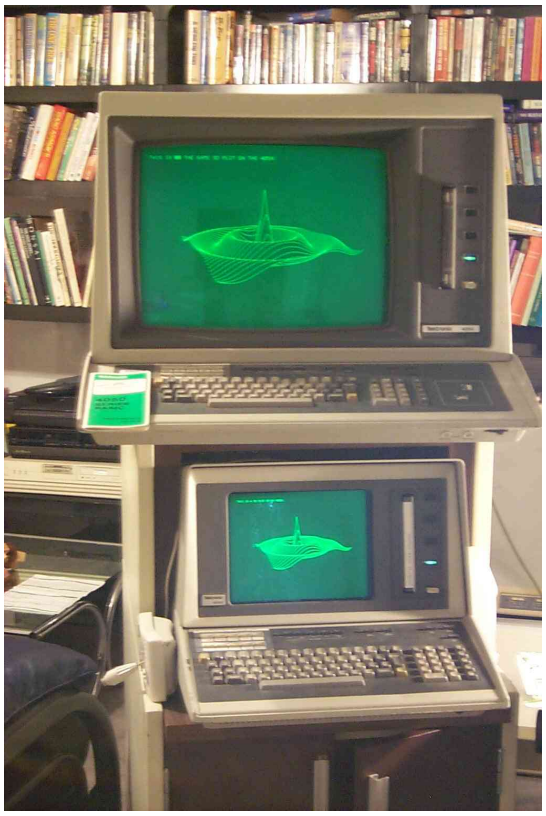


Tektronix 4052 through 4054A Computer Architecture

ABSTRACT

Technical details of the internals of Tektronix bit-slice microprocessor design in the 4052/4054 and 4052A/4054A computers including the new instructions and system ROM entry points.

Monty McGraw

December 8, 2018

Table of Contents

Figures

Credits for cover photo - Monty McGraw's 4054A and 4052 computers in 2000

1. Goal

Document the internal architecture of the bit-slice micro-coded processor in the 4052/4054 and 4050A-Series computers including the new microcode instructions.

2. References

- Tektronix 4052/4054 and 4052A/4054A Technical Data service manual, Section 6 Theory, Rev. Nov 1982
- Tekniques Application Library Vol. 6 No. 4 Tape 1, July 1980
    - Tektronix 4052A/4054A Assembler Program (file 1 on the tape)
    - Tektronix 4052A/4054A Assembler documentation including complete instructions set listing
        - Original 4052/4054 new instructions
        - Plus additional 4050A new instructions
        - And 4050A changes to 6800 instructions for 16-bit A and B register operations
        - Global ROM entry points
    - Tektronix 4052A/4054A BASIC and GPIB Enhancements - Programmers Reference, 070-4383-00 Rev JUN 1983
- Motorola M6800 Programming Reference manual M68PRM(D) Nov 1976

3. 4052x/4054x Microcode MPU Description

The Tektronix 4052/4054 computers were introduced in1978 to significantly improve the performance of the original 4051 computer introduced in 1976. The 4051 computer was designed around a Motorola 6800 microprocessor. The Tektronix 4052/4054 and 4050 A-Series were designed with a microcode architecture using the AMD2901 bit-slice family parts with a design goal to improve performance 8x by doubling the clock to 2MHz, doubling width to 16-bit memory, and simultaneous code and data fetch – according to TekWiki article on the 4052. The initial 4052/4054 performance improvement during development was closer to 3x, until late in development they added floating point microcode instructions which resulted in the final product speedup of about 30x due to the significant speedup of all the graphics commands (TekWiki article). The 4052A/4054A computers were introduced in 1980 and included even more new microcode instructions and a new TI 9914 GPIB controller to improve GPIB performance over the previous bit-banged GPIB hardware interface design.

4052/4054 MPU improvements over the 6800 microprocessor included:

- 42 new instructions including microcode 64-bit floating point
- 16-bit parallel processing
- 128KB memory space: 64KB RAM and 64KB ROM
- Sixteen microcode registers, A, B, INDEX, SP, ten new 8 or 16 bit registers and a 16 bit debug mode flag register
- Two new condition code register bits CCRD (Data Space pointer) and CCRF (Fetch Space pointer)

4052A/4054A MPU improvements over the 4052/4054 included:

- 22 additional new instructions including 16-bit G register operations
- 14 extensions to 6800 A & B register opcodes for 16-bit operations including the resulting CC bits

The 4052A/4054A MPU improvements were also offered to the 4052/4054 customers as a 4052F39 or 4054F39 field upgrade kit which included:

- Replacement ALU microcode ROMs and PLA
- Replacement MAS board including two more system EPROMs
    - The new MAS board eliminated the Patch ROMs – giving 2KB more space for the system ROMs.
    - In addition the new A-Series MAS board included **16KB** of additional system ROMs in the bank switch area at 0000H in the ROM space, including **31** new BASIC keywords for the A-Series including:
        - The MOD command
        - Commands for structured programming
        - More GPIB commands
        - More graphics commands
        - Binary commands
    - Total 4052A/4054A system ROM size = **72KB** (64KB+8KB constants ROM) versus 56KB in the original 4052/4054 design.
- Replacement I/O board including the TI 9914 GPIB controller instead of the PIA logic

4. 4052A/4054A New and enhanced BASIC and GPIB Commands

The most important BASIC enhancements are in the areas of:

- Program presentation - up to 31 character variable & subprogram names, comment tails after !, OLD/APPEND without remarks, formatted list
- Program structuring - subprograms, IF THEN/ELSE/END IF, DO/EXIT IF, DO/EXIT IF/LOOP
- Strings - ALTER, character input, concatenation, string searching, ASC and CHR extended to 8-bits
- Graphics - dashed lines, cross-hatched areas
- Miscellaneous - MOD operator, logical units (I/O) as expressions

