MOS TECHNOLOGY, INC.
VALLEY FORGE CORPORATE CENTER (215) 666-7950
950 RITTENHOUSE ROAD, NORRISTOWN, PA. 19401

February 1976

Dear Sir:

Here is our latest newsletter updating you on our activities including new products, more detailed pricing and information on various items raised by our customers over the past few months. We are now delivering TIM chips (MCS6530-004) in volume and a special "ROM-less" MCS6530 (RAM, I/O and timer only, the MCS6530-005) in prototype systems where the mask-programmed ROM is not required.

You have recently received a brochure on our KIM-1 Microcomputer System and we have referenced that offering in both our Price List and Order Form in this mailing. We are indeed gratified by the response to this product by the marketplace.

MOS TECHNOLOGY, INC. continues to deliver the lowest cost and fastest 8 bit Microprocessor on the market. Synertek, our second source, will very shortly be delivering this product as well. We are also delivering 2 MH_Z microprocessors, identifiable on the price list by our "A" suffix.

Included in this newsletter are the following:

1. Price List - indicating low volume price and delivery of the microcomputer products.

2. TIM Program Description - providing a basic description of the features found in the pre-programmed MCS6530-004.

3. MDT System Description - providing a basic description of the Microcomputer Development Terminal. This sophisticated but easy to use system development tool will be available in the second quarter of 1976.

4. Handling to prevent static damage - some comments regarding recommended care in handling MOS TECHNOLOGY, INC. products.

5. Single Cycle/Single Instruction Schematic - several of our customers have brought to our attention that our Static Test Control Logic Schematic on page 125 of the Hardware Manual is incorrect. We regret the delays this may have caused any of our customers who attempted to use this circuitry. The enclosed schematic will allow you to perform Single Cycle and Single Instruction executions and should be used in place of that found in the Hardware Manual.

6. New Order Form

7. Discussion of MCS650X Circuit Modifications

8. Discussion of Crystal and RC Time Base Generation.

9. Listing of MOS TECHNOLOGY, INC. Sales Representatives.

10. UCS Timesharing Systems Brochure for the MCS650X product line.

Our next newsletter will introduce some of our new products which will be coming out in the remainder of 1976. In the mean time, we will continue to augment our customer support activities both in the factory and in the field to keep pace with our rapidly growing customer base.

Very truly yours,

MOS TECHNOLOGY, INC.

Charles I. Peddle
Marketing Director
Microcomputers

CIP/nac
Encl.

If your name or address is incorrect or if you are receiving duplicate mailings please return this portion with the correct information.

NAME_____

COMPANY_____

ADDRESS_____

_____

This is a duplicate mailing_____

MOS TECHNOLOGY, INC.
VALLEY FORGE CORPORATE CENTER (215) 666-7950
950 RITTENHOUSE ROAD, NORRISTOWN, PA. 19401

PRICE LIST

PART NUMBER	1 - 99	100 - 999	Availability
MCS6501 - 1 MH_z	$ 20.00	$ 18.00	
MCS6502 - 1 MH_z	25.00	21.00	
MPS6503 - 1 MH_z	20.00	18.00	Off the shelf
MPS6504 - 1 MH_z	20.00	18.00	
MPS6505 - 1 MH_z	20.00	18.00	
MCS6502 A*	37.50	31.50	
MPS6503 A	30.00	27.00	Off the shelf
MPS6504 A	30.00	27.00	
MPS6505 A	30.00	27.00	
MCS6530-004 (TIM) (Includes TIM Manual)	30.00	26.00	Off the shelf
MCS6530-005 (No ROM Available)	18.00	16.00	Off the shelf
MCS6530 - Custom Program**	30.00	26.00	8 - 10 weeks after receipt of order
KIM-1 System	$245.00/system		Off the shelf

Postage and Handling
$4.50 - Domestic
20.00 - International

* "A" suffix imples 2 MH_z product

** Mask tooling for custom programs is $1,000.00 with a minimum purchase quantity of 50 units. The $1,000.00 will be refunded if more than 1,000 units of the unique pattern is purchased within the first 12 months.

Questions concerning large volume price and delivery on all products should be directed to Mr. Julius C. Hertsch, Product Manager, MOS TECHNOLOGY, INC. (215-666-7950 x 220).

MCS6530-004 (TIM)

SOFTWARE DEVELOPMENT AID

TIM is the Terminal Interface Monitor program for MOS TECHNOLOGY, INC.'s MCS650X microprocessors. It is supplied in read-only memory (ROM) as part of the MCS6530 multi-function chip. Because the TIM code is non volatile, it is available at system power-on and cannot be destroyed inadvertently by user programs. Furthermore, the user is free to selectively use only those TIM capabilities which he needs for a particular program. Both interrupt types, interrupt request (IRQ) and non-maskable interrupt (NMI) may be set to transfer control to TIM or directly to the user's program.

TIM communicates with the user via a serial full-duplex port (using ASCII codes) and automatically adjusts to the speed of the user's terminal. Any speed... even non-standard ones...can be accommodated. If the user's terminal has a long carriage return time, TIM can be set to perform the proper delay. Commands typed at the terminal can direct TIM to start a program, display or alter registers and memory locations, set breakpoints, and load or punch programs. If available in the system configuration, a high-speed paper tape reader may be used to load programs through a parallel port on the MCS6530 chip. Programs may be punched in either of two formats -- hexadecimal (assembler output) of BNPF (which is used for programming read-only memories). On loading or modifying memory, TIM performs automatic read-after-write verification to insure that addressed memory exists, is read-write type, and is responding correctly. Operator errors and certain hardware failures may thus be detected using TIM.

TIM also provides several subroutines which may be called by user programs. These include reading and writing characters on the terminal, typing a byte in hexadecimal, reading from high-speed paper tape, and typing a carriage-return, line-feed sequence with proper delay for the carriage of the terminal being used. Program tapes loaded by TIM may also specify a start address so that programs may be started with a minimum of operator action.

TIM is normally entered when a 'BRK' instruction is encountered during program execution. At that time CPU registers are output: * PC F A X Y S and control is given to the terminal.

Note: PC = Program Counter
F = Processor Status Register
A = Accumulator
X = Index Register, X
Y = Index Register, Y
S = Stack Pointer

In summary, TIM's features include the following capabilities:

- Self adapting to any terminal speed for 10-30 cps
- Display and Alter CPU registers
- Display and Alter Memory locations
- Read and Write/Punch hex formatted data

- Write/Punch BNPF format data for PROM programmers
- Unlimited breakpoint capability
- Separate non-maskable interrupt entry and identification
- External device interrupts directable to any user location or defaulted to TIM recognition
- Capability to begin or resume execution at any location in memory
- Completely protected, resident in Read-Only Memory
- Capability to bypass TIM entirely to permit full user program control over system
- High speed 8-bit parallel input option
- User callable I/O subroutines

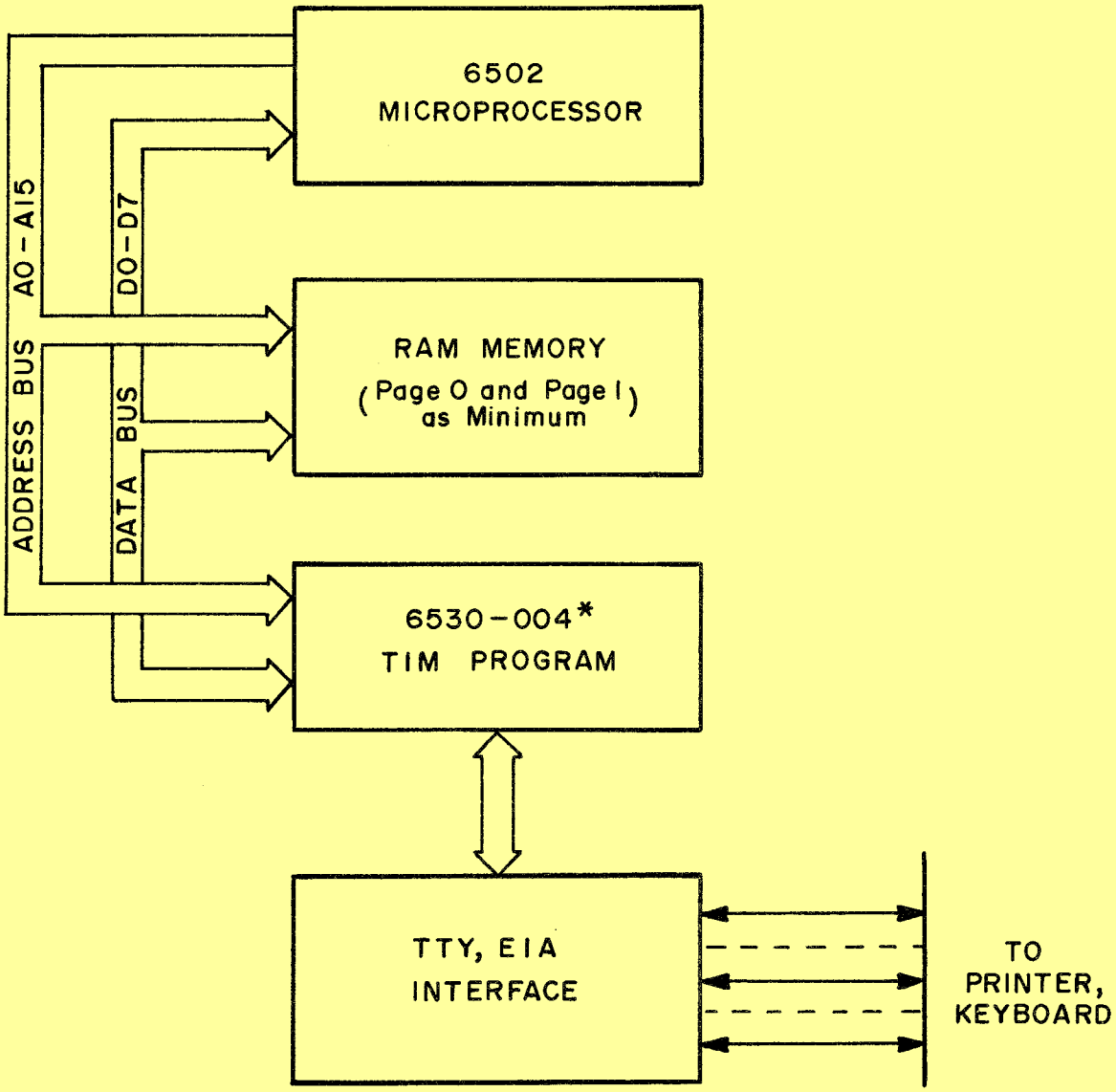
The commands used for directing TIM to perform these functions have been held to a minimum. This means that TIM is easily learned and readily remembered. TIM's Command Set consists of:

