Data Address and Data field bits are designated as "Don't Care" since they are not used or interpreted for any function.

Host:
Address, 'FA', Don't Care, Don't Care, LRC
Response
Address, 'FA', Don't Care, LRC

11.4.3 FB_H Loop

Command 'FB' has no effect on the SCU 1 other than echoing the message. The Data Address and Data field bits are designated as "Don't Care" since they are not used or interpreted for any function.

The 'FB' Command is also a valid response from an SCU 1 if an incorrect Command is issued or an invalid port is addressed.

Host: Address, 'FB', Don't Care, Don't Care, LRC
Response
Address, 'FB', Don't Care, Don't Care, LRC

11.4.4 'FC' Reset Report/Task

Command 'FC' resets a Report/Task of a Monitor Command. This Command must be issued in order to inhibit the activity response from a Monitor Command and to permit continued monitoring by that SCU 1. If this Command is not issued each time a previously activated SCU 1 is polled (assuming no reinitialization), it will respond with an activity response.

The Data Address and Data field bits are designated as "Don't Care" since they are not used or interpreted for any function.

Host: Address, 'FC', Don't Care, Don't Care, LRC
Response
Address, 'FC', Don't Care, Don't Care, LRC
Description:

11.4.5 'FD' Long Poll

Command 'FD' is used to check the status of Monitor Commands in individual SCU 1's. If no activity has occurred in an SCU 1 when polled, the message will simply be echoed. If an activity has occurred, a new message will be returned to the host. The Data Address and Data field bits are designated as "Don't Care" since they are not used or interpreted for any function.

The message response indicating SCU 1 activity has 3 variable parts. The YY and ZZ data in the Data Address and Data fields are the appropriate responses as defined by the programmed Monitor Command. Only the '88', '89', '8A', and '8B' Commands will generate an activity type response.

The Long Poll Command is valid when the Short Poll Command is enabled.

Host: Address, 'FD', Don't Care, Don't Care, LRC
Response
No Activity: Address, 'FD', Don't Care, Don't Care, LRC
Activity: Address, XX, YY, ZZ, LRC

11.4.6 FE_H Short Poll

Command 'FE' is used to check the status of Monitor Commands in individual SCU 1's. The format is designed to optimize polling speed in a system. If no activity has occurred in an SCU 1 when polled, the Command will simply be echoed. If an activity has occurred, a full message will be returned to the host.

The message response indicating SCU 1 activity has 3 variable parts. The XX in the Command field indicates which command the SCU was programmed to monitor. The YY and ZZ data in the Data Address and Data fields are the appropriate responses as defined in the programmed Monitor Command. Only the '88', '89', '8A', and '8B' Commands will generate an activity type response.

The Short Poll Command can only work if the SCU has its short poll enable flag set. This flag is set and cleared by the 'FA' and 'F9' Commands respectfully. The SCU 1 is initialized and reset with the Short Poll disabled. If an SCU 1 is addressed with the Short Poll Command while it is not enabled, the unit will lose message synchronization.

Host: Address, 'FE'
Response
No Activity: Address, 'FE'
Activity: Address, XX, YY, ZZ, LRC

11.4.7 FF_H Synchronization

Command 'FF' is used to provide synchronization for the network and all SCU's. Upon initialization or whenever the SCU 1's lose synchronization, this sequence is issued in order to achieve re-synchronization. This command is only valid when used in conjunction with 'FF' in the Address, Data Address and Data fields with the LRC being '00'.

Host: 'FF', 'FF', 'FF', 'FF', '00'
Response
All SCU 1's: No Response

TABLE 11-1 I/O GROUP COMMAND/RESPONSE SUMMARY

COMMAND DESCRIPTION				CORRECT RESPONSE				ERROR RESPONSE	
HEX COMMAND	DESCRIPTION	DATA 1	DATA 2	HEX COMMAND	DATA 1	DATA 2		DATA 1	DATA 2
00-79	USER DEFINED								
80	Byte Output to Port	Port Select	Data Out	80	Port Selected	Verified Data	FB	Port Selected	Data Out
81	Byte Input From Port	Port Select	Or Mask	81	Port Selected	Verified Data	FB	Port Selected	Or Mask
82	Set Bit in Port	Port Select	Or Mask	82	Port Selected	Verified Output	FB	Port Selected	Or Mask
83	Clear Bit In Port	Port Select	And Mask	83	Port Selected	Verified Output	FB	Port Selected	And Mask
84	Toggle Bit(s) in Port	Port Select	XOR Mask	84	Port Selected	Verified Output	FB	Port Selected	XOR Mask
85	Output Two Bytes	Port 1 Data	Port 0 Data	85	Port 1 Data	Port 0 Data	FB	Port 0 Data	Port 2 Data
86	Input Two Bytes	Port 1 And Mask	Port 0 And Mask	86	Port 1 Input	Port 0 Input	FB	Port 1 And Mask	Port 0 And Mask

TABLE 11-2 MONITOR AND A/D GROUP COMMAND/RESPONSE SUMMARY

COMMAND DESCRIPTION				CORRECT RESPONSE			ERROR RESPONSE	
HEX COMMAND	DESCRIPTION	DATA 1	DATA 2	HEX COMMAND	DATA 1	DATA 2	DATA 1	DATA 2
87	12 Bit A/D Conversion	MPX Channel	FF	87	MPX & Data	Data	MPX	FF
88	Monitor Port 0 (=) equality	And Mask	Pattern	Responds on Poll:			N/A	N/A
89	Monitor Port 1 (=) equality	And Mask	Pattern	Responds on Poll:			N/A	N/A
8A	Monitor Port 0 (≠) inequality	And Mask	Pattern	Responds on Poll:			N/A	N/A
8B	Monitor Port 1 (≠) inequality	And Mask	Pattern	Responds on Poll:			N/A	N/A
8C FB	Reserved for Future Mostek Commands							

N/A Not Applicable

TABLE 11-3 SUPERVISORY GROUP COMMAND/RESPONSE SUMMARY

				ACTIVE RESPONSE			INACTIVE RESPONSE		
HEX COMMAND	DESCRIPTION	DATA 1	DATA 2	HEX COMMAND	DATA 1	DATA 2	HEX COMMAND	DATA 1	DATA 2
F9	Short Poll Disable	XX	XX	F9	XX	XX	N/A	N/A	N/A
FA	Short Poll Enable	XX	XX	FA	XX	XX	N/A	N/A	N/A
FB	Loop (Echo)	XX	XX	FB	XX	XX	N/A	N/A	N/A
FC	Reset Report/Task	XX	XX	FC	XX	XX	N/A	N/A	N/A
FD	Long Poll	XX	XX	FD	Mask	Pattern	FD	XX	XX
FE	Short Poll	None	None	FE	Mask	Pattern	FE	None	None
FF	Initialization	FF	FF	← No Response →					

N/A = Not Applicable
X = Don't Care

3870/F8 MICROCOMPUTER DATA BOOK

NOT FOR NEW DESIGN

MOSTEK®

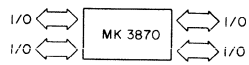
F8 MICROCOMPUTER DEVICES

F8 Central Processing Unit MK 3850

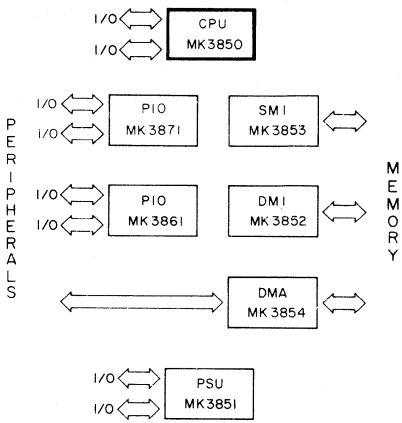
FEATURES

- ☐ N-channel Isoplanar MOS technology
- ☐ 2 μs cycle time
- ☐ 64 byte RAM on the CPU chip
- ☐ Two bi-directional, 8-bit I/O ports
- ☐ 8-bit arithmetic and logic unit, supporting both binary and decimal arithmetic
- ☐ Interrupt control logic
- ☐ Both external and crystal clock generating modes
- ☐ Over 70 instructions
- ☐ Low power dissipation—typically less than 330mW

SINGLE CHIP MK3870



F8 FAMILY

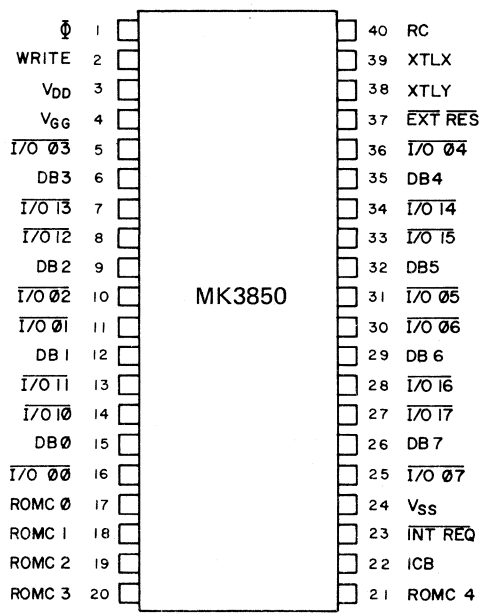


GENERAL DESCRIPTION

The MK3850 is the Central Processing Unit (CPU) for the F8 Microprocessor family. It is used in conjunction with other F8 family devices to configure the optimal microprocessor system for the amount of RAM, ROM/PROM, and I/O required in the users application. A minimum system may be configured with as few as two devices (CPU & PSU), while larger systems may have up to 64K bytes of memory, 128 I/O ports, direct memory acccess, and even multiple processors. Single chip microcomputer systems are also possible using the MK3870

PIN NAME	DESCRIPTION	TYPE
DB0-DB7	Data Bus Lines	Bi-directional (3-State)
Φ, WRITE	Clock LInes	Output
I/O 00-I/O 07	I/O Port Zero	Input/Output
I/O 10-I/O 17	I/O Port One	Input/Output
RC	RC Network Pin	Input
ROMC0-ROMC4	Control Lines	Output
$\overline{\text{EXT RES}}$	External Reset	Input
$\overline{\text{INT REQ}}$	Interrupt Request	Input
ICB	Interrupt Control Bit	Output
XTLX	Crystal Clock Line	Output
XTLY	External Clock Line	Input
V_{SS}, V_{DD}, V_{GG}	Power Lines	Input

PIN CONNECTIONS



NOT FOR NEW DESIGN

MOSTEK®

F8 MICROPROCESSOR DEVICES

Program Storage Unit MK3851

FEATURES

- ☐ 1024 x 8 ROM storage
- ☐ Two 8-bit I/O Ports
- ☐ Programmable timer
- ☐ External/timer interrupt circuitry
- ☐ Low power dissipation < 275mW typical

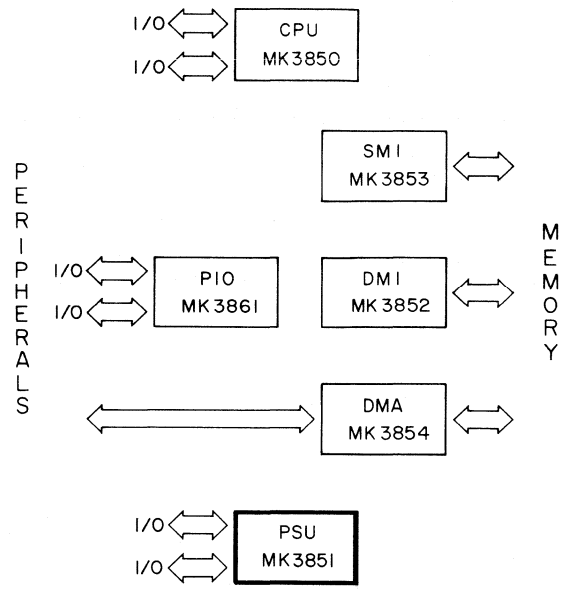
GENERAL DESCRIPTION

The MK 3851 program storage unit (PSU) provides 1024 bytes of read only memory (ROM) for the F8 system. Additionally each PSU provides two 8-bit I/O ports, a programmable timer and vectored timer and external interrupts. The PSU contains three 16-bit address registers and a 16-bit incrementer/adder. On command from the F8 CPU the MK 3851 accesses its internal memory using one of these three registers and increments or adds displacement to the register if required.

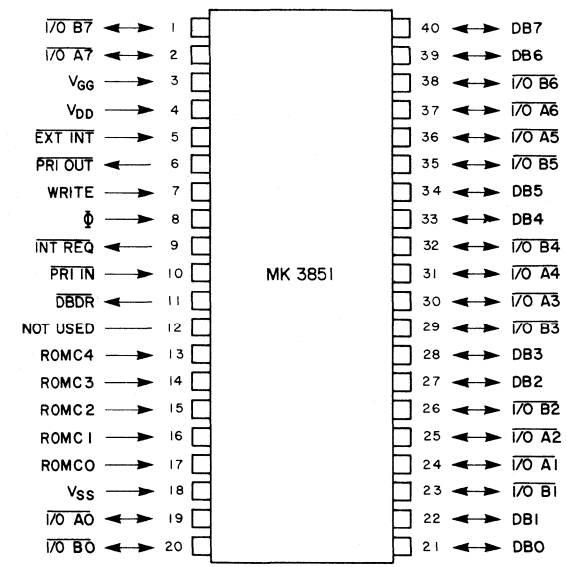
The MK 3851 PSU is manufactured using N-channel Isoplanar MOS technology. Power dissipation is very low, typically less than 275mW.

PIN NAME	DESCRIPTION	TYPE
$\overline{\text{I/O A0-I/O A7}}$	I/O Port A	Bi-directional
$\overline{\text{I/O B0-I/O B7}}$	I/O Port B	Bi-directional
DB0-DB7	Data Bus	Bi-directional, tri-state
ROMC0-ROMC4	Control Lines	Input
Φ, WRITE	Clock Lines	Input
$\overline{\text{EXT INT}}$	External Interrupt	Input
$\overline{\text{PRI IN}}$	Priority In	Input
$\overline{\text{PRI OUT}}$	Priority Out	Output
$\overline{\text{INT REQ}}$	Interrupt Request	Output
$\overline{\text{DBDR}}$	Data Bus Drive	Output
V_{SS}, V_{DD}, V_{GG}	Power Supply Lines	Input

F8 FAMILY



PIN CONNECTIONS



IV F8 MICROCOMPUTER FAMILY

NOT FOR NEW DESIGN

MOSTEK®

F8 MICROCOMPUTER DEVICES

Dynamic Memory Interface MK 3852

FEATURES

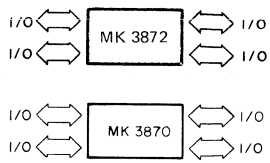
- ☐ Provides interface for 64K of dyanmic or static RAM
- ☐ Interfaces with MK3854 for DMA channel
- ☐ Provides automatic refresh for dynamic RAMs.
- ☐ Low Power Dissipation Typically Less Than 335mW

GENERAL DESCRIPTION

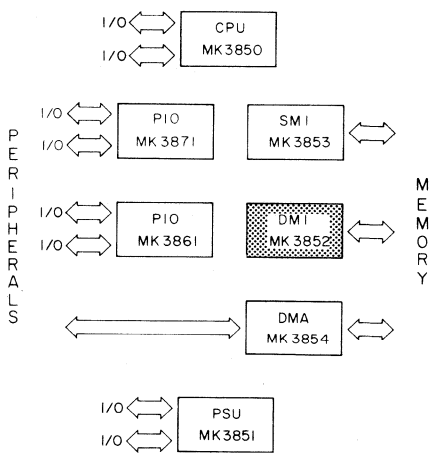
The 3852 DMI provides all interface logic needed to include up to 64K bytes of dynamic or static RAM memory in an F8 microcomputer system. In response to control signals output by the 3850 CPU, the 3852 DMI generates address and control signals needed by standard static and dynamic RAM devices. The MK3852 DMI is manufactured using N-channel Isoplanar MOS technology.

PIN NAME	DESCRIPTION	TYPE
DB0-DB7	Data Bus Lines	Bi-directional(3-State)
ADDR0-ADDR15	Address Lines	Output (3-State)
Φ, WRITE	Clock LInes	Input
MEMIDLE	DMA Timing Line	Output
CYCLE REQ	RAM Timing Line	Output
CPU Slot	Timing Line	Input/Output
CPU READ	RAM Timing Line	Output
REGDR	Register Drive Line	Input/Output
RAM WRITE	Write Line	Output (3-State)
ROMC0-ROMC4	Control Lines	Input
V_{SS}, V_{DD}, V_{GG}	Power Lines	Input

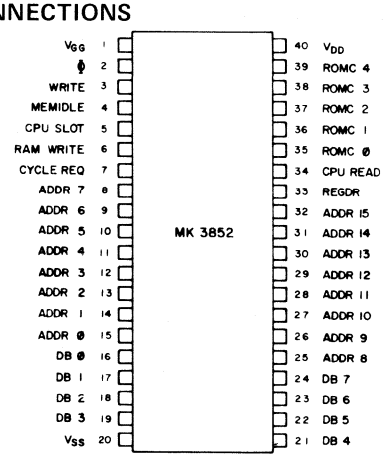
SINGLE CHIP 3870 MICROCOMPUTER FAMILY



F8 FAMILY



PIN CONNECTIONS



IV F8 MICROCOMPUTER FAMILY

NOT FOR NEW DESIGN

MOSTEK®

F8 MICROCOMPUTER DEVICES

Static Memory Interface MK 3853

FEATURES

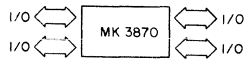
- ☐ Static Memory Interface to RAM,ROM or PROM
- ☐ Programmable Timer
- ☐ Programmable Interrupt Vectors for Timer and External Interrupts
- ☐ Low Power Dissipation Typically Less Than 335 mw

GENERAL DESCRIPTION

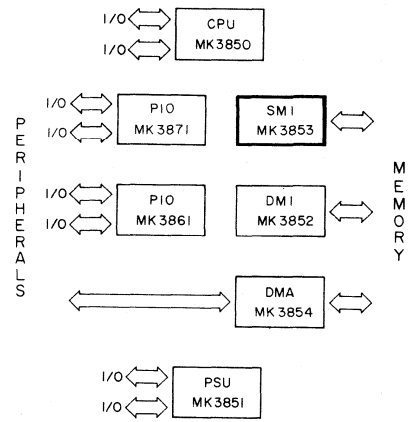
The MK 3853 Static Memory Interface (SMI) provides all necessary address lines and control signals to interface up to 65,536 bytes of Static RAM, ROM or PROM to an F8 microcomputer system. When quantities do not justify the mask charges for the MK 3851 PSU, or a fast turn around is of high importance, the MK 3853 SMI can be used to interface the F8 to EPROM or fusible-link bipolar PROMs. The 3853 SMI along with standard PROM can emulate the memory function of the 3851 PSU, while the 3861 provides the I/O ports, interrupt and timer features of the 3851 PSU. The 3853 is a high performance MOS/LSI circuit using N-channel Isoplanar technology.

PIN NAME	DESCRIPTION	TYPE
DB0-DB7	Data Bus Lines	Bi-directional, tri-state
ADDR0-ADDR15	Address Lines	Output
Φ,WRITE	Clock Lines	Input
$\overline{\text{INT REQ}}$	Interrupt Request	Output
$\overline{\text{PRI IN}}$	Priority In Line	Input
$\overline{\text{RAM WRITE}}$	Write Line	Output
$\overline{\text{EXT INT}}$	External Interrupt Line	Input
REGDR	Register Drive Line	Input/Output
CPU READ	CPU Read Line	Output
ROMC0-ROMC4	Control Lines	Input
V_{GG}, V_{DD}, V_{SS}	Power Supply Lines	Input

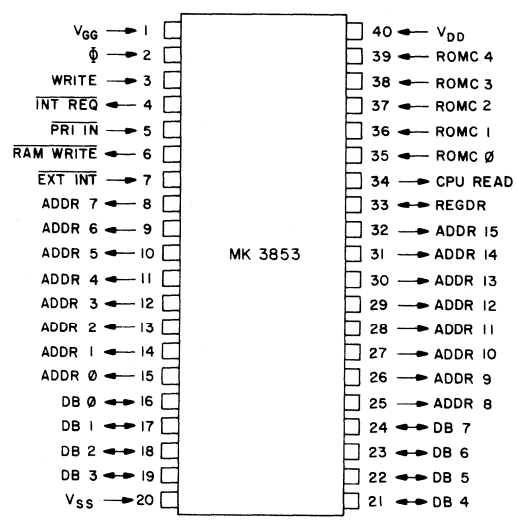
SINGLE CHIP MICROCOMPUTER



F8 FAMILY



PIN CONNECTIONS



IV F8 MICROCOMPUTER FAMILY

NOT FOR NEW DESIGN

MOSTEK®

F8 MICROCOMPUTER DEVICES

F8 Direct Memory Access MK3854

FEATURES

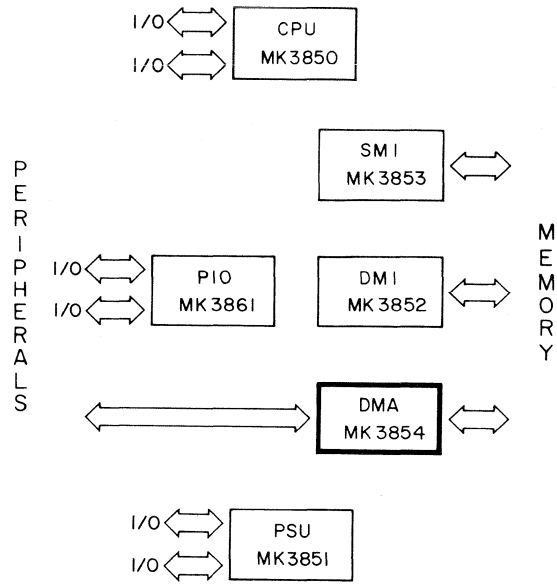
- ☐ 2 μsec cycle time
- ☐ Provides strobe for timing peripherals
- ☐ 16-bit address
- ☐ 12-bit byte count
- ☐ Control registers
- ☐ Port address selection
- ☐ +5V and +12V power supplies
- ☐ Low power dissipation—280mW

GENERAL DESCRIPTION

The MK 3854 Direct Memory Access (DMA) chip facilitates high speed data transfer between the main memory of an F8 system and peripherals. This transfer occurs without suspending normal operation of the processor, allowing DMA with no reduction of program execution speed. The MK 3854 DMA is manufactured using N-channel, Isoplanar MOS technology. Power dissipation is low, typically less than 280mW.

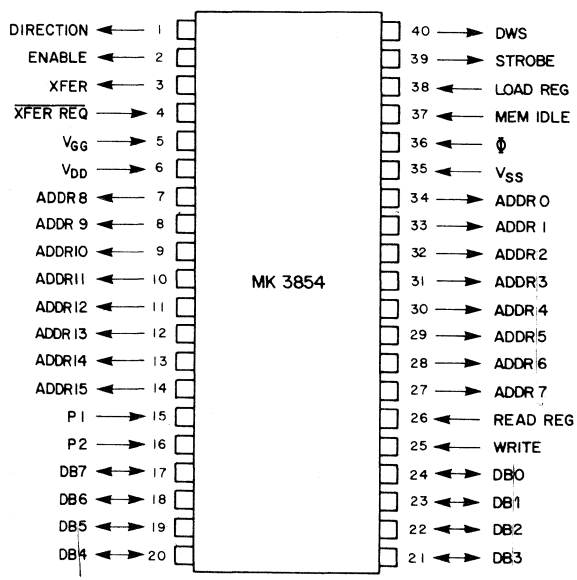
PIN NAME	DESCRIPTION	TYPE
DB0-DB7	Data bus lines	Bidirectional three state
ADDR0-ADDR15	Address lines	Output three state
Φ, WRITE	Clock lines	Input
LOAD REG/ READ REG	Registers load/ read line	Input
P1, P2	Port address select	Input
MEM IDLE	Memory idle line	Input
XFER REQ	Transfer request line	Input
ENABLE, DIRECTION	Control status lines	Output
DWS, XFER	DMA Write slot, transfer	Output
STROBE	Output strobe line	Output
V_{SS}, V_{DD}, V_{GG}	Power lines	Input

F8 FAMILY



IV F8 MICROCOMPUTER FAMILY

PIN CONNECTIONS



NOT FOR NEW DESIGN

MOSTEK®

F8 MICROCOMPUTER DEVICES

Peripheral Input/Output MK3861

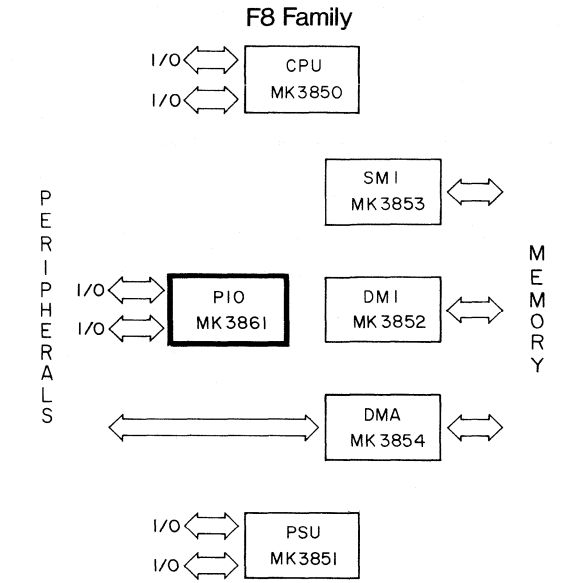
FEATURES

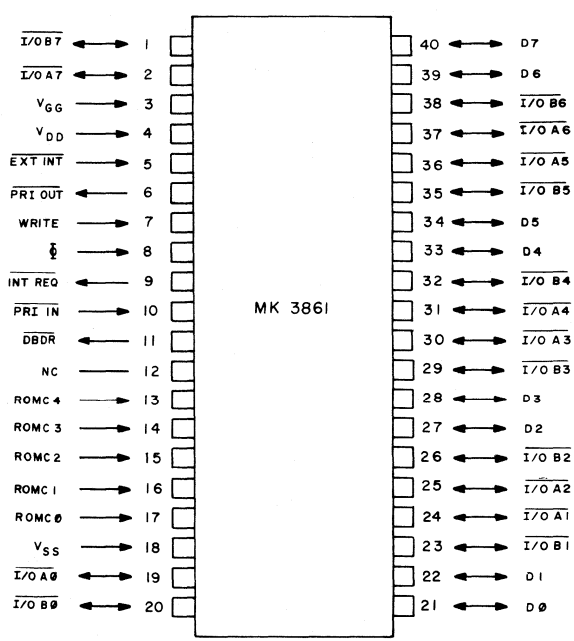
- ☐ Two 8-bit I/O ports
- ☐ Programmable timer
- ☐ External/timer interrupt control circuitry
- ☐ Low power dissipation-typically less than 200mW

GENERAL DESCRIPTION

Each 3861 Peripheral Input/Output Circuit (PIO) provides two 8-bit I/O ports, a programmable timer and a vectored timer or external interrupt for the F8 system. The timer, I/O ports and interrupt circuitry are identical to those of the MK 3851 PSU. The 3861 may be used to provide extra I/O, timer, and interrupt functions compatible with those of the 3851 PSU, or the 3861 may be used as the only I/O peripheral in non PSU systems. This circuit in conjunction with the 3853 and standard PROM is particularly useful in prototyping a PSU system. The 3853 MI circuit along with standard PROM can emulate the memory functions of the PSU while the 3861 provides the I/O, interrupt, and timer features of the PSU. The 3861 is manufactured using the same high performance N-channel Isoplanar technology as the F8 CPU.

PIN NAME	DESCRIPTION	TYPE
D0-D7	Data Bus Lines	Bi-directional, Tri-State
$\overline{\text{I/O A0 - I/O A7}}$	I/O Port A	Bi-directional
$\overline{\text{I/O B0 - I/O B7}}$	I/O Port B	Bi-directional
ROMC0-ROMC4	System Control Lines	Input
ϕ, WRITE	Clock Lines	Input
$\overline{\text{EXT INT}}$	External Interrupt	Input
$\overline{\text{PRI IN}}$	Priority In	Input
$\overline{\text{PRI OUT}}$	Priority Out	Output
$\overline{\text{INT REQ}}$	Interrupt Request	Output
$\overline{\text{DBDR}}$	Data Bus Drive	Output
V_{SS}, V_{DD}, V_{GG}	Power Lines	Input





IV F8 MICROCOMPUTER FAMILY

NOT FOR NEW DESIGN

MOSTEK®

F8 MICROPROCESSOR DEVICES

Peripheral Input/Output MK 3871

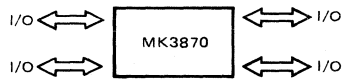
FEATURES

- ☐ Two 8-bit I/O ports
- ☐ Programmable binary timer
- ☐ External/timer interrupt control circuitry
- ☐ Low power dissipation – typically less than 200mW

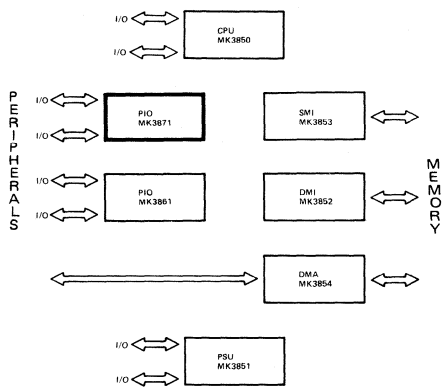
GENERAL DESCRIPTION

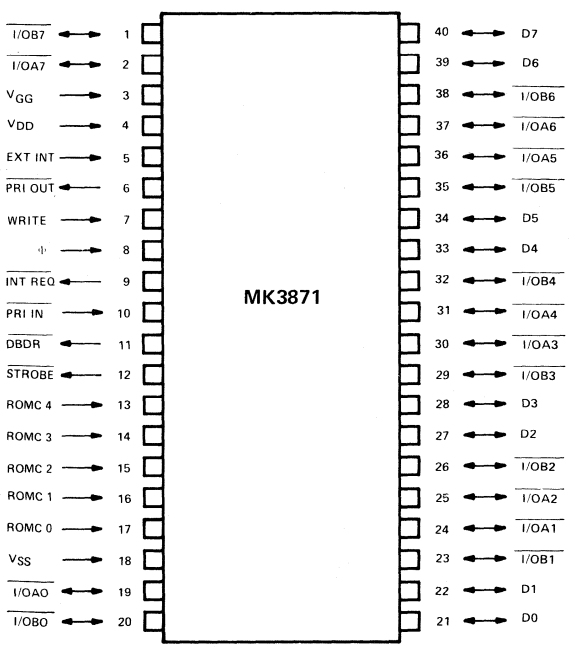
The MK3871 Peripheral Input/Output Circuit (PIO) provides two 8-bit I/O ports and a programmable timer for an F8 multi-chip system (MK3850 family). The MK3871 has the same improved timer and ready strobe output as are on the MK3870 single-chip microcomputer. Thus, for software compatibility with the MK3870, the MK3871 PIO should be used in F8 multi-chip configurations rather than the MK3861 PIO. The MK3871 is manufactured using the same N-channel silicon-gate technology as the single chip MK3870 and the multi-chip F8 family.

SINGLE CHIP MK3870



F8 FAMILY





PIN NAME	DESCRIPTION	TYPE
D0-D7	Data Bus Lines	Bi-Directional, Tri-State
$\overline{\text{I/O A0 - I/O A7}}$	I/O Port A	Bi-Directional
$\overline{\text{I/O B0 - I/O B7}}$	I/O Port B	Bi-Directional
ROMC 0 - ROMC 4	System Control Lines	Input
Φ, WRITE	Clock Lines	Input
EXT INT	External Interrupt	Input
$\overline{\text{PRI IN}}$	Priority In	Input
$\overline{\text{PRI OUT}}$	Priority Out	Output
$\overline{\text{INT REQ}}$	Interrupt Request	Output
$\overline{\text{DBDR}}$	Data Bus Drive	Output
V_{SS}, V_{DD}, V_{GG}	Power Lines	Input
$\overline{\text{STROBE}}$	Ready Strobe	Output

IV F8 MICROCOMPUTER FAMILY

3870/F8 MICROCOMPUTER DATA BOOK

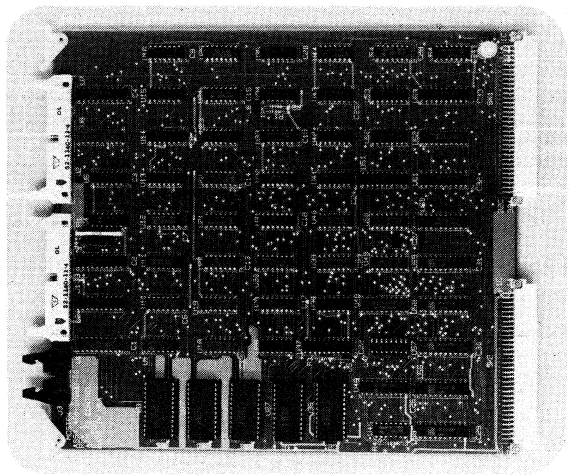
MOSTEK®

MK3870 MICROCOMPUTER SYSTEMS

Application Interface Module (AIM-7XE)

FEATURES

- ☐ Direct interface to Mostek's MATRIX-80/SDS and SYS-80F disc-based development systems
- ☐ In-circuit emulation of 3870 family microprocessors
- ☐ Real-time execution (1-4 MHz and no wait states)
- ☐ Flexible breakpoints (hardware, eight single-byte software), and any number of manually-inserted breakpoints.
- ☐ Single-step execution
- ☐ 4K bytes emulation RAM (expandable to 8K bytes)
- ☐ Option of on-board oscillator or user clock
- ☐ Illegal write-to-memory detection
- ☐ One independently-programmable scope trigger output
- ☐ Forty-eight-channel-by-1024-words history memory.
- ☐ Event counter
- ☐ Delay counter
- ☐ Execution T-state timer
- ☐ Keyboard escape function
- ☐ Simple-to-use single-character commands
- ☐ Flexible display format includes dissassembly of op-codes



GENERAL DESCRIPTION

AIM-7XE is an advanced development tool which provides debug assistance for both software and hardware via in-circuit emulation of the 3870 family microprocessors. Use of the AIM-7XE 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 $\overline{\text{RESET}}$ and $\overline{\text{EXT INT}}$, are functional during emulation.

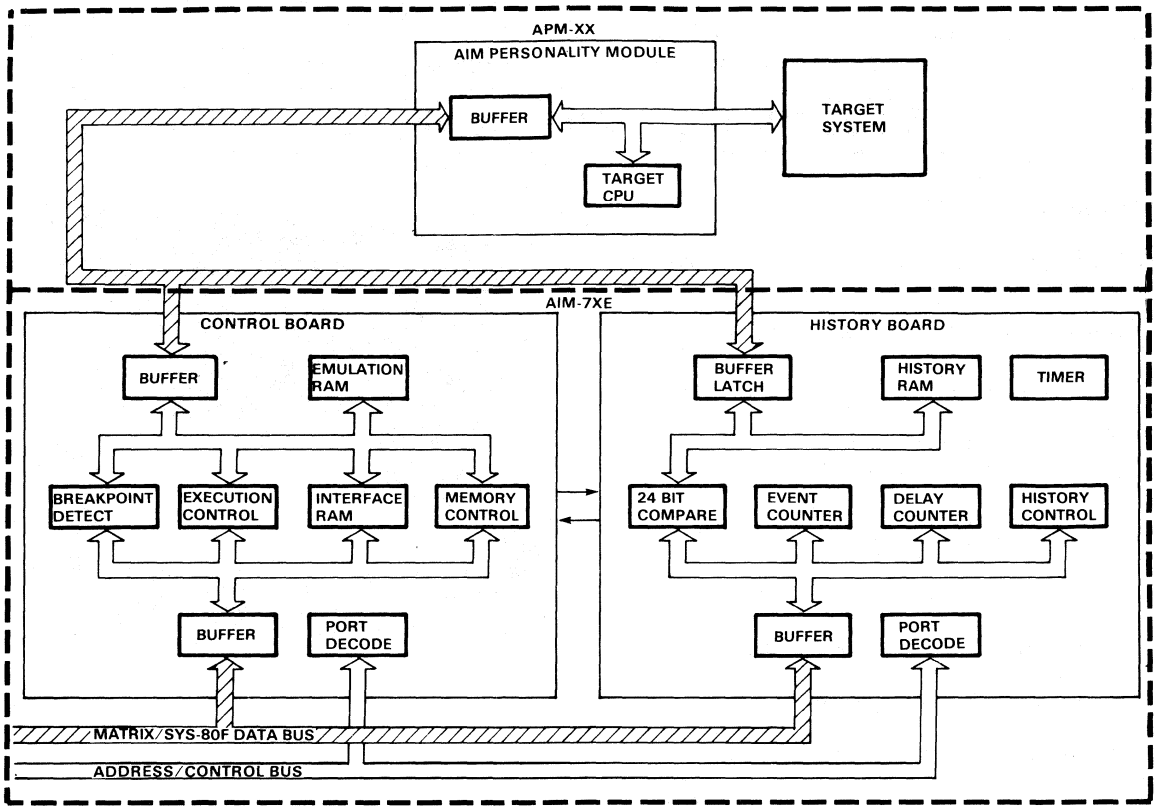
Single-step circuitry allows the user to execute Target instructions one at a time to see the exact effect of each instruction. Single step is functional in ROM as well as in RAM.

4K bytes of emulation RAM may be mapped into the Target memory map so that software may be developed even before Target memory is available. This RAM can be expanded to 8K bytes.

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 CPU status information saved so that execution may later be resumed. Real-time execution may also be terminated at any time by depressing the Escape key. EVENT and DELAY counters give added flexibility for viewing the the exact point of interest in the user's program.

A forty-eight-channel history circuit will simultaneously record any bus transaction which the user may desire to see. The address bus, data bus, ports 0, 1, 4, 5, control signals plus a selection of nine external probes can be used to monitor the Target system's circuitry.

AIM-7XE BLOCK DIAGRAM



BLOCK DIAGRAM DESCRIPTION

The MK3870 Family emulation system is composed of both the AIM-7XE and personality modules. AIM-7XE consists of two boards as is shown in Figure 1. The cable attached to the personality module contains the Target CPU and plugs directly into the Target system CPU socket. Address, data and control signals are buffered by the personality module and cabled to the Control and History boards installed in the development system.

The Control board has the circuitry for detecting the breakpoint condition(s) and forces program execution to begin in the System Interface RAM. The System Interface RAM 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 History board has a 24-bit comparator circuit to detect the hardware breakpoint condition, the EVENT counter, and the DELAY counter. 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.

USING THE AIM-7XE

AIM-7XE consists of two boards. The Control and History boards are installed directly into the development system. To complete the emulation system a personality module is required. This module is a buffer interface between the first two modules and the Target system's MK3870-family CPU socket. Upon completion of installation of AIM-7XE, the development system is turned on, and the operating software is booted up as normal.

All development system software and hardware remain functional. The software to control the AIM-7XE emulation system is named AIM7X. Using the implied run command, AIM7X will sign on, and take control of the Target system. The user can then initialize the Target system and use any of the AIM7X commands to load, test, and debug the Target program.

AIM7X SOFTWARE

AIM7X is the software designed to operate the AIM-7XE emulation system in the Mostek MATRIX-80/SDS Dual-Floppy-Disk Software Development Systems. The software is supplied on standard FLP-80DOS diskette. The commands available with AIM7X 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 locations s through f to Target memory, starting at location d.
,D s,f	Dump the Target memory block locations s through f to any desired binary disk file.
,E s,f	Begin real-time execution starting at Target memory address s with an optional breakpoint set at location f.
,F s,f,d	Fill the Target memory block, locations s through f, with data d.
,G s	Get binary file s and load it into Target memory.
,H	Hexadecimal arithmetic.
,I	Initialize the AIM-7XE system.
,L s,f,d	Locate data d in Target memory range locations 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 disassembly of Target memory.

,O s	Set relative offset equal to s for all address operands. (This feature is extremely useful in debugging relocatable modules.)
,P s	Display and update Target port number s.
,Q	Quit AIM7X and return to FLP-80DOS Monitor.
,R s,f	Display Target registers. Option s specifies the number of registers to be displayed.
,S s,f	Single-step through Target memory starting at 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 block starting at d.
,W s	Write all output in parallel to logical unit s.

Target system programs are developed using the Mostek 3870/F8 Macro Cross Assembler (MACRO-70) and linked using the resident Linker. AIM7X is then used to complete debugging on the user's Target system.

SPECIFICATIONS

Operating Frequency
1 MHz to 4 MHz

Interface Compatibility
MATRIX-80/SDE and SYS-80F

Target Interface
All signals meet the specifications for the MK3870 family with the following exceptions:

1. The XTL2 input will not accept a user crystal. It requires a TTL clock input.

Operating Temperature Range
0°C to +50°C

System Power Supply Requirements (typical)
+5 V ± 5% @ 1.2 A

ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NO.
AIM-7XE	Includes the AIM7XE in-circuit-emulation Control Board, History board, cabling, Operation Manual, and software (AIM7X) on diskette. The emulation Control module is capable of simulating up to 4K bytes of Target memory.	MK79094
APM-70	AIM personality module with cabling for emulation of MK3870 and MK3875	MK79093
APM-73	AIM personality module with cabling for emulation of MK3873.	MK79092
	AIM-7XE Operation Manual only	MK79839
MACRO-70	3870/F8 Macro Cross Assembler, binary program supplied on a standard FLP-80DOS diskette. Includes F8DUMP utility, an extended instruction set macro definition file, and the Operation Manual.	MK79085
	MACRO-70 Operation Manual only	MK79658

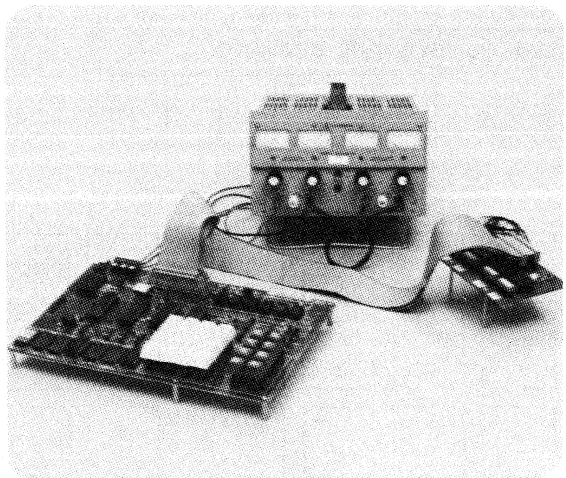
3870 Evaluation System (EVAL-70)

FEATURES

- ☐ An ideal hardware and/or software design aide for the MK38P70 and MK3870 family of Single Chip Microcomputers
- ☐ Includes a 2K byte firmware monitor
- ☐ Keypad for command and data entry
- ☐ 7-segment address and data display
- ☐ Programming socket for MK2716/2758's
- ☐ Crystal controlled system clock
- ☐ 2K bytes of MK4118 static RAM (up to 4K optional)
- ☐ Sockets for up to 4K bytes of MK2716 PROM's
- ☐ Flexible memory map strapping options
- ☐ Current loop or RS-232 serial loader optional (110-300-1200 baud)
- ☐ 3 general purpose timer/counters
- ☐ 3 general purpose external interrupts
- ☐ Easy to use - requires only two supplies for normal operation (+5, +12)
- ☐ Ideal for evaluation of MK3870 family single-chip microcomputers
- ☐ Full in-circuit emulation of MK3870 single-chip microcomputer family.

DESCRIPTION

EVAL-70 is a single board computer with on-board keypad, address and data displays, and 2716 PROM programmer. EVAL-70 is designed to be an easy-to-use introduction to the industry standard MK3870 family of single-chip computers. Programs can be written and debugged in RAM using the powerful DDT-70 operating system. The 40 pin AIM cable can be used to perform real-time emulation of the MK3870 family of devices. After debugging, programs can be loaded into MK2716's for final circuit checkout (and emulation).



USING EVAL-70

The photograph above shows how EVAL-70 is used as a program development tool. Only an external power supply is required for operation of EVAL-70; the built-in keyboard and display offer all the functions needed to design, develop, and debug programs for the MK3870 family of single-chip microcomputers at the machine code level.

COMMAND SUMMARY

DM:	Display memory: allows memory to be displayed and (RAM) updated.
DR:	Display registers: allows the user's register values to be displayed and updated.
DP:	Display ports: allows the contents of ports 0 thru F to be displayed and updated
HX:	Hex calculator: allows hexadecimal arithmetic calculations to be performed (add and subtract)
GO:	causes execution of a user program at a specified address

BK: Breakpoint: allows a breakpoint to be set or reset

ST: Step: causes single-step execution of a user program at a specified address

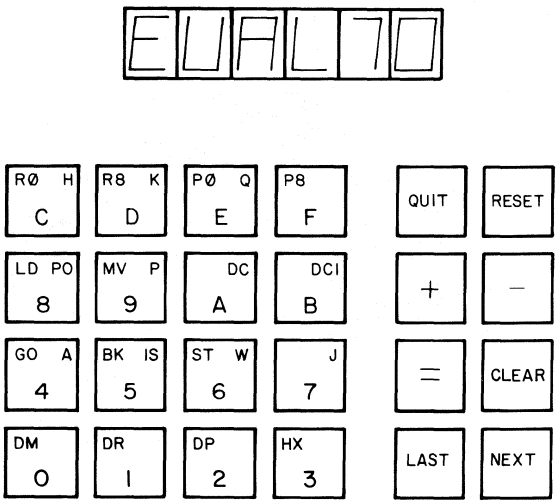
LD: Load: initiates the serial loader (optional)

MV: Move: allows a block of memory to be moved or copied from one space to another

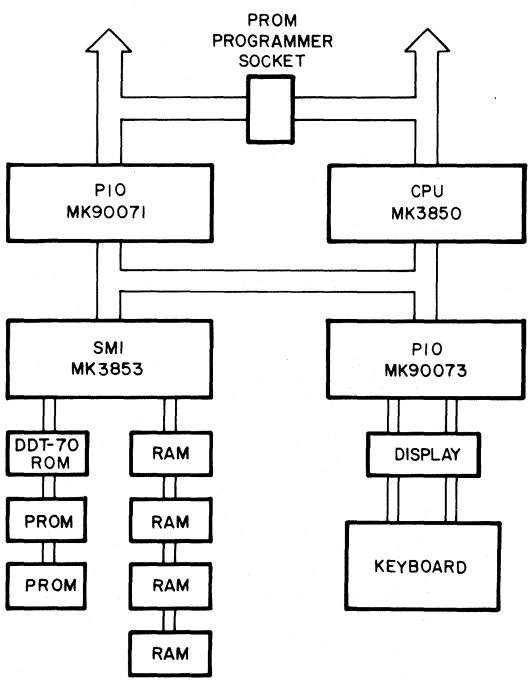
RO, R8: Read PROM: causes the PROM programmer socket to be read into address space 00-7FF or 800-F7F

PO, P8: Program PROM: causes the contents of address space 000-7FF or 800-F7F to be programmed into the PROM programmer socket

EVAL-70 KEYBOARD DRAWING



BLOCK DIAGRAM



BLOCK DIAGRAM

EVAL-70 uses several members of the F8 multichip family. A MK3850 Central Processing Unit (CPU) provides the ALU, registers, system control and two 8-bit ports. A MK90071 Peripheral Input Output chip (PIO) provides two more 8-bit ports plus a flexible timer/interrupt control block. These four ports are connected to the AIM cable connector for in-circuit emulation of the MK3870 family devices, and also to the PROM programmer socket. An additional PIO (MK90073) interfaces the LED display and keyboard. A MK3853 Static Memory Interface chip (SMI) interfaces the operating system ROM, up to two 2K PROMs and up to four 1K RAMs. A switch option allows either the 4K of PROM or the 4K of RAM to appear at address 0000H, with the other 4K appearing at 1000H. The operating system ROM may be up to 8K (currently 2K) starting at 8000H. A switch option allows reset to either 0000H or to the 8000H ROM.

USING EVAL-70 WITH LARGER SYSTEMS

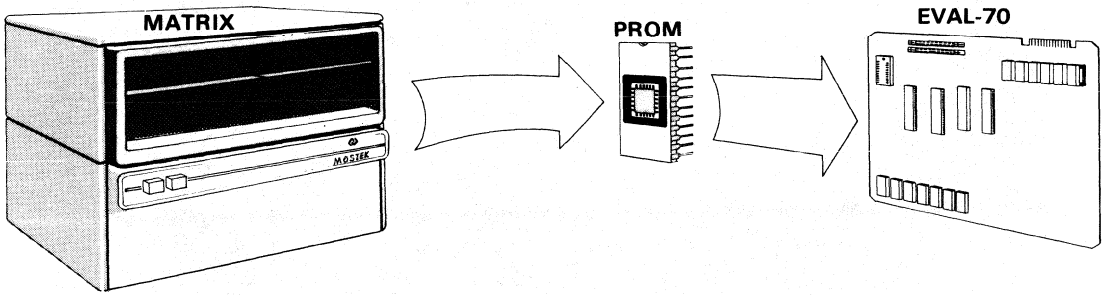
Although the EVAL-70 operating system (DDT-70) was designed to make program machine code entry simple and quick, many users will find it more efficient to assemble their programs on a larger computer and then download to EVAL-70.

The download to EVAL-70 may be accomplished in either of two ways:

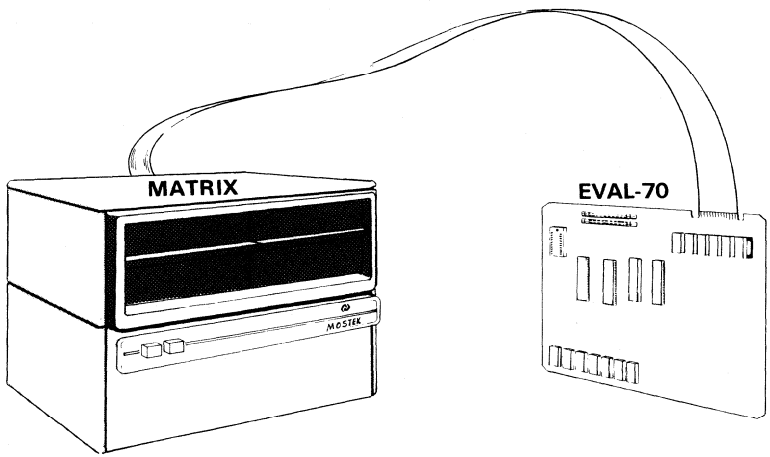
1) A PROM may be programmed on the Development System, and then read into RAM by the EVAL-70 for debugging.
2) A direct connection may be made between a serial port on the Development System and the serial loader port on EVAL-70. An optional serial loader program is provided in the EVAL-70 Operations Manual.

Owners of minicomputers may purchase XFOR-70, a 3870 cross-assembler written in ANSI standard Fortran IV. It may be compiled and executed on any computer system which has at least a 16-bit word length for integer storage and 13K (typical) of memory for program storage.

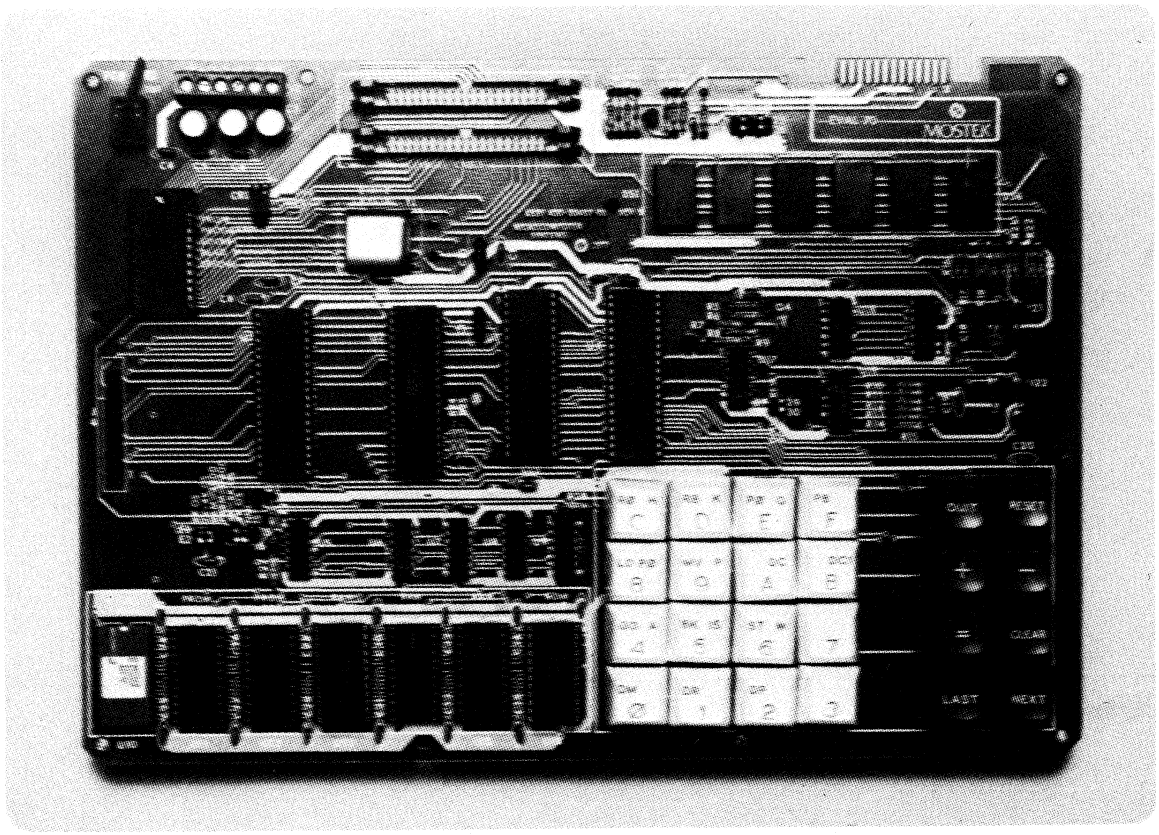
DEVELOPMENT SYSTEM



DEVELOPMENT SYSTEM



EVAL-70 BOARD



Owners of a Matrix Disk Development System may purchase MACRO-70, an advanced 3870 cross assembler. MACRO-70 will generate relocatable, linkable object modules and provides MACRO assembly capability.

SPECIFICATIONS

Operating Temperature: 0°C - 50°C

Power Supplies Required: +5VDC ±5% 1.0A max
+12VDC ±5% 0.1A max
+25VDC ±5% 0.1A max

Board Size: 8.5 in. (21.6 cm) x 12 in. (30.5cm) x 2 in. (5cm)

Connectors and Cables: 40 pin in-circuit-emulation cable is provided.

ORDERING INFORMATION

NAME	DESCRIPTION	PART NO.
EVAL-70	3870 Evaluation System, Assembled and tested with Operations Manual and In-circuit Emulation cable.	MK79086
EVAL-70 Manual	EVAL-70 Operations Manual only	MK79717
XFOR-50/70	FORTRAN Cross assembler for 3870 series	MK79012
MATRIX	Disk Based Development System	MK78189 (50Hz) MK78188 (60Hz)
AIM-72E	In-circuit emulation module for the MATRIX for the 3870, 3872, 3874 and 3876 Microcomputer	MK79077
MACRO-70	3870 Cross Assembler with MACRO capability for the Matrix Disk Development System	MK79085

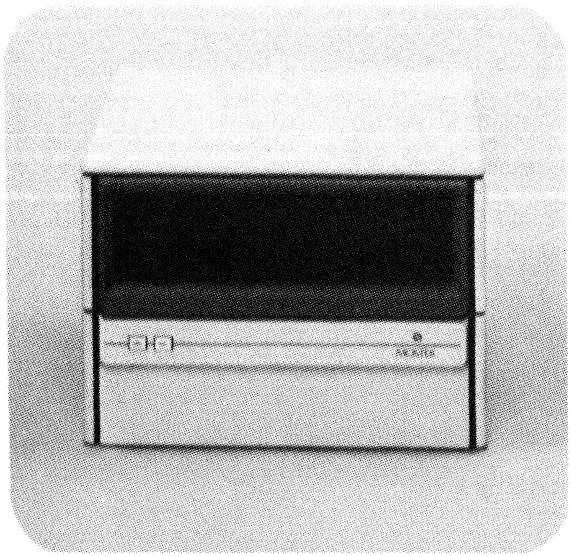
MOSTEK®

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.

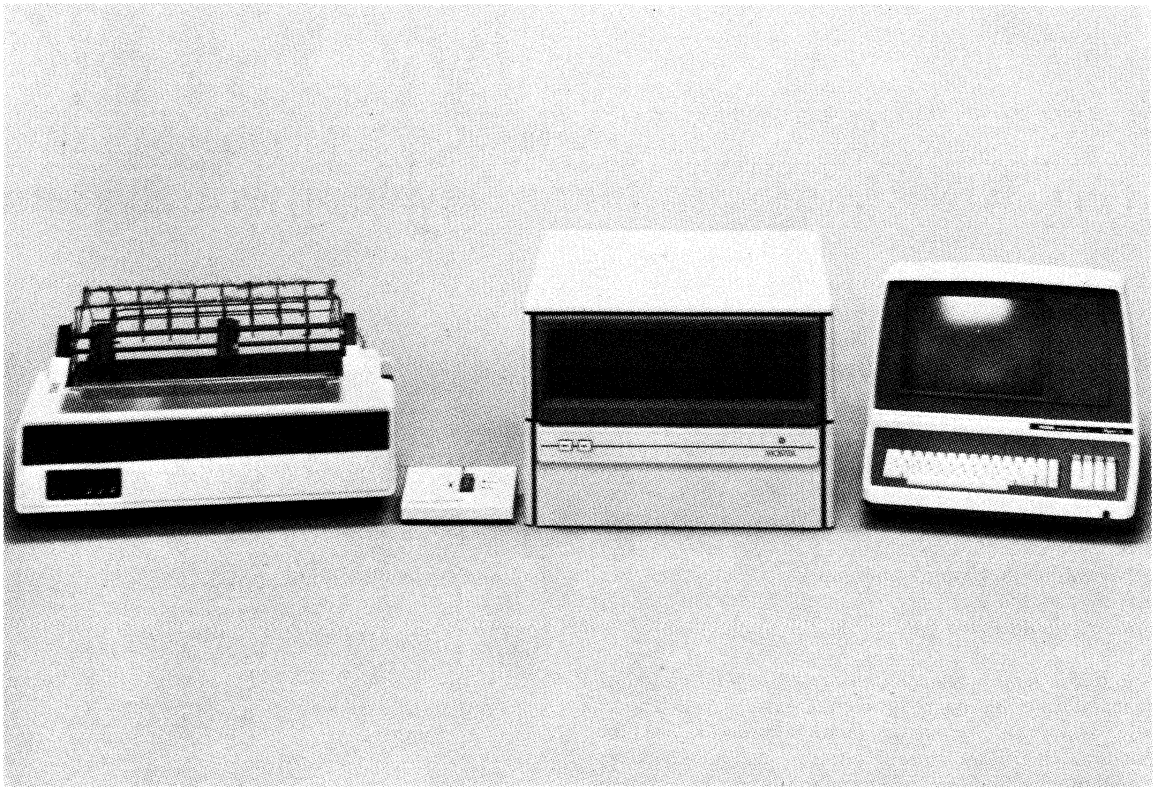
Development System Features

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 general purpose accounting systems is made easy.



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)

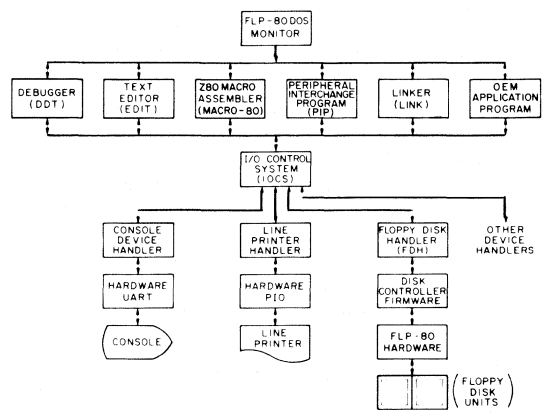
A totally integrated package of resident software is offered in conjunction with the MATRIX consisting of:

- 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 SI 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.

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. DK0) 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 (DK0, DK1, 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

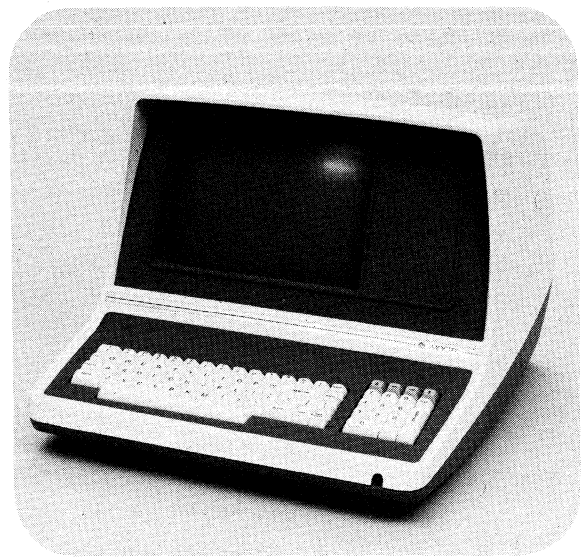
CRT

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

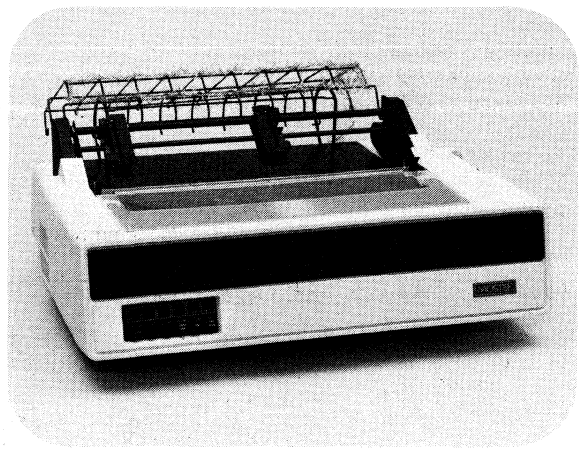
Line Printer

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

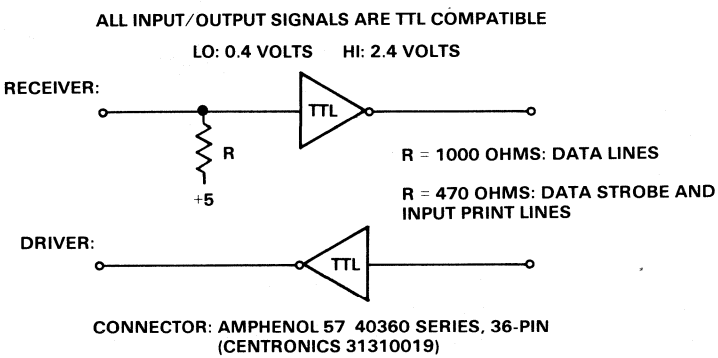
Forms Handling
Tractor feed, for rear or bottom feed forms
8 ips slew rate
Usable paper 4 in. (102 mm) to 17.3 in. (439 mm) width
Paper tension adjustment

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.

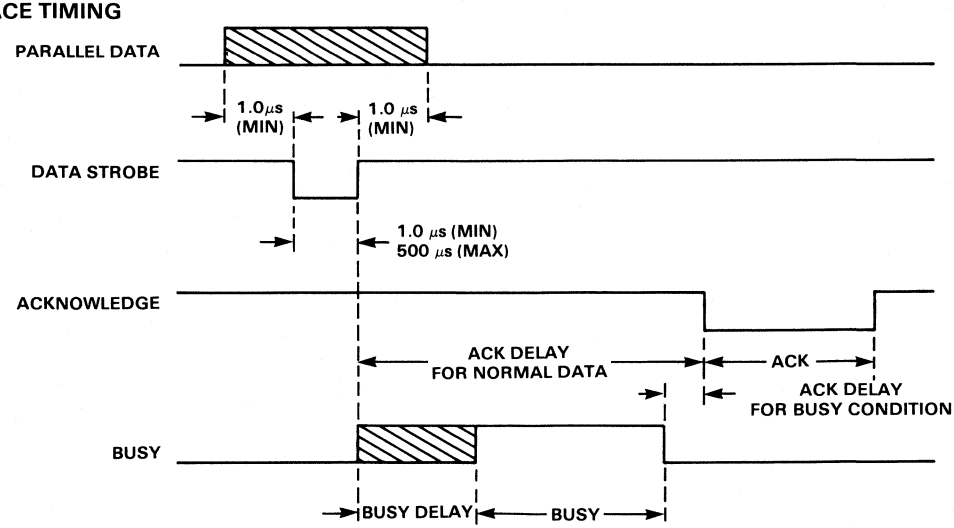
Panel Indicators
Power On: Indicates AC power is applied to printer.
Select: Indicates printer can receive date.
Alert: Indicates operator-correctable error condition.

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



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

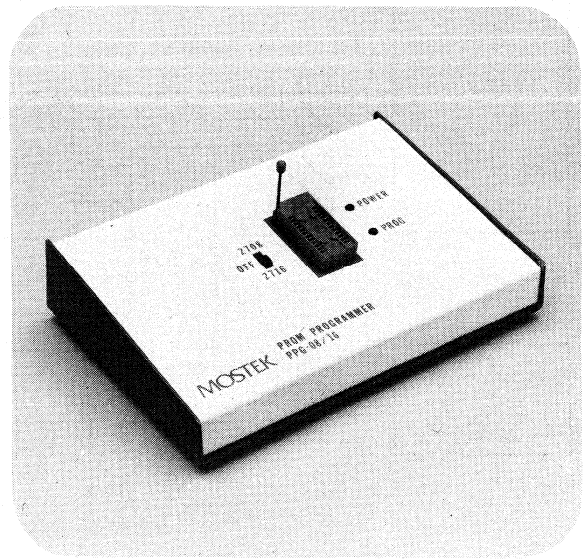
The PPG-8/16 PROM Programmer is a peripheral which provides a low-cost means of programming 2708, 2758, or 2716 PROMs. It is compatible with Mostek's MATRIX Microcomputer Development System and the MDX-PIO. The PPG-8/16 has a generalized computer interface (two 8-bit I/O ports) allowing it to be controlled 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.

SOFTWARE DESCRIPTION

The driver software accomplishes four basic operations.

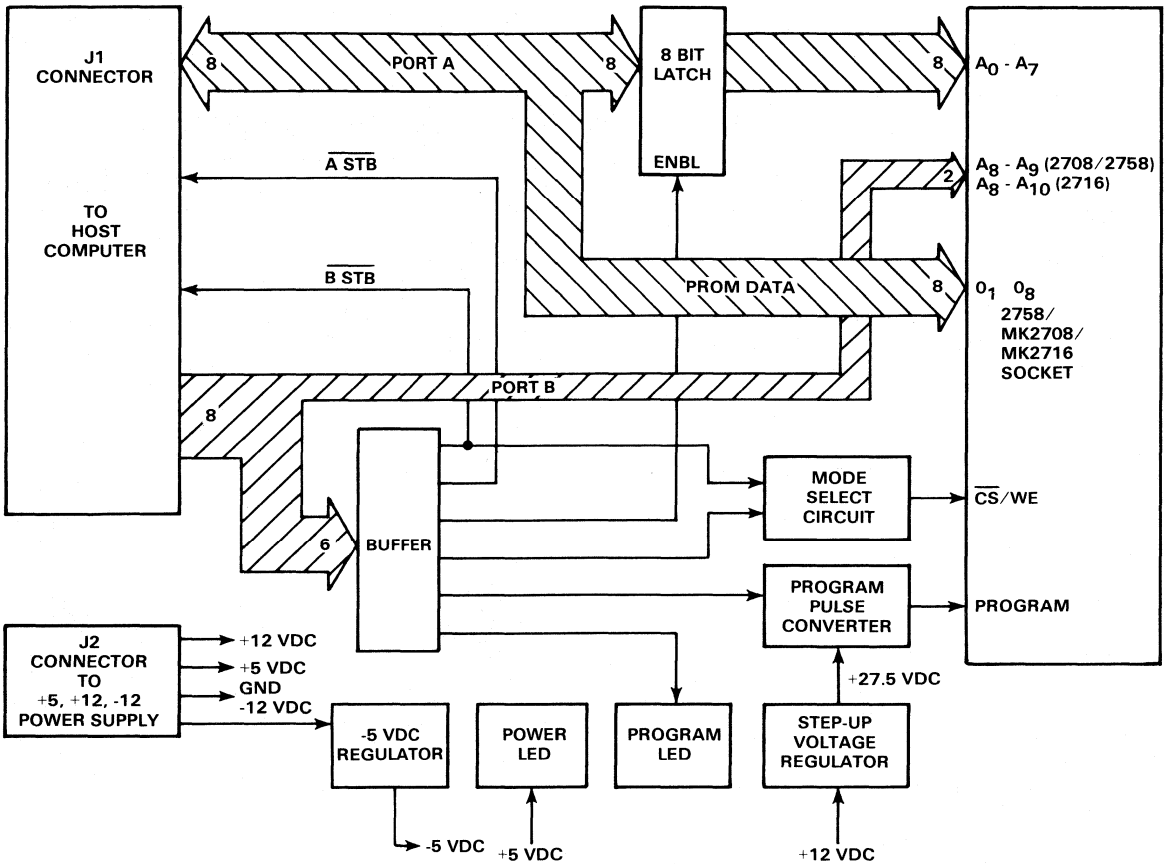
PPG 8/16 PHOTO



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
WAI,					

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
AMAX0	AMAX1	MAX0	MAX1
DMAX1	AMIN0	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.

FLP-80DOS

$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.

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.

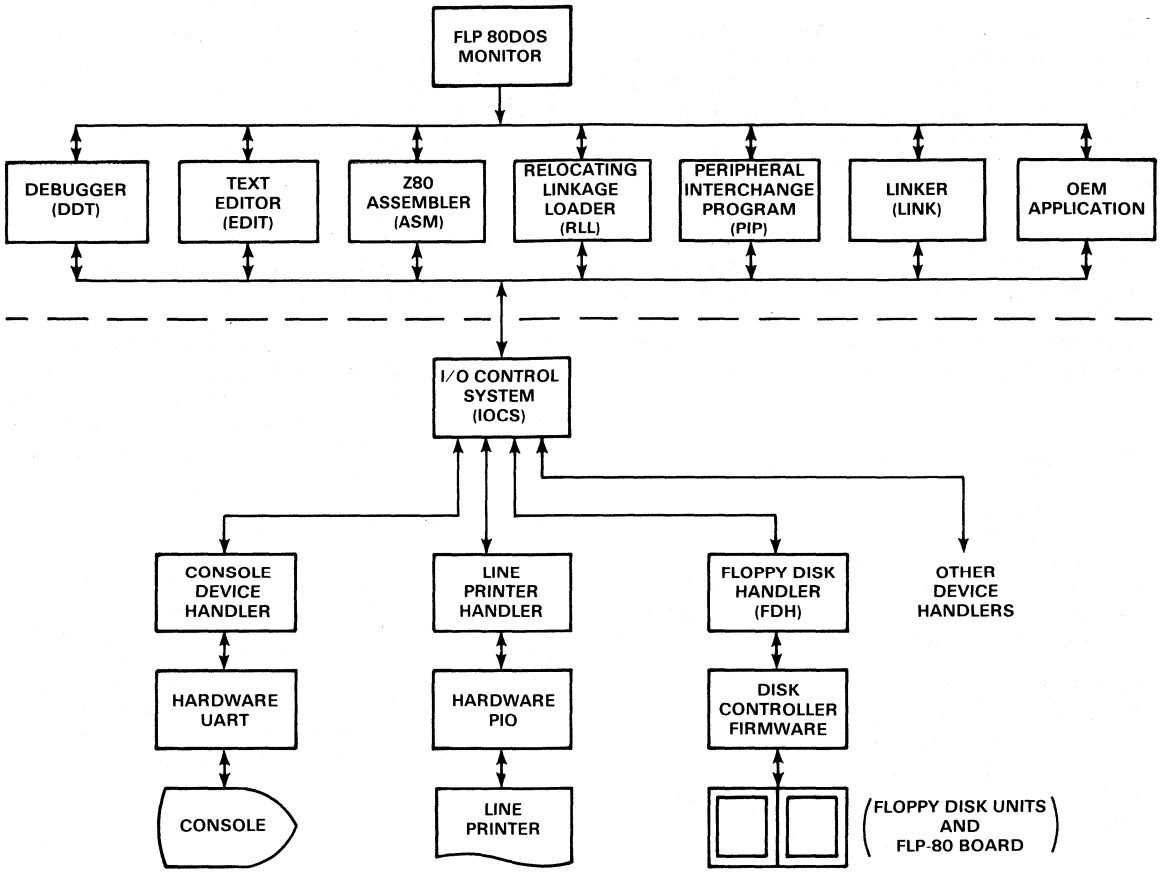
FLOPPY DISK HANDLER - FDH

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

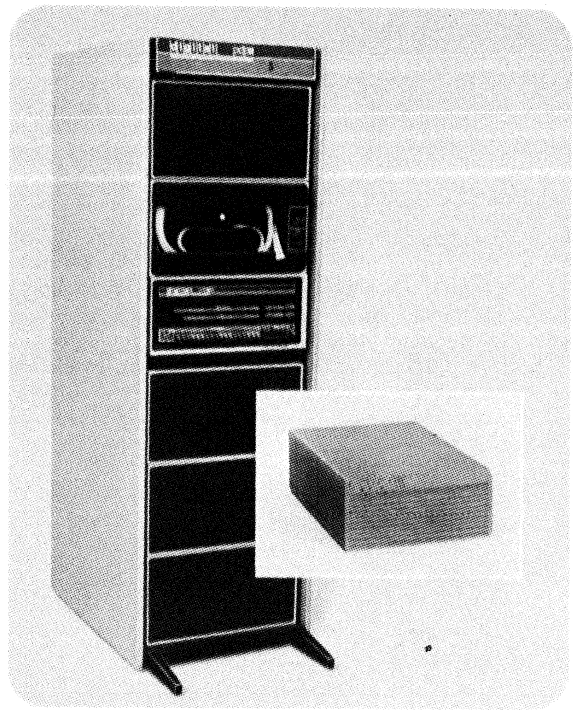
MOSTEK®

3870/F8 MICROCOMPUTER SOFTWARE SUPPORT

Fortran IV Cross Assembler (XFOR-70)

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 available
- ☐ Memory required: 13K words (typical)
- ☐ Assembler directives
  - TITLE 'Set page title'
  - EJECT 'Page'
  - EQU 'Value'
  - ORG 'Beginning address'
  - PUNCH 'Create load tape F8 loader format'
  - PRINT 'Off and On enable for output listing'
  - DC 'Define constants'
  - END
- ☐ Supplied as a standard source card disk



The Mostek 3870/F8 Cross Assembler XFOR-70 is written in ANSI FORTRAN IV. It may be compiled and executed on any computer system which has at least a 8 bit word length for integer storage and 13K of 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 accommodate re-reading of the user input for the second pass of the assembly. Output is in F8 loader format.