New GPIB enhancements included conformance to IEEE Standard 488-1978, and provides the following improvements over the 4052/4054 GPIB:

- Increased GPIB binary data transfer speed
- Elimination of undesirable state transitions in the GPIB bus management signals
- Increased GPIB ASCII transfer speed
- Ability to do data block transfers between the GPIB port and system memory
- Additional capabilities as standard user accessible features

New or enhanced BASIC commands include:

- ALTER
- ANGLE function
- AREA function
- ASC function (extended to 8-bits)
- Binary Operations (BITAND,OR,XOR,CMP,ROT,SHI,TES,SET)
- CALL, SUB, LOCAL, END SUB, and "SYMREUSE"
- CCINPUT
- CENTROID
- CHR function (extended to 8-bits)
- DASH
- DIM (can increase the size)
- DISTANCE function
- DO, LOOP, and EXIT IF
- EXCLUDE
- HATCH
- IF, ELSE, and END IF
- INSIDE function
- LET String Operations
- MOD operator
- RENUMBER
- RND function (added RND(0) values for A-Series)
- RSUM and CSUM functions
- SEARCH function
- TABLE function
- TRIM function
- UBOUND

New GPIB commands or enhancements include:

- Address List Arrays
- CONFIGURE
- IFC
- OFF SRQ
- OFF TIMEOUT
- ON SRQ
- ON TIMEOUT
- POLL
- RENOFF
- RENON
- TIMSET (timeout thresholds)

The 4052A Assembler program takes full advantage of the new A-Series BASIC commands including long variable names, comment tails instead of REM statements, structured BASIC, 8-bit character operations to create the executable binary program string, and the new SEARCH command.

5. Hardware Description

Tektronix designers doubled the memory space to 128KB with 64KB of ROM space and 64KB of RAM space. Both ROM and RAM were also organized 16-bits wide to increase fetch and data access speed. Two additional condition code register bits (CCRF and CCRD) indicated whether the next instruction fetch or data access would be from the ROM bank or the RAM bank.

Tektronix did not document a BASIC CALL "EXEC" command to access assembly code for the original 4052/4054 computers, and the CALL "EXEC" command was undocumented in the 4051. The introduction of the 4052A/4054A computers was accompanied by the publishing of the 4052A/4054A Assembler program in July 1980 including documentation of global ROM entry points that could be used by assembly language programs with a CALL "EXEC" command. Following that tape was a Tekniques Vol. 7 No. 4 Tape in October 1982 which included a 4051 Assembler program and documented the 4051 global ROM entry points and system variables located in low 4051 RAM. A Tekniques article about the 4051 assembler indicated that users of the 4052A Assembler would find the information useful. Indeed, the 4052A Assembler documentation only includes the names of the global ROM entry points, where the 4051 Assembler documentation gives more information on the usage of the calls – and since the global names are the same, this additional information is very useful.