.R	Display registers (PC, F, A, X, Y, S)
.M ADDR	Display memory (8 bytes beginning at ADDR)
.: DATA	Alters perviously displayed item
.LH	Load Hexadecimal tape
.WB ADDR1, ADDR2	Write BNPF tape (from ADDR1 to ADDR2)
.WH ADDR1, ADDR2	Write hex tape (from ADDR1 to ADDR2)
.G	Go, continue execution from current PC address
.H	Toggles high-speed-reader option (if its on, turns it off; if off, turns on)

TIM is offered in the form of a 1K x 8 program resident in the MCS6530-004 at a 1 to 99 price of $30.00

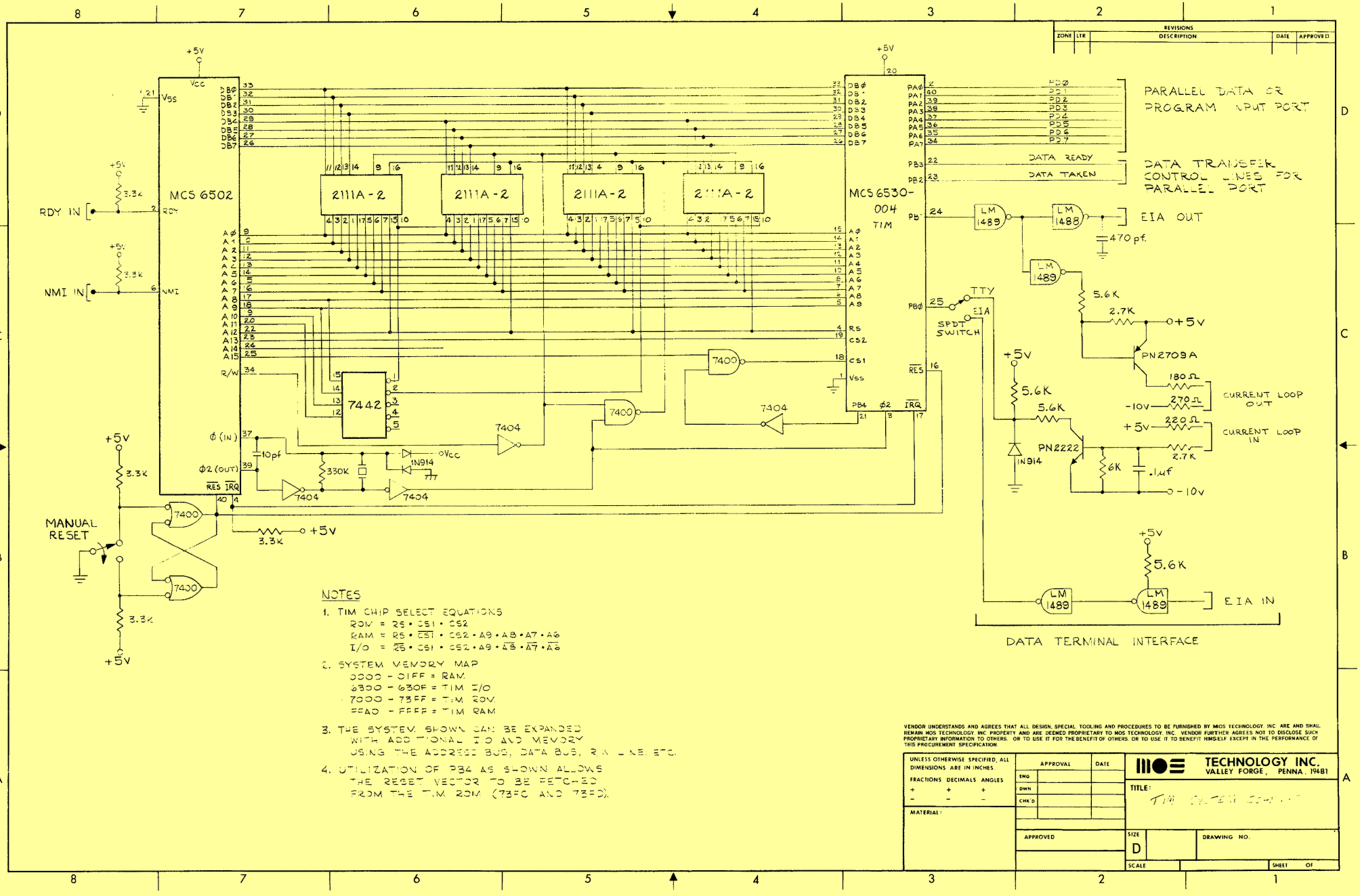
The TIM unit comes with:

1 - MCS6530-004 Multi-function Chip
1 - TIM Users Manual
1 - TIM System Schematic



* Note that the TIM as sold consists only of the MCS6530-004 component accompanied by supporting information to build this system

TYPICAL MINIMUM CONFIGURATION
FOR "TIM" SYSTEM



NOTES

1. TIM CHIP SELECT EQUATIONS
   ROM = RS • CS1 • CS2
   RAM = RS • $\overline{CS1}$ • CS2 • A9 • A8 • A7 • A6
   I/O = $\overline{RS}$ • CS1 • CS2 • A9 • $\overline{A8}$ • $\overline{A7}$ • $\overline{A6}$
2. SYSTEM MEMORY MAP
   0000 - 01FF = RAM
   6300 - 630F = TIM I/O
   7000 - 73FF = TIM ROM
   FFA0 - FFFF = TIM RAM
3. THE SYSTEM SHOWN CAN BE EXPANDED WITH ADDITIONAL I/O AND MEMORY USING THE ADDRESS BUS, DATA BUS, R/W LINE, ETC.
4. UTILIZATION OF PB4 AS SHOWN ALLOWS THE RESET VECTOR TO BE FETCHED FROM THE TIM ROM (73FC AND 73FD).

UNLESS OTHERWISE SPECIFIED, ALL DIMENSIONS ARE IN INCHES.
FRACTIONS DECIMALS ANGLES
+ + +
− − −

MATERIAL:

APPROVAL	DATE
ENG	
DWN	
CHK'D	
APPROVED	

MOS TECHNOLOGY INC.
VALLEY FORGE, PENNA. 19481

TITLE: TIM [illegible]

SIZE D
DRAWING NO.
SCALE
SHEET OF

MDT 650...THE MICROCOMPUTER DEVELOPMENT TERMINAL

The MDT650 is a high level development tool for modeling new designs. New flexible modular techniques of system verification are provided the user prior to a "hard" finalized design commitment. Fully programmable user control is among the many outstanding features. Interactive design allows the user to assemble programs, debug software with on-line editing capability and, when the design is considered correct, program PROMS. The MDT can be interfaced through the integral keyboard/display or via a port for TTY or higher speed peripherals. The standard resident assembler, in conjunction with systemized option cards for interactive debugging, allow the user to engage this system as a total development tool for preproduction and final production systems.

GENERAL SYSTEM DESCRIPTION

The MDT650 series microcomputer development terminal evaluates and debugs the user's programs and system hardware. The unit can be configured to a wide range of design applications. Considerable development time and money may be saved through the unit's unique ability to allow user-system emulation. The MDT650 provides the user a completely separate processor and bus structure to configure his application without regard to memory or executive I/O functions. This eliminates executive overhead time during real time execution...particularly important for interrupt routines.

Two MCS650X series microprocessors are used to control all system functions. Interaction with the MDT650 is normally with the integral keyboard/display. A TTY or other terminal device can also be used. Expandable port configurations are TTL compatible.

The standard MDT650 system allows the user to assign up to 65K of memory as desired (with independent address and data bases). The ROM resident system monitor includes all necessary functions for program loading, debugging, and execution. A resident assembler may be used to assemble machine instructions. Interpretation of machine codes is language to original OP codes, labels and mnemonics. A resident editor provides source language editing capability.

HARDWARE FEATURES

Integral Keyboard with separate function keys for control.

Built-in 32 character display.

Terminal inputs for variety of terminals. Selectable BAUD rates of 110, 150, 300, 600, 1200, 2400, 4800 or 9600 are also provided.

Two address traps are provided to halt the user processor on: any address; instruction (operand fetch); read cycle or write cycle

Two scope syncs: address trap sync and instruction fetch (operand)cycle sync.

Single instruction mode with firmware enhancement.

HARDWARE FEATURES CONTINUED

Trace stack memory for storing the last 128 machine cycles. (Visually displayed using the disassembler firmware).

Seven board positions provided for user memory, bus light display, I/O or user options such as custom wire wrap boards.

Control firmware for the assembler, disassembler, and test editor that is independent of the user's 65K memory limit.

SOFTWARE SUPPORT

The MDT650 software consists basically of three straightforward and highly useful programs: the assembler, the disassembler and the text editor.

The MDT assembler is upwards compatible with the MCS650X Cross-Assembler. Features include:

- Can assemble from source tape or user RAM... eliminates having to feed the source tape through the reader two times.
- Six character labels and symbols.
- Free-form entry of source statements.
- Symbol table output.
- Error flags on listing.
- Assembled program is user memory ready for execution.
- Paper tape output option - compatible with loader

The MDT disassembler commands include:

Note: For maximum user control, most of the following commands have corresponding function keys provided on the integral keyboard. Additionally, the MDT has function keys to show register, program counter, accumulator, index and status registers, stack pointer, and either trap address; also provided are function keys to alter individual registers...display shows what register is being altered, and a selectable enable/disable switch for TTY printing.