The XFOR-70 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 or Customer Purchase Order to:

ORDERING INFORMATION

The XFOR-70 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 or Customer Purchase Order to:

Purchase Order to:

MOSTEK CORPORATION
Microcomputer Systems Dept.
1215 West Crosby Road
Carrollton, Texas 75006

DESIGNATOR	DESCRIPTION	PART NO.
XFOR-70	3870/F8 Cross Assembler written in ANSI Fortran IV is supplied as a source card deck with Operations Manual.	MK79012

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 PARTICULAR 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

MACRO-70

MK79085

FEATURES

- ☐ Assembles standard 3870/F8 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
DC	- define constant
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 MK3870 opcodes via macros, and obtaining an unused-symbol reference table.
- ☐ Provides the most advanced macro handling capability on the microcomputer market which includes:
  - optional arguments
  - defualt 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 statements only
- ☐ An extended instruction set for the MK3870 is defined via a macro definition file and is shipped with the MACRO-70 diskette.
- ☐ Listing and object modules can be output on disk files or any device.
- ☐ Compatible with other Mostek 3870/F8 assemblers and FLP-80DOS Version 2.0 or higher. Requires 32K or more of system RAM.

DESCRIPTION

MACRO-70 is an advanced upgrade from the 3870/F8 Cross Assembler (FZCASM). 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-70 is upward compatible with all other Mostek 3870/F8 Assemblers.

The Mostek 3870/F8 Macro Assembler (MACRO-70) 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 facilities available in MACRO-70 and provides a host of examples

and sample print-outs. An extended instruction set which is designed to ease programming for the MK3870 is defined in the manual. The new instructions are provided on the MACRO-70 diskette in the form of a macro definition file which can be included in a source program.

Downloading to other Mostek systems is facilitated by a utility program called F8DUMP, which is supplied on the MACRO-70 diskette.

ORDERING INFORMATION

DESGINATOR	DESCRIPTION	PART NO.
MACRO-70	3870/F8 Macro Cross Assembler, binary program supplied on a standard FLP-80DOS diskette. Includes F8DUMP utility, an extended instruction set, macro definition file, and the Operations Manual.	MK79085
	MACRO-70 Operations Manual	MK79635

3870/F8 MICROCOMPUTER DATA BOOK

MOSTEK®

USING THE MK3873 SERIAL PORT

Application Note

GENERAL

The purpose of this application note is to familiarize the user with the MK3873 serial port and to provide information to assist the user in the application of the MK3873. The MK3873 is part of the growing family of single-chip 3870 microcomputers. It contains up to 2K bytes of on-chip ROM, a 64 byte scratch pad RAM, and an optional 64 bytes of executable RAM, in addition to the versatile timer, parallel ports and interrupt system which characterize the family. The MK3873 is software compatible with other members of the 3870 family.

The distinguishing feature of the MK3873 is its serial port, which is essentially a USART integrated into the I/O space of a basic MK3870 microcomputer. Supporting the serial port are a list of features that include the following:

- ☐ Programmable internal Baud rate divider
- ☐ Programmable character length
- ☐ Double buffered receive and transmit data
- ☐ Start bit detection
- ☐ Half-duplex asynchronous operation
- ☐ Half- or full-duplex synchronous operation
- ☐ Data underrun/overrun error detection
- ☐ Separate vectored interrupts for receive and transmit

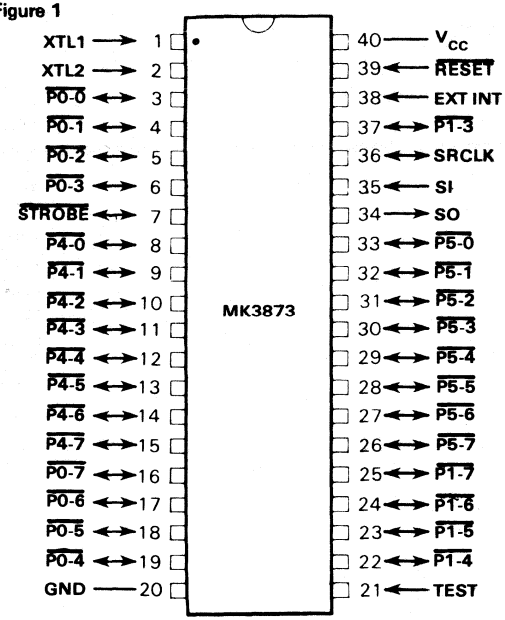
The serial port completely takes care of the serializing and deserializing of data. Interface to the serial port is simplified by the addition of other on-chip ports, including one for Baud Rate, one for control, and two for data. The new ports are read and written in exactly the same way all the MK3870 ports are accessed. There are no new instructions required. Control of the serial port is handled through I/O reads and writes.

DESCRIPTION OF THE SERIAL PORT

The serial port is accessed in firmware using INS and OUTS instructions to the addresses shown in the programming model in Figure 1. OUTS to the four port addresses, 0CH, 0DH, 0EH, and 0FH, allow writing of information into the serial port section. INS from the four port addresses read

PIN CONNECTIONS

Figure 1



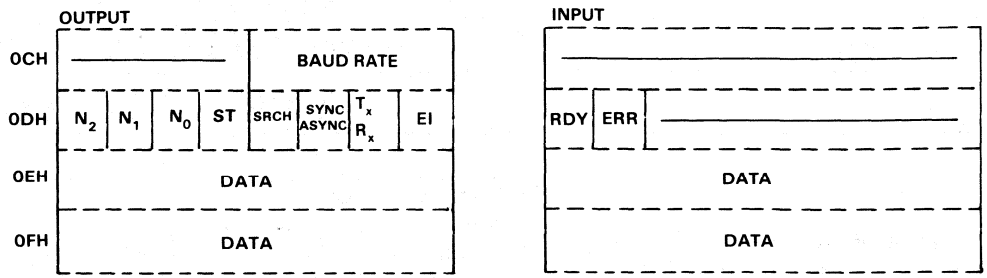
information from the serial port section as shown in the figure. The Baud rate port, 0CH, is written to set the transmission frequency. The Control port is written to set word length and mode, and may be read to determine the state of the Ready and Error flags. Characters to be transmitted are output to the register pair made up of ports OEH and OFH. Received data are read by inputs from the same register pair.

BAUD RATE PORT

Port 0CH is used for setting the Baud rate divisor factor for the serial port. The serial data rate for both the transmit and receive direction of the serial port depend on the divisor value and whether or not the divide-by-sixteen prescale is enabled. The prescale is enabled by the SYNC/ASYNC bit in the control port, 0DH. This bit is also called the 1X/16X bit, since the 1X clock is used in synchronous mode and the 16X clock is used in asynchronous mode. The clock that is actually presented to the SRCLK pin when the internal Baud rate clock is determined exclusively by the Baud rate port value, and is not affected by the SYNC/ASYNC bit.

SERIAL PORT PROGRAMMING MODEL

Figure 2



Baud Rate Code in 0CH	Actual Baud Rate SYNC/ASYNC Bit = 1	Actual Baud Rate SYNC/ASYNC bit = 0
0	EXTERNAL	EXTERNAL
1	NOT USED	NOT USED
2	NOT USED	NOT USED
3	1200 Baud	75 Baud
4	1759 Baud	110 Baud
5	2400 Baud	150 Baud
6	4800 Buad	300 Baud
7	9600 Baud	600 Baud
8	19.2 Kbps	1200 Baud
9	38.4 Kbps	2400 Baud
A	76.8 Kbps	4800 Baud
B	164 Kbps	9600 Baud
C-F	NOT USED	NOT USED

N_2	N_1	N_0	Word Length (bits)
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

Mnemonic	Meaning
ST	Start bit detect mode
SRCH	Enter hunt mode (search)
SYNC/ASYNC	Baud rate clock multiplier
TX/RX	Transmit or receive mode
EI	Enable serial port interrupts
RDY	Rx data ready or Tx buffer empty
ERR	Underrun or overrun error

In general, the Baud rate port will be initially loaded upon power-up with a value stored in ROM or input by 1 or more port pins. The Baud rate port may be updated periodically by hardware or software conditions. The Baud rate port cannot be read. If it is desired to retain a copy of the current setting of the Baud rate, a location in RAM should be reserved for that purpose.

A similar condition exists for both the Serial Control Port, ODH, and for the Interrupt Control Port, 06H. That is, if it is desired to remember what was written to these ports, locations in RAM need to be set aside for that purpose.

CONTROL PORT

Port ODH is the control port used to set the operating characteristics of the receiver and transmitter portions of the serial port. The word length bits N_2 - N_0 are used in determining the length of the serial character including the start bit, data bits, parity and stop bits. For ASCII async, this usually totals eleven bits: a start bit, seven bits of data, a parity bit and two stop bits. Often the parity bit is removed, making a total of ten bits. Synchronous transmission almost always uses eight bit lengths. No start or stop bits are used.

The start detect bit, ST, is used to achieve bit synchronization when in async mode. Setting ST to one permits the sampling of the incoming data stream at sixteen times the bit rate looking for logical zero which signals the start of a character frame. To filter out noise on the serial line, when a zero is first detected it is retested later at a time equal to half a bit time to verify the validity of the data frame.

In the event that the second sample fails to result in zero, an error is assumed, and the control sequence starts over. If the second sample is a zero, the strobing will continue every bit time that follows until the complete word is shifted in. The half bit timing offset thus achieved puts the strobing in the optimum phase for detecting data for the rest of the frame.

Search mode, as controlled by the SRCH bit in the control port, is used in the synchronous mode while the firmware is attempting to achieve character synchronization with the incoming data stream. An interrupt is given by the serial port every bit time, enabling software examination of the incoming stream on every bit and comparing the result against the sync character. Once the sync character is found, the SRCH bit is cleared causing the interrupt to occur only after each entire word is shifted into the serial port.

SYNC/ASYNC is used to control the clock prescale. In the synchronous mode the SYNC/ASYNC bit is set, turning off the prescale and shifting the serial port at the same rate as the clock. In the asynchronous mode, the bit is clear, causing the serial rate to be one-sixteenth that of the clock. This allows the bit synchronizer to sample the bit-stream in the center, as was described in the discussion on the START DETECT bit. The actual clock presented to the output is unaffected by this bit. The bit only serves to determine the internal prescale.

XMIT/REC enables the transmitter portion of the serial port. When set, the port is in the transmit mode. Data written to ports 0EH and 0FH will be serialized and sent out through the port. When reset, the transmitter is disabled and the port is in the receive mode. Full duplex operation is accomplished in the sync mode by first setting the search bit, SRCH, to achieve character synchronization, and then operating with the XMIT/REC bit set, allowing simultaneous functioning of the receiver and the transmitter. XMIT/REC also determines the serial port interrupt vector. If XMIT/REC is set, the vector is E0H. If reset, the vector is 60H. The EI bit is used to enable interrupts when either the receiver has a character ready to input or the transmitter is ready to accept another character.

When the Control port, 0DH, is read the information presented are the ready bit, RDY, and the overrun/underrun error bit, ERR. The RDY bit signifies that the port has counted down the number of bits specified in the WLEN field in the control port. This bit is equivalent to a buffer ready bit in a common serial port configuration, except that it is multiplexed between the receiver and the transmitter. If the bit is set while in the receive mode, for example, it means that the receiver has data waiting to be read from the data ports. If the port is configured in the transmit mode, that bit is used to signify that the transmitter is ready to be loaded. The RDY bit is reset by an INS or an OUTS to either port 0EH or 0FH.

The ERR bit is set by the receiver when it loads a new character into the receive holding register before the RDY bit has been reset. It is also set by the transmitter when the character going out clears the port without the RDY bit being reset. In the case of the receiver setting the error flag, it is termed an overrun error. When the transmitter sets the error flag it is called an underrun error.

A certain amount of caution is recommended in the use of the RDY and ERR bits in deciding when to switch the port from Transmit mode to Receive mode. In half-duplex operation it is customary to wait until the transmitted character has cleared the shift register prior to dropping Request To Send, and turning the line around. The ERR bit goes set when the last bit clears the shift register. The RDY bit had to have remained set to enable the setting of ERR. When the serial port is now switched to the receive mode, the RDY bit is still set, offering the opportunity for an erroneous setting of the overrun condition while in receive mode. To avoid this, it is necessary to do an input of the data port, 0EH or 0FH, to reset the RDY bit for subsequent receive mode operation.

DATA PORTS

Ports 0EH and 0FH are access ports to two 8-bit registers used as data holding registers. When outputting to the port, the most significant byte of data is written to port 0EH, and enters the upper position of the transmitter holding register. The lower byte of data is written into the lower position of the transmitter holding register via port 0FH. When inputting data, the upper byte of the receiver holding register is read through port 0EH, and the lower byte is read through port 0FH. In operation, data is put into the receiver holding register by the serial port logic at end-of-word time after the number of bits, specified by WLEN in port 0DH, has been shifted into the shift register. When the transmitter is enabled by the XMIT/REC bit, the transmitter holding register is gated into the shift register at every end-of-word time.

The entire sixteen bits are read and written between the holding registers and the shift register. Therefore, characters are not right justified, and start and stop bits are not added or stripped.

REVIEW OF DATA COMMUNICATIONS

This section presents a brief overview of some of the terms and fundamentals of data communications which may be useful background information for the remainder of the document.

SERIAL TRANSMISSION

The topic of serial data communications falls necessarily into two different areas: asynchronous and synchronous. The difference between asynchronous, (async), and synchronous, (sync), lies primarily in the fact that async carries no clock with the data, and sync does. Sync communications always has the clock along with the data, so that receiver bit synchronization is maintained. Data

occurs regularly and continuously in a predictable manner. In the case of async, data occurs at unpredictable times and uses a start bit to signal the beginning of a character. The character is terminated with one or more stop bits. The extra bits in async contribute to loss of efficiency in this category of transmission.

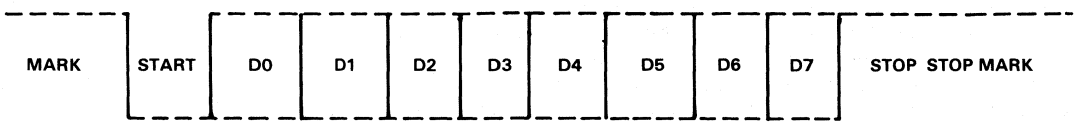
ASYNC

The start bit in an async character allows the receiver to time the strobe and extract the data from the bit stream. One or more stop bits may be present at the end of the data bits to terminate the character. Figure 3 shows a format for async data composed of one start bit, then eight data bits, and then two stop bits, which makes up one character of async.

Since the characters are allowed to occur anytime and with any arbitrary phase relationship with other characters, they are considered to be truly asynchronous events. Because two adjacent characters can be end-on-end, the spacing between data may be controlled by the stop bits at the end of the characters. There may be one or two stop bits in most applications.

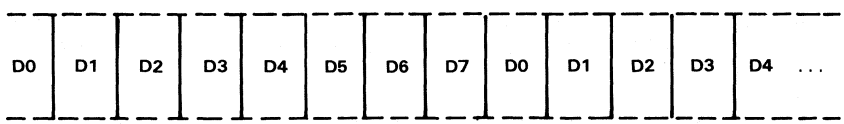
Baud rates for async range from 75 Baud, for low speed telegraph, 110 Baud, for standard teletype, 300 Baud for Bell 103 modems, to 600, 1200, 2400, 4800, 9600, etc. at specific rates up to 19.2K Baud as a practical limit. Actually, the practical limit to Baud rates is a result of the ability of the parts available to serialize and deserialize the data. The frequency tolerances are specified in RS232 to be maintained to about three percent of nominal so that the timing of the sender matches the timing of the receiver.

ASYNCHRONOUS DATA FORMAT
Figure 3



WHERE:
MARK - INACTIVE LINE
D0-7 - EIGHT DATA BITS
START - SYNC BIT
STOP - STOP BITS

SYNCHRONOUS DATA FORMAT



WHERE
D0-7 - DATA BITS

SYNC

Sync data transmission carries the clock with the data. Modems normally manage the clocking and provide the shift clocks to the USART logic in the Data Terminal Equipment. Where no modems are used, as in directly connected links, the shift clocks are provided by one or more of the devices on the link.

Modems are devices for modulating and demodulating a carrier with digital data from a computer. Modulated audio signals are much better suited for transmission down a phone line than are digital signals, mainly because digital signals have sharp rise and fall times that require very high bandwidths. The common voice grade phone line can support frequencies between 300 and 3000 Hz. The modem uses the digital data to shift the frequency of a carrier. At the receiver, the carrier frequency shift is detected and the data is restored to digital logic levels.

Modems are used in sync and async communications. In sync, the data must be continuous, otherwise the receiver would lose synchronization. That is, the receiver recognizes the fact that it has character sync because every so often it decodes a pre-specified bit sequence defined by the protocol as a sync character. If a transmitter must cease sending new information, it may be made to transmit continuous sync characters for some maximum length of time determined by the line protocol. In this way the receiver remains synchronized.

FULL AND HALF DUPLEX TRANSMISSION

The following analogy is presented to give an intuitive insight to defining half- and full-duplex communication.

When two people are carrying on a conversation, usually one person talks until he is through and then the other person talks. When person A talks, person B listens; when person A is through talking for a while, he stops and person B may acknowledge the truth in what he heard A say and then go on with his side of the story. This mode of conversation, where one person talks while the other listens, is called half-duplex communications. Imagine what confusion there would be if both A and B tried talking at the same time! Actually, the only reason this causes trouble is because person A and person B are half-duplex people. If they were full-duplex people, and capable of hearing and understanding while they talk, this could be a very efficient way to communicate. Full-duplex communication consists of simultaneous two-way communications: two stations transmitting and receiving at the same instant in time. Generally, the two transmissions do not occupy the same frequency spectrum, as would be the case where two people talked simultaneously in the same voice frequency range. Typically, the available bandwidth, such as that provided in a telephone line of 300-3000 Hz, is divided into two parts and used by the originator and by the answering station without conflict. The originator in a Bell 103 full-duplex modem marks at a frequency of 1270 Hz, while the answering station transmits marks at a frequency of 2225 Hz, resulting in both stations band-pass filtering the frequency range each expects to receive and rejecting the rest. This provides a very satisfactory full-duplex communications channel across which both stations freely transmit and simultaneously receive data.

In actuality, the people talking at each other at the same time were really using a full duplex channel. The air satisfactorily conducts both voices to the ears of the listeners without distortion. Communication would have occurred if both person A and person B had agreed to a protocol whereby acknowledgements would be permitted on the full-duplex media. Then person B, for instance, would only talk when solicted by A for comment.

Very often in communications, a full-duplex link is used to conduct communications in a half-duplex protocol. BISYNC is an example of a half-duplex protocol. BISYNC often uses a full-duplex modem to avoid the lengthy waits associated with turning the line around in a half-duplex medium.

RS232

RS232 is the most common and by far the most popular data communications interface standard in the industry. It allows the creation of communication controllers, peripheral controllers and many more interfaces that interconnect with usually a minimum of adjustment.

RS232 is an interface standard for serial data communications, set up by a standards committee of interested users in order to define a common set of rules by which equipment made by different manufacturers could interface predictably. The RS232 interface standard covers the assignment of pin numbers in a standard 25 pin "D" connector to a set of defined functions for use in the interface between Data Terminal Equipment, (DTE), and Data Communications Equipment, (DCE), as defined by the EIA. DTE refers to computer terminal devices such as CRTs or ports on a computer peripheral controller. DCE refers to modems and other telecommunications equipment that is generally associated with the common carriers.

RS232 also defines the voltage levels, rise and fall times, receiver impedance, etc. RS232 states pin 2 is to carry transmitted data from the DTE to receivers on the DCE, (modems, for instance), and that pin 3 is to carry received data on the DCE back to receivers on the DTE. Also included among the signals defined are Data Carrier Detect - DCD, Ring Indicator - RI, Request to Send - RTS, Clear to Send - CTS, Data Terminal Ready - DTR, Data Set Ready - DSR, and several others for use in both async and sync communications.

MK3873 PROGRAMMING

This section deals with practical examples of applications of the serial port on the MK3873 microcomputer. The programming examples have been actually run and verified on the MK38P73/02 EPROM version.

INITIALIZATION OF THE SERIAL PORT

Initialization of the serial port sets the default parameters for the port. Following a power-on reset, it is necessary to set the Baud rate, word length, modes of transmission, and enable the serial port interrupts. The following sequence is offered as an example.

***INITIALIZE THE SERIAL PORT**

```
BDRATE   EQU    6          DEFAULT BAUD RATE TO 300
INITSP   EQU    H'91'      CONTROL PORT INIT VALUE
BRPORT   EQU    H'C'       BAUD RATE PORT
CNPORT   EQU    H'D'       CONTROL PORT
DUPORT   EQU    H'E'       UPPER DATA PORT
DLPORT   EQU    H'F'       LOWER DATA PORT
*
INIT     LI     BDRATE     BAUD RATE TO ACCUM
         OUTS   BRPORT     OUTPUT TO BAUD RATE PORT
         LI     INITSP     CONTROL VALUE
         OUTS   CNPORT     WRITE TO CONTROL PORT
         CLR               CLEAR TRANSMIT DATA
         OUTS   DUPORT     UPPER HALF
         OUTS   DLPORT     LOWER HALF
         INS    CNPORT     CLEAR ERROR STATUS
         INS    DUPORT     CLEAR READY STATUS
         *
```

RECEIVER PROGRAMMING EXAMPLES

In programming the serial port receiver, once the Baud Rate port has been initialized, generally there are two parts of code involved: a start routine and an interrupt level read routine. The start routine puts the port into the receive mode and enables the port to interrupt when a character arrives. The interrupt level routine reads the port, checks for errors and sets a ready flag in scratchpad RAM.

In the example, measures are taken to clear errors and flags that stray into the picture from one source or another. While the examples are not intended to represent the only way, or even the best way, to handle the serial port, they should serve to illustrate ways of avoiding some common errors often made in operating the serial port. The examples may be built upon to handle errors, protocols, etc.

```
         *START THE RECEIVER
         *
DATA     EQU    6          DATA PLACED IN REG 6
RXRDY    EQU    H'80'      READY FLAG "OR"ED INTO DATA
RXCMD    EQU    H'B1'      RECEIVE COMMAND FOR CNTL PORT
ASAVE    EQU    5          ACCUM SAVED IN REG 5
         *
         *
RXSTRT   LI     RXCMD      OUTPUT THE RECEIVE COMMAND
         OUTS   CNPORT       TO THE CONTROL PORT
         INS    CNPORT     CLEAR ANY LEFT OVERS
         INS    DLPORT       HERE, TOO
         POP               RETURN FROM SUBROUTINE
         *RECEIVER INTERRUPT ROUTINE
         *
                ORG    H'60'
         *
         *
RXINT    LR     J,W        SAVE STATUS
         LR     ASAVE,A    SAVE ACCUMULATOR
         *
         INS    DHPORT     READ UPPER DATA HALF
         SL     1          ASSUME EIGHT DATA BITS
         SL     1          TWO STOP BITS
         LR     DATA,A     STORE TEMP
         INS    DLPORT     GET LOWER DATA
         SR     4          MOVE THE DATA OVER
         SR     1
         SR     1            SIX PLACES
         XS     DATA       MERGE THE DATA
```

```
                LR      DATA,A      SAVE THE DATA FOR AWHILE
                *
                *
                INS     CNPORT      READ THE READY/ERROR
                SL      1           CHECK FOR OVERRUN ERROR
                LR      A,DATA      READY TO MARK READY
                BP      RXINT1      CONTINUE IF NONE
                *
                LI      H'7F'       REPLACE WITH RUB OUT
RXINT1          OI      RXRDY       MARK THE DATA READY
                LR      DATA,A
                *
                LR      A,ASAVE     GET READY TO RETURN
                LR      W,J         FROM INTERRUPT
                EI                  RESTORE INTERRUPTS
                POP
                *
```

TRANSMITTER PROGRAMMING EXAMPLES

In async, the transmitter is usually programmed and used without the serial port interrupt enabled. This is common in half-duplex applications, since it is required to hold off turning the port around into the receive direction until the shift register has cleared the transmit character. The emptying of the shift register is detected by repeatedly inputting the control port looking for an underrun error. This flag is set by the port when the last transmitted bit clears the shift register output. Reading the control port automatically clears the underrun error. The following example just sends one character and returns the port to the receive mode after the character clears the port. It may be called repeatedly.

These examples show subroutines completely unprotected from interrupts. In practice, it is necessary to make some provision for the possibility of interrupts. In these examples, interrupts should be disabled prior to calling the subroutines, and enabled again upon return to the in-line code.

```
                *
                *TRANSMIT A CHARACTER
                *
TXCMD           EQU     H'B2'       COMMAND FOR CNTL PORT
                *
                *
TXMIT           LR      A,DATA      DATA TO SEND IN "DATA"
                SL      1           SHIFT FOR TWO PORTS
                OUTS    DLPORT      GOES IN LOWER PORT
                LR      A,DATA      UPPER PORT
                SR      4           SHIFT RIGHT 7 PLACES
                SL      1
                SR      4
                OI      6           STOP BITS
                OUTS    DHPORT      GOES IN UPPER PORT
                *
                LI      TXCMD       COMMAND FOR CNTL PORT
                OUTS    CNPORT      MAKE IT GO
                *
TX   TXLP       INS     CNPORT      CHECK FOR TX MT
                SL      1           SHIFT ERROR INTO SIGN
                BP      TXLP
                *
                LI      RXCMD       RESTORE PORT TO RX MODE
                OUTS    CNPORT
                *
                POP                 RETURN FROM SUBROUTINE
                *
```

REAL TIME CONSIDERATIONS

This section deals with the special attention to real time required by the serial port. Control of real time functions entails an awareness of the actual speed of the processor, including how fast it responds to interrupts, and how long certain instructions take to execute. The speed of the processor, and that of devices communicating with it, may ultimately determine the maximum practical Baud rate the serial port may be programmed to, rather than the speed at which the receiver or transmitter can handle the raw data.

Assuming a clock speed of 3.6864 MHz in order to make the Baud rates come out right, the fastest MK3873 instruction takes 2.17 microseconds to execute. At a Baud rate of 9600, an eleven bit async character takes 1.15 milliseconds start to finish. That means that in one character time, an MK3873 is able to execute up to 528 instructions and still not have the communication channel set away from it. Some time is spent by the MK3873 just responding to an interrupt from the serial port, and some more time saving context, reading in the data, shifting it, etc.

RESPONDING TO INTERRUPTS

On the average, an interrupt routine takes about 25 microseconds to get started. That is, from the time an interrupt is received by the CPU until the first instruction in the code located at the vector location can execute, is about 25 microseconds. Fifteen of those microseconds are actually used up by the CPU in responding, and the remaining ten is queuing delay.

Section 4 shows an example of an interrupt routine that reads data from the serial port and places it into a cell in scratchpad RAM. That particular routine, called RXINT, takes approximately 80 microseconds to execute, at the 3.6864 MHz clock rate. Typically, there would be in line code that would periodically check the location, DATA, to see if a character has arrived. The character would then be displayed, or put into a buffer somewhere. The code would have to check the cell every character time to be effective.

It is readily seen that the MK3873 should have no difficulty handling 9600 Baud async with the interrupt routine in the example, provided it could process the received data and do whatever transmitting was required during the one millisecond, approximately, that remains. The overhead involved in handling the reading of the port is of the order of ten percent of the total amount of time available.

DOUBLE BUFFERING BENEFITS

The MK3873 has what is called "double buffered" receive and transmit ports. This means simply that the shift register that carries the serial data is not read or written directly, but through holding registers. The transmit holding register and the receive holding register have the same address, namely 0EH and 0FH. When an output is made to 0EH or 0FH, data goes into the transmit holding register. Likewise, when address 0EH, and 0FH are input, the receiver holding register is what is read. The advantage of these registers is that service is required on a character basis instead of on a bit basis. There is one full character time, eleven bits, from the time an assembled character is loaded into the receive port holding register until the firmware must read the holding register so as not to overrun. If the holding register were not provided, it would be necessary to read out the shift register within one bit time from the time data was ready. Holding registers are an absolute essential when dealing with speeds in the 9600 Baud range.

SPECIAL APPLICATIONS

This section covers some special application details that may prove helpful in using the MK3873 serial port.

FIFO BUFFERING

The topic of double buffering was discussed in the previous section to some degree. Typically, an application will have to handle some amount of interpretation of the incoming data stream, or possibly do some processing on the character string that sometimes occupies the CPU for periods of time that are longer even than a full character time. To accommodate this kind of situation, the use of a circular buffer, or FIFO buffer is useful. FIFO stands for "First In, First Out".

The FIFO buffer is a memory based structure that is used to temporarily store an incoming data stream. The FIFO may be accessed sequentially by reads and data writes. The first written is the data read.

The amount of data in the FIFO buffer varies. The FIFO is often used to connect two asynchronous processes. For instance, a communications port may be receiving data from a sending station located remotely. The sender transmits at its own rate, which might be quite sporadic. The receiver is often connected to a protocol handler which gets data from the buffer, checks it against a special character table, and loads it into a fixed length data storage receptical. The receiver rate is highly unpredictable, and so it is directed into one end of the FIFO buffer. The protocol handler takes characters out the other end of the FIFO buffer at its own rate and processes the data until the FIFO becomes empty.

The FIFO structure typically has a read pointer, a write pointer, a maximum length, and either a full flag or an empty flag. To write into the FIFO, a test is first made to see if the FIFO is full. If full, the write attempt must be put off. If the FIFO is not full, the write proceeds by writing the character at the location pointed to by the write pointer. The write pointer is then incremented to the next available cell in the FIFO. If the write made the buffer full, the full flag is set. Successive writes increment the write pointer all the way to the physical end of the FIFO, where it wraps around to the beginning.

Reading the FIFO is analogous. The FIFO is first checked to see if it is empty, in which case the read attempt is put off. If the FIFO is not empty, the character residing where the read pointer points is read. The read pointer is incremented much as the write pointer was, until the end of the FIFO is reached. At the end of the physical buffer the read pointer also wraps around.

The following example is offered to illustrate the mechanics of FIFO buffering.

The format of the FIFO is eight bytes total, including one byte for the read pointer, write pointer, and full flag, seven bytes for data storage. The FIFO is located in an eight byte grouping in scratchpad RAM. Incrementing of the read and write offsets are automatic and wrap around by themselves. The wrap around function must be checked, since an offset of zero corresponds to the pointer byte. The FIFO name and symbolic address is RXFIFO. RXFIFO is also the address of the pointer byte, which is laid out as follows.

* FULL * N.U. * READ OFFSET * WRITE OFFSET *

READ OFFSET and WRITE OFFSET occupy three bit fields. The FULL flag has one bit. Bit six is not used.

Data storage is accommodated in locations RXFIFO + 1 to RXFIFO + 7.

The full flag being set means no more data can be written into the FIFO. If the FULL flag is not set, then the condition of the read offset equalling the write offset means that the FIFO is empty. When the write offset equals the read offset after a write, a full condition is declared and the FULL flag is set.

```
*
*WRITE THE FIFO
*
RXFIFO    EQU     0'30'     BASE LOCATION OF FIFO
RXFIFU    EQU     3         UPPER HALF OF RXFIFO
DATA      EQU     6         TRANSFER DATA LOCATION
*         THIS ROUTINE DESTROYS THE ISAR
*
WRFIFO    LI      RXFIFO    PUT FIFO ADDRESS IN ISAR
          LR      IS,A
          CLR               TEST TO SEE IF FULL
          XS      S
          BP      WRF1      BRANCH IF NOT FULL
          *
          POP               RETURN FROM SUBROUTINE
WRF1      LR      IS,A      LOAD WRITE OFFSET INTO LOWER ISAR
          LISU    RXFIFO    UPPER HALF OF FIFO ADDRESS
          LR      A,DATA    PICK UP DATA TO WRITE
          LR      A,I       WRITE AND INCREMENT ISAR
          *
          LR      A,IS      SEE IF WRAP TO 0
          NI      7         JUST LOWER PART
          BNZ     WRF2      SKIP IF NO WRAP

          LIS     1         SET TO 1 IF WRAP
WRF2      LR      DATA,A    STORE TEMPORARILY
          LI      RXFIFO    COMPARE RD AND WRT
          LR      IS,A      PICK UP WHOLE POINTER
          LR      A,S
          SL      1         ADJUST WRT OVER READ
          SR      4         SHIFT RT THREE
          XS      DATA      COMPARE
          BZ      WRF3      FULL IF EQUAL
          *
          LR      A,S       UPDATE THE POINTER
          NI      0'70'     ISOLATE READ OFFSET
          XS      DATA      MERGE IT WITH WRITE OFFSET
          LR      S,A       PUT POINTER BACK
          POP               THAT IS ALL
          *
WRF3      LR      A,S       MARK THE FLAG FULL
          NI      0'70'     ISOLATE READ OFFSET
```

```
                OI          H'80'       MARK THE POINTER
                XS          DATA        ADD THE WRT OFFST
                LR          S,A         PUT THE POINTER BACK
                NI          0           STATUS TO POSITIVE
                POP
                *
                *

*
*READ THE FIFO
*
GP0             EQU         0           GP REG 0
*               THIS ROUTINE DOES NOT RESTORE THE CONTENTS OF THE ISAR
*
RDFIFO          LI          RXFIFO      PUT FIFO ADDRESS IN ISAR
                LR          IS,A
                CLR                     TEST TO SEE IF FULL
                XS          S           FULL MEANS NOT MT
                BM          RDF1        BRANCH IF FULL
                *
                LR          GP0,A       STORE POINTER HERE TEMPORARILY
                SL          1           ALIGN RD OVER WRT
                SR          4           SHIFT RIGHT 3
                XS          GP0         COMPARE RD=WRT?
                NI          7           JUST THE FIELD IN QUESTION
                BNZ         RDF1        BRANCH IF NOT EQUAL
                *
                OI          H'80'       FORCE NEG STATUS FOR RETURN
                POP                     RTURN
                *
RDF1            LR          A,S         GET THE POINTER
                SL          1           ALIGN TO LOAD READ OFFSET
                SR          4             INTO ISAR
                LR          IS,A
                LISU        RXFIFO      FIX UPPER PART
                LR          A,I         READ AND INCREMENT
                LR          DATA,A      DATA TRANSFER REG
                NI          7           ZERO IF YES
                BNZ         RDF2        BRANCH AROUND IF WRAPPED
                *
                LIS         1           1ST AVAIL CELL IN FIFO
RDF2            SL          4           ALIGN IT WHERE IT GOES
                SR          1
                LR          GP0,A       TEMP STORE READ OFFSET
                *
                LI          RXFIFO      POINT TO THE POINTER
                LR          IS,A
                LR,A,S      READ POINTER
                NI          7           GET JUST THE WRT PART
                XS          GP0         MIX WITH READ OFFSET
                LR          S,A         UPDATE THE POINTER
                *           FULL FLAG UNCONDITIONALLY CLEARED
                POP                     STATUS NON-NEG FROM MIX
                *
```

ERROR HANDLING

The serial port does not provide much information about errors, except to set a flag on receiver overrun and transmitter underrun. Other errors generally of interest in a serial communication port are parity, framing errors, sometimes CRC errors. Parity is the modulo two sum of the ones in a data character. Framing errors are when the stop bits do not happen when they should. And CRC is a cyclic redundancy code that is similar to a check sum taken over a block of data characters.

For systems where it is essential that data is verified and acknowledged, protocols are established setting forth rules for the generating and checking of parity or CRC, and for the way acknowledgements are made. Also important is the recovery strategy in the event an error occurs. Typically, a block of data in error is simply retransmitted. Most protocols have the transmitter send data in a block protected by a CRC at the end. When the receiver detects the end of the block, if the CRC checked, a positive acknowledgement is made. If the CRC did not check, a negative acknowledgement is sent and the transmitter retransmits the data.

Where simple parity is used, it is difficult to take recovery action other than to merely throw away the character that had the bad parity. This is often the approach taken in CRT terminals where the data is displayed as a # if parity on it failed. This method is also used by some telegraph services. Since the MK3873 serial port does not support hardware parity or framing error detection, this must be done in software.

Parity may be computed in software by shifting the data through the carry or sign bit, using the results of each shift in a conditional branch to a short piece of code that complements a parity bit. The bit thus computed could be attached to the data and transmitted. Checking would use the same routine.

Framing error detection could be accomplished with a simple match on the stop bits after receiving each character. With two stop bits, the stop bits always land in port 0EH bits six and seven independent of word length. A straightforward routine could be called that did an input of port 0EH, tested for negative sign, shifted left one, and tested for sign again. If either test produced a positive sign, an error could be declared.

CRC is a little more difficult. This can be handled, usually, by a shift algorithm that does an exclusive "OR". CRC routines normally take up quite a bit of real time, and can only be implemented if the CPU has little else to do, and the Baud rate is low. Sometimes, where CRC is needed at Baud rates over about 4800, an off-chip CRC generator is the best approach. Most async systems fortunately do not require the use of CRC. Often, if block verification is desired, a modulo-256 checksum provides adequate error detection. Checksum algorithms are easily implemented in firmware.

SUMMARY AND CONCLUSIONS

The MK3873 is a very useful member of the 3870 family. The serial port gives it the ability to extend its I/O far beyond the range of the parallel ports. It is easy to imagine many, many ways the MK3873 could be configured with I/O devices and with other MK3873s to accomplish very sophisticated control and networking functions.

The MK3873 serial port eases the complexity normally associated with communication interface designs, especially since special care has been taken to facilitate both sync and async cases. Having the serial port control in hardware makes the task of attending the port very simple for the firmware.

The MK3873 should prove useful as a highly intelligent control component in such applications as portable terminals, printer controllers, printer interfaces, and many, many more.

It has been illustrated that data rates of as high as 9600 Baud in asynchronous mode may be accommodated while having a large percentage of the MK3873's total processing power still available for other system tasks and operations.

MOSTEK®

MULTI-LEVEL SUBROUTINE HANDLING OF F8 AND MK3870 FAMILY OF MICRO COMPUTERS

Application Note

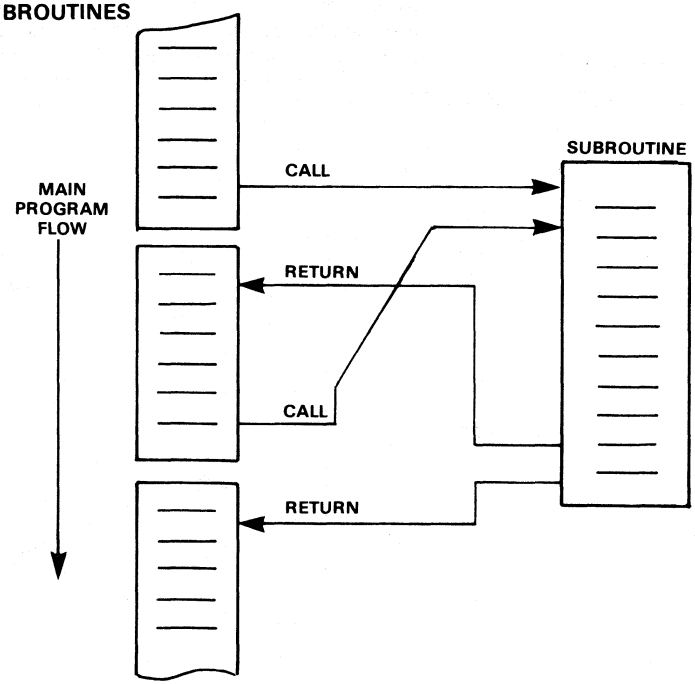
INTRODUCTION

The 3870 and F8 Microcomputer Families have become recognized as a cost effective method of placing computing power into types of equipment which could not have justified the cost of computer control just a few years ago. The sharply falling cost per computer function afforded by advances in Metal Oxide Semiconductor-Large Scale Integration, (MOS-LSI), has brought computer technology and techniques into areas where until now, mechanical controllers, random logic, and relay logic predominated. The availability of a large number of Input/Output pins in the MK3870 Family of Microcomputers, coupled with its minimum system configuration requirements of just one device, make the 3870 series ideal replacements for many previously used control devices. The purpose of this note is to discuss the use and implementation of subroutines and interrupts as they apply to programming an F8 or 3870 based microcomputer system. The intent of this note is to describe the use of subroutines and interrupts for the hardware designer who might not be totally familiar with the programming of a computer, and also for the firmware designer who might not be totally familiar with the F8 or 3870 architecture.

SUBROUTINES

A subroutine is a sequence of computer instructions which can be called upon to execute a function from many different parts of the program. The purpose of a subroutine is to reduce the total length of a computer program by consolidating in one portion of the program a sequence of instructions that are used over and over again by different sections of the program. When this subroutine is required, the program counter contents are replaced with the starting address of the subroutine. At the end of the subroutine, the original program counter contents are restored and execution continues back in the main body of the program. Figure 1 depicts program execution flow when using subroutines. The main program calls a subroutine which

PROGRAM FLOW WHEN USING SUBROUTINES
Figure 1



causes the program counter to be loaded with the address of the subroutine. The calling action causes the return address to be saved. The last statement in the subroutine causes a return to the main program flow by retrieving the saved program counter value, forcing a return to the main program. The subroutine is called again any place in the main program flow where the sequence of instruction contained in the subroutine is required. Every time the subroutine is called, a savings in program length (and ROM size) equal to the length of the subroutine (minus three bytes of calling overhead) is realized compared to a program which does not use subroutines. Many times a subroutine will call another subroutine resulting in what is referred to as nested or multi-level subroutines.

Interrupts are a form of spontaneous subroutine calls and bear many similarities to multi-level subroutine situations, which make up the central topic of this application note.

INTERRUPTS

Interrupts are used in a microcomputer system to make it responsive to the devices it is controlling. By interrupting the microcomputer, the I/O device can signal its requirement for attention or service by the microcomputer. As in the case of the subroutine, the interrupt can divert the main program flow to a sequence of instructions called the Interrupt Service Routine (see Figure 2). This routine
may input or output data to the device being controlled. At the end of this service routine the program counter value at the time of the system interrupt is retrieved from a temporary register where it was stored and reloaded into the program counter to cause a return to the main program flow. Interrupts, like subroutines, can be made multi-level or nested to enable an Interrupt Service Routine to be interrupted by a higher priority device. Likewise, it may be desirable for an Interrupt Service Routine to call a subroutine, producing a situation very similar to nesting.

SUBROUTINE RELATED INSTRUCTIONS

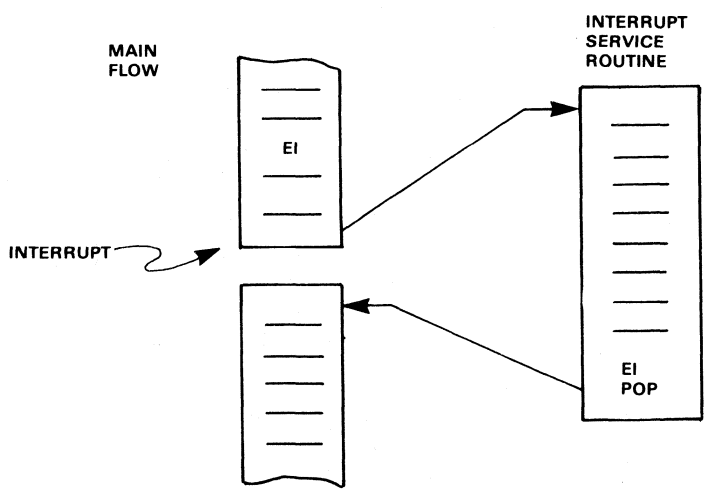
The F8 or 3870 instructions which are used to transfer program flow to or from subroutines or interrupts are illustrated in Figure 3. The Program Counter (PO) holds the address of the next instruction to be executed by the microcomputer, while the Stack Register (P) is a temporary storage location for the Program Counter. In addition, two pairs of registers in the Scratchpad have been designated K and Q with instructions that link them to PO and P. The instructions that link and affect these registers are the following:

- PI ASUB — Call subroutine ASUB
- PK — Return or call subroutine through K
- POP — Return from subroutine through Stack P
- LR P,K — Move K to Stack P
- LR K,P — Save Stack P in reg. K
- LR PO,Q — Return from subroutine through Q

PI ASUB This is an immediate call to the subroutine ASUB. When the PI operation code is encountered by the F8 or 3870 processor the next two bytes, namely ASUB, are loaded into the Program Counter PO in order to transfer control to the subroutine. The old contents of PO (the return address) are saved in the Stack P.

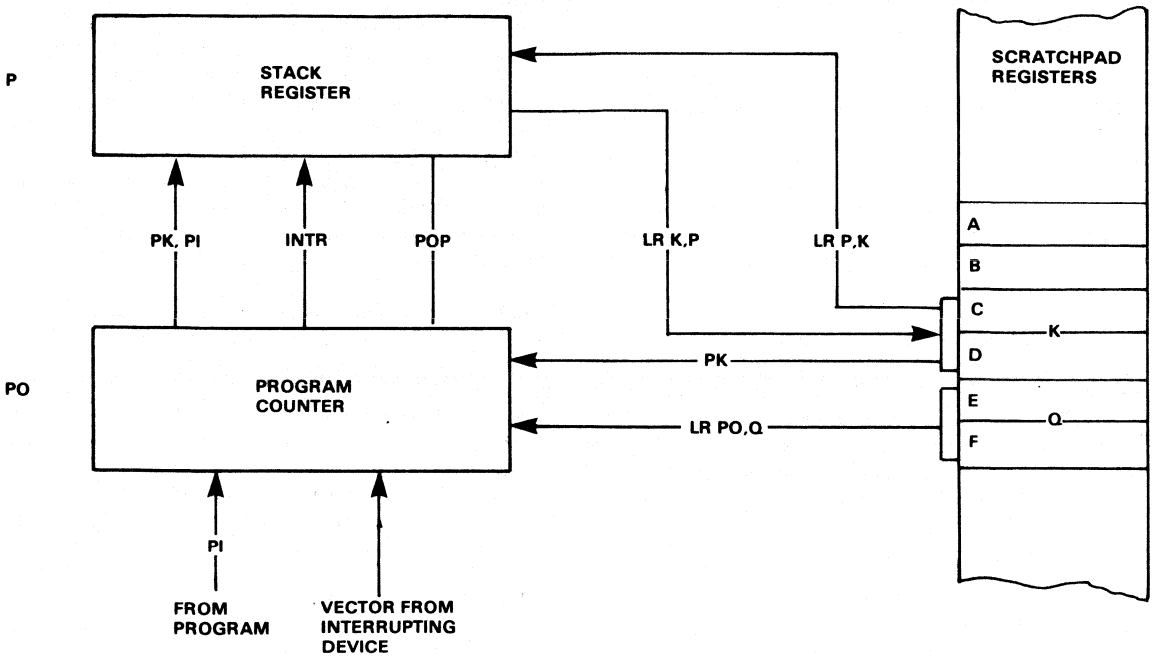
PK This instruction can be used in two different ways. First, PK is a return from subroutine via the K register, since the contents of the K are placed into the Program Counter PO. Secondly, PK can be used to make an indirect subroutine call to the address loaded in the K, since the old contents of the PO are saved in the Stack P.

PROGRAM FLOW WHEN INTERRUPTED
Figure 2



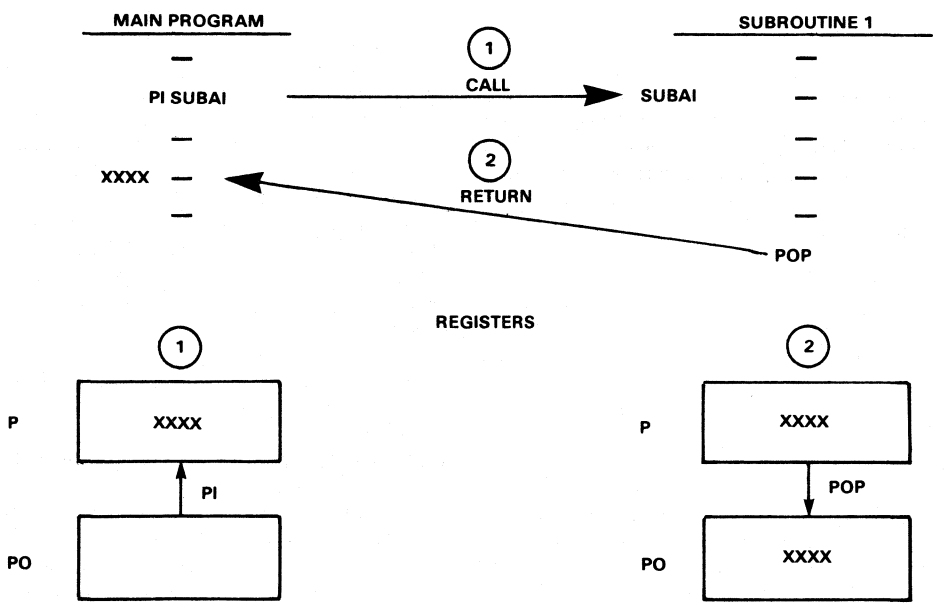
F8 OR 3870 REGISTERS USED IN SUBROUTINES AND INTERRUPTS

Figure 3



ONE LEVEL SUBROUTINES OR INTERRUPTS

Figure 4



POP This is the most common return from subroutine and is also used for a return from interrupt. The POP instruction places the contents of the Stack Register P into the Program Counter PO.

LR K,P and LR P,K These two load instructions are used to move saved addresses between the Stack Register P and the K register.
This is essential in multi-level subroutine handling.

LR PO,Q This load instruction performs a return from subroutine through the Q register. This one cannot be used as a subroutine call since the old program counter contents are not saved.

An instruction is said to be "privileged" if interrupts are not sampled at the end of its execution. Interrupts are usually sampled by the processor at the end of each instruction and, if set, the processor initiates an interrupt cycle. Privileges are given to certain instructions to hold off the occurrences of interrupts to protect code sequences and insure that context can be saved before return information is lost.

The F8 and 3870 Family privilege the following set of instructions:

- JMP ADDR	Jump to ADDR
- PK	Return through K
- PI ASUB	Call subroutine ASUB
- POP	Return through Stack P
- EI	Enable system interrupts
- LR W,J	Load status from J register
- OUT (S) PORT	Output (Short) to port PORT

APPLICATIONS

It is one thing to arrange for one subroutine to call another by moving addresses around. It is another thing for these subroutines to be calling and called while the possibility of interrupts exists. Both multi-level subroutine handling and interruptable subroutine handling are discussed in detail in the following paragraphs.

INTERRUPTABLE SUBROUTINES

For a subroutine to be interruptable means simply that an interrupt can come along at any time, forcing an equivalent subroutine call, overwrite the Stack P and nothing will be lost. This imposes some constraints on both the Interrupt Service Routine and the subroutine calling protocol.

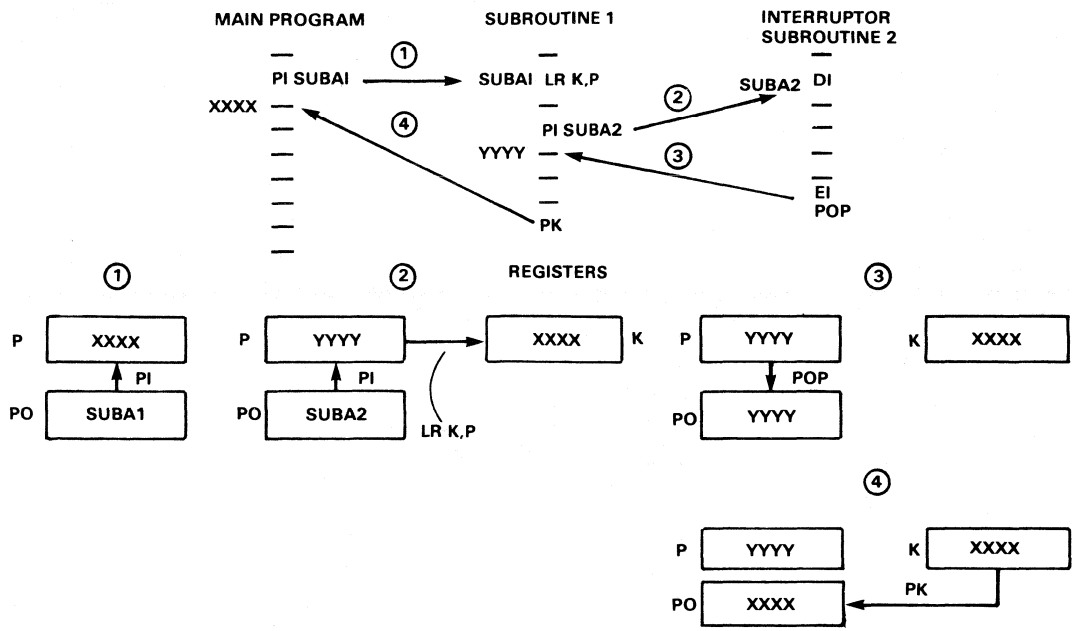
Many applications can be handled by two levels of subroutine, or one level of interruptable subroutines. This implies that only two return addresses need to be saved, which can be handled easily by registers within the F8 or 3870 for this purpose. The calling of subroutines is under the control of the programmer and thus only the return addresses need be saved as other registers (such as the Data Counter) can either be saved by the calling or the called routines if the registers are needed by the subroutine. Interrupts are under control of the programmer only to the extent that they can be masked or enabled. Assuming that interrupts are enabled upon entry to the Interrupt Service Routine, it may not be known which registers in the CPU contain data which cannot be overwritten. In this case, these registers should be stored in scratchpad during the Interrupt Service Routine and restored before exiting this routine. Examples of using the ISAR to store CPU registers in a push down stack are given in this note but in many cases the programmer will tailor the status saving routine for the specific circumstances of his system design (by using specific Scratchpad registers to save CPU registers).

Figure 4 shows the instructions usually used to call a subroutine (one level deep)in an F8 or 3870 system. SUBA1 is the symbolic name of the two byte address of the subroutine and PI causes the return address (XXXX) to be saved in P. POP reverses the procedure at the end of the subroutine causing PO to be reloaded with the address saved in P and the program flow to return to the next instruction in the main program (XXXX). Response to an interrupt from the main program is similar to this example except that the interrupt causes a path similar to 1 to the Interrupt Service Routine with the address (vector) being supplied by the interrupting circuitry and loaded into PO.

To call a second subroutine or to respond to an interrupt from SUBA1 the instuctions in Figure 5 could be used. In this case, PI SUBA1 transfers the program flow to SUBA1 while saving the return address (XXXX) in the Stack register P. Subroutine 1 now transfers P to K in preparation for another subroutine call or an interrupt (note that if an interrupt occurs during the PI SUBA1, LR K,P sequence it will not be serviced until after the LR K,P instruction because the PI SUBA1 is a privileged instruction). Subroutine 2 is called by PI SUBA2 which saves YYYY in P which was just vacated. Program flow transfers to Subroutine 2 and the POP instruction reloads PO with YYYY from P. At the end of Subroutine 1, the return address is now in K so a PK is used to load XXXX into PO, thereby returning to the main program. (Note that Subroutine 2 must contain a DI instruction at the starting address to reserve the exclusive use of the Stack register while it executes. The EI at the end of Subroutine is privileged and insures the safe return of the program flow even if an interrupt hits).

Interrupts can occur any time, and they will divert program flow out of the main program or out of the subroutines similar to Subroutine 1, which uses the K register for their return address and are thus interruptable. Subroutines like Subroutine 2 are uninterruptable by virtue of their DI instructions and the fact that they require the Stack P for their return address storage. If the main program is considered for this discussion to be of class A, and subroutines like Subroutine 1 are considered class B, then class C would refer to subroutines like Subroutine 2 and would include interrupts. Class A routines can call class B or class C routines. Class B routines can call class C routines.

TWO LEVEL SUBROUTINES OR INTERRUPTS
Figure 5



Class C routines cannot call any other routines. Classes A and B are interruptable. Interrupts, being class C routines, cannot call subroutines. As can be seen by this analysis, special provisions must be made if it is required that Interrupt Service Routines be allowed to call subroutines (see discussion on push down stacks).

THREE LEVEL SYSTEM

By using the Q register in addition to the K and the P to store return addresses, a three-deep system of subroutine nesting can be realized. Figure 6 shows programming with three levels. For this discussion the first level subroutine will be referred to as an A-level subroutine, the second is a B-level subroutine, and the third level is a C-level routine. The main program can call any level subroutine. All subroutines are called using the PI ASUB instruction which, remember, is privileged. A-level subroutines can call B-level or C-level subroutines. B-level subroutines can call C-level subroutines. C-level subroutines cannot call any subroutines. The main program, A-level subroutines, and B-level subroutines are interruptable. C-level subroutines are not interruptable and execute with interrupts off.

Programs and routines do not have to know what level a routine they are about to call is except that the called routine must be of a lessor level than the calling routine, where the levels are related by the expression MAIN>A>B>C. Interrupts are equivalent to C-level.

The A-level subroutine entry could be easily made a Macro that expands into the following instructions:

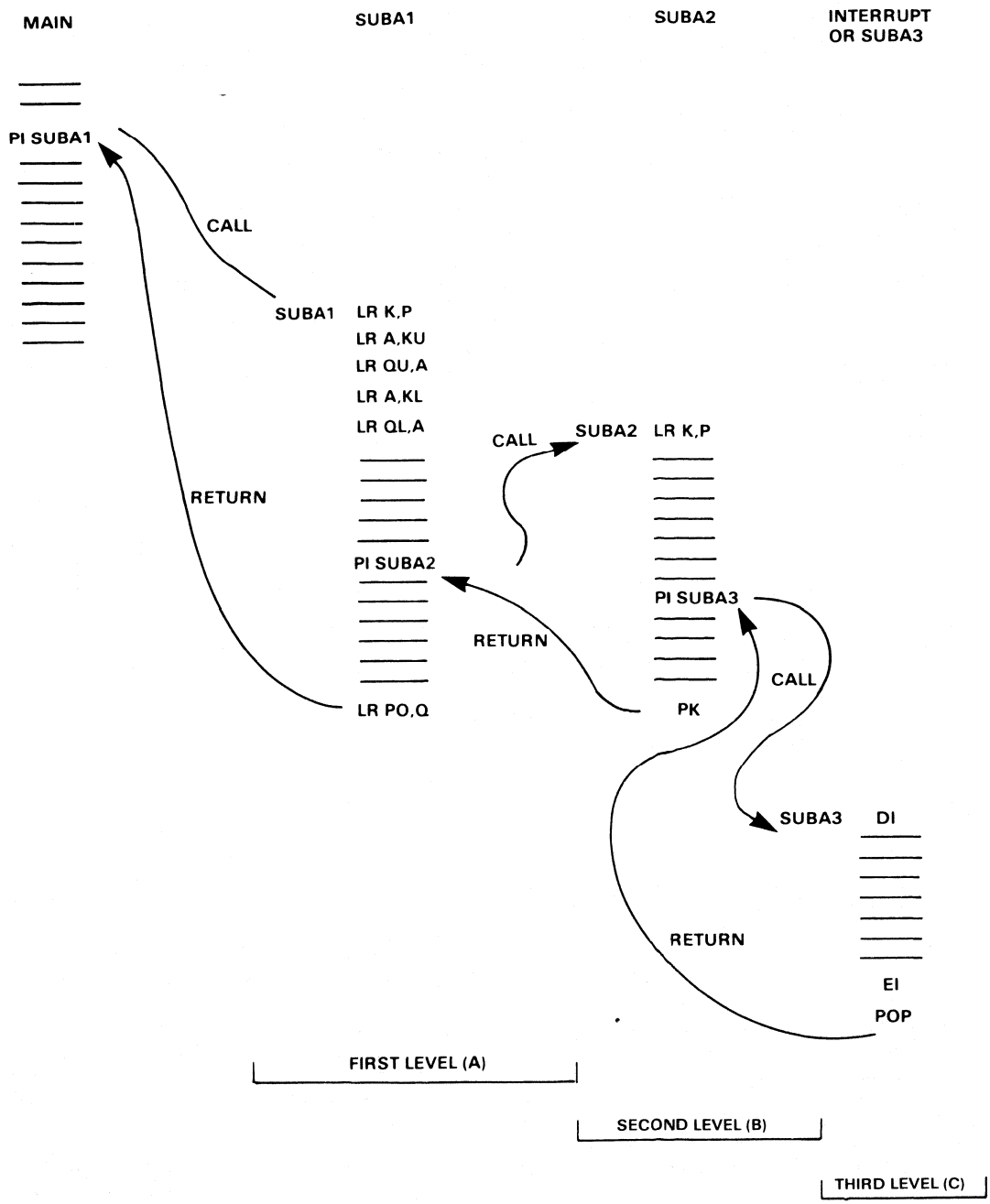
```
*
*          A-LEVEL SUBROUTINE ENTRY CODE
*
AENT    LR          K,P
        LR          A,KU
        LR          QU,A
        LR          A,KL
        LR          QL,A
```

These five instructions take up five bytes and execute in sixteen microseconds when the F8 or 3870 is run with an internal clock rate of 2 MHz. The LR K,P instruction is the critical one where it comes to saving the return address. This instruction must be executed as the first instruction in the A-level subroutine so that it will be protected by the PI ASUB that called it. There is no need to disable interrupts during the A-level call. The return is made via the LR PO,Q instruction.

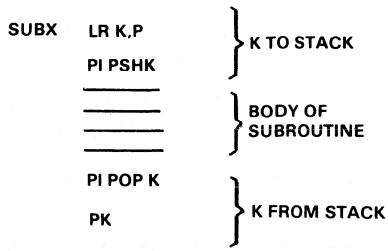
The B-level entry is a simple LR K,P instruction that is protected by its call. As with the A-level subroutine, interrupts are not a threat. The return from the B-level subroutine is simply a PK.

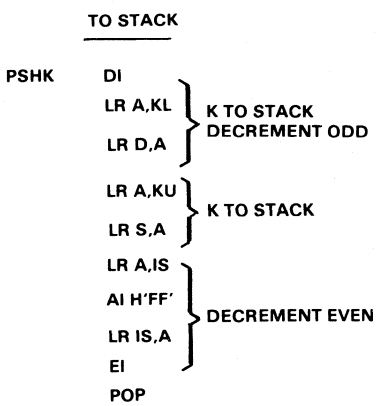
C-level routines require a DI instruction as their entry point. It is essential that C-level routines do not try to call any other

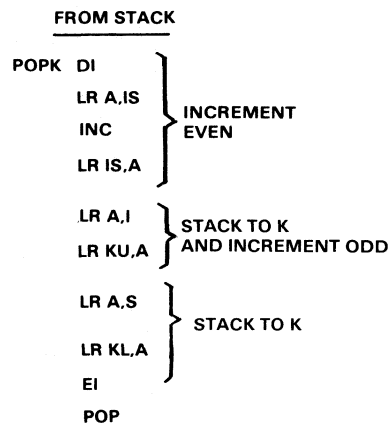
THREE LEVELS OF SUBROUTINES OR INTERRUPTS
Figure 6

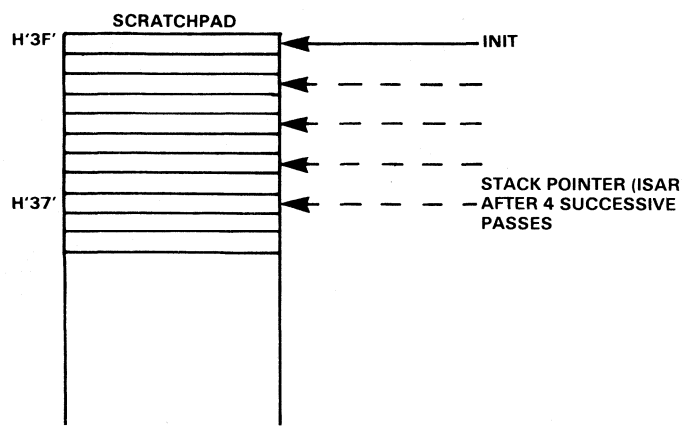


STACKING OF K REGISTER
Figure 7



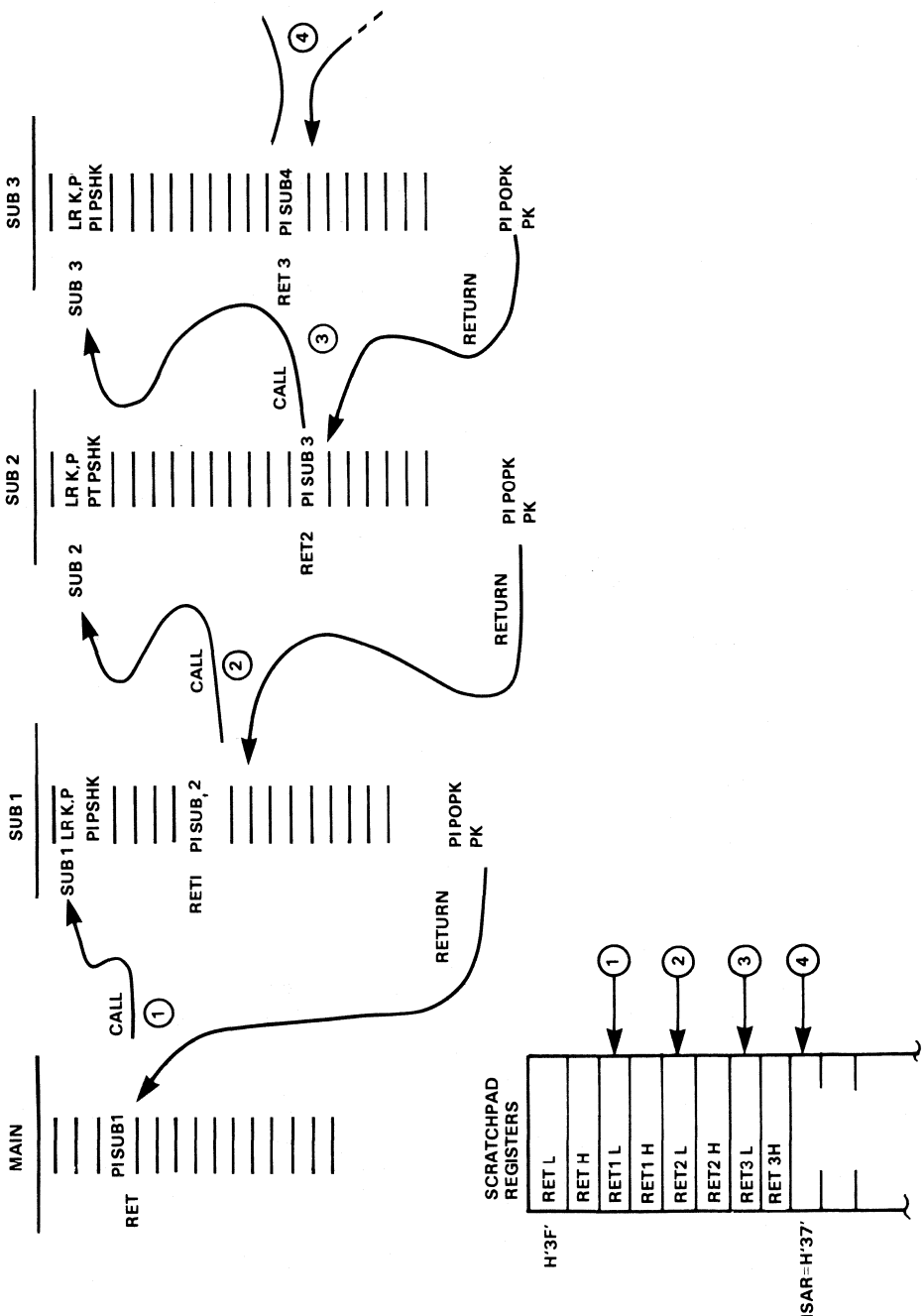






N-LEVEL STACK NESTING OF SUBROUTINES

Figure 8



subroutines or enable the system interrupts, except to enable interrupts as they return to their caller.

The return sequence should be as follows:

```
*
*          C-LEVEL RETURN SEQUENCE
*
CRET       EI
           POP
*
```

This completes the discussion on a dedicated register approach to return address storage. The three-deep system thus described matches most uses in the distribution of the overhead since the normal system would probably have a central MAIN calling loop that makes A-level subroutine calls to major subsystems. These A-level subroutines would then make numerous calls to B-level subroutines which would do the bulk of the repetitive work. Occasionally a B-level might have to call a C-level I/O driver type of subroutine, roughly parallel in complexity to an Interrupt Service Routine. The system thus described allows interrupts to remain turned on throughout all but the lowest level of operation, thus maintaining a high degree of responsiveness to the interrupt driven I/O, timers, etc.

The paragraphs to follow discuss a stacked arrangement to store return addresses. This uses the highest overhead of all the methods, but allows the use of subroutine calls by the Interrupt Service Routines, and permits the accommodation of multi-level interrupts to some extent.

N-LEVEL NESTING

At a minimum, when using the F8 or 3870 in a system with greater than three levels of interrupts or subroutines, a consistent method of placing return addresses into the scratchpad must be used to allow their recovery. In many cases it will be desirable to stack more registers than just the return addresses. Previous examples have shown three levels deep with the three return addresses in P, K, and Q registers. Any further nesting would either destroy P, K, or Q so the technique to be described is to move K into the scratchpad to make room for another level.

Figure 7 shows a generalized subroutine which first copies its return address into K with a LR K,P instruction. Then it calls the PSHK routine to stack the address. The PSHK routine shown in Figure 7 assumes that the ISAR (Indirect Scratchpad Address Register) has been initialized at an odd value, probably H'3F' which is the top of the scratchpad area. The odd starting value is required to insure that the ISAR is not pointing to an 8-byte boundary when the LR D,A instruction is executed. (The LR D,A instruction loads the accumulator from the scratchpad location pointed to by ISAR and then does a modulo 8 decrement of ISAR. This means that only the lower three bits of ISAR are decremented resulting in an 8 byte range for these auto incrementing and auto decrementing instructions which does not allow crossing of page boundaries. By intializing ISAR at on odd value, every time the LD D,A instruction is executed ISAR will be odd and therefore will not have to cross page boundaries which are even.) The decrementing of the ISAR from even values is accomplished by loading ISAR into the accumulator and adding hexadecimal FF to it, which results in an 8 bit decrement of the contents of the ISAR. PSHK then moves the contents of the K onto the stack and leaves ISAR pointing to the next empty location thus implementing a push down stack. When in the body of the subroutine, both the P and the K are available for further subroutine calling or for interrupts. POPK is called to undo the stacking done in the PSHK subroutine. Since POPK only pops the return address off the stack and leaves it in the K register, it is only necessary to execute a PK in order to return to the caller.

Notice that both the PSHK and the POPK subroutines utilize the P register and therefore the entry to these routines contains a DI to disable interrupts. Upon return from these routines, an EI is used.

*N-LEVEL NESTING WITH EXECUTABLE RAM

The executable RAM which is optionally available in the 3870 series can be used to implement a stack, the data counter then serves as a stack pointer.

A suitable push routine which assumes the accumulator is available and operates with interrupts off is shown in Figure 9. This routine assumes that the data counter points to the next available location on the stack.

The complimentary POP routine takes advantage of the sign extension of the accumulator in the ADC instruction. See Figure 9. The POP routine exits with the data counter pointing to the next available location on the stack. These routines require that the data counter be initialized before use as a stack pointer and they do not include stack over/underflow checking, in other words, pushes must equal POPs and no more than available RAM can be used safely.

Figure 9

```
Push K      DI
            LR      A, KU
            ST                      DC + 1 → DC
            LR      A, KL
            ST                      DC + 1 → DC
            EI
            POP

POP K       DI
            LIS                     1  Create a H - 2
            COM                        Accumulator
            ADC                     DC - 2 → DC
            LM                      DC + 1 → DC
            LR      KU, A
```

LM		DC + 1 → DC
LR	KL, A	
LIS	1	
COM		
ADC		DC - 2 → DC
EI		
POP		

CONCLUSIONS

This application note has discussed several methods of handling subroutines in the face of and including interrupts in F8 or 3870 systems. Many applications for which the F8 or 3870 is suited will have minimal subroutines of a minimum number of interrupts so that the internal P, K, and Q registers can be used exclusively for holding return addresses.

In the cases where deeply nested subroutines or multi-level interrupts must be handled, a push-down stack implementation can be used by taking space in the upper scratchpad and typing up the ISAR. Care should be exercised to ensure that the ISAR is not in use when these stacked calls are made.

The routines used to push return address pointers may as well be used to stack any data or other information for use in passing parameters to subroutines, etc. While the F8 or 3870 architecture is not a stack architecture machine, stacks can be implemented in it for whatever use there might be for them.

MOSTEK®

FULL DUPLEX OPERATION OF THE 3873 SERIAL PORT

Application Note

INTRODUCTION

Mostek's single-chip microcomputer family had its start with the MK3870. The family has grown to include several new members, with versatile options for amounts of on-chip RAM and ROM. Some new features available include low-power mode, serial port, and, "Piggy-Back"™ EPROM.

This application note explores some of the ways of using the MK3873, a single-chip microcomputer family member with a serial port. The MK3873 has features that characterize the family, which include:

- 1K or 2K bytes of mask programmable ROM
- Software compatible with 3870 instruction set
- 64 byte scratchpad RAM
- Available with 64 byte executable RAM
- 29 bits (4 ports) TTL compatible parallel I/O
- Vectored interrupts
- External interrupt
- Multi-mode timer

The MK3873 is equipped with a serial port which has been integrated into the chip's normal I/O space. Figure 1 shows the system block diagram for the MK3873, illustrating the serial port logic architecturally. The serial port is accessed through regular I/O instructions. There are no new instructions required for driving the serial port.