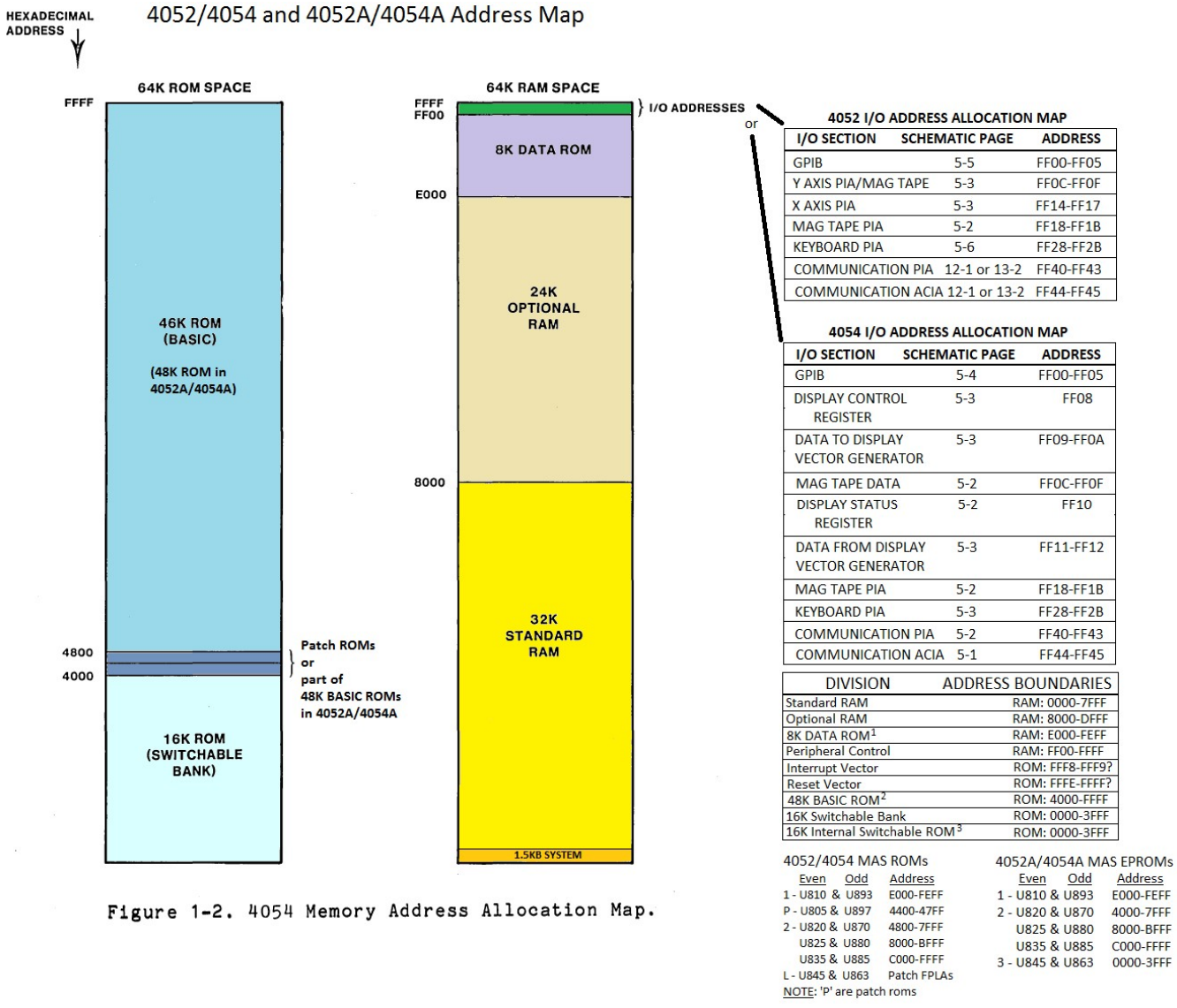


Figure 1-2. 4054 Memory Address Allocation Map.

Figure 1 4052/4054 and 4052A/4054A Address Map

RALU REGISTERS

REGISTER NAME	FUNCTION	NUMBER of BITS
ACCA	ACCUMULATOR A	16
ACCB	ACCUMULATOR B	16
ACCG* A-series only	A extended to 16-bits	16 (A is low byte)
X, XH, XL	INDEX REGISTER	16
SP, SPH, SPL	STACK POINTER	16
PC, PCH, PCL	PROGRAM COUNTER	16
CC	CONDITION CODES	8

CONDITION CODE REGISTER

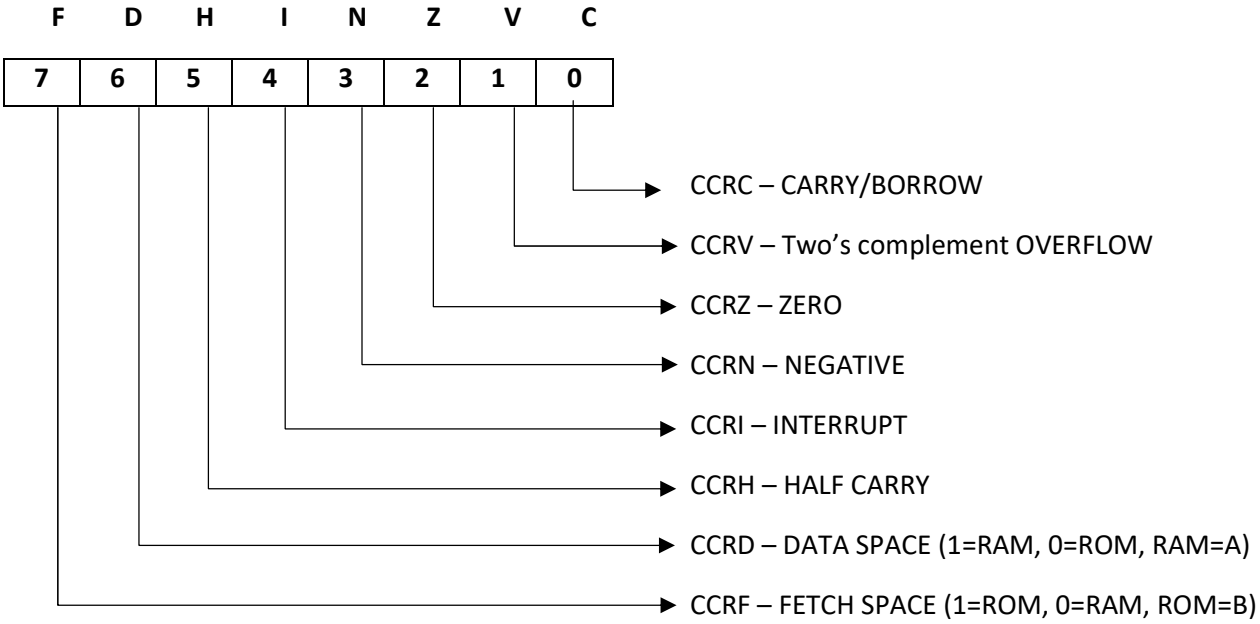


Figure 2 PROCESSOR REGISTERS

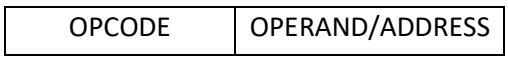
6. 4052/4054 FIRMWARE

The same firmware is used in both 4052 and 4054 to interpret user programs, to transfer data, and to control the I/O devices (internal tape drive, display, keyboard and GPIB). The firmware is located in ROM space and the Data ROM (system constants) in RAM space.

A firmware instruction is one, two or three bytes in length. The first byte of each firmware instruction is the operation code (opcode). The second byte or the second and third bytes are the operand (data) or address of the operand (Figure 2).

ONE BYTE INSTRUCTION

TWO BYTE INSTRUCTION

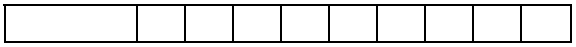


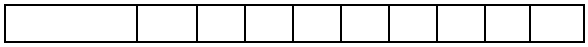
THREE BYTE INSTRUCTION

Figure 3 FIRMWARE INSTRUCTION LENGTH

	Dual Operand	ACCX	Immediate	Direct	Extended	Indexed	Inherent/Implied	Relative	Ext. Immediate
ABA									
ADC									
ADD									
AND									
ASL									
ASR									
BCC									
BCS									
BEA									
BGE									
BGT									
BHI									
BIT									
BLE									
BLS									
BLT									
BMI									
BNE									
BPL									
BRA									
BSR									
BVC									
CBA									
CLC									
CLI									
CLR									
CLV									
CMP									
COM									
CPX									
DAA									
DEC									
DES									
DEX									
EOR									
LDAG				D		X			
TAPX							P		
TPAX							P		
ADXI			I						
ASPI			I						
LDXX							P		