- Display address. 8 bytes are displayed and cursor is positioned at start of first byte to allow it to be altered. If you want to alter, just start typing. The display is automatically changed after each two input characters (one byte of memory) so multiple alters are easily made. (Function keys provide the capability of displaying forward by one or eight characters.)
- Load interface file (symbols and code).
- Go to address. Execute one instruction and automatically disassemble.
- Run. Executes user program.

SOFTWARE SUPPORT CONTINUED

- Trap address and mode. Sets appropriate instruction.
- Forward one instruction from current location in stack.
- Backward one instruction.
- Show last/next cycle. Address, label at this address (if any), data, and decoded flags are displayed.

The MDT test editor program saves considerable money and frustration compared with trying to edit a paper tape by hand. The following capabilities are provided:

- Loads text buffer
- Insert or delete line
- Insert or delete character at current position in line
- Forward/backward one character
- Will step forward or backward to next blank (Very useful for jumping from label field to OP Code field, etc.)
- Goes to top or bottom of text
- Indexes up or down one line
- Search/find capability on first field
- Provides output text to printer/punch

STANDARD EQUIPMENT (BASE SYSTEM)

Dual Micro Processor Module
RAM Memory Module
Program trace and address trap board
  I/O board for Keyboard, Display and
  Peripherals
Resident Monitor ROM Module
Chassis with 14 Board Slots
Power Supplies
Finished Cabinet
Keyboard and Display
System Monitor
Assembler
Text Editor
User's Manual
MCS650 Assembly Language Programming Manual
MCS650 Assembly Language Reference Card

OPTIONS

PROM programmer available for 82S115, 2708 or 1702A.
Wire-wrap boards for custom designs.
Extender board module

Light display board to display address and data busses
4K RAM board
8K PROM board
2K RAM/4K PROM board
I/O board...allows interfacing system to various peripheral devices and terminals
Write protect available on user RAM
High speed ports for printer, card readers, etc. (and associated cables).
Floppy disc interface will be available at a later date
ICE (In-Circuit Emulator)
Source Listing

AVAILABILITY

The base system will be available in second quarter, 1976 with the various Hardware options becoming available in second and third quarters, 1976.

COST

$3,950.00 (Base System). Cost of options will be available at a later date.

PRECAUTIONARY HANDLING PROCEDURES

FOR "MOS" TYPE PRODUCTS

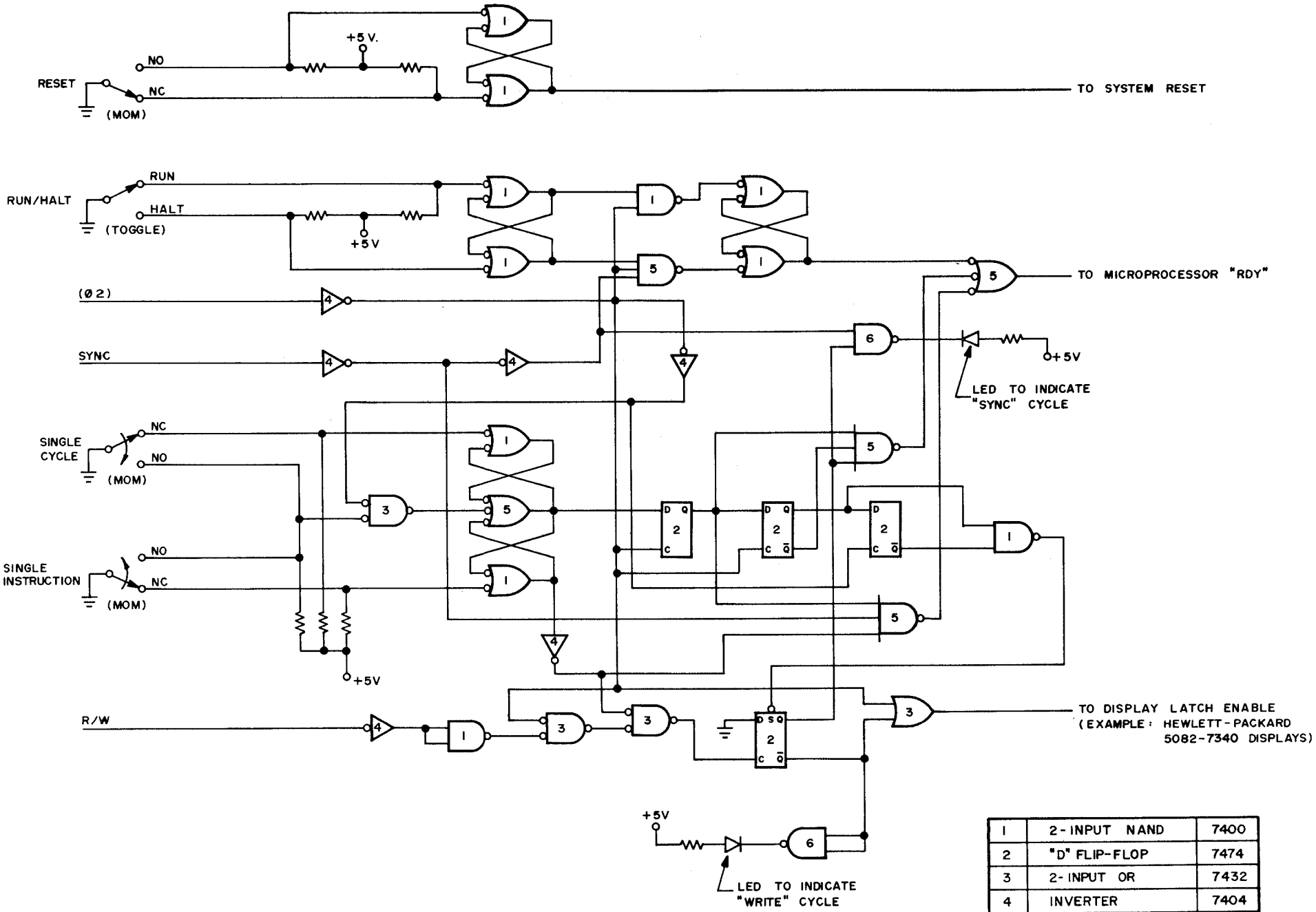
The MCS6500 Product Line has been designed with protective circuitry to guard against static charge damage on the inputs. However, normal precautions should be taken whenever possible to prevent exposure to environments of potential static charge. The following guidelines are recommended in handling the "MOS" type products and should be used whenever possible:

* Keep devices in the conductive shipping carrier until used.

* Perform work involving the "MOS" devices on a conductive surface where possible.

* In board assembly, place the "MOS" devices on the boards as late in the assembly cycle as possible. For low volume applications we recommend usage of a plug-in socket for the devices.

* Do not place the "MOS" device into position with power on. Always power up after the device is in place in the board assembly.



1	2-INPUT NAND	7400
2	"D" FLIP-FLOP	7474
3	2-INPUT OR	7432
4	INVERTER	7404
5	3-INPUT NAND	7410
6	2-INPUT NAND OPEN COLLECTOR	7403

SINGLE CYCLE / SINGLE INSTRUCTION

CONTROL LOGIC

ORDER FORM

COMPONENTS	QUANTITY	PRICE
MCS6501 @ $20.00	_____	_____
MCS6502 @ $25.00	_____	_____
MPS6503 @ $20.00	_____	_____
MPS6504 @ $20.00	_____	_____
MPS6505 @ $20.00	_____	_____
MCS6502A @ $37.50	_____	_____
MPS6503A @ $30.00	_____	_____
MPS6504A @ $30.00	_____	_____
MPS6505A @ $30.00	_____	_____
MCS6530-004 @ $30.00 (Includes TIM Manual)	_____	_____
MCS6530-005 @ $18.00	_____	_____
KIM-1 System @ $245.00 * (Includes all documentation)	_____	_____

*Postage and Handling $4.50 - Domestic
$20.00 - International

DOCUMENTATION		
Hardware Manual @ $5.00 (Pub. No. 6500-10)	_____	_____
Programming Manual @ $5.00 (Pub. No. 6500-50)	_____	_____
Cross Assembler Manual @ $4.00 (Preliminary)	_____	_____
TIM Manual @ $4.00 (If purchased separately from the MCS6530-004)	_____	_____

TOTAL _____

NAME _____

COMPANY _____

ADDRESS _____

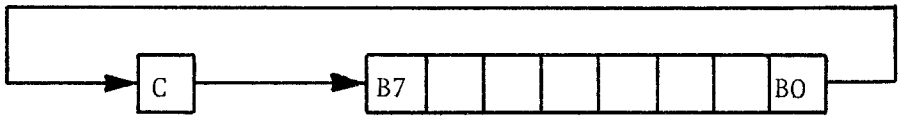
_____

Purchase Order Encl. _____

Check Enclosed _____

MCS650X FAMILY CIRCUIT MODIFICATIONS

1. Since introduction of the MCS650X Family numerous customers have requested the addition of the ROR Instruction. This instruction is now being added to all MCS650X processors. The addressing modes will be Absolute (6 cycles, 3 bytes); Zero Page (5, 2); Accumulator (2, 1); Zero Page, X (6, 2); Absolute, X (7, 3). The implementation of ROR involves shifting all addressed locations one bit to the right with the carry bit shifted into Bit location 7 and Bit location 0 shifted into the carry position.



2. An additional modification is being made to the Branch, Ready circuitry. The present MCS650X processors do not execute the branch instructions correctly when the ready signal is used in the Single Cycle Mode if the low order effective address is FF without crossing page boundaries. For clarification the following program is executed, in both Single Cycle Mode and Single Instruction Mode.

Sample Program

Memory	Contents	
F5	18	CLC
F6	90	BCC
F7	07	offset
F8	A9	LDAIMM

Sample Program Execution

Single Cycle

ABH	ABL	DB	SYNC	
XX	F5	18	1	CLC
XX	F6	90	0	
XX	F6	90	1	BCC
XX	F7	07	0	
XX	F8	09	0	
XX	FF(XX+1,FF)		1	

Single Instruction

ABH	ABL	DB	SYNC
XX	F5	18	1
XX	F6	90	1
XX	FF	(XXFF)	1

Note that in the Single Cycle Mode the ADH of the branch destination is incremented while in the Single Instruction Mode the branch is properly executed. The only time this occurs is when the ADL of the branch destination is FF and then only during Single Cycle with no page crossing; hence, the probability of this occurring in normal application is remote.