The serial port is a USART, or Universal Synchronous/Asynchronous Receiver/Transmitter. It has holding registers for both transmit and receive, which require CPU service only on a character-at-a-time basis. The serial port can interrupt the CPU when it needs attention.

DESCRIPTION OF THE SERIAL PORT

Figure 2 shows a simplified view of the serial port hardware. The blocks labelled SI, SO, and SRCLK represent connections to the outside of the chip to terminals. The mnemonics stand for Serial Input, Serial Output, and Serial Clock, respectively. Serially received data is routed from SI to two locations within the serial port: the Start/Stop Synchronizer, and the Data Serialization sections. The Start/Stop Synchronizer is used to synchronize the internal clock divider circuitry with the leading edge of data in the async mode. It is disabled in the sync mode. The Data Serialization section is used for converting the form of the data both from serial to parallel, for input, and from parallel to serial, for output.

The internal shift clock, derived either from the internal Baud Rate Divider, or from an externally supplied Serial Clock, is used to shift the Shift Register and also to decrement the Word Length Counter to determine data framing.

There are two serial port status flags, designated READY and ERROR. READY is an indication that a data frame, as determined by the word length specification, had either been shifted in from SI, or shifted out to SO. The format of the frame, or character, is also determined in part by the mode of transmission, sync or async. The ERROR flag is set whenever two successive frames are shifted without an intervening read or write of the holding registers by the CPU. The interrupt request to the CPU is set, if enabled, by the same signal that sets READY, and may be used to prompt the CPU to service the holding registers.

Figure 3 details the four I/O ports that are used for accessing the serial port. All manipulation of the serial port is handled through inputs and outputs of the four ports whose addresses are C, D, E and F.

The mnemonics used in Figure 3 are defined as follows:

BAUD RATE CODE. Sets the Baud Rate Divider constant, resulting in shift rates as listed in the table. A zero value for BAUD RATE CODE switches SRCLK from output to input.

WORD LENGTH CODE. Sets the total number of bits in a data frame as listed in the figure. Length includes start, parity, and stop bits, if used, along with the data bits.

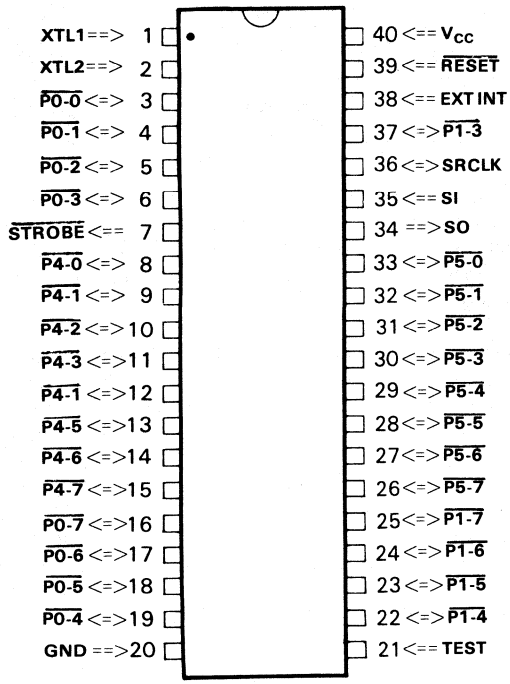
EDGE. Set to put serial port into a mode to enable async receive operation. Enables Start/Stop Synchronizer. Used only in async receive mode.

SEARCH. Set to effectively override the word length setting to one bit. Causes end-of word condition on every bit shifted, resulting in READY, ERROR and interrupt activation. Used only in sync mode while searching the incoming data stream for the sync character.

SYNC. Set to put the serial port in sync mode. Reset to put serial port to async mode. When SYNC is set, the serial port shifts at the same rate as SRCLK. When reset, serial port shifts at one-sixteenth the rate of SRCLK.

XMIT. Set to enable the output to pin SO. Also enables loading of the shift register from the transmit holding

PIN CONNECTIONS



PIN DESCRIPTIONS

$\overline{P0\text{-}0}$ - $\overline{P0\text{-}7}$	Bi-Directional Parallel I/O Port 0
$\overline{P1\text{-}3}$ - $\overline{P1\text{-}7}$	Bi-Directional Prallel I/O Port 1
$\overline{P4\text{-}0}$ - $\overline{P4\text{-}7}$	Bi-Directional Parallel I/O Port 4
$\overline{P5\text{-}0}$ - $\overline{P5\text{-}7}$	Bi-Directional Parallel I/O Port 5
SI	Serial Port Receive Data Input
SO	Serial Port Transmit Data Output
SRCLK	Serial Clock Output * or Input
$\overline{STROBE}$	Port 4 Strobe Output *
$\overline{RESET}$	Chip Reset Input
EXT INT	External Interrupt Input
XTL1, XTL2	Time Base Inputs

*NOTE: These outputs are capable of driving up to 3 TTL loads.

register when end-of-word condition is reached. Sets serial port interrupt vector to E0 (hex). Reset puts interrupt vector to 60 (hex).

INTS. Set to enable serial port interrupts when system interrupts are enabled.

FULL DUPLEX CONSIDERATIONS

In the sync, full-duplex mode, the SYNC and XMIT bits in port D are set. This enables loading the Shift Register from the Transmit Holding Register, and sets the interrupt vector to E0 (hex). It also enables the output, SO, continuously, starting with the first end-of-word condition.

The serial port shifts the transmit character out of SO at the same time as the receive character is being shifted in SI. When the end-of-word condition occurs, it is signalling the CPU that it has just loaded the Receive Holding Register with new data, and that it has just loaded the shift register with what was in the Transmit Holding register. To annunciate this fact, the READY flag becomes set, and the interrupt request is sent to the CPU. If the READY flag was previously set and had not been cleared with a holding register read or write, then the ERROR flag would be set.

A single interrupt announces that both the Receive Holding Register and the Transmit Holding Register require service. Likewise, the single READY flag represents the readiness of both the receive and the transmit holding registers. The READY flag also serves as the serial port's indication of whether or not it should set the ERROR flag, since if the READY flag has not been cleared by the next end-of-word, ERROR is set. The READY flag is reset by either a read or a write of either holding register. Thus, it is not possible for the serial port to recognize whether an ERROR flag represents a receiver overrun or a transmitter underrun.

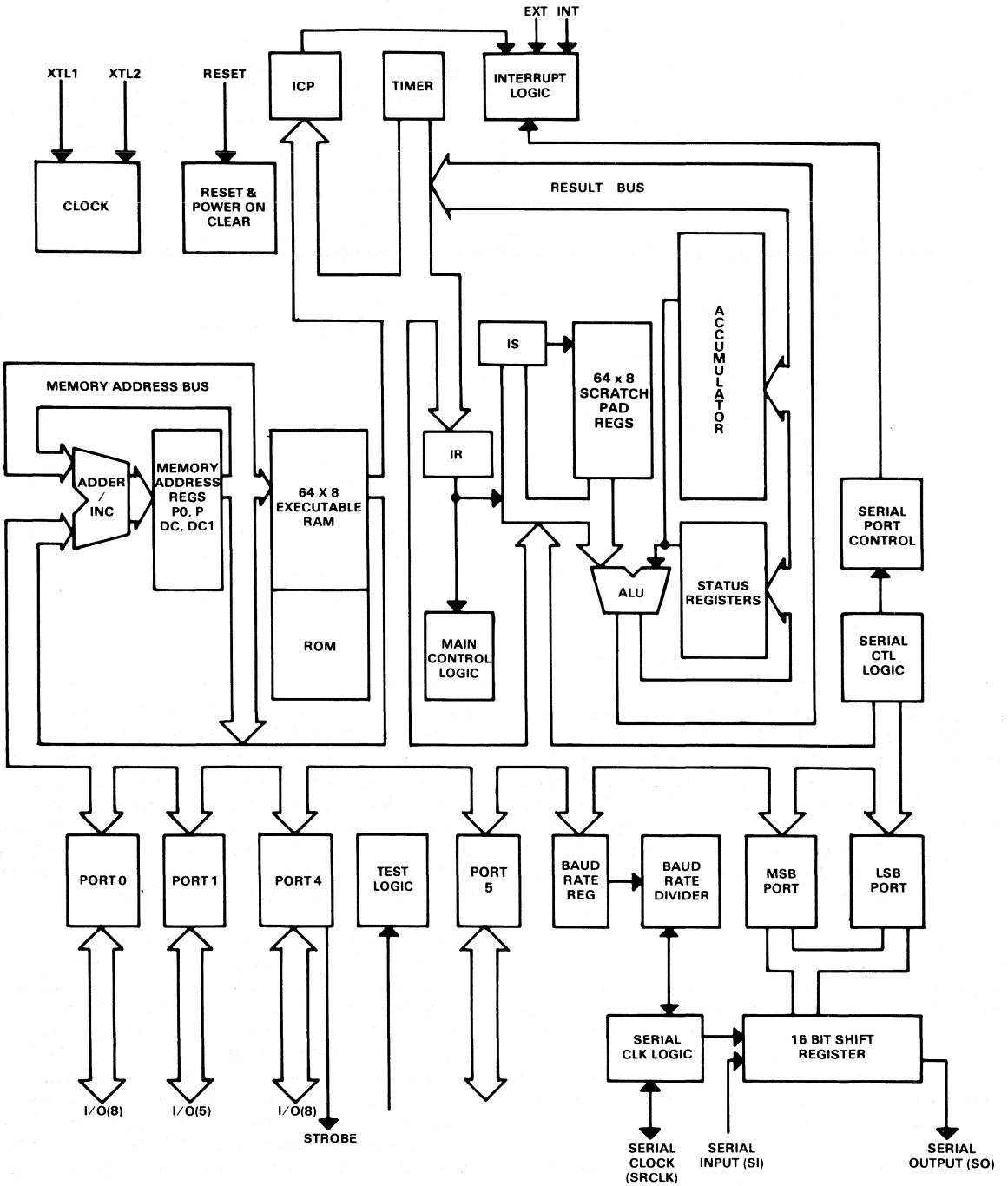
Full-Duplex transmission implies that when the interrupt from the serial port occurs, data must be read from the Receive Holding Register and new data must be written to the Transmit Holding Register. Getting the process started requires some protocol. When the communication channel is first opened, it is necessary to have defined whether the station will lead or be led on the channel. That is, the station must be defined as a primary or a secondary station. The primary station opens the channel by going directly to full-duplex transmit mode, sending sync patterns to allow the secondary station to attain character synchronization. The secondary station opens the channel by first entering SEARCH mode to attain character sync, and then entering full-duplex transmit mode sending sync patterns. The primary station detects the fact that the secondary has entered transmit mode when sync patterns begin arriving. At this point, the full-duplex communication link has been established, and data transfer may commence.

The hardware in the serial port automatically loads the contents of the Transmit Holding Register into the Shift Register, whether or not new data was loaded into the Transmit Holding Register. Similarly, the hardware automatically writes the Shift Register contents over the previous contents of the Receive Holding Register, whether or not the old information was read by the CPU. It is, therefore, necessary to service both holding registers on each interrupt to avoid errors in transmission. In some circumstances, it is beneficial to load the Transmit Holding Register with the sync pattern and let the port send syncs repeatedly, without the need to write into it further.

Figure 4 illustrates the hardware connections required for

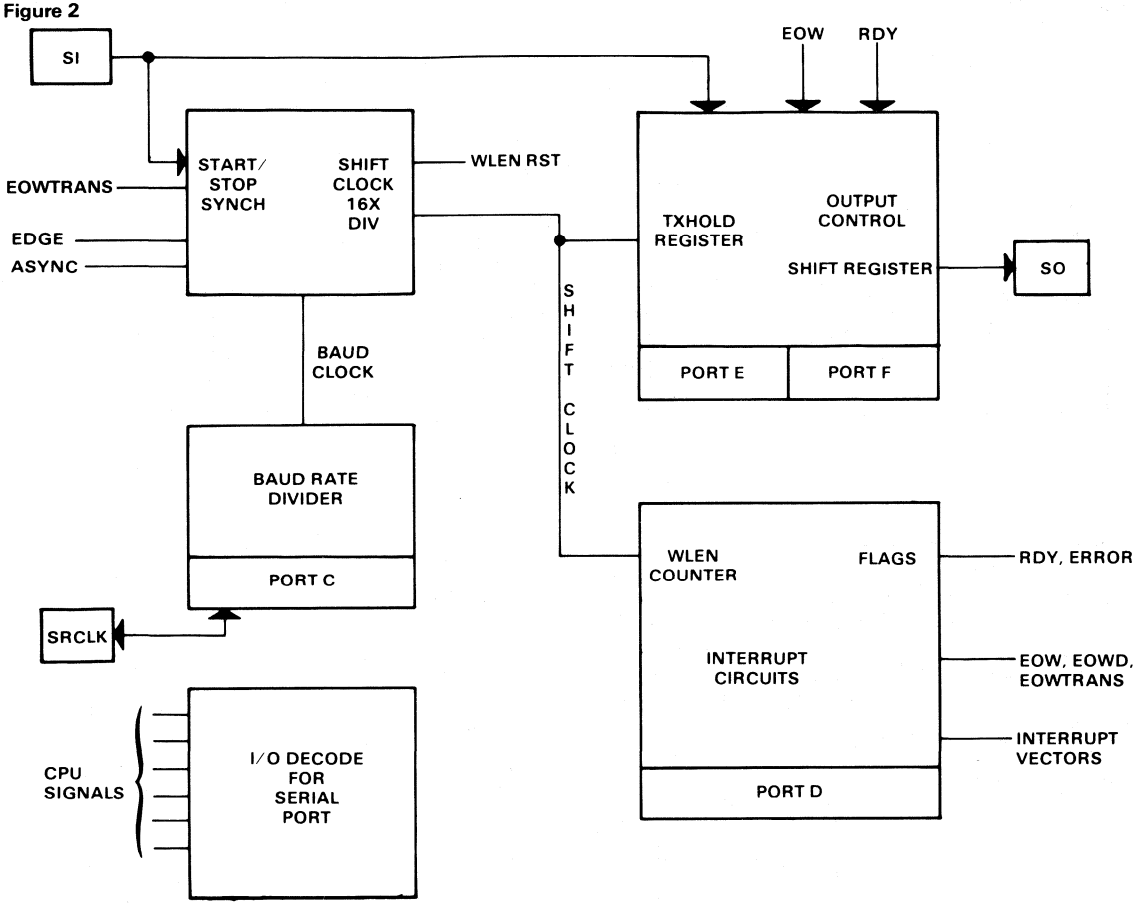
MK3873 SYSTEM BLOCK DIAGRAM

Figure 1



MK3873 SERIAL PORT SIMPLIFIED BLOCK DIAGRAM

Figure 2



this application. As shown, buffers are used between the communication line and the MK3873. Port 1 bit 3 is used for changing the direction of SRCLK from output to input. Direct connection without a modem requires that one of the two microcomputers drive the clock from its internal Baud Rate Divider, while the other microcomputer operates in external SRCLK mode. The figure also shows port 1 bit 7 connected to the switch that determines whether the station is a primary or a secondary.

FULL DUPLEX PROGRAMMING

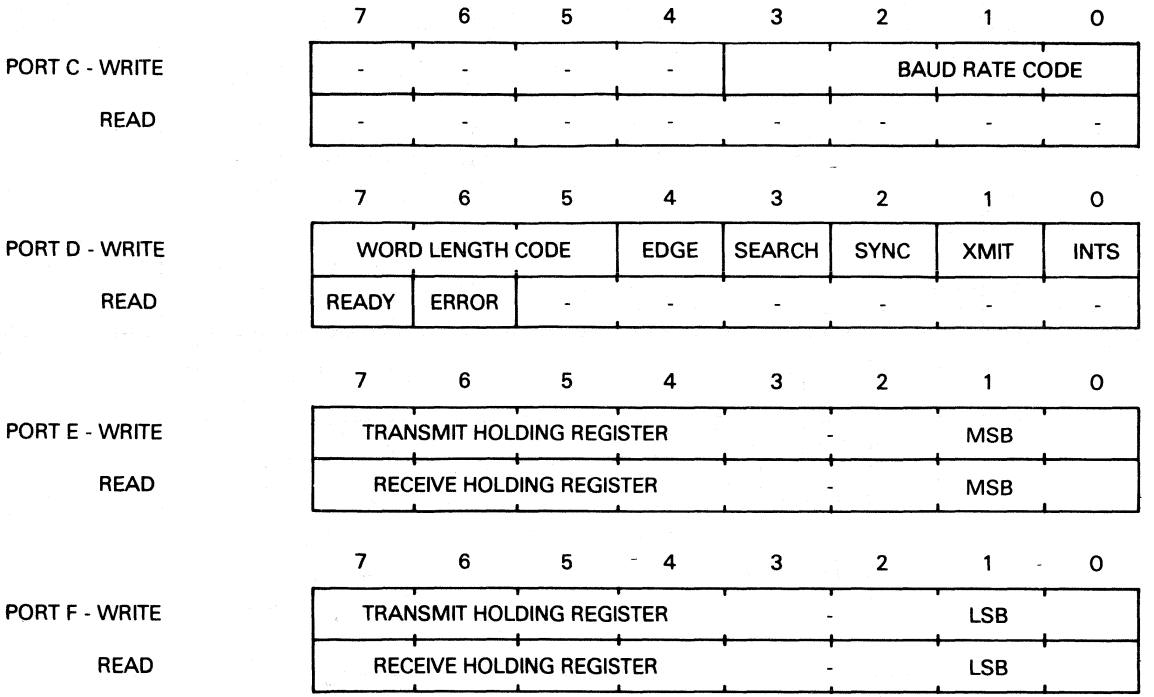
In the sync full-duplex mode, the most appropriate structure for sending and receiving data is the FIFO buffer. Once started, the serial port runs automatically, putting received data into the receive FIFO and taking data from the transmit FIFO to send. The serial port interrupt is the chief stimulus which triggers the serial port service routine at the end of every data frame.

Figure 5 is a simplified flow chart showing what takes place when the sync full-duplex mode is entered. The PRIMARY input is the I/O pin connected to the switch. This input is available on port 1 bit 7, which is a one for primary or zero for secondary. The SYNC FLAG being tested is one which is set by the receive service code whenever a sync character is detected and stripped. Since the service routines operate at interrupt level, constant testing of the flag eventually produces the true result. The tests for SYNCX are tests against the sync character. A pair is required because of the relatively high probability of a single match when character synchronization has not yet been accomplished.

Figure 6 shows the interrupt level functions that are activated by the entry routine when the serial port is placed in transmit mode. Figure 6 also shows the pin connections, SI and SO, as well as the FIFOs that provide logical connection to the calling program. In operation, sending data consists of the calling program writing into the transmit buffer, TXFIFO. Assuming that the interrupt level code is activated, the data will immediately be started through the serial transmit port, continuously, until TXFIFO is emptied. As more data is written into TXFIFO, it is picked up and sent out the serial port. When TXFIFO is empty, sync characters will be sent out to keep the transmit side of the line active. As data appears from the receiver, it is written

MK3873 SERIAL PORT ACCESS I/O PORTS

Figure 3



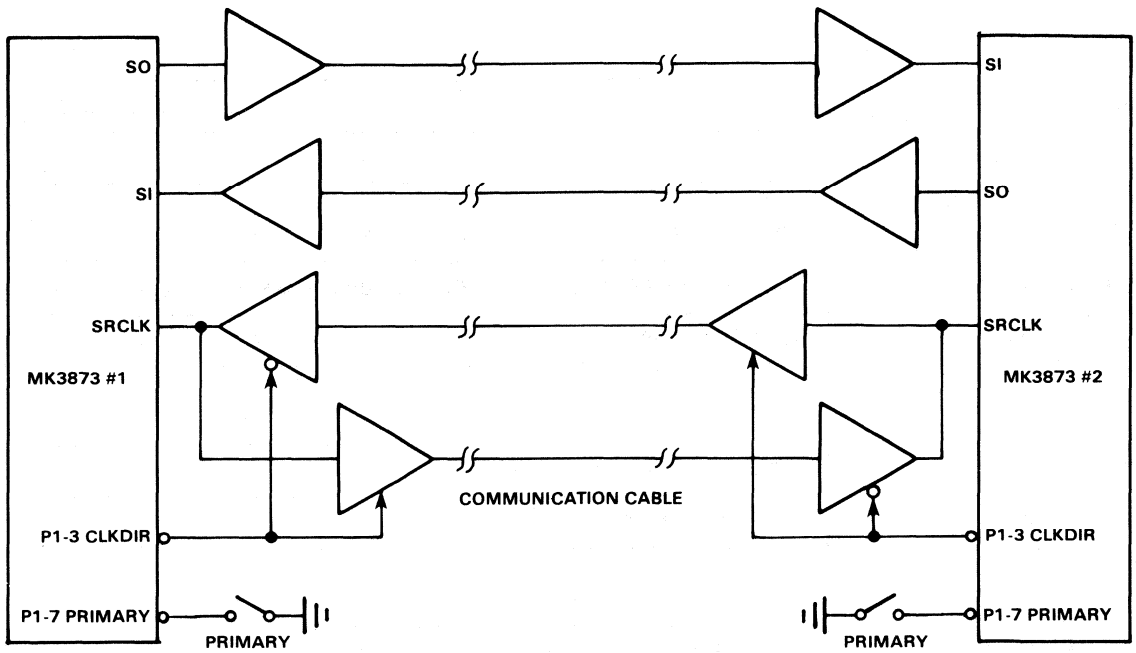
Crystal Frequency = 3.6864 MHz

PORT D WORD LENGTH	NUMBER OF BITS IN WORD	PORT C VALUE	DIVIDE FACTOR	SHIFT RATE (Hz) SYNC	SHIFT RATE (Hz) ASYNC	STANDARD VALUE	PERCENT ERROR (%)
		00	----	EXTERNAL MODE		-	-
		01	----	-NOT USED-		-	-
		02	----	-NOT USED-		-	-
		03	3072	1200	75	BOTH	0.00
0	4	04	2096	*1759	110	ASYNC	-0.07
1	7	05	1536	2400	150	BOTH	0.00
2	8	06	768	4800	300	BOTH	0.00
3	9	07	384	9600	600	BOTH	0.00
4	10	08	192	19200	1200	BOTH	0.00
5	11	09	96	38400	2400	BOTH	0.00
6	12	10	48	76800	4800	BOTH	0.00
7	16	11	24	153600	9600	ASYNC	0.00
		12	----	-NOT USED-		-	-
		13	----	-NOT USED-		-	-
		14	----	-NOT USED-		-	-
		15	----	-NOT USED-		-	-

***NOTE:** 1759 Hz is within 2.3% of 1800 Hz, which is a standard baud rate.

SYNC FULL-DUPLEX HARDWARE DIAGRAM

Figure 4



into the receive buffer, RXFIFO. The calling program has only to read RXFIFO to receive data from the serial port. Sync characters that arrive are of little value, if any, and will be deleted from the data stream if sync stripping is enabled. The sync flag, however, is set whenever the sync character is detected.

Figures 7, 8, and 9 show the actual code for the sync full-duplex serial port driver. The programming example is based upon the use of two registers in scratchpad RAM: CONFIG, which holds the Word Length and Baud Rate values, and FLAGS, which contains various enable and error flags used in the driver routines. The registers are described in detail below.

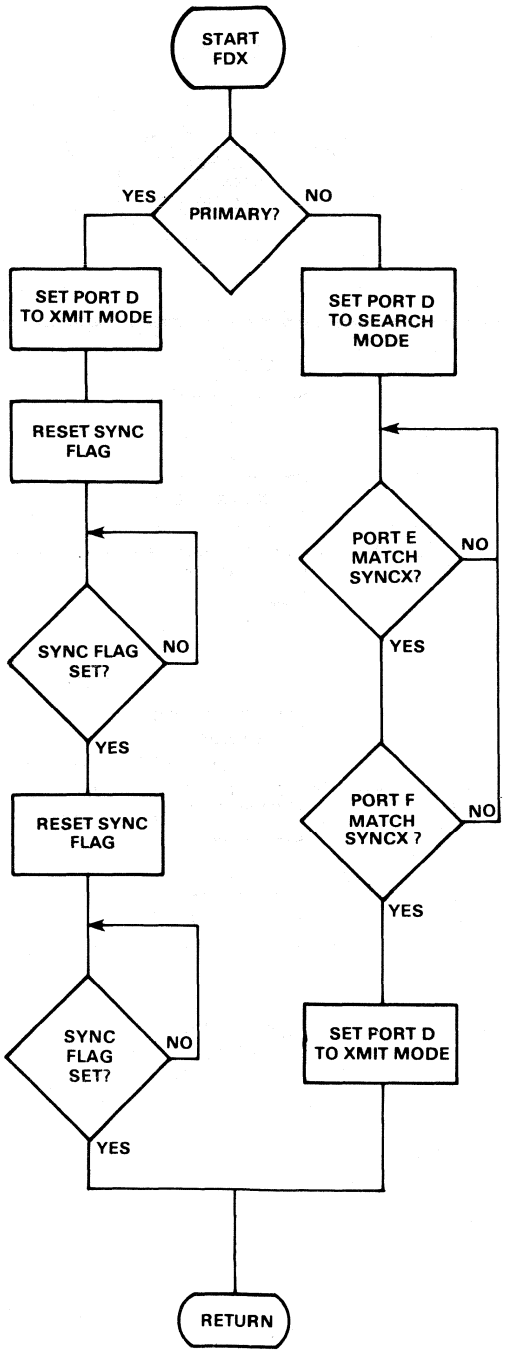
	7	6	5	4	3	2	1	0
CONFIG	WLEN			STRIP	BAUD RATE CODE			

	7	6	5	4	3	2	1	0
FLAGS	PTYEN	SYINT	SYNDT	OVRN	LTERR	PTERR	STOP	STAK

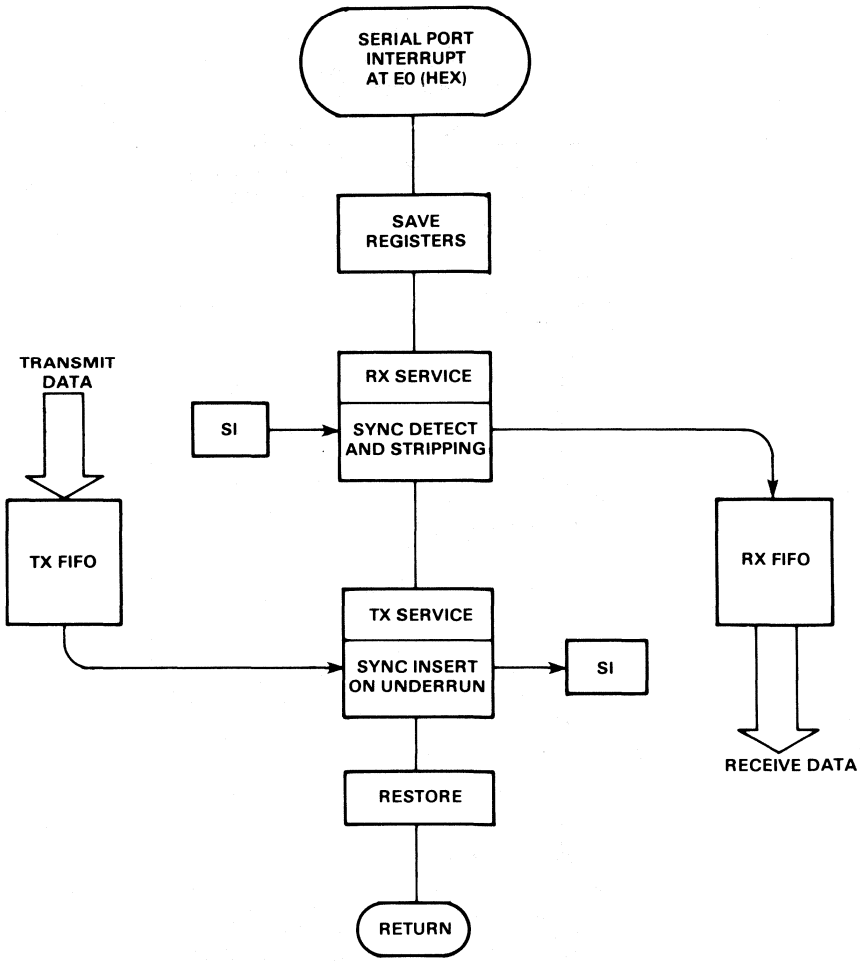
MNEM	MEANING
WLEN	Length code for word including data and parity bits
STRIP	Set to enable sync stripping
PTYEN	Set to enable parity generating and checking
SYINT	Set when system interrupts are enabled
SYNDT	Set when sync character is detected
OVRN	Receiver overrun error flag
LTERR	Latent error, set on serial int with no READY bit
PTERR	Parity error, set when parity check fails
STOP	Set to stop transmit. Reset by transmit code
STAK	Stop acknowledge bit

SYNC FULL-DUPLEX ENTRY ROUTINE

Figure 5



SYNC FULL-DUPLEX INTERRUPT SERVICE
Figure 6



```
;
;
;               SERIAL PORT INITIALIZATION
;
PORT 1:   EQU     1           ;PORT CONTAINING P1CLKD AND P1PRMY
PIPRMY:   EQU     H'80'       ;PRIMARY SENSE BIT
P1CLKD:   EQU     8           ;CLOCK DIRECTION CONTROL BIT
PORTC:    EQU     H'C'        ;BAUD RATE PORT ADDRESS
PORTD:    EQU     H'D'        ;SERIAL CONTROL PORT ADDRESS
PDSRCH:   EQU     8           ;PORT D SEARCH BIT
PDSYNC:   EQU     4           ;PORT D SYNC BIT
PDXMIT:   EQU     2           ;PORT D TRANSMIT BIT
PDINTS:   EQU     1           ;PORT D INT ENABLE BIT
PORTE:    EQU     H'E'        ;UPPER DATA PORT ADDRESS
SYNCX:    EQU     H'16'       ;SYNC MATCH CHAR
PORTF:    EQU     H'F'        ;LOWER DATA PORT ADDRESS
CONFIG:   EQU     8           ;CONFIGURATION REGISTER ADDRESS
MSKWLN:   EQU     H'E0'       ;WORD LENGTH MASK IN CONFIG
MSKSTR:   EQU     H'10'       ;SYNC STRIPPING ENABLE BIT
MSKBRT:   EQU     H'F'        ;MASK FOR BAUD RATE IN CONFIG
FLAGS:    EQU     7           ;FLAG REGISTER ADDRESS
MSKPTY:   EQU     H'80'       ;PARITY ENABLE BIT IN FLAG REG
MSKSYI:   EQU     H'40'       ;SYSTEM INTERRUPT FLAG
MSKSYD:   EQU     H'20'       ;SYNC DETECT FLAG
MSKORE:   EQU     H'10'       ;DATA OVERRUN FLAG
MSKLER:   EQU     8           ;LATENT ERROR FLAG
MSKPTE:   EQU     4           ;PARITY ERROR FLAG
MSKSTP:   EQU     2           ;STOP TRANSMIT FLAG
MSKSTK:   EQU     1           ;STOP ACK BIT IN FLAG REG
GPO:      EQU     0           ;GENERAL PURPOSE REGISTER
;
INITSP:   DI                  ;UNINTERRUPTABLE SUBROUTINE
          LI      MSKPTY+MSKSYI   ;PRESERVE ONLY PTY EN AND SYS INTS
          NS      FLAGS       ;INIT FLAG REGISTER
          LR      FLAGS,A     ;
;
          CLR                 ;ZERO TO PORT D
          OUTS    PORTD       ;
          OUTS    PORT1       ;CLOCK DIR TO OUTPUT
;
          LI      MSKBRT      ;PICK UP BAUD CODE
          NS      CONFIG      ; OUT OF CONFIG REG
          OUTS    PORTC       ;SET THE BAUD RATE
;
          LI      SYNCX       ;PUT SYNC CHAR IN DATA PORTS
          OUTS    PORTE       ;
          OUTS    PORTF       ;
;
          LI      MSKSYI      ;RESTORE SYSTEM INTS
          NS      FLAGS       ;NON-ZERO IF SET
          BZ      INISP1      ;DON'T ENABLE IFS ZERO
          EI                  ;ELSE ENABLE SYSTEM INTS
INISP1:   POP                 ;RETURN TO CALLER
;
;
```

ENTERING SYNC FULL-DUPLEX MODE
Figure 8

```
;
;           ROUTINE TO ENTER SYNC FULL-DUPLEX MODE
;
STFDX:      LR      K,P             ;SUBROUTINE IS INTERRUPTABLE
            LI      MSKPTY+MSKSYI   ;INIT FLAG REG
            NS      FLAGS           ;
            LR      FLAGS,A         ;
;
            INS     PORT1           ;READ PRIMARY SWITCH
            NI      P1PRMY          ;
            BNZ     STFDXP          ;TAKE BR IF PRIMARY
;ELSE THIS IS SECONDARY STATION
            OUTS    PORTC           ;PUT SRCLK PIN IN INPUT MODE
            LIS     P1CLKD          ;SET DIRECTION BIT TO INPUT
            OUTS    PORT1           ;
;
            LI      PDSRCH+PDSYNC   ;SET SYNC & SRCH BITS IN PORT D
            OUTS    PORTD           ;PUT SER PORT IN SEARCH MODE
            OUTS    PORTE           ;RESET THE READY FLAG
;
STFDX1:     INS     PORTD           ;LOOK FOR READY FLAG
            BP      STFDX1          ;LOOP
;
            SL      1               ;TEST FOR OVERRUN
            OUTS    PORTE           ;CLEAR READY FLAG
            BM      STFDX1          ;NO GOOD IF OVERRUN
;
            INS     PORTE           ;READY THE UPPER DATA
            CI      SYNCX           ;MATCH SYNC CHAR
            BNZ     STFDX1          ;LOOP IF NO MATCH
;
            INS     PORTF           ;NEED TWO TO CALL IT A MATCH
            CI      SYNCX           ;2ND CHAR SHOULD ALSO MCH FOR SYNC
            BNZ     STFDX1          ;KEEP TRYNG IF NO EQ
;
            LI      MSKWLN          ;SET PORT TO XMIT MODE
            NS      FLAGS           ;PICK UP WLEN
            OI      PDSYNC+PDXMIT+PDINTS    ;FORM PORT D CODE
            OUTS    PORTD           ;OUTPUT IT
;
            LR      A,FLAGS         ;SET SYNC DET
            OI      MSKSYD          ;
            LR      FLAGS,A         ;
;
            LI      SYNCX           ;SYNC CHAR TO DATA PORTS
            OUTS    PORTE           ;
            OUTS    PORTF           ;
;
            PK                      ;RETURN TO CALLER
;
;
;           (CONTINUED ON NEXT PAGE)
```

```
;
;           (CONTINUATION FROM PREVIOUS PAGE)
;
;
;START PRIMARY STATION
;
STFDXP:     CLR                             ;SET CLK DIRECTION TO OUTPUT
            OUTS    PORT1                   ;
            LI      MSKBRT                  ;PUT SRCLK IN OUTPUT MODE
            NS      CONFIG                  ;GET BAUD RATE CODE
            OUTS    PORTC                   ;BAUD RATE PORT
;
            LI      SYNCX                   ;PUT SYNC CHARS IN THE DATA PORTS
            OUTS    PORTE                   ;
            OUTS    PORTF                   ;
;
            LI      MSKWLN                  ;PUT SER PORT IN XMIT MODE
            NS      CONFIG                  ;FORM PORT D CODE
            OI      PDSYNC+PDXMIT+PDINTS    ;
            OUTS    PORTD                   ;OUTPUT IT
;
STFDX2:     LI      MSKSYD                  ;TEST FOR SYNC DET FLAG
            NS      FLAGS                   ;
            BZ      STFDX2                  ;LOOP TIL SET
;
            LI      .NOT.MSKSYD             ;RESET IT FOR A SECOND TEST
            DI                              ;PROTECT REG FROM CONTENTION ACCESS
            NS      FLAGS                   ;
            LR      FLAGS,A                 ;
            EI                              ;
;
STFDX3:     LI      MSKSYD                  ;TEST SYNC DET AGAIN
            NS      FLAGS                   ;
            BZ      STFDX3                  ;STAY TIL SET
;
            PK                              ;RETURN
;
;
;
```

INTERRUPT LEVEL SYNC FULL-DUPLEX

Figure 9

```
; ROUTINE TO HANDLE SYNC FULL-DUPLEX INTERRUPTS
;
;
ISCRU:    EQU     7               ;INTERRUPT SCRATCH AREA
ISCRL:    EQU     0               ; ISAR VALUES
JR:       EQU     9               ;REGISTER 9
;
          ORG     H'E0'
;
FDXINT:   LR      JR,A            ;TEMP STASH ACC IN J
          LR      A,IS            ;SAVE ISAR
          LISU    ISCRU           ;POINT TO SCRATCH AREA
          LISL    ISCRL           ;
          LR      I,A             ;SAVE ISAR IN 1ST LOC
          LR      A,JR            ;PICK UP THE ACC
          LR      I,A             ;ACC IN LOC #2
          LR      J,W             ;SAVE PROC STATUS
          LR      A,KU            ;FREE UP THE K REG TO HOLD P1
          LR      I,A             ; FOR SUBROUTINE CALLS
          LR      A,KL            ; USING P1
          LR      I,A             ;
          LR      A,GP0           ;SAVE THIS FOR DATA
          LR      S,A             ;
          LR      K,P             ;SAVE P1 IN K
;
          LI      .NOT.MSKSYI     ;RESET SYSTEM INT FLAG
          NS      FLAGS           ;
          LR      FLAGS,A         ;
;
          LI      MSKSTK          ;TEST STOP ACKNOWLEDGE
          NS      FLAGS           ;
          BZ      FDXIO           ;BR AROUND CALL IF NOT SET
          PI      INITSP          ;INITIALIZE SERIAL PORT
          JMP     FDXI9           ;GO RESTORE REGS AND RETURN
;
;SYNC FULL-DUPLEX RECEIVE SERVICE ROUTINE
; ASSUMES 8-BIT DATA & NO PTY, OR 7-BIT + ODD PTY IF ENABLED
;
FDXIO:    INS     PORTD           ;TEST STATUS
          SL      1               ;OVERRUN?
          BP      FDXI1           ;BZ AROUND ERR SET IF POS
;
          LR      A,FLAGS         ;SET OVRN FLAG
          OI      MSKORE          ;
          LR      FLAGS,A         ;
;
FDXI1:    INS     PORTD           ;TEST FOR LATENT ERROR
          BM      FDXI2           ;BR AROUND ERR SET IF READY ON
          LR      A,FLAGS         ;SET LTERR
          OI      MSKLER          ;
          LR      FLAGS,A
;
FDXI2:    INS     PORTE           ;GET THE RX DATA
          LR      GP0,A           ;SAVE IT IN GP0
;
;         (CONTINUED ON NEXT PAGE)
```

```
;           (CONTINUATION OF PREVIOUS PAGE)
;
;
            LI      MSKPTY          ;PARITY ENABLED?
            NS      FLAGS           ;
            BZ      FDXI3           ;BR IF NO PARITY
;
            PI      PARITY          ;PASSES IF ODD
            BM      FDXI4           ;MINUS MEANS ODD
;
            LR      A,FLAGS         ;SET ERROR IF EVEN
            OI      MSKPTE          ;
            LR      FLAGS,A         ;
FDXI4:      LI      H'7F'           ;TRIM DATA TO 7 BITS
            NS      GP0             ;
            LR      GP0,A           ;
;
FDXI3:      LR      A,GP0           ;SEE IF SYNCX CHARACTER
            CI      SYNCX           ;
            BNZ     FDXI5           ;BR AROUND IF NO MATCH
;
            LR      A,FLAGS         ;
            OI      MSKSYD          ;SET SYNC DETECT FLAG
            LR      FLAGS,A         ;
;
            LI      MSKSTR          ;TEST STRIPPING ENABLE BIT
            NS      CONFIG          ;
            BNZ     FDXI6           ;BR TO END RX SVC IF STRIP ENABLED
;
FDXI5:      PI      WRRXF           ;WRITE DATA TO RXFIFO
            BP      RDXI6           ;END RX SERVICE IF FIFO NOT FULL
;
            LR      A,FLAGS         ;SET OVERFLOW IF FIFO FULL
            OI      MSKORE          ;
            LR      FLAGS,A         ;
;
;TRANSMIT SERVICE ROUTINE
;
FDXI6:      PI      RDTXF           ;READ THE TRANSMITTER FIFO
            BP      FDXI7           ;BR AROUND SYNC INSERTION IF NOT MT
;
            LI      MSKSTP          ;STOP REQUESTED?
            NS      FLAGS           ;
            BZ      FDXIB           ;BR IF NO STOP REQUEST
;
            LR      A,FLAGS         ;SET STOP ACK BIT
            OI      MSKSTK          ;
            LR      FLAGS,A         ;
;
FDXIB:      LI      SYNCX           ;INSERT SYNC CHAR
            LR      GP0,A           ; AND CONTINUE
;(FDX17  IS CONTINUED ON THE NEXT PAGE)
```

```
;               (CONTINUATION FROM PREVIOUS PAGE)
;
FDXI7:      LI      MSKPTY      ;PARITY ENABLED?
            NS      FLAGS       ;
            BZ      FDXI8       ;BR AROUND IF NOT ENABLED
;
            PI      PARITY      ;GENERATE PARITY
            BM      FDXI8       ;BR AROUND IF ALREADY ODD
;
            LI      H'80'       ;FLIP BIT 7 IF EVEN
            XS      GP0         ;
            LR      GP0,A       ;
;
FDXI8:      LR      A,GP0       ;GET DATA
            OUTS    PORTF       ;PLACE IT IN PORT F
;
;
;RESTORE REGISTERS FOR RETURN FROM INTERRUPT
;
FDXI9:      LR      A,FLAGS     ;PUT SYSTEM INTS BACK ON
            OI      MSKSYI      ;
            LR      FLAGS,A     ;
            LISU    ISCRU       ;POINT TO SCR AREA
            LISL    ISCRL+4     ;
            LR      A,D         ;RESTORE GP0
            LR      GP0,A       ;
            LR      P,K         ;RESTORE INT RETURN ADDRESS
            LR      A,D         ;GET K CONTENTS
            LR      KL,A        ;
            LR      A,D         ;
            LR      KU,A        ;
            LR      W,J         ;RESTORE PROC STATUS
            LR      A,D         ;GET ACC VALUE
            LR      JR,A        ;TEMP STORAGE IN J
            LR      A,S         ;ISAR LAST
            LR      IS,A        ; RESTORE IT
            LR      A,JR        ; RESTORE THE ACC
            EI                  ;RESTORE INTS
            POP                 ; AND RETURN
;
;
```

ROUTINES REFERENCED BUT NOT INCLUDED

Undefined routines have been referred to in the programming example, and others have been mentioned implicitly in the text. These have to do with reading and writing the FIFOs, and generating and checking parity. These items have been covered in other Mostek literature, and are easily characterized in words.

Read and write FIFO routines check the empty and full conditions of the FIFOs prior to executing the read or write. If the condition precludes successful completion of the read or write function, then negative status is returned to the caller. If the read or write is successfully done, positive status is returned.

In the case of parity computation, the routine is called with the target value in the GPO register. The routine counts the number of ones in GPO. If the result is odd, negative status is returned. If the result is even, positive status is returned. The same routine is useful for checking or generating parity on any data length, up to eight bits.

Stopping the automatic code in the serial port should be done synchronously to avoid fragmenting any last characters going out the transmitter. With the stop logic in the transmit service routine, all that is necessary to stop the port is to set the STOP bit in the configuration register, CONFIG, and then to wait for STOP to be reset by the transmit service routine. When STOP is reset, the serial port will have been gracefully shut down and put in the initialized state.

ANALYSIS OF SYNC FULL-DUPLEX PROGRAMMING EXAMPLE

The initialization routine of Figure 7 is used to put the serial port into the inactive state. Following the execution of the routine, the interrupts are disabled, the SRCLK pin is in the output mode at the 16 times the current Baud Rate, and the serial output, SO, is in a marking or disabled state. The routine takes as much as 87.9 microseconds to complete, once it is begun. This assumes a crystal frequency of 3.6864 MHz running the chip.

The full-duplex entry routine of Figure 8 has code that requires response at the bit level for setting the secondary station into character synchronization. The time starting with the detection of READY being set, to the two character match for sync characters, to putting the serial port in the transmit mode, can be as much as 111.8 microseconds.

The interrupt level code has several parts that contribute to the overall throughput and the maximum Baud rate that is sustainable.

The critical time values are shown.

TIME (μsec)	DESCRIPTION
111.8*	Time from READY of sync char bit to putting the serial port into XMIT mode.
25.0	Time from interrupt to execution of code at vector location.
48.8	To save important registers in scratch-pad RAM.
28.2*	To get to stop processing routine from reg save.
87.9*	To execute initialization and stop serial port.
58.6	Save to first read of RX holding register.
431.9	To process RX character incl parity, write FIFO.
424.3	To process and output TX character.
59.7	To restore registers and return from interrupt.

***NOTE:** Times marked with asterisk critical on per bit basis.

It is significant to note that the time to generate and check data parity is as high as 532 microseconds, or about 51% of the total.

The numbers above indicate that an interrupt service execution takes up to 1048.3 microseconds to complete (this assumes seven bit data with odd parity, and no errors detected). Time to handle errors is relatively short. The assumed time for the parity subroutine was 266 microseconds, based on 122.5 machine cycles; the time for read or write of the FIFO was 81.4 microseconds, based on 37.5 cycles.

Interpreting the numbers, and assuming a requirement to keep a certain amount of the available processing power of the MK3873 free for other use, some recommendations can be made with respect to maximum data rates.

When considering the need to achieve character synchronization, and considering the need to stop the port once it has been transferring data, the bit rate is of interest. On the other hand, when determining the data handling throughput, the character rate is important. A 4800 Baud channel would require the microcomputer to synchronize the port within its bit time, or 208 microseconds. This means that since the code described would synchronize in 112 microseconds, that it would have an operating margin of approximately 46.4%. At eight bits per character, the interrupts would be arriving every 1667 microseconds, and the MK3873 is capable of completing an interrupt service every 1048 microseconds. The margin for character handling is 37.1%. The table below summarizes the margins for the functions to be performed. As shown, to function with a positive operating margin, the maximum internally programmable Baud rate supportable is 4800 Baud.

Baud Rate	Character Handling Margins (%) With Parity	Without Parity	Bit Sync Margins (%) Start	Stop
1200	84.3	92.3	86.6	72.2
2400	68.6	84.5	73.2	54.4
4800	37.1	69.0	46.4	8.8
9600	-25.8	38.0	-7.3	-82.3

WORK-AROUNDS FOR FULL-DUPLEX ASYNC

As was indicated earlier, the MK3873 serial port supports full-duplex in sync mode only. This section discusses methods of accomplishing full-duplex async transmission using the serial port for receiving, and Baud rate clock, and by synthesizing the transmitter from other means.

SOFTWARE ASYNC TRANSMITTER

The functions performed by an async transmitter are simpler to handle in software than those of the receiver. Included are adding start bit, stop bits, and parity, and then shifting the results out an output pin. The receiver, on the other hand, required edge detection and start bit verification, in addition to shifting and stripping start and stop bits. stop bits.

The MK3873 provides the Baud rate clock at the pin SRCLK. By connecting SRCLK to the external interrupt pin, EXT INT, and operating the MK3873 timer facility in the event counter mode, it is possible to accomplish the async transmitter function very efficiently. The transmitter thus constructed operates at the same Baud rate as the receiver in the serial port, but the phase of the two data paths relative to each other is independent. The independence between the receiver and transmitter is realized because of the separate interrupt logic, and the fact that the word length counts are now separate, since the transmitter's counter is done in software. The main drawback to this approach is that the receiver interrupt must re-enable interrupts extermely fast. Otherwise, excessive phase jitter in the transmitter data may result.

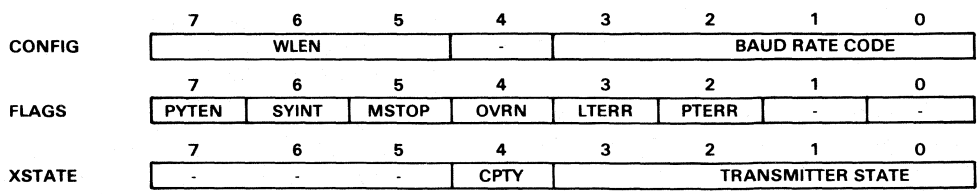
This method is implemented in Figures 10, 11 and 12. The use of the timer and external interrupt facilities for this purpose make them essentially unavailable for any other purpose while transmitting. The timer, however, is free for use whenever it is not being used for transmission.

In this example, the Serial Clock is used to decrement the value in the timer port, which is set to sixteen. When sixteen clock pulses are counted, the timer interrupts with a vector of 20 (hex). There is a state register that holds the state of the transmitter. The states vary in operation from zero, which is idle, to twleve, which is the last state. The states are defined in the table below.

STATE	DEFINITION
0	Idle state
1	Starting state - outputting start bit
2	Last data bit being sent
3-9	Intermediate data bits being sent
10	Parity bit being outputted
11	Stop bit being sent
12	Second stop bit being sent

The state transitions and what goes on in each state are shown in the flow chart of Figure 11. The async transmitter code is started by calling the ASXMT routine, illustrated in Figure 10. Initially, the FIFO is not full, and the transmitter is in state 0. The ASXMT routine writes a 10 (hex) into port 7 so that the event counter will count sixteen Serial Clocks. It then writes an OE (hex) into the interrupt control port, port 6, starting the count and enabling timer interrupts. The transmitter state is set to 1. The rest of the transmitter activity takes place under interrupts. The state is used to direct a dispatch jump to the routine needed based upon the state. At each successive interrupt, the data in the output port is shifted once to present the next serial bit. The interrupt level routine is coded in Figure 12.

The registers used for configuration and flags are shown below.



MNEM	MEANING
WLEN	Length code for word including data and parity bits
PTYEN	Set to enable parity generating and checking
SYINT	Set when system interrupts are enabled
NSTOP	Set for two stop bits. Clear for one stop bit
OVRN	Receiver overrun error flag
LTERR	Latent error, set on serial int with no READY bit
PTERR	Parity error, set when parity check fails
CPTY	Current parity value of outgoing data

SENDING ASYNC FULL-DUPLEX DATA
Figure 10

```
;
; ROUTINE TO SEND ASYNC FULL DUPLEX DATA
;
XSTATE:   EQU     6              ;SCRATCHPAD CELL FOR TX STATE
MSKXMS:   EQU     H'F'           ;XMTR STATE MASK IN XSTATE
;
ASXMT:    LR      K,P            ;SUBROUTINE IS INTERRUPTABLE
ASXMT1:   PI      WRTXF          ;WRITE TRANSMIT FIFO
          BM      ASXMT1         ;LOOP UNTIL NOT FULL
;
          LI      MSKXMS         ;TEST TRANSMIT STATE VALUE
          NS      XSTATE         ;
          BNZ     ASXMT2         ;BR AROUND INIT IF NON-ZERO
;
          LIS     1              ;INIT STATE TO 1
          LR      XSTATE,A       ;
          LI      H'10'          ;16X CONSTANT INTO COUNTER
          OUTS    PORT7          ;SET TIMER COUNT VALUE
          LIS     H'E'           ;COME TO ENABLE EVENT COUNTER
          OUTS    PORT 6         ; AND TURN ON INTERRUPTS
;
ASXMT2:   PK                     ;RETURN TO CALLER
;
```

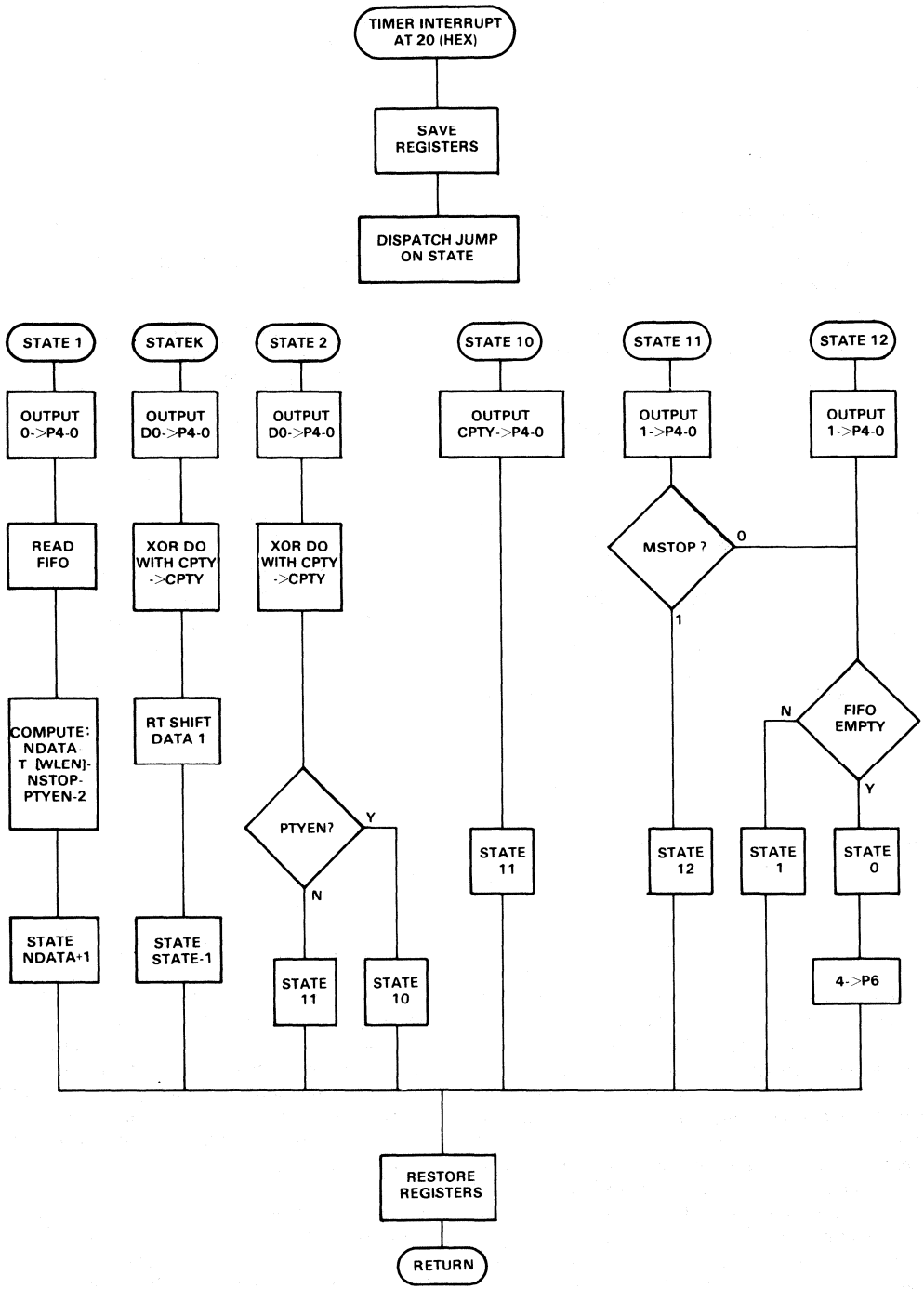
Special consideration must be taken in the receiver interrupt code in order to allow the transmitter data to be shifted under the direct control of the timer interrupt. The timer has a lower interrupt priority than the serial port, and thus the transmitter interrupt latency will be the nominal 15 to 35 microsecond system interrupt latency, plus the time the receiver interrupt code keeps interrupts disabled. Since it is necessary to keep phase jitter in async transmission below 3% to meet most interface specifications for signal quality, the total interrupt latency for the transmitter (timer) interrupt should be kept within 3% of a bit time. For 300 Baud async, this means holding interrupt latency within 100 microseconds. 600 Baud would require 50 microseconds.

It is not practical to process the received data in a time period as short as 50 to 100 microseconds. Therefore it is necessary to save the accumulator, processor status, and return address immediately upon entering the receiver interrupt code, thus permitting the timer interrupt to suspend operation of the receiver processing code. To accomplish this requires the use of additional scratchpad RAM space.

Figure 13 illustrates a receiver interrupt service routine that has provision for being interrupted as soon as possible after entry. Absolute care must be taken to ensure that the interrupts in the system are not left disabled for more than a few microseconds so that the transmitter code may be permitted to proceed when necessary.

INTERRUPT LEVEL ASYNC FULL-DUPLEX TRANSMIT FLOW

Figure 11



INTERRUPT LEVEL ASYNC FULL-DUPLEX TRANSMIT CODE
Figure 12

```
;
; ROUTINE TO HANDLE ASYNC FULL-DUPLEX TRANSMITTER INTERRUPTS
;
ISCRU:    EQU     7                ;INTERRUPT SCRATCH AREA
ISCRL:    EQU     0                ; ISAR VALUES
JR:       EQU     9                ;REGISTER 9
;
          ORG     H'20'
;
XMTINT:   LR      JR,A             ;TEMP STASH ACC IN J
          LR      A,IS             ;SAVE ISAR
          LISU    ISCRU            ;POINT TO SCRATCH AREA
          LISL    ISCRL            ;
          LR      I,A              ;SAVE ISAR IN 1ST LOC
          LR      A,JR             ;PICK UP THE ACC
          LR      I,A              ;ACC IN LOC #2
          LR      J,W              ;SAVE PROC STATUS
          LR      A,KU             ;SAVE THE K REG TO TRANSFER P1
          LR      I,A              ;
          LR      A,KL             ;
          LR      I,A              ;
          LR      K,P              ;SAVE THE P1
          LR      A,KU             ;
          LR      I,A              ;
          LR      A,KL             ;
          LR      I,A              ;
          LR      A,GP0            ;SAVE THIS FOR DATA
          LR      I,A              ;
;
          LR      A,S              ;GET SHIFTED DATA
          LR      GP0,A            ;
;
          LI      .NOT.MSKSYI      ;RESET SYSTEM INT FLAG
          NS      FLAGS            ;
          LR      FLAGS,A          ;
;
          DCI     ASXMJT           ;TRANSMITTER JUMP TABLE
          LI      MSKXMS           ;SET UP DISPATCH JUMP ON STATE
          NS      XSTATE           ;
          SL      1                ;
          ADC                      ;POINT DC TO ADDRESS
;
          LM                       ;GET UPPER BYTE OF ADDRESS
          LR      KU,A             ;
          LM                       ;GET LOWER BYTE OF ADDRESS
          LR      KL,A             ;
          PK                       ;PERFORMS DISPATCH JUMP
;
;
;
;         (CONTINUED ON NEXT PAGE)
;
```

```
;
;           (TRANSMITTER JUMP TABLE
;
;
ASXMJT:   DEFW    STATE0          ;CENTRAL RETURN POINT
          DEFW    STATE1          ;STATE UP ROUTINE
          DEFW    STATE2          ;LAST DATA BIT
          DEFW    STATEK          ;INTERMEDIATE DATA BITS
          DEFW    STATEK          ;INTERMEDIATE DATA BITS
          DEFW    STATEK          ;INTERMEDIATE DATA BITS
          DEFW    STATEK          ;INTERMEDIATE DATA BITS
          DEFW    STATEK          ;INTERMEDIATE DATA BITS
          DEFW    STATEK          ;INTERMEDIATE DATA BITS
          DEFW    STATEA          ;OUTPUT PARITY
          DEFW    STATEB          ;OUTPUT STOP BIT
          DEFW    STATEC          ;OUTPUT 2ND STOP BIT/
;
;
STATE0:   LR      A,FLAGS         ;PUT SYSTEM INTS BACK ON
          OI      MSKSYI          S;
          LR      FLAGS,A         ;
          LISU    ISCRU           ;POINT TO SCR AREA
          LISL    ISCRL+7         ;
          LR      A,GP0           ;SAVE SHIFTED DATA
          LR      D,A             ;
          LR      A,D             ;RESTORE GP0
          LR      GP0,A           ;
          LR      A,D             ;GET P1 CONTENTS
          LR      KL,A            ;
          LR      A,D             ;
          LR      KU,A            ;
          LR      P,K             ;RESTORE INT RETURN ADDRESS
          LR      A,D             ;GET K CONTENTS
          LR      KL,A            ;
          LR      A,D             ;
          LR      KU,A            ;
          LR      W,J             ;RESTORE PROC STATUS
          LR      A,D             ;GET ACC VALUE
          LR      JR,A            ;TEMP STORAGE IN J
          LR      A,S             ;ISAR LAST
          LR      IS,A            ; RESTORE IT
          LR      A,JR            ; RESTORE THE ACC
          EI                      ;RESTORE INTS
          POP                     ; AND RETURN
;
;
;
;
```

```
;
;
;CODE FOR STATE 1
;
STATE1:   LIS     1               ;TX DATA PORT BIT IN PORT 4
          OUTS    PORT4           ;DATA AT OUTPUT IS TRUE, SO IT MUST BE COMPLEMENTED
;                                 GOING OUT
          DCI     WLENTB          ;BASE OF WORD LENGTH TABLE
          LI      MSKWLN          ;WORD LENGTH FIELD IN CONFIG
          NS      CONFIG          ;PICK UP WLEN TO COMPUTER NDATA
          SR      4               ;ADJUST FOR INDEXING
          SR      1               ;
          ADC                     ;
          LM                      ;PICK UP ACTUAL LENGTH
          LR      GP0,A           ;TEMP STORE IT IN GP0
;
          LI      PTYEN           ;DECR LENGTH FOR PARITY BIT
          NS      FLAGS           ;
          BZ      STAT11          ;BR TO AVOID DECREMENT
;
          DS      GP0             ;COMPENSATE FOR PARITY BIT
STAT11:   LI      MSKNSP          ;TEST FOR TWO STOP BITS
          NS      FLAGS           ;
          BZ      STAT12          ;BR AROUND IF ONE STOP BIT
;
          DS      GP0             ;DECREMENT TO COMPENSATE
STAT12:   DS      GP0             ;DECREMENT FOR START BIT
          LR      A,GP0           ;SET STATE AND CPTY TO INITIAL VALUE
          LR      XSTATE,A        ;
;
          PI      RDTXF           ;GET THE TX DATA
          LR      A,GP0           ;COMPLEMENT THE DATA
          COM                     ;
          LR      GP0,A           ;
;
          JMP     STATE0          ;RETURN
;
;
WLENTB:   DEFB    4               ;LENGTH OF WORD
          DEFB    7               ;
          DEFB    8               ;
          DEFB    9               ;
          DEFB    10              ;
          DEFB    11              ;
          DEFB    12              ;
          DEFB    16              ;
;
;
;
```

```
;
;STATEK ROUTINE FOR INTERMEDIATE DATA BITS
;
;
STATEK:   LR      A,GP0          ;GET THE DATA
          NI      1
          OUTS    PORT4          ;TRANSMIT IT
          BNZ     STATK1         ;BR AROUND PARITY TOGGLE IF DATA COMPLEMENTED IS A
;                                ONE
          LI      MSKCPT         ;FLIP THE CPTY BIT
          XS      XSTATE         ;
          LR      XSTATE,A       ;
;
STATK1:   LR      A,GP0          ;SHIFT THE DATA
          SR      1              ;
          LR      GP0,A          ;
;
          DS      XSTATE         ;DECREMENT THE TRANSMITTER STATE
;
          JMP     STATE0         ;RETURN
;
;
;STATE 2 CODE - LAST DATA BIT
;
STATE2:   LR      A,GP0          ;GET THE DATA
          NI      1
          OUTS    PORT4          ;TRANSMIT IT
          BNZ     STAT21         ;BR AROUND PARITY TOGGLE IF DATA COMPLEMENTED IS A
;                                ONE
          LI      MSKCPT         ;FLIP THE CPTY BIT
          XS      XSTATE         ;
          LR      XSTATE,A       ;
;
STAT21:   LI      MSKCPT         ;CLEAR STATE
          NS      XSTATE         ;PRESERVE CPTY
          LR      XSTATE,A       ;
;
          LI      MSKPTY         ;IS PARITY ENABLED
          NS      FLAGS          ;
          LIS     10             ;NEXT STATE IF PTY ENABLED
          BNZ     STAT22         ;BR AROUND INCR IF PTY ENABLED
;
          INC                    ;SET NEXT STATE TO 11
STAT22:   XS      XSTATE         ;MERGE WITH PARITY BIT
          LR      XSTATE,A       ;
;
          JMP     STATE0         ;RETURN
;
;
```

```
;
;
;STATE 10 - OUTPUT PARITY BIT
;
STATEA:   LI      MSKCPT          ;PICK UP PARITY
          NS      XSTATE          ;
          CLR                     ;OUTPUT VALUE FOR CPTY=1
          BZ      STATA1          ;BR AROUND INCR IF CPTY=1
;
          INC                     ;VALUE TO 0 AT OUTPUT
STATA1:   OUTS    PORT4           ;WRITE THE PARITY BIT
;
          LIS     H'B'            ;NEXT STATE
          LR      XSTATE,A        ;
;
          JMP     STATE0          ;RETURN
;
;
;STATE B - OUTPUT FIRST STOP BIT
;
STATEB:   CLR                     ;OUTPUT STOP BIT
          OUTS    PORT4           ;
;
          LI      MSKNSP          ;TEST FOR TWO STOP BITS
          NS      FLAGS           ;
          BZ      STATC1          ;GO JOIN STATE C ROUTINE IF ALST
;
          LIS     H'C'            ;LAST STATE
;
          JMP     STATE0          ;RETURN
;
;
;STATE C - LAST STATE
;
STATEC:   CLR                     ;OUTPUT STOP BIT
          OUTS    PORT4           ;
;
STATC1:   LISU    TXFIFU          ;POINT TO TX FIFO
          LISL    TXFIFL          ; POINTER
          LR      A,S             ;READ POINTER
          SL      1               ;MT FLAG IN BIT 6
          LIS     1               ;CONTINUE STATE  IF NOT MT
          BP      STATC2          ;BR AROUND TO CONTINUE
;
          LIS     4               ;TURN OFF EVENT COUNTER
          OUTS    PORT6           ;
          CLR                     ;STATE TO ZERO
STATC2:   LR      XSTATE,A        ;SET THE STATE VALUE
;
          JMP     STATE0          ;AND RETURN
;
```

INTERRUPT LEVEL ASYNC FULL-DUPLEX RECEIVE CODE

Figure 13

```
;
; ROUTINE TO HANDLE ASYNC FULL-DUPLEX RECEIVER INTERRUPTS
;
RXSCRU:   EQU     7                   ;RX INTERRUPT SCRATCH AREA
RXSCRL:   EQU     0                   ; ISAR VALUES
JR:       EQU     9                   ;REGISTER 9
GP0       EQU     0                   ;GEN PURPOSE REG 0
GP1       EQU     1                   ;GEN PURPOSE REG 1
GP2       EQU     2                   ;GEN PURPOSE REG 2
;
          ORG     H'60'
;
RCVINT:   LR      JR,A                ;TEMP STASH ACC IN J
          LR      A,IS                ;SAVE ISAR
          LISU    RXSCRU              ;POINT TO SCRATCH AREA
          LISL    RXSCRL              ;
          LR      I,A                 ;SAVE ISAR IN 1ST LOC
          LR      A,JR                ;PICK UP THE ACC
          LR      I,A                 ;ACC IN LOC #2
          LR      J,W                 ;SAVE PROC STATUS
          LR      A,JR                ;
          LR      I,A                 ;
          LR      A,KU                ;SAVE THE K REG TO TRANSFER P1
          LR      I,A                 ;
          LR      A,KL                ;
          LR      I,A                 ;
          LR      K,P                 ;SAVE THE P1
          EI                          ;INTERRUPTABLE AT THIS POINT!
;
          LR      A,GP0               ;SAVE THIS FOR DATA
          LR      I,A                 ;
          LR      A,GP1               ;MORE GP SAVE
          LR      I,A                 ;
          LR      A,GP2               ;
          LR      S,A                 ;
;
          INS     PORTD               ;TEST STATUS
          SL      1                   ;OVERRUN?
          BP      RCVIN1              ;BZ AROUND ERR SET IF POS
;
          LR      A,FLAGS             ;SET OVRN
          OI      MSKORE              ;
          LR      FLAGS,A             ;
;
RCVIN1:   INS     PORTD               ;TEST FOR LATENT ERROR
          BM      RCVIN2              ;BR AROUND ERR SET IF READY ON
          LR      A,FLAGS             ;SET LSTERR
          OI      MSKLER              ;
          LR      FLAGS,A             ;
;
;RX ASYNC SERVICE ASSUMES 7 DATA BITS AND ODD DPARITY IF ENABLED
;
RCVIN2:   LI      MSKWLN              ;SEE HOW BIG WLEN IS
          NS      CONFIG              ;
          SR      4                   ;CONVENIENT FOR ALU
          CI      8                   ;NEG=11, EQ=10, POS=9BITS
          BZ      RCVIN3              ;BR TO HANDLE 10 BITS IF 0
; (CONTINUED ON NEXT PAGE)
```

```
                BM      RCVIN3          ;BR TO HANDLE 11 BITS IF NEG
;
                INS     PORTE           ;STAY TO HANDLE 9 BITS
LR              GP0,A   ;STORE DATA IN GP0
;
RCVIN5:         LI      MSKPTY          ;SEE IF PARITY ENABLED
                NS      FLAGS           ;
                BZ      RCVIN6          ;BR AROUND PARITY CHECK IF 0
; ROUTINE TO COMPUTE PARITY
                LIS     7               ;BIT COUNT
                LR      GP1,A           ;
                CLR                     ;PARITY IN LSB OF GP2
                LR      GP2,A           ;RESET PARITY BIT
;
                XS      GP0             ;SAMPLE DATA BIT 7
PAR2:           BP      PAR1            ;BR AROUND PARITY TOGGLE IF 0
                DS      GP2             ;TOGGLE PARITY BIT
PAR1:           DS      GP1             ;DECREMENT THE BIT COUNT
                BM      PAR3            ;STOP WHEN NEGATIVE
                SL      1               ;SAMPLE NEXT DATA BIT
                BR      PAR2            ;LOOP
;
PAR3:           LIS     1               ;TEST PARITY BIT
                NS      GP2             ;ZERO FOR EVEN
                BNZ     RCVIN7          ;BR AROUND ERROR SET IF ODD
;
                LR      A,FLAGS         ;PICK UP FLAG REG
                OI      MSKPTE          ;SET PARITY ERROR
                LR      FLAGS,A         ;
RCVIN7:         LR      A,GP0           ;STRIP PARITY BIT FROM DATA
                NI      H'7F'           ;SEVEN BIT DATA ONLY
                LR      GP0,A           ;
;
RCVIN6:         PI      WRRXF           ;WRITE RECEIVER FIFO SUB
                BP      RCVIN8          ;BR AR'ND ERR SET IF FIFO NOT FULL
;
                LR      A,FLAGS         ;OVRN ERROR GETS SET
                OI      MSKORE          ;
                LR      FLAGS,A         ;
;
                BR      RCVIN8          ;GO RETURN
;
RCVIN3:         INS     PORTE           ;MOST OF 10 BITS IN E
                SL      1               ;ALIGN DATA
                LR      GP0,A           ;TEMP STORE IN GP0
                INS     PORTF           ;PICK UP BIT 0
                SR      4               ;SHIFT IT FROM BIT 7 POS'N
                SL      1               ;
                SR      4               ;SHIFT RIGHT 7
RCVIN9:         XS      GP0             ;MERGE IT WITH GP0 DATA
                LR      GP0,A           ;
                BR      RCVIN5          ;BACK IN LINE WITH DATA
;
RCVIN4:         INS     PORTE           ;GET BULK OF DATA
                SL      1               ;
                SL      1               ;ALIGN IT
                LR      GP0,A           ;TEMP STORE
                INS     PORTF           ;BITS 0 & 1 IN I
;(CONTINUED ON NEXT PAGE)
```

```
                SR      4               ;MOVE IT OVER
                SR      1               ;
                SR      1               ;SHIFT RIGHT 6
                BR      RCVIN9          ;COMBINE CODE
;
;
;
RCVIN8:         LISU    RXSCRU          ;POINT TO SCR AREA
                LISL    RXSCRL+7        ;
                LR      A,D             ;RESTORE GP2
                LR      GP2,A           ;
                LR      A,D             ;GET GP1 CONTENTS
                LR      GP1,A           ;
                LR      A,D             ;GET GP0 CONTENTS
                LR      GP0,A           ;
                DI                      ;SYSTEM INTERRUPTS ARE NOW OFF!
;
                LR      P,K             ;RESTORE INT RETURN ADDRESS
                LR      A,D             ;GET K CONTENTS
                LR      KL,A            ;
                LR      A,D             ;
                LR      KU,A            ;
                LR      A,D             ;GET PROC STATUS
                LR      JR,A            ;
                LR      W,J             ;RESTORE PROC STATUS
                LR      A,D             ;GET ACC VALUE
                LR      JR,A            ;TEMP STORAGE IN J
                LR      A,S             ;ISAR LAST
                LR      IS,A            ; RESTORE IT
                LR      A,JR            ; RESTORE THE ACC
                EI                      ;RESTORE INTS
                POP                     ; AND RETURN
;
;
```

EXTERNAL HARDWARE TO LIGHTEN CONCERN WITH REAL TIME

It is evident that with the example given for full-duplex async, Baud rates will be restricted to less than 600 Baud to ensure adherence to the 3% phase jitter restriction. 300 Baud full-duplex async systems are quite common, and the system described in the example would more than satisfactorily handle the requirements. To operate at higher Baud rates, an alternate approach uses off-chip hardware to build up a parallel-to-serial converter to handle the async transmit function. This solution involves two four bit counters and two eight bit parallel-to-serial converters. The cost effectiveness of this solution should be compared with a third alternative, which uses an off-chip UART.