	Dual Operand	ACCX	Immediate	Direct	Extended	Indexed	Inherent/Implied	Relative	Ext. Immediate
INC									
INS									
INX									
JMP									
JSR									
LDA									
LDS									
LDX									
LSR									
NEG									
NOP									
ORA									
PSH									
PUL									
ROL									
ROR									
RTI									
RTS									
SBA									
SBC									
SEC									
SEI									
STA									
STS									
STX									
SUB									
SWI									
TAB									
TAP									
TBA									
TPA									
TST									
TSX									
TXS									
WAI									
LDAX							P		
LDBX							P		
STAX							P		
JMPAX							P		
JMPIN					N				
FPSH				D		X			





7. 4052A/4054A BASIC ROM - GLOBAL ENTRY POINTS

ADDRESS (HEX)	NAME	DESCRIPTION
4000	ADRDEV	Address a device
4003	AFPITT	Convert ASCII to floating point number
4006	APPOLD	Do the I/O portion of APPEND
4009	ASCFPN	Convert string of ASCII chars into FPN
400C	ATNOFF	Clear ATN on GPIB
4012	BACKUP	Back up one stack entry using R0
4015	BAKARS	Backup one record on magtape
4018	BELCAL	Ring the BELL
401B	BFRALC	Allocate I/O buffers using A.PRIM
401E	RBGIN	? in 4052A assembler doc
4021	CIDLE	Routine that displays cursor while waiting for keyboard I/O to come to logical end
4024	CLRARG	Clear argument - special entry point into TYPARG
4027	CLRBNK	Set the bank switch to zero
402A	CRLF	Send CR to output buffer
402D	CRLFLF	Send two CR's to output buffer
4030	CRTRST	Restore shared PIAs to CRT
4033	CTLCHR	Display control character on screen - also handles page full conditions
4036	DATIN	Read from a DATA statement
4039	DEFPNT	Default PRINT formatter
403C	DELAY	Delay number of MS in IX(INDEX?) register
403F	DELET	Delete program lines
4042	DELY1S	Wait some time in 40usec increments
4045	DELY3S	Another wait routine
4048	DIMSTR	Dimension (allocate memory) for a string variable
404B	DISPLE	Disable interrupts - no BREAK BREAK allowed
404E	DISKCL	Call disk interface ROMpack
4051	DLTALL	Perform the basic DELETE ALL function
4054	DRBUSY	Wait for display to be ready
4057	DSDRAW	Send GDU numbers to display vectors (DRAW)
405A	DSHOME	HOME the display cursor
405D	DSMOVE	Send GDU numbers to vectors (MOVE)
4060	DSPAGE	Send FF (PAGE) command to display
4063	DSPCHR	Send char to display (control char will be converted to LIST format)
4066	DSPCPY	Make COPY of display, but set copy suspended flag in CRTSTT if display disabled
4069	DSPOUT	Output to display as dictated by A.SEC
406C	EOTCLR	Edit clear - routine clears the line buffer, sets buffer to spaces
406F	EOTCLS	Close the edit buffer - line is not lost
4072	ENABLE	Enable interrupts - allows BREAK BREAK
4075	EOIOFF	Release EOI control line on GPIB
4078	EOION	Assert EOI control line on GPIB
407B	FIX1	Convert floating point number to an integer
407E	FIXNUM	Fix I/O list entries for internal use
4081	FLOAT	Convert an integer to a floating point number
4084	FMTPNT	Formatted print handler
4087	FPAITT	Convert FPN to ASCII (internal buffers)

408A	FPCMP	Compare two FP numbers on stack
408D	FPNASC	Reduce FPN to fraction (0.1 to 1.0) for later ASCII conversion
4090	FPNEG	Negate FPN on top of stack
4093	FULSCN	Show "blinking F" while waiting for PAGE key
4096	FUZZIE	Do a fuzzie compare on stack
4099	GENCUR	Generate a blinking cursor
409C	GETCHR	Get next character from input buffer (for FP input conversion?)
409F	GETKEY	Handle keyboard interrupt
40A2	GETLAR	Find address and line number of the line with the smallest line number larger than or equal to the argument passed on the stack
40A5	GETLN	Convert FPN on the stack to the address of a given line
40A8	GETLNA	Get line A - convert integer in R0 to address of the given line
40AB	GETSMA	Find address and line number of the line with the largest line number smaller than or equal to the argument passed on the stack
40AE	GINSET	Set up pointers for GIN input
40B1	GRFDR	Perform clipping on vectors
40B4	GRFINI	Reset graphic system default values
40B7	HALTA	Halt and take DREXTA
40BA	HALTR	Halt and RETURN
40BD	IDLE	Main basic idle loop entry point
40C0	IECEOI	Service EOI hardware interrupts
40C3	IECIFC	Assert IFC on GPIB
40C6	IECIN	Input a character from the GPIB
40C9	IECOFF	Get off the GPIB
40CC	IECOUT	Output a character on the GPIB
40CF	IECRD	Read from the GPIB
40D2	IECSND	Send to the GPIB
40D5	INAGTE	Convert integer to ASCII
40D8	INAITT	Convert ASCII to a floating point number
40DB	INFO	Returns the information in OPTABLE for specified BASIC token
40DE	INITMT	Switch shared PIAs to magtape (disables display)
40E1	INPCTL	Input control module
40E4	INPSTG	Input a string
40E7	INPVAL	Input a value
40EA	INTACP	Set GPIB hardware to LISTEN state
40ED	INTCN	Integer constant, routine to float and stack integer
40F0	INTMLA	Integer multiply with accumulator
40F3	INTMLC	Integer multiply, no accumulate
40F6	INTSRC	Initialize GPIB hardware to SOURCE state
40F9	IOCLNR	Clean up stack after I/O and return to original caller
40FC	IODARK	Turn off I/O light on the front panel
40FF	IOLITE	Turn on I/O light on the front panel
4102	IOSCAN	I/O list scanner
4105	KBQOUT	Fetch a key from the type-ahead queue
4108	KEYIN	Input from keyboard
410B	LEX	Lexical analyzer - Convert input lines from ASCII to postfix internal form
410E	LINCN	Float a line number and stack it
4111	LITCN	Literal constant - push pointer to literal
4114	LITREL	Literal relation - string compare