Availability - MCS650X microprocessors incorporating the above changes will be available in sample quantities in April with production deliveries beginning in May. Pricing for these versions of the microprocessor will be identical to the product currently being shipped.

CLOCK GENERATOR INFORMATION

Initial characterization of the MCS650X clock generator circuit has provided us with sufficient information to update the clock generator information found in our manuals and data sheets. The following discussion provides the user with information needed to obtain best performance for the time base generation scheme chosen.

Generally one would consider the following when designing systems with the MCS650X.

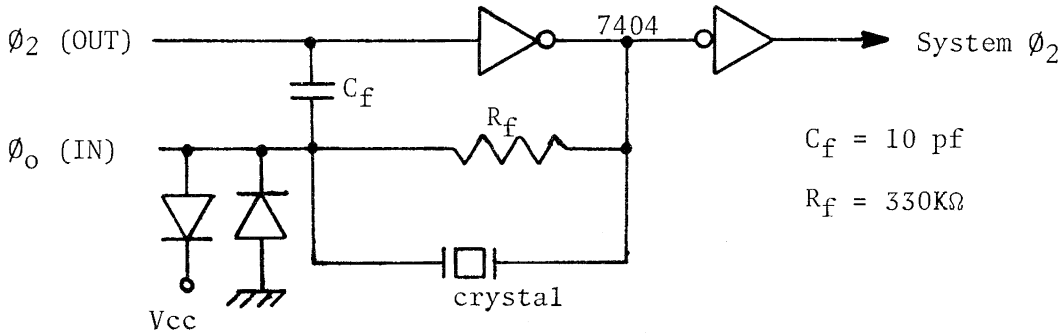
1. Which clock scheme is the best?

   There is no one answer, however depending on the system there is one best way

   A. TTL generated clock - drive $\emptyset_o$ (IN) with a TTL level clock, it doesn't require a high level clock; merely V_{OL} = .4V, V_{OH} = 2.4V. Buffer $\emptyset_2$ (OUT) for use as system $\emptyset_2$ clock. This scheme allows maximum control of all clock variables (i.e. symmetry, frequency, frequency variation from system to system).

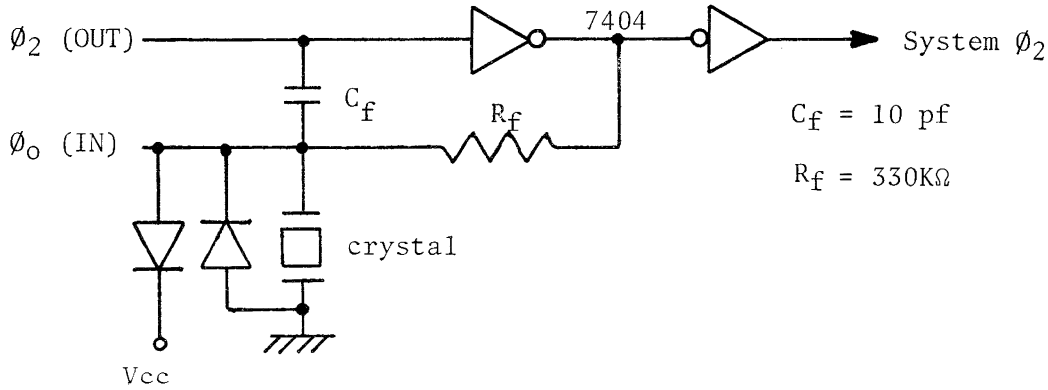
   In the following discussion, reference will be made to $\emptyset_o$ (IN) and $\emptyset_2$ (OUT). The applicable pin numbers on the various MCS650X processors are found in the manuals or data sheets. The diodes (IN914's) are for the purposes of clamping the clock swings near ground and near V_{DD} and may not be required in all crystal applications.

   B. Series Mode Crystal Controlled



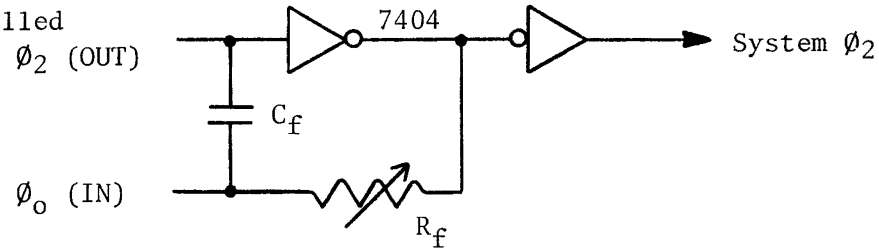
   This scheme allows for crystal controlled operation which is least sensitive to crystal parameters and feedback circuit variables. Because the crystal is in the feedback path and not shunting as the parallel mode crystal controlled scheme, the serial mode is most reliable from a start-up standpoint.

   C. Parallel Mode Crystal Controlled



This scheme should be used when the symmetry is more desirable than the Serial Mode crystal controlled scheme symmetry. This scheme is most sensitive to feedback parameters as related to start-up. By varying the feedback resistor an appropriate combination can be found for the crystal chosen.

D. RC Controlled



This scheme is recommended for those systems which do not require symmetry control, frequency variation control from system to system without manual adjustment, and systems where noise has been minimized.

Because of the ease of use of this scheme it is recommended for those systems which are in development, used as a microprocessor learning vehicle or in general systems in which noise on the clock circuit has been carefully handled (i.e. clean supply to microprocessor and clock buffers, isolate $\emptyset_2$ (OUT) and $\emptyset_2$ (IN) from stray system noise).

For frequency of operation around 1 MH_z a value of 10K to 50K should be used with C_f = 10 pf. To decrease noise sensitivity increase C_f and decrease R_f. Also a shunting capacitor to ground from $\emptyset_2$ (OUT) the value of which should be the same range as C_f, will help to decrease sensitivity to noise in the system.

2. What is the maximum clock loading?

One standard TTL Load and 30 pf

If the clock outputs ($\emptyset_1$ (OUT) and $\emptyset_2$ (OUT)) are loaded, there is the possibility of causing overlap, but this has no effect on the internal clocks on the microprocessor. The system designer should therefore be careful if non-overlapping system clocks are necessary such that the microprocessor can function properly with overlapped clocks but system problems can develop.

3. How can clock ringing be prevented?

A. Eliminate noise from $\emptyset_o$ (IN). The clock generator on the MCS650X will react to frequencies as high as 20 MH_z, therefore it will also respond to noise.

B. Be sure the negative feedback (R_f) occurs after the positive feedback (C_f). This is most easily accomplished by tapping off the negative feedback after the positive feedback (note the inverter delay in the RC controlled schematic).

4. What are the typical clock widths and separations for TTL clock levels in? See graph #1.

5. What are typical frequencies for various RC combinations?

The RC values listed below provide the approximate operating frequencies listed. It is recommended that variable resistor pots be used if accuracy in the value of the operating frequency is required.

Typical Frequency	R_f	C_f
.5 MH_z	42K	10 pf
1.0 MH_z	17K	10 pf
2.0 MH_z*	6K	10 pf

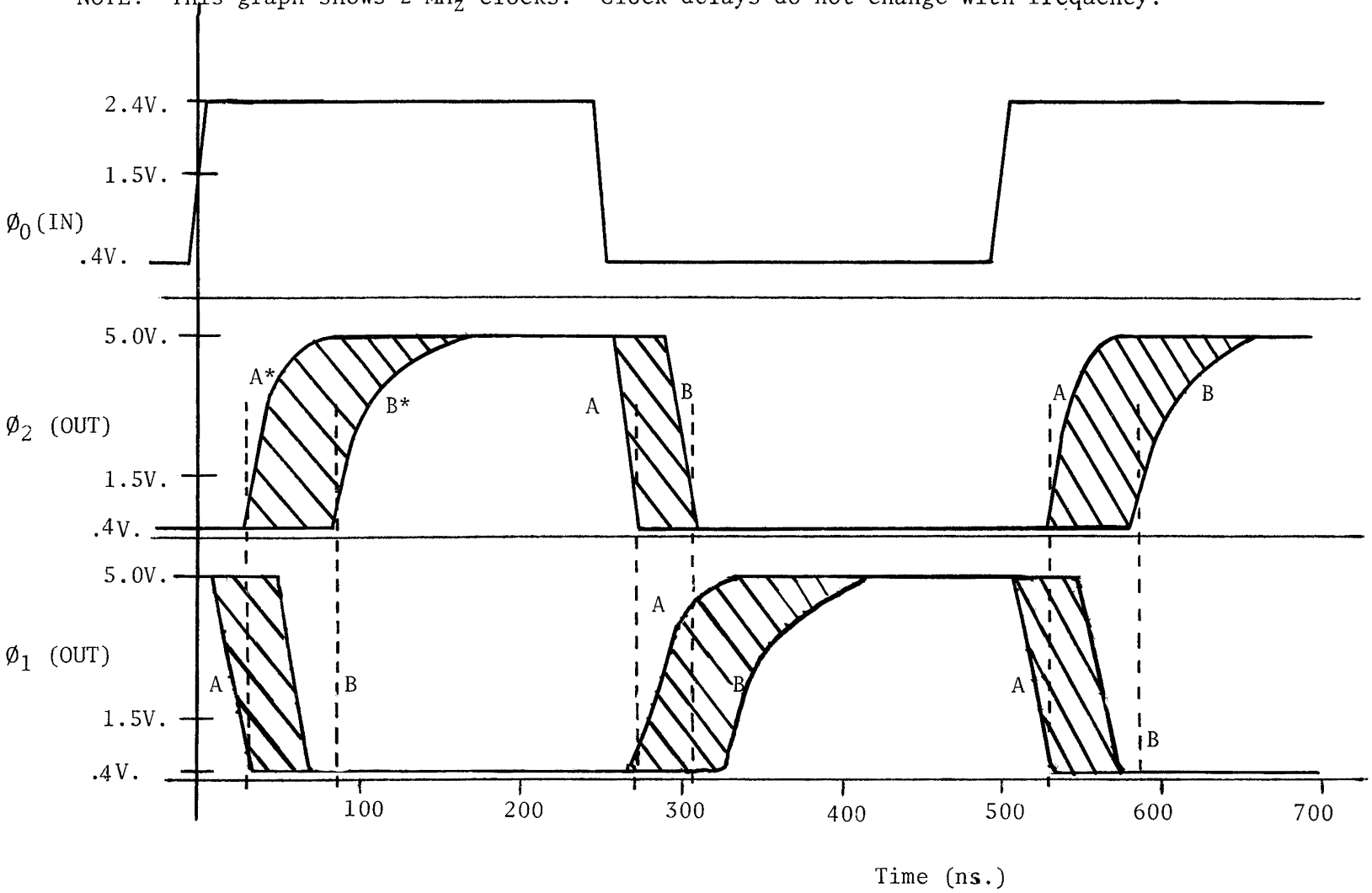
*Applies to product guaranteed at 2 MH_z operation.