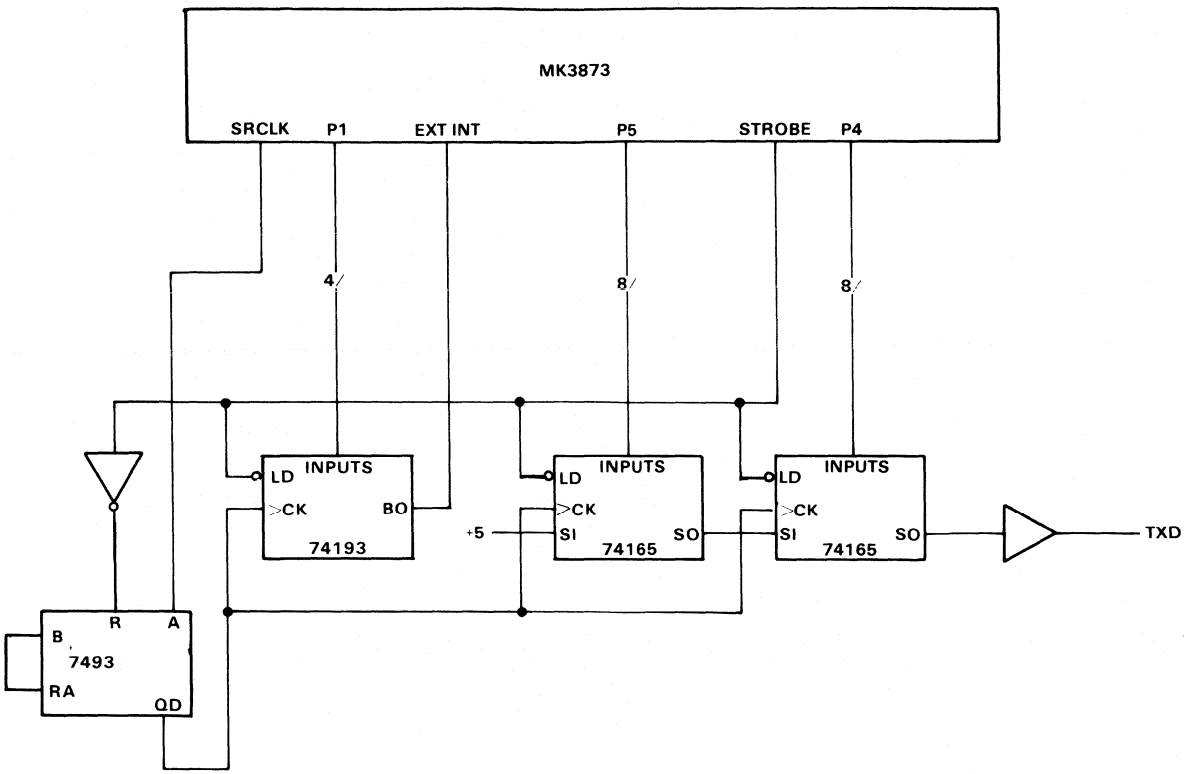
Figure 14 shows the off-chip shift register hardware. The software would be quite straight forward and so is not included in this application note. High Baud rates would be achievable under this approach, owing to the fact that half of a serial channel is added in external hardware. The first counter is used to divide the clock pulses from SRCLK, while the second counter counts down the word length. The EXT INT input is used to signal the MK3873 when the sixteen bit data shift register is empty. The shift register is loaded analogously to the MK3873 serial port transmit holding register, comprised of port E and port F. This approach removes the tight restriction for timing that was placed upon the receiver code in the previous example.

The external hardware could be implemented with the following readily available components:

- 7493 16X Divider
- 79193 Word Length Counter
- 74165(2) Shift Registers
- 7404 Inverter

The hardware arrangement emulates the MK3873 serial port to some extent. The divide by sixteen circuit is reset each time new data is loaded. The end-of-word condition is used to generate the external interrupt. It is necessary to

ASYNC FULL-DUPLEX TRANSMIITER HARDWARE DIAGRAM
Figure 14



add start, stop and parity bits to the two data ports, 4 and 5. The Baud rate of the transmitter will track that of the receiver, without the interdependence of phase which comes about within the serial port.

The real time available to the remainder of the MK3873 using this scheme is generous up to over 9600 Baud.

SUMMARY AND CONCLUSIONS

This application note described the means of dealing with the MK3873 serial port in full-duplex applications. It was shown that while the sync mode full-duplex could be handled entirely within the serial port, special considerations were required for async mode full-duplex. The example given for sync full-duplex was effective for automatically generated and checked parity at data rates of up to 4800 Baud. By using an externally supplied clock source, there was evidence that rates up to 7200 Baud could be adequately supported.

Async applications for full duplex were developed using the serial port of the MK3873 for receive only. Software and hardware methods were described to handle the transmitter, while the Baud rate was still set by the common SRCLK source. The SRCLK source could be provided either internally or externally. The software approach to full-duplex async did not permit a very high Baud rate because of the interference from the receiver interrupts contributing to the phase jitter of the interrupt triggered transmit data. The interference effect was minimized by saving context within the receive service routine and then re-enabling system interrupts during the routine. The maximum Baud rate using software implementation of sync full-duplex was still less than 600 Baud, in keeping with the 3% maximum allowable phase distortion criterion.

The hardware implementation allowed substantially higher data rates for async full-duplex, but the added cost could be estimated to be approaching four dollars, or the same order as an off-chip UART might cost, which could handle both receive and transmit.

MOSTEK®

CONTROLLING THE MK3873 SERIAL PORT

Application Note

INTRODUCTION

The computer industry has evolved generation after generation of architecture, packaging, and fabrication technology for computer components. The State of the Art has moved from computers that filled rooms, requiring wizards to program, to computers on a single chip that can be programmed through human engineered, high level languages. Costs, likewise, have gone from multi-million dollar figures down to the neighborhood of a few hundred dollars for complete computers.

The Mostek 3870 Family of Microcomputer components are representative of the advancement of the State of the Art. This collection of components gives the system designer the building blocks with which he or she may construct very low cost solutions to computing and control problems in areas never before considered suitable applications for computers. As a result, device controllers are becoming more intelligent, main CPUs are relieved of mundane tasks and can concentrate more on computation, and, in general, intelligence is becoming more and more distributed. In addition, it is becoming more realistic to put computers into small, lightweight, inexpensive packages to handle remote communications, control, and monitoring tasks.

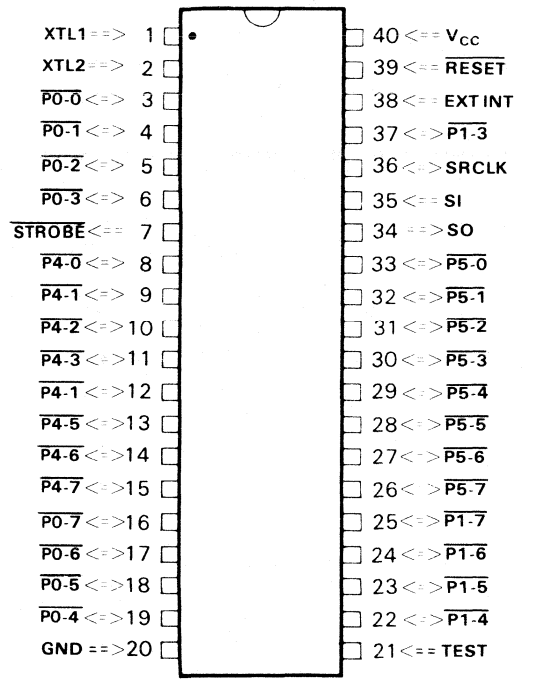
The 3870 Family single-chip microcomputers contain ROM, RAM, parallel I/O, a versatile timer, a clock oscillator, and an external interrupt. One member of the family, the MK3873, has a serial I/O port. EPROM versions are available, which greatly simplify development.

THE MK3873

(Refer to Figure 1)

This application note covers, in detail, the control aspects of the MK3873 serial port. The MK3873 is a special member of the 3870 family because it has a USART mapped into its regular I/O space. The USART, or Universal Synchronous/Asynchronous Receiver/Transmitter, is a device to convert the bytes that are transferred between registers in the computer, into serial bit streams outside the computer. Data within the computer is shifted out by the USART through the serial output pin, SO. Serial data is shifted into the computer by the USART through the serial input pin, SI. A clock pin, SRCLK, is available for external synchronization of the serial data. SRCLK may be programmed to be an input or an output.

PIN CONNECTIONS



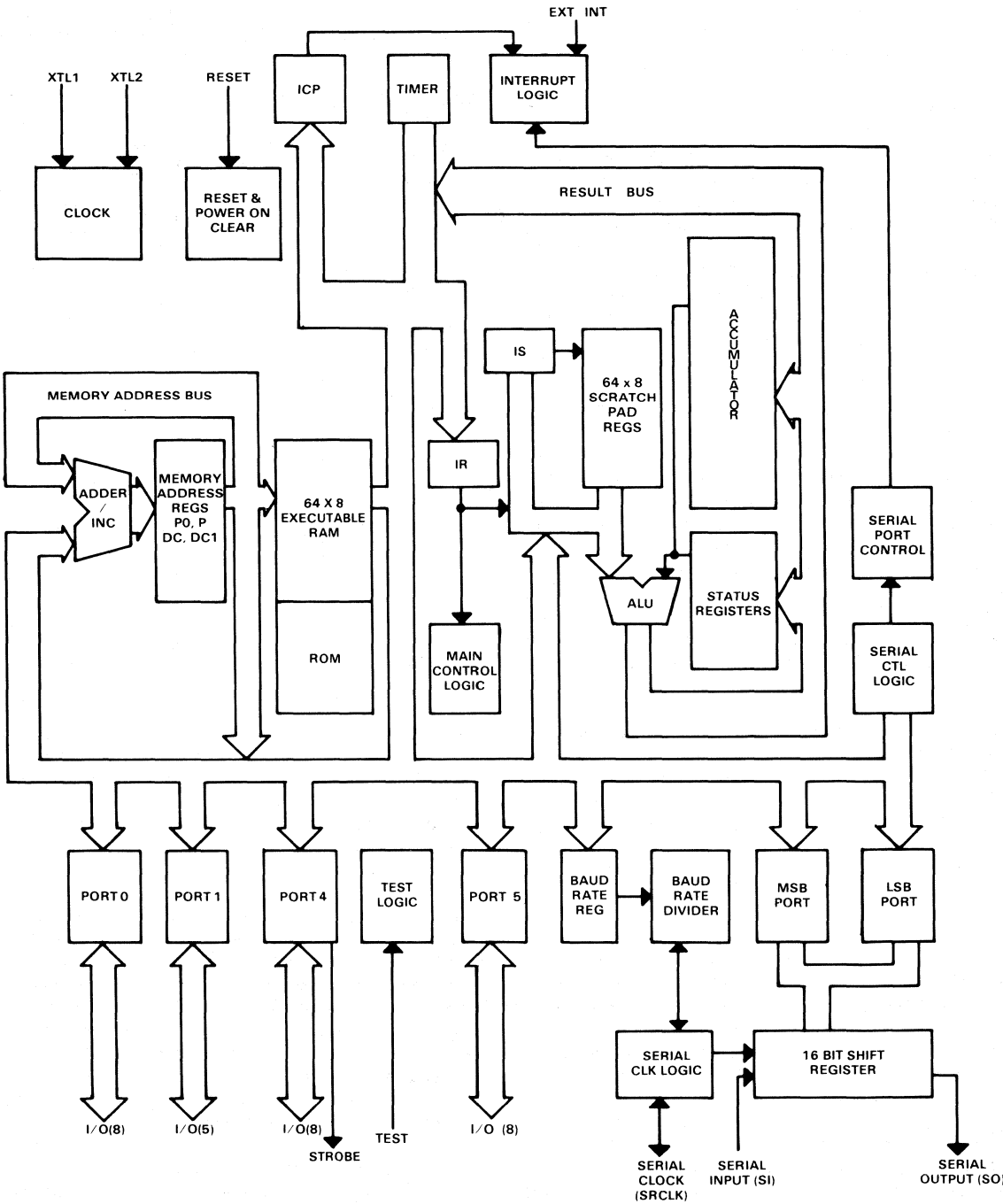
The serial port is a simple extension of the MK3870 I/O system. Control of the serial port is accomplished through inputs and outputs to a set of four I/O port addresses. The serial port functions as a USART which has the capability to interrupt the CPU when enabled, upon receiving serial inputs or upon completing serial outputs.

The MK3873 has many diverse applications areas. It has obvious applicability to data communications, as a terminal or a line adaptor. It can be used in many types of test equipment, since the serial port may be configured in sync or async, with several different word length options and Baud rates: for example, a waveform generator, a test pattern synthesizer, or line monitor. It can also be used to advantage as a cassette tape controller, a serial printer intelligent interface, a remote data logger, or even an intelligent switching power supply controller.

The EPROM version of the MK3873 is called the MK38P73, and has a socket for a "piggy-back" EPROM. This part may be used for development or for low volume applications, such as laboratory test equipment. The MK38P73 behaves

MK3873 SYSTEM BLOCK DIAGRAM

Figure 1



like and has the same pin-out as the MK3873, and thus may also be used for prototyping.

DESCRIPTION OF THE SERIAL PORT

This section deals with the hardware of the MK3873 serial port. Detailed descriptions include block diagram and logic drawings. The diagrams are intended as functional equivalents and are not meant to be indicative of the actual design to the circuit level.

ARCHITECTURAL OVERVIEW

The architectural discussion breaks the serial port down into the following sections, (refer to architectural diagram in Figure 2).

SERIAL PORT I/O DECODER. Decodes signals from the CPU to provide access gating within the serial port.

BAUD RATE DIVIDER

Provides basic frequency of operation for serial transmission. It also controls direction of I/O pin, SRCLK, depending on programmed Baud rate value.

START/STOP SYNCHRONIZER AND SHIFT CLOCK DIVIDER. Generates 1X or 16X shift clock, depending on the value of the SYNC bit in the serial port control register. It also provides edge detection and synchronization of the 16X clock to received async characters.

DATA SERIALIZATION AND OUTPUT CONTROL. Includes the shift register, receive and transmit holding registers, and serial output enable circuit, which handle storage and shifting of serial data.

SERIAL PORT SEQUENCING CIRCUITS. Contains word length counter, READY and ERROR flagg, and interrupt circuits.

The architectural drawing shows three squares that represent pads, or pins that connect to the outside of the chip. The pads are labelled, SI, SO, and SRCLK. The labels stand for Serial Input, Serial Output, and SeRial CLocK, respectively. Control and access from the CPU are shown as CPU SIGNALS, entering the I/O Decoder section. The four ports that permit access to the serial port are shown in the sections where they are mainly used. Port C is the Baud rate port. Port D is the main control port. Ports E and F are the data access ports.

Data entering the chip via the pad, SI, is routed to two places: to the shift register, where it is deserialized and stored, temporarily, in the receiver holding register pair, referred to in the figure as RXHOLD; and to the start/stop synchronizer where it is used, in async mode, to initialize the phase of the 16X Baud clock divider, and where the data validation test is done. The CPU places data to be transmitted into the transmitter holding register pair, called TXHOLD in the figure. Then the serial port shifts the data through the shift register, the output control gate, and out the pad labelled SO. The Baud rate clock may be programmed to be generated internally and presented to the pad called SRCLK or, alternately, the Baud rate clock may be programmed to be generated externally, and brought into the chip through SRCLK as an input.

Separately vectored interrupts for transmit or receive may be generated as a result of the word length counter being decremented to zero. Zero signifies an end-of-word condition, which means that new data has arrived from the receiver, or that new data can be loaded for transmitting.

Interrupts may also be generated on each received bit, when in the search mode, so that incoming data can be matched against a synchronizing bit pattern by the firmware.

As shown in Figure 3, reads and writes of the access ports produce access to different information in the serial port. Writes are accomplished through OUT or OUTS instructions, and reads are done using IN or INS instructions. These operations transfer data between the accumulator and the addressed port.

The Baud rate code in port C is written to one of ten possible values, nine of which set internal divide factors, and the tenth, zero, which puts the port into external clock mode. The serial clock frequency, whether it is generated internally or externally, becomes the actual data shift rate in sync mode, but is divided by sixteen to produce the async mode shift clock. Sync mode or async mode is a function of the SYNC bit in port D.

When port D is written, the following information is transferred.

WORD LENGTH CODE. Sets the number of bits in the character, including start and stop bits, if any.

EDGE. Enables synchronization of the 16X shift clock divider with the received data start bit, also allows start bit verification by testing for the continued presence of a zero start bit one-half bit time after the starting edge is detected.

SEARCH. Causes an end-of-word condition to occur on every shift clock, which allows interrupts to be generated and causes the shift register contents to be transferred to the receive holding register after each bit shift.

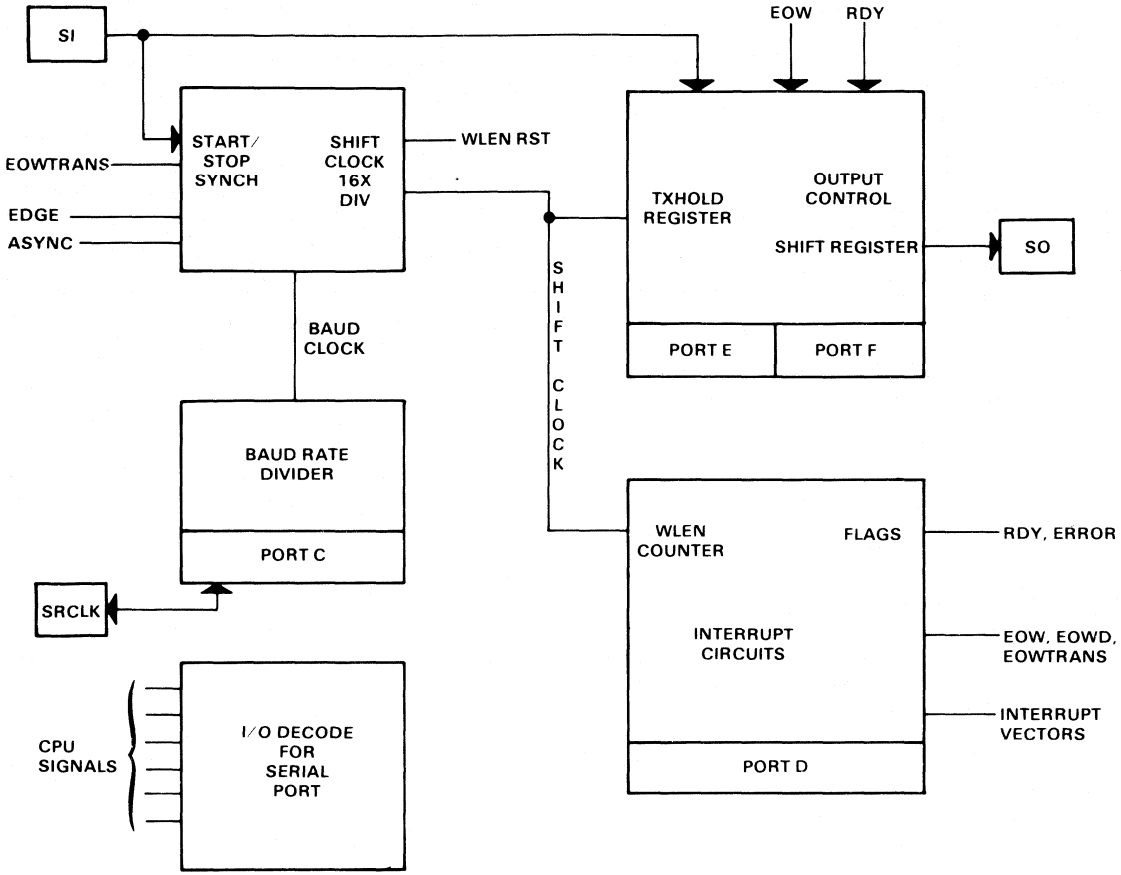
SYNC. Puts serial port in sync mode, shifting data in the shift register at 1X the serial clock, also allows the output control gate to be continuously enabled when in transmit mode.

XMIT. Puts port in the transmit mode, which allows data to be shifted out the serial output pin, SO.

INTS. Allows generation of an interrupt when end-of-

MK3873 SERIAL PORT ARCHITECTURAL DIAGRAM

Figure 2



word condition occurs, provided that the system interrupts are enabled.

The mnemonics on the bits represent the asserted state of the respective function. That is, EDGE being a "1" enables the edge detecting mode, SEARCH enables search mode when set to a "1", SYNC puts port in synchronous mode when a "1" and asynchronous mode when a "0", XMIT results in transmit mode when a "1" and receive mode when a "0", and INTS enables interrupts when "1" and disables them when "0".

Reading port D accesses two bits of status information.

READY and ERROR. READY becomes set to a "1" whenever an end-of-word condition occurs. ERROR is set to "1" whenever an end-of-word condition occurs while READY is still true. READY is reset to "0" by doing a read or write of either the receive holding register or the transmit holding register. ERROR is reset to "0" whenever port D is read.

The receive and transmit holding registers are each sixteen bits long. These registers "shadow" the shift register and exchange information with it at end-of-word time. In the transmit mode, when an end-of-word condition occurs, the entire contents of the shift register is loaded into the receive holding register and the contents of the transmit holding register is loaded into the shift register at the same time.

In the receive mode, the transfer from the transmit holding register is inhibited, and only the transfer to the receive holding register takes place. In both receive and transmit modes, when the end-of-word transfer occurs, the READY flag in port D is set and an interrupt will occur if enabled. Reading or writing either half of the receive or the transmit holding register resets the READY flag to "0".

MK3873 SERIAL PORT ACCESS I/O PORTS

Figure 3

	7	6	5	4	3	2	1	0
PORT C - WRITE	-	-	-	-	BAUD RATE CODE			
READ	-	-	-	-	-	-	-	-

	7	6	5	4	3	2	1	0
PORT D - WRITE	WORD LENGTH		CODE	EDGE	SEARCH	SYNC	XMIT	INTS
READ	READY	ERROR	-	-	-	-	-	-

	7	6	5	4	3	2	1	0
PORT E - WRITE	TRANSMIT HOLDING REGISTER				-		MSB	
READ	RECEIVE HOLDING REGISTER				-		MSB	

	7	6	5	4	3	2	1	0
PORT F - WRITE	TRANSMIT HOLDING REGISTER				-		LSB	
READ	RECEIVE HOLDING REGISTER				-		LSB	

PORT D WORD LENGTH	NUMBER OF BITS IN WORD
0	4
1	7
2	8
3	9
4	10
5	11
6	12
7	16

Crystal Frequency = 3.6864 MHz

PORT C VALUE	DIVIDE FACTOR	SHIFT RATE (Hz) SYNC	SHIFT RATE (Hz) ASYNC	STANDARD VALUE	PERCENT ERROR(%)
00	----	External Mode		-	-
01	----	- Not Used -		-	-
02	----	- Not Used -		-	-
03	3072	1200	75	Both	0.00
04	2096	*1759	110	Async	-0.07
05	1536	2400	150	Both	0.00
06	768	4800	300	Both	0.00
07	384	9600	600	Both	0.00
08	196	19200	1200	Both	0.00
09	96	38400	2400	Both	0.00
10	48	76800	4800	Both	0.00
11	24	153600	9600	Async	0.00
12	----	- Not Used -		-	-
13	----	- Not Used -		-	-
14	----	- Not Used -		-	-
15	----	- Not Used -		-	-

*NOTE: 1759 Hz is within 2.3% of 1800 Hz, which is a standard baud rate.

SERIAL PORT I/O DECODER

(Refer to Figure 4)

This section of the serial port circuitry is used to decode signals from the CPU and timing sections of the MK3873. The decoder detects the fact that INS or IN and OUTS or OUT instructions are being executed with port addresses of C, D, E, or F. This section also detects the functioning of the chip reset sequence, which operates following a power-up or activation of the chip RESET pin. The reset decode is used to initialize the output control, the port D register, and the READY and ERROR flags. The CPU interrupt acknowledge sequence is also detected to generate the signals FREEZE, IADRU, and IADRL. These signals are uses to acquire the vector and sequence the interrupt circuitry.

BAUD RATE DIVIDER

The Baud rate divider is outlined in Figure 5. This section includes the Baud Rate Code register, Baud Rate count modulo ROM, variable modulo counter, Internal serial clock driver, the SRCLK pad, and the Positive Edge Detector.

The lower four bits of the accumulator are loaded into the Baud Rate Code register using the LOAD "C" signal from the I/O decoder. The Baud Rate code values are listed in the table in Figure 3. Some of the values are not supported. The Baud Rate Count Modulo ROM translates the four bit codes from the Baud Rate Code Register into values for use in the counter. The ROM also decodes input values that are non-zero to enable the SRCLK driver as an output. A zero code results in the enable bit being zero, allowing the pad SRCLK to be driven from an external source.

The Variable Modulo Counter is a nine-bit Shift Register Generator, or SRG, that detects a match of its inputs with the internal states to determine its count cycle. The output is at the rates listed in the table in Figure 3.

The SRCLK driver is an amplifier that can be enabled to drive up to 3 standard TTL loads. When the enable is low, the driver enters a high impedence state which allows an external source to drive the SRCLK pin as an input.

The positive Edge Detector is used to generate a standard width, low going pulse, in response to a rising edge at SRCLK. This permits the chip to function even at extremely low SRCLK rates, since the logic is only dynamic while the pulse is low and static otherwise.

The outputs of this section may be driven externally or internally. The two outputs of the Baud Rate Divider section are SERIAL CLOCK, which is a low going pulse string, and SCLK, which is the same as what appears on pin, SRCLK. These outputs are presented to the Data Serialization section and to the Serial Port Sequencing section.

START/STOP SEQUENCER AND SHIFT CLOCK DIVIDER. The Start/Stop Sequencer and Shift Clock Divider section, illustrated in Figure 6, performs three functions: 1), it provides the Shift Register and Word Length Counter with a shift clock; 2), in async receive mode it produces a signal, INIT, to hold the Word Length Counter initialized until an incoming character arrives; and 3), it validates the start bit in async receive mode by sampling its value one-half bit time after its leading edge is detected to insure that it is a zero.

Figure 6 has three parts. The upper part shows the synchronizer, the middle contains the 16X clock divider, and the bottom part contains a sequence diagram for events within the section. The synchronizer is only used in the async receive mode, which is denoted by an active state of the port D signal, EDGE. The 16X clock divider is used in async transmit and receive mode, since its output is not selected for sync mode. The sequence diagram covers the synchronizer actions for async receive mode.

The synchronizer is enabled by EDGE being high, and SYNC being low. The signal that makes up the third input to the gate labelled "3", called EOWTRANS for end-of-word transfer, is generated in the Word Length Counter section, and goes active true after the last bit in the character is shifted into the shift register. The signal ST7, for state 7, is decoded from the 16X clock divider in the middle of the figure. An output to port D puts the synchronizer into the state shown in the portion of the sequence diagram under the encircled letter A.

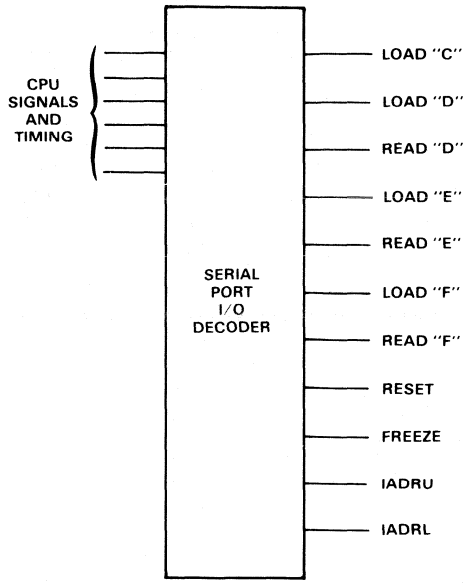
The signal LOAD "D" is derived from the output to port D in the I/O Decoder section. LOAD "D" initializes the logic by presetting FFA to the set state, resetting FFB to the cleared state, and issuing an INIT signal to the Word Length Counter via the OR gate labelled "2".

The serial data input, SI, is inactive in a marking state, which corresponds to a logical "1", or high. Examination of the sequence chart shows that after being initialized by LOAD "D", the states shown under circle A exist. Since the INIT signal is inactive, or low, the divide counter is able to run, eventually producing an ST7 signal. The ST7 signal clears FFB, but leaves FFA unaltered, since SI being high disables gate "6". FFB going set, FFA remaining set, and SI being high results in gate "1" being enabled, producing the INIT signal. INIT resets the divide counter, stopping further activity.

The next event that takes place is that a data character arrives, placing a zero start bit on SI. This starts the events shown under circle B in the sequence diagram. The low state of SI, coupled with a low on ST7 and FFA being set, puts a high on the clear of FFB, resetting it. With FFB cleared, INIT is removed, allowing the counter to count up to state ST7. ST7 does two things at this time, under circle B: 1), it will clear FFA, but only if SI is still low; 2), it unconditionally sets FFB. When FFB was set by ST7, if SI had returned to a high, as might happen with noise, INIT would have been reasserted, returning to logic to its "armed" state, as in circle A. But since SI was still low under

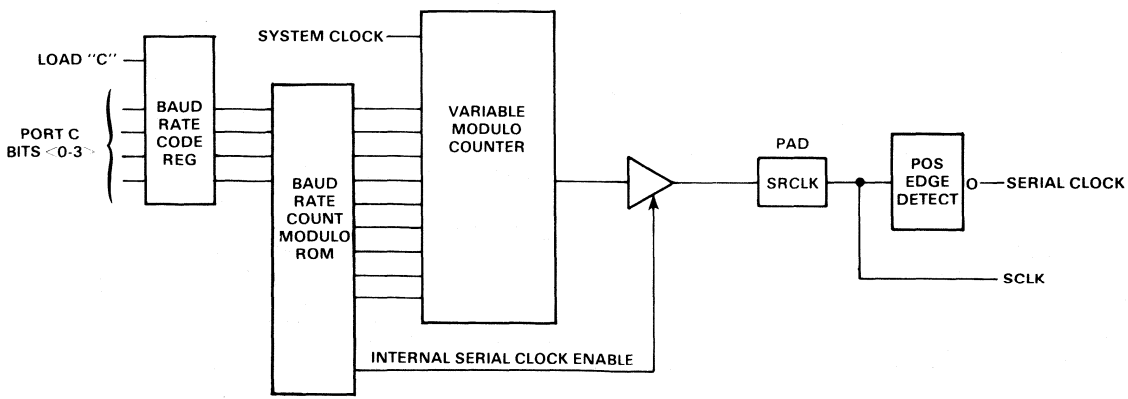
SERIAL PORT I/O DECODER FUNCTIONAL DIAGRAM

Figure 4



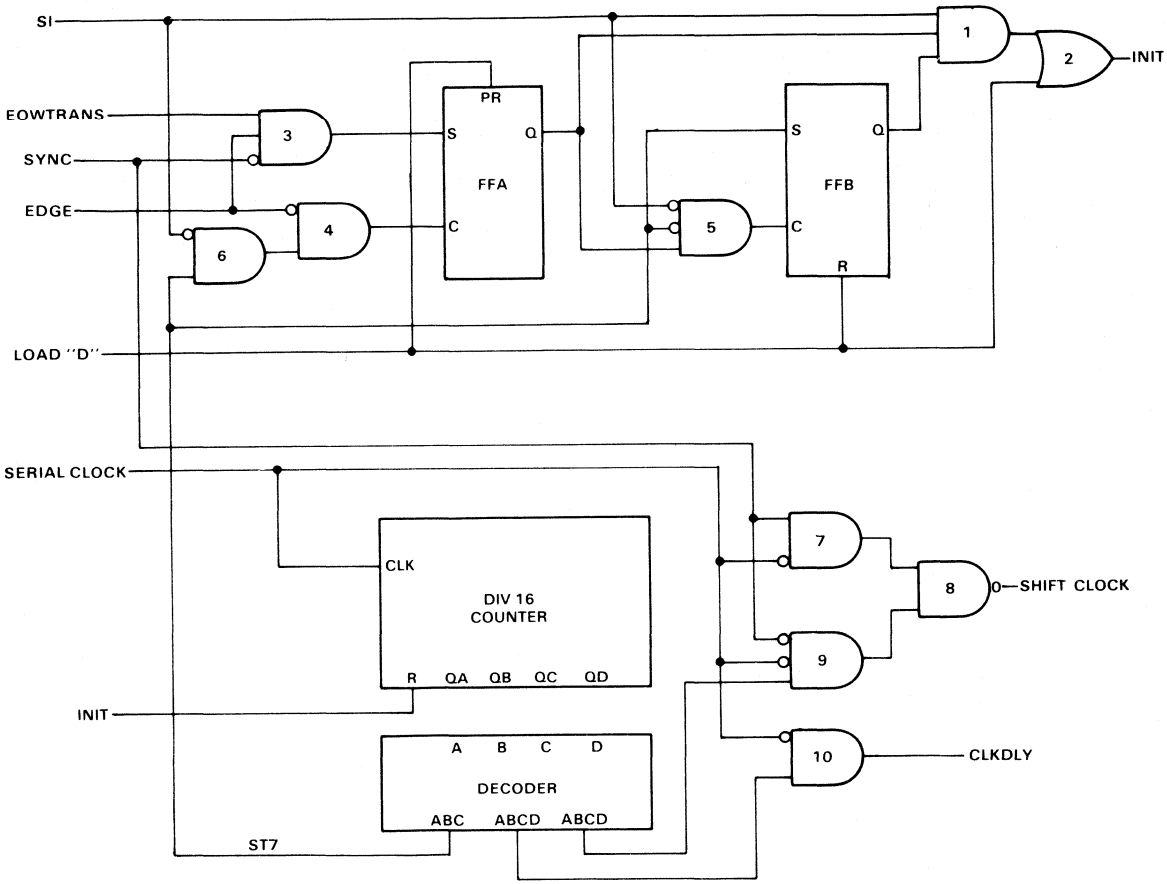
SERIAL PORT BAUD RATE DIVIDER FUNCTIONAL DIAGRAM

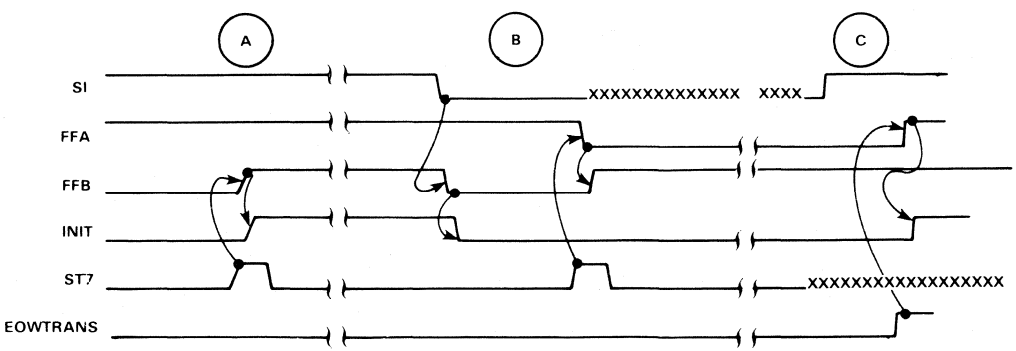
Figure 5



START/STOP SYNCHRONIZER FUNCTIONAL DIAGRAM

Figure 6





the circle B, FFA was cleared, removing the enable on gate "1", and allowing the divider counter to produce additional shift pulses.

The portion of the sequence diagram under circle C shows how the end-of-word condition is handled by this section. When EOWTRANS becomes true, the set input on FFA is made high, setting FFA. When FFA is set, if SI has a stop bit of "1", INIT will be asserted, readying the synchronizer for the next received character. Thus, it can be seen that the synchronizer is capable of restarting immediately after shifting in the last bit of the character.

The decoder shown in the middle of the figure produces three outputs: ST7, which is the AND of the first three counter bits, ABCD = 111X; ST7 Q4, used to signal the low going edge of the 16X shift clock, ABCD = 111; and ST7 Q4, used to signal the high going edge of the 16X shift clock, ABCD = 1110. The pulse for the high going edge of the shift clock occurs one quarter of the way through the counter cycle, which is started with the leading edge of the first data bit. The resulting phase causes the incoming data to be sampled precisely in the middle of the bit times. This results in an optimum strobing relative to phase jitter rejection.

The selector made up of gates "7", "8", and "9" is controlled by SYNC, which selects the SERIAL CLOCK when in sync mode, and selects the 16X divider clock when SYNC is false. The output of the selector is inverted to maintain a shift clock that is low during the low duration of SERIAL CLOCK to preserve the protection of the dynamic circuitry provided by the positive edge detector function in the Baud Rate Divider section. Gate "10" produces an output that is active at a point three quarters of the way through the count cycle, for use in the Transmitter Output control section.

DATA SERIALIZATION AND OUTPUT CONTROL

This section centers on the sixteen bit shift register, through which all serial data passes. Figure 7 shows the registers and control circuitry in the section. DB0 through DB7 correspond to the MK3873 data bus.

The SHIFT signal originates in the Start/Stop Synchronizer section, and is used here to clock most of the logic in this section. EOW is the signal that identifies an end-of-word condition, and is generated in the Word Length Counter. EOW is used here to transfer data from the shift register to the receive holding register, and from the transmit holding register to the shift register. XMIT comes from the port D register in the Serial Port Sequencing section. XMIT is used here to enable loading of the shift register from the transmit holding register with EOW. XMIT is also used in the output control gating near the bottom of the figure. SI is the serial input data. SI is directed into the shift register for data deserialization, and to the first bit of the receive holding register to allow simultaneous shifting and loading of the receive holding register. The LOAD "X" and READ "X" signals originate at the I/O Decoder section and are used to access the holding registers.

The receive holding register, made up of 8-bit latches labelled RXHOLD UPPER and RXHOLD LOWER, are shown skewed from bit-to-bit correspondance with the transmit holding registers, TXHOLD, by one bit. This skew is because of the relationship between the RXHOLD and SI, as compared with the relationship between TXHOLD and SO. Data is actually shifted and loaded into RXHOLD simultaneously.

Data from TXHOLD going broadside into the shift register is directly analogous to a shift; thus for simultaneous events, TXHOLD bit 15 goes into SHIFT REGISTER bit 15 at the same time as SI goes into RXHOLD bit 15 and SHIFT REGISTER bit 15 goes into RXHOLD bit 14. Likewise, TXHOLD bit 0 goes into SHIFT REGISTER bit 0 at the same time as SHIFT REGISTER bit 1 goes into RXHOLD bit 0, and SHIFT REGISTER bit 0 does not load into any register.

FFC, shown below the shift register, follows the data in SHIFT REGISTER bit 0 by half of a bit time. SYNC is used in the data selector to pick CLKDLY for async mode, and SCLK for sync mode, to use in clocking the delay flip-flop, FFC. SCLK is the unaltered version of the serial clock presented at the pin, SRCLK. The low phase of SCLK holds the correct timing for the half bit time delay when in sync mode. CLKDLY, generated in the 16X clock divider circuit, contains the equivalent timing for the async case.

FFD is used to force the output, SO, to a high, or marking, state. When FFD is reset, gate "11" is forced high. When FFD is set, the force is released and data from FFC is allowed to pass to the output pin. FFD is reset when the port is not in transmit mode, and during an MK3873 reset sequence, via the OR gate labelled "15". During transmit mode, the force is not released by FFD until the first EOW condition occurs, as seen by gates "13" and "12". If the port is in the sync transmit mode, the output will remain enabled. In async transmit mode, however, if an underrun is allowed to take place, gate "14" will clear FFD, disabling the output until new data is loaded to clear READY.

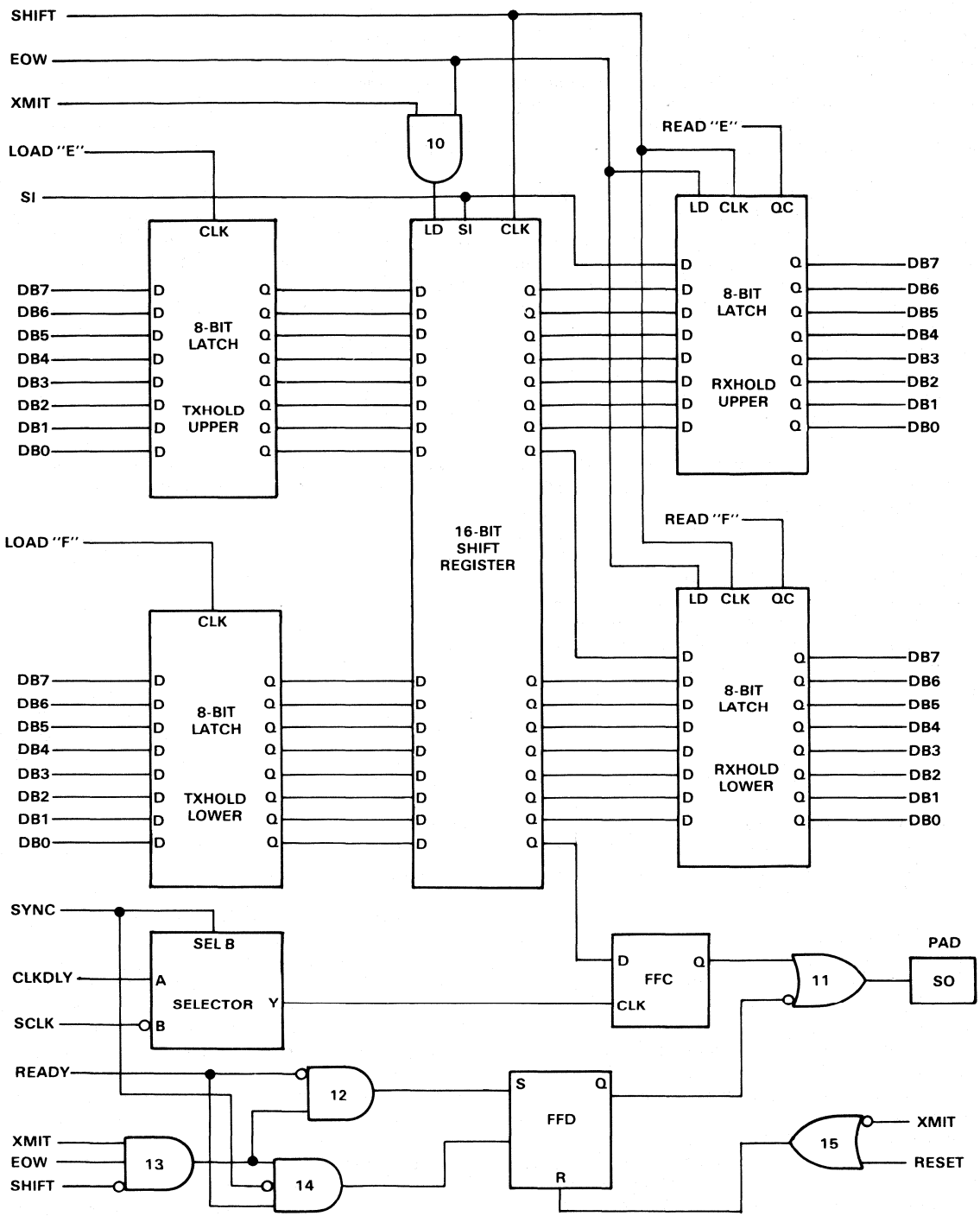
SERIAL PORT SEQUENCING CIRCUITS

This section discusses the Word Length Counter, the READY and ERROR flags circuit, and the Interrupt circuit. The Word Length Counter counts shift pulses, signals the end-of-word condition, and is illustrated in Figure 8. The READY and ERROR flags respond to the end-of-word signals, as does the interrupt circuit. The READY and ERROR flags circuit and the Interrupt circuit are illustrated in Figure 9.

Figure 8 also includes a sequence diagram that shows what events take place before, during and after the end-of-word condition. It may be beneficial to refer to this sequence diagram during the discussion of this section.

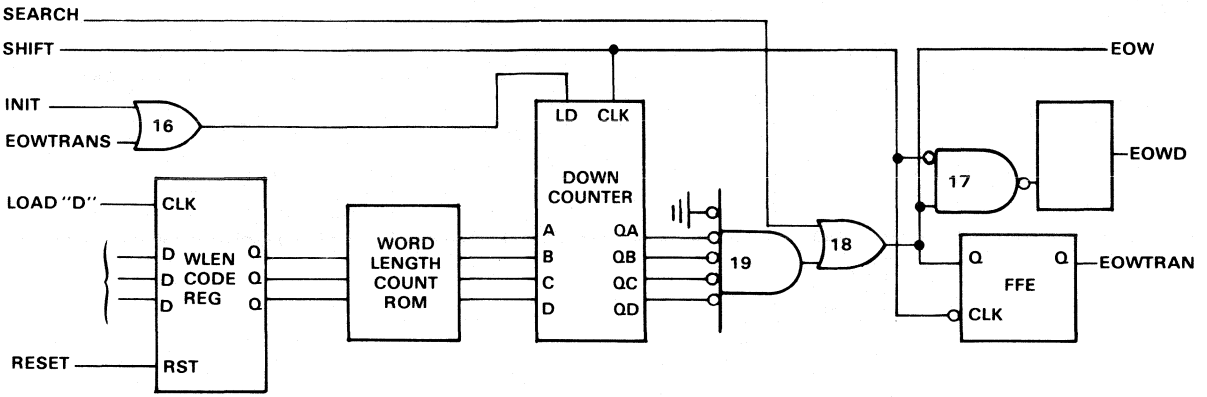
DATA SERIALIZATION FUNCTIONAL DIAGRAM

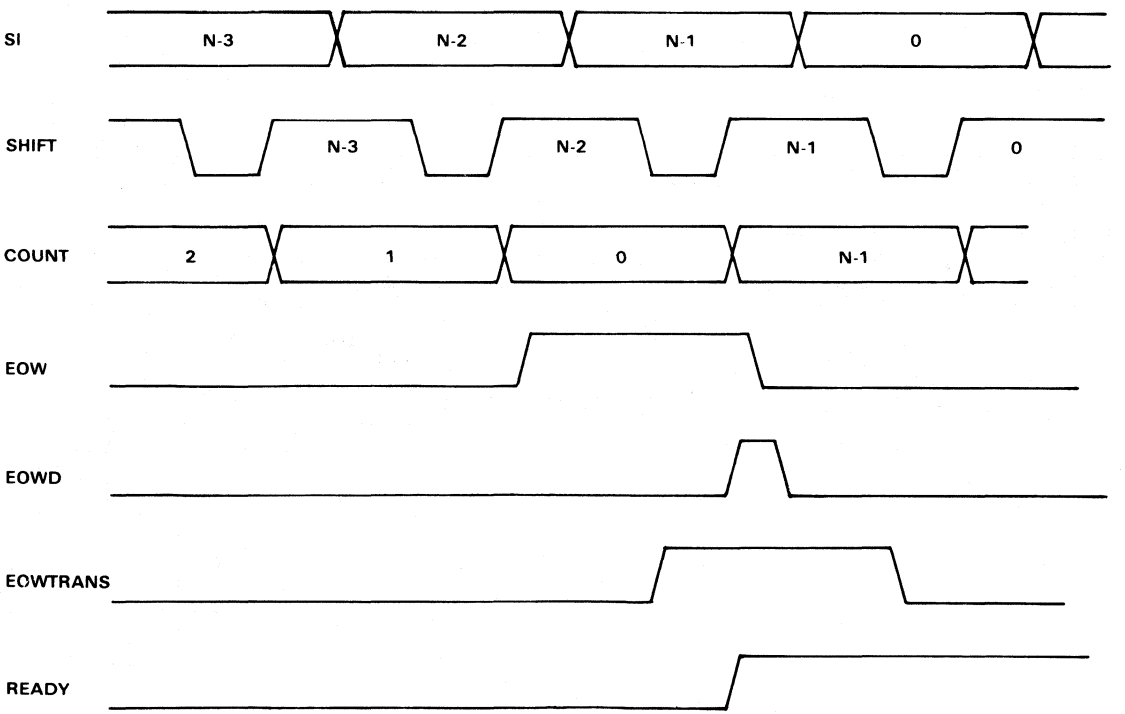
Figure 7



WORD LENGTH COUNTER FUNCTIONAL DIAGRAM

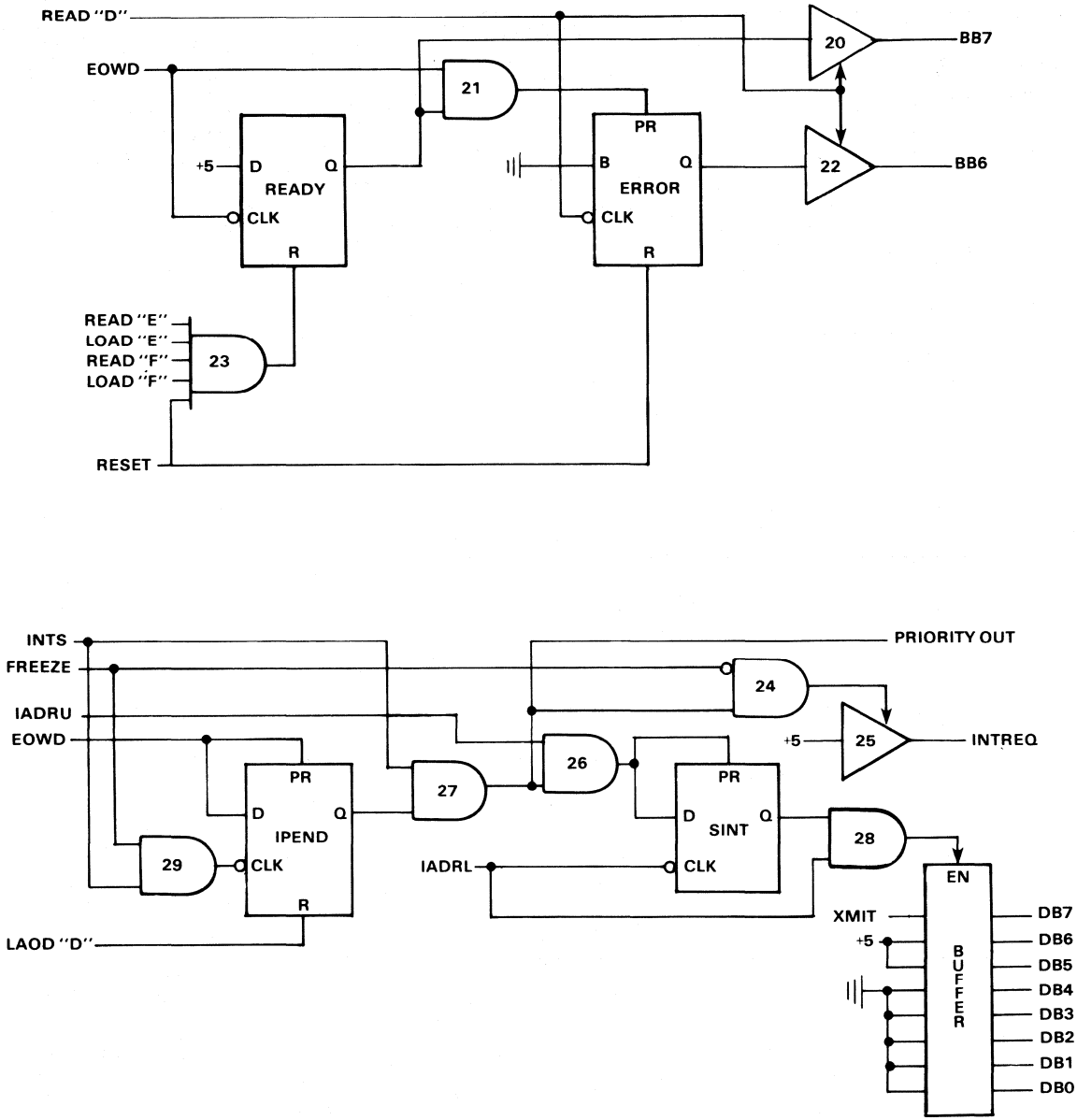
Figure 8





FLAGS AND INTERRUPTS FUNCTIONAL DIAGRAM

Figure 9



The Word Length Counter consists of circuitry similar to that of the Baud Rate Divider in Figure 5. The upper three bits of the data bus during a LOAD "D" are saved in the WLEN CODE Register, shown in Figure 8. The WLEN CODE values have corresponding lengths that are listed in the table in Figure 3. The mapping represented by the table is done by the WORD LENGTH COUNT ROM. This ROM provides counter load values to the 4-bit DOWN COUNTER. The DOWN COUNTER is allowed to count only when its load gate "16" is low. Gate "16" is low when the OR of the INIT signal, from the Start/Stop Synchronizer, and the EOWTRANS signal is false. When the DOWN COUNTER counts all the way down to zero, then gate "19" will go high, signalling the end-of-word-condition. When SHIFT subsequently goes low, FFE will become set, activating EOWTRANS.

SHIFT being low while EOW is true also arms the Positive Edge Detector via gate "17". When SHIFT again returns high, the Positive Edge Detector pulses the signal called EOWD, or EOW delayed. EOWD is the triggering signal for the READY and ERROR flags, and for the Interrupt circuit.

As shown in the sequence diagram, the SHIFT pulses so high centered between the transitions of SI. These high-going pulses clock the value of SI into the Shift Register, and also decrement the DOWN COUNTER. Notice that the state of the counter reaches zero after the shift pulse that clocked input bit number N-2. The count going to zero caused EOW to be asserted. The next shift pulse occurs while EOW is asserted and clocks SI bit number N-1, the last bit, into the shift register. While the last bit is being shifted into the shift register, it is also simultaneously being shifted into the receive holding register, RXHOLD bit 0. Immediately after shifting the final bit, EOWD is generated, setting the READY flag and pulsing the Interrupt circuitry.

EOWTRANS wraps back around to reload the DOWN COUNTER when the end-of-word condition occurs. INIT from the Start/Stop Synchronizer also causes loading of the DOWN COUNTER, essentially resetting the bit count. Recall that INIT is asserted by the signal LOAD "D", as well as by actions of the Start/Stop circuit. Notice, also, that the RESET function places zero in the WLEN CODE Register. This results in defaulting the character length to 4-bits.

Figure 9 shows the operation of the READY and ERROR flag circuits. READY becomes set on the trailing edge of EOWD. If READY is true when EOWD occurs, ERROR will be set. When READ "D" occurs, indicating a read of status, ERROR and READY are read onto the MK3873 data bus. ERROR will be cleared on the trailing edge of READ "D". Gate "23" goes high to reset READY when any one of the inputs goes high. Thus, any read or write of either holding register resets the flag, independent of mode.

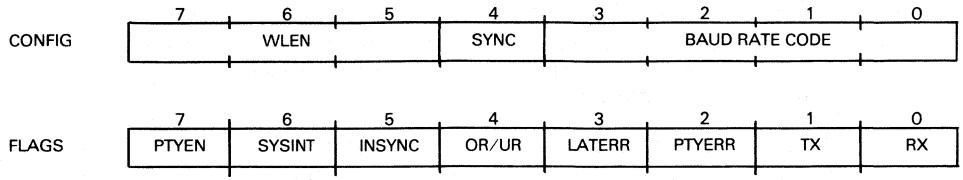
The Interrupt circuitry is also shown in Figure 9. The flip-flop labelled IPEND, for Interrupt PENDing, is unconditionally set on every end-of-word condition, as indicated by EOWD being connected to the preset input of IPEND. The presence of LOAD "D" on the reset input of IPEND assures that an end-of-word condition occuring at a time when INTS was off will not propagate when INTS is subsequently turned on. This is because a LOAD "D" is necessary to alter the state of the serial port interrupt enable, INTS.

When the MK3873 responds to an interrupt, a "freeze" cycle is executed to enable the priority chain to settle. Since the serial port is the highest on the MK3873 interrupt priority chain, there is no PRIORITY INPUT from upstream devices. When the serial port has an enabled interrupt, it asserts the PRIORITY OUT signal to inhibit downstream interrupting devices from putting their vector on the bus when the interrupt address is requested by the CPU. The CPU requests the address using first IADRU and then IADRL. In the MK3873, only response during IADRL is registered by the CPU. In Figure 9, it can be seen that IADRU serves only to latch the interrupt from the serial port into SINT, which allows the serial port vector, Eo (hex) for the transmit mode, and 60 (hex) for receive mode, to be gated onto the bus by IADRL. SINT is cleared by the trailing edge of IADRL.

PROGRAMMING INTERFACE

The information provided in section 2 presented a working background for the MK3873 serial port hardware. This section concerns itself with controlling the serial port from the software point of view. Programming examples are included in the subsequent discussions which may be used in actual MK3873 applications as presented, or which may provide the basis upon which applications may be built.

A configuration register and a flag register are used in the programming examples to hold information about the serial port. In some instances it may also be desirable to retain copies of other information written to the microprocessor ports. The configuration and flag registers are illustrated below.



MNEM	MEANING
WLEN	Length of word including data, start bits and stop bits
PTYEN	Set to enable parity generating and checking
SYNC	Set means sync mode, reset means async
SYSINT	Set when system interrupts are enabled
INSYNC	Set when character sync achieved
OR/UR	Overrun/underrun flag, copy of ERROR bit
LATERR	Latent error, set on serial int with no READY bit
PTYERR	Parity error, set when parity fails
RX	Receive mode
TX	Transmit mode, Full-duplex if RX also set

Initialization of the serial port consists of disabling the output gate, disabling the serial port interrupts, and setting the Baud rate. This is illustrated in the programming example in Figure 10.

INITIALIZING THE SERIAL PORT
Figure 10

```
;       PROGRAMMING EXAMPLE TO INITIALIZE THE SERIAL PORT
;
PORTC:    EQU   H'C'        ;BAUD RATE PORT ADDRESS
PORTD:    EQU   H'D'        ;SERIAL CONTROL PORT ADDRESS
PDEDGE:   EQU   H'10'       ;PORT D EDGE BIT
PDSRCH:   EQU   8           ;PORT D SEARCH BIT
PDSYNC:   EQU   4           ;PORT D SYNC BIT
PDXMIT:   EQU   2           ;PORT D TRANSMIT BIT
PDINTS:   EQU   1           ;PORT D INT ENABLE BIT
PORTE:    EQU   H'E'        ;UPPER DATA PORT ADDRESS
SYNC2:    EQU   H'16'       ;SYNC CHAR 2
PORTF:    EQU   H'F'        ;LOWER DATA PORT ADDRESS
SYNC1:    EQU   H'16'       ;SYNC CHAR1
CONFIG:   EQU   8           ;CONFIGURATION REGISTER ADDRESS
MSKWLN:   EQU   H'E0'       ;WORD LENGTH MASK IN CONFIG
MSKSYN:   EQU   H'10'       ;SYNC BIT IN CONFIG
MSKBRT:   EQU   H'F'        ;MASK FOR BAUD RATE IN CONFIG
FLAGS:    EQU   7           ;FLAG REGISTER ADDRESS
MSKPTY:   EQU   H'80'       ;PARITY ENABLE BIT IN FLAG REG
MSKSYI:   EQU   H'40'       ;PARITY ENABLE FLAG
MSKORE:   EQU   H'10'       ;OR/UR FLAG
MSKKER:   EQU   8           ;LATENT ERROR FLAG
MSKPTE:   EQU   4           ;PARITY ERROR FLAG
MSKTXM:   EQU   2           ;TRANSMIT MODE FLAG
MSKRXM:   EQU   1           ;RECEIVE MODE FLAG
;
INITSP:   DI                ;DELICATE CODE
```

```
                LI      MSKWLN+MSKBRT   ;PRESERVE ONLY WLEN AND BRATE
                NS      CONFIG          ;INIT CONFIG
                LR      CONFIG,A        ;
;
                LI      MSKPTY+MSKSYI   ;PRESERVE ONLY PTY EN AND SYS INTS
                NS      FLAGS           ;INIT FLAG REGISTER
                LR      FLAGS,A         ;
;
                CLR                     ;ZERO TO PORT D
                OUTS    PORTD           ;
                LI      MSKBRT          ;PICK UP BAUD CODE
                NS      CONFIG          ; OUT OF CONFIG REG
                OUTS    PORTC           ;SET THE BAUD RATE
;
                LI      SYNC2           ;PUT IDLE CHAR IN DATA PORTS
                OUTS    PORTE           ;
                LI      SYNC1           ;
                OUTS    PORTF           ;
;
                LI      MSKSYI          ;RESTORE SYSTEM INTS
                NS      FLAGS           ;NON-ZERO IF SET
                BZ      INISP1          ;DON'T ENABLE IF ZERO
                EI                      ;ELSE ENABLE SYSTEM INTS
INISP1:         POP                     ;RETURN TO CALLER
```

ASYNC OPERATION

This section describes operation in async mode, elaborating on details of what physically happens in the serial port.

A routine used in both sync and async, receive and transmit mode, is one which generates parity over a data character. The same routine, PARITY, is used to generate as well as to check parity. Figure 11 illustrates the routine.

COMPUTING PARITY

Figure 11

```
;
;   ROUTINE TO COMPUTE PARITY
;   ROUTINE USES GP1 AND GP2 AND COMPUTES PARITY OVER
;   THE CONTENTS OF GP0; NEGATIVE RETURN STATUS MEANS
;   ODD PARITY, POSITIVE STATUS MEANS EVEN PARITY.
;
GP0:    EQU     0               ;GENERAL PURPOSE REGISTER 0
GP1:    EQU     1               ;GENERAL PURPOSE REGISTER 1
GP2:    EQU     2               ;GENERAL PURPOSE REGISTER 2
;
PARITY: DI                      ;PROTECT RETURN ADDRESS IN P1
        LIS     7               ;BIT COUNT
        LR      GP1,A           ;
        CLR                     ;PARITY IN LSB OF GP2
        LR      GP2,A           ;RESET PARITY BIT
;
        XS      GP0             ;SAMPLE DATA BIT 7
PAR2:   BP      PAR1            ;BR AROUND PARITY TOGGLE IF 0
        DS      GP2             ;TOGGLE PARITY BIT
PAR1:   DS      GP1             ;DECREMENT THE BIT COUNT
        BM      PAR3            ;STOP WHEN NEGATIVE
        SL      1               ;SAMPLE NEXT DATA BIT
        BR      PAR2            ;LOOP
;
PAR3:   LI      MSKSYI          ;SPLIT FINALE FOR SYS INTS
        NS      FLAGS           ;TEST SYSTEM INTS BIT
        BZ      PAR31           ;BR IF LEAVE INTS OFF
```

```
;
        LIS     1               ;TEST PARITY BIT
        NS      GP2             ;ZERO FOR EVEN
        BZ      PAR40           ;BR AROUND NEG STAT IF EVEN
        OI      H'80'           ;SETS NEG STATUS
PAR40:  EI                      ;ENABLE SYSTEM INTS
        POP                     ; AND RETURN
;
PAR31:  LIS     1               ;SAME AS ABOVE, TEST PTY
        NS      GP2             ; BIT IN GP2
        BZ      PAR41            BR AROUND STAT SET IF 0
        OI      H'80'           ;SET NEG STATUS
PAR41:  POP                     ;RETURN WITH INTS OFF
```

TRANSMIT MODE OPERATION

In transmit mode, the output gate is initially disabled, and remains disabled until the first occurance of EOW, which signifies end-of-word condition. During that time, the data in TXHOLD is loaded into the shift register and the output gate becomes enabled. An interrupt is triggered and the READY flag is set, indicating that the TXHOLD register may be loaded with new data. If no new data is loaded, and the port is in async mode, then when the next EOW occurs, the output gate will become disabled once again. There will still be an interrupt, and the contents of TXHOLD will still be loaded into the shift register. If no further action is taken on the serial port, interrupts will continue to be triggered after every EOW pulse. Reading the ERROR flag will clear it each time, but it will become set again after every subsequent EOW.

If, in response to one of the interrupts, data is loaded into TXHOLD, then just prior to the next interrupt the output gate will be enabled, the contents of TXHOLD will be loaded into the shift register, and the data will be shifted out the output pin SO.

Once the interrupts start occurring, it is hazardous to attempt loading TXHOLD at times other than immediately after the interrupt, since otherwise the interrupt may occur unexpectedly, resulting in inadvertent transmission. For example, if the interrupt were to occur in between the loading of the first and second halves of TXHOLD, the first transmission would consist of part of the old data and part of the new data. The same hazard is there if the response to the interrupt is slow, on the order of one character time. It is, therefore, advisable to operate the serial port at a slow enough Baud rate to insure that worst case interrupt handling guarantees service within a character time. This consideration becomes more critical in sync mode transmission. Loading the control port has the effect of restarting the timing process, and can be used to initialize the counter.

The programming example of Figure 12 illustrates putting the serial port into automatic transmit mode. Subsequently, data put into the TXFIFO buffer will be assembled and sent out the serial port under the interrupt level code, shown in Figure 13.

ENTERING ASYNC TRANSMIT MODE

Figure 12

```
;
;     ROUTINE TO PUT PORT IN ASYNC TRANSMIT MODE
;SETS UP FOR INTERRUPT DRIVEN TRANSMIT FROM TXFIFO
;
;SEE INITIALIZATION FOR EQUATES
;
ASYXMT: DI                      ;PROTECT RETURN IN P1
        LI      MSKWLN          ;WORD LENGTH MASK
        NS      CONFIG          ;ASSEMBLE PORT D CODE
        OI      PDXMIT+PDINTS   ;
        OUTS    PORTD           ;PUT PORT IN XMIT MODE
        INS     PORTD           ;CLEAR ERROR FLAG
        CLR                     ;
        OUTS    PORTE           ;CLEAR READY FLAG
        OUTS    PORTF           ;CLEAR DATA PORTS
;
        LI      MSKWLN+MSKBRT   ;CLEAR SYNC BIT IN CONFIG
        NS      CONFIG          ;
        LR      CONFIG,A        ;
```

```
;
                LR      A,FLAGS             ; CLEAR ERRORS AND RX BIT
                NI      MSKPTY+MSKSYI+MSKNSY        ;     IN FLAG REG
                OI      MSKTXM              ;SET TRANSMIT MODE
                LR      FLAGS,A
;
                LI      MSKSYI              ;LEAVE SYS INT LIKE IT WAS
                NS      FLAGS               ;
                BZ      ASYX1               ;BR IF NOT SET
                EI                          ;ELSE SET INTS
ASYX1:          POP                         ;RETURN TO CALLER
;
```

INTERRUPTS FOR ASYNC TRANSMIT MODE

Figure 13

```
;
;     INTERRUPT ROUTINE FOR ASYNC TRANSMIT MODE
;
ISCRU:      EQU     7               ;INTERRUPT SCRATCH AREA
ISCRL:      EQU     0               ;     ISAR VALUES
JR:         EQU     9               ;REGISTER 9
;
            ORG     H'E0'
;
XMTINT:     LR      JR,A            ;TEMP STASH ACC IN J
            LR      A,IS            ;SAVE ISAR
            LISU    ISCRU           ;POINT TO SCRATCH AREA
            LISL    ISCRL           ;
            LR      I,A             ;SAVE ISAR IN 1ST LOC
            LR      A,JR            ;PICK UP THE ACC
            LR      I,A             ;ACC IN LOC #2
            LR      J,W             ;SAVE PROC STATUS
            LR      A,KU            ;FREE UP THE K REG TO HOLD P1
            LR      I,A             ;     FOR SUBROUTINE CALLS
            LR      A,KL            ;     USING P1
            LR      I,A             ;
            LR      GPO,A           ;SAVE THIS FOR DATA
            LR      S,A             ;
            LR      K,P             ;SAVE P1 IN K
;
            LI      .NOT.MSKSYI     ;RESET SYSTEM INT FLAG
            NS      FLAGS           ;
            LR      FLAGS,A         ;
            LI      MSKSYN          ;SYNC OR ASYNC?
            NS      CONFIG          ;ZERO FOR ASYNC
            BZ      XMTI1           ;BRANCH AROUND JUMP TO SYNC ON 0
            JMP     INTSYX          ;HANDLE SYNC INTERRUPT (FIG. 3.2-C)
XMTI1:      PI      RDTXF           ;READ TX FIFO
;
            BP      ASYXOT          ;GO HANDLE DATA IT NOT MT
;
            LR      A,FLAGS         ;SET UNDERRUN CONDITION
            OI      MSKORE          ;
            LR      FLAGS,A         ;
XMTI2:      LR      A,FLAGS         ;PUT SYSTEM INTS BACK ON
            OI      MSKSYI          ;
            LR      FLAGS,A         ;
            LISU    ISCRU           ;POINT TO SCR AREA
            LISL    ISCRL+4         ;
            LR      A,D             ;RESTORE GP0
```

```
            LR      GP0,A           ;
            LR      P,K             ;RESTORE INT RETURN ADDRESS
            LR      A,D             ;GET K CONTENTS
            LR      KL,A            ;
            LR      A,D             ;
            LR      KU,A            ;
            LR      W,J             ;RESTORE PROC STATUS
            LR      A,0             ;GET ACC VALUE
            LR      JR,A            ;TEMP STORAGE IN J
            LR      A,S             ;ISAR LAST
            LR      IS,A            ; RESTORE IT
            LR      A,JR            ; RESTORE THE ACC
            EI                      ;RESTORE INTS
            POP                     ; AND RETURN

;CONTINUATION OF ASYNC TRANSMIT INTERRUPT
;
;ASSUMES 7-BIT DATA AND ODD PARITY, IF ENABLED
;
ASYXOT:     LI      MSKPTY          ;IS PARITY ENABLED
            NS      FLAGS           ;ZERO IF NOT ENABLED
            LIS     7               ;FOR PORT E
            BZ      ASYX01          ;BRANCH AROUND CALL TO PARITY
            PI      PARITY          ;RETURNS NEG STATUS IF ODD
            LIS     6               ;SET UP FOR PORT E VALUE
            BM      ASYX01          ;BR ARND INCR IF ALREADY ODD
            INC                     ;PUTS E VALUE TO 7
;
ASYX01      OUTS    PORTE           ;LOAD PORT E
;
            LR      A,GP0           ;DATA IN GP0
            SL      1               ;PUT IN START BIT
            OUTS    PORTF           ;FINAL OUTPUT
            BR      XMTI2           ;RETURN TO EXIT INTS
;
RDTXF:      EQU     $               ;FIFO ROUTINES NOT DESCRIBED HERE
;
```

RECEIVE MODE

In async receive mode, the Start/Stop Synchronizer is used to detect the presence of the incoming character. On spurious input, or noise on the receive line, filtering takes place automatically, with no indications visible to the receiver routines. If a noise hit occurs that is less than one half bit time in duration, the serial port hardware will filter it out with its start bit validation logic.

The async receive mode is activated by asserting the EDGE bit in the control register, port D, with the SEARCH and XMIT bits off. Once a valid start bit is detected, SI is sampled once per bit time and shifted into the shift register until the programmed number of bits corresponding to WLEN in the control port have been clocked in. Then the assembled character is loaded into the receive holding register, RXHOLD, the interrupt is triggered, and the READY flag is set. The Start/Stop Synchronizer becomes ready to accept a new start bit immediately. If no new characters appear, the Word Length Counter is held off by the INIT signal generated in the Start/Stop Synchronizer. Similarly, the 16X Baud Rate divider is held reset by INIT to ensure that the correct phase relationship between the shift pulses and the data bits is achieved. The arriving start bit releases the counters in the serial port in phase with the new data being shifted in.

As seen in the discussion on async transmit mode, there is a hazard if the received character is not read from RXHOLD before the next incoming character is due to arrive. A good policy is to require that in the worst case, the serial port interrupt can be serviced within a character time. If, as in the example given in the async transmit mode discussion, the next interrupt occurs between reading the first and second half of RXHOLD, the data read will be split, and there will be no ERROR signal due to the fact that the first READ "X" signal resets the READY flag. Additionally, while the reads of the data ports E and F reset the READY flag, they do not affect the interrupt pending flip-flop, IPEND (see Figure 9). This means that when the return is made to in-line code from the receive interrupt handling routing, and the system

interrupts are once again enabled, an interrupt will take place from the serial port with READY not set. This condition can be used to alert the firmware to the fact that an error has taken place. This type of error is referred to as a latent error, symbolized in the programming examples by the mnemonic LATERR.

The async receive code is illustrated in Figure 14 and 15.