4117	LNEVL	Evaluate one BASIC line
411A	LOCTG	Locate a tagged stack entry
411D	LOCTGR	Locate a tag in a range - search stack for tag in a range of values
4120	MARKS	Mark new magtape files
4123	MATMAT	Matrix-Matrix - apply scaler operator over two arrays
4126	MATMOV	Matrix-Move - assign one matrix to another
4129	MATSCL	Matrix-Scaler - apply scaler to an array and a scaler
412C	MATSIZ	Calculate control constants for a matrix
412F	MAXANS	Push largest FP number on stack
4132	MKFILE	Mark a magtape file
4135	MODMTH	Modify magtape header
4138	MTAFIN	Locate and open a new magtape file
413E	MTBSWP	Swap magtape buffer pointers
4141	MTCLOS	Close a magtape file
4144	MTFIND	Find and open the file specified on the stack
4147	MTKILL	Kill a file on the magtape
414A	MTMARK	Mark a file on the magtape
414D	MTNULL	Null the magtape buffer
4150	MTPADR	Address and setup the magtape system
4153	MTPIN	Get a logical buffer from the magtape
4156	MTPINR	Magtape input request validity interpreter
4159	MTPINW	Magtape output request validity interpreter
415C	MTPOUT	Send a buffer to the magtape
415F	MTPRWD	Controlled rewind of the magtape
4162	MTRBFR	Controlled read of a magtape buffer
4165	MTREAD	Read a physical magtape record
4168	MTSSET	Set the magtape format status register (MTSREG): PRINT @33,0: LENGTH,CHECKSUM,HEADER
416B	MTWRIT	Write a record to the magtape
416E	MTWRT	Same as above
4171	NEWBFR	Get a new input buffer
4174	NEWTAP	Swap in magtape hardware, find self on new tape
4177	NFR8	Position magtape back in file gap
417A	NIODEV	Set no I/O device error message
417D	OLDONE	Special entry point for OLD control
4180	OUTBFR	Send a full output buffer to the current I/O device
4183	OUTCTL	Output controller
4186	PCHAR	Put a char to display
4189	PGMEVL	Program evaluator - routine to start and run a program
418C	PGMLIS	Insert a new line into the current program
418F	PLOTVL	Convert user space to screen space and dump the resultant vector into the output buffer
4192	POPES	Pop evaluator status and reset it in page zero
4195	PRISTG	Print a string
4198	PRIVAL	Print a value
419B	PRIERR	Print an error message
419E	PUPTAP	Initialize magtape hardware
41A1	PUSHES	Push evaluator status
41A4	PUSHX	Push index register and Acc A on stack
41A7	PUSHXT	Push index register and give it a special tag

41AA	PUTBYT	Put a character in the output buffer
41AD	RBYTVL	Get a single byte from the GPIB in RBYTE format
41B0	RDFILH	Read and validate file header record
41B3	READRS	Read a physical magtape record
41B6	REASTG	Read a string
41B9	REAVAL	Read a value
41BC	RENUM	Renumber a BASIC program
41BF	RESTZ	Reset the DATA statement pointers
41C2	REWIND	Rewind the tape
41C5	RNDATA	Round FPN and convert to ASCII
41C8	SCLMAT	Scaler-Matrix - apply a scaler operator over a scaler and an array
41CE	SETARG	Routine to prime TYPARG calls
41D1	SKIPS	Skip a magtape record
41D4	SNDBFR	Send a buffer to a device
41D7	SNGMAT	Apply a monadic scaler operator over one array
41DA	SPOLON	Send Serial POLL Enable
41DD	SQUISN	Compress user memory
41E0	SRQOFF	Reset SRQ pending bit
41E3	SRQRDY	Process SRQ level requests
41E6	STKBLD	Build special stack entries for I/O system
41E9	STORIT	Update magtape control register
41EC	STPMTS	Stop motor
41EF	SYMTAB	Symbol table handler
41F2	SYSERR	Flame out for FATAL FATAL error (should not have occurred)
41F5	TAPE	Process MARK, FIND commands
41F8	TAPFIL	Process LIST, SAVE, OLD commands
41FB	TESTCS	Conditional upcase depending on SET CASE/NOCASE
41FE	TPIMS	Wait for no tape motion
4201	TRANSL	Translator mainline
4204	TRUTH	Determine if FPN closer to 0 or 1
4207	TSTEOF	Test for End of File conditions
420A	TSTINT	Test for raised ON conditions and pending user keys
420D	TST8NF	Alternate action based on file gap size (8NFRs - physical end of file)
4210	TUTSS	Wait for motor up to speed
4213	TYPARG	Determine type of argument (test and simplify function arguments on the stack)
4216	TYPEXT	Special entry piont in MTCTL
4219	TYPIN	Line editor for BASIC
421C	TYPRES	Test for valid result areas for operands
421F	UNADR	Release a device
4222	UNCOMP	Convert postfix internal lines to ASCII
4225	UPCASE	Convert lower case to upper case
4228	VECTOR	Send a vector to the screen
422B	WBYTVL	Send one byte from the WBYTE list
422E	WRISTG	Write a string
4231	WRIVAL	Write a value
4234	WRRS	Write a physical record
4237	XFRCTL	Attach an I/O driver given a A.PRIM and a table pointer
423A	ZANS	Put a floating point zero on stack