NOTE: It should be understood that for maximum confidence in control of symmetry of the system clocks, it is recommended the TTL level $\emptyset_o$ (IN) be used. This can be generated from, for example a divide down from a high frequency crystal. In any case maximum pulse width control is formed with these inputs to $\emptyset_o$ (IN) in which the user has maximum control of the edges.

GRAPH #1

CLOCK PHASE RELATIONSHIPS

NOTE: This graph shows 2 MH_Z clocks. Clock delays do not change with frequency.



*Typical process related values. Edges marked "A" would correspond to part "A"
Edges marked "B" would correspond to part "B"

SALES OFFICES AND REPRESENTATIVES FOR MOS TECHNOLOGY, INC.

EASTERN REGION

REGIONAL DIRECTOR

William Whitehead
MOS TECHNOLOGY, INC.
410 Jericho Turnpike, Suite 312
Jericho, N.Y. 11753
516-822-4240

NEW ENGLAND STATES

The Orion Group
P. O. Box 248
Mont Vernon, N.H. 03057
617-890-3777

SOUTHERN N.J., EASTERN PA, DELA.

Rivco, Inc.
P. O. Box 338
King of Prussia, Pa. 19406
215-265-5211

MD., WASH, D.C., VA. W. VA.

Bernard White & Company
7 Church Lane
Baltimore, Maryland 21208
301-484-5400

NORTHERN N.J., NEW YORK CITY, WESTCHESTER COUNTY, N.Y. FAIRFIELD COUNTY, CONN.

Falk-Baker Associates
382 Franklin Ave.
Nutley, N. J. 07110
201-661-2430

ALA, FLA, GA, TENN., N. C.,S. C., MISS.

Currie, Peak & Frazier, Inc.
6880 Lake Ellenor Drive
Suite 237
Orlando, Florida 32809
305-855- 0843

NEW YORK, EXCEPT NEW YORK CITY AND WESTCHESTER COUNTY

Labtronics, Inc.
166 Pickard Bldg
Syracuse, N.Y. 13211
315-454-9314

CENTRAL REGION

OHIO, KENTUCKY, WESTERN PA.

McShane, Inc.
P. O. Box 523
Medina, Ohio 44256
216-725-4568

TEXAS, OKLAHOMA, ARKANSAS, LOUISIANA

Norvell Associates, Inc.
P. O. Box 20279
10210 Monroe Drive
Dallas, Texas 75220
214-350-6771

WESTERN REGION

REGIONAL DIRECTOR

Jack Turk
MOS TECHNOLOGY, INC.
2172 Dupont Drive
Patio Bldg.
Irvine, CA. 92715
714-833-1600

APPLICATIONS MANAGER

Petr Sehnal
MOS TECHNOLOGY, INC.
22300 Foothill Blvd.
Hayward, CA. 94541
415-881-8080

ARIZONA, NEW MEXICO

Toward Engineering Assoc., Inc.
P. O. Box 15268
Phoenix, Arizona 85018
602-955-3193

COLORADO, UTAH, WYOMING MONTANA (EAST OF GREAT FALLS)

R. G. Enterprises, Inc.
1898 So. Flatiron Ct.
Boulder, CO 80301
303-447-9211

WASHINGTON, OREGON, IDAHO, MONTANA (WEST OF GREAT FALLS)

J. A. Tudor & Associates, Inc.
P. O. Box 21947
Seattle, Washington 98111
206-682-7444

NORTHERN CALIFORNIA, NORTHERN NEVADA

W. W. Posey Company, Inc.
895 Sherwood Ave.
Los Altos, CA. 94022
415-948-7771

SALES OFFICE AND FOREIGN REPRESENTATIVES

EUROPEAN SALES DIRECTOR

Mr. Gerold G. Wiese
MOS TECHNOLOGY, INC.
6382 Friedrichstdorf 2 (F. Frankfurt)
Otto-Hahn-Strasse 40
W. Germany
(06175) 1510, 1519

HONG KONG AND TAIWAN

Mr. Lyle Ronalds
SOLID STATE INTERNATIONAL
Daily House Suite 48
35-39 Haiphong Road
Kowloon, Hong Kong
China

JAPAN AND ITS TERRITORIES

Nihon Teksel Co., Ltd.
13-14 Sakuragaoka-Machi
Shibuya-Ku, Tokyo
Japan
(03) 461-5121

INDIA

Mr. Dinish C. Gupta
Superior Electronics
E13 Dattaguru Society
Deonarpada Road
Bombay 400 088 India

MOS TECHNOLOGY, INC.
NORRISTOWN, PA. 19401

MICROCOMPUTERS

MCS6500 MICROPROCESSOR SOFTWARE SUPPORT

MOS TECHNOLOGY'S support software is now available on United Computing Systems time-sharing service. The package available provides online support to assist the microcomputer applications design engineer or programmer in program development for the MCS650X microcomputer family.

TO USE MOS TECHNOLOGY SUPPORT SOFTWARE:

1. Contact your local USC sales representative and request MOS TECHNOLOGY'S MCS650X Software System under user catalog number M490. Also request the UCS System Guide and the UNIEDIT manuals.
2. Order your copy of the MCS6500 Microprocessor Hardware, Programming, Simulator, And Cross Assembler manuals from:
   MOS Technology Inc., 950 Rittenhouse Rd., Norristown, Pa. 19401
3. Dial the appropriate telephone number supplied by your USC sales representative, sign on with your terminal, and begin entering your MCS650X microprocessor program.

THE SOFTWARE SUPPORT PACKAGE CONSISTS OF:

–MOS/*** - A text file containing the latest bulletins regarding MOS TECHNOLOGY Microprocessor Software.

–ASM/*** - An interactive program which builds the job control language required to submit your source code to ASM650X.

ASM650X MCS650X Cross Assembler: the Cross Assembler is a program which translates a mnemonic or symbolic form of a computer program to machine language.

–SIM/*** - An interactive program which builds the job control language required to submit your simulator command file to SIM650X.

SIM650X - MCS650X Simulator. The simulator uses the command file to simulate execution of the machine language instructions created by the cross assembler in the MCS650X microprocessor.

–DMP/*** - ROM dump program. This program creates an output file of machine language instructions in a format suitable for MOS microcomputer loader programs.

The sample program shown in this brochure uses the UCS time-sharing system to give the user an overview of the procedure to be followed for using MOS TECHNOLOGY'S support software.

In brief the procedure to be followed is:

1. Create a source file using the time-sharing editor and save the file.
2. Submit the source file to the Cross Assembler by answering the questions asked by -ASM/***.
3. When the Cross Assembler run is completed list the output file to obtain a listing of the assembled program.
4. Create a file of simulator commands using the time-sharing editor and save the file.
5. Submit the simulator command file and the machine language file to the simulator by answering the questions asked by -SIM/***.
6. When the simulator run is completed list the output file to obtain the results of the program simulation.
7. Obtain a ROM dump object tape by answering the questions asked by -DMP/***.

1. CREATE A SOURCE FILE.

```
|>p|>`p|pT63

UCS  11/19/75. 09.10.41.  I150
USER NUMBER:  M490010,EXAMPLE

 GENERAL:
 MOS TECHNOLOGY 650X MICROPROCESSOR SOFTWARE.
 FOR THE LATEST INFORMATION TYPE  -MOS/***

 MESSAGE(S) COMPLETE.

     0.013 /      0.038 /          9
READY - FOR!
-MOS/***

 11/19/75. 09.11.22.
PROGRAM   MOS

LAST UPDATED ON 11/19/75
.
BULLETINS REGARDING THE MOS TECHNOLOGY MICROPROCESSOR
SOFTWARE WILL APPEAR FROM TIME TO TIME IN THIS MANNER.
.
TO RUN THE 650X CROSS ASSEMBLER YOU MUST FIRST CREATE A
SOURCE FILE. THEN ENTER -ASM/*** TO SUBMIT YOUR SOURCE FILE
FOR BACKGROUND BATCH EXECUTION.
.
TO RUN THE 650X SIMULATOR YOU MUST FIRST CREATE A SIMULATOR
COMMAND FILE AND A CROSS ASSEMBLER INTERFACE FILE. THEN TYPE -
-SIM/*** TO SUBMIT YOUR  COMMAND FILE FOR SIMULATION.
.
THE 650X ROM DUMP PROGRAM WILL CREATE A REFORMATED FILE
SUITABLE FOR INPUT TO THE MOS MICCROCOMPUTER LOADER PROGRAMS.
YOU MUST HAVE CREATED AN INTERFACE FILE WITH THE CROSS
ASSEMBLER. TO RUN THE DUMP PROGRAM ENTER -DMP650X/***
.
THANK YOU.....MOS TECHNOLOGY
RUN COMPLETE.
NEW,SAMP4
READY - FOR!
AUT
00100 .PAGE ´MULTIPLE BYTE ADD´
00110 ;ADDITION OF TWO MULTIPLE PRECISION NUMBERS (BCD)
00150  *=0   ALLOCATE A DATA AREA IN FIRST PAGE OF MACHINE
00170 ADDR *=*+1
00190 NB=8
00200 PP *=*+NB
00210 Q *=*+NB
00220 RES *=*+NB
00270 MAIN LDX #$8F BEGIN MAIN ROUTINE TO TEST SUB. BCD.
00280  TXS   INITIALIZE STACK POINTER
00290  LDX #PP
00300  STX ADDR
00310  JSR BCD
00320  NOP
00330  JMP *-1   END OF MAIN PGM
00360  *=100      BEGIN SUBROUTINE
00370 BCD LDY #NB
00380  LDX ADDR  LOADS DATA ADDRESS
00390  CLC
00400  SED
00410 NEXT LDA NB-1,X
00420  ADC 2*NB-1,X
00430  STA 3*NB-1,X
00440  DEX
00450  DEY
00460  BNE NEXT   END OF LOOP
00470  CLD
00480  RTS
00490 ABCDEFGH NOP   THIS IS AN INTENTIONAL ERROR.
00500 .END
00510 *DEL*
SAVE
READY.
```

Enter proper response so that computer can determine your terminal's speed.