ENTERING ASYNC RECEIVE MODE

Figure 14

```
;
;     ROUTINE TO PUT PORT IN ASYNC RECEIVE MODE
;SETS UP FOR INTERRUPT DRIVEN RECEIVE TO RXFIFO
;
;SEE INITIALIZATION FOR EQUATES
;
ASYRCV:   DI                    ;PROTECT RETURN IN P1
          LI      MSKWLN        ;WORD LENGTH MASK
          NS      CONFIG        ;ASSEMBLE PORT D CODE
          OI      PDEDGE+PDINTS ;
          OUTS    PORTD         ;PUT PORT IN RECEIVE MODE
;
          INS     PORTD         ;CLEAR ERROR FLAG
          INS     PORTE         ;CLEAR READY FLAG
;
          LI      MSKWLN+MSKBRT ;RESET SYNC BIT IN CONFIG
          NS      CONFIG        ;
          LR      CONFIG,A      ;
;
          LR      A,FLAGS       ;CLAR ERRORS AND
          NI      MSKPTY+MSKSYI ;     TRANSMIT MODE BITS
          OI      MSKRXM        ;SET RECEIVE MODE
          LR      FLAGS,A       ;
;
          LI      MSKSYI        ;LEAVE SYS INT LIKE IT WAS
          NS      FLAGS         ;
          BZ      ASY1          ;BR IF NOT SET
          EI                    ;ELSE SET INTS
ASY1:     POP                   ;RETURN TO CALLER
;
;
```

INTERRUPTS FOR ASYNC RECEIVE MODE

Figure 15

```
;
;     INTERRUPT ROUTINE FOR ASYNC RECEIVE MODE
;
          ORG     H'60'
;
RCVINT:   LR      JR,A          ;TEMP STASH ACC IN J
          LR      A,IS          ;SAVE ISAR
          LISU    ISCRU         ;POINT TO SCRATCH AREA
          LISL    ISCRL         ;
          LR      I,A           ;SAVE ISAR IN1ST LOC
          LR      A,JR          ;PICK UP THE ACC
          LR      I,A           ;ACC IN LOC #2
          LR      J,W           ;SAVE PROC STATUS
          LR      A,KU          ;FREE UP THE K REG
          LR      I,A           ;     FOR SUBROUTINE CALLS
          LR      A,KL          ;     USING P1
          LR      I,A           ;
          LR      GP0,A         ;SAVE THIS FOR DATA
```

```
                LR      S,A             ;
                LR      K,P             ;SAVE P1 IN K
;
;
                LI      .NOT.MSKSYI     ;RESET SYSTEM INT FLAG TO 0
                NS      FLAGS           ;
                LR      FLAGS,A         ;
;
                INS     PORTD           ;TEST STATUS
                SL      1               ;OVERRUN?
                BP      ASYR1           ;BZ AROUND ERR SET IF POS
;
                LR      A,FLAGS         ;SET OR/UR
                OI      MSKORE          ;
                LR      FLAGS,A         ;
;
ASYR1:          INS     PORTD           ;TEST FOR LATENT ERROR
                BM      ASYR2           ;BR AROUND ERR SET IF READY ON
                LR      A,FLAGS         ;SET LSTERR
                OI      MSKLER          ;
                LR      FLAGS,A         ;
;
ASYR2:          LI      MSKSYN          ;SYNC OR ASYNC?
                NS      CONFIG          ;ZERO FOR ASYNC
                BZ      ASYRIN          ;BR TO INPUT DATA FOR ASYNC ON 0
                JMP     SYNRIN          ;HANDLE SYNC INTERRUPT (FIG. 3.2-E)
;
;
;ROUTINE TO INPUT AND ASSEMBLE ASYNC DATA FROM SERIAL PORT
;
;ASSUMES 7 DATA BITS AND ODD PARITY IF ENABLED
;
;
ASYRIN:         LI      MSKWLN          ;SEE HOW BIG WLEN IS
                NS      CONFIG          ;
                SR      4               ;CONVENIENT FOR ALU
                CI      8               ;NEG=11,EQ=10,POS=9BITS
                BZ      ASYRI1          ;BR TO HANDLE 10 BITS IF 0
                BM      ASYRI2          ;BR TO HANDLE 11 BITS IF NEG
;                       OR STAY AND HANDLE 9 BITS
                INS     PORTE           ;ALL OF THE DATA IS IN E
ASYRI3:         LR      GP0,A           ;STORE DATA IN GP0
;
                LI      MSKPTY          ;SEE IF PARITY ENABLED
                NS      FLAGS           ;
                BZ      ASYRI4          ;BR AROUND PARITY CHECK IF 0
                PI      PARITY          ;CALL PARITY CHECK SUB
                BM      ASYRI4          ;BR AROUND PTY ERR SET IF ODD
;
                LR      A,FLAGS         ;PICK UP FLAG REG
                OI      MSKPTE          ;SET PARITY ERROR
                LR      FLAGS,A         ;
ASYRI4:         LR      A,GP0           ;STRIP PARITY BIT FROM DATA
                NI      H'7F'           ;SEVEN BIT DATA ONLY
                LR      GP0,A           ;
;
                PI      WRRXF           ;WRITE RECEIVER FIFO SUB
                BP      ASYRI5          ;BR AR'ND ERR SET IF FIFO NOT FULL
;
                LR      A,FLAGS         ;OR/UR ERROR GETS SET
```

```
                OI      HSKORE          ;
                LR      FLAGS,A         ;
ASYRI5:         JMP     XMTI2           ;RETURN FROM INTS THRU XMIT CODE
;                                       (IN FIGURE 3.1-C)
ASYRI1:         INS     PORTE           ;MOST OF 10 BITS IN E
                SL      1               ;ALIGN DATA
                LR      GP0,A           ;TEMP STORE IN GP0
                INS     PORTF           ;PICK UP BIT 0
                SR      4               ;SHIFT IT FROM BIT 7 POS'N
                SL      1               ;
                SR      4               ;SHIFT RIGHT 7
ASYRI6:         XS      GP0             ;MERGE IT WITH GP0 DATA
                LR      GP0,A           ;
                BR      ASYRI3          ;BACK IN LINE WITH DATA
;
ASYRI2:         INS     PORTE           ;GET BULK OF DATA
                SL      1               ;
                SL      1               ;ALIGN IT
                LR      GP0,A           ;TEMP STORE
                INS     PORTF           ;BITS 0 & 1 IN F
                SR      4               ;MOVE IT OVER
                SR      1               ;
                SR      1               ;SHIFT RIGHT 6
                BR      ASYRI6          ;COMBINE CODE
;
WRRXF:          EQU     $               ;FIFO ROUTINES NOT DESCRIBED
;
```

LINE CONTROL

Async mode in the MK3873 serial port necessarily implies half-duplex transmission. This is primarily due to the fact that the same Word Length Counter is used for both receive and transmit. The MK3873 serial port will support half-duplex and simplex transmission, or half-duplex transmission on a full-duplex medium. (The serial port will, however, support full-duplex in the sync mode. This topic will be discussed later.)

Half-duplex transmission generally requires a handshake for determining whether to transmit or to receive. The act of changing from receive-to-transmit, or from transmit-to-receive, is referred to as "turning the line around".

Most communication systems use some form of Request-To-Send, (RTS), Clear-To-Send, (CTS), handshake method whereby a station raises RTS and then samples CTS before proceeding into transmit mode. It is up to the receiving station to give the sender the OK to transmit. The RTS output from one station is inverted and connected to the CTS input of the other; thus, each station would control one output, RTS, and sense one input, CTS.

The rules for using the RTS/CTS handshake in the half-duplex mode would go as follows:

ON POWER-UP

- clear RTS; (raises other station's CTS)
- set station into idle mode.

TO TRANSMIT

- wait for CTS to go high;
- raise RTS;
- if CTS went low, drop RTS and go to receive;
- go into transmit mode;
- send data;
- drop RTS;
- return to idle mode.

TO RECEIVE

- wait for CTS to go low;
- go into receive mode;
- receive data;
- wait for CTS to go high;
- return to idle mode.

ON DETECTING CTS GOING LOW

- if idle, so to receive mode;
- if attempting to transmit, abandon attempt and drop RTS;
- if transmitting, ignore until end of transmission.

The rules listed correspond to those followed by a secondary station. A primary station would not give up during the transmit attempt, as the secondary station must. This difference is necessary to resolve the contention that exists when two statons simultaneously attempt to transmit.

SYNC OPERATION

Sync mode capability is a unique feature of the MK3873, not currently found in other single-chip microcomputers with serial ports. In a sync mode system, the clock has to be present with the data. This is due to the tight phase relationship between clock and data in sync mode. In async mode, the phase is adjusted at every start bit. Sync mode transmission has no start bit, and therefore requires some other means of keeping the phase from drifting.

In sync systems using modems, the modem supplies the clocks, and the Baud rate port would be programmed for external clock source. Systems that are directly connected to each other require one station, usually the primary station, to supply the clock from its internal source, while the secondary station is programmed for external Baud clock source.

The programming example of Figure 16 suggests a way of accomplishing character sync in the serial port using a tight programmed loop to follow the data at the bit shift level. This is recommended over interrupt driven tracking for this mode, because of the overhead paid in interrupt handling. In SEARCH mode it is very important to respond quickly to the new data as it is shifted into the shift register. Interrupt driven SEARCH mode routines could be employed for systems that require them. For example, the regular sync mode receiver code could be used by first placing the receiver into SEARCH mode with interrupts enabled. Then the returned data could be examined for the sync pattern. This would be satisfactory only for low Baud rates.

SEARCH MODE

Figure 16

```
;
;ROUTINE TO ACQUIRE CHARACTER SYNC
;
;ROUTINE USES K REG FOR RETURN ADDRESS POINTER
;INTERRUPTS ARE ENABLED THROUGHOUT THIS SUBROUTINE
;
;
;SEE INITIALIZATION FOR EQUATES
;
NUSYNC:   LR     K,P              ;SAVE RETURN IN K
          LI     MSKPTY+MSKSYI    ;INIT FLAG REG
          NS     FLAGS            ;
          LR     FLAGS,A          ;
;
          LR     A,CONFIG         ;SET SYNC MODE
          OI     MSKSYN           ;     IN CONFIG
          LR     CONFIG,A         ;
;
          LI     POSRCH+PDSYNC    ;SET RX, SYNC & SRCH MODES IN PORT D
          OUTS   PORTD            ;
          OUTS   PORTE            ;RESET THE READY FLAG
;
NUSYN1:   INS    PORT0            ;LOOK FOR READY
          BP     NUSYN1           ;LOOP
;
          SL     1                ;TEST FOR OVERRUN
          OUTS   PORTE            ;CLEAR READY FLAG
          BM     NUSYN1           ;NO GOOD IF OVERRUN
;
          INS    PORTE            ;READ THE UPPER DATA
          CI     SYNC2            ;MATCH SYNC CHAR
          BNZ    NUSYN1           ;LOOP IF NO MATCH
;
          INS    PORTF            ;NEED TWO TO CALL IT A MATCH
          CI     SYNC1            ;2ND CHAR SHOULD ALSO MCH FOR SYNC
          BNZ    NUSYN1           ;KEEP TRYING IF NO EQ
;
          LR     A,FLAGS          ;SET IN SYNC IF MATCH
          OI     MSKNSY           :
          LR     FLAGS,A          ;
;
```

```
        PK                      ;RETURN
;
;
;
```

TRANSMIT MODE

Sync transmission is very similar to async transmission from a control point of view. Once programmed for transmit mode, the data is shifted out as in the case of async. The main distinguishing feature of sync transmission is that the output gate that enables data to flow out the output pin, SO, stays enabled after the first end-of-word condition following the output to port D that places the port in the transmit mode. With the output gate enabled, if for some reason new data cannot be loaded into TXHOLD, the serial port will retransmit the old data. In async mode, the output to port D could be used to restart the counters in the serial port; in sync mode it is not desirable to output to port D except when leaving transmit mode, since the Word Length Counter could be altered, causing the station being transmitted to lose character sync.

Programming examples are given for sync transmit: both for initially entering transmit mode, in Figure 17, and in handling the data interrupts, in Figure 18. Entering sync transmit mode is very similar to entering transmit mode in async; the only difference is that the SYNC bits in port D and the configuration register are turned on. Handling of the interrupts is noticably different, since when the FIFO becomes empty, the interrupt routine must substitute sync characters so that the previous data will not be retransmitted. Also, for simplicity, word lengths in the examples for sync mode will be assumed to be fixed at eight bits. If parity is enabled, it will be handled as seven bits of data and one bit of odd parity.

ENTERING SYNC TRANSMIT MODE

Figure 17

```
;
;     ROUTINE TO PUT PORT IN SYNC TRANSMIT MODE
;SETS UP FOR INTERRUPT DRIVEN TRANSMIT FROM TXFIFO
;
;SEE INITIALIZATION FOR EQUATES
;
SYNXMT:  DI                         ;PROTECT RETURN IN P1
         LI      MSKWLN             ;WORD LENGTH MASK
         NS      CONFIG             ;ASSEMBLE PORT D CODE
         OI      PDXMIT+PDINTS      ;
         OUTS    PORTD              ;PUT PORT IN XMIT MODE
         INS     PORTD              ;CLEAR ERROR FLAG
         LI      SYNC2              ;SYNC CHARS IN DATA PORTS
         OUTS    PORTE              ;     AND CLEAR READY FLAG
         LI      SYNC1              ;
         OUTS    PORTF              ;
;
         LR      A,CONFIG           ;SET SYNC BIT IN CONFIG
         OI      MSKSYN             ;
         LR      CONFIG,A           ;
;
         LI      MSKPTY+MSKSYI+MSKNSY       ;CLEAR ERRORS
         NS      FLAGS              ;  AND RECEIVE BIT IN FLAG REG
         OI      MSKTXM             ;SET TRANSMIT MODE
         LR      FLAGGS,A
;
         LI      MSKSYI             ;LEAVE SYS INT LIKE IT WAS
         NS      FLAGS              ;
         BZ      SYNX1              ;BR IF NOT SET
         EI                         ;ELSE SET INTS
SYNX1:   POP                        ;RETURN TO CALLER
;
```

As was observed in the async transmit interrupt code in Figure 18, a test was made for sync mode. When the test was negative, async transmit interrupt handling took place starting at the code labelled, ASYXOT. There the async data was read in from the ports E and F, and the transmitter FIFO was written. When ASYXOT was completed, a jump was made back to code labelled, XMTI2, for a restoration of context and a return to the in-line code.

Figure 18 contains code labelled INTSYX, which conducts the sync transmit mode data handling. The return from this code is back through XMTI2, the same as in the async case.

INTERRUPT ROUTINE FOR SYNC TRANSMIT MODE
Figure 18

```
;
;     INTERRUPT ROUTINE FOR SYNC TRANSMIT MODE (FROM FIGURE 3.1-C)
;
;
INTSYX:   LI     MSKRXM      ;TEST FOR FULL DUPLEX
          NS     FLAGS       ;RX SET IF FDX MODE
          BNZ    INTFDX      ;BR TO FDX ROUTN IF SET (FIG. 3.2-G)
;
INTSY1:   PI     RDTXF       ;READ THE TRANSMITTER FIFO
          BP     INTSY2      ;BR AROUND SYNC INSERTION IF NOT MT
;
          LI     SYNC1       ;INSERT ONE SYNC CHAR
          LR     GP0,A       ;     AND CONTINUE
INTSY2:   LI     MSKPTY      ;PARITY ENABLED?
          NS     FLAGS       ;
          BZ     INTSY3      ;BR AROUND IF NOT ENABLED
;
          PI     PARITY      ;GENERATE PARITY
          BP     INTSY3      ;BR AROUND IF ALREADY ODD
;
          LI     H'80'       ;FLIP BIT 7 IF EVEN
          XS     GP0         ;
          LR     GP0,A       ;
;
INTSY3:   LR     A,GP0       ;GET DATA
          OUTS   PORTF       ;PLACE IT IN PORT F
          JMP    XMTI2       ;RETURN (FIG. 3.1-C)
;
```

RECEIVE MODE

Receiving in sync mode is considerably different from receiving in async. In async receive mode the serial port hardware establishes bit and character synchronization automatically. In sync mode, however, bit syncronization is inherant in the common clock usage, and character sync must be obtained by the firmware observing the incoming date, bit-by-bit, as it is shifted in, until a certain sync pattern is recognized. Once character sync is achieved, sync mode receive proceeds very much like async. Interrupts occur regularly, every time the Word Length Counter reaches the end-of-word condition.

In the serial port, the search mode is entered by setting the SEARCH bit in port D, with SYNC also being set and XMIT reset. The READY and ERROR flags function as they would if the word length could be programmed for one bit. EOW is generated on every shift, resulting in a trigger for the interrupt, for the READY flag, and for the ERROR flag. Since the shift register is simultaneously shifting and loading RXHOLD on every shift pulse, the data read from ports E and F appears to be shifting as it does in the shift register. When the sync character is recognized, the port must be set to the normal sync receive mode by outputting the correct pattern to port D to reset the SEARCH bit.

In the programming example of Figure 19, the serial port is placed into the sync receive mode. Prior to calling this routine, it is assumed that NUSYNC was called and synchronization has been achieved. Figure 20 shows the interrupt level sync receive routine.

ENTERING SYNC RECEIVE MODE

Figure 19

```
;
;     ROUTINE TO PUT PORT IN SYNC RECEIVE MODE (FROM FIG. 3.1-E)
;SETS UP FOR INTERRUPT DRIVEN RECEIVE TO RXFIFO
;
;SEE INITIALIZATION FOR EQUATES
;
SYNRCV:   DI                        ;PROTECT RETURN  IN P1
          LI      MSKWLN            ;WORD LENGTH MASK
          NS      CONFIG            ;ASSEMBLE PORT D CODE
          OI      PDSYNC+PDINTS     ;
          OUTS    PORTD             ;PUT PORT IN RECEIVE MODE
;
          INS     PORTD             ;CLEAR ERROR FLAG
          INS     PORTE             ;CLEAR READY FLAG
;
          LR      A,CONFIG          ;
          OI      MSKSYN            ;SET SYNC BIT IN CONFIG
          LR      CONFIG,A          ;
;
          LR      A,FLAGS           ;CLEAR ERRORS AND
          NI      MSKPTY+MSKSYI+MSKNSY      ;     TX MODE BITS
          OI      MSKRXM            ;SET RECEIVE MODE
          LR      FLAGS,A           ;
;
          LI      MSKSYI            ;LEAVE SYS INT LIKE IT WAS
          NS      FLAGS             ;
          BZ      SYNR1             ;BR IF NOT SET
          EI                        ;ELSE SET INTS
SYNR1:    POP                       ;RETURN TO CALLER
;
;
```

INTERRUPT ROUTINE FOR SYNC RECEIVE MODE

Figure 20

```
;
;     INTERRUPT LEVEL ROUTINE FOR SYNC RECEIVE MODE
;ASSUMES 8-BIT DATA & NO PTY, OR 7-BIT + ODD PTY IF ENABLED
;
SYNRIN:   INS     PORTE             ;GET THE RX DATA
          LR      GP0,A             ;SAVE IT IN GP0
          LI      MSKPTY            ;PARITY ENABLED?
          NS      FLAGS             ;
          BZ      SYNRI1            ;BR IF NO PARITY
;
          PI      PARITY            ;PASSES IF ODD
          BM      SYNRI2            ;   MINUS MEANS ODD
;
          LR      A,FLAGS           ;SET ERROR IF EVEN
          OI      MSKPTE            ;
          LR      FLAGS,A           ;
SYNRI2:   LI      H'7F'             ;TRIM DATA TO 7 BITS
          NS      GP0               ;
          LR      GP0,A             ;
;
SYNRI1:   EQU     $                 ;CONTINUED NEXT PAGE
;
;
```

```
;SYNRI1:   EQU   $              FROM PREVIOUS PAGE
;
          LR    A,GP0          ;STRIP IF SYNC1 CHAR
          CI    SYNC1          ;
          BNZ   SYNRI3         ;BR AROUND RETURN IF NO MATCH
;
          LR    A,FLAGS        ;
          OI    MSKNSY         ;SET IN-SYNC FLAG
          LR    FLAGS,A        ;
;
SYNRI4:   JMP   XMTI2          ;RETURN FROM INTERRUPT (FIG. 3.1-C)
;
SYNRI3:   PI    WRRXF          ;WRITE DATA TO RXFIFO
          BP    SYNRI4         ;RETURN IF FIFO NOT FULL
;
          LR    A,FLAGS        ;SET OVERFLOW IF FIFO FULL
          OI    MSKORE         ;
          LR    FLAGS,A        ;
          BR    SYNRI4         ;     AND RETURN
;
;
```

FULL DUPLEX TRANSMISSION

The unique design of the MK3873 serial port makes it possible to simultaneously transmit and receive. In this full-duplex operation, character synchronization is maintained between transmit and receive data, allowing the sharing of the Word Length counter between the two data paths. Interrupt circuitry, READY, and ERROR, are also shared between receive and transmit.

As in previously described sync applications, it is necessary to define the stations as either primary or secondary. To begin, the primary station puts its serial port in sync transmit mode, and commences transmitting the sync pattern to the secondary station. Upon receiving interrupts, the primary must read the incoming data to check it against the sync pattern; if it matches, it means that the secondary has acquired character sync and started transmitting. The secondary station acquires character sync. Once the secondary has detected the sync pattern from the primary, the secondary puts its serial port into sync transmit mode and begins transmitting the sync pattern to the primary.

Once both stations are receiving sync characters, full-duplex communication can proceed.

Programming examples in Figure 21 and 22 illustrate use of the serial port in the sync full-duplex mode. Entering the mode first requires calling NUSYNC, then calling FULDUX to start operating. FULDUX is shown in Figure 21.

The interrupt level code is based on the sync transmit code and takes the branch within it that tests for RX, or the receive bit in FLAGS, being true. An interrupt to vector E0 (hex) with RX true implies full-duplex mode.

ENTERING SYNC FULL-DUPLEX MODE

Figure 21

```
;
;
FULDUX:   DI                   ;PROTECT RETURN  IN P1
          LI    MSKWLN         ;WORD LENGTH MASK
          NS    CONFIG         ;ASSEMBLE PORT D CODE
          OI    PDXMIT+PDINTS  ;
          OUTS  PORTD          ;PUT PORT IN XMIT MODE
          INS   PORTD          ;CLEAR ERROR FLAG
          LI    SYNC2          ;SYNC CHARS IN DATA PORTS
          OUTS  PORTE          ;     AND CLEAR READY FLAG
          LI    SYNC1          ;
          OUTS  PORTF          ;
;
          LR    A,CONFIG       ;SET SYNC BIT IN CONFIG
          OI    MSKSYN         ;
```

```
                LR      CONFIG,A            ;
;
                LI      MSKPTY+MSKSYI+MSKNSY    ;CLEAR ERRORS
                NS      FLAGS               ;     IN FLAG REG
                OI      MSKTXM+MSKRXM ;SET TRANSMIT AND RECEIVE MODE
                LR      FLAGS,A
;
                LI      MSKSYI              ;LEAVE SYS INT LIKE IT WAS
                NS      FLAGS               ;
                BZ      FULD1               ;BR IF NOT SET
                EI                          ;ELSE SET INTS
FULD1:          POP                         ;RETURN TO CALLER
;
```

INTERRUPT ROUTINE FOR SYNC FULL-DUPLEX MODE

Figure 22

```
;
;     INTERRUPT LEVEL ROUTINE FOR SYNC FULL-DUPLEX MODE (FROM FIG. 3.2-G)
;ASSUMES 8-BIT DATA & NO PTY, OR 7-BIT + ODD PTY IF ENABLED
;
INTFDX:     INS     PORTD           ;TEST STATUS
            SL      1               ;OVERRUN?
            BP      INTFD1          ;BZ AROUND ERR SET IF POS
;
            LR      A,FLAGS         ;SET OR/UR
            OI      MSKORE          ;
            LR      FLAGS,A         ;
;
INTFD1:     INS     PORTD           ;TEST FOR LATENT ERROR
            BM      FDXRIN          ;BR AROUND ERR SET IF READY ON
            LR      A,FLAGS         ;SET LSTERR
            OI      MSKLER          ;
            LR      FLAGS,A         ;
;
FDXRIN:     INS     PORTE           ;GET THE RX DATA
            LR      GP0,A           ;SAVE IT IN GP0
            LI      MSKPTY          ;PARITY ENABLED?
            NS      FLAGS           ;
            BZ      FDXRI1          ;BR IF NO PARITY
;
            PI      PARITY          ;PASSES IF ODD
            BM      FDXRI2          ; MINUS MEANS ODD
;
            LR      A,FLAGS         ;SET ERROR IF EVEN
            OI      MSKPTE          ;
            LR      FLAGS,A         ;
FDXRI2:     LI      H'7F'           ;TRIM DATA TO 7 BITS
            NS      GP0             ;
            LR      GP0,A           ;
;
FDXRI1:     LR      A,GP0           ;STRIP IF SYNC1 CHAR
            CI      SYNC1           ;
            BNZ     FDXRI3          ;BR AROUND RETURN IF NO MATCH
;
            LR      A,FLAGS         ;
            OI      MSKNSY          ;SET IN-SYNC FLAG
            LR      FLAGS,A         ;
;
FDXRI4:     JMP     INTSY1          ;RETURN TO TRANSMIT (FIG. 3.2-C)
;
```

```
FDXRI3:   PI    WRRXF       ;WRITE DATA TO RXFIFO
          BP    FDXRI4      ;RETURN IF FIFO NOT FULL
;
          LR    A,FLAGS     ;SET OVERFLOW IF FIFO FULL
          OI    MSKORE      :
          LR    FLAGS,A     ;
          BR    FDXRI4      ;    AND RETURN
;
;
```

SUMMARY AND CONCLUSIONS

This app note has attempted to cover every aspect of control of the serial port on the MK3873 microcomputer. It was shown that the serial port supported operation in several modes of serial transmission, including:

- Async receive;
- Async transmit;
- Sync search mode;
- Sync receive;
- Sync transmit
- Sync full-duplex.

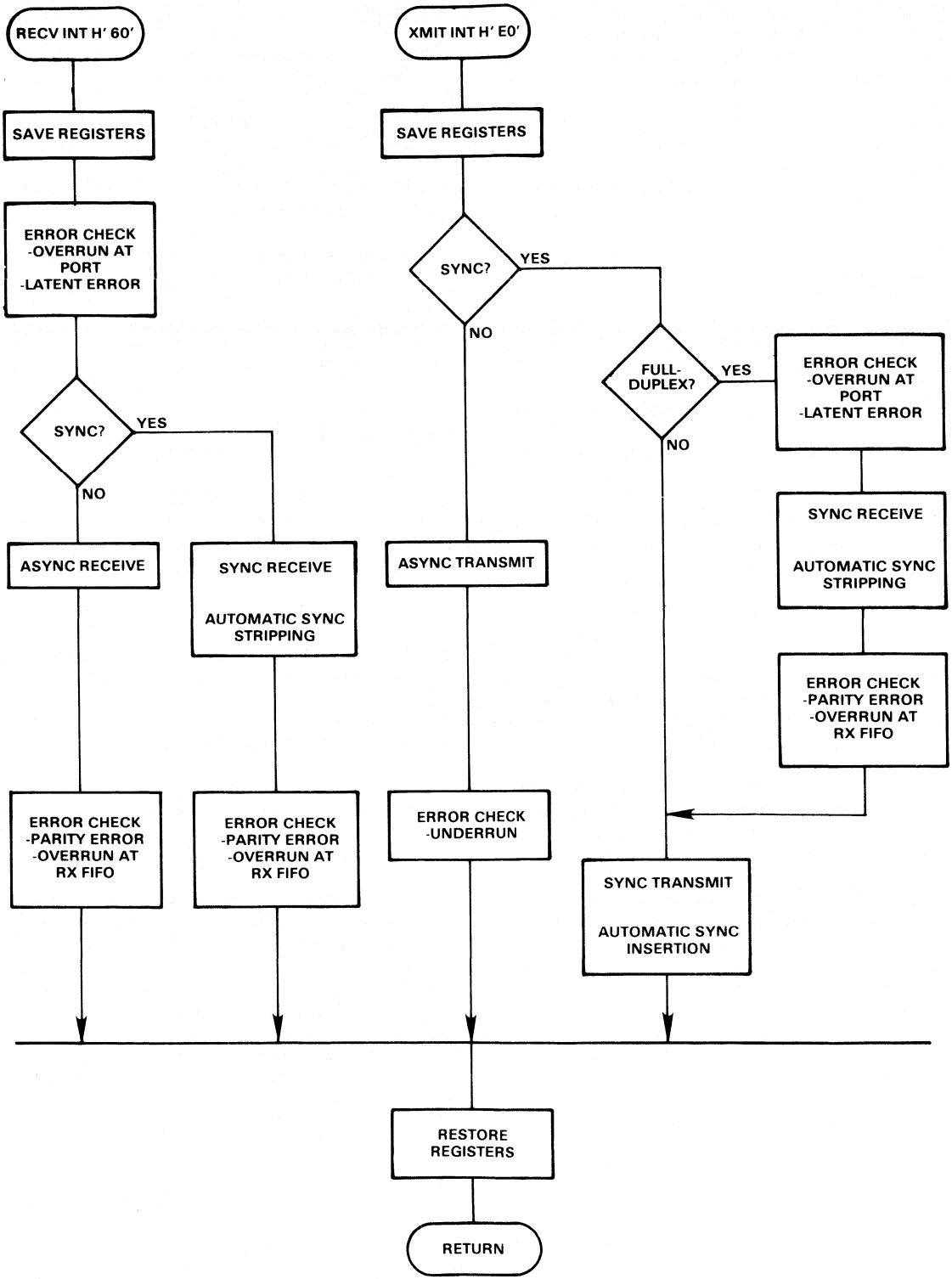
Means were described for using internal and external Baud clocks to set the pace for transmission.

Differences were noted between sync and async transmission for the purposes of control. Differences are also present from the system viewpoint. Sync mode, carrying its own clock with the data, eliminates the need for start and stop bits which add to the overhead of the channel, lowering the effective information carrying rate for async to less than 75%. In sync mode some overhead is present in the form of sync characters, but they only lower the effective Baud rate to about 97%, since only one pair of sync characters is generally required for every 80 data characters for most modems.

The software was shown to be made up of mode setting routines, and interrupt level routines. The interrupt level structure is summarized in Figure 23.

SERIAL PORT INTERRUPT OPERATION SUMMARY

Figure 23



APPENDIX - GLOSSARY OF TERMS

ASYNC - Short for ASYNCHRONOUS, which relates two events as having no phase correlation. Async communication implies start/stop mode, where the timing of the start of a burst of information is unpredictable.

BAUD - Rate of data flow. Strictly speaking, a Baud is a rate of signal level changes. In the serial portion of the 3873, a signal level change corresponds to one bit. Thus Baud and bit rate are synonymous, and 110 Baud is the same as 110 bits-per-second.

BIT - Short for BInary digit. The smallest indivisible information cell.

BUFFER - Isolates a signal from its origin. May have storage, as in the case of a buffer memory; or not, as in the case of the output buffer gate in the Serial Clock buffer.

BYTE - A close grouping of eight bits.

CRYSTAL - A frequency stabilizing device used in the 3873 to hold the 3.6864 MHz frequency from drifting.

DUPLEX - Refers to double usage of something. In duplex communications, the channel conducts data flow in two different directions.

ERROR - A recoverable loss of data, such as that induced noise.

FIFO - Stands for First-In-First-Out. This type of buffer memory is very useful for taking up differences in timing between two asynchronous processes, such as a remote transmitter and a local processor.

FRAMING - In serial communications, this refers to the format of adjacent bits in a serial stream. Async data frames may be grouped as a start bit, seven data bits, and two stop bits, for example.

FULL DUPLEX - In data communications, this is the simultaneous support of transmit and receive data flow.

HALF DUPLEX - Refers to using communication equipment in both directions of transmission, but only one direction at a time.

HEX - Short for HEXADECIMAL. Base sixteen numbering notation. Digits take on the values of 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, AND F. Weights are from zero to fifteen, respectively.

HIT - Refers to an abrupt and short-lived noise condition on a transmission line that results in an error.

PORT - An avenue of conduction for data, from inside a computer to the outside.

RAM - Short for Random Access Memory. Generally means read/write random access memory.

RECEIVE - The receive direction referred to the computer is data that enters from the outside.

REGISTER - A storage cell, usually containing as many bits as the size of the computer word. Address registers contain a number of bits representing the addressability of the computer.

ROM - Stands for Read Only Memory, but generally refers to nonvolatile read only memory.

SYNC - Short for SYNCHRONOUS, or synchronizing. Sync communication is relatively continuous and data is accompanied by a clock. Sync characters in the data flow are used to attain and maintain character synchronization.

TRANSMISSION - Data flow across a medium.

TRANSMIT - The transmit direction referred to the computer is the direction of data leaving the computer.

TURN-AROUND - The act of switching a transmission media from carrying data in one direction, to carrying data in the other.

UART - Stands for Universal Asynchronous Receiver/-Transmitter. A device for serializing and deserializing data between a computer and a serial data line.

USART - Short for Universal Synchronous/Asynchronous Receiver/Transmitter. Different from the UART in that it supports both sync and async communications.

VECTOR - Specifically, interrupt vector. The code an interrupting device passes to a CPU during an interrupt acknowledge cycle which determines the address of the interrupt service routine for that device.

CRC HANDLING IN THE MK3870 FAMILY OF MICROCOMPUTERS

Application Note

INTRODUCTION

This application note is intended to provide an efficient and effective means of accomplishing the task of error detection using Cyclic Redundancy Codes, (CRC). While the implementations presented use the 3870 Family of single-chip microcomputers, the principles described are generally applicable.

CRC BACKGROUND

Codes are usually described in mathematics as closed sets of values that comprise all the allowed number sequences in the code. In data communications, transmitted numbers are essentially random data patterns which are not related to any predetermined code set. The sequence of data, then, is forced into compliance with the code set by adding to it at the transmitter. Thus, a string of original data would become the original string concatenated with a string of extra numbers that make the total string one of allowed code set values.

At the receiver, the incoming data is checked to see if it is one of the allowed code set values. The assumption is made that if an error occurred in transmission, the likelihood of the result also being a valid set member would be very low. If the received data string is found to be of the allowed code set, it is assumed that no errors have occurred and that the data is valid.

Several points have emerged from the above discussion; 1). There is a need to have a scheme of determining what precise extra string to append to the original data stream, to make the concatenation of transmitted data a valid member of the code set 2). There must be a consistant way of extracting the original data from the code value at the receiver, to deliver the actual data to the location where it is ultimately used and 3). For the code scheme to be effective, the set must contain allowed values sufficiently different from one another that expected errors will not be able to alter one allowed value such that it becomes a different allowed value of the code set.

A system for coding and detecting errors in common use in Data Communications and in systems using serial data storage devices is called CRC, for Cyclic Redundancy Code. The code set is made up of all strings of binary data that are evenly divisible by what is referred to as a "generator polynomial" — a specially selected number that results in a code set of values different enough from one another to achieve a certain low probability of an undetected error. To determine what to append to the string of original data, a division is made of the original string as it is being transmitted. When the last data is past, the remainder from the division is the required string to add since the string including the remainder will be evenly divisible by the generator polynomial. Since the generator polynomial is of a known length, the remainder added to the original string is of a fixed length.

At the receiver, the incoming string is divided by the generator polynomial, and if the incoming string does not divide evenly - that is if the remainder after division is not zero - then an error is assumed to have occurred. If the remainder is zero, then the data is assumed to be error free, and the data delivered to the ultimate destination is the incoming data with the fixed length remainder field removed.

Figure 1 illustrates the stages of this coding method.

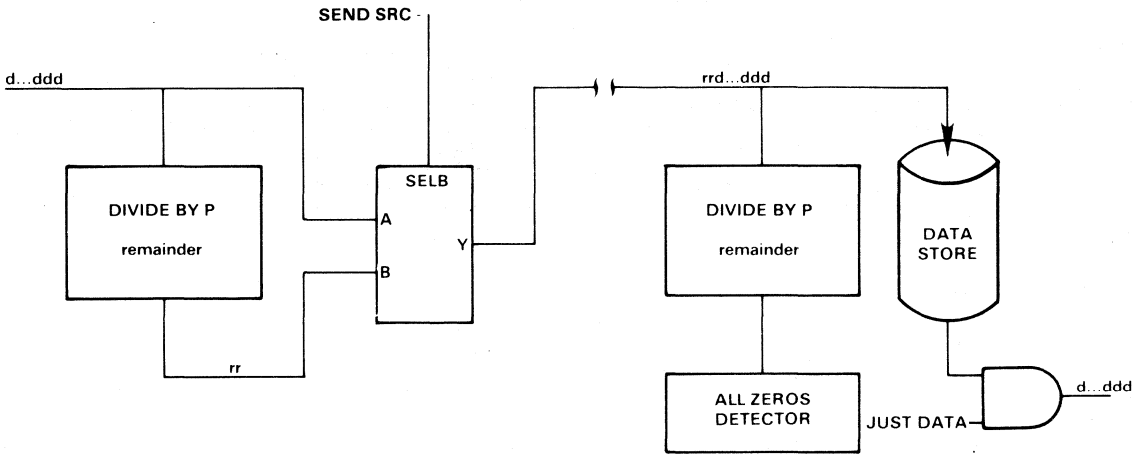
In binary CRC schemes, the generator polynomial is designated as a sum of terms of "X" raised to the power of the bit the term represents. For example, the CRC-16 generator polynomial is actually the binary number, 1 1000 0000 0000 0101, while it is customary to represent it with the following expression:

$$x^{16}+x^{15}+x^{2}+x^{0}$$

SHIFT REGISTER IMPLEMENTATION

The division process is simpler in modulo-two arithmetic than it is in decimal arithmetic. Implementation of a divider for a seventeen bit polynomial can be done using a sixteen bit shift register with an exclusive-OR feedback gate for each term of the polynomial, except the most significant bit, (bit 16). The exclusive-OR gate that corresponds to bit 16 of the polynomial is the one with one input connected to the rightmost shift register bit and the other input connected to the incoming data stream, INPUT. The output of this gate is used to turn the feedback on and off to the rest of the shift register. Figure 2 shows the classical shift register circuit used to perform the division for the case of the CRC-16 polynomial. To generate the remainder for transmission, the shift register is first preset to all zeros. The serial data is shifted in at the point marked INPUT. When the end of the data is reached, the final contents of the shift register are appended to the serial data stream by lowering the feedback enable input labelled ENFB. The stream thus generated is a

ERROR DETECTION SYSTEM
Figure 1



member of the CRC-16 code set, since it is evenly divisible by the polynomial.

The box in Figure 2 labeled FEEDBACK POLYNOMIAL is an AND gate array. The output signals PXX refer to the X terms in the polynomial. CRC-16 would have the outputs, P00, P02 and P15 enabled such that when the input went high, those outputs would go high but the other outputs would not go high. In applications not requiring more than one polynomial, much of the indicated circuitry could be eliminated. This example is intended to be general enough to handle any seventeen term CRC polynomial, simply by enabling the appropriate PXX outputs. The shift register cells are shown in detail in the insert at the bottom of the figure. The inverter inside the detail is used when testing the contents of the shift register for zero at the end of the receive operation.

The selector in the lower right hand portion of Figure 2 is used to select the data on the signal labelled INPUT to be forwarded to the OUTPUT line, while the shift register is accumulating the CRC. The shift register contents are selected at the same time as the feedback enable is removed when it is desired to concatenate the CRC value to the data stream. The rightmost bit in the shift register is the first to be shifted out.

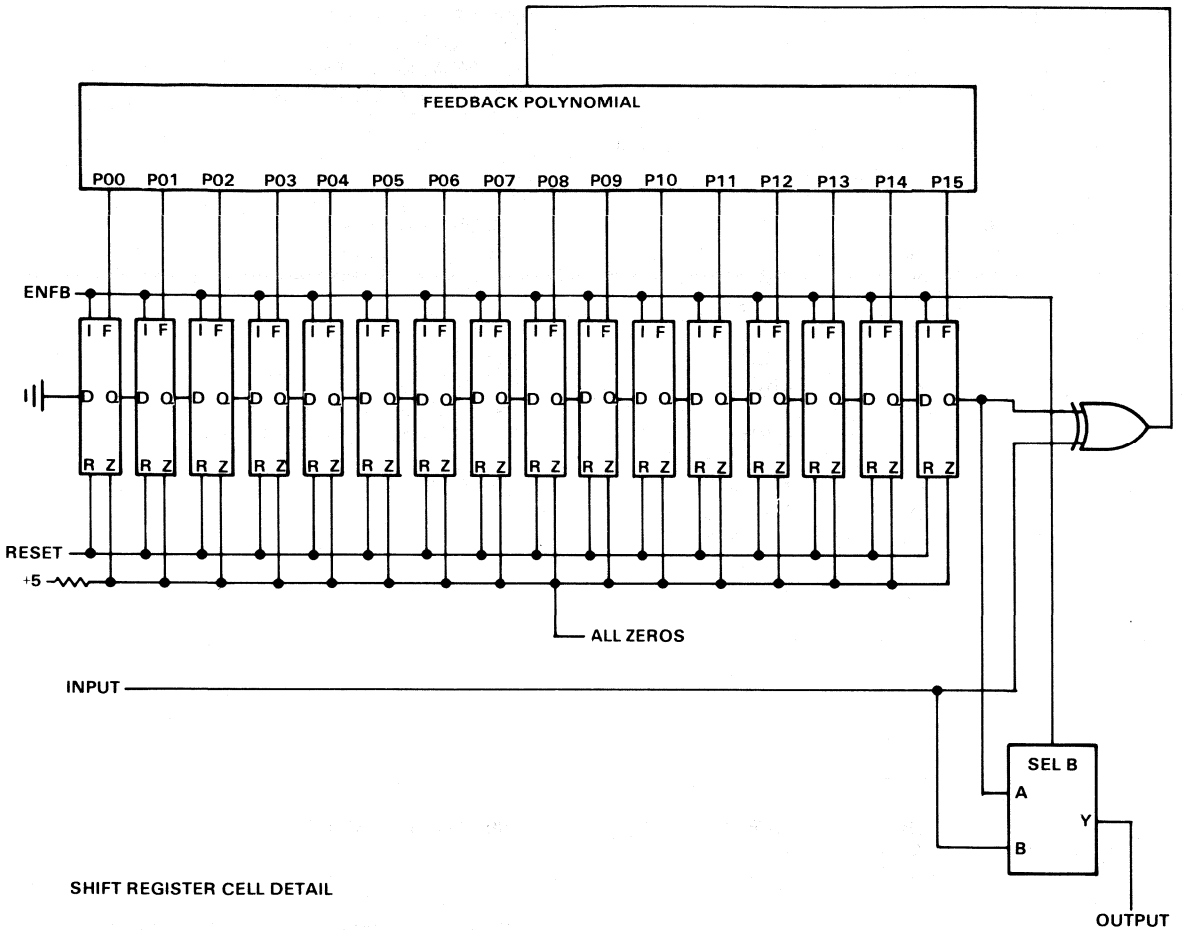
The shift register can also be implemented with a software algorithm. The MK3870 assembly language is used to define the algorithm, which appears in Figure 3. Like the hardware implementation, the software algorithm also accommodates any polynomial.

By altering the value in the equation for POLYU and POLYL, any CRC generator polynomial can be facilitated. It is necessary to call the subroutine for each bit of data. As this routine may take up to 111 microseconds to execute with a 4 MHz clock, the software implementation is often only useful for slower data transmission rates. The calling program is responsible for initializing the values of CRCU and CRCL, shifting each bit of data, and making the call to CRCSHF for each data bit. When the end of data is reached, the calling program places the CRC bytes into the data stream, CRCL first, followed by CRCU.

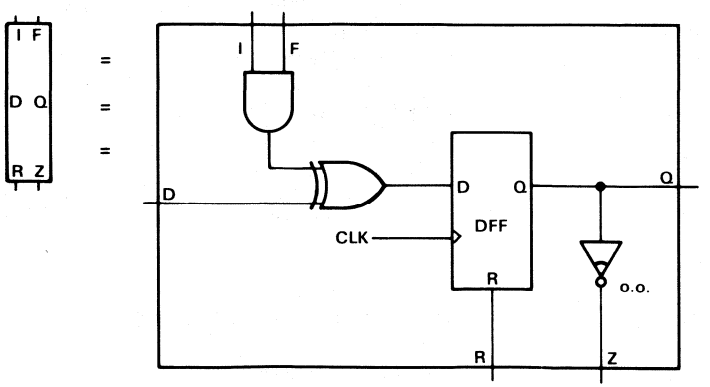
The table of Figure 4 shows the values taken on by the shift register in the single bit shift approach in response to the input stream in the left column. The feedback gating, generated by the exclusive-OR of input data with the rightmost bit of the register, is shown in the next column. Notice its value in the lower half. Each line of the table shows the register contents after the clocking takes place. These values are the same whether the hardware or the software implementation is used.

SHIFT REGISTER IMPLEMENTATION

Figure 2



SHIFT REGISTER CELL DETAIL



CRC SHIFT SOFTWARE IMPLEMENTATION

Figure 3

```
FLAGS:   EQU    4              ;FLAG REGISTER
MSKCRC:  EQU    1              ;MASK FOR CRC FLAG POSITION
CRCU:    EQU    6              ;UPPER CRC BYTE
CRCL:    EQU    5              ;LOWER CRC BYTE
POLYU:   EQU    H'A0'          ;POLYNOMIAL <0-7>
POLYL:   EQU    H'01'          ;POLYNOMIAL <8-15>
GP0:     EQU    0              ;TEMPORARY DATA STORAGE
;
;        CRC ONE-BIT SHIFT ALGORITHM
;        DATA IN GP0
;
CRCSHF:  LR     K,P            ;SAVE RETURN ADDRESS
         LI     .NOT.MSKCRC    ;RESET CRC FLAG
         NS     FLAGS          ;
         LR     FLAGS,A        ;
;
         LR     A,GP0          ;PICK UP DATA
         XS     CRCL           ;XOR WITH LOWER CRC BIT
         NI     1              ;ONLY USE LOWEST BIT
         BZ     CRC1           ;BRANCH AROUND FLAG SET IF 0
;
         LR     A,FLAGS        ;SET CRC FLAG
         OI     MSKCRC         ;
         LR     FLAGS,A        ;
;
CRC1:    EQU    $              ;
;

;

;

:
         LR     A,CRCL         ;SHIFT LOWER CRC BYTE
         SR     1              ;
         LR     CRCL,A         ;
;
         LIS    1              ;PROPAGATE UPPER BYTE LSB TO LOWER
         NS     CRCU           ;  BYTE MSB
         BZ     CRC2           ;BRANCH IF THE BIT IS 0
;
         LR     A,CRCL         ;ELSE SET CRCL MSB
         OI     H'80'          ;
         LR     CRCL,A         ;
;
CRC2:    LR     A,CRCU         ;SHIFT UPPER BYTE
         SR     1              ;
         LR     CRCU,A         ;
;
         LI     MSKCRC         ;TEST CRC FLAG FOR XOR OF POLY
         NS     FLAGS          ;
         BZ     CRC3           ;BRANCH IF FLAG NOT SET
;
         LI     POLYL          ;XOR LOWER PART
```

CRC SHIFT SOFTWARE IMPLEMENTATION (Continued)

```
                XS      CRCU            ;
                LR      CRCU,A          ;
;
CRC3:           PK                      ;RETURN POINT
;
;
```

CRC SHIFT TABLE

Figure 4

INPUT	FEEDBACK	CRC-16 VALUE AFTER SHIFT CLOCK			
0	0	0000	0000	0000	0000
1	1	1010	0000	0000	0001
0	1	1111	0000	0000	0001
1	0	0111	1000	0000	0000
1	1	1001	1100	0000	0001
0	1	1110	1110	0000	0001
0	1	1101	0111	0000	0001
1	0	0110	1011	1000	0000
0	0	0011	0101	1100	0000
1	1	1011	1010	1110	0001
1	0	0101	1101	0111	0000
0	0	0010	1110	1011	1000
0	0	0001	0111	0101	1100
0	0	0000	1011	1010	1110
1	1	1010	0101	1101	0110
1	1	1111	0010	1110	1010
0	0	0111	1001	0111	0101
(INPUT VALUE FROM THIS POINT IS CURRENT SHIFT REGISTER CONTENTS)					
1	0	0011	1100	1011	1010
0	0	0001	1110	0101	1101
1	0	0000	1111	0010	1110
0	0	0000	0111	1001	0111
1	0	0000	0011	1100	1011
1	0	0000	0001	1110	0101
1	0	0000	0000	1111	0010
0	0	0000	0000	0111	1001
1	0	0000	0000	0011	1100
0	0	0000	0000	0001	1110
0	0	0000	0000	0000	1111
1	0	0000	0000	0000	0111
1	0	0000	0000	0000	0011
1	0	0000	0000	0000	0001
1	0	0000	0000	0000	0000
0	0	0000	0000	0000	0000

DISCUSSION OF CLASSICAL IMPLEMENTATIONS

The hardware implementation of the single shift logic is usually fast enough for most applications. There are components made with selectable polynomials integrated into a single chip. The main drawbacks to the hardware implementation are its inherent cost, plus the fact that most single chip hardware is also single sourced. The drawbacks to the software emulation center around the large cost in time needed to execute the shift algorithm. The hardware implementation uses exclusive-OR gates, and other simple components that have been available for many years. The

software implementation actually does a "blind emulation" of the hardware, but the MK3870, for example, has the power to manipulate data in a much more sophisticated way than doing simple exclusive-ORs and shifting a bit at a time. There is actually very little gained by restricting the software CRC accumulation function to one bit at a time. A possible reason for restricting the treatment to one bit at a time might be to maintain very strict emulation of the hardware.

Looking back for a moment at the hardware shift implementation of Figure 2, a few key functions are evident:

- The least significant accumulated amount is exclusive-ORed with the incoming data to produce a gating function that activates the feedback paths.
- The result of the gating function from above activates a feedback pattern that depends strictly on the generator polynomial.
- The activated legs to the shift register are exclusive-ORed with the previous stage of the shift register and accumulated.

The single feature about the single bit implementation that makes it attractive also makes it deceptively simple. The exclusive-OR of two single bits can only result in either a "one" or a "zero". Thus, the polynomial is either gated into the register or it is not. This fact is not true of hexadecimal numbers, for example. Two hexadecimal numbers exclusive-ORed together can form any of sixteen different values. Bytes exclusive-ORed form a set of two hundred and fifty-six different values.

FOUR BIT CRC METHOD

In deriving a multi-bit method for CRC calculating, it should be sufficient to compare the new method with the old one and show that the differences are in implementation only and do not alter the function. This is true since shifting several bits sequentially in the single bit method produces the same CRC accumulation, whether or not there is an awareness of the intermediate values.

THEORETICAL DISCUSSION

This discussion derives the Boolean expression for the CRC shift register contents following four clocks. The notation employed uses bit positions within parentheses. The table of Figure 5 illustrates the four shifts in the case of the CRC-16 polynomial.

To make the table readable, the following substitutions are made:

Let F(0) = C(0). XOR. I(0)
F(1) = C(1). XOR. I(1). XOR. F(0)
F(2) = C(2). XOR. I(2). XOR. F(1)
and F(3) = C(3), XOR, I(3), XOR, F(2)

WHERE:

C(i) = the value of CRC register bit i at the start;
I(i) = the value of input string bit i.

FOUR BIT CRC-16 SHIFT TABLE
Figure 5

AFTER FIRST SHIFT		AFTER SECOND SHIFT		AFTER THIRD SHIFT		AFTER FOURTH SHIFT	
F(0)		F(1)		F(2)		F(3)	
	C(15)	F(0)		F(1)		F(2)	
F(0).XOR.	C(14)	F(1).XOR.	C(15)	F(0).XOR.	F(2)	F(1).XOR.	F(3)
	C(13)	F(0).XOR.	C(14)	F(1).XOR.	C(15)	F(0).XOR.	F(2)
	C(12)		C(13)	F(0).XOR.	C(14)	F(1).XOR.	C(15)
	C(11)		C(12)		C(13)	F(0).XOR.	C(14)
	C(10)		C(11)		C(12)		C(13)
	C(09)		C(10)		C(11)		C(12)
	C(08)		C(09)		C(10)		C(11)
	C(07)		C(08)		C(09)		C(10)
	C(06)		C(07)		C(08)		C(09)
	C(05)		C(06)		C(07)		C(08)
	C(04)		C(05)		C(06)		C(07)
	C(03)		C(04)		C(05)		C(06)
	C(02)		C(03)		C(04)		C(05)
F(0).XOR.	C(01)	F(1).XOR.	C(02)	F(2).XOR.	C(03)	F(3).XOR.	C(04)

Upon examination of the fourth column of values in the table, the following points become apparent:

- the cells that used to contain C(12) through C(15) now contain a value determined by an operation on I(0) through I(3) and C(0) through C(3)
- the lower three groups of four cells contain values made up of the previous contents of cells four shifts upstream from them and of the function of C(0)-C(3) or I(0)-I(3)
- the functions F(C,I) can be expanded to show that no terms other than C(0)-C(3) or I(0)-I(3) occur, meaning that the function is entirely made up of those terms, and so it may be implemented with a single exclusive-OR of I(0)-I(3) with C(0)-C(3) followed by a table look-up.

Each group of four cells of the CRC register is entirely dependent on the four cells upstream and the feedback function. It can be seen that continuation of the shifting would not alter the conclusions drawn, as the original values of the register would propagate toward the rightmost position without losing their respective positions in sequence. The F terms would eventually take the places of the C terms. The property of the F function that makes it a deterministic result of the input and the rightmost shift register contents hinges on the number of bits and the number of shifts used. The number of shifts - whether one, two, four, eight, or even sixteen - must equal the number of bits of the register that go into calculating the function, F. This is intuitively correct, since the number of shifts equals the number of input bits, which must be the same as the number of bits of the register in order to have a meaningful exclusive-OR.

Looking into what makes up the F functions, a basis for the table look-up scheme begins to appear. F(0) is the exclusive-OR of the terms I(0) and C(0). If I(i).XOR.C(i) is replaced by the term T(i), the following simplifications can be made:

F(0) = T(0)
F(1) = T(1).XOR.T(0)
F(2) = T(2).XOR.T(1).XOR.T(0)
F(3) = T(3).XOR.T(2).XOR.T(1).XOR.T(0)

and

F(1).XOR.F(3) = T(3).XOR.T(2)
F(0).XOR.F(2) = T(2).XOR.T(1)

The T functions are what result from exclusive-ORing the input with the rightmost CRC shift register contents. If a four bit wide exclusive-OR is taken, the bit-by-bit result can be expressed in the following way:

C(3),C(2),C(1),C(0) .XOR. I(3),I(2),I(1),I(0) = T(3),T(2),T(1),T(0)

A table may then be generated using the T values as the index into the table. The contents of the table are the F functions that must be exclusive-ORed with CRC register contents as shown in Figure 5.

It can be seen that the table need only contain seven bit entries, corresponding to the seven bits in Figure 5 that require other than a straight shift. Furthermore, the index made up of the vector T(3),T(2),T(1),T(0) defines the depth of the table to be sixteen words. In order to preserve generality and accommodate all polynomials, the table should be sixteen words by sixteen bits. The bits of the CRC register that do not require modification would result in the table value being zero.

Figure 6 shows the table contents for the CRC-16 polynomial. The individual bits were generated using exclusive-ORs over the index, as suggested by the simplified equations for the F terms from the preceeding paragraph.

FOUR BIT CRC-16 LOOK-UP TABLE
Figure 6

T(3)	T(2)	T(1)	T(0)	E(15)	E(14)	E(13)	E(12)	E(11)	E(10)	E(09)	E(08)	E(07)	E(06)	E(05)	E(04)	E(03)	(02)	E(01)	E(00)
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	1
0	0	1	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	1
0	0	1	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1
0	1	0	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	1
1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	1
1	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	1
1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
1	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0

The T values in the figure represent the index into the table. The E values are the entries, which are exclusive-ORed with their respective bits in the CRC shift register as presented in Figure 5. The exclusive-OR is done after the shift.

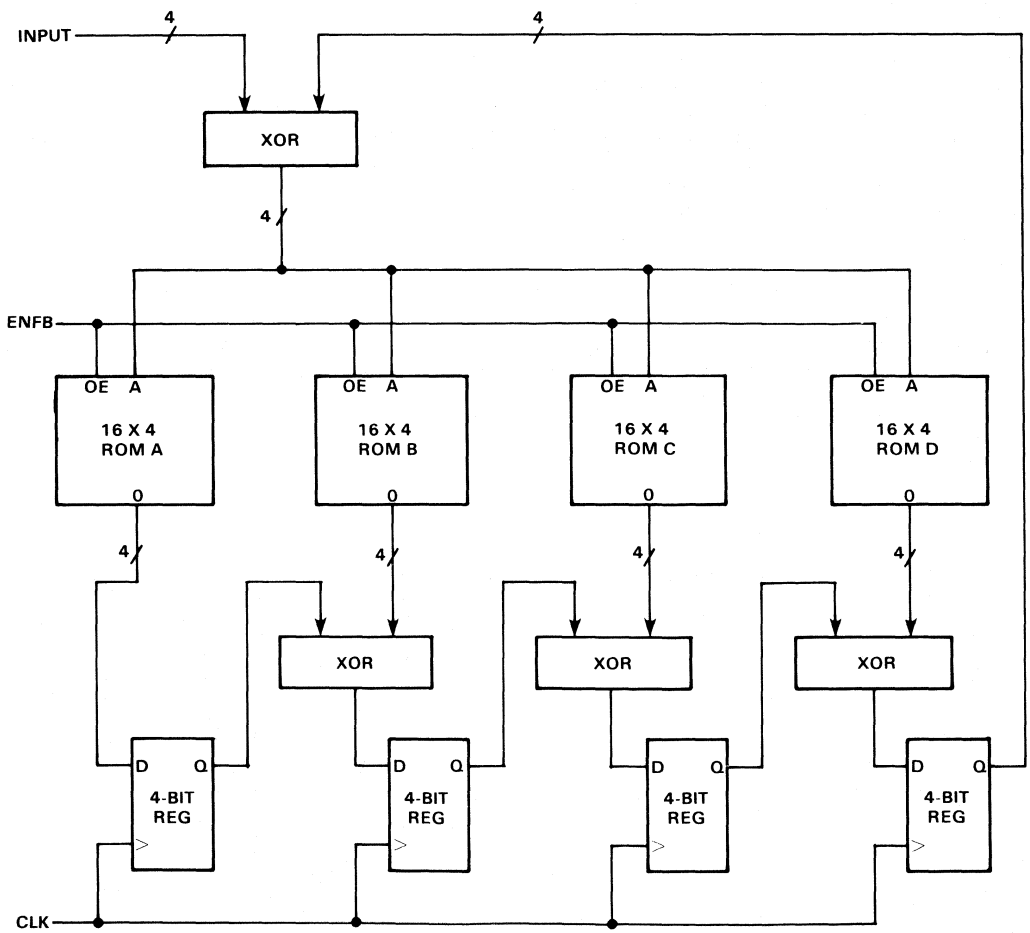
PRACTICAL IMPLEMENTATIONS

The four-bit shift algorithm for CRC calculation can be implemented in hardware or in software. Figure 7 shows the hardware implementation, while Figure 8 shows the software approach. The hardware implementation is shown mainly for reference and would not offer a great improvement in cost effectiveness over the single bit shift approach.

The boxes in Figure 7 marked XOR are 7486's, or equivalent. The ROMs and 4-bit registers are best implemented with 32 X 8 PROMs and 8-bit D-flip-flops, such as 74288s and 74273s. For CRC-16, ROM A and ROM B contain the portion of the table of Figure 6 designated E(15) through E(08), while ROM C and ROM D are programmed with E(07) through E(00). Each shift of the clock accomplishes what four clock shifts did in the implementation in Figure 2.

This implementation takes eight ICs, not counting those required for testing for all zeros. An advantage gained by this approach is that readily available parts are used throughout, and cycling at a clock rate of five Megahertz would yield an equivalent data rate of twenty megabits per second.

FOUR BIT SHIFT CRC HARDWARE IMPLEMENTATION
Figure 7



The software implementation is a straightforward emulation of the hardware implementation of Figure 7. The table is the same as that in Figure 6 for CRC-16. A different generator polynomial would require a different table, but the rest of the routine would be the same.

The software implementation is listed in Figure 8. The routine, including the CRC-16 table, takes up eighty-four bytes of memory. CRC4BT is called once for each byte of data and executes in 150 microseconds in an MK3870 microcomputer with a 4 Megahertz crystal. This is about six times as fast as the single bit shift routine listed in Figure 3, which occupies 42 bytes of memory. The speed improvement realized using the four-bit method permits 3870 Family microcomputers to play an even more important role in data communications applications, where data rates up to 9600 Baud can be readily protected with CRC.

CRC 4-BIT SHIFT SOFTWARE IMPLEMENTATION
Figure 8

```
;
CRCU:    EQU     6                ;UPPER CRC BYTE
CRCL:    EQU     5                ;LOWER CRC BYTE
GP0:     EQU     0                ;TEMPORARY DATA STORAGE
GP1:     EQU     1                ;
GP2:     EQU     2                ;
;
;        CRC FOUR-BIT SHIFT ALGORITHM
;        DATA IN GP0
;
CRC4BT:  LR      K,P              ;SAVE RETURN ADDRESS
:
         DCI     TCRC16           ;SET DATA COUNTER TO BASE OF TABLE
         LR      A,GP0            ;FORM INDEX INTO TABLE
         XS      CRCL             ;'XOR DATA AND LOW NIBBLE OF CRC
         NI      H'F0'            ;
         SL      1                ;2 BYTE TABLE
         ADC                      ;LOOK-UP TABLE ENTRY
;
         LM                       ;READ TABLE
         LR      GP2,A            ;STORE E(15)-E(08) IN GP2
         LM                       ;
         LR      GP1,A            ;E(07) THROUGH E(00) IN GP1
                                  ;
         LR      A,CRCL           ;SHIFT LOWER HALF OF CRC RIGHT 4
         SR      4                ;
         LR      CRCL,A           ;
;
         LR      A,CRCU           ;APPEND PART FROM UPPER HALF
         SL      4                ;
         XS      CRCL             ;MERGE TWO PARTS TOGETHER
         XS      GP1              ;DO THE XOR WITH TABLE VALUE
         LR      CRCL,A           ;LOWER HALF IS DONE
;
         LR      A,CRCU           ;SHIFT UPPER HALF
         SR      4                ;
         XS      GP2              ;DO XOR WITH TABLE VALUE
         LR      CRCU,A           ;UPPER PART DONE
; FIRST NIBBLE SHIFT IS DONE
;
         DCI     TCRC16           ;SET DATA COUNTER TO BASE OF TABLE
         LR      A,GP0            ;FORM INDEX INTO TABLE
         SR      4                ;THIS ONE IS FOR THE UPPER NIBBLE
         XS      CRCL             ;XOR DATA AND LOW NIBBLE OR CRC
         NI      H'F0'            ;
         SL      1                ;2 BYTE TABLE
         ADC                      ;LOOK-UP TABLE ENTRY
;
```

CRC 8-BIT SHIFT SOFTWARE IMPLEMENTATION (Continued)

```
                LM                              ;READ TABLE
                LR      GP2,A                   ;STORE E(15)-E(08) IN GP2
                LM                              ;
                LR      GP1,A                   ;E(07) THROUGH E(00) IN GP1
;
;
;
;
                LR      A,CRCL                  ;SHIFT LOWER HALF OF CRC RIGHT 4
                SR      4                       ;
                LR      CRCL,A                  ;
;
                LR      A,CRCU                  ;APPEND PART FROM UPPER HALF
                SL      4                       ;
                XS      CRCL                    ;MERGE TWO PARTS TOGETHER
                XS      GP1                     ;DO THE XOR WITH TABLE VALUE
                LR      CRCL,A                  ;LOWER HALF IS DONE
;
                LR      A,CRCU                  ;SHIFT UPPER HALF
                SR      4                       ;
                XS      GP2                     ;DO XOR WITH TABLE VALUE
                LR      CRCU,A                  ;UPPER PART DONE
;
                PK                              ;RETURN TO CALLER
:
:
TCRC16:         DEFW    B'0000000000000000'             ;T=0000
                DEFW    B'1100110000000001'             ;T=0001
                DEFW    B'1101100000000001'             ;T=0010
                DEFW    B'0001010000000000'             ;T=0011
                DEFW    B'1111000000000001'             ;T=0100
                DEFW    B'0011110000000000'             ;T=0101
                DEFW    B'0010100000000000'             ;T=0110
                DEFW    B'1110010000000001'             ;T=0111
;
                DEFW    B'1010000000000001'             ;T=1000
                DEFW    B'0110110000000000'             ;T=1001
                DEFW    B'0111100000000000'             ;T=1010
                DEFW    B'1011010000000001'             ;T=1011
                DEFW    B'0101000000000000'             ;T=1100
                DEFW    B'1001110000000001'             ;T=1101
                DEFW    B'1000100000000001'             ;T=1110
                DEFW    B'0100010000000000'             ;T=1111
;
;
;
```

ALTERNATIVE APPROACHES

This section considers some of the variations and alternatives for the protection of data.

EIGHT BIT SHIFT ALGORITHM

Section 2 covered the grouping of the basic CRC shift algorithm into four bits at a time. It is also possible to gain additional speed at the expense of memory usage by extending the approach to eight bits. The hardware would use two eight bit exclusive-OR gates, two eight bit registers, and a 256 X 16 ROM.

Figure 10 lists the software routine for implementing the eight bit approach. The 256 X 16 table is built by first

continuing the symbolic shift procedure, outlined in Figure 5, until the eighth shift. This produces the following list:

AFTER EIGHTH SHIFT

F(7)
F(6)
F(5).XOR.F(7)
F(4).XOR.F(6)

F(3).XOR.F(5)
F(2).XOR.F(4)
F(1).XOR.F(3)
F(0).XOR.F(2)

F(1).XOR.C(15)
F(0).XOR.C(14)
C(13)
C(12)

C(11)
C(10)
C(09)
F(7).XOR.C(08)

Then, as before, the next step is to build the generator expressions for the F functions in terms of the T functions as follows:

F(0) = T(0) where T(i) = I(i).XOR.C(i)
F(1) = T(1).XOR.T(0)
F(6) = T(6).XOR.T(5).XOR.T(4).XOR.T(3).XOR.T(2). XOR.T(1).XOR.T(0)
F(7) = T(7).XOR.T(6).XOR.T(5).XOR.T(4).XOR.T(3). XOR.T(2).XOR.T(1).XOR.T(0) and
F(5).XOR.F(7) = T(7).XOR.T(6)
F(4).XOR.F(6) = T(6).XOR.T(5)
F(3).XOR.F(5) = T(5).XOR.T(4)
F(2).XOR.F(4) = T(4).XOR.T(3)
F(1).XOR.F(3) = T(3).XOR.T(2)
F(0).XOR.F(2) = T(2).XOR.T(1)

The table is generated by computing parity over the address, as indicated in the above expressions, letting each T term refer to the corresponding table address bit, (assuming the table base address to be zero). The C terms without coefficients in the list are assumed to have zero entries in the table, and so the corresponding column in the table would be filled with zeros.

The eight bit routine shown in Figure 10 executes in an average of about 82 microseconds per byte, or about 45 percent faster than the four bit method. Memory usage for the eight bit routine plus one look-up table is 568 bytes.

LRC

LRC, or Longitudinal Redundancy Code, is a special case of CRC where the particular polynomial chosen results in the same CRC code as would be obtained by doing a sixteen bit wide exclusive-OR once every sixteen bits. If the data stream were represented as a succession of sixteen bit words, the LRC code added to the end of the stream would equal the first word exclusive-ORed with the second, exclusive-ORed with the third, and so on. When the check is made at the receiver, the result is zero if no errors occurred, since the exclusive-OR of anything with itself is zero.

LRC is also often done on an eight bit word length, since software implementation is a little bit simpler than with sixteen bits.

LRC is a form of CRC, and as such it can be handled by the CRC implementations discussed in this app note. The polynomial for LRC-16 is X(16)+1; that for LRC-8 is X(8)+1. A table for LRC-16 in the four bit implementation of section 2 could be constructed in the same way as the table for CRC-16 was done. The result is listed in Figure 9.

FOUR BIT LRC-16 LOOK-UP TABLE
Figure 9

T(3)	T(2)	T(1)	T(0)	E(15)	E(14)	E(13)	E(12)	E(11)	E(10)	E(09)	E(08)	E(07)	E(06)	E(05)	E(04)	E(03)	E(02)	E(01)	E(00)
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	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

CRC 8-BIT SHIFT SOFTWARE IMPLEMENTATION

Figure 10

```
;
CRCU:     EQU     6                   ;UPPER CRC BYTE
CRCL:     EQU     5                   ;LOWER CRC BYTE
GP0:      EQU     0                   ;TEMPORARY DATA STORAGE
;
;         CRC EIGHT-BIT SHIFT ALGORITHM
;         DATA IN GP0
;         CRC TABLE IS STORED LOWER BYTE BEFORE UPPER BYTE TO MAKE
;             THIS IMPLEMENTATION FASTER
;
CRC8BT:   LR      K,P                 ;SAVE RETURN ADDRESS
;
          DCI     T8CR16              ;SET DATA COUNTER TO BASE OF TABLE
          LR      A,GP0               ;FORM INDEX INTO TABLE
          XS      CRCL                ;XOR DATA AND LOW BYTE OF CRC
          BM      MXOR                ;UPPER HALF OF TABLE IF MINUS
;
          SL      1                   ;FORM INDEX TO 2-BYTE TABLE
          BM      MSHL                ;2ND QUADRANT OF TABLE IF MINUS
          ;
          ADC                         ;ELSE 1ST QUADRANT OF TABLE
          LR      A,CRCU              ;XOR UPPER CRC WITH LOWER BYTE
          XM                          ;     FROM TABLE
          LR      CRCL,A              ;SHIFT RIGHT 8
          LM                          ;GET UPPER BYTE FROM TABLE
          LR      CRCU,A              ;REPLACES UPPER CRC BYTE
;
          PK                          ;THIS LEG TAKES 33.5 CYCLES
;
;
MSOR:     SL      1                   ;FORM INDEX TO 2-BYTE TABLE
          BM      MSXL                ;FOURTH QUADRANT OF TABLE IF MINUS
;
          ADC                         ;ELSE THIRD QUADRANT
          LI      H'40'               ;OFFSET TO THIRD QUADRANT
          ADC                         ;
          ADC                         ;
;   NEXT SECTION REPEATED IN THE INTEREST OF SPEED
;         LR      A,CRCU              ;XOR UPPER CRC WITH LOWER BYTE
          XM                          ;     FROM TABLE
          LR      CRCL,A              ;SHIFT RIGHT 8
          LM                          ;GET UPPER BYTE FROM TABLE
          LR      CRCU,A              ;REPLACES UPPER CRC BYTE
;
          PK                          ;THIS LEG TAKES 41.5 CYCLES

;
;

;
;

;

;
```

CRC 4-BIT SHIFT SOFTWARE IMPLEMENTATION (Continued)

```
MSHL:   NI      H'7F'           ;TRIM OFF MINUS SIGN
S       ADC                     ;INDEX INTO TABLE
S;
        LI      H'40'           ;THIS PUTS IT TO 2ND QUADRANT
        ADC
; NEXT SECTION REPLICATED TO KEEP IT FAST
        LR      A,CRCU          ;XOR UPPER CRC WITH LOWER BYTE
        XM                      ;     FROM TABLE
        LR      CRCL,A          ;SHIFT RIGHT 8
        LM                      ;GET UPPER BYTE FROM TABLE
        LR      CRCU,A          ;REPLACES UPPER CRC BYTE
;
        PK                      ;THIS LEG TAKES 41.5 CYCLES
;
;
MXSL:   NI      H'7F'           ;TRIM OFF SIGN BIT
        ADC                     ;INDEX INTO TABLE
        LI      H'40'           ;SET THE RIGHT QUADRANT
        ADC                     ;
        ADC                     ;GOES TO FOURTH QUADRANT
        ADC                     ;
;  NEXT SECTION REPEATED TO AVOID BRANCH
        LR      A,CRCU          ;XOR UPPER CRC WITH LOWER BYTE
        XM                      ;   FROM TABLE
        LR      CRCL,A          ;SHIFT RIGHT 8
        LM                      ;GET UPPER BYTE FROM TABLE
        LR      CRCU,A          ;REPLACES UPPER CRC BYTE
;
        PK                      ;THIS LEG TAKES 47 CYCLES
;
;
;
T8CR16: DEFW    0               ;TABLE EXCLUDED FOR BREVITY
;  THIS TABLE IS DIFFERENT FROM THAT OF THE FOUR BIT TABLE
;IN THAT IT IS STORED LOW BYTE BEFORE UPPER, RATHER THAN
;HIGH BYTE FIRST. THIS IS DONE TO MAKE THE ROUTINE OPERATE FASTER.
;
;
;
;
```

CHECKSUM

The checksum is an accumulation of the remainder of modulo 256 addition of a string of data organized in bytes. This method of error detection is in widespread use throughout the microcomputer industry since it is easily generated and is very effective in detecting errors. Often, in the case of data which is coded into ASCII characters that represent the data in hexadecimal form, the checksum is taken over the values of the hexadecimal numbers rather than over the actual bit patterns themselves.

Typically, the initial value of the eight bit checksum is minus one. This is so that when zero occurs often in the data, the effectiveness of the code is not diminished. Thus, when the checksum has been taken over received data, the final value is minus one, and not zero.

CONCLUSIONS AND SUMMARY

The protection of data involves knowing when an error occurs. None of the methods described will detect all possible errors that could occur in a data transfer system. The effectiveness of a code is a measure of how low the probability is that an error could get through the code system undetected.

CRC based systems can have different effectiveness factors dependent on the randomness of the data transferred and on the actual generator polynomial that is used.

Storage systems impose their own set of characteristics in determining the randomness of errors, and thus which method of error detection to employ.

CRC-16 is most common in data communications and in disk applications because it is especially effective where errors are more likely to occur in bursts. Where errors are more likely to be single bit, or two bit errors, LRC may prove equally effective. Checksums are used to protect assembler object code integrity in the Mostek development software, since it is virtually independent of the storage media, and also since it is relatively easy to generate and check.

The four bit CRC algorithm described offers a good trade-off between execution speed and memory usage for applications that include data communications controllers, mini-floppy disk controllers, tape controllers, and many more. The approach outlined can be used to advantage in any computer that facilitates the table look-up.

MOSTEK®

EXPANDING MOSTEK'S F8 EXTERNAL INTERRUPT CAPABILITIES

Application Note

INTRODUCTION

One of the considerations involved in the design of any microprocessor based system is how to structure the interface between the peripherals (inputs or devices being controlled) and the CPU. The data line interface is usually dictated by the peripheral itself (e.g., a paper tape reader is eight bits of parallel data, a teletype is two lines of serial data, and a switch or front panel lamp usually only requires one line of data). The control lines of these peripherals however, can be handled in one of two basic ways by the system designer. The first method of handling these control lines, which is probably the most common, is to have the CPU periodically scan the control lines (connected to a I/O Port) to see if they require service. This is done by a small program which inputs the control lines through an I/O Port into the accumulator. They are then tested to determine if a line is active and the program flow diverted to service the active control line. The second method is to allow these control lines to interrupt the processor and divert program flow to service that peripheral. Servicing of these control inputs in a F8 based system is the topic of this application note with particular emphasis placed on implementing interrupt driven systems.

SCANNED VS INTERRUPT DRIVEN SYSTEMS

The basic difference between scanned and interrupt driven systems is that in a scanned system the peripherals are checked periodically to see if they need service. This periodic interval can be determined by the count down of a hardware timer (a software timer could be used, but the CPU would be tied up implementing a ripple counter—not a very effective use of the microprocessor). This technique is good for peripherals which can wait for service by the CPU (the maximum time would be the time between counter outputs), and good examples are any peripheral activated or observed by a human. For example a keyboard/display might be scanned at 1 ms intervals, as determined by the timer, which would be slow by microprocessor standards but exceedingly fast by human standards (after pressing a key or throwing a switch an extra 1 ms delay in service would not be noticeable).

On the other hand many microprocessors are involved in the control of fast peripherals (Floppy Disk) or real time systems where quick response by the processor is required. In these situations, interrupt driven systems are mandatory, because the processor can be diverted from its present task to service the interrupting device in the order of tens of microseconds. Scanned systems are usually perferred by the system designer because they usually require less hardware, especially when implemented in a F8 System with its hardware timers. Figure 1 is a flow chart of a scanned system where the interval between scans is determined by the value preset into the timer. Note that priority is established by the order in which the control bits are tested and can be changed entirely by software.