8. Vectors into COMM I/F Modules

ADDRESS (HEX)	NAME	DESCRIPTION
423D	AIN	Input a char from channel A
4240	AOUT	Output a char into channel A
4243	CMRPIA	? in asm manual
4246	CMSPIA	?
4249	COMMEM	?
424C	MVBYTF	?
424F	MVBYTE	?
4252	PRTMSF	?
4244	PRTMSG	?
4258	RESREG	?
425B	RSBANK	?
425E	SAVREG	Save registers
4261	TBADD	Two byte add
4264	TBADDI	Two byte add immediate
4267	TBCMP	Two byte compare
426A	TBCMPI	Two byte compare immediate
426D	TBSUB	Two byte subtract

9. Vectors into 4054/Refresh Option Modules

ADDRESS (HEX)	NAME	DESCRIPTION
4270	APPN8X	Append to an object
4273	BLNK8X	Blink an object
4276	BRT8X	Set intensity and focus
4279	B54ELC	Ring the bell
427C	CLEA8X	Re-initialize
427F	CLOS8X	Close an open object
4282	CRO	POINTER command
4285	C54RTR	Restore display hardware
4288	DRAG8X	Drag an object
428B	DRAW8X	Refresh draw
428E	DSINIT	Initialize the display
4291	DSPAGE	PAGE the screen
4294	D54HCO	Hardcopy 4054 screen load
4297	D54RBU	Wait for display not busy
429A	D54SHO	HOME the cursor
429D	FLXV	Lindsay's funny fixer
42A0	FIXBX	Fix an object
42A3	FNDOBJ	Pointer locate
42A6	HNDSHK	Handshake with refresh
42A9	KILL8X	R Delete an object
42AC	LOCA8X	Return objects setpoint
42AF	MEM8X	Return refresh memory left
42B2	OEDINI	Initialize display for Opt. 1 and editor
42B5	OEDQUI	Clean up after Option 1 or Editor
42B8	OPC8X	Open an object
42BE	PICK8X	Set the pointer object
42C1	POIN8X	Post object in X
42C4	POST8X	Post object in X
42C7	PRA24	Do PRI@32,24:
42CA	PRWU54	Power up 4054
42CD	P54CHA	Print a character on the screen
42D0	REBILD	Rebuild the cursor object
42D3	REPL8X	Replace an object with another
42D6	REVXFM	Reverse transform X from 63/64 to 4096 space
42D9	RFIDLE	Refresh replacement for CIDLE
42DC	RFSHND	Force END to refresh
42DF	RFSHCS	Restore refresh context
42E2	RFSHSV	Save refresh context
42E5	SETMSK	Set storage dash mask
42E8	SETP8X	Setpoint an object
42EB	SHACMD	Handshake command with 4054 display
42EE	SPAC8X	Return refresh memory used
42F1	UNPO8X	Unpost object in X
42F4	UNSYOB	Unpost all system objects
42F7	VCTWT	Draw a vector after setup is done

42FA	VISI8X	Set an object's visibility
42FD	VIPCMD	Wait until NOT(VIP ! SHAKE) then data and cmd
4300	V54ECT	Basic MOVE or DRAW
4303	SYMNLN	SYMTAB for n-char variable
4306	SYMLBL	SYMTAB for SUB names
4309	INTEXT	R14 entry into Intsrv
430C	CUTSAB	Safely cuts the stack back (re. BASIC SUBS)
430F	PNPROC	Entry to PAGE - Hardcopy Processor
4312	KBNTCP	Entry to Kb int so ROMpacks can intercept keyboard interrupts

10. OVERVIEW OF THE I/O SYSTEM VARIABLES

This document presents an overview of the I/O System. It also contains a detailed description of the commonly used I/O System variables.

THE4051 I/O SYSTEM WAS DESIGNED TO BE AS MODULAR AS POSSIBLE TO PROVIDE FOR AS MUCH DEVICE INDEPENDENCE AS RESONABLE.

THE FOLLOWING DESCRIBES THOSE I/O VARIABLES REFERED TO AS THE NORMAL I/O SYSTEM VARIABLES IN THE ROUTINE DOCUMENTATION. THESE VARIABLES MUST BE SET UP BEFORE CALLING MANY OF THE I/O ROUTINES. SOME ROUTINES DO EXIST FOR MANAGEMENT OF MANY OF THESE VARIABLES, THESE ROUTINES INCLUDE ADRDEV, BFRALC, UNADR.

NOTE: ^HNN IS THE NOTATION USED TO REPRESENT NN AS A HEX (BASE 16) NUMBER.

```
****************** IOFUNC          ; PRESENT I/O OPERATION (BASIC KEYWORD)
****************** A.STAT          ; CHANNEL A STATUS
;           STATUS BYTE FORMAT
DIRCT  = ^H80                      ; DIRECTION 0-OUTPUT 1-INPUT
BFRSTT = ^H20                      ; BUFFER STATUS 0-EMPTY 1-FULL
BUSACT = ^H10                      ; ACTIVE BUS 0-NO 1-YES
FMTVLD = ^H08                      ; SET IF USING STATEMENT APPLIES
ATVLD  = ^H04                      ; SET IF <ATSIGN> OR <PERCENTSIGN>
                                   ; INFORMATION IS ON STACK
SECDEF = ^H02                      ; SECONDARY DEFINED 0-NO 1-YES
PRIDEF = ^H01                      ; PRIMARY DEFINED 0-NO 1-YES
;
****************** A.PRIM          ; PRESENT PRIMARY ADDRESS
****************** A.SEC           ; PRESENT SECONDARY ADDRESS
****************** A.STRT          ; POINTER TO FIRST VALID
                                   ; CHARACTER POSITION IN THE BUFFER
****************** A.MAX           ; POINTER TO LAST VALID CHARACTER
                                   ; POSITION IN THE BUFFER
****************** A.PTR           ; MOVING POINTER IN I/O BUFFER
                                   ; IT GENERALLY POINTS TO THE NEXT
                                   ; USABLE CHARACTER SLOT
****************** A.END           ; MOVING POINTER IN I/O BUFFER
                                   ; DURING OUTPUT IT POINTS TO THE
                                   ; MOST RECENTLY OUTPUT CHARACTER
```

```
****************** A.MAX          ; POINTER TO LAST VALID CHARACTER
                                  ; POSITION IN THE BUFFER
;
;
;
****************** PPMODE         ; <PERCENTSIGN> MODE FLAG
                                  ; (SET IF <PERCENTSIGN> INVOKED)
****************** IECRLF         ; CR VS. CR/LF FLAG
****************** NOOUT          ; OUTPUT BUFFER FLAGS
NOWRIT = ^H01                     ; IF SET NEVER OUTPUT BUFFER
LSTFMT = ^H02                     ; SET TO OUTPUT IN LIST MODE
SNDIT  = ^H80                     ; SET TO PUT EOI WITH LAST BYTE IN BUFFER
```