For 10 CPS enter ?63
For 15 CPS enter 863
For 30 CPS enter T63

Enter your user number and password to log on to UCS system.

Indicates FORTRAN system is ready. (FORTRAN is automatically assigned.)

Enter -MOS/*** to obtain latest bulletins.

Indicates the end of the bulletin.

Create a new file with file name "SAMP4".

Auto line number assignment.

Assembler directive to advance listing to top of page and title the page "MULTIPLE BYTE ADD".

Semicolon indicates the start of a comment field.

*= assembler directive sets the program counter.

Sets NB equal to 8.

Reserves 8 bytes of memory for the label "PP".

Start of program labeled "MAIN"
Note that there is only one space between a line number and a label. There are two or more spaces between a line number and an instruction. Comments may begin one space after the operand.

.END assembler directive defines the end of the source program.

Hitting the "ESC" key ends the auto line number assignment. The system replies "*DEL*".

SAVE is the command to save the new file just created.

2. SUBMIT TO CROSS ASSEMBLER.

```
-ASM/***

MOS TECHNOLOGY 650X CROSS ASSEMBLER SUBMITTOR

DO YOU WANT INSTRUCTIONS (YES OR NO) -- ? NO
ENTER USERNUM,PASSWORD, AND PID (IF NEEDED) --  ? M490010,EXAMPLE
DO YOU WANT TO CHANGE THE PRIORITY --  ? NO

ENTER SOURCE     FILE NAME --  ? SAMP4

SAVE  OUTPUT     FILE (YES OR NO) --  ? YES
ENTER OUTPUT     FILE NAME --  ? OUT4

SAVE  INTERFACE  FILE (YES OR NO) --  ? YES
ENTER INTERFACE  FILE NAME --  ? INT4

SAVE  ERROR      FILE (YES OR NO) --  ? YES
ENTER ERROR      FILE NAME --  ? ERR4

SAVE  DAYFILE    FILE (YES OR NO) --  ? YES
ENTER DAYFILE    FILE NAME --  ? DAY4

ENTER CONTROL    FILE NAME --  ? CON4

TO RUN ASSEMBLER TYPE --
OLD,CON4
RJE    (OR RBE)


 STOP.
OLD,CON4
READY - EXE!
RJE

 11/19/75. 09.15.45.
PROGRAM   CON4

 RJE COMPLETE,ID =  RJEDZQM
```

-ASM/*** invokes the cross assembler submittor software.

SOURCE file is the file containing the source code to be assembled.

OUTPUT file will contain the assembler listing.

INTERFACE file will contain the object code, line number and label information required by the simulator.

ERROR file will contain a listing of any errors that occur during the assembly.

DAY file is a history of steps taken by the UCS system in running your job.

CONTROL file is the file of JCL built by -ASM/*** to run your assembly.

Submits assembly job to the UCS system.

Indicates that the job has been submitted under the job name "RJEDZQM".

3. LIST OUTPUT FILE

```
OLD,OUT4
READY - EXE!
LIS

 11/19/75. 09.18.14.
PROGRAM   OUT4

+    MULTIPLE BYTE ADD                                          PAGE    1
0LINE  LOC    CODE       SOURCE
   110                  ;ADDITION OF TWO MULTIPLE PRECISION NUMBERS (BCD)
   150  0000             *=0   ALLOCATE A DATA AREA IN FIRST PAGE OF MACHINE
   170  0000            ADDR *=*+1
   190                  NB=8
   200  0001            PP *=*+NB
   210  0009            Q *=*+NB
   220  0011            RES *=*+NB
   270  0019  A2 8F     MAIN LDX #$8F BEGIN MAIN ROUTINE TO TEST SUB. BCD.
   280  001B  9A         TXS   INITIALIZE STACK POINTER
   290  001C  A2 01      LDX #PP
   300  001E  86 00      STX ADDR
   310  0020  20 64 00   JSR BCD
   320  0023  EA         NOP
   330  0024  4C 23 00   JMP *-1   END OF MAIN PGM
   360  0027             *=100      BEGIN SUBROUTINE
   370  0064  A0 08     BCD LDY #NB
   380  0066  A6 00      LDX ADDR  LOADS DATA ADDRESS
   390  0068  18         CLC
   400  0069  F8         SED
   410  006A  B5 07     NEXT LDA NB-1,X
   420  006C  75 0F      ADC 2*NB-1,X
   430  006E  95 17      STA 3*NB-1,X
   440  0070  CA         DEX
   450  0071  88         DEY
   460  0072  D0 F6      BNE NEXT   END OF LOOP
   470  0074  D8         CLD
   480  0075  60         RTS
   490  0076  EA EA EA  ABCDEFGH NOP   THIS IS AN INTENTIONAL ERROR.
 *****  ERROR ** LABEL GREATER THAN SIX CHARACTERS - NEAR COLUMN    1
   500                  .END

 END OF MOS/TECHNOLOGY 650X ASSEMBLY VERSION 4
 NUMBER OF ERRORS =   1,  NUMBER OF WARNINGS =   0
1          SYMBOL TABLE

   SYMBOL    VALUE  LINE DEFINED        CROSS-REFERENCES

   ADDR      0000        170   300  380
   BCD       0064        370   310
   MAIN      0019        270  ****
   NB        0008        190   200  210  220  370  410  420  430
   NEXT      006A        410   460
   PP        0001        200   290
   Q         0009        210  ****
   RES       0011        220  ****
RUN COMPLETE.
```

Terminal input to list the output file "OUT4".

Title created by .PAGE assembler directive.

Program counter. (Hexadecimal)

Hexadecimal instruction, data, or value.

Program counter set to hexadecimal 64 by assembler directive *=100.

Error line will also appear in the ERROR file.

The version number is changed as improvements are made to the Cross Assembler.

Note: For more detailed information refer to the MCS6500 Microprocessor Programming and Cross Assembler manuals.

4. CREATE SIMULATOR COMMANDS

```
NEW,ECSAMP1
READY - FOR!
AUTO
00100 SM 1 1 2 3 4 5 6 7 8
00110 SM 9 8 7 6 5 4 3 2 1
00120 DUMP 1 $18
00130 TRACE 0 $FFFF
00140 DO MAIN NEXT 3 .TIMES
00150 DUMP 1 $18
00160 EXIT
00170 *DEL*
SAVE
READY.
```

Create simulator command file called "ECSAMP1".

Starting at location 1 set consecutive memory locations to the specified values.

Dump the contents of memory from decimal 1 to hexadecimal 18.

Trace every instruction executed.

Begin simulated execution at label "MAIN" and continue until instruction at label "NEXT" has been executed 3 times.

EXIT terminates simulator run.

5. SUBMIT TO SIMULATOR

```
-SIM/***

MOS TECHNOLOGY 650X SIMULATOR SUBMITTOR

DO YOU WANT INSTRUCTIONS (YES OR NO) -- ? NO
ENTER USERNUM,PASSWORD, AND PID (IF NEEDED) --  ? M490010,EXAMPLE
DO YOU WANT TO CHANGE THE PRIORITY --  ? NO

ENTER COMMAND     FILE NAME --  ? ECSAMP1

ENTER INTERFACE   FILE NAME --  ? INT4

SAVE  OUTPUT      FILE (YES OR NO) --  ? YES
ENTER OUTPUT      FILE NAME --  ? EOUT4

SAVE  DAYFILE     FILE (YES OR NO) --  ? YES
ENTER DAYFILE     FILE NAME --  ? EDAY4

ENTER CONTROL     FILE NAME --  ? ECON4

TO RUN SIMULATOR TYPE --
OLD,ECON4
RJE     (OR RBE)


 STOP.
OLD,ECON4
READY - EXE!
RJE

 11/19/75. 09.23.50.
PROGRAM   ECON4

 RJE COMPLETE,ID =  RJEDZRY
```

-SIM/*** invokes the simulator submittor software.

COMMAND file is the file containing the simulator commands.

INTERFACE file is the interface file created by the cross assembler.