FLOWCHART FOR SCANNING N CONTROL LINES

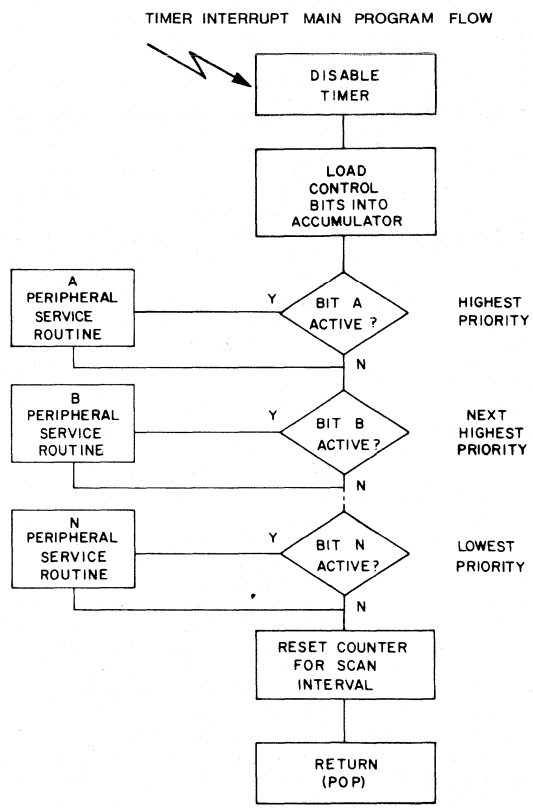


Figure 1

HARDWARE VECTORED INTERRUPTS

The interrupt technique used by F8 Family devices capable of interrupting the CPU (PSU, PIO, or SMI) is to have the interrupting device provide to the CPU a Interrupt Vector unique to that interrupt. The CPU then loads this vector directly into the program

counter (saving the previous program counter in P) directing the CPU to the service routine for this interrupt. This technique provides a fast response to the interrupt because no time is consumed in polling to locate the interrupting device. In addition to providing automatic vectoring of the interrupts, the F8 devices provide automatic prioritizing of the interrupts. Priority is determined by the placement of the interrupting device in a daisy chain structure — a location closer to the CPU means higher priority — as shown in Figure 3. $\overline{ICB}$ is an output from the F8 CPU to indicate if interrupts have been enabled by the use of an EI instruction in the program being executed. $\overline{ICB}$ goes low when interrupts have been enabled, thereby enabling the daisy chain of interrupting devices. One or more of the three $\overline{EXT\ INT}$ inputs shown goes low signaling a request(s) for service by one or more of the peripherals. The device or devices that have $\overline{EXT\ INT}$ low now pull their $\overline{INT\ REQ}$ line low (assuming interrupts are not disabled at the local level) signaling the processor to begin an interrupt service sequence. The status of the $\overline{INT\ REQ}$ line is tested by the CPU at the end of every instruction which is not privileged. Privileged instructions cannot be interrupted so the CPU waits until the end of the next instruction (which is not privileged) to test the $\overline{INT\ REQ}$ line. When the CPU finds the $\overline{INT\ REQ}$ line low it begins the interrupt sequence by saving the Program Counter in P and using the ROM Control Lines to command the interrupting peripheral to transfer its vector address to the Program Counter. The 3851 is the highest priority device in Figure 3 and if its $\overline{EXT\ INT}$ line is low it sets its $\overline{PRI\ OUT}$ signal high thereby disabling all lower priority devices and outputs its vector address on the Data Bus. Should the PSU not be the interrupting device, it leaves its $\overline{PRI\ OUT}$ signal low passing the request to the second device in the chain (the PIO in this case). If the PIO is interrupting, it raises its $\overline{PRI\ OUT}$ line to a logic one and outputs its vector address. $\overline{PRI\ OUT}$ going high prevents all devices of lower priority from outputing their vector address even though they may be trying to interrupt. Twenty two cycles of the Φ clock are required to complete this interrupt vector fetch sequence. The next event that occurs is an instruction fetch from the location specified from the vector address. The SMI doesn't have a $\overline{PRI\ OUT}$ signal therefore it must be the lowest priority device in the system. The time required to get to an interrupt service routine can be calculated as shown in Figure 2 (at a 2 MHz Φ rate).

INTERRUPT VECTOR FETCH

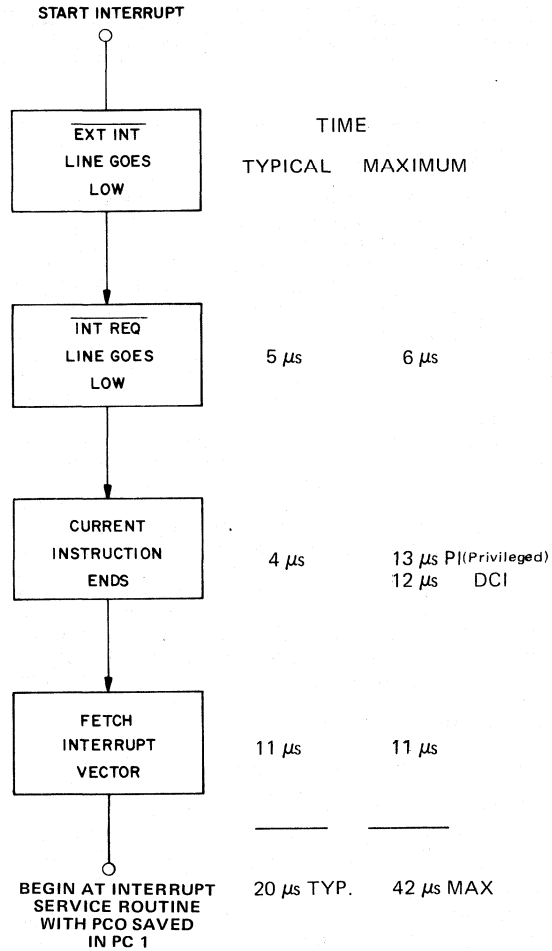


Figure 2

The time from an interrupt striking to the start of execution of its service routine is highly dependent on the instruction being executed at the time of the interrupt. The maximum number was based on a long privileged instruction such as PI followed by a long non-privileged instruction such as DCI. The typical instruction time is based on a 2 cycle instruction although many F8 instructions are one byte/one cycle instructions. The 6.0μs max number represents the propagation delay through the peripheral device from $\overline{EXT\ INT}$ to $\overline{INT\ REQ}$ (interrupts from the timer do not incur this delay). Once the $\overline{INT\ REQ}$ is recognized, 22 cycles are required to stack the program counter and fetch the interrupt vector. One technique that can be used to minimize the maximum delay that would be incurred upon an interrupt is to constrain the instructions that are executed when the interrupt is expected. A method that would reduce the maximum delay from the interrupt striking to the execution of the first instruction of the service routine would be to put the processor in a branch on self loop (BR*). This essentially provides a wait for interrupt situation with the CPU running in a loop waiting for the interrupt.

EXPANDING INTERRUPT INPUTS IN A MINIMUM SYSTEM

In a two-chip F8 microcomputer system (MK 3850 CPU and MK 3851 PSU) the system can be interrupted by either the timer in the PSU or the $\overline{EXT\ INT}$ line of the PSU. Thirty-two lines of bi-directional I/O are available and it may be desirable to have more than one input capable of interrupting

F8 SYSTEM INTERRUPT CONNECTION

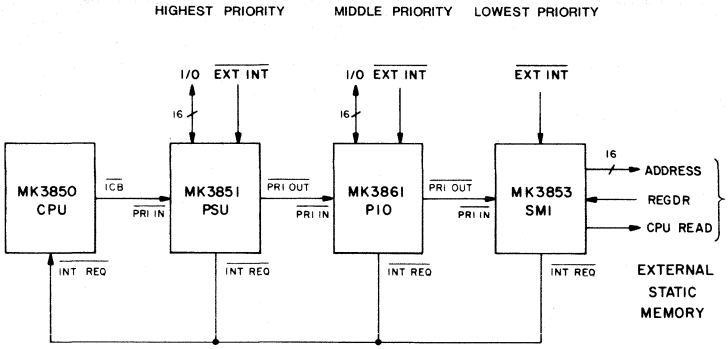
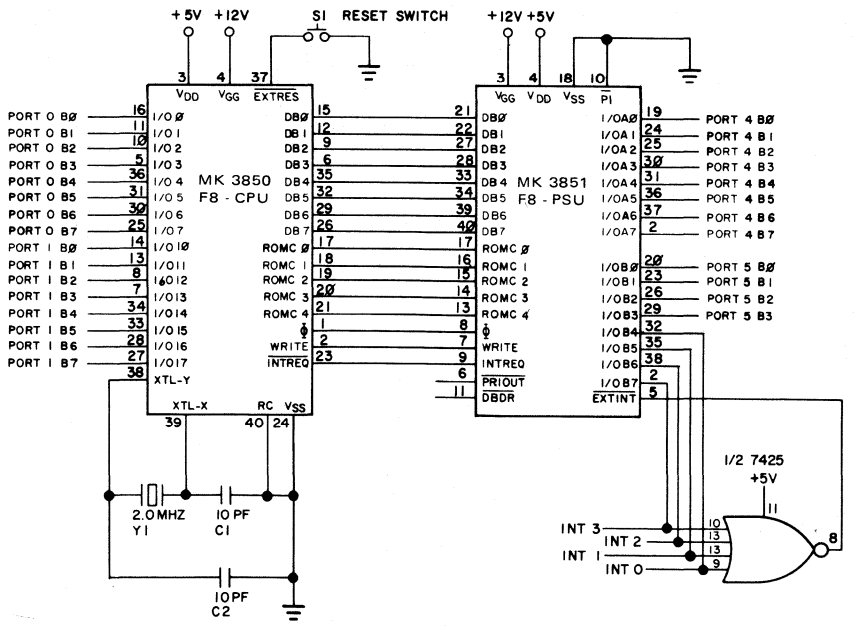


Figure 3

rupting the system. Figure 4 depicts this minimum F8 system, with four signals (INT0-INT3) capable of interrupting the system. The four external interrupting signals are defined active high and the presence of any one in the high state causes the output of the NOR gate to go low causing the interrupt. The interrupt service routine flowchart to locate the interrupting input is shown in Figure 5, with the actual program in Figure 6.

This service routine is entered with the interrupts automatically disabled at the CPU so that no further interrupts can occur until the interrupt is cleared by its service routine. The port containing the INT0-INT3 signals is loaded into the accumulator and tested to determine if bit 7 is low (a positive number). If bit 7 is low INT3 is active and the branch is taken to the service routine for INT3 (SERV3) (there is an inversion from the Port to the accumulator). If bit 7 is high a shift left one instruction is performed on the accumulator and it is again tested for bit 7 = 0 (bit 6 shifted). This process continues until all four of the interrupt lines have been tested. If an active interrupt bit has been found the proper service routine is branched to in order to service the active device and

EXPANSION OF INTERRUPT INPUTS IN A MINIMUM F8 SYSTEM



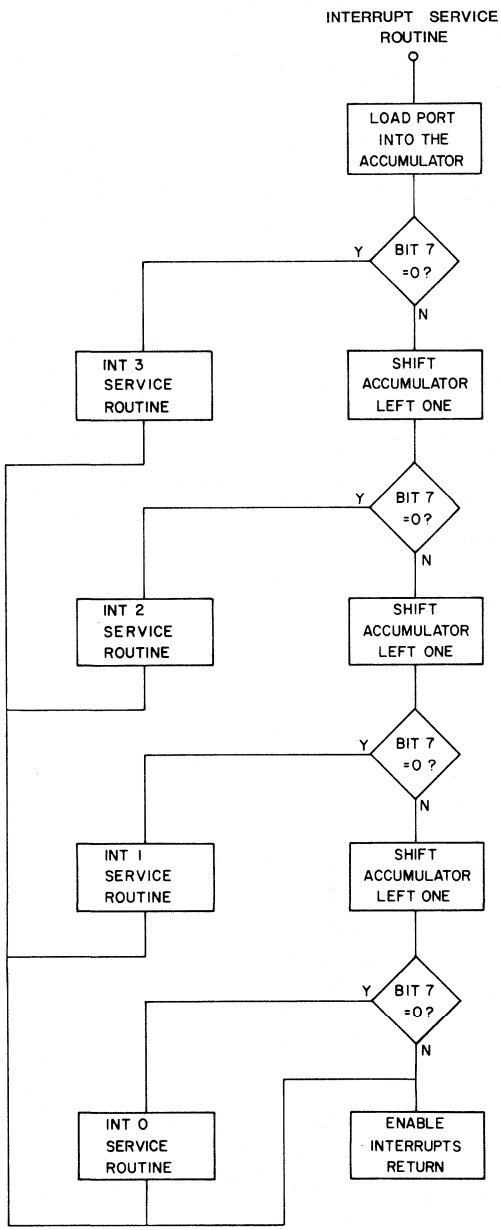
NOTE: MK 3870 Single Chip F8 will replace this two chip minimum system.

Figure 4

clear the interrupt. The routine ends by enabling the interrupts at the CPU and returning to the main program flow should no interrupt be found.

The additional time required to locate the active interrupt is a function of which interrupt is active due to the polling used. As shown in Figure 6, the additional delay to service interupts produced by polling varies from 15 μs for the highest priority device to 42 μs for the lowest priority device. To these times must be added the delays calculated earlier of 20 μs typical and 42 μs maximum which is required to get to the polling routine.

FLOWCHART OF INTERRUPT SERVICE ROUTINE



INTERRUPT SERVICE ROUTINE TO LOCATE INTERRUPTING DEVICE

CUMM μS	μS			
	8 INTSVC	INS	PORT	GET SIGNALS
INT 3 15	7	BP	SERV 3	INT 3 ACTIVE 7
	2	SL	1	NO, SHIFT LEFT
INT 2 24	7	BP	SERV 2	INT 2 ACTIVE 7
	2	SL	1	NO, SHIFT LEFT
INT 1 33	7	BP	SERV 1	INT 1 ACTIVE 7
	2	SL	1	NO, SHIFT LEFT
INT 0 42	7	BP	SERV 0	
	4	EI		ENABLE INTERRUPTS
	4	POP		RETURN

Figure 6

SINGLE CHIP MICROCOMPUTER

The MK 3870 Single Chip Microcomputer is the natural evolution of the F8 chip set. It will combine the functions of the 3850/3851 onto a single chip with the additions of another 1K bytes of ROM storage and an improved timer/interrupt structure. The techniques discussed in this application note apply also to the single chip F8 as it is software and hardware compatible with the multiple chip F8 family.

MOSTEK®

USING MOSTEK'S F8 IN A SCANNED SEVEN-SEGMENT DISPLAY APPLICATION

Application Note

INTRODUCTION

Many microprocessor based devices require a numeric display as an integral part of the system. For reasons of cost and reliability, it is usually desirable to keep the chip count as low as possible with the microprocessor performing the control logic in software. Time multiplexed digit scanning is a common solution and works very well using a single F8 port for up to 8 digits.

THEORY OF OPERATION

An eight digit display can be scanned with one F8 port (fig. 1) by using half of the port for the BCD number and half for the digit select. When using the digit scanning technique an 'image' of the display must be maintained in memory, with a byte (or half byte) of memory containing the BCD number to be displayed in each of the eight digits. The following five steps show the basic control the software is required to execute:

Step 1 Output digit select and BCD number for this digit (from 'image')

Step 2 Turn on strobe

Step 3 Delay

Step 4 Turn off strobe

Step 5 Increment digit select, return to step 1

The scan rate should be fast enough to prevent the display from 'flickering'. It has been found that a 80 to 100Hz rate is sufficient for a stationary display. An approximate 100Hz rate is achieved in an eight digit display by making the delay in step 3, 1.25 milliseconds.

Maximum brilliance will be provided by leaving the strobe on for the whole delay time. This provides a 1/8 or 12.5% duty cycle. Reducing the strobe width will reduce the duty cycle and cause the display to be dimmer.

Interdigit blanking to prevent a blurring effect is accomplished by strobing the digit decoder after digit select/BCD number data is present on the port and removing the strobe before changing the data (see fig. 2)

EXAMPLE HARDWARE DESIGN

The example design in Figure 3 shows the hardware simplicity in an LED display scanning circuit interfaced to the F8. Bits 0-2 are used to select the digit, bit 3 as a strobe and bits 4-7 for the BCD number.

In this eight digit display the current required from the segment drivers and anode drivers is approximately 6-8 times what it would be for a static non-scanned display of equal brillance because only one digit is receiving current at a time (12.5% Duty Cycle).

NUMERIC DISPLAY BLOCK DIAGRAM

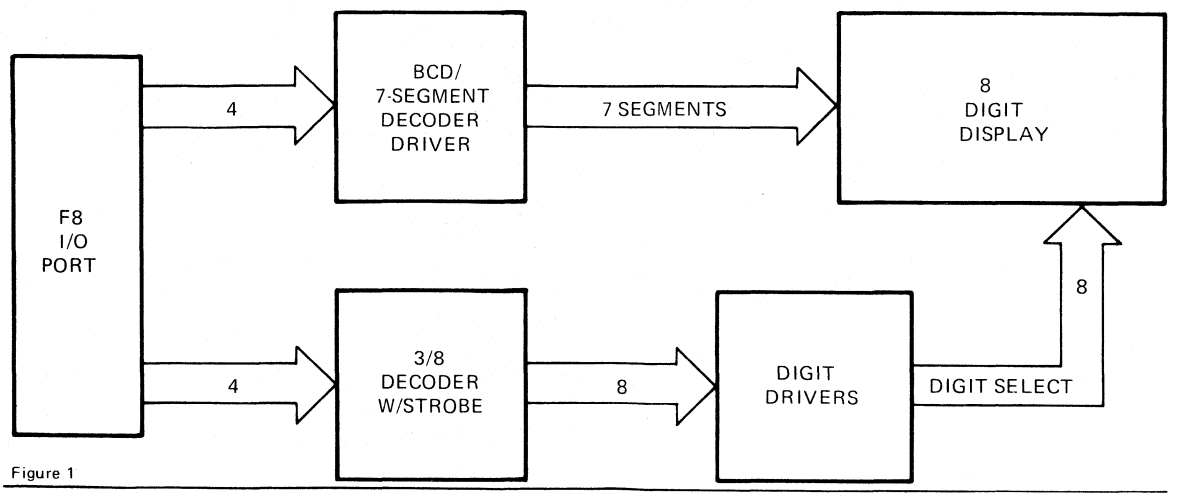


Figure 1

INTERDIGIT BLANKING

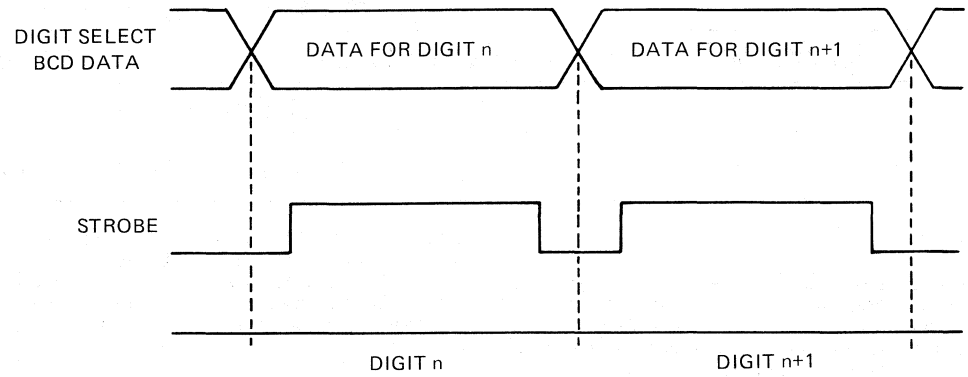


Figure 2

The SN7447 seven segment decoder/driver sinks 40mA per segment which will supply a maximum average current of 5mA per segment to each digit. This is an acceptable current level for many 7 segment LED displays such as the .43 inch HP7650. Since the anode transistors must drive seven segments, they will be required to source 280mA peak at a 12.5% duty cycle. Many discrete transistors (such as the 2N2907) and transistor arrays will handle this load.

7 SEGMENT DISPLAY INTERFACE

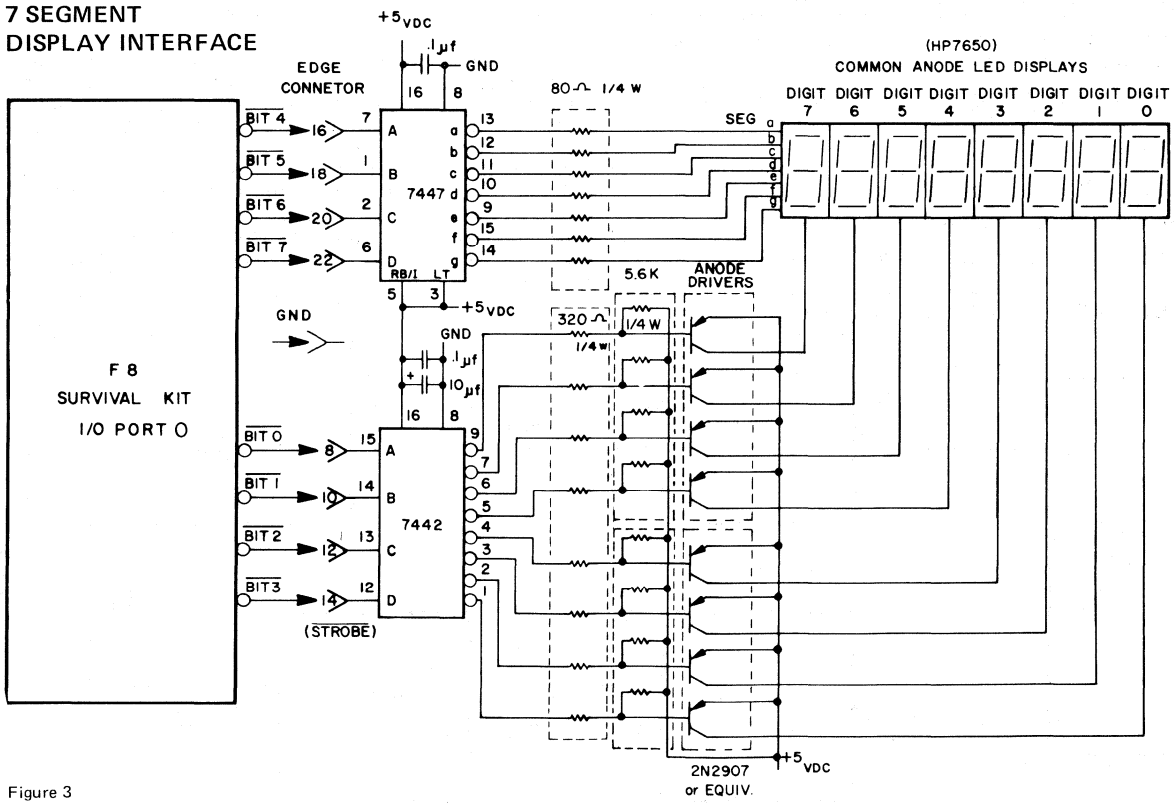


Figure 3

MAIN PROGRAM

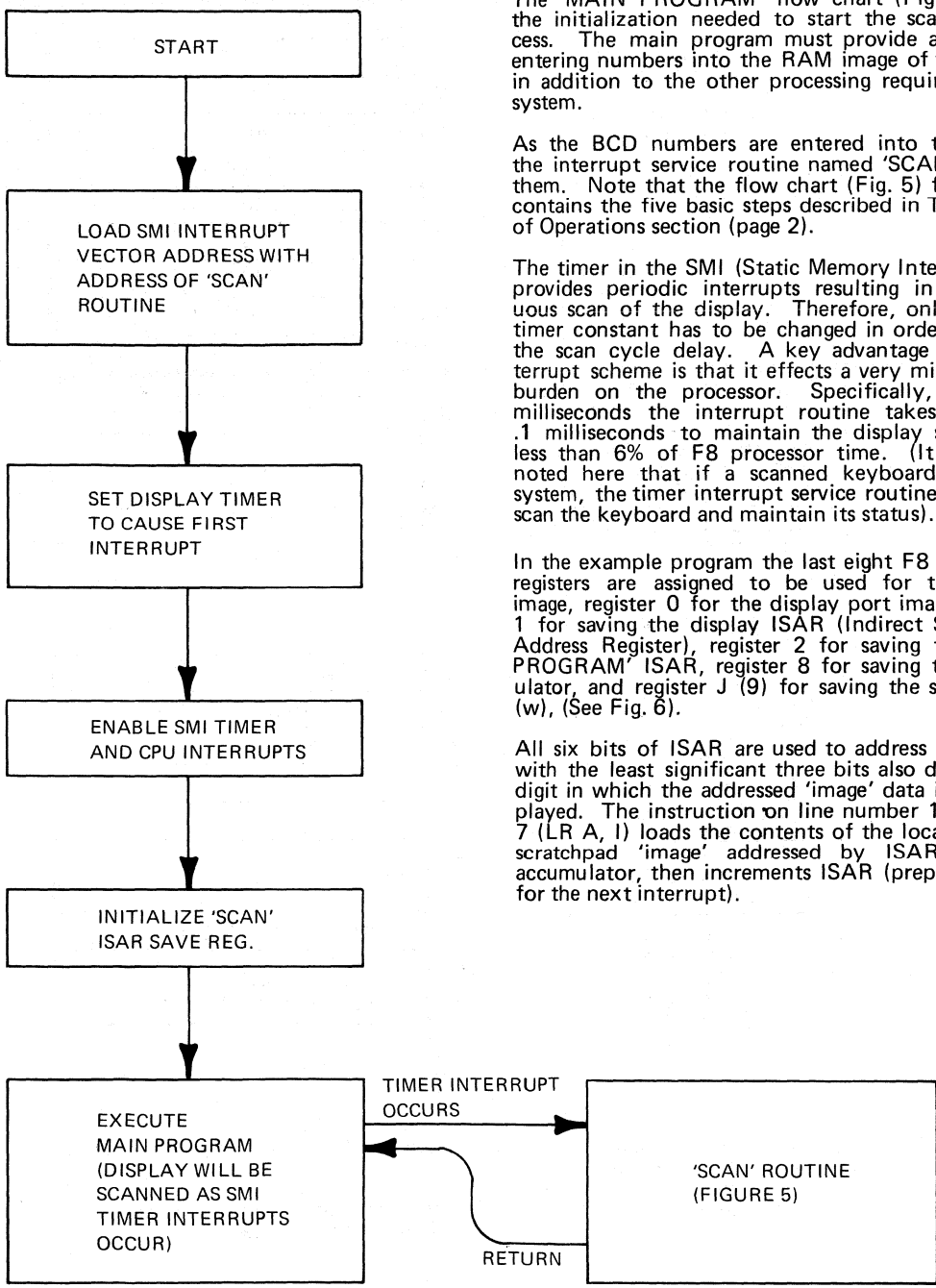


Figure 4

EXAMPLE CONTROL SOFTWARE

The 'MAIN PROGRAM' flow chart (Fig. 4) shows the initialization needed to start the scanning process. The main program must provide a means of entering numbers into the RAM image of the display in addition to the other processing required by the system.

As the BCD numbers are entered into the 'image' the interrupt service routine named 'SCAN' displays them. Note that the flow chart (Fig. 5) for 'SCAN' contains the five basic steps described in The Theory of Operations section (page 2).

The timer in the SMI (Static Memory Interface) chip provides periodic interrupts resulting in a continuous scan of the display. Therefore, only the SMI timer constant has to be changed in order to adjust the scan cycle delay. A key advantage to this interrupt scheme is that it effects a very minimal time burden on the processor. Specifically, every 1.5 milliseconds the interrupt routine takes less than .1 milliseconds to maintain the display scan, using less than 6% of F8 processor time. (It should be noted here that if a scanned keyboard is in the system, the timer interrupt service routine could also scan the keyboard and maintain its status).

In the example program the last eight F8 scratchpad registers are assigned to be used for the display image, register 0 for the display port image, register 1 for saving the display ISAR (Indirect Scratchpad Address Register), register 2 for saving the 'MAIN PROGRAM' ISAR, register 8 for saving the accumulator, and register J (9) for saving the status word (w), (See Fig. 6).

All six bits of ISAR are used to address the 'image' with the least significant three bits also defining the digit in which the addressed 'image' data is to be displayed. The instruction on line number 12 of figure 7 (LR A, I) loads the contents of the location in the scratchpad 'image' addressed by ISAR into the accumulator, then increments ISAR (preparing ISAR for the next interrupt).

INTERRUPT SERVICE ROUTINE FOR NUMERIC DISPLAY

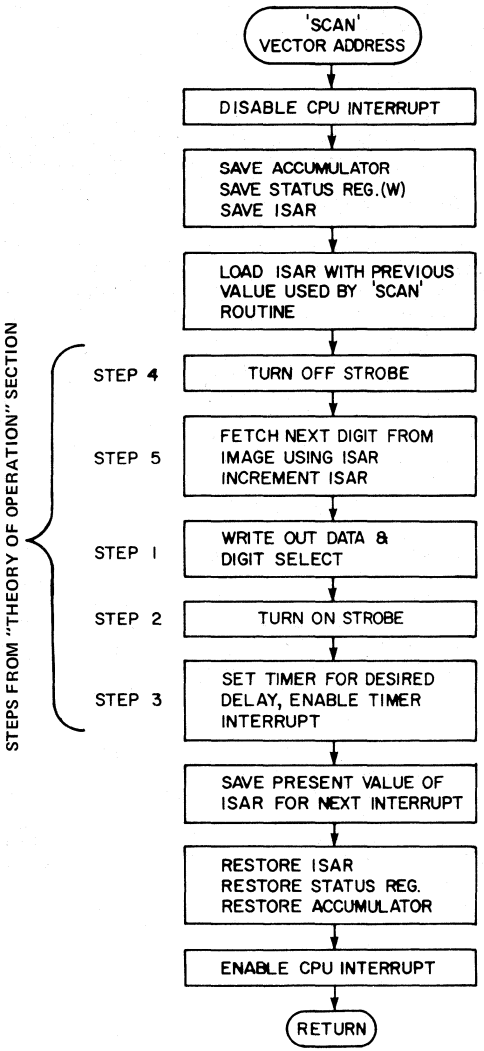


Figure 5

F8 SCRATCHPAD REGISTER USAGE MAP

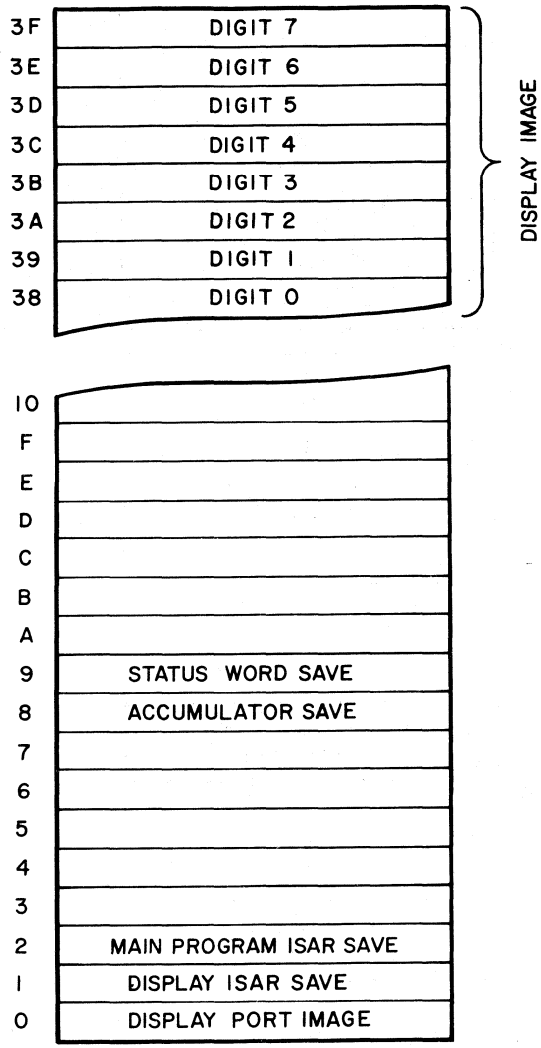


Figure 6

Output port H'F' is the timer constant register in the SMI chip (see line 1C in figure 7). Port H'E' is a register used to enable the timer interrupt in the SMI (line 1F). Note also that all outputs to the display port are 'OUTS 0' selecting port 0 (line E, line 17 & line 19).

The program listing (Fig. 7) contains comments that specify the purpose of each instruction.

SDB RESIDENT ASSEMBLER LISTING Figure 7

```
LINE # ADDRESS  OBJECT CODE        SOURCE CODE          COMMENTS
0000                        *
0001                        *
0002                        *
0003                        *
0004                        ORG   H'700'
0005 0700 1A        SCAN    DI                         DISABLE CPU INTERRUPTS
0006 0701 58                LR    8,A                  SAVE ACCUMULATOR
0007 0702 1E                LR    J,W                  SAVE STATUS REG
0008 0703 0A                LR    A,IS                 LOAD ISAR INTO ACCUMLATOR
0009 0704 52                LR    2,A                  SAVE ISAR FROM MAIN PROGRAM
000A 0705 41                LR    A,1                  LOAD ACCUMULATOR WITH PREV ISAR
000B 0706 0B                LR    IS,A                 LOAD ISAR FOR SCAN
000C 0707 40                LR    A,0                  LOAD PREVIOUS DISPLAY PORT DATA
000D 0708 21 F7             NI    H'F7'                MASK OUT STROBE BIT
000E 070A B0                OUTS  0                    TURN OFF STROBE
000F 070B 0A                LR    A,IS                 LOAD ISAR INTO ACCUMULATOR
0010 070C 21 07             NI    7                    MASK OUT ISAR(U)
0011 070E 50                LR    0,A                  ISAR(L) TO R0 FOR DIGIT # SELECT
0012 070F 4D                LR    A,I                  GET BCD DATA USING ISAR, INC ISAR
0013 0710 15                SL    4                    MOVE IT TO MS HALF OF ACCUMULATO
0014 0711 C0                AS    0                    ADD DIGIT # TO BCD DATA
0015 0712 18                COM                        INVERT DATA SINCE PORTS NEG TRUE
0016 0713 21 F7             NI    H'F7'                MASK OUT STROBE BIT
0017 0715 B0                OUTS  0                    WRITE NEW DATA OUT (NO STROBE)
0018 0716 22 08             OI    8                    STROBE BIT ON
0019 0718 B0                OUTS  0                    TURN ON STROBE
001A 0719 50                LR    0,A                  SAVE DISPLAY PORT DATA
001B 071A 20 C4             LI    H'C4'                TIMER CONSTANT
001C 071C BF                OUTS  H'F'                 WRITE TO SMI TIMER
001D 071D 73                LIS   3                    LOCAL INTERRUPT ENABLE BITS
001E 071E BE                OUTS  H'E'                 ENABLE LOCAL INTERRUPTS
001F 071F 0A                LR    A,IS                 LOAD ISAR INTO ACCUMULATOR
0020 0720 51                LR    1,A                  SAVE DISPLAY SCAN ISAR
0021 0721 42                LR    A,2                  LOAD MAIN PROGRAM ISAR VALUE
0022 0722 0B                LR    IS,A                 RESTORE ISAR WITH IT
0023 0723 1D                LR    W,J                  RESTORE STATUS REG
0024 0724 48                LR    A,8                  RESTORE ACCUMULATOR
0025 0725 1B                EI                         ENABLE CPU INTERRUPTS
0026 0726 1C                POP                        RETURN TO MAIN PROGRAM
0027                        END
00

SCAN 0700
```

ALTERNATE DESIGN APPROACHES

There are several other approaches to a numeric display interface with the F8. For example, the BCD to seven segment conversion and 3/8 digit decoding could be done in software. This approach (Fig. 8) uses two ports.

If four ports are available, the display could also be driven statically, with each port controlling two digits. This approach (Fig. 9) would require one BCD to 7-segment decoder/driver (and 7 resistors) for each digit.

The best design approach depends on the application and the number of F8 ports available.

ALTERNATE SCANNING APPROACH

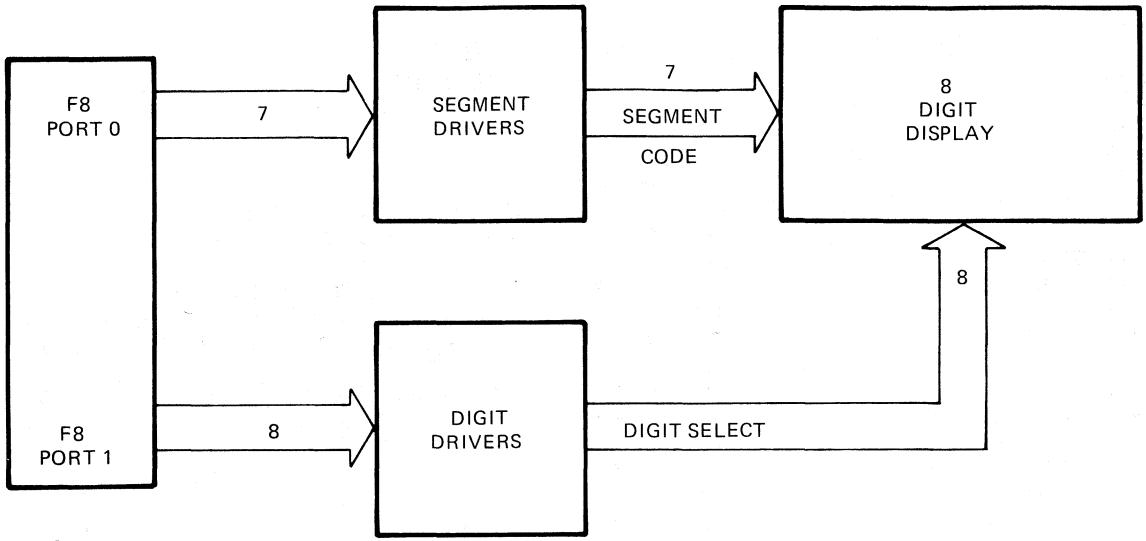


Figure 8

STATIC DISPLAY APPROACH

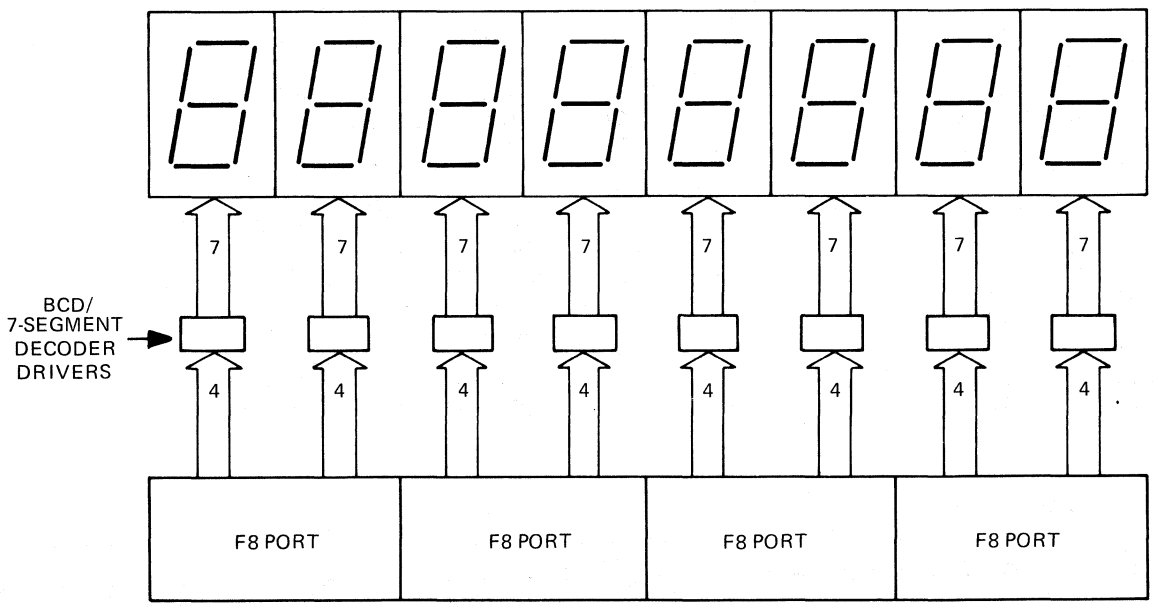


Figure 9

MOSTEK®

USING MOSTEK'S F8 IN A SCANNED KEYBOARD APPLICATION

Application Note

INTRODUCTION

Many microprocessor based systems require input from a keyboard of some type. The hardware required to encode a keyboard outside of the processor can be eliminated by using a keyboard scanning technique. With one F8 port, a 16 switch keyboard can be scanned (see fig. 1) using no external hardware. This is because of the bi-directional quality of the F8 ports.

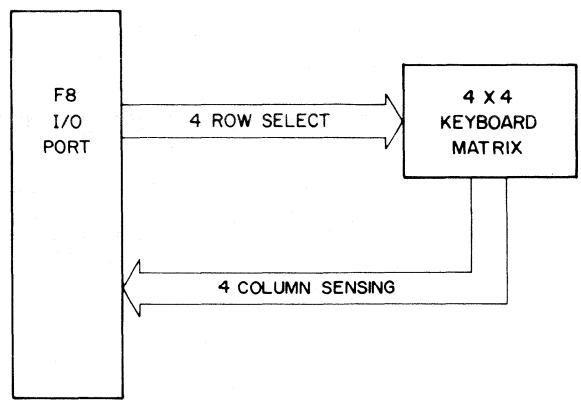
THEORY OF OPERATION

When scanning the keyboard, one of the four row select bits is turned on supplying a ground return for one row of switches. The column data is then read back into the processor via the four column bits. These four bits will indicate the condition of all four switches in the selected row. Each of the four rows is selected, one at a time, continously providing current status of all 16 switches.

"BOUNCE" is a problem encountered when using mechanical switches (see fig. 2). In order to prevent multiple detection of the switch closure, the bounce must be filtered out. A conventional solution to the bounce problem was to use an R–C filter and attempt to eliminate it electrically. However, when using the F8 scanning technique the switch bounce can be filtered in software by taking multiple samples of the switch to verify switch depression and release.

4x4 KEYBOARD MATRIX

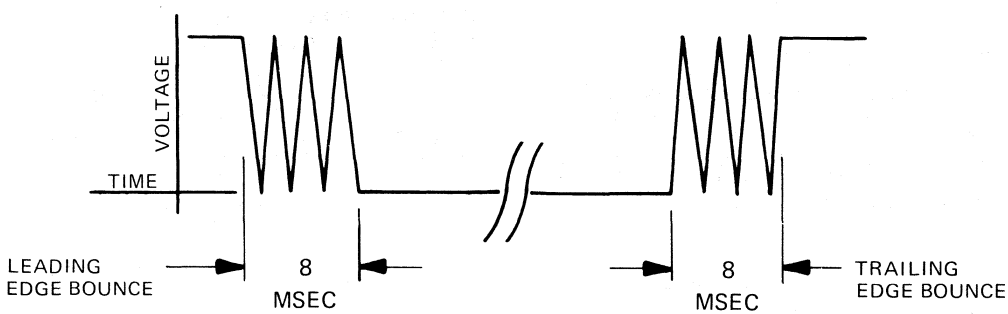
Figure 1



Since the software must usually scan all switches continuously, a register (or half) can be used to maintain the status of each switch.

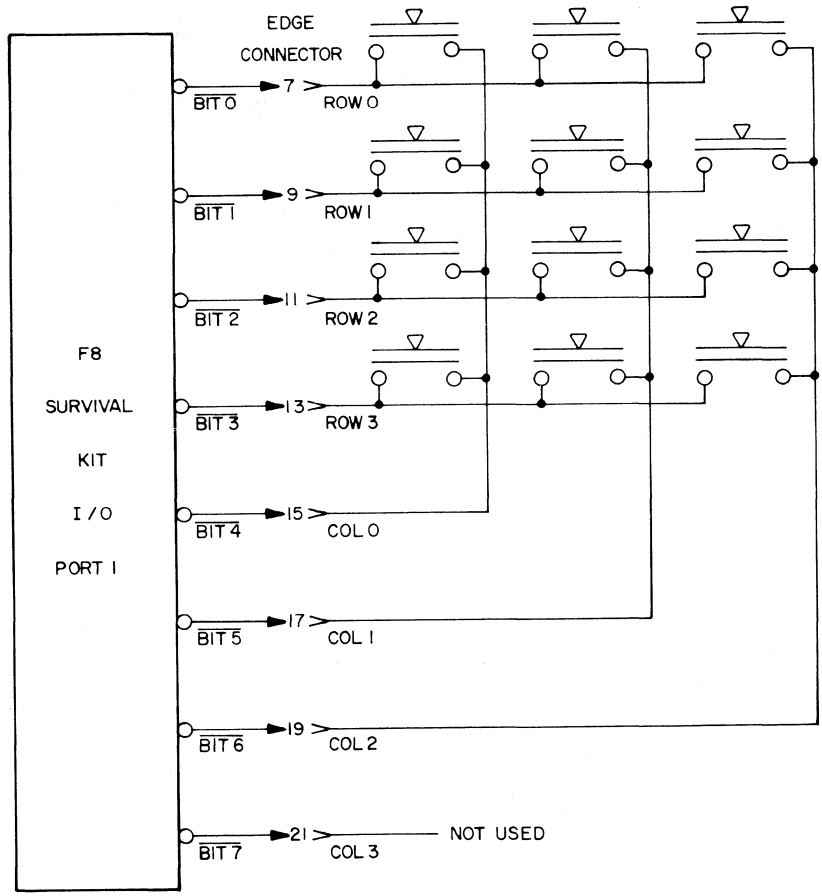
A common requirement for keyboards is "N–key rollover", meaning that if more than one switch is depressed at a time, all switch closures will be detected. This requirement can be met when using the scanning technique as described above. Since all switches are continuously scanned, the condition of each switch is always available to the processor.

SWITCH BOUNCE



4 x 3 KEY MATRIX

Figure 3



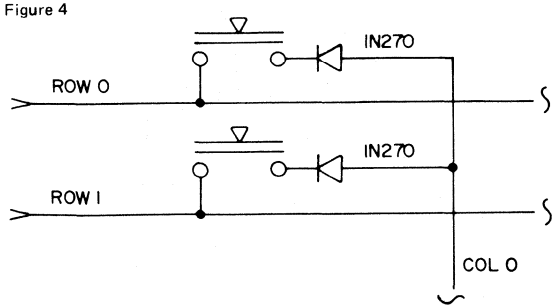
EXAMPLE HARDWARE DESIGN

The example in figure 3 shows a 4x3 matrix interfaced to an F8 port. This arrangement will provide N–key rollover input to the processor unless three keys are depressed simultaneously to form an L configuration. Then erroneous input could occur. If this presents a problem for a given application, one germanium diode (1N 270) should be added on the column pole of each switch (see fig. 4).

The operation of this keyboard (fig. 3) is simple. To sense the condition of row 0, a Hex '01' is written to port 1. Port 1 is then read back. The state of bits 4, 5 and 6 (COL 0, COL 1, COL 2) will be 1 if the respective switches in row 0 are closed and 0 if

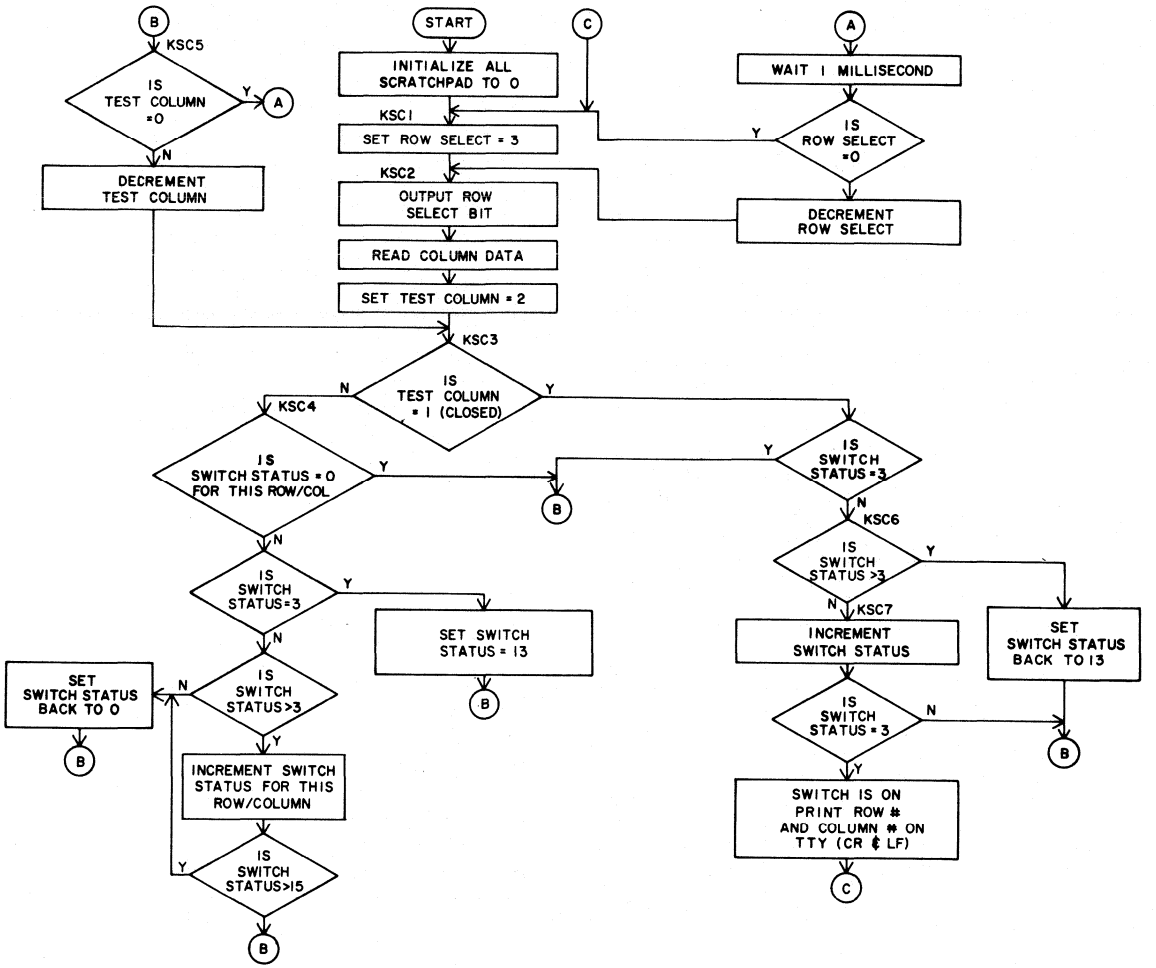
FOR SOME APPLICATIONS DIODES ARE NECESSARY

Figure 4



KEYBOARD SCAN ROUTINE (4 x 3 MATRIX)

Figure 5



open. (Note: The F8 I/O ports contain internal pull-ups). The other three rows are read similarly.

EXAMPLE SOFTWARE FOR THE 4x3 MATRIX KEYBOARD

An example program was written to run on the F8 Survival Kit to demonstrate software switch sensing and debounce.

One scratchpad register is used to maintain current status for each switch. When a switch is inactive it maintains a status of 0. In order for the switch to be processed, three consecutive scans must occur in which the switch is sensed to be closed.

When a switch is first sensed closed, its status is incremented to 1. In succeeding scans its status is either incremented (if sensed closed) or reset to 0 (if sensed open) until the status reaches 3, thus requiring three consecutive scans with the switch closed.

The switch is then processed, which in the example means the column number and row number are printed on the TTY (terminal).

A status of 3 is maintained by the switch until the first time it is sensed open. At that time its status is set to 13. Then three consecutive scans with the switch open are required to get the switch back to inactive status (0). This is accomplished by incrementing the status (if sensed open) or resetting the status to 13 (if sensed closed) until it reaches 15. The status is then reset to 0. As long as bounce occurs, however, the status will be reset to 13.

The flowchart (fig. 5) shows the logic described above. Note that at the end of each row scan there is a one millisecond delay which effects an interscan delay of 4 milliseconds for each switch. This means that the switch must be on 'solid' for 8 milliseconds before being processed and off 'solid' 8 milliseconds before becoming inactive again; so the switch will only be processed one time per depression. This debounce time sets the max keyboard entry rate for a given switch at 1 entry/24 milliseconds.

Figure 6 shows the scratchpad register assignments used by the example program.

For an instruction by instruction description of the example program see the listing (fig. 7).

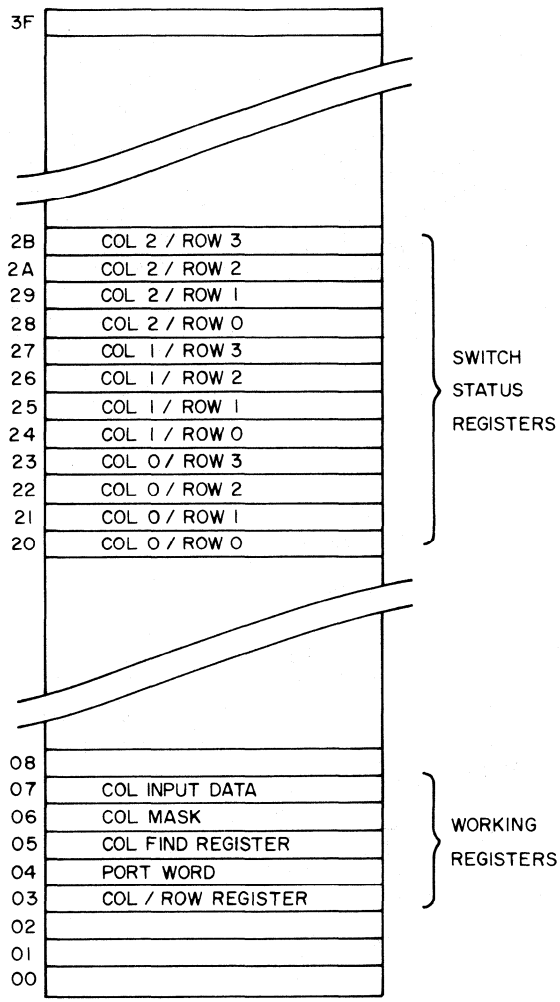
ALTERNATE DESIGN APPROACHES

When more than 16 switches are needed, an additional chip must be used. By adding a 4 to 16 decoder (see figure 8) to select 1 of 16 rows, up to 64 switches can be scanned.

Many off-the-shelf keyboards are available which have a 4x3 or 4x4 physical arrangement, but all switches have one common pole (on the P.C. Board). This type of keyboard can be scanned by using a 4 bit code to select one of up to 16 switches. The code is

SCRATCHPAD REGISTER ASSIGNMENTS

Figure 6



then decoded by a 4 to 16 decoder which supplies a ground return to the selected switch. The switch common line is then read to sense the condition of that switch (see figure 9).

If more ports can be assigned to the keyboard interface, other options may become advantageous. For example, with two ports 16 switches can be read without scanning. The basic requirements such as switch debounce and N-Key rollover will remain regardless of which option is taken. The best approach to a given design application will be determined by the system requirements and structure.

ASSEMBLY LISTING OF EXAMPLE PROGRAM

Figure 7

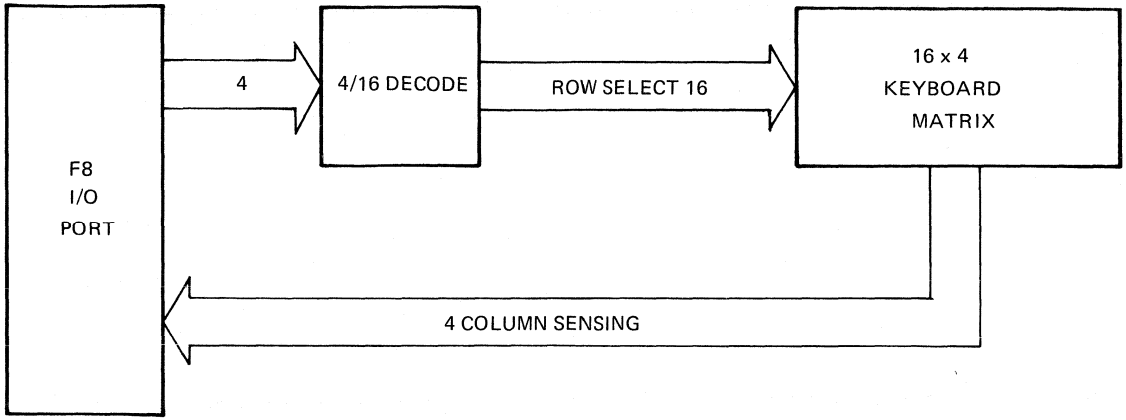
```
LINE # ADDRESS OBJECT CODE      SOURCE CODE                COMMENTS
0000                    *
0001                    *       (DETECT AND DEBOUNCE)
0002                    *
0003                    *
0004                            ORG   H'400'
0005 0400 20 3F         KSCN    LI    H'3F'      INITIALIZE ISAR
0006 0402 50                    LR    0,A        SAVE IN R0
0007 0403 40                    LR    A,0
0008 0404 0B                    LR    IS,A       NEXT ISAR
0009 0405 70                    CLR              CLEAR ACC
000A 0406 5C                    LR    S,A        CLEAR A SCRATCHPAD REG
000B 0407 30                    DS    0          DECREMENT ISAR POINTER
000C 0408 94 FA                 BNZ   *-5        LOOP TO CLEAR ALL SCRATCHPAD
000D 040A 73            KSC1    LIS   3
000E 040B 53                    LR    3,A        SET COL/ROW REG = 3
000F 040C 78                    LIS   8
0010 040D 54                    LR    4,A        SET PORT WORD = 8
0011 040E 44            KSC2    LR    A,4        LOAD ACC WITH PORT WORD
0012 040F B1                    OUTS  1          OUTPUT ROW SELECT
0013 0410 A1                    INS   1          READ COLUMN DATA
0014 0411 14                    SR    4          RIGHT JUSTIFY IT
0015 0412 57                    LR    7,A        SAVE COLUMN DATA
0016 0413 72                    LIS   2
0017 0414 55                    LR    5,A        SET TEST COLUMN = 2
0018 0415 74                    LIS   4
0019 0416 56                    LR    6,A        SET COL MASK = 4
001A 0417 45            KSC3    LR    A,5        LOAD COLUMN #
001B 0418 13                    SL    1
001C 0419 13                    SL    1          SHIFT 2 PLACES
001D 041A C3                    AS    3          ADD ROW #
001E 041B 22 20                 OI    H'20'
001F 041D 0B                    LR    IS,A       SET UP ISAR FOR THIS SWITCH
0020 041E 47                    LR    A,7        LOAD COLUMN READ DATA
0021 041F F6                    NS    6          MASK OUT ALL COLUMNS EXCEPT TEST
0022 0420 84 29                 BZ    KSC4       IF SWITCH NOT CLOSED , BRANCH
0023 0422 4C                    LR    A,S        GET SWITCH STATUS FOR TEST SWITC
0024 0423 25 03                 CI    H'03'
0025 0425 94 15                 BNZ   KSC6       IF STATUS NOT 3, JUMP
0026 0427 35            KSC5    DS    5          DECREMENT COLUMN #
0027 0428 46                    LR    A,6        LOAD COLUMN MASK REG
0028 0429 12                    SR    1          SHIFT MASK BIT
0029 042A 56                    LR    6,A
002A 042B 94 EB                 BNZ   KSC3       IF MORE COLUMNS, JUMP
002B 042D 20 64                 LI    H'64'
002C 042F 5B                    LR    11,A       SET UP 1 MSEC TIMER
002D 0430 3B                    DS    11
002E 0431 94 FE                 BNZ   *-1        WAIT 1 MSEC
002F 0433 33                    DS    3          DECREMENT ROW SELECT
0030 0434 44                    LR    A,4        LOAD PORT WORD
0031 0435 12                    SR    1          SHIFT
0032 0436 54                    LR    4,A        SAVE
0033 0437 84 D2                 BZ    KSC1       IF ROW SELECT WAS ZERO, BRANCH
0034 0439 90 D4                 BR    KSC2       LOOK AT NEXT ROW
0035 043B 82 05         KSC6    BC    KSC7
0036 043D 7D                    LIS   13
0037 043E 5C                    LR    S,A        SET STATUS BACK TO 13
0038 043F 90 E7                 BR    KSC5
0039 0441 1F            KSC7    INC              INCREMENT STATUS
```

```
003A 0442 5C                LR   S,A        SAVE SWITCH STATUS
003B 0443 25 03             CI   H'03'      IS STATUS NOW = 3?
003C 0445 94 E1             BNZ  KSC5       IF NOT = 3, BRANCH
003D 0447 29 04 65          JMP  PROC       JUMP TO PROCESS THIS KEY
003E 044A 4C          KSC4  LR   A,S        LOAD STATUS TO ACC
003F 044B 25 00             CI   0
0040 044D 84 D9             BZ   KSC5       IF STATUS = 0, BRANCH
0041 044F 25 03             CI   3          IS STATUS = 3?
0042 0451 94 06             BNZ  KSC8       NO
0043 0453 20 0D             LI   13
0044 0455 5C                LR   S,A        SET STATUS = 0
0045 0456 90 D0             BR   KSC5
0046 0458 92 04       KSC8  BNC  KSC9       IF STATUS > 3, BRANCH
0047 045A 70                CLR
0048 045B 90 F9             BR   *-6        SAVE STATUS & RETURN
0049 045D 1F          KSC9  INC             INC STATUS
004A 045E 5C                LR   S,A        SAVE IT
004B 045F 25 10             CI   H'10'      IS STATUS > 15 ?
004C 0461 84 F8             BZ   KSC8+2
004D 0463 90 C3             BR   KSC5       IF NOT, BRANCH
004E                  *
004F                  *
0050                  *                     PROCESS SWITCH ( PRINT ON TTY)
0051                  *
0052                  TTO   EQU  H'35D'
0053 0465 0A          PROC  LR   A,IS       USE ISAR FOR ROW/COL
0054 0466 21 03             NI   3          MASK FOR ROW
0055 0468 22 30             OI   H'30'      FORM ASCII FOR ROW#
0056 046A 50                LR   0,A        SAVE IN R0
0057 046B 0A                LR   A,IS
0058 046C 21 0C             NI   H'0C'      MASK FOR COL
0059 046E 12                SR   1
005A 046F 12                SR   1
005B 0470 22 30             OI   H'30'      FORM ASCII FOR COL#
005C 0472 51                LR   1,A        SAVE IN R1
005D 0473 20 FF             LI   H'FF'
005E 0475 0B                LR   IS,A
005F 0476 54                LR   4,A
0060 0477 34                DS   4
0061 0478 56                LR   6,A
0062 0479 71                LIS  H'1'
0063 047A B6                OUTS 6
0064 047B 0B                LR   IS,A       SET ISAR = 1
0065 047C 28 03 5D          PI   TTO        PRINT COL#
0066 047F 4E                LR   A,D        DECREMENT ISAR
0067 0480 28 03 5D          PI   TTO        PRINT ROW#
0068 0483 7D                LIS  H'D'       LOAD 'CR' CODE
0069 0484 5C                LR   S,A        PUT IN SCRATCHPAD
006A 0485 28 03 5D          PI   TTO        CR TO TTY
006B 0488 46                LR   A,6        LOAD TTO STATUS
006C 0489 18                COM
006D 048A 94 FD             BNZ  *-2        WAIT FOR CR/LF
006E 048C 29 04 0A          JMP  KSC1       BACK FOR NEXT KEY
006F                        END
00

KSC1 040A  KSC2 040E  KSC3 0417  KSC4 044A  KSC5 0427
KSC6 043B  KSC7 0441  KSC8 0458  KSC9 045D  KSCN 0400
PROC 0465  TTO  035D
```

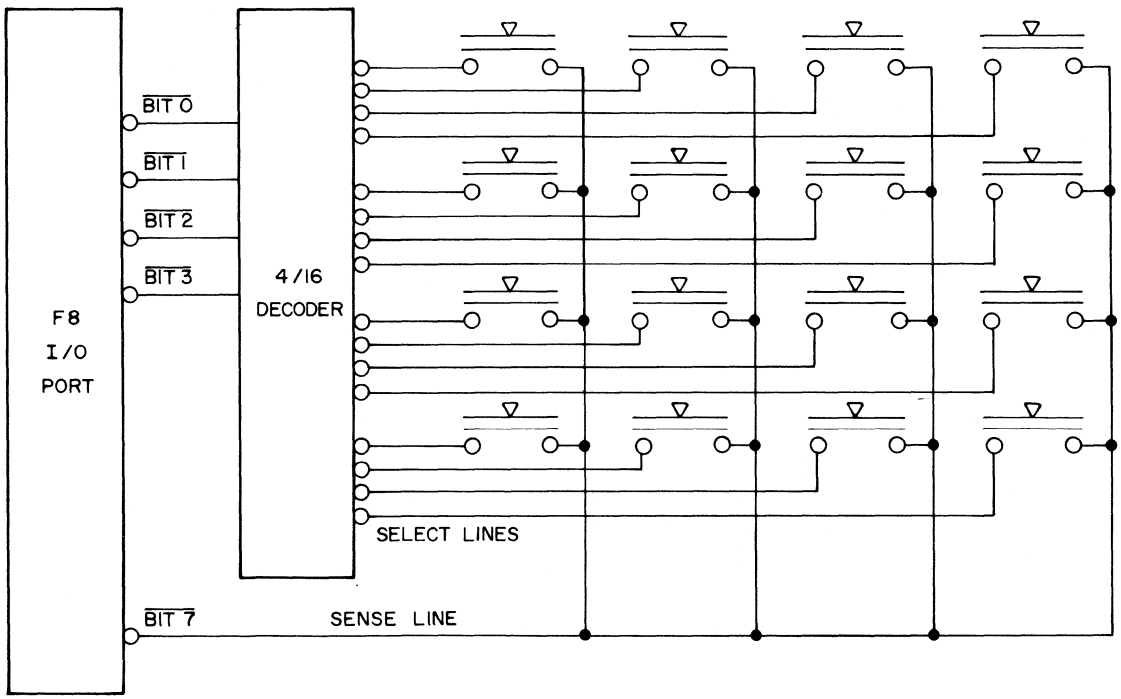
16 x 4 KEYBOARD

Figure 8



KEYBOARD WITH COMMON POLE

Figure 9



MOSTEK®

MICROCOMPUTER BECOMES SERIAL CONTROL UNIT

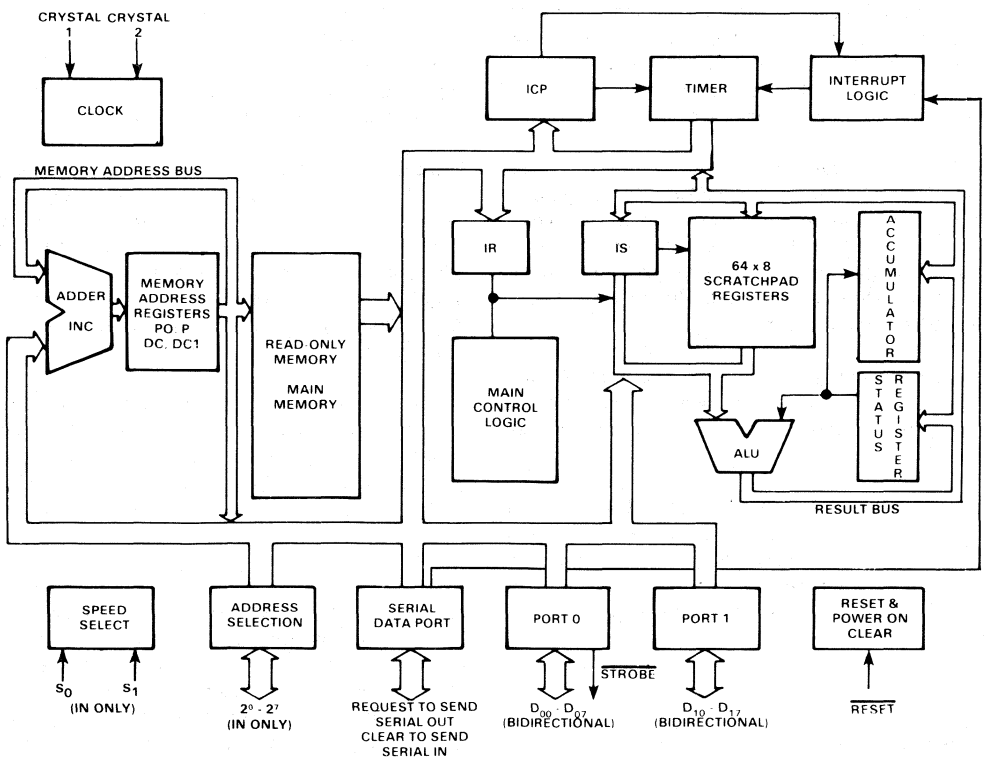
Application Note

Remote data acquisition and control is becoming the byword for many microcomputer-based systems, where the objective is to carry out operations at various locations under control of a central processor. Such applications are cropping up virtually everywhere, from factories, for, say, checking inventories, to gas stations, for monitoring gas pumps.

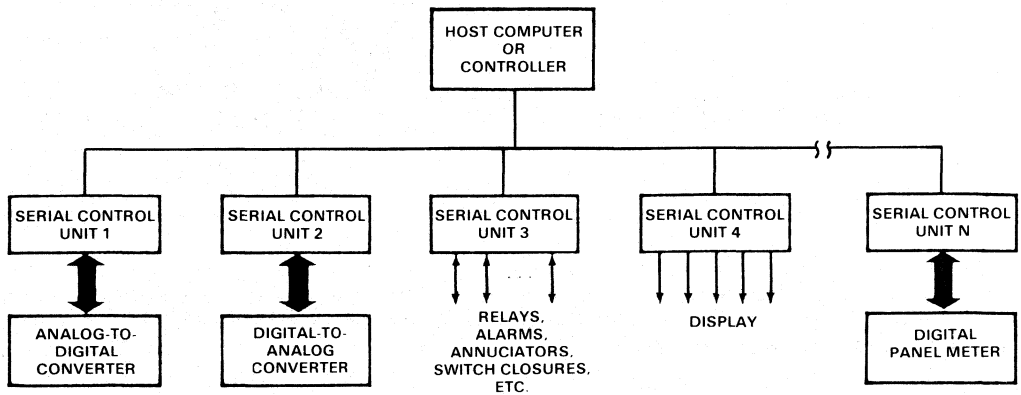
The trouble is that setting up the serially linked communications system, including the protocols, turns out to be no mean feat—and almost prohibitively expensive—since it often requires lots of random logic chips. Large-scale integration has changed the picture, adding programmability. But still it is difficult to attain from the available communications controllers the power, simplicity, and flexibility of a serial control unit, the SCU-1, developed by Mostek.

The SCU-1 takes much of the worry out of serial communication between processors. Up to 255 of them can be used as front-end controllers operating under direction from one central processing unit, and simple commands can initiate up to 19 different preprogrammed procedures each. Moreover, the network communication protocol for the SCU-1 derives from the attributes of multidrop communications systems developed in the minicomputer world; simple and reliable, it specifies asynchronous operation in a half-duplex mode at a rate of up to 1,200 bits per second.

The power of the SCU lies in the fact that it is a single-chip microcomputer (Figure 1). Housed in a 40-pin dual in-line package, it is able to change its mode of operation by interpreting commands received in a newly defined network communication protocol.



1. **Processor-peripheral.** Mostek's SCU-1 serial control unit is housed in a 40-pin DIP and implemented using ion-implanted n-channel silicon-gate technology, which yields a typical power requirement of 275 mW. All eight port 0 and all eight port 1 I/O lines are bidirectional.



2. **Party line.** Up to 255 serial control units can be placed in a one-data-link system under control of a host processor. A positive-true transistor-transistor-logic-convention is observed for selectable address strapping using eight dedicated chip pins.

The preprogrammed tasks include single-bit input and output, byte input and output, monitor or control input for selected patterns, and handshaking with analog-to-digital converters and digital panel meters. This functional flexibility means that the part can actually be used for both monitoring and control.

Even more useful to the cost-minded system designer is the fact that multiple units can be hooked up on a single half-duplex communication channel (Figure 2). Connected thus, all the units share the same "party line" communications link and are controlled by a central computer or controller. Orderly system operation is maintained by a user-defined polling sequence, as the SCUs cannot initiate a transmission—they can only respond after being polled.

Within a SCU is a communications controller, a task monitor, a command library, and an I/O interface (Figure 3). With the communications controller, a complete communications line can be set up for sending and receiving messages, checking errors, and synchronizing the unit. It also interfaces with the task monitor to allow specified tasks to be executed and reported on.

The task monitor interprets received commands and controls their execution. It also compiles results from I/O operations and passes them back through the communications controller.

The preprogrammed functions are stored within the command library. These functions are grouped into two categories: supervisory and timing (supervisory/timer) and memory and I/O (memory/input-output) commands. They give the user a great deal of flexibility in revising the software dynamically.

APPLICATIONS

The SCU has 16 I/O lines that can be addressed individually or together, depending on configuration. They are used to interface products such as analog-to-digital and digital-to-analog converters, 3½-digit panel meters, relays, and switches.

The chip generates and receives asynchronous serial data composed of 1 start bit, 8 data bits, 1 even parity bit, and 1 stop bit. Therefore, communications can be initiated by ASCII-compatible devices, from cathode-ray-tube terminals to mainframe computers.

The SCU is designed to work in locations far from the central controller, with the distance a function of the communications link, not the SCU. A minimum link configuration requires a half-duplex serial channel with a signaling capacity of 300 bits per second, but the 5-volt chip has pins for selecting 300 or 1,200 b/s.

The SCU transmits and receives a TTL-compatible serial asynchronous bit stream. No modulation or demodulation capability is provided. Handshaking and control signals allow the SCU to be interfaced with single-ended and differential line drivers and receivers, TTL-compatible radio-frequency modems, or fiber-optic transceivers. Since the data transmission is asynchronous, there are no critical timing or signaling parameters.

As mentioned earlier, up to 255 serial control units may be on one data link. Control is provided by a selectable address on the SCU using the appropriate pins. The 256th address, FF_{16}, is not allowed, since it is used internally by the SCU itself.