6. LIST SIMULATOR OUTPUT

```
OLD,EOUT4
READY - FOR!
LIST
```

Terminal commands required to list the Simulator output file.

```
 11/19/75. 09.26.05.
PROGRAM   EOUT4

1++++++ MOS TECHNOLOGY 650X MICROPROCESSOR SIMULATOR +++++


 00100 SM 1 1 2 3 4 5 6 7 8
 00110 SM 9 8 7 6 5 4 3 2 1
 00120 DUMP 1 $18
                    CONTENTS OF MEMORY LOCATION AT BASE ADDRESS PLUS........
   BASE ADDRESS    +0  +1  +2  +3  +4  +5  +6  +7  +8  +9  +A  +B  +C  +D  +E  +F
 DUMP ADDR=0000    00  01  02  03  04  05  06  07  08  08  07  06  05  04  03  02
 DUMP ADDR=0010    01  00  00  00  00  00  00  00  00  A2  8F  9A  A2  01  86  00
 00130 TRACE 0 $FFFF
 00140 DO MAIN NEXT 3 .TIMES

   IA  LABEL  OPCODE  A  S  X  Y  P  STATUS   PC   EA  EO  ICNT  TCNT  6501 TIME
 T0019 MAIN   LDX A2 00 00 8F 00 90 N  B     001B 001A 8F     1     2       0.
 T001B        TXS 9A 00 8F 8F 00 90 N  B     001C 001B 00     2     4       0.
 T001C        LDX A2 00 8F 01 00 10    B     001E 001D 01     3     6       0.
 T001E        STX 86 00 8F 01 00 10    B     0020 0000 01     4     9       0.
 T0020        JSR 20 00 8D 01 00 10    B     0064 0064 00     5    15       0.
 T0064 BCD    LDY A0 00 8D 01 08 10    B     0066 0065 08     6    17       0.
 T0066        LDX A6 00 8D 01 08 10    B     0068 0000 01     7    20       0.
 T0068        CLC 18 00 8D 01 08 10    B     0069 0068 00     8    22       0.
 T0069        SED F8 00 8D 01 08 18    BD    006A 0069 00     9    24       0.
 T006A NEXT   LDA B5 08 8D 01 08 18    BD    006C 0008 08    10    28       0.
 T006C        ADC 75 09 8D 01 08 18    BD    006E 0010 01    11    32       0.
 T006E        STA 95 09 8D 01 08 18    BD    0070 0018 09    12    36       0.
 T0070        DEX CA 09 8D 00 08 1A    BD Z  0071 0070 00    13    38       0.
 T0071        DEY 88 09 8D 00 07 18    BD    0072 0071 00    14    40       0.
 T0072        BNE D0 09 8D 00 07 18    BD    006A 006A 00    15    43       0.
 T006A NEXT   LDA B5 07 8D 00 07 18    BD    006C 0007 07    16    47       0.
 T006C        ADC 75 09 8D 00 07 18    BD    006E 000F 02    17    51       0.
 T006E        STA 95 09 8D 00 07 18    BD    0070 0017 09    18    55       0.
 T0070        DEX CA 09 8D FF 07 98 N  BD    0071 0070 00    19    57       0.
 T0071        DEY 88 09 8D FF 06 18    BD    0072 0071 00    20    59       0.
 T0072        BNE D0 09 8D FF 06 18    BD    006A 006A 00    21    62       0.
 EMUL MONITOR DETECTED A WARNING-PAGE ZERO WRAP
 T006A NEXT   LDA B5 06 8D FF 06 18    BD    006C 0006 06    22    66       0.
 +HILEV+ BREAKPOINT-NORMAL DO SEQUENCE END
 00150 DUMP 1 $18
                    CONTENTS OF MEMORY LOCATION AT BASE ADDRESS PLUS........
   BASE ADDRESS    +0  +1  +2  +3  +4  +5  +6  +7  +8  +9  +A  +B  +C  +D  +E  +F
 DUMP ADDR=0000    01  01  02  03  04  05  06  07  08  08  07  06  05  04  03  02
 DUMP ADDR=0010    01  00  00  00  00  00  00  09  09  A2  8F  9A  A2  01  86  00
 00160 EXIT
 STOP.
RUN COMPLETE.
```

Output generated as a result of the DUMP command.

Trace output generated during execution of the DO sequence.

A warning to the user that his program execution caused an index register to wrap around from hexadecimal FF to OO. This may not have been planned.

Indicates normal DO sequence termination.

Note: For more detailed information refer to the MCS6500 Simulator manual.

7. PUNCH OBJECT TAPE

```
-DMP/***

MOS TECHNOLOGY -- ROM DUMP

ENTER INTERFACE FILENAME ? INT4
ENTER OBJECT FILE NAME FOR OUTPUT --  ? OBJ4
OBJ4    CONTAINS OBJECT OUTPUT

 STOP.

     0.135 /     0.809 /        18
OLD,OBJ4
READY - EXE!
PUNCH


 ;0E0019A28F9AA2018600206400EA4C230004F8
 ;100064A008A60018F8B507750F9517CA88D0F607D6
 ;050074D860EAEAEA046F
 ;0000030003


BYE


CT=00:20
M490010   LOG OFF.   09.30.38.
```

—DMP/*** invokes the ROM dump program.

INTERFACE file is the file created by the cross assembler.

OBJECT file is the file name the object code is to be saved in.

Terminal commands required to list and punch the object tape.

Note: The paper tape punch should be turned on after the carriage return is entered.

Sign-off the system by entering "BYE"

MCS6500 MICROPROCESSOR LANGUAGE

INSTRUCTION SET

ADC	Add Memory to Accumulator with Carry
AND	"AND" Memory with Accumulator
ASL	Shift Left One Bit (Memory or Accumulator)
BCC	Branch on Carry Clear
BCS	Branch on Carry Set
BEQ	Branch on Result Zero
BIT	Test Bits in Memory with Accumulator
BMI	Branch on Result Minus
BNE	Branch on Result not Zero
BPL	Branch on Result Plus
BRK	Force Break
BVC	Branch on Overflow Clear
BVS	Branch on Overflow Set
CLC	Clear Carry Flag
CLD	Clear Decimal Mode
CLI	Clear Interrupt Disable Bit
CLV	Clear Overflow Flag
CMP	Compare Memory and Accumulator
CPX	Compare Memory and Index X
CPY	Compare Memory and Index Y
DEC	Decrement Memory by One
DEX	Decrement Index X by One
DEY	Decrement Index Y by One
EØR	"Exclusive-or" Memory with Accumulator
INC	Increment Memory by One
INX	Increment X by One
INY	Increment Y by One
JMP	Jump to New Location
JSR	Jump to New Location Saving Return Address
LDA	Load Accumulator with Memory
LDX	Load Index X with Memory
LDY	Load Index Y with Memory
LSR	Shift One Bit Right (Memory or Accumulator)
NØP	No Operation
ØRA	"OR" Memory with Accumulator
PHA	Push Accumulator on Stack
PHP	Push Processor Status on Stack
PLA	Pull Accumulator from Stack
PLP	Pull Processor Status from Stack
RØL	Rotate One Bit Left (Memory or Accumulator)
RTI	Return From Interrupt
RTS	Return From Subroutine
SBC	Subtract Memory from Accumulator with Borrow
SEC	Set Carry Flag
SED	Set Decimal Mode
SEI	Set Interrupt Disable Status
STA	Store Accumulator in Memory
STX	Store Index X in Memory
STY	Store Index Y in Memory
TAX	Transfer Accumulator to Index X
TAY	Transfer Accumulator to Index Y
TSX	Transfer Stack Pointer to Index X
TXA	Transfer Index X to Accumulator
TXS	Transfer Index X to Stack Pointer
TYA	Transfer Index Y to Accumulator

EXECUTION TIMES (IN CLOCK CYCLES)

	Accumulator	Immediate	Zero Page	Zero Page, X	Zero Page, Y	Absolute	Absolute, X	Absolute, Y	Implied	Relative	(Indirect, X)	(Indirect), Y	Absolute Indirect
ADC	.	2	3	4	.	4	4*	4*	.	.	6	5*	.
AND	.	2	3	4	.	4	4*	4*	.	.	6	5*	.
ASL	2	.	5	6	.	6	7	.	.	.	.	.	.
BCC	.	.	.	.	.	.	.	.	.	2**	.	.	.
BCS	.	.	.	.	.	.	.	.	.	2**	.	.	.
BEQ	.	.	.	.	.	.	.	.	.	2**	.	.	.
BIT	.	.	3	.	.	4	.	.	.	.	.	.	.
BMI	.	.	.	.	.	.	.	.	.	2**	.	.	.
BNE	.	.	.	.	.	.	.	.	.	2**	.	.	.
BPL	.	.	.	.	.	.	.	.	.	2**	.	.	.
BRK	.	.	.	.	.	.	.	.	.	.	.	.	.
BVC	.	.	.	.	.	.	.	.	.	2**	.	.	.
BVS	.	.	.	.	.	.	.	.	.	2**	.	.	.
CLC	.	.	.	.	.	.	.	.	2	.	.	.	.
CLD	.	.	.	.	.	.	.	.	2	.	.	.	.
CLI	.	.	.	.	.	.	.	.	2	.	.	.	.
CLV	.	.	.	.	.	.	.	.	2	.	.	.	.
CMP	.	2	3	4	.	4	4*	4*	.	.	6	5*	.
CPX	.	2	3	.	.	4	.	.	.	.	.	.	.
CPY	.	2	3	.	.	4	.	.	.	.	.	.	.
DEC	.	.	5	6	.	6	7	.	.	.	.	.	.
DEX	.	.	.	.	.	.	.	.	2	.	.	.	.
DEY	.	.	.	.	.	.	.	.	2	.	.	.	.
EØR	.	2	3	4	.	4	4*	4*	.	.	6	5	.
INC	.	.	5	6	.	6	7	.	.	.	.	.	.
INX	.	.	.	.	.	.	.	.	2	.	.	.	.
INY	.	.	.	.	.	.	.	.	2	.	.	.	.
JMP	.	.	.	.	.	3	.	.	.	.	.	.	5
JSR	.	.	.	.	.	6	.	.	.	.	.	.	.
LDA	.	2	3	4	.	4	4*	4*	.	.	6	5*	.
LDX	.	2	3	.	4	4	.	4*	.	.	.	.	.
LDY	.	2	3	4	.	4	4*	.	.	.	.	.	.
LSR	2	.	5	6	.	6	7	.	.	.	.	.	.
NØP	.	.	.	.	.	.	.	.	2	.	.	.	.
ØRA	.	2	3	4	.	4	4*	4*	.	.	6	5*	.
PHA	.	.	.	.	.	.	.	.	3	.	.	.	.
PHP	.	.	.	.	.	.	.	.	3	.	.	.	.
PLA	.	.	.	.	.	.	.	.	4	.	.	.	.
PLP	.	.	.	.	.	.	.	.	4	.	.	.	.
RØL	2	.	5	6	.	6	7	.	.	.	.	.	.
RTI	.	.	.	.	.	.	.	.	6	.	.	.	.
RTS	.	.	.	.	.	.	.	.	6	.	.	.	.
SBC	.	2	3	4	.	4	4*	4*	.	.	6	5*	.
SEC	.	.	.	.	.	.	.	.	2	.	.	.	.
SED	.	.	.	.	.	.	.	.	2	.	.	.	.
SEI	.	.	.	.	.	.	.	.	2	.	.	.	.
STA	.	.	3	4	.	4	5	5	.	.	6	6	.
STX*	.	.	3	.	4	4	.	.	.	.	.	.	.
STY**	.	.	3	4	.	4	.	.	.	.	.	.	.
TAX	.	.	.	.	.	.	.	.	2	.	.	.	.
TAY	.	.	.	.	.	.	.	.	2	.	.	.	.
TSX	.	.	.	.	.	.	.	.	2	.	.	.	.
TXA	.	.	.	.	.	.	.	.	2	.	.	.	.
TXS	.	.	.	.	.	.	.	.	2	.	.	.	.
TYA	.	.	.	.	.	.	.	.	2	.	.	.	.

* Add one cycle if indexing across page boundary
** Add one cycle if branch is taken, Add one additional if branching operation crosses page boundary

ADDRESSING MODES

ACCUMULATOR ADDRESSING
This form of addressing is represented with a one byte instruction, implying an operation on the accumulator.

IMMEDIATE ADDRESSING
In immediate addressing, the operand is contained in the second byte of the instruction, with no further memory addressing required.

ABSOLUTE ADDRESSING
In absolute addressing, the second byte of the instruction specifies the eight low order bits of the effective address while the third byte specifies the eight high order bits. Thus, the absolute addressing mode allows access to the entire 65K bytes of addressable memory.

ZERO PAGE ADDRESSING
The zero page instructions allow for shorter code and execution times by only fetching the second byte of the instruction and assuming a zero high address byte. Careful use of the zero page can result in significant increase in code efficiency.

INDEXED ZERO PAGE ADDRESSING — (X, Y indexing)
This form of addressing is used in conjunction with the index register and is referred to as "Zero Page, X" or "Zero Page, Y". The effective address is calculated by adding the second byte to the contents of the index register. Since this is a form of "Zero Page" addressing, the content of the second byte references a location in page zero. Additionally due to the "Zero Page" addressing nature of this mode, no carry is added to the high order 8 bits of memory and crossing of page boundaries does not occur.

INDEXED ABSOLUTE ADDRESSING — (X, Y, indexing)
This form of addressing is used in conjunction with X and Y index register and is referred to as "Absolute, X", and "Absolute, Y". The effective address is formed by adding the contents of X or Y to the address contained in the second and third bytes of the instruction. This mode allows the index register to contain the index or count value and the instruction to contain the base address. This type of indexing allows any location referencing and the index to modify multiple fields resulting in reduced coding and execution time.

IMPLIED ADDRESSING
In the implied addressing mode the address containing the operand is implicitly stated in the operation code of the instruction.

RELATIVE ADDRESSING
Relative addressing is used only with branch instructions and establishes a destination for the conditional branch.

The second byte of the instruction becomes the operand which is an "Offset" added to the contents of the lower eight bits of the program counter when the counter is set at the next instruction. The range of the offset is -128 to +127 bytes from the next instruction.

INDEXED INDIRECT ADDRESSING
In indexed indirect addressing (referred to as (Indirect, X)), the second byte of the instruction is added to the contents of the X index register, discarding the carry. The result of this addition points to a memory location on page zero whose contents is the low order eight bits of the effective address. The next memory location in page zero contains the high order eight bits of the effective address. Both memory locations specifying the high and low order bytes of the effective address must be in page zero.

INDIRECT INDEXED ADDRESSING
In indirect indexed addressing (referred to as (Indirect), Y), the second byte of the instruction points to a memory location in page zero. The contents of this memory location is added to the contents of the Y index register, the result being the low order eight bits of the effective address. The carry from this addition is added to the contents of the next page zero memory location, the result being the high order eight bits of the effective address.

ABSOLUTE INDIRECT
The second byte of the instruction contains the low order eight bits of a memory location. The high order eight bits of that memory location is contained in the third byte of the instruction. The contents of the fully specified memory location is the low order byte of the effective address. The next memory location contains the high order byte of the effective address which is loaded into the sixteen bits of the program counter.

ASSEMBLER DIRECTIVES

.OPT — If used must be the first executable statement in the program.

.OPTIONS ARE: — (Options listed are the default value.)

COUNT (COU or CNT)	- List all instructions and their usage.
NOGENERATE (NOG)	- Do not generate more than one line of code for ASCII strings.
XREF (XRE)	- Produce a cross-reference list in the symbol table.
ERRORS (ERR)	- Create an error file.
MEMORY (MEM)	- Create an assembler object output file.
LIST (LIS)	- Produce a full assembly listing.

.BYTE — Produces a single BYTE in memory equal to each operand specified.

.WORD — Produces two BYTES in memory equal to each operand specified.

*= - Defines the beginning of a new program counter sequence.

.PAGE — Advances the listing to the top of a new page.

.END — Defines the end of a source program.

LABELS:

Labels begin in column 1 and are separated from the instruction by at least one space.

Labels can be up to 6 alphanumenic characters long and must begin with an alpha character.

A, X, Y, S, and P are reserved and cannot be used as labels.

LABEL = Expression can be used to equate labels to instructions.

LABEL * = * + N can be used to reserve areas in memory.

CHARACTERS USED AS SPECIAL PREFIXES:

.	Indicates an assembler directive.
#	Specifies the immediate mode of addressing.
$	Specifies a hexadecimal character.
@	Specifies an octal number.
%	Specifies a binary number.
'	Specifies an ASCII literal character.
()	Indicates indirect addressing.
;	In column 1 indicates a comment.

UCS SALES OFFICES

ATLANTA
Bldg. 1, Suite 106
5825 Glenridge Drive N.E.
Atlanta, Georgia 30328
Phone: (404) 256-3610

BOSTON
Fourth Floor
1050 Massachusetts Avenue
Cambridge, Massachusetts 02138
Phone: (617) 661-1720

CALGARY
Suite 1910
Bow Valley Square 2
P. O. Box 9235
Calgary, Alberta, Canada T2P2W5
Phone: (403) 265-4926

CHICAGO
Suite 1016
150 North Wacker Drive
Chicago, Illinois 60606
Phone: (312) 782-0865

CLEVELAND*
Two Commerce Park Square
23200 Chagrin Blvd.
Beachwood, Ohio 44122
Phone: (216) 464-9205

COLUMBUS*
P. O. Box 781
Delaware, Ohio 43015
Phone: (614) 548-6371

DALLAS
Suite 1112, Twin Towers South
8585 Stemmons Freeway
Dallas, Texas 75247
Phone: (214) 638-8260

DENVER
Suite 20C
2460 West 26th Avenue
Denver, Colorado 80211
Phone: (303) 458-8001

EAST ORANGE
33 Evergreen Place
East Orange, New Jersey 07018
Phone: (201) 677-2400

FT. WAYNE
Suite 101
3702 Rupp Drive
Fort Wayne, Indiana 46805
Phone: (219) 484-8522

FT. WORTH
Phone: (214) 263-0584
(Dallas office)

HOUSTON
4544 Post Oak Place
Suite 346
Houston, Texas 77027
Phone: (713) 622-5351

KANSAS CITY
500 W. 26th Street
Kansas City, Missouri 64108
Phone: (816) 221-9700

LOS ANGELES*
Suite 410
101 Continental Boulevard
El Segundo, California 90245
Phone: (213) 640-0891

MILWAUKEE
Suite 107
10701 West North Avenue
Wauwatosa, Wisconsin 53226
Phone: (414) 475-9392

NEW HAVEN
35 Worth Avenue
Hamden, Connecticut 06518
Phone: (203) 288-6287

NEW YORK
Suite 1847
Two Pennsylvania Plaza
New York, New York 10001
Phone: (212) 868-7785

OKLAHOMA CITY
Suite 252
Northwest Office Center
4334 N.W. Expressway
Oklahoma City, Oklahoma 73116
Phone: (405) 843-9784

ORLANDO
Suite 149
7200 Lake Elenor Drive
Orlando, Florida 32809
Phone: (305) 855-1810

PALO ALTO*
Suite 217
1032 Elwell Court
Palo Alto, California 94303
Phone: (415) 964-6990

PHILADELPHIA
Suite 210
500 Office Center
Ft. Washington, Pennsylvania 19034
Phone: (215) 542-8600

PHOENIX
Suite 104
5350 N. 16th St.
Phoenix, Arizona 85016
Phone: (602) 248-9176

SANTA ANA*
Suite 212
1651 East Fourth
Santa Ana, California 92701
Phone: (714) 835-3801

SAN FRANCISCO*
Suite 222
681 Market Street
San Francisco, California 94105
Phone: (415) 777-1885

SEATTLE
Suite B
Koll Commerce Center
699 Strander Blvd.
Tukwila, Washington 98188
Phone: (206) 243-8041

ST. LOUIS
Suite 100
7750 Clayton Road
Clayton, Missouri 63117
Phone: (314) 781-0123

TAMPA
Suite 518
1000 Ashley Drive
Tampa, Florida 33602
Phone: (813) 223-3921

TULSA
Suite 403
16 East 16 Street
Tulsa, Oklahoma 74119
Phone: (918) 582-7291

WASHINGTON, D.C.
Suite 319
7115 Leesburg Pike
Falls Church, Virginia 22043
Phone: (703) 532-1551

NATIONAL DATA CENTER
2525 Washington
Kansas City, Missouri 64108
Phone: (816) 221-9700

CORPORATE OFFICES
2525 Washington
Kansas City, Missouri 64108
Phone: (816) 221-9700

HEADQUARTERS –
MOS TECHNOLOGY, INC. 950 Rittenhouse Road
Norristown, Pa. 19401, (215) 666-7950, TWX: 510/660/4033

EASTERN REGION –
Mr. William Whitehead
MOS TECHNOLOGY, INC., Suite 312,
410 Jericho Turnpike, Jericho, N.Y. 11753
(516) 822-4240

WESTERN REGION –
MOS TECHNOLOGY, INC. 2172 Dupont Drive,
Patio Bldg., Suite 221. Newport Beach, CA. 92660
(714) 833-1600

Mr. Petr Sehnal, Regional Applications Mgr.
MOS TECHNOLOGY, INC., 22300 Foothill Blvd., Suite 311
Hayward, CA 94541
(415) 881-8080

MOS TECHNOLOGY, INC.