NEW PROTOCOL

Mostek has defined a new data-communications protocol for the SCU. This protocol provides easy access to the units, as well as good data throughput and data integrity. As with all protocols, it was designed to be easy to use and implement, flexible, and expandable and to have a low data-bit overhead. In addition, it had to be error-resistant and computer- or controller-independent, operate in a factory environment, and provide as much intelligence as

possible at the remote site.

These considerations led to the choice of a character-oriented protocol, which means that even though messages are sent in a bit-serial format, they are reconstructed and processed in 8-bit characters. In this protocol, five characters constitute a message. The characters are address, command, data address, data, and LRC (the error-check character, literally, "longitudinal redundancy check"). Before they can be understood, the message structure itself must be analyzed in some detail.

Messages to and from the serial control unit are sent in an asynchronous bit-serial format identical to ASCII transmission. For each character, first a start bit, followed by 8 data bits (least significant bit first) is sent, then an even parity bit and a stop bit. A total of five characters is transmitted for each message, yielding a total of 55 bits per message. A communication sequence consists of a message sent to an SCU and a corresponding acknowledgment.

Within the bit stream are two parity checksums. The first checks for the integrity of the previous 8-bit data word. the second, the LRC, checks the integrity of the four data words that make up the actual message. By using these two checks, any odd number of bit errors, as well as 2 bit errors, can be detected.

The individual characters specified by the protocol can be grouped to provide a normal or a special short message sequence. The latter is made up of only the address and command elements and is used during a fast polling operation. It trades some error-detection capability for an increased total-message throughput.

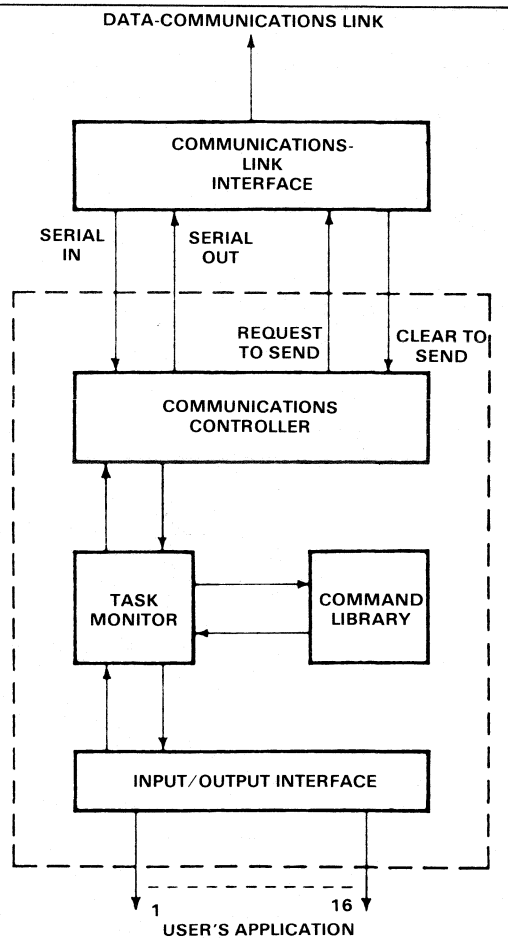
ADDRESS AND COMMAND

Eight bits are needed to represent the address of any serial control unit in a data link. These range in base 10 notation from 0 to 254. All inputs use positive-true logic and are coded in binary format. Therefore, if an SCU is to be defined as unit 100, for example, address-select pins 2^2, 2^5, and 2^6 should be signaled.

A bit-oriented command structure has been specified to ensure that the system is flexible and expandable. In this approach, 128 commands are reserved for use by the SCU and 128 commands left undefined for the user.

Even though the undefined commands cannot be executed by the SCU, the user can integrate other I/O controllers, in addition to the SCUs, into a single network. Bit C7—the most significant bit in the command word—controls such units. If it is set, one of Mostek's reserved commands will be accessed; if it is cleared, the SCU will ignore the command and send a loop command.

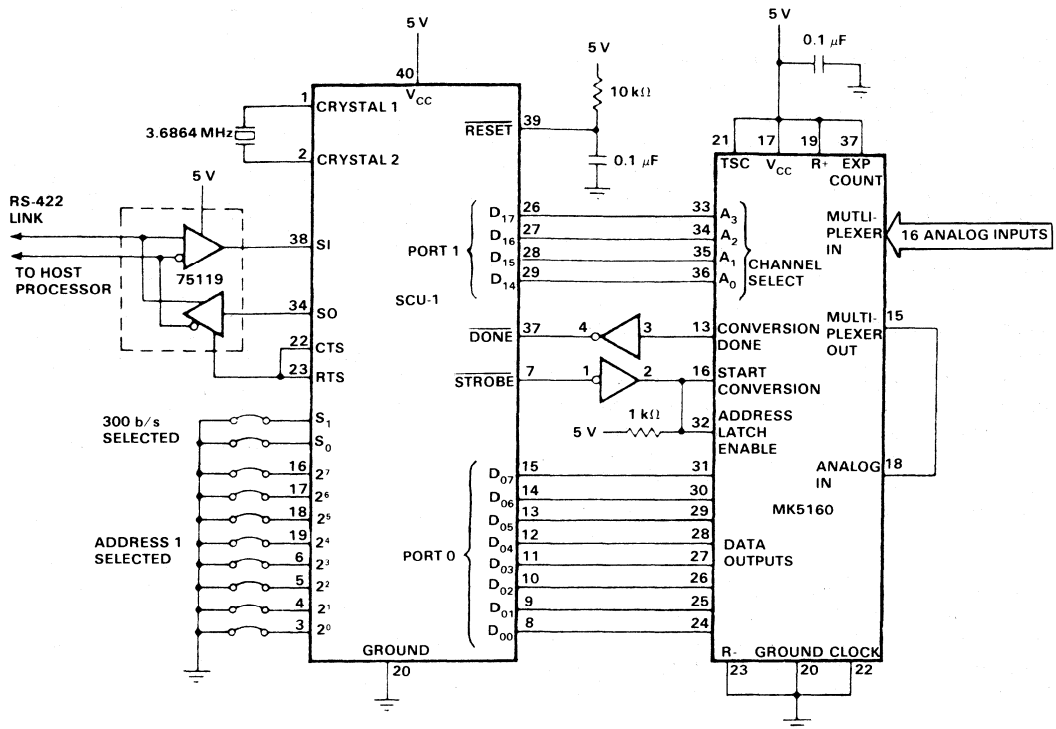
The command structure is subdivided further. Bit C6 differentiates between the two different kinds of tasks. If it is set, the supervisory and timer commands are accessed; if it is cleared, the memory and input/output commands are accessed.



3. **Flexible.** The SCU-1's architecture is designed for flexibility—up to 19 commonly used supervisory or memory-oriented functions may be accessed by a single command from a remote host processor. They permit system software to be reconfigured dynamically.

The six bits C5 through C0 are used within the Mostek command word to select one of 64 possible tasks in the groups specified by bit C6. The command assignment within the SCU is: 00_{16} - $7F_{16}$ (128 user-defined commands); 80_{16} - BF_{16} (64 memory and I/O commands) and $C0_{16}$ - FF_{16} (64 supervisory and timer commands). Any command that is not defined within the SCU but is nevertheless received by it will cause a loop supervisory command to be issued in response.

The 8-bit word of the data address serves two purposes. It is used to specify either the address of a port or memory or up to 8 bits of data. The actual determination is defined within the command word. The 8-bit word of the data character forms a byte and can be information from either a memory



4. **Conversions.** The SCU-1 is ideal for controlling analog-to-digital or digital-to-analog converters. In this case, an 8-bit a-d chip with 16 analog inputs sends its data to a host processor over an RS-422 transmission line, which provides good noise immunity and drive capability.

or an I/O port.

The final element in the protocol is the longitudinal redundancy check, or horizontal error-detection character. It is created by generating a parity check on each of the four previous elements of the protocol. Combined with the vertical parity check provided in each element, the LRC provides a high margin of error detection and a virtually error-free message interchange between the SCU and the host processor.

SYNCHRONIZING MESSAGES

The SCU uses a special procedure to synchronize network messages. Bit and word synchronization, on the other hand, are no problem, since they are provided by the asynchronous format. Network-message synchronization is needed when an SCU is initialized or whenever an individual SCU or the network is restarted.

There are three responses an SCU can generate to a host message. They are: address, command, data address, data, LRC; address, loop, data address, data LRC; and no response.

The first response is the normal reply to a host message. The address and command characters are identical to those generated by the host, and the data-address and data characters are modified to reflect the response requested in the command field. The LRC is generated to ensure that correct parity is maintained for the four previous characters.

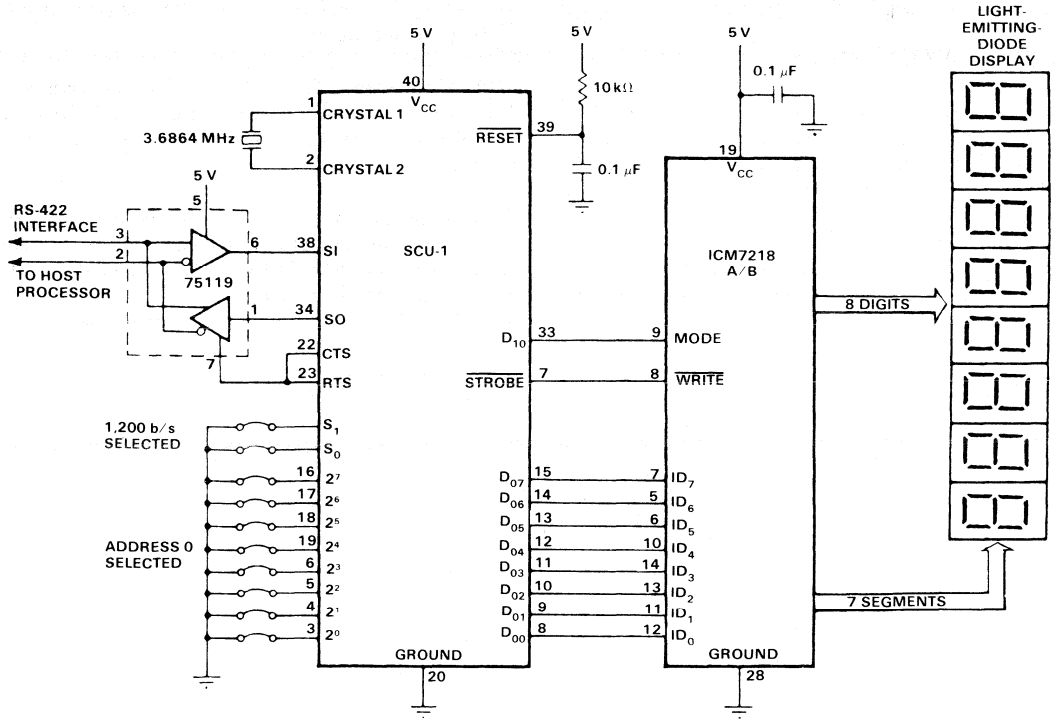
The second response is generated if the host processor issues a loop command or a memory address.

The third, literally no response, occurs if a message is sent to the SCU and an error is detected. The unit simply remains in synchronization and the message must be retransmitted.

SHORT-POLL RESPONSES

As mentioned earlier, the short-poll format is especially useful in speeding up message throughput, as, for example, when the SCUs are performing monitoring functions only. Three SCU responses are possible in this mode: address, poll; address command, data address, data, LRC; or no response.

If a short poll is issued and there has been no change since the previous poll, the address and poll command is sent back to the host. If an activity has occurred since the previous poll, the SCU will issue the second response to the host. This reply indicates the SCU's designated task and the new condition that caused the reply to be generated. The unit will continue to generate this response until it is reset by a host command.



5. **Display.** Mostek's SCU-1 peripheral microcomputer can both provide data to and control the operation of a light-emitting-diode display driver. Here, it is connected to the host processor by an RS-422 link. Only 5 volts is needed for the complete system.

If a parity error has been detected by an SCU, no response will be generated in answer to the short poll. At the same time, all regular command formats are fully functional and will be acknowledged when the short-poll mode is activated.

IMPLEMENTATION

The logic flow within the SCU to implement the protocol is an interrupt-driven routine. In the receiving mode, the message is always checked for synchronization, errors, unit address, and message completeness. If all is well, a new task is then placed in the executive routine for further processing.

In the transmitting mode, a message is assembled in the output buffers and then sent in a bit-serial format to the host processor. Here, the executive routine acts on command placed in its buffer by the protocol handler. It also manages the task library and reports the results of requested operations into an output buffer for transmission by the protocol routine.

The executive routine continually scans for a new task. When an error-free message is received and a new task requested, the executive searches for the task in the task library. The task is then executed and if required an appropriate response is made.

When this task is completed, the routine tests to see whether a previous task was suspended. If so, it restores that task. When all tasks are completed, the executive returns and once again scans for additional tasks.

One special feature implemented in the software is request-to-send (RTS) and clear-to-send (CTS) command. These signals permit the SCU to be used over a variety of communication media by synchronizing the data with the channel direction. The two can be tied together or be used in conjunction with an external control; data synchronization can thus be maintained on channels with radio links and modems that have slow turnaraound times.

SYNCHRONIZING THE DATA

With this feaure, when a message is ready to be sent, the RTS signal is activated. Data will then be transmitted only when the CTS signal becomes active.

Note that only the SCU issues an RTS. After a interval selected by the user, a CTS command is received and data appears on the serial-out pin. If a CTS is not received within two seconds, the RTS becomes inactive and the output-message request is cancelled.

The SCU can be made to work with virtually any analog-to-digital converter. For example, it can control a 16-channel

multiplexer while accepting the value from a 12-bit a-d unit.

REMOTE DATA ACQUISITION

In one application (Figure 4), a Mostek 50816N 8-bit 16-channel single-ended a-d converter is interfaced with a serial control unit. Port 0 (D_{00} - D_{07}) of the SCU is used to pass data, and D_{14} - D_{17} of port 1 select the input multiplexer channel.

A conversion is begun when a request is commanded by the strobe line. Upon completion, the conversion-done flag is sent to the SCU and the digitized voltage is sent via the unit to the host processor.

In this application an RS-422 communications link is used. This channel provides high noise immunity and good drive capability and allows the whole system to use a single 5 V supply.

The SCU can also be used to interface with display circuits such as the Intersil 7218 series eight-digit light-emitting-diode driver (Figure 5). This chip, which includes digit and segment drivers, all multiplex scan circuitry, and an on-board 8-by-8-bit static memory, is interfaced with the SCU.

MOSTEK®

VFC PROVIDES A/D CONVERSION FOR SINGLE CHIP MICROCOMPUTERS

Application Note

The introduction of single chip microcomputers has substantially lowered the cost of digital computer processing, but converting analog signals to a usable digital format still presents a problem with expensive solutions. Low cost methods of analog to digital (A/D) conversion are necessary for single chip microcomputers to be utilized in cost sensitive applications where measurement of real-world parameters is necessary.

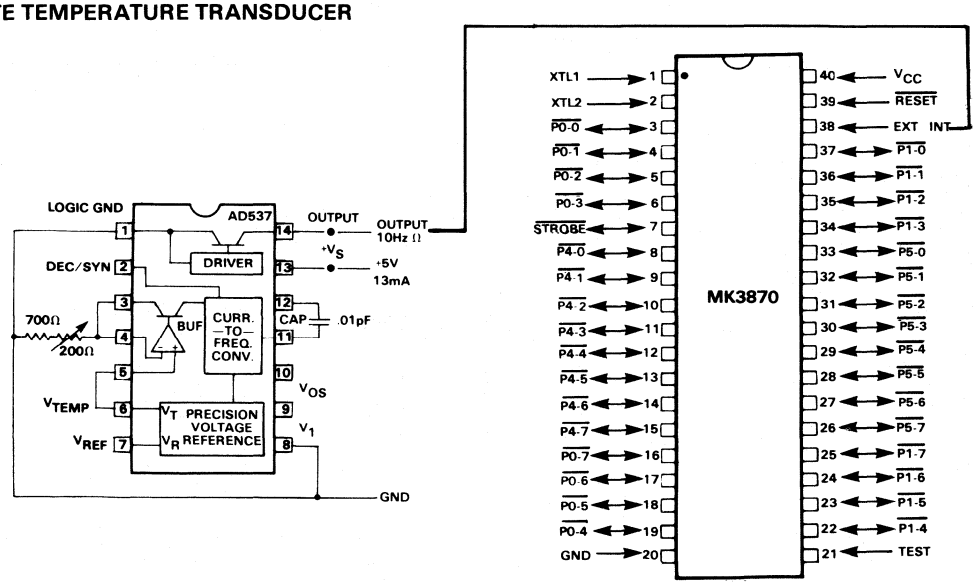
By allowing the microcomputer software to perform much of the conversion, cost and external parts count may be kept to a minimum. The Mostek MK3870 has the advantage of an on-board programmable timer which can be used to time intervals, measure pulse widths or count events. Several methods of A/D conversion exist which make use of the timer, circumventing the need for expensive A/D converter chips.

One easy way to implement such a system involves digitizing the analog voltage via a Voltage-to-Frequency Converter (VFC). The pulses from the VFC may be used to trigger external interrupts while the timer of the 3870 operates in the interval timer mode. By counting the external interrupts that occur during a predetermined time period, the frequency (and hence the analog voltage) may easily be determined.

Since the analog parameter most often measured is temperature, several methods of temperature measurement with the Mostek 3870 microcomputer will be examined, using a variety of transducers and a low cost integrated circuit VFC.

The Analog Devices AD537 VFC may itself be configured to generate an accurate square wave output frequency which is directly proportional to absolute temperature (in degrees Kelvin). This can be accomplished without the use of any other external temperature sensing device, since this function is built into the VFC. Additional advantages include operation from the 3870 microcomputer power supply (+5 Volts), one point calibration (since frequency extrapolates to zero at absolute zero) and direct interfacing to the 3870 EXT INT pin (which provides an internal pullup resistor). Figure 1 shows this circuit. Only two external components are required by the VFC: one resistor and one capacitor. The resistor should be a low temperature coefficient metal-film type. The capacitor should be chosen for low temperature coefficient and low dielectric absorption to provide high linearity. Polystyrene capacitors are recommended for operation up to +85°C.

ABSOLUTE TEMPERATURE TRANSDUCER
Figure 1



The V_{temp} output of the AD537 drives the high impedance buffer amplifier input directly. This output is scaled at 1 millivolt per degree Kelvin, and has a maximum error of 5° at room temperature (298°K). Also, since the output frequency is equal to V/(10RC), any error in the values of the resistor and capacitor will add to the scaling error of the VFC itself (5%). Therefore, a one-point calibration can be accomplished by trimming only the timing resistor. A ±10% adjustment range as shown should be adequate.

With a 1K Ohm timing resistance and a .01 microFarad capacitor, the VFC is scaled for 0 to 100KHz output with 0 to 10 Volts input. Since the temperature range being measured is likely to be 0° to 100°C, the input to the VFC buffer amplifier is only 273 to 373 millivolts (output 2730 to 3730 Hz). The linearity of the VFC over such a small portion (1%) of its dynamic range is quite good, and overall accuracy of this circuit is on the order of a few tenths of a degree.

SOFTWARE

Since the output frequency equals 10 Hz/°K, the microcomputer need only count the cycles in a 100 millisecond duration to directly compute the temperature in Kelvin. The 3870 timer is programmed as follows. Assume an external crystal frequency of 2.5 MHz. The internal clock frequency is one half the external freqency, or 1.25 MHz and the clock period is .8 microseconds. The 3 most significant bits of the Interrupt Control Port (ICP, Port 6) control the Timer Prescaler which may divide the clock by any combination of 2, 5 or 20. With all 3 bits set, the prescaler will divide the clock by (2 x 5 x 20) = 200. Thus, once every (200 x .8 microseconds) = 160 microseconds the timer will decrement one binary count.

When the timer (Port 7) is loaded with an 8 bit number, the value is also stored in the modulo-N Register. The timer will count down to 1 with each clock pulse from the prescaler, and roll over to the module-N value and continue running. Upon each roll over of the timer, a timer Interrupt Request occurs, which, when serviced, transfers program execution to a subroutine at ROM address H'020'. By loading the timer with 125 (H'7D'), a timer Interrupt Request will occur every (125 x 160 microseconds) = 20 milliseconds. The timer Interrupt Subroutine may count 5 such interrupts for a total time interval of (5 x 20 milliseconds) = 100 milliseconds. The subroutine may then stop the timer, disable all interrupts, and signal that the time interval is complete.

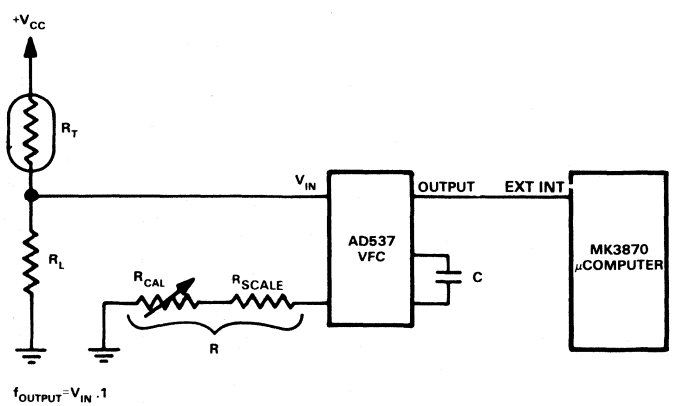
During the time interval, the VFC frequency may be counted by using the signal to trigger external interrupts. Upon each transition of the signal applied to the EXT INT pin from its inactive to its active level (determined by ICP, bit 2), an External Interrupt Request occurs, transferring program execution to a subroutine at ROM address H'OAO'.

Each time the External Interrupt Subroutine is called, the 4 digit BCD count held in two Scratchpad Registers is incremented. When the time interval is complete, all interrupts will be disabled so counting will cease and the BCD count will represent the Kelvin temperature. Centigrade temperature may be found by subtracting 273 from the Kelvin temperature.

Other Temperature Sensing Methods

For temperatures above 125°C, most solid state devices will not function, so other devices such as thermistors must be used. A thermistor may be linearized for a particular temperature range by use of a voltage divider (see reference 4). This is done by choosing the proper ratio of thermistor resistance (measure at 25°C) and the load resistance. The voltage across the load resistor is input to the VFC as shown in Figure 2. The VFC output is handled by the microcomputer as before.

Figure 2



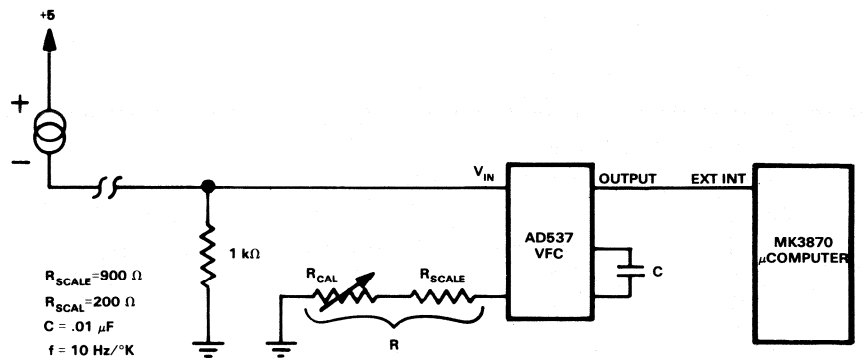
Thermistor resistance goes down as temperature goes up. The voltage across the load resistor, chosen to linearize the thermistor for a particular range, is input to the VFC to produce a frequency proportional to temperature.

For very high temperature applications, thermocouples must be used. Although non-linear, the output voltage of the thermocouple is predictable, and a temperature value for a given output voltage can be stored in a ROM lookup table.

For remote temperature sensing to 150°C, an Analog Devices AD590 temperature-to-current device may be used, as it has an output current directly proportional to absolute temperature (Figure 3). Line resistance in long wire runs do not degrade reading and line noise may be removed by filtering. Also less expensive VFC grades, with lower operating temperature ranges, may be used.

Any physical parameter that may be converted to a proportional linear voltage and scaled to vary within the operating limits of the VFC may be digitized by this method. This includes outputs from strain gauges, pressure transducers and linear position transducers.

Figure 3



The AD590 provides remote temperature sensing increasing a current to proportional to absolute temperature (1 μA/°K). In series with a 1KΩ K is input to the VFC.

Acknowledgement

This applications note was prepared through a cooperative effort of Tim Curran of Mostek and Doug Grant of Analog Devices Corp.

References

1) Grant, D., "Applications of the AD537 IC Voltage-to-Frequency Converter", publication number E478-10-8/78, Analog Devices Corp., 1978.

2) 3870 Programming Manual, publication number MK79630, Mostek Corp., 1979.

3) "Single Chip Microcomputer MK3870", data sheet, publication number MK79540, Mostek Corp., 1978.

4) Molee, C.S., Vitale, P., "Thermistors Make Good Thermometers", Electronic Design, April 12, 1978, pp. 90 - 92.

5) "Two-Terminal IC Temperature Transducer", data sheet, publication number C426a-12-9/78, Analog Devices Corp., 1978

```
ADDR   OBJECT FLAG ST # SOURCE STATEMENT     DATASET = DK1:ATOD  .3

                      0064    * MAIN PROGRAM BLOCK.
                      0065    *
>0100                 0066            ORG     H'100'
                      0067    * INITIALIZE CONSTANTS
>0005                 0068    RLCNT   EQU     5           TIMER ROLLOVER COUNT.
>007D                 0069    TCNT    EQU     125         TIMER MOD-N VALUE.
                      0070    *
                      0071    * CLEAR TEMPERATURE Ie, R6,R7.
'0100   70            0072    STRT    CLR
'0101   56            0073            LR      6,A
'0102   57            0074            LR      7,A
                      0075    *
                      0076    * LOAD TIMER ROLLOVER COUNT IN R0.
'0103   75            0077            LIS     RLCNT
'0104   50            0078            LR      0,A
                      0079    * LOAD TIMER
'0105   207D          0080            LI      TCNT
'0107   B7            0081            OUTS    7
                      0082    *
                      0083    * BEGIN TIMING WHEN EXT INT HIGH. FOR UNIFORM TIMING.
'0108   A6            0084            INS     6
'0109   84FE          0085            BZ      *-1
                      0086    *
                      0087    * LOAD ICP, BITS DEFINED AS FOLLOWS:
                      0088    * BIT 0 - ALLOW EXTERNAL INTERRUPTS
                      0089    * BIT 1 - ALLOW TIMER INTERRUPTS.
                      0090    * BIT 2 - EXT INT ACTIVE LEVEL.
                      0091    * BIT 3 - START/STOP TIMER (1/0).
                      0092    * BIT 4 - PULSE WIDTH/INTERVAL TIMER (1/0)
                      0093    * BIT 5 - DIVIDE CLOCK BY 2.
                      0094    * BIT 6 - DIVIDE CLOCK BY 5.
                      0095    * BIT 7 - DIVIDE CLOCK BY 20.
                      0096    *
'010B   20EF          0097            LI      B'11101111'   START TIMER.
'010D   B6            0098            OUTS    6
                      0099    * NOTE INTERRUPTS CAN BE MASKED BY ICP BITS 0 AND 1
                      0101    * OR BY CLEARING INTERRUPT CONTROL BIT (STATIC REG, BIT 4
                      0102    * ENABLE INTERRUPTS BY SETTING ICB.
'010E   1B            0103            EI
                      0104    *
                      0105    * MAY DELAY OR CONTINUE PROCESSING.
                      0106    * CONVERSION COMPLETE WHEN R9 = H'ff'.
'010F   49            0107    LOOP    LR      A,9
'0110   25FF          0108            CI      H'FF'
'0112   94FC          0109            BNZ     LOOP
                      0110    *
                      0111    * CONTINUE HERE WHEN CONVERSION DONE.
                      0112    *
                      0113            END

ERRORS=0000
```

```
ATOD3 A/D TEMP VIA FREQ;                          MOSTEK 3870/F8 CROSS ASSEMBLER PAGE 0001
 ADDR   OBJECT FLAG ST # SOURCE STATEMENT     DATASET = DK1:ATOD  .3

                      0001     * ANALOG TO DIGITAL TEMPERATURE VIA FREQ
                      0002     * TIM CURRAN  1/9/79
                      0003     *
                      0004              NAME    ATOD3
                      0006     * THIS 3870 ROUTINE COUNTS EXTERNAL INTERRUPTS FOR A
                      0007     * PREDEFINED PERIOD OF TIME TO DETERMINE FREQUENCY
                      0008     * AND HENCE THE TEMPERATURE.
                      0009     *
                      0010     * TIMER INTERRUPT ROUTINE
>0020                 0011              ORG     H'020'
                      0012     * SAVE STATUS AND ACCUMULATOR
'0020   1E            0013              LR      J,W
'0021   58            0014              LR      8,A
                      0015     *
'0022   30            0016              DS      0             R0 COUNTS TIME OUTS.
'0023   9407          0017              BNZ     RETT
                      0018     *
                      0019     * WHEN R0 IS ZERO, STOP TIMER AND MASK INTERRUPTS.
                      0020     *
'0025   20E4          0021              LI      B'11100100'   STOP TIMER,
'0027   B6            0022              OUTS    6             INTRPTS.
                      0023     *
                      0024     * SIGNAL CONVERSION COMPLETED BY SETTING R9 TO H'FF'.
                      0025     * THOUGH R9 (J) STORES STATUS (W), IT NEVER OTHERWISE
                      0026     * EQUALS H'FF'.
'0028   20FF          0027              LI      H'FF'
'002A   59            0028              LR      9,A
                      0029     *
'002B   1D            0030     RETT     LR      W,J           RESTORE STATUS AND A.
'002C   48            0031              LR      A,8
'002D   1B            0032              EI                    ALLOW INTERRUPTS.
'002E   1C            0033              POP                   RETURN.
                      0034     *
                      0035     *
```

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.

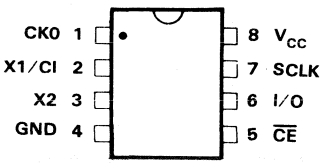
The real-time clock/calendar provides all timekeeping functions. It contains registers for seconds, minutes, hours, day, date, month, and year. The end of the month date is automatically adjusted for months with less than 31 days. The clock operates in either the 24 hour or 12 hour format with an AM/PM 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.

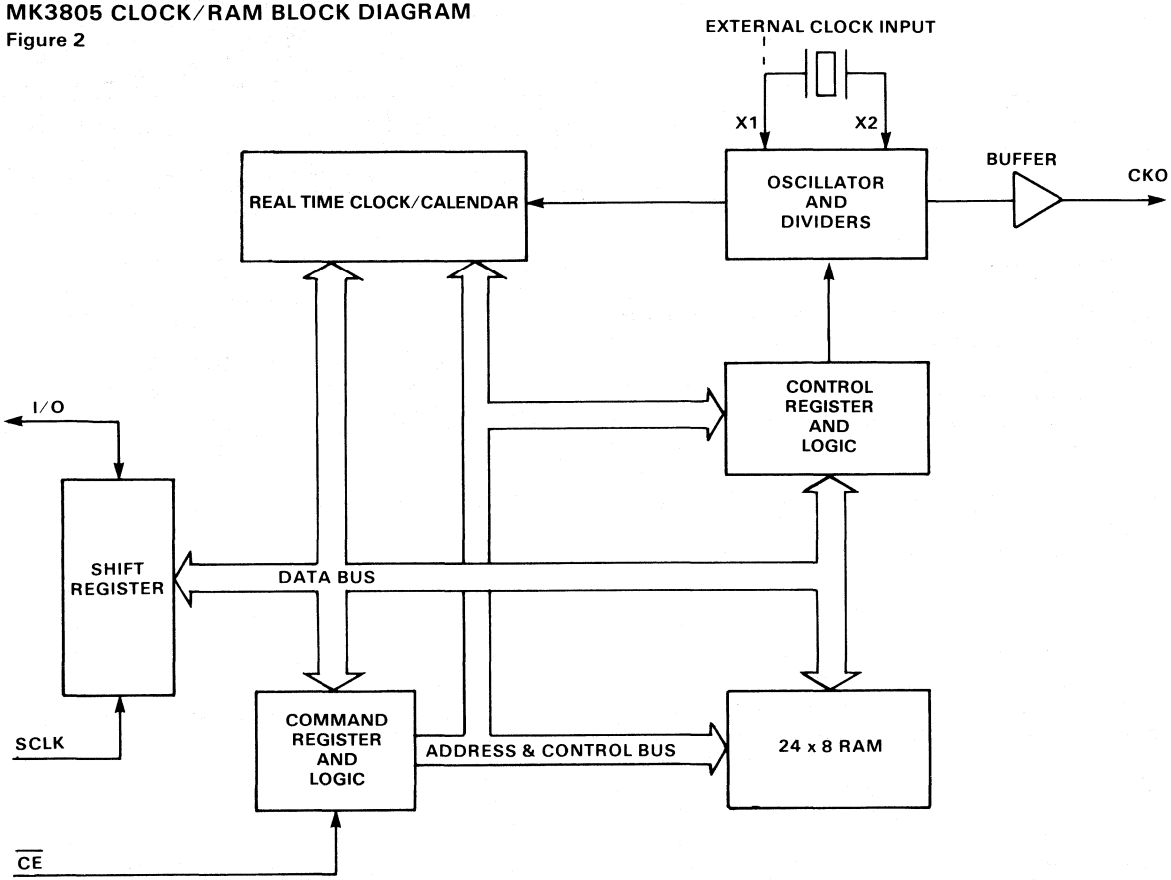
TECHNICAL DESCRIPTION

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 / $\overline{W}$

III. RAM COMMAND FORMAT:

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 / $\overline{W}$

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).

requires that $\overline{CE}$ be true and 200 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.

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 $\overline{CE}$ goes low (see Figure 4). After $\overline{CE}$ 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 $\overline{CE}$ 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 $\overline{CE}$ goes high, and the transfer must be reinitiated by the proper command when $\overline{CE}$ goes low again. The I/O pin will be in the high impedance state when $\overline{CE}$ 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 $\overline{STB}$ line associated with that port. The $\overline{STB}$ 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 $\overline{STB}$ 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.

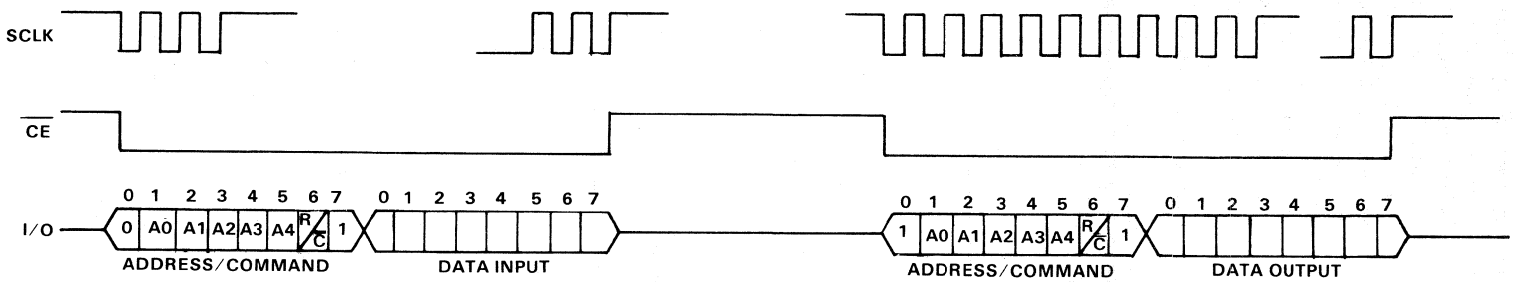
Before calling the subroutine, the necessary parameters must be set up in the proper registers. The ISAR is used as a pointer to where the data is to be read from or written to in the MK38P73 RAM area.

The scratchpad register 'CMD' must contain the command to be sent to the CLOCK/RAM. (See the description of the command given earlier.)

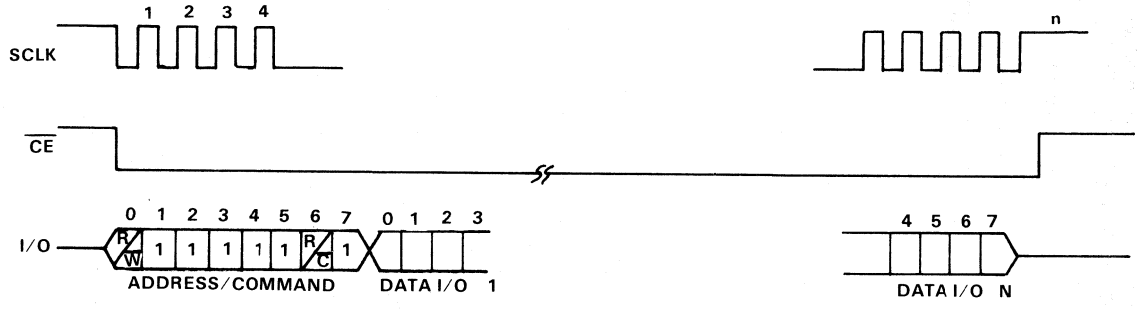
The bit pattern for enabling the CLOCK/RAM must be

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.

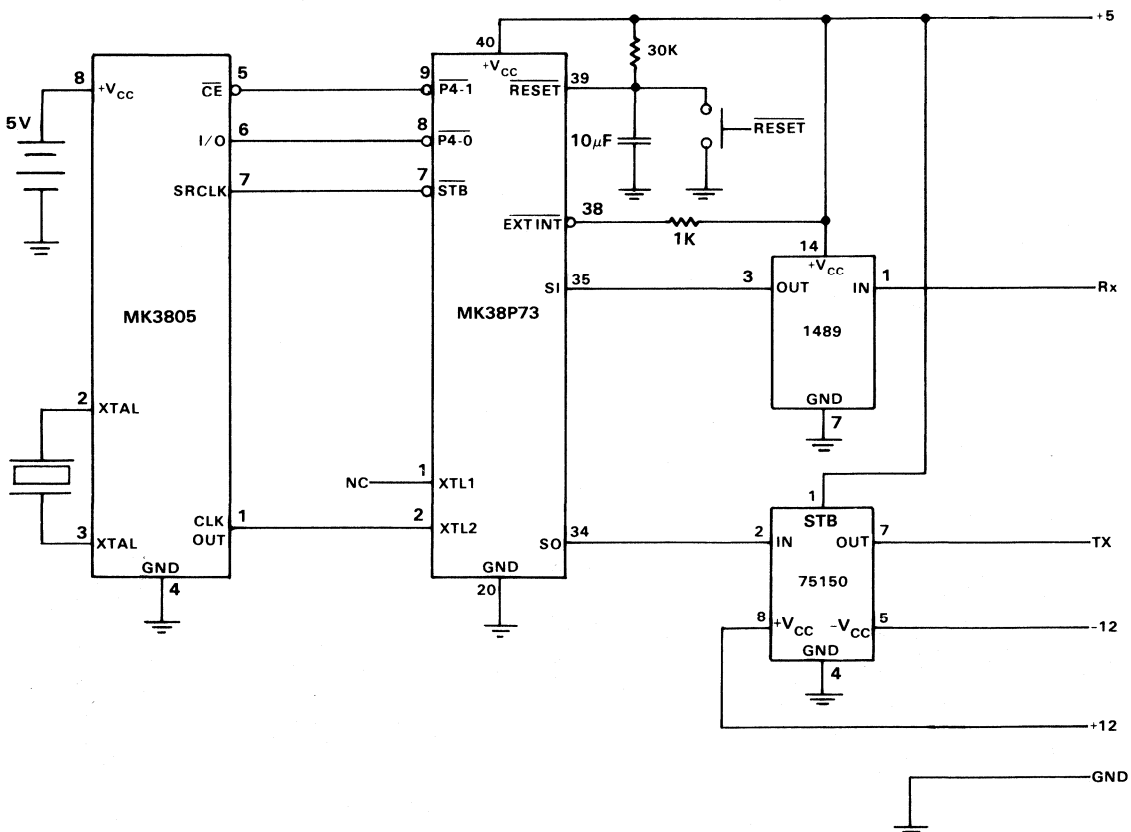
The main demonstration routine (listing 1) is quite basic. Its purpose is to print the features of the CLOCK/RAM on the CRT, then read the clock and display it's contents once every second. A reentry point is provided in order that the clock/calendar settings may be changed after power up. (See the flowchart 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.

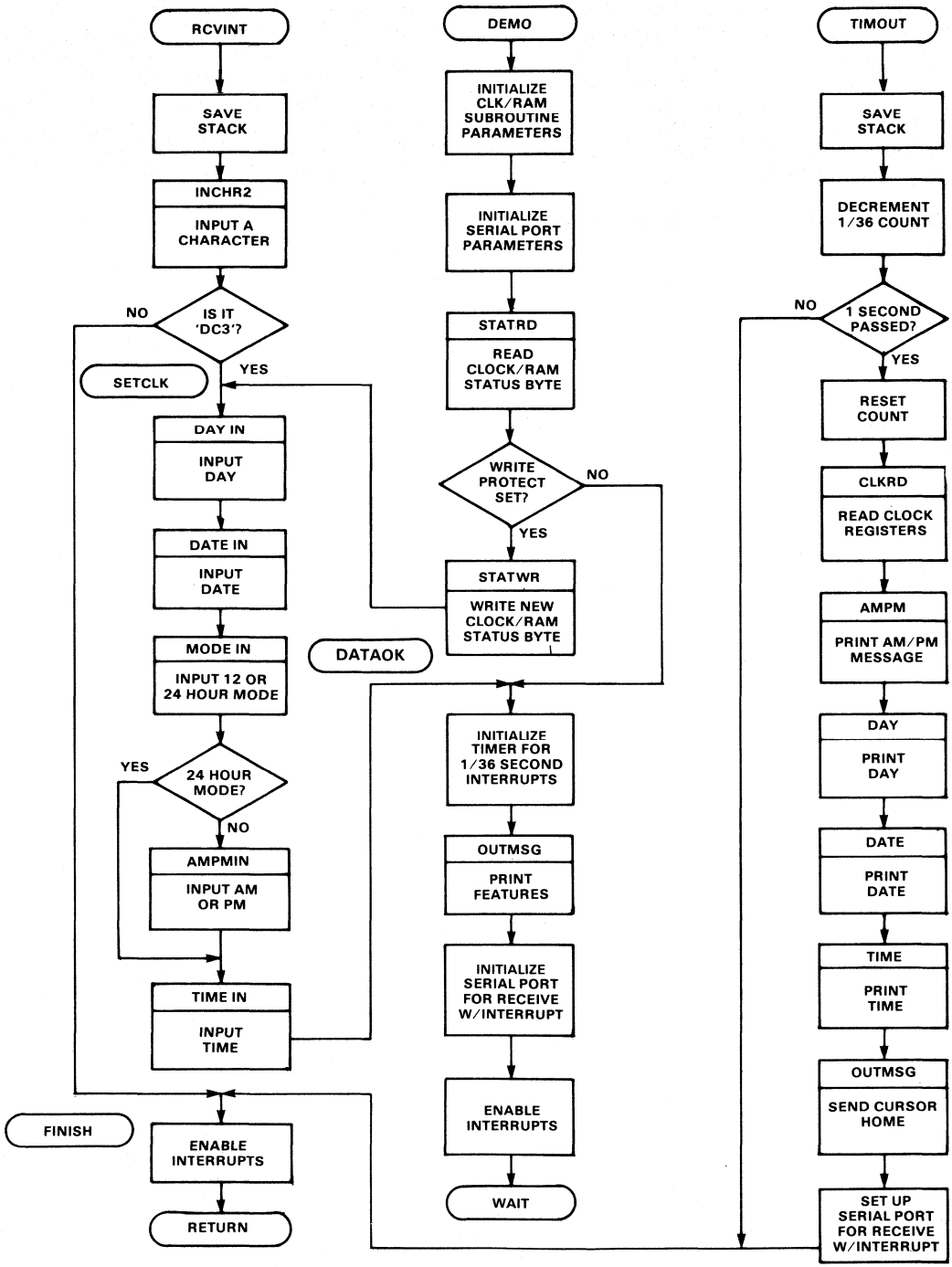
When a receiver interrupt occurs, the serial port contains a valid character from the keyboard. The service routine checks to see if it is a 'DC3' (control-S) character. If not, the routine returns to the wait for interrupt state. If it is, the routine goes to the clock set entry point of the main routine 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.

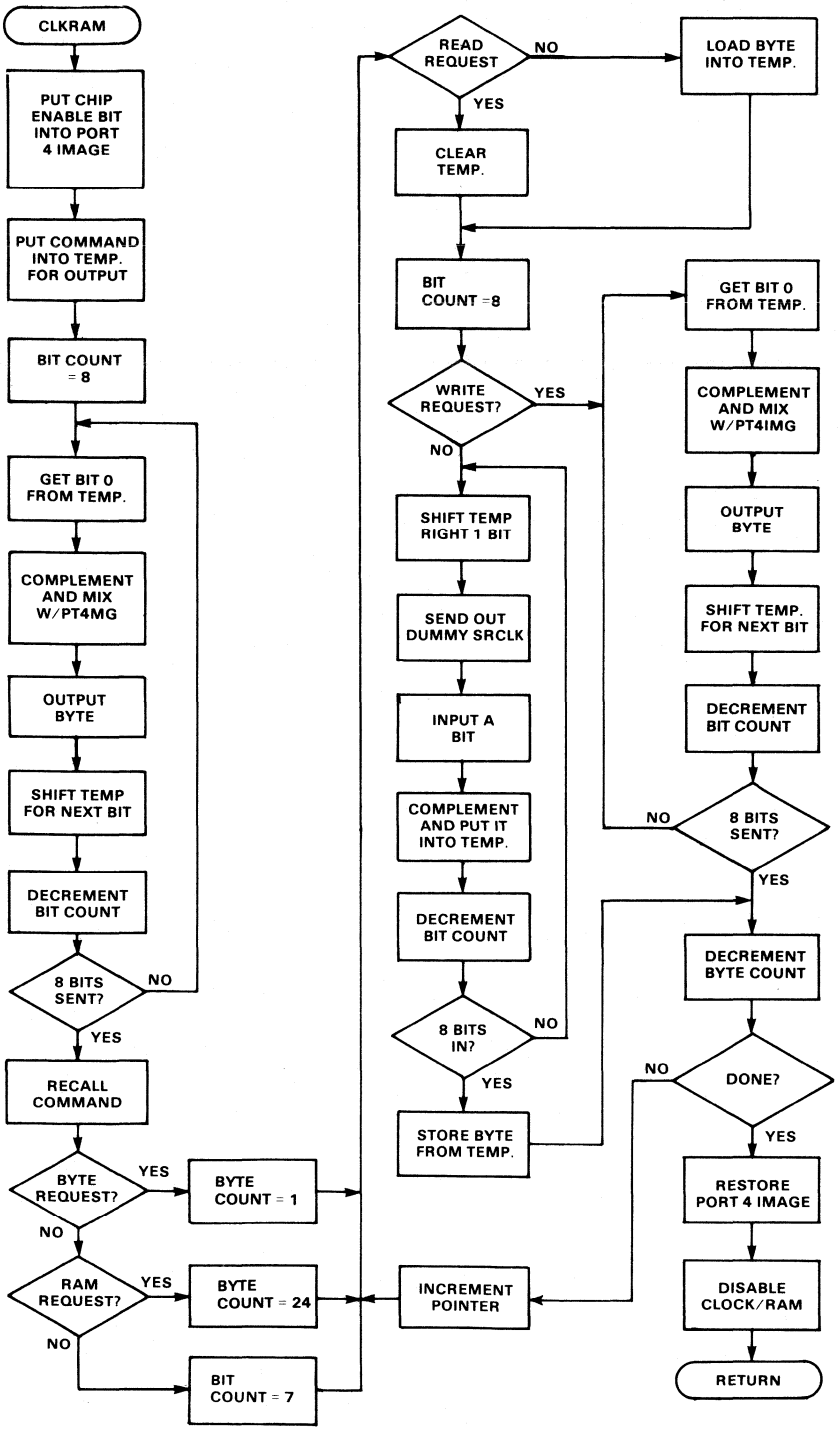
As seen in the flowchart (Figure 7), either 1, 7, or 24 bytes of data may be transferred between the microcomputer and the CLOCK/RAM. The command sent to the subroutine is exactly the command sent to the CLOCK/RAM, so there is no confusion as to the format of the command byte. When this routine is called, 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

```
:LOCK/RAM DEMONSTRATION MODULE    F8/3870 MACRO CROSS ASSM.  V2.2
.OC  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
```

```
LOC  OBJ.CODE     STMT-NR SOURCE-STMT PASS2 DEMO   DEMO   DEMO   ABS

                          *
=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
```

```
:LOCK/RAM DEMONSTRATION MODULE    F8/3870 MACRO CROSS ASSM.  V2.2
.OC  OBJ.CODE     STMT-NR SOURCE-STMT PASS2 DEMO   DEMO   DEMO   ABS

                          ******************
                          *                  *
                          * 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
                          *
000                  209          ORG  0000H
000 70               210          CLR               ;CLEAR ALL SCRATCH PAD
001 0B               211 INIT     LR   IS,A         ;PUT POINTER INTO ISAR
002 70               212          CLR               ;CLEAR THAT LOACTION
003 5C               213          LR   S,A          ;
004 0A               214          LR   A,IS         ;BUMP POINTER
005 1F               215          INC               ;BUMP POINTER
006 213F             216          NI   ISMASK       ;MASK TO 6 BITS
008 94F8             217          BNZ  INIT         ;GO IF NOT DONE
                          *
                          * SET UP CLOCK/RAM SUBROUTINE PARAMETERS.
                          *
00A 2002             221          LI   CRCHIP       ;SET CLK/RAM CHIP ENABLE
00C 51               222          LR   CHIPEN,A     ;
                          *
                          * INITALIZE SERIAL PORT PARAMETERS.
                          *
00D 200B             226          LI   BAUD         ;SET SERIAL BAUD RATE
00F BC               227          OUTS RXCTRL       ;
010 20FE             228          LI   PARITY       ;SET PARITY TO 'SPACE'
012 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.
                          *
013 2802AE           235          PI   STATRD       ;READ CLK/RAM STATUS
016 47               236          LR   A,CTRL       ;CHECK WRITE PROTECT BIT
017 F7               237          NS   CTRL         ;
018 8169             238          BP   DATAOK       ;BRANCH IF DATA GOOD
```

```
CLOCK/RAM DEMONSTRATION MODULE    F8/3870 MACRO CROSS ASSM.  V2.2
LOC  OBJ.CODE     STMT-NR SOURCE-STMT PASS2 DEMO   DEMO   DEMO   ABS

                          *
                          * CLOCK/RAM JUST POWERED UP, SO INITALIZE IT.
                          *
001A 2802B6               242         PI   STATWR      ;WRITE CLK/RAM STATUS
001D 29006C               243         JMP  SETCLK      ;SET CLOCK
```

```
LOC  OBJ.CODE     STMT-NR SOURCE-STMT PASS2 DEMO   DEMO   DEMO   ABS

                          ************************************
                          *                                  *
                          * 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.
                          *
 040 20B1            297          LI   RCVI         ;ENABLE RCV INTERUPT
 042 BD              298          OUTS RXSTAT
 043 1B              299 FINISH   EI                 ;ENABLE INTERUPTS
 044 0D              300          LR   P0,G         ;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 9408            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 2081        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
```

```
LOC  OBJ.CODE     STMT-NR SOURCE-STMT PASS2 DEMO   DEMO   DEMO   ABS

                          ************************************
                          *                                  *
                          * 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 SUBRCUTINE  *
                        *                       *
                        *************************
                        *
                        * 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
```

```
LOC  OBJ.CODE     STMT-NR SOURCE-STMT PASS2 DEMO   DEMO   DEMO   ABS

                          **********************************
                          *                                *
                          * 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 INPUT 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 DIGIT (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
```

```
LOC  OBJ.CODE     STMT-NR SOURCE-STMT PASS2 DEMO   DEMO   DEMO   ABS

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   DIGIT0       ;GET DIGIT (0-9)
0135 15          608         SL   4            ;STORE IT AT TENS
0136 5C          609         LR   S,A          ;
0137 28024D      610         PI   DIGIT0       ;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   MNTH0X       ;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 MNTH0X  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
```

```
LOC  OBJ.CODE     STMT-NR SOURCE-STMT PASS2 DEMO   DEMO   DEMO   ABS

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, SO 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
```

```
CLOCK/RAM DEMONSTRATION MODULE    F8/3870 MACRO CROSS ASSM.  V2.2
LOC  OBJ.CODE     STMT-NR SOURCE-STMT PASS2 DEMO   DEMO   DEMO   ABS

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
```

```
LOC  OBJ.CODE     STMT-NR SOURCE-STMT PASS2 DEMO   DEMO   DEMO   ABS

                          ***************************
                          *                         *
                          * 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
```

```
LOC  OBJ.CODE     STMT-NR SOURCE-STMT PASS2 DEMO   DEMO   DEMO   ABS

                          ************************
                          *                      *
                          * 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
```

```
                        **************************
                        *                        *
                        * 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 POINT.
                        *
                        * 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
                        *
```

```
LOC  OBJ.CODE     STMT-NR SOURCE-STMT PASS2 DEMO   DEMO   DEMO   ABS

023E 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
```

```
LOC  OBJ.CODE     STMT-NR SOURCE-STMT PASS2 DEMO   DEMO   DEMO   ABS

                          ********************************
                          *                              *
                          * 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
```

```
LOC  OBJ.CODE     STMT-NR SOURCE-STMT PASS2 DEMO   DEMO   DEMO   ABS

                          ****************************
                          *                          *
                          * PRINT MESSAGE SUBROUTINE *
                          *                          *
                          ****************************
                          *
                          * FUNCTION:
                          * THIS SUBROUTINE PRINTS THE MESSAGE WHOSE NUMBER IS
                          * 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.
```

```
LOC  OBJ.CODE     STMT-NR SOURCE-STMT PASS2 DEMO   DEMO   DEMO   ABS

                          ******************************
                          *                            *
                          * MESSAGE OUTPUT SUBROUTINE  *
                          *                            *
                          ******************************
                          *
                          * FUNCTION:
                          * THIS SUBROUTINE PRINTS THE MESSAGE STARTING AT
                          * 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 BD              1093         OUTS  RXSTAT
02A2 16              1094 LOOP3   LM                ;GET CHARACTER
02A3 2504            1095         CI    EOT         ;CHECK FOR END OF TEXT
02A5 84CD            1096         BZ    LOOP1       ;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 NEX
02AA 81FE            1100         BP    LOOP4       ;BRANCH IF NOT READY
02AC 90F5            1101         BR    LOOP3       ;NEXT CHARACTER
```

```
                        **********************************
                        *                                *
                        * CLOCK/RAM SUBRCUTINE 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 THE
                        * 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 208F          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
```

```
LOC  OBJ.CODE      STMT-NR SOURCE-STMT PASS2 DEMO   DEMO   DEMO   ABS

                           ******************
                           *                *
                           * 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 POW
     4F532044
     45534947
     4E20464F
```

```
LOC  OBJ.CODE     STMT-NR SOURCE-STMT PASS2 DEMO   DEMO   DEMO   ABS

     52204558
     5452454D
     454C5920
     4C4F5720
     504F5745
     52
0368 20434F4E          1189          DEFM ' CONSUMPTION.'
     53554D50
     54494F4E
     2E
0375 0D0A              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 0D0A              1193          DEFB CR,LF
03B8 2A203132          1194          DEFM '* 12/24 HOUR CLOCK/CALENDAR WITH AUTO'
     2F323420
     484F5552
     20434C4F
     434B2F43
     414C454E
     44415220
     57495448
     20415554
     4F
03DD 2041444A          1195          DEFM ' ADJUST FOR SHORT MONTHS AND LEAP'
     55535420
     464F5220
     53484F52
     54204D4F
     4E544853
     20414E44
     204C4541
     50
03FE 20594541          1196          DEFM ' YEARS.'
     52532E
0405 0D0A              1197          DEFB CR,LF
0407 2A203234          1198          DEFM '* 24 BYTES OF RAM FOR POWER DOWN'
     20425954
     4553204F
     46205241
     4D20464F
     5220504F
```

```
LOC  OBJ.CODE      STMT-NR SOURCE-STMT PASS2 DEMO   DEMO   DEMO   ABS

     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
     45482052
     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
0548 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
```

```
LOC  OBJ.CODE     STMT-NR SOURCE-STMT PASS2 DEMO   DEMO   DEMO   ABS

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
     184804
                          *
                          * 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
065B 04              1298          DEFB EOT
                          *
065C                 1300          END
```

```
 NAME TYP VALUE DEF   REFERENCES      PASS2 DEMO   DEMO   DEMO   ABS

AMPM      0020  140
AMPM1   '  0189  740    737
AMPM2   '  018C  741    739
AMPMIN  '  008C  478    363
AMPMOT  '  01A4  720    287
AMPMSG  '  0631 1292    483
APRIL      0004   93
ASCII   '  027A 1001   1011
AUG        0008   97
BAUD       0008  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
```

```
 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   ' 00E8  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
```

```
 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
```

```
 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     00BE  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 PT4
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 DIRECTIO
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 DIRECTI
                        *
```

```
LOC  OBJ.CODE     STMT-NR SOURCE-STMT PASS2 CLKRAM CLKRAM CLKRAM REL

                          * 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
```

NAME	TYP	VALUE	DEF	REFERENCES
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* 169* 178 180* 188 204 209 21
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).

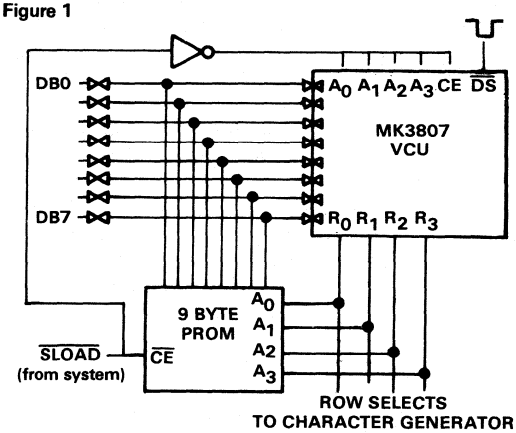
In addition to the seven control registers, two additional registers are provided to store the cursor character and row addresses for generation of the cursor video signal. The contents of these two registers can be read out onto the bus for update by the program or used by the microprocessor as two memory locations. (See Figure 2).

PROGRAM REGISTERS

The VCU contains 9 working registers (7 control registers and 2 data location registers).

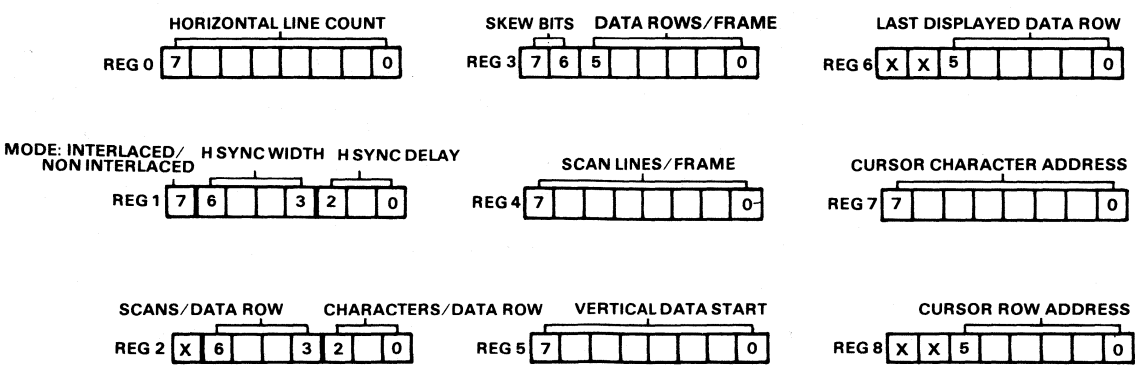
SELF LOADING SCHEME FOR VCU SET-UP

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

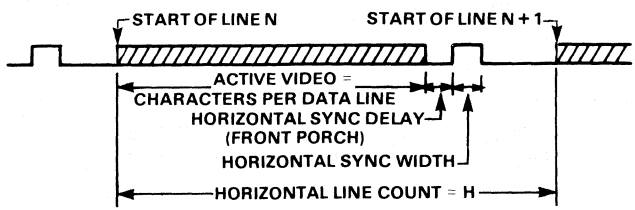
DB7	DB6	Sync/Blank Delay (Character Times)	Cursor Delay
0	0	0	0
1	0	1	0
0	1	2	1
1	1	2	2

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).

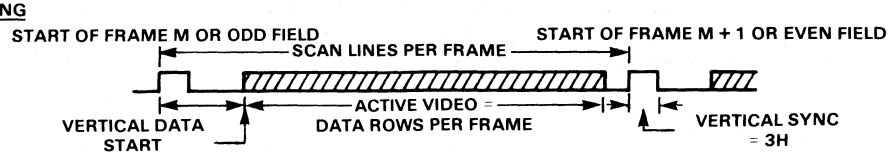
HORIZONTAL AND VERTICLE TIMING

Figure 3

HORIZONTAL TIMING

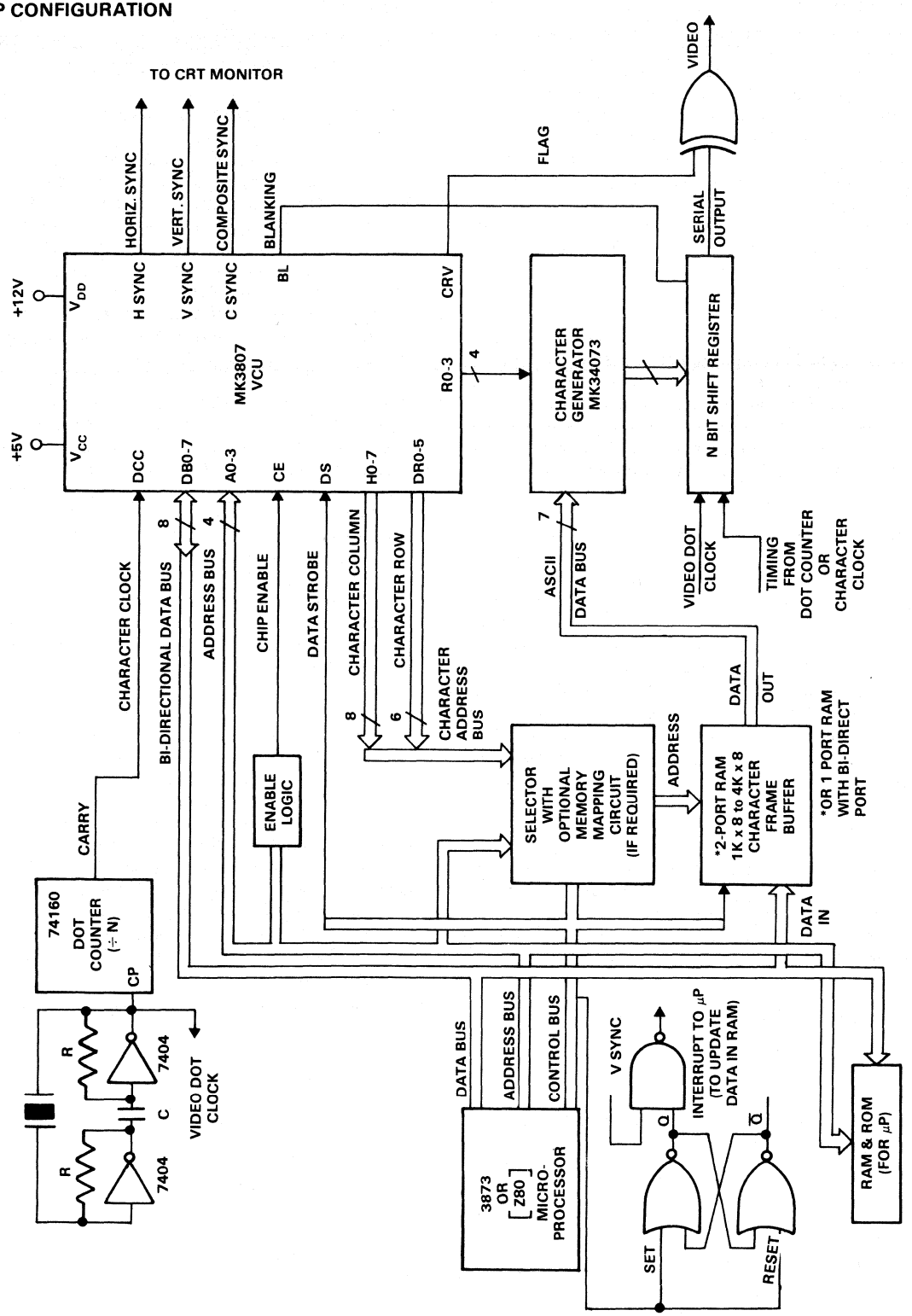


VERTICAL TIMING



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.

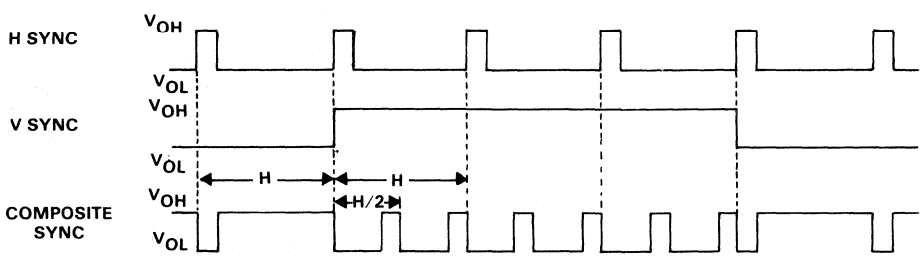
BASIC DISPLAY CONFIGURATION

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

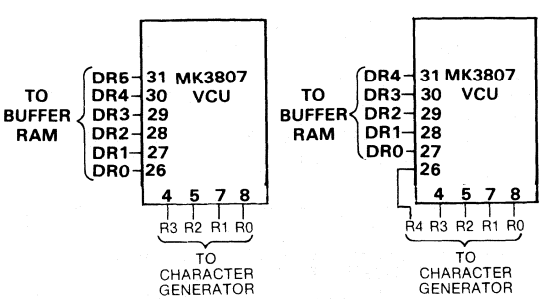
The VCU outputs the character position via the character counter outputs (H0-H7) and the data row counter outputs (DR0-DR5). These outputs define the character column and row location. They are used to address a character frame buffer RAM in which the frame image is stored. Since the VCU keeps counting 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

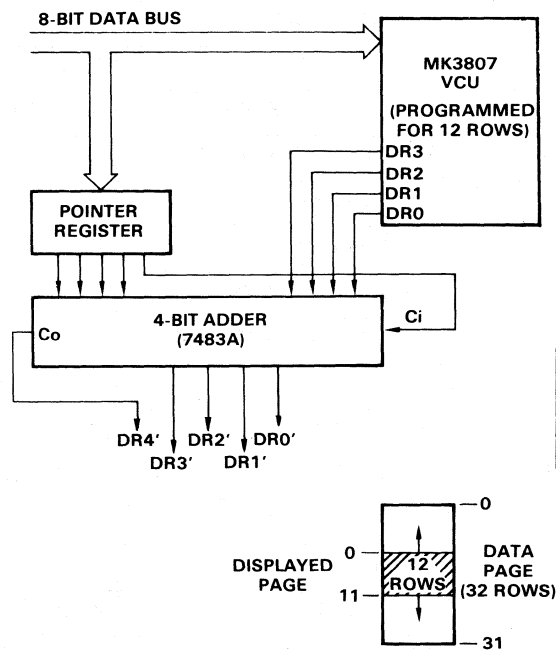
Page Scrolling

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

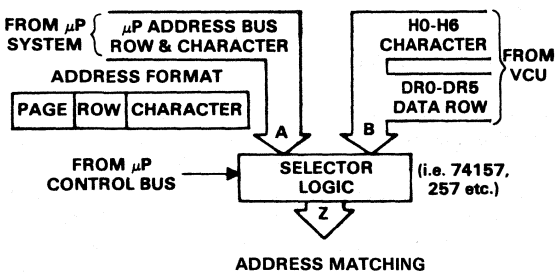
The character column and character row outputs combine to form the character address bus. This bus, along with the microprocessor address bus, is connected to a 2 X 1 selector which addresses the character frame buffer RAM. Figure 8 shows the selector and the mapping for the various formats of the standard 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.

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

Figure 8



ADDRESS BUS MAPPING

Table 1

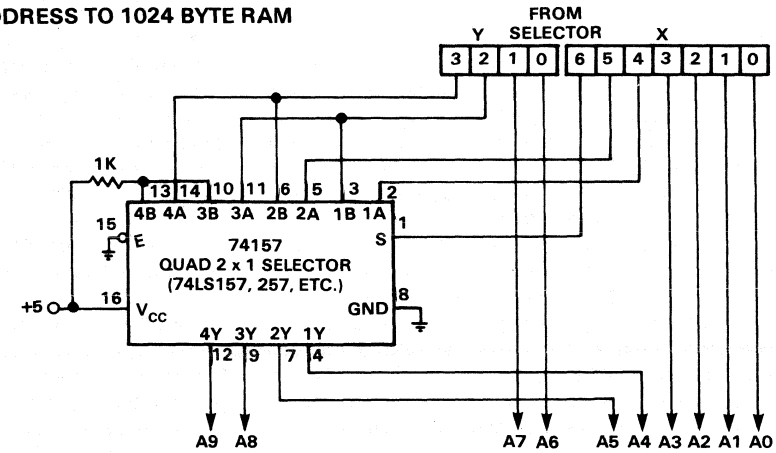
	SELECTOR													
μP ADDRESS BUS (UNUSED BITS ARE FOR PAGE LOCATION)	INPUT (A)	AB12	AB11	AB10	AB9	AB8	AB7	AB6	AB5	AB4	AB3	AB2	AB1	AB0
20 & 32 CHARACTERS/LINE														
FUNCTIONS					ROW						CHARACTER			
VCU OUTPUTS	INPUT (B)			DR5	DR4	DR3	DR2	DR1	DR0	H4	H3	H2	H1	H0
SELECTOR OUTPUTS	OUTPUT (Z)			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

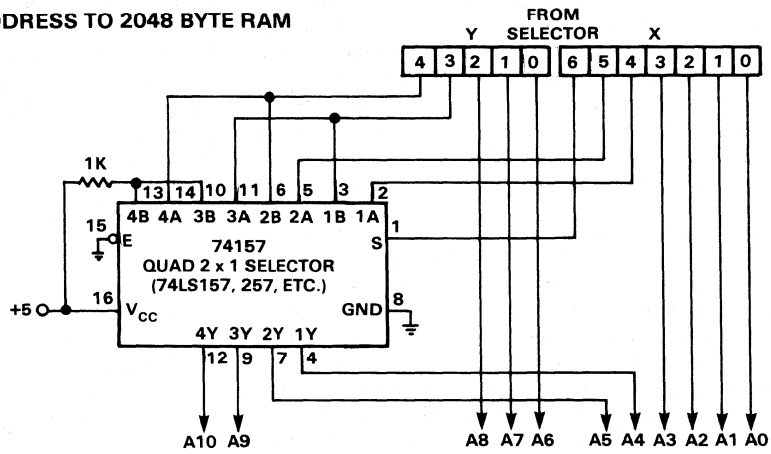
10 BIT BINARY ADDRESS TO 1024 BYTE RAM 12 LINES INTO 1K

Figure 9A



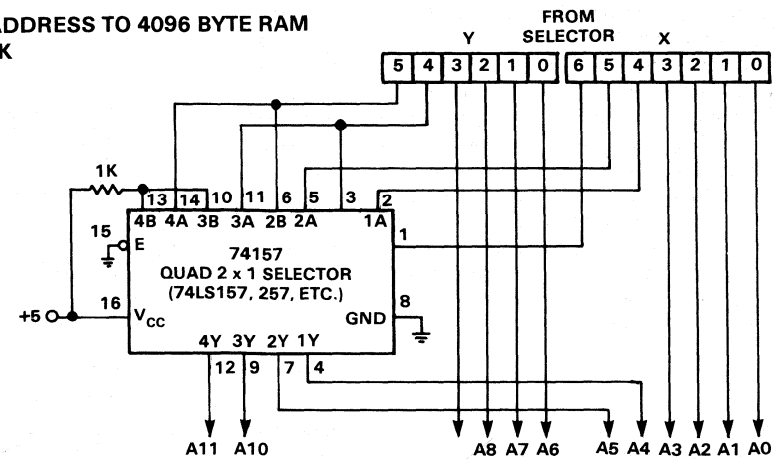
11 BIT BINARY ADDRESS TO 2048 BYTE RAM 24 LINES INTO 2K

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 (R0-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

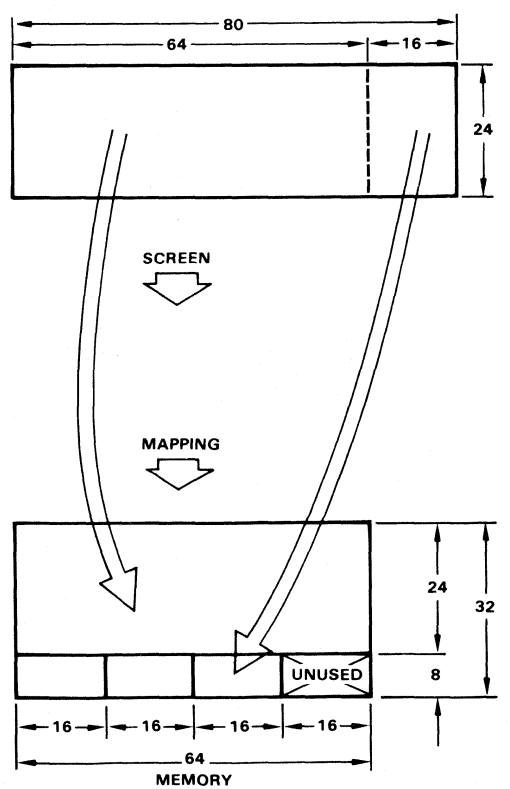
Some applications require adding alphanumeric characters (text) or graphics to the same screen as closed circuit or external (off-the-air) video. Figure 11 illustrates a simple technique of externally synchronizing the VCU using 2 chips (7474 and 7402 or equivalent). The external video can come from 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.

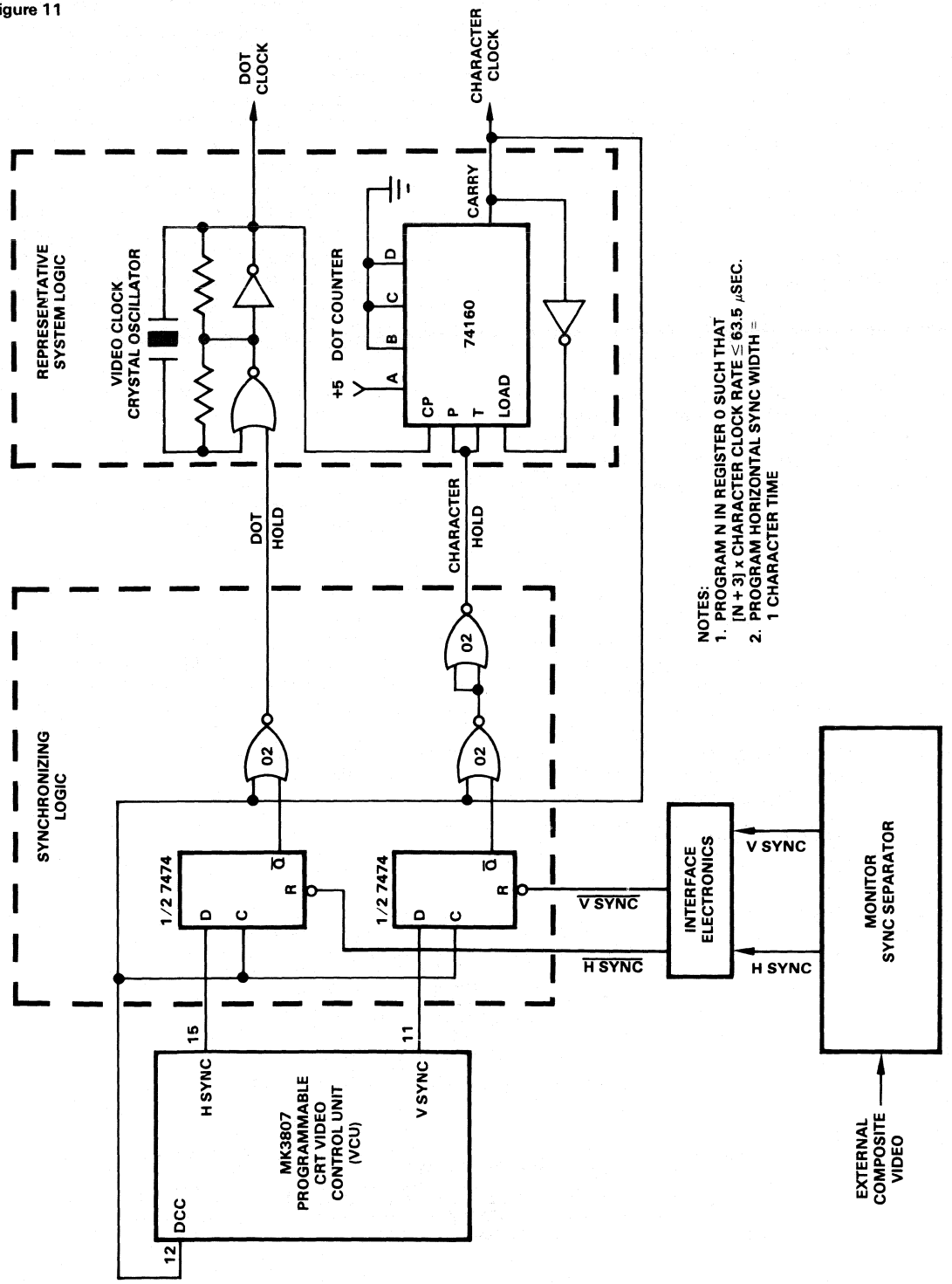
Tables 2A, B list work sheets which give the designer an

ADDRESS COMPRESSION SCHEME FOR 80 CHARACTERS/LINE
Figure 10



MK3807 EXTERNAL VIDEO SYNCHRONIZING CIRCUIT

Figure 11



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.

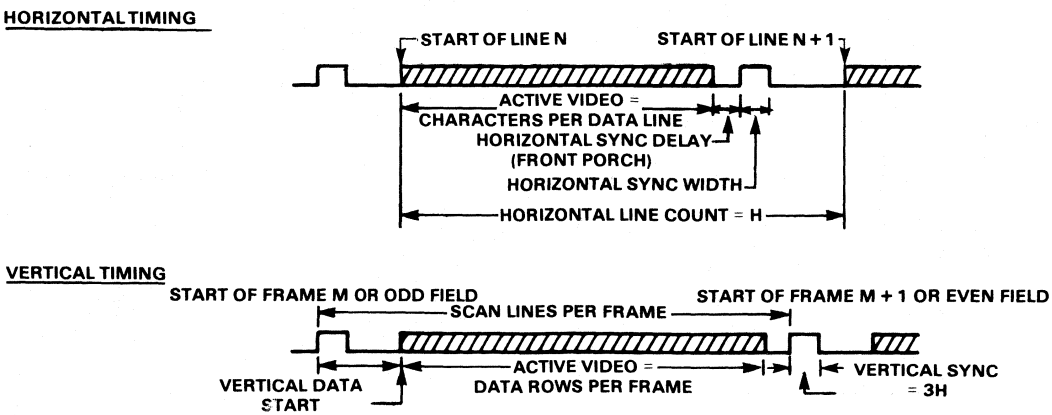
Vertical sync delay is the number of scan lines delay before vertical sync. Vertical sync pulse width should be expresed in scan line units. The VCU is fixed at the standard vertical sync width of 3 horizontal scan lines (3H). Scan line delay is the delay between vertical sync and the display information 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

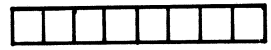
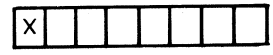
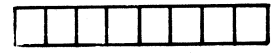
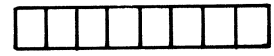
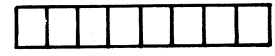
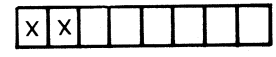


MK3807 VCU WORK SHEET
Table 2A

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

REG. #	ADDRESS 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

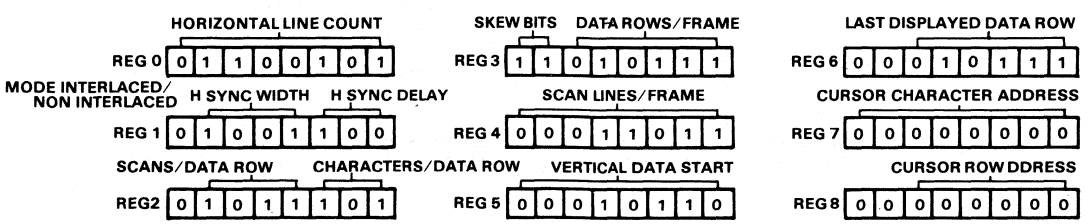
Table 2C

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 Spacing = 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.

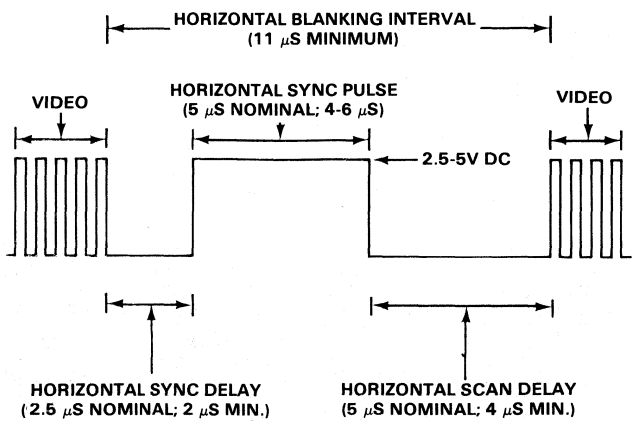
EXAMPLE FOR 40 CHARACTER BY 12 ROWS

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

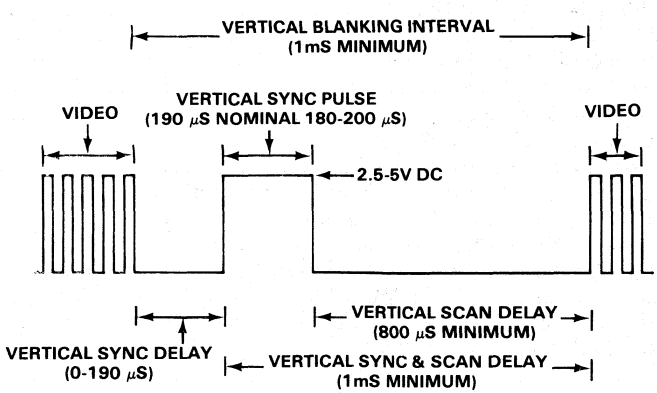


nominally with a 2 μs. minimum. At the same time, the horizontal blanking interval, which is composed of the front porch, horizontal sync pulse, and the back porch is defined as 11 μs. minimum. See Figures 14 and 15.

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:

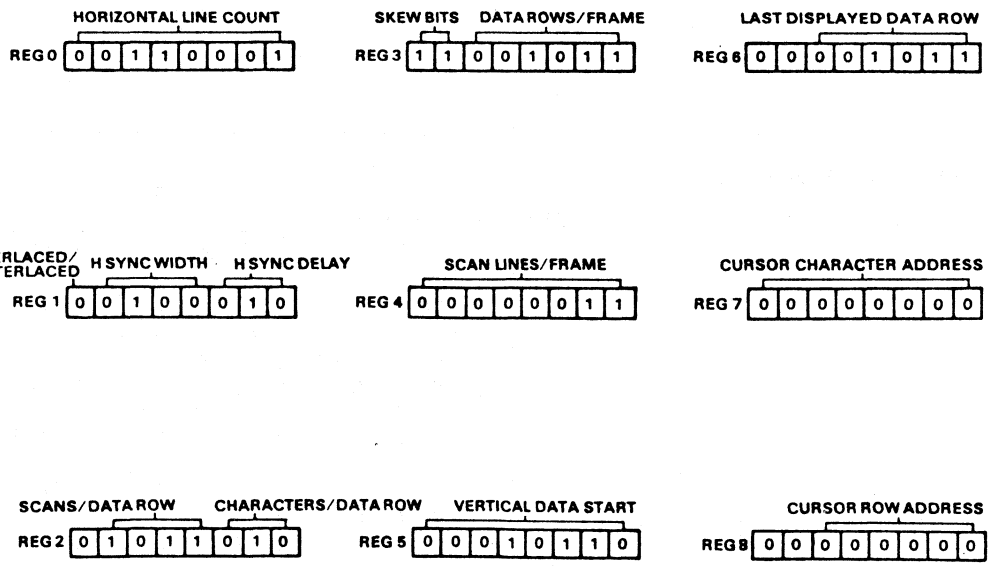
> Using 12 rows of data with row 0 on top of the screen and row 11 on the bottom and as programmed in register 6 with eleven, this will be the case. Now, if another number is programmed into register 6, such as 5, data row 5 will be on the bottom of the screen, while data row 6 will be on the top followed 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



APPLICATION NOTES

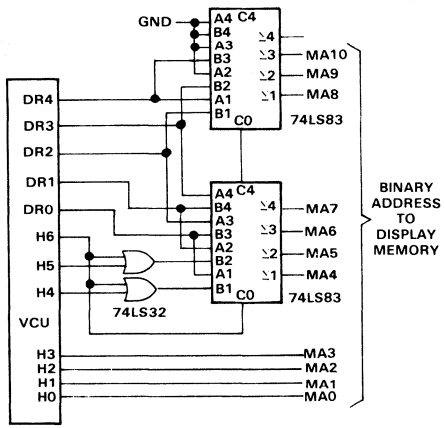
Conversion of Row Column to Binary Address

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.

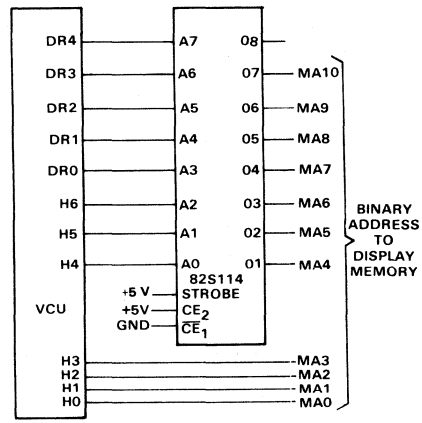
In essence the adders are used to add groups of 16. Since there are 5 groups of 16 in each data row of 80 characters, the adders effectively multiply the data row count (DR0-DR4) by 5 to obtain the starting binary address for each row. This is done by adding DR0-DR4 to itself shifted two positions to 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

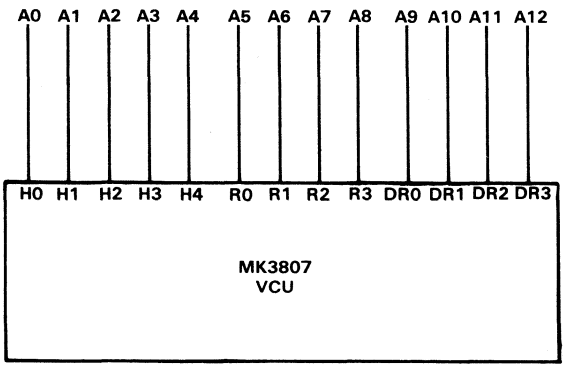
The VCU can be used for dot matrix graphic displays as well as alphanumeric displays. The following is an example of a 256 x 256 dot matrix graphic display using the raster line counter outputs (R0-R3) as part of the RAM addressing.

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



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.

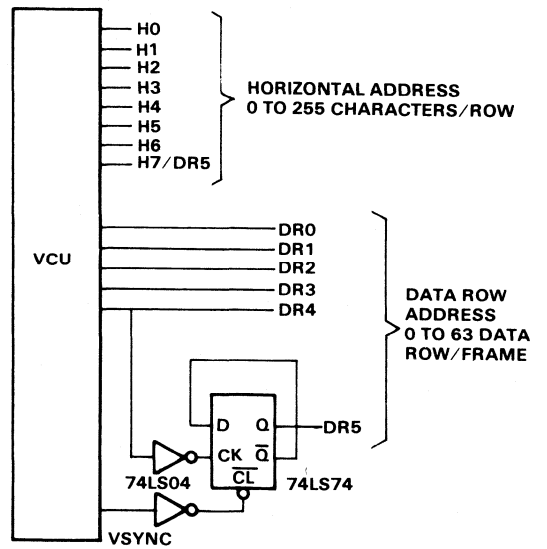
The horizontal row address will appear on outputs H0 to H7. Data row outputs DR0 to DR4 will provide five of the six bits required for the data row addressing. The circuit shown in Figure 19 will generate the required sixth row address bit.

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



3870/F8 MICROCOMPUTER DATA BOOK

PRELIMINARY

MOSTEK®

MICRO PERIPHERAL COMPONENTS

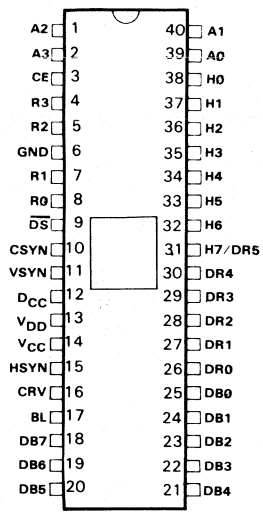
MK3807

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.

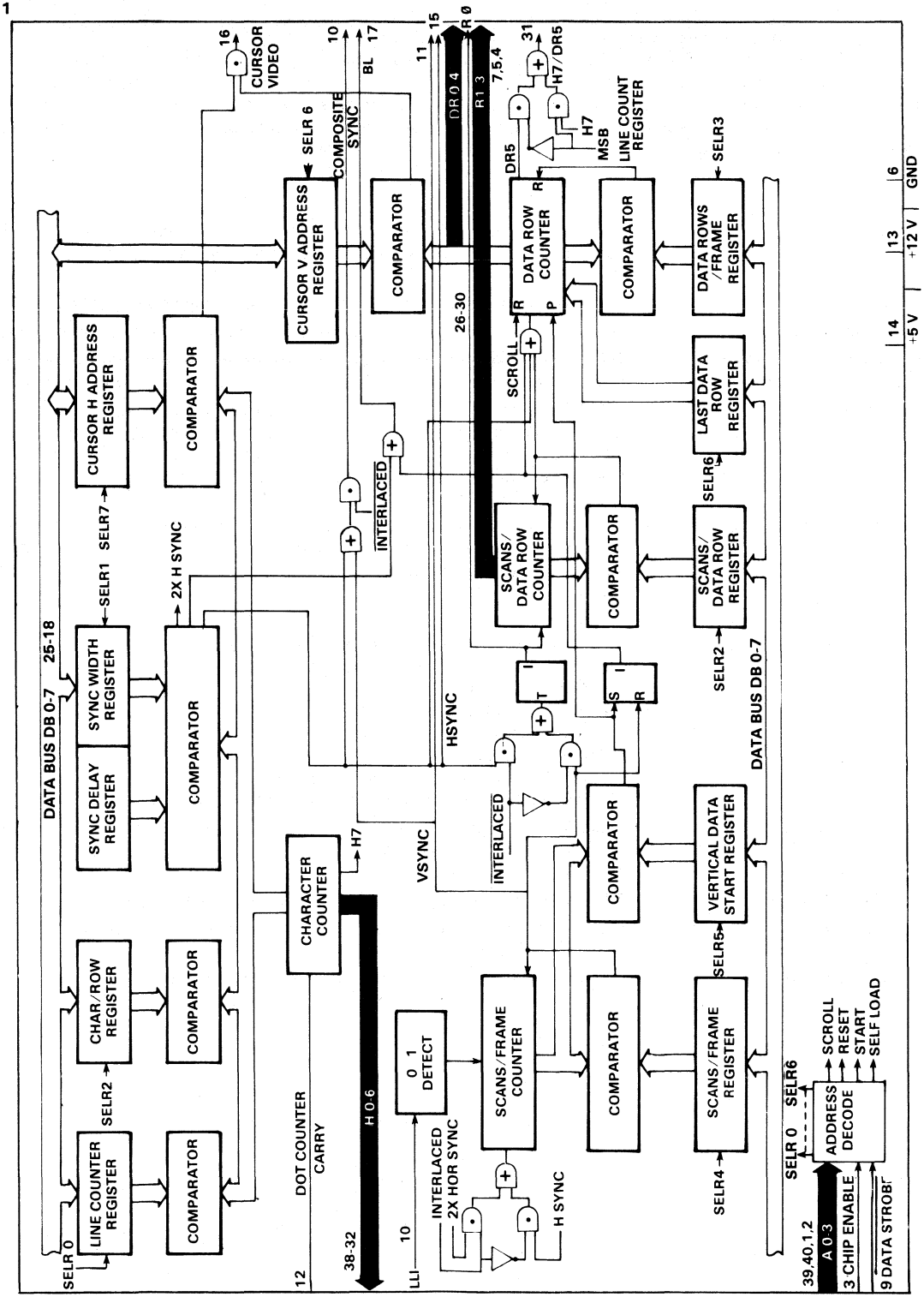
In addition to the seven control registers, two additional registers are provided to store the cursor character and data row addresses for generation of the cursor video signal. The contents of these two registers can also be read out onto the bus for update by the program.

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 $\geq$ 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:

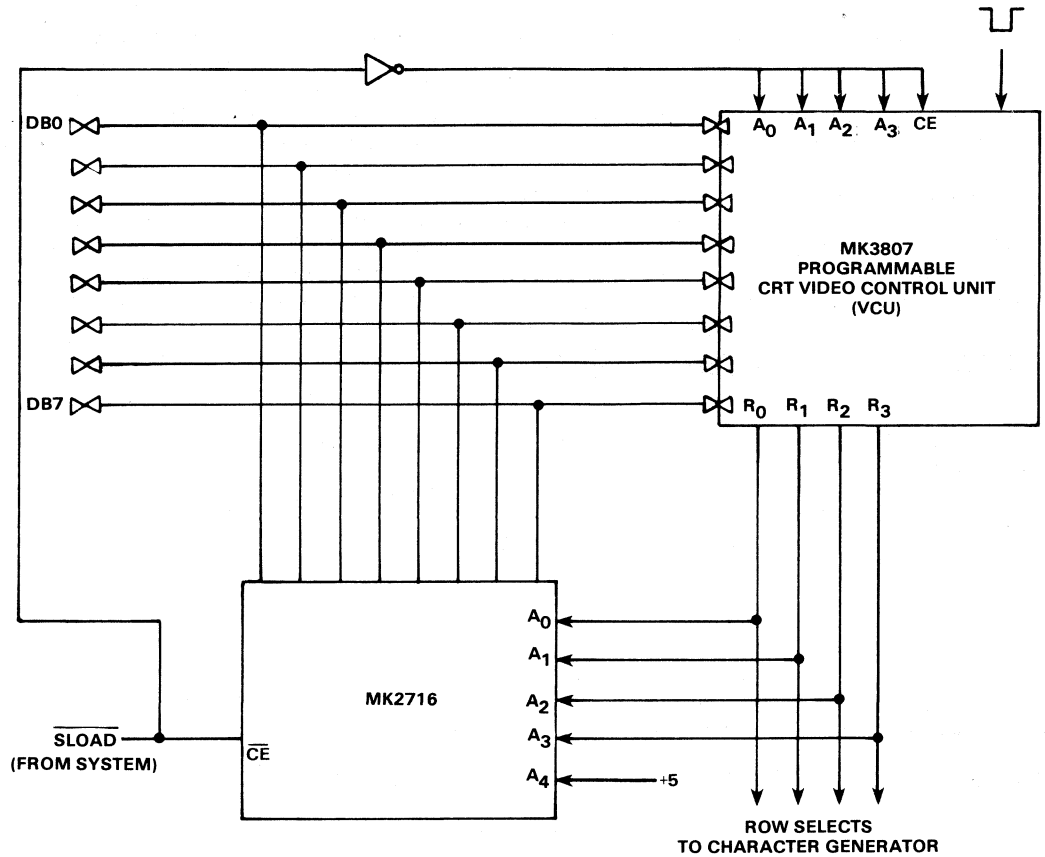
Interlace Mode

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

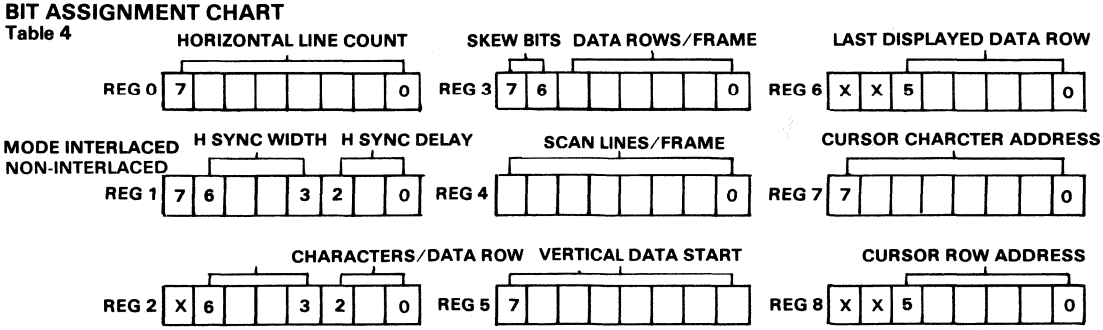
Table 3

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

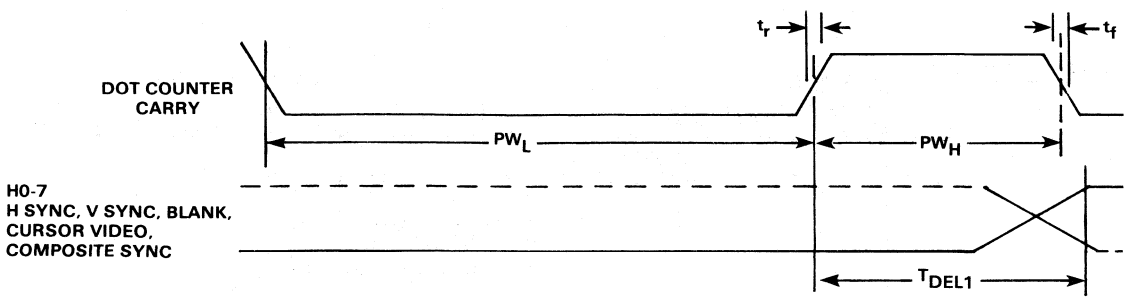
(T_A = 0°C to 70°C, V_{CC} = +5V ± 5%, V_{DD} = +12V ± 5%, unless otherwise noted)

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, $\overline{CRV}$					
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



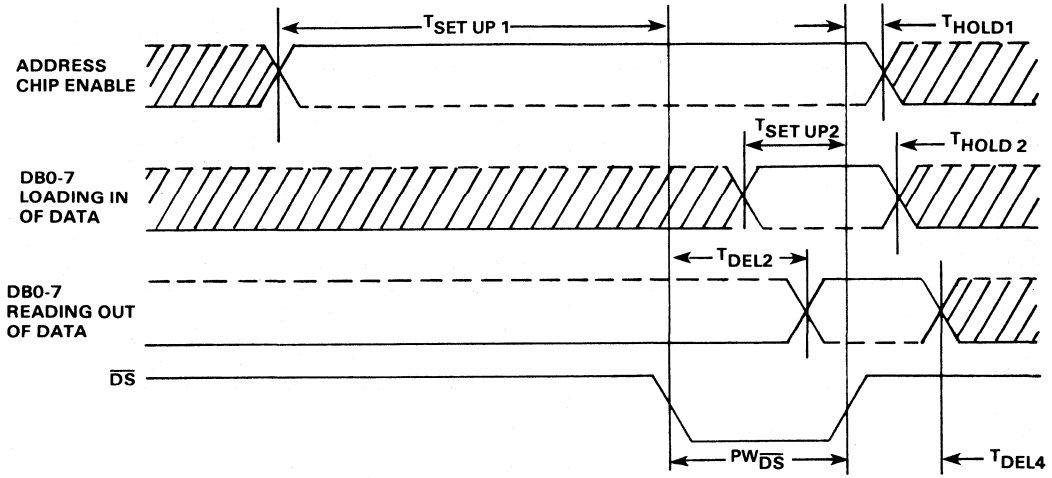
RESTRICTIONS

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.

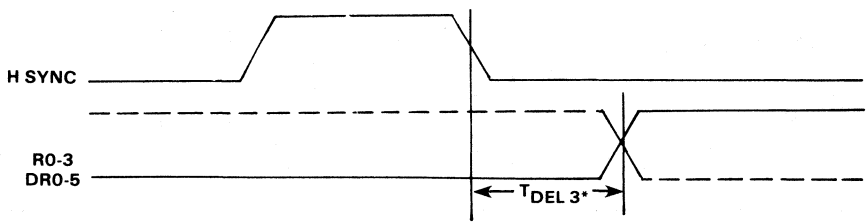
LOAD/READ TIMING

Figure 4



SCAN AND DATA ROW COUNTER TIMING

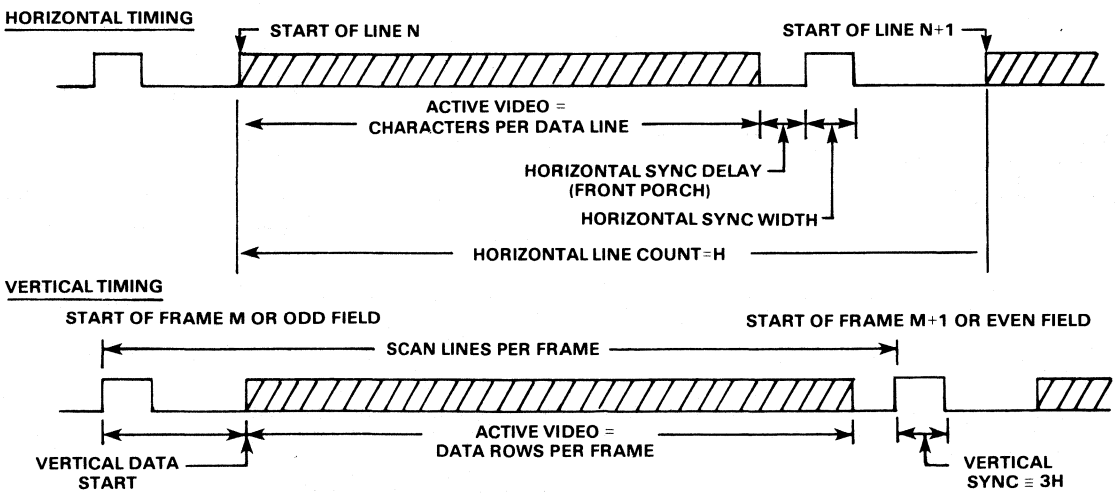
Figure 5



*R0-3 and DR0-5 may change prior to the falling edge of H sync

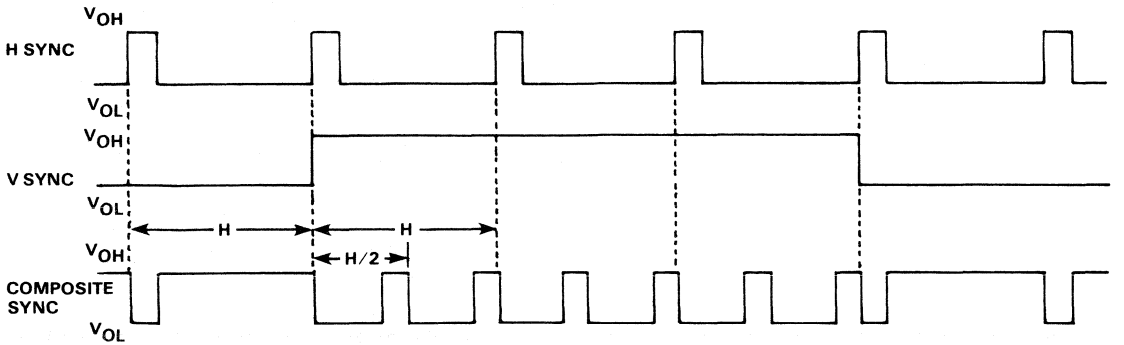
GENERAL TIMING

Figure 6



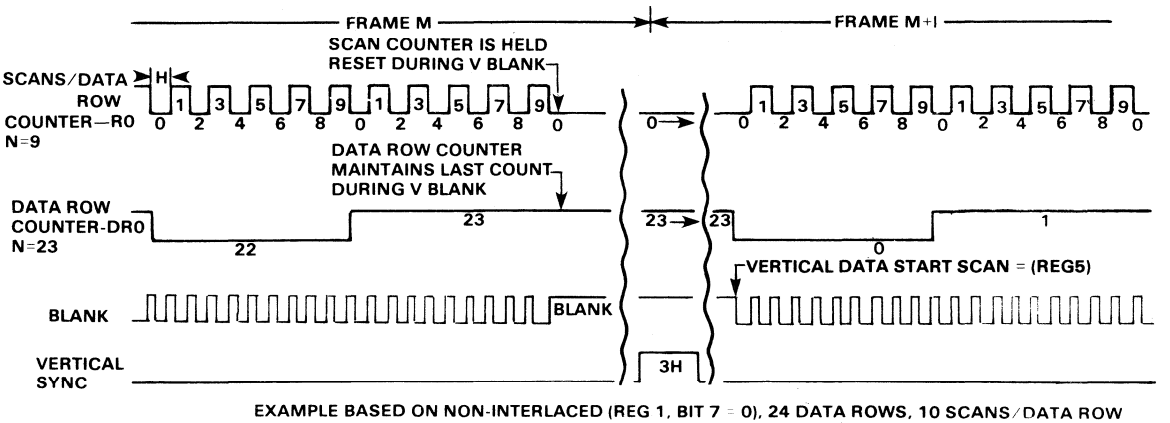
COMPOSITE SYNC TIMING

Figure 7



VERTICAL SYNC TIMING

Figure 8



EXAMPLE BASED ON NON-INTERLACED (REG 1, BIT 7 = 0), 24 DATA ROWS, 10 SCANS/DATA ROW

MOSTEK®

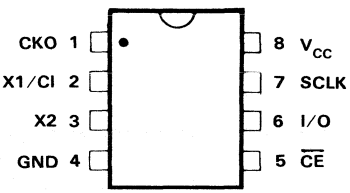
CMOS MICROCOMPUTER CLOCK/RAM

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.

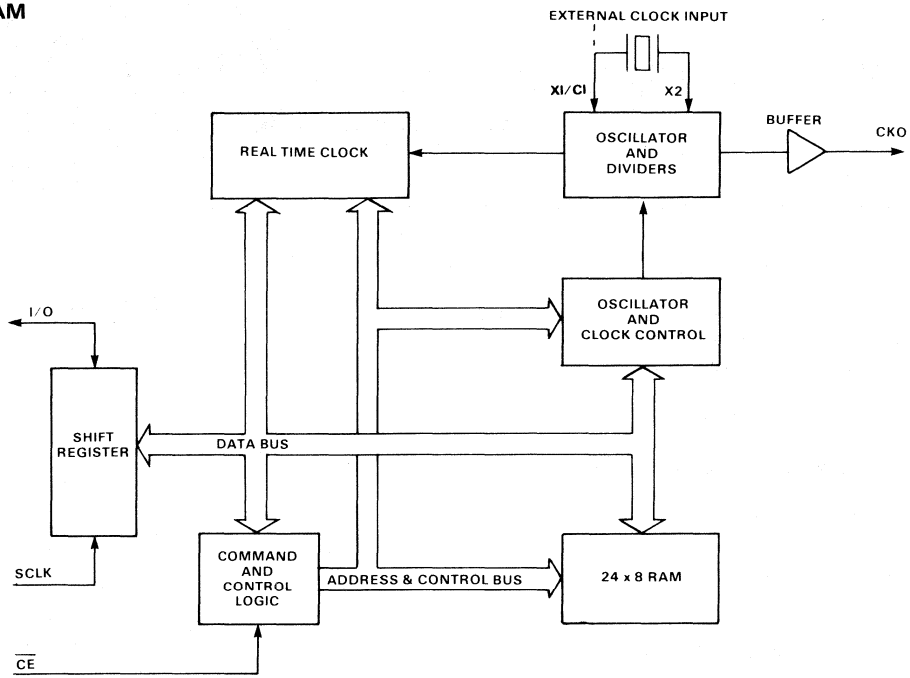
TECHNICAL DESCRIPTION

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.

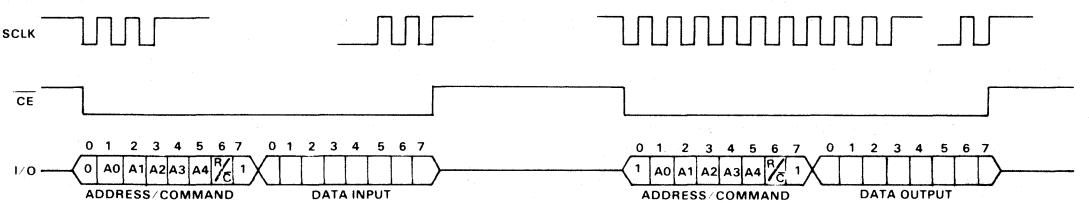
DATA TRANSFER SUMMARY

A data transfer summary is shown in Figure 2.

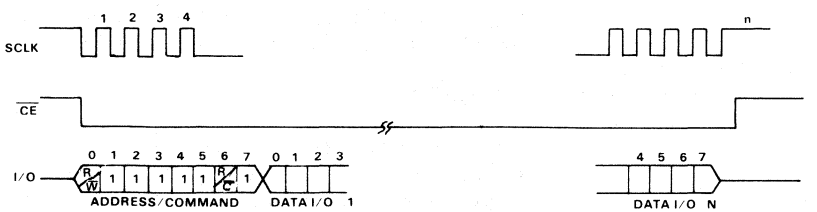
DATA TRANSFER SUMMARY

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

	7	6	5	4	3	2	1	0	REGISTER DEFINITION	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	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 / 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

$-40°C \leq T_A + 85°C$, V_{CC} 5V ± 10%

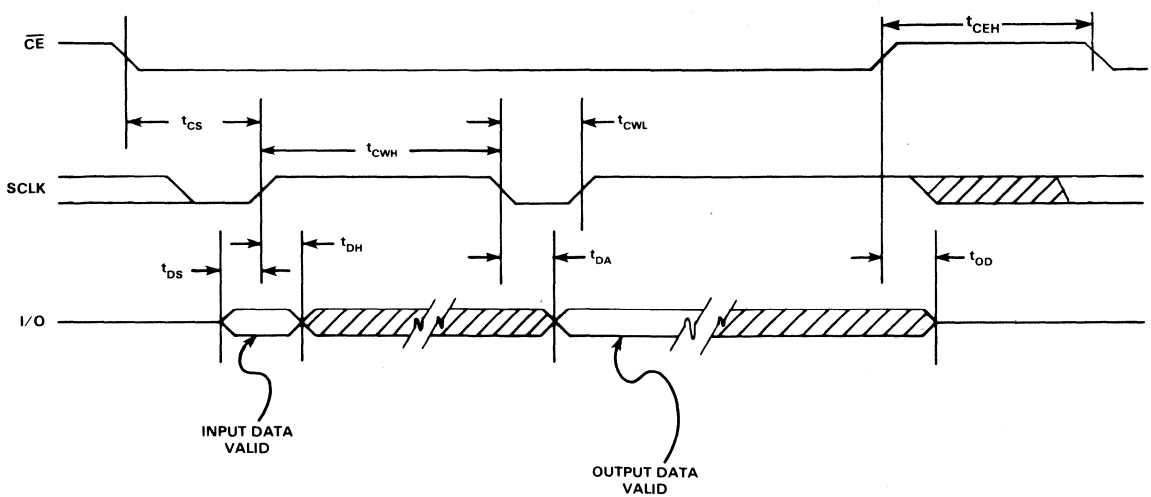
SYMBOL	PARAMETER	MIN	TYP	MAX	UNIT	NOTES
C_I	Capacitance on Input pin		6	10	pF	5
$C_{I/O}$	Capacitance on I/O pin		7	12	pF	5
C_X	Capacitance on XI/CI and X2		7	12	pF	5
f_X	Crystal frequency	27		8400	kHz	
t_{CS}	$\overline{CE}$ to SCLK set up time	1.0			μs	1,6
t_{DS}	Input Data to SCLK set up time	1.0			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 $\Delta V = 3V$, 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 $V_{OH} = 2.4$V and $V_{OL} = 0.4$V.
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®

INDUSTRIAL PRODUCTS

μP-Compatible A/D Converter

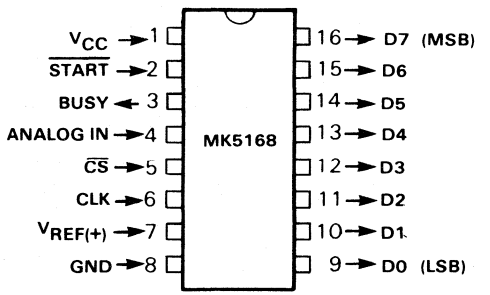
MK5168(N)-1

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 $\overline{CS}$ to a valid logic level with 56 pF load). A $\overline{START}$ signal starts the conversion process and, upon completion, BUSY is driven to logic 0. Continuous conversion is possible by tying the $\overline{START}$ 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)

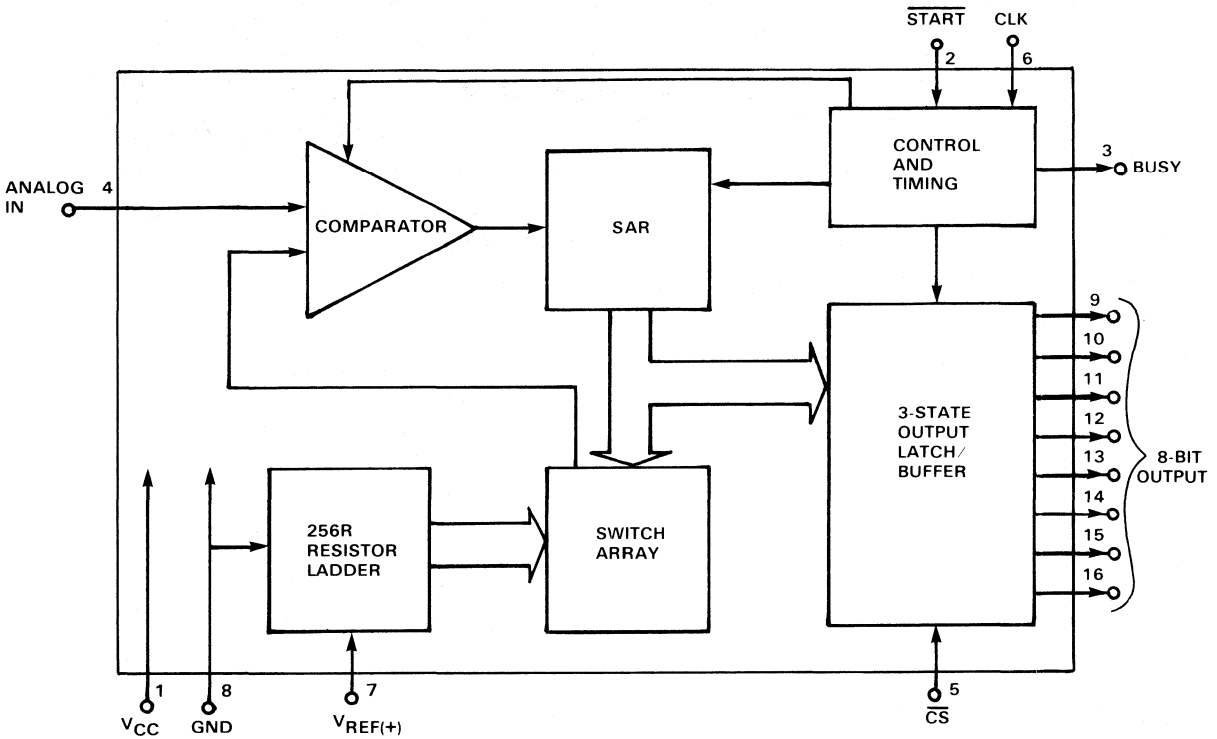
V_{CC}, Pin 1

V_{CC} must be connected to +5 Vdc ±5%.

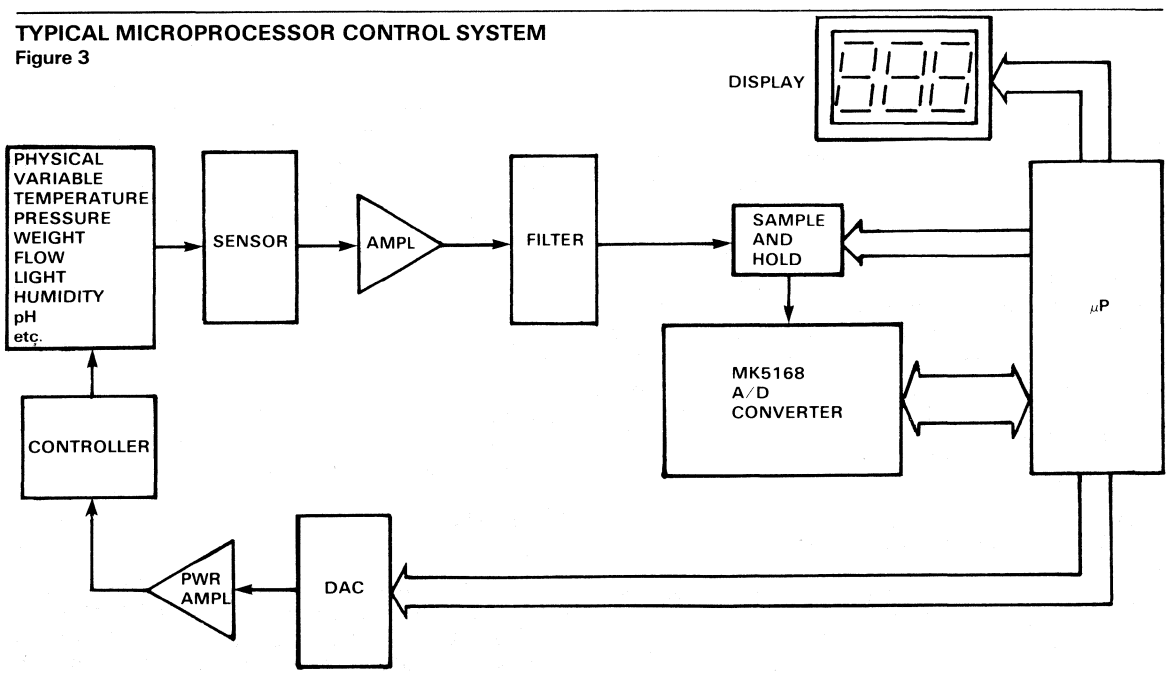
$\overline{START}$, Pin 2

The A/D Converter's successive approximation register (SAR) is reset by the falling edge of the $\overline{START}$ pulse. Conversion begins on the rising edge of the $\overline{START}$ pulse. A conversion in progress will be interrupted if a new $\overline{START}$ 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%.

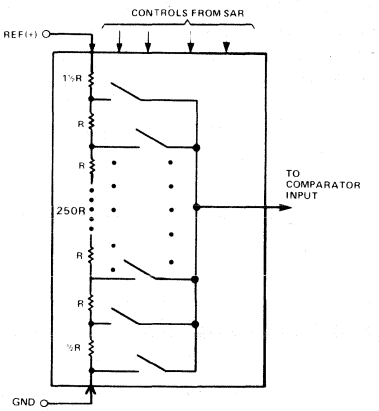
If CLK is grounded, the conversion process will be controlled by an on-chip oscillator, resulting in a typical conversion time of 150 μs.

$V_{REF(+)}$, 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 $-40 \leq T_A \leq +85°C$,

SYMBOL	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
R_{PS}	Power Supply Rejection	$4.75 \leq V_{CC} \leq 5.25$ $V_{REF(+)} = V_{CC}$		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 $-40 \leq T_A \leq +85°C$ unless otherwise noted

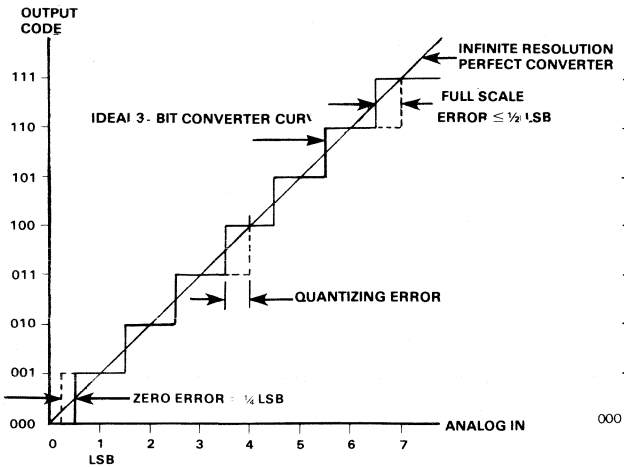
PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
Resolution				8	Bits	
Non-Linearity Error			± ¼	± ½	LSB	2
Zero Error			± ¼	± ½	LSB	4
Full-Scale Error			± ¼	± ½	LSB	5
Total Unadjusted Error	 T_A = 25°C		 ± ¼ ± ¼	 ± ¾ ± ½	 LSB LSB	 6 6
Quantizing Error				± ½	LSB	7
Absolute Accuracy	 T_A = 25°C		 ± ¾ ± ¾	 ± 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	

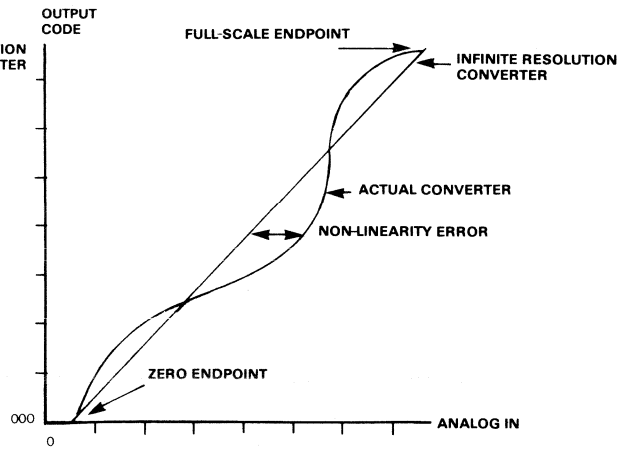
FULL-SCALE, QUANTIZING AND ZERO ERROR

Figure 5



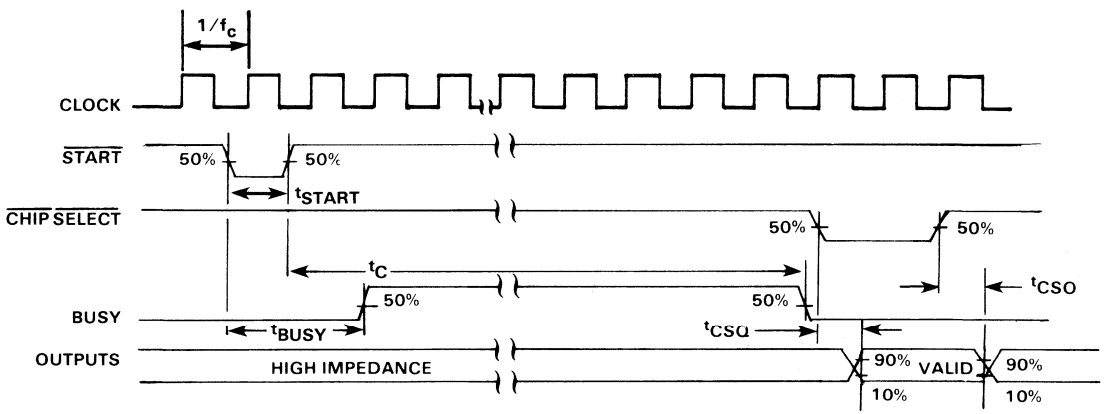
NON-LINEARITY ERROR

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/8-CHANNEL ANALOG MULTIPLEXER

MK50808(N/P)

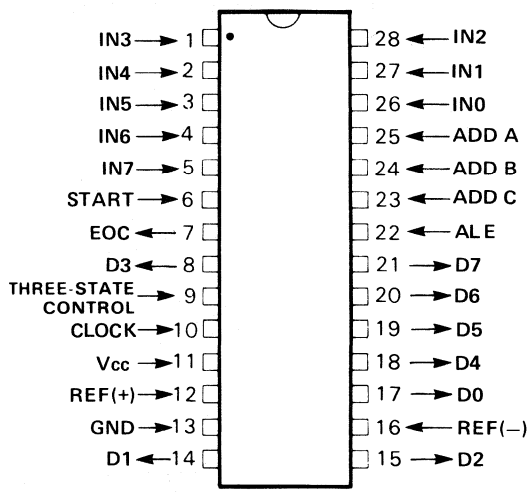
FEATURES

- ☐ 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

The pin configuration of the MK50808 is shown in Figure 1.

PIN CONNECTIONS

Figure 1



DESCRIPTION

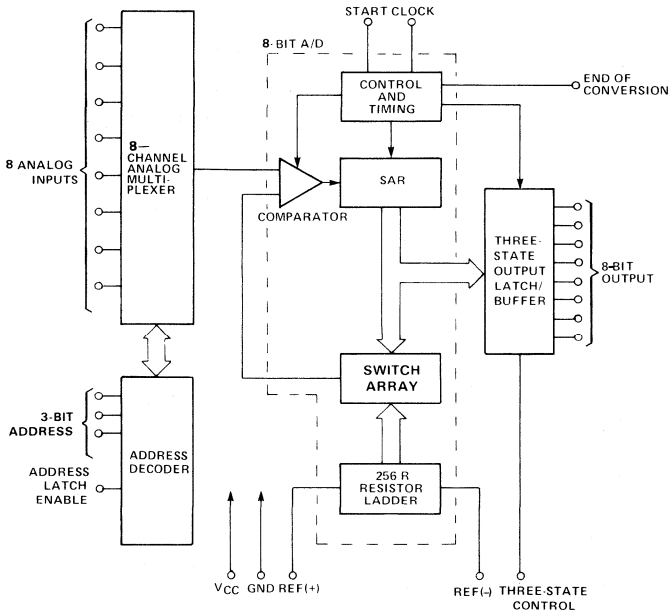
The MK50808 is a monolithic CMOS device with an 8-bit successive approximation A/D converter, an 8-channel analog multiplexer and microprocessor-compatible control logic. The 8-channel multiplexer can directly access any one of 8 single-ended analog channels. The 8-bit A/D converter consists of 256 series 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.

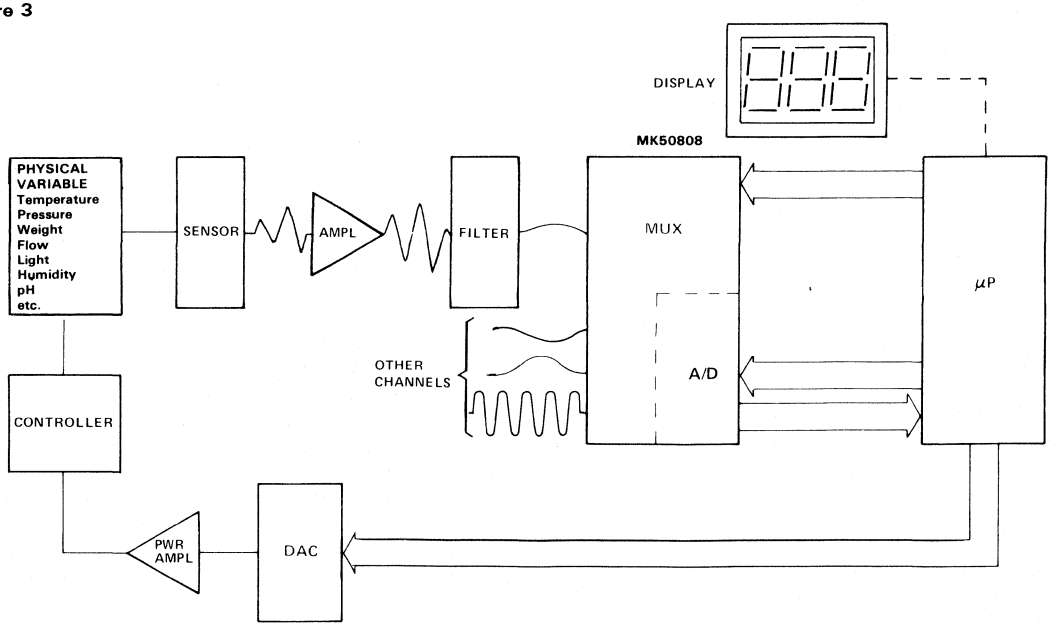
The MK50808 features low power, high accuracy, minimal temperature dependence, and excellent long-term accuracy and repeatability. These characteristics make this device ideally suited to machine and industrial controls.

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.

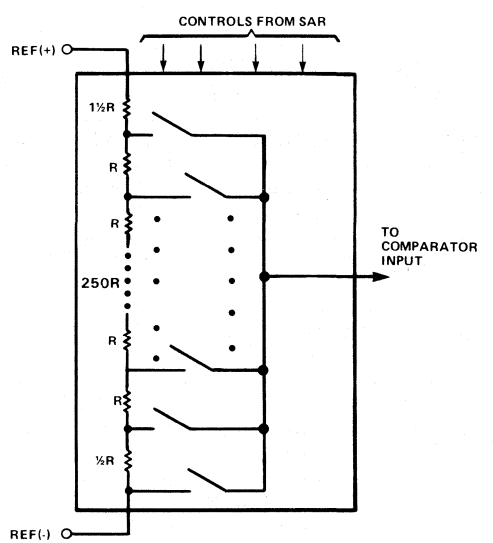
The chopper-stabilized comparator converts the DC input signal into an AC signal. This signal is amplified by a high-gain AC amplifier and the DC level is restored. This technique limits the drift component of the comparator because the drift is a DC component which is not passed by the AC amplifier.

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	
$\frac{(V_{REF(+)}+V_{REF(-)})}{2}$	Voltage at Center of Ladder	Measured at $R_{LADDER}/2$	$\frac{V_{CC}-0.1}{2}$	$\frac{V_{CC}}{2}$	$\frac{V_{CC}+0.1}{2}$	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} = 0V	 -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 noted

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}=0V, T_A = 25°C	-200	-10		nA	

CONVERTER SECTION

$V_{CC} = V_{REF}(+) = 5V$, $V_{REF}(-) = GND$, $V_{IN} = V_{COMPARATOR\ IN}$, $f_C = 640kHz$

MK50808-1, $-40 \leq T_A \leq +85°C$ unless otherwise noted

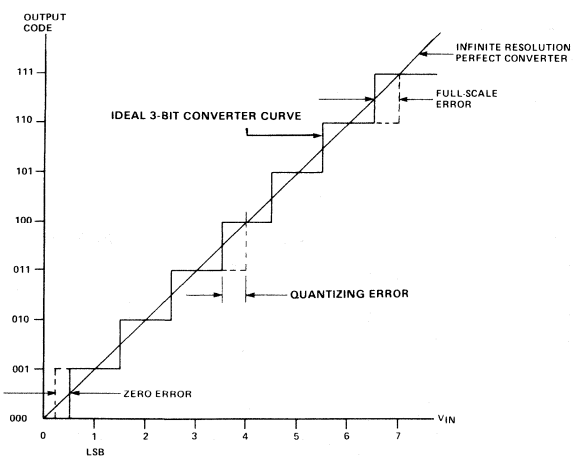
PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
Resolution				8	Bits	
Non-Linearity Error			± ¼	± ½	LSB	3
Zero Error			± ¼	± ½	LSB	5
Full-Scale Error			± ¼	± ½	LSB	6
Total Unadjusted Error	$T_A = 25°C$		± ¼ ± ¼	± ½ ± ¾	LSB LSB	7
Quantizing Error				±½	LSB	8
Absolute Accuracy	$T_A = 25°C$		± ¾ ± ¾	±1 ± 1¼	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

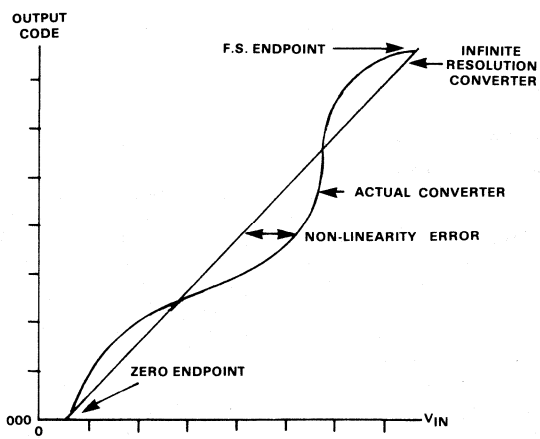
FULL-SCALE, QUANTIZING AND ZERO ERROR

Figure 5



NON-LINEARITY ERROR

Figure 6

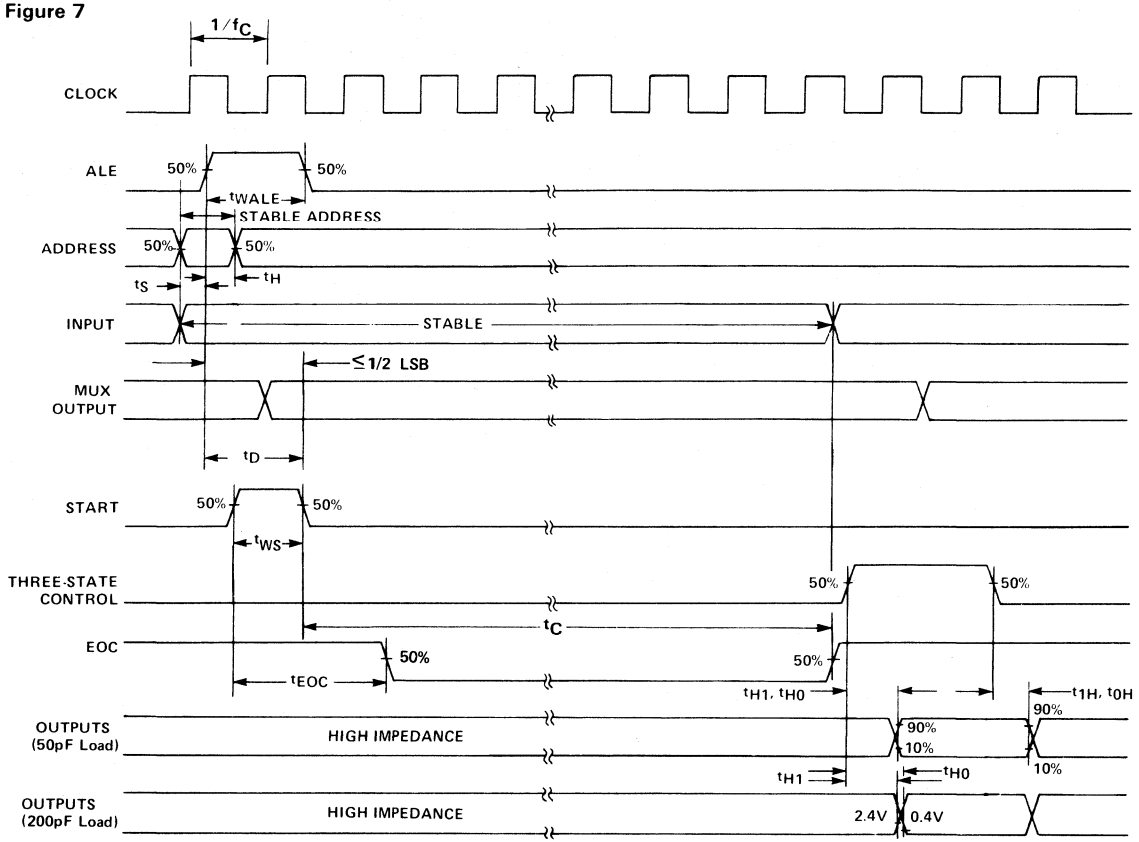


AC CHARACTERISTICS (Figure 7)
MK50808, MK50808-1, T_A = 25°C, V_{CC} = $V_{REF}(+)$ = 5V or 5.12V, $V_{REF}(-)$ = GND

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	$R_S + R_{ON} \leq 5k\Omega$		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.

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
- □ 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
- □ 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

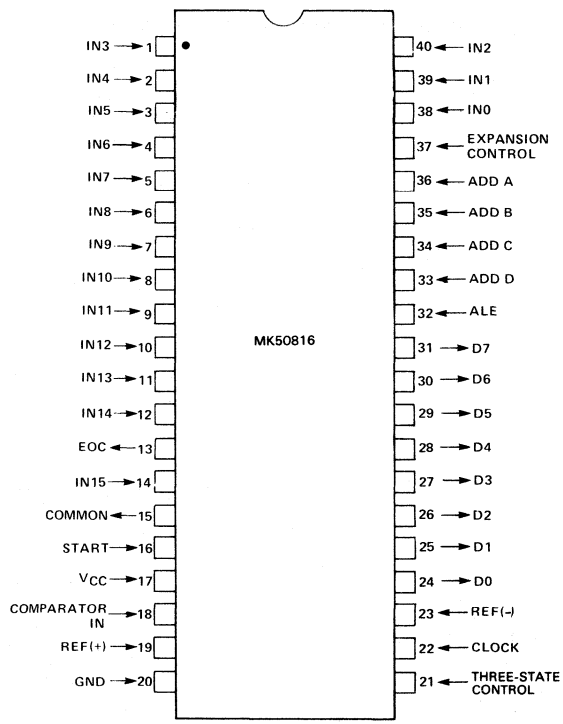
DESCRIPTION

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

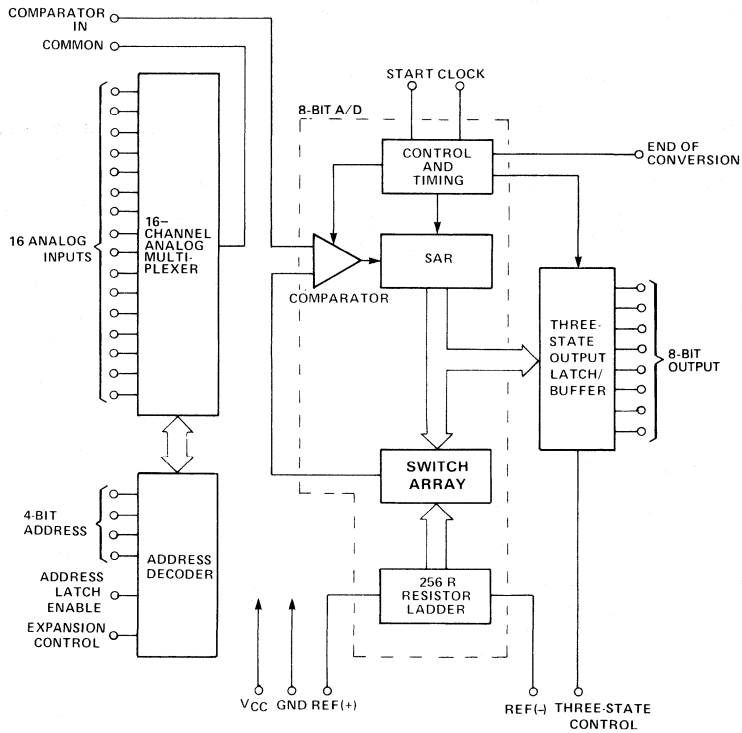


All digital outputs are TTL-compatible, all digital inputs are TTL-compatible with a pull-up resistor, and all digital inputs and outputs are CMOS-compatible; this makes it easy to interface with most microprocessors. The output latch is three-state and provides true bus-driving capability (300ns from 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.

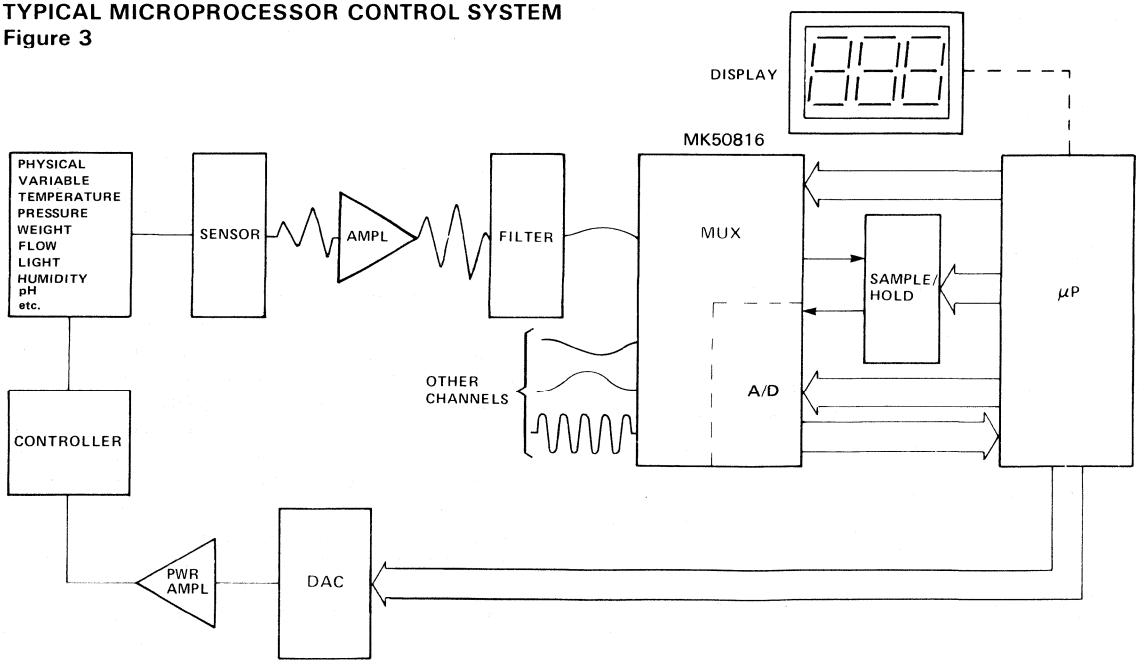
The MK50816 features low power, high accuracy, minimal temperature dependence, and excellent long-term accuracy and repeatability. These characteristics make this device ideally suited to machine and industrial controls.

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.

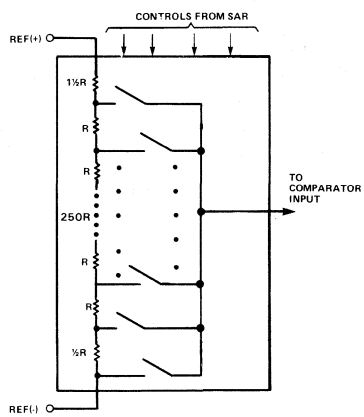
The top and bottom resistors of the ladder network in Figure 4 are not the same value as the rest of the resistors in the ladder. They are chosen so that the output characteristic will be symmetrical about its full-scale and zero points. The first output transition occurs when the analog signal reaches +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	
$\left(\frac{V_{REF(+)}+V_{REF(-)}}{2}\right)$	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} = –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} = 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
$-40° \leq T_A \leq +85°C$ 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 $-40 \leq T_A \leq +85°C$ 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

MK50816 $0° \leq T_A \leq +70°C$

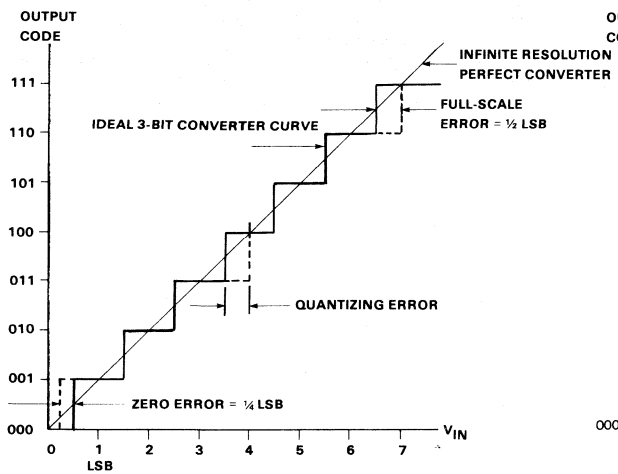
PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
Resolution				8	Bits	
Non-Linearity Error			± ½	± 1	LSB	3
Zero Error			± ¼	± ½	LSB	5
Full-Scale Error			± ¼	± ½	LSB	6
Total Unadjusted Error			± ½	± 1	LSB	7
Quantizing Error				± ½	LSB	8
Absolute Accuracy			± 1	± 1½	LSB	9

AC CHARACTERISTICS (Figure 7)

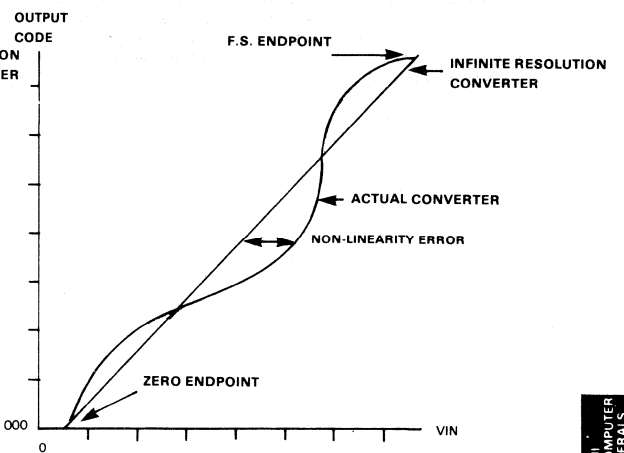
MK50816, MK50816-1 T_A = 25°C, V_{CC} = $V_{REF}(+)$ = 5V or 5.12V, $V_{REF}(-)$ = GND

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	

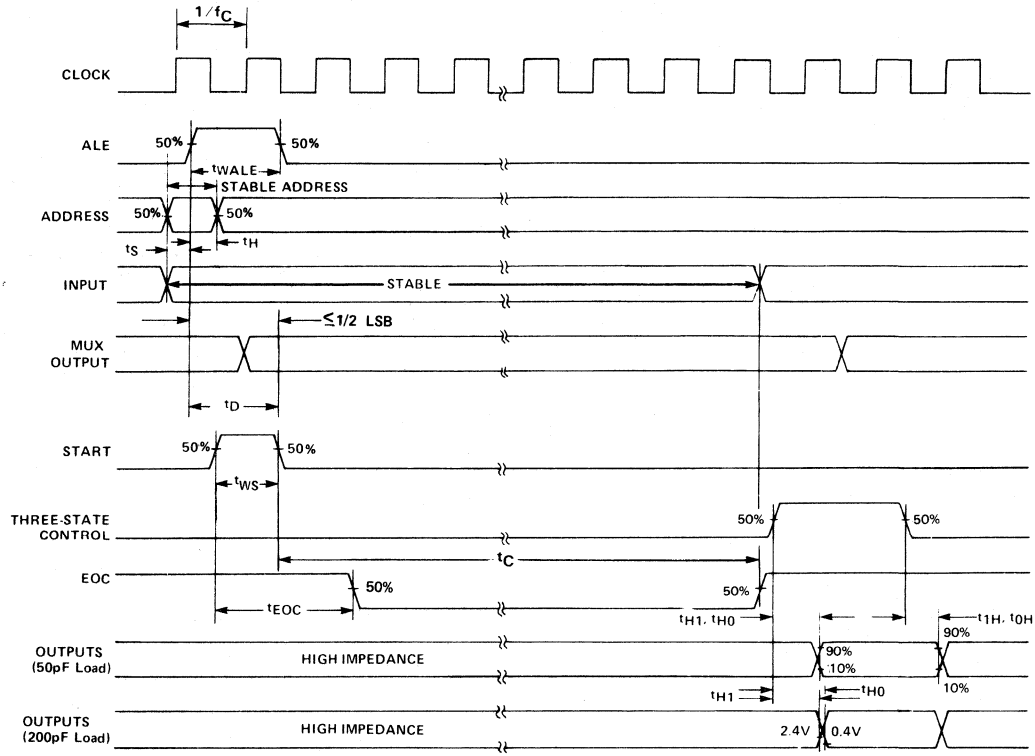
FULL SCALE, QUANTIZING AND ZERO ERROR
Figure 5



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.

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. MK79610