THE FOLLOWING VARIABLES ARE OF IMPORTANCE DURING INPUT OPERATIONS

```
****************** CRSTAT         ; INPUT BUFFER STATUS BYTE
                                  ; (CRSTAT=0 IF NO DELIMITER IN BUFFER)
          CRNORM  = ^H01          ; NORMAL EOLCHR <CR> FOUND
          CRDC3   = ^H02          ; NOT USED
          CREOI   = ^H04          ; EOI FOUND
          CREOF   = ^H08          ; EOF FOUND (FILE ONLY)
          CRETX   = ^H10          ; ETX FOUND (LOGICAL END-OF-FILE CHAR.)
          CREOT   = ^H20          ; EOT FOUND (EOF ON INTERNAL TAPE)
          EOFTYP  = ^H38          ; (EOF+ETX+EOT)
          CRVLD   = ^H80          ; CRSTAT IS VALID IF SET
                                  ; (REACHED END OF BUFFER)
****************** EOLCHR         ; CHARACTER USED AS EOL DELIMITER
                                  ; NOTE THIS CHARACTER IS ALSO
                                  ; USED DURING OUTPUT FUNCTIONS!!!!
****************** ETXCHR         ; CHARACTER USED AS EOF DELIMITER
****************** NULCHR         ; CHARACTER USED AS NULL CHARACTER
                                  ; NOTE: IF 128 OR GREATER NO CHARACTER
                                  ; WILL BE USED AS A NULL CHARACTER
```

4051 SYMBOL DEFINITIONS

CONTAINED WITHIN IS A LIST OF THE 4051 SYSTEM GLOBAL SYMBOLS. THE SYMBOLS ARE LISTED IN ALPHABETICAL ORDER. EACH SYMBOL ENTRY CONTAINS INFORMATION IN THE FOLLOWING FORMAT:

SYMBOL(XXXX)T COMMENT ABOUT THE SYMBOL.

SYMBOL — SPECIFIES THE GLOBAL SYMBOL NAME.
(XXXX) — SPECIFIES THE VALUE ASSOCIATED WITH THE GLOBAL SYMBOL.
T — SPECIFIES THE TYPE OF THE SYMBOL ENCODED AS BELOW

- E -- SYMBOL REFERS TO AN ENTRY POINT FOR AN INDIVIDUAL ROUTINE. THE ADDRESS OF THE ENTRY POINT IS INDICATED BY THE (XXXX) ENTRY.
- C -- SYMBOL IS A CONSTANT WITH THE VALUE XXXX.
- V -- SYMBOL IS A GLOBAL VARIABLE WITH THE ADDRESS XXXX. NOTEZ IF XXXX IS LESS THAN 100 (HEX) THEN THE VARIABLE IS ON PAGE ZERO AND MAY BE ADDRESSED USING THE DIRECT MODE.
- M -- SYMBOL IS A MAJOR MODULE NAME (THE NAME ASSOCIATED WITH EACH BLOCK OF LISTINGS). THE VALUE XXXX ASSOCIATED WITH THIS SYMBOL IS MEANINGLESS (AT LEAST PSEUDO RANDOM).

R SYMBOLS:

R0 (0000)V	THE FIRST OF 24 16-BIT SYSTEM REGISTERS
R1 (0002)V	""
R2 (0004)V	""
R3 (0006)V	""
R4 (0008)V	""
R5 (000A)V	""
R6 (000C)V	""
R7 (000E)V	""
R8 (0010)V	""
R9 (0012)V	""
R10(0014)V	""
R11(0016)V	""
R12(0018)V	""
R13(001A)V	""
R14(001C)V	""
R15(001E)V	""
R16(0020)V	""
R17(0022)V	""
R18(0024)V	""
R19(0026)V	""
R20(0028)V	""
R21(002A)V	""
R22(002C)V	""
R23(002E)V	""

```
A.END (0095)V   MOVING POINTER IN I/O BUFFER.  DURING OUTPUT
                 IT POINTS TO THE MOST RECENTLY OUTPUT CHARACTER
A.MAX (0093)V   POINTER TO LAST VALID CHARACTER POSITION IN THE BUFFER
A.PRIM(0090)V   PRESENT PRIMARY ADDRESS
A.PTR (0098)V   MOVING POINTER IN I/O BUFFER.  IT GENERALLY POINTS
                 TO THE NEXT USABLE CHARACTER SLOT
A.SEC (0097)V   PRESENT SECONDARY ADDRESS
A.STAT(008F)V   I/O SYSTEM STATUS FLAG BYTE
A.STRT(0091)V   POINTER TO FIRST VALID CHARACTER POSITION IN BUFFER
ATSNTG(0012)C   STACK TAG TO MARK @<ATSIGN> OR <PERCENTSIGN> INFORMATION

BLINK (00AE)V   CURSOR BLINK CONTROL. SET TO DO WRITE-THRU CHARACTERS

CHAR  (0080)V   GLOBAL CONTAINING FIRST CHARACTER OF ASCII STRING
CRSTAT(006A)V   INPUT BUFFER STATUS FLAG BYTE

DB    (05E0)V   DATA BASE POINTER (PRINT USING)
DIGFLG(05A7)V   FLAG FOR FP INPUT: "NO DIGITS SEEN YET."

EFLAG (05A9)V   FLAG SET ON FLOATING POINT INPUT CONVERSION. "SEEN AN E?"
EQU   (0030)V   FLAG SET DURING FP INPUT CONVERSION
ERRCD (004B)V   ERROR CODE - HOLDING AREA FOR ERROR CODE. IF ZERO NO
                 ERRORS HAVE OCCURED IN THIS LINE

FPC   (053A)V   8-BYTE BUFFER FOR FP NUMBERS

IECRLF(0463)V   INDICATES CR VS CR/LF FLAG STATUS
IOFUNC(008E)V   PRESENT I/O STATEMENT CODE FOR I/O PROCESSOR
IOKONS(E61E)M   MODULE CONTAINING THE DEFINITION OF ALL I/O SYSTEM
                 CONTROL CONSTANTS
IOSYSF(0467)V   I/O SYSTEM ADDRESSED FLAG.  USED BY ROM PACKS
ITSINT(05A8)V   FLAG FOR FP INPUT: "IT'S AN INTEGER."

NOOUT (0087)V   OUTPUT BUFFER STATUS FLAG BYTE

OPRADR(0056)V   OPERATOR ADDRESS - HOLDING AREA FOR PRIMARY EVALUATOR
                 DISPATCH DATA

POINTB(0476)V   I/O PROCESSOR POINTER FOR RECOVERING AFTER PAGE FULL
PPEOF (0465)V   <PERCENTSIGN> MODE EOF CHARACTER.
PPEOR (0464)V   <PERCENTSIGN> MODE EOL CHARACTER.
PPMODE(0077)V   <PERCENTSIGN> MODE ENVOKED FLAG.
PPNUL (0466)V   <PERCENTSIGN> MODE NULL CHARACTER.


SIGNS (00C3)V   FLAG SET IN FP INPUT CONVERSION: "SIGN OF NUMBER."

TABCNT(060B)V   CURRENT LOGICAL OUTPUT POSITION
TFLGS1(0035)V   TRANSLATOR FLAGS BYTE
```

IMM	ASS	DEF FLG	STR	PARM	RAND	OF	

11. DEFINITION of some 4052/4054 GLOBAL ENTRY POINTS (from 4051 manual)

This document provides information pertaining to the 4051 system firmware. The following is a list of the parts.

1) Routine documentation--this is a package containing documentation and interface specifications on routines within the 4051 which may be of use to potential ROM writers.

2) I/O System variable definitions--this is an appendix to the routine documentation. This appendix defines the I/O system variables which must be used for most of the I/O routines. This appendix is referred to in the documentation as the normal I/O system variables.

11.1. ADRDEV

FUNCTION: THIS ROUTINE ASSIGNS PRIMARY AND SECONDARY ADDRESSES AND IN GENERAL SETS UP THE I/O SYSTEM FOR THE CURRENT I/0 REQUESTED BY BASIC.

INPUTS: 1) OPRADR - 2 BYTES: A CONSTANT THAT IS SET UP BY THE EVALUATOR FOR EVERY I/O STATEMENT EXCEPT POLL, RBYTE AND WBYTE, THIS CONSTANT IS DIFFERENT FOR EVERY I/O STATEMENT AND ITS VALUE MAY BE FOUND IN IOKONS.

2) OPTIONAL ADDRESS DATA IS STORED on THE STACK SOMEWHERE AND IS MARKED BY AN ATSNTG.

3) A.STAT - ATVLD BIT 2 <4.> IS SET IF OPTIONAL ADDRESS INFORMATION IS ON THE STACK.

4) PPMODE: FLAG FOR PERCENT MODE DELIMITERS.
PPNUL: PERCENT MODE NULL CHARACTER.
PPEOR : PERCENT MODE END-OF-LINE CHARACTER.
PPEOF: PERCENT MODE EOF CHARACTER.
IECRLF : OPTIONAL CR/LF DELIMITER MODE.

OUTPUTS: IOFUNC: SET TO CODE TO INDICATE PRESENT I/O FUNCTION.
A.STAT: SET APPROPRIATELY - SEE IOEQU FDR DEFINITION OF BITS.
ERRCD: SET IF ILLEGAL ADDRESS
A.PRIM: PRIMARY ADDRESS
A.SEC: SECONDARY ADDRESS
TABCNT : COUNTER FOR PRINTF
IOSYSF=1: I/O SYSTEM FLAG FOR FIRST ACCESS - GENERALLY USED BY ROMPACKS (SPECIFICALLY FILE SYSTEM)
NULCHR: NULL CHARACTER - NORMALLY 255
EOLCHR: END-OF-LINE CHARACTER NORMALLY <CR>
ETXCHR: EOF CHARACTER NDRMALLY 255.
POINTB =65535: FOR ON FULL
DB =65535: FOR ON FULL
BLINK =0: SO SCREEN I/O WILL WORK
NOOUT =0
CRSTAT =0: INITIALIZE BUFFER FLAGS

CALLS: ATPROC
MTPADR

NOTE: WILL ALSO TURN ON THE I/O LIGHT UNLESS PRIMARY ADDRESS IS 32 <DISPLAY> OR 34 <DATA>

11.2. AFPITT

FUNCTION: CONVERTS AN ASCII STRING TO AN FPN

INPUTS:
- R0: POINTER TO FIRST CHARACTER OF STRING
- R1: POINTER TO DELIMITER OR ONE PAST LAST VALID CHARACTER

OUTPUTS:
- FPC: RESULTANT NUMBER
- R0: POINTER TO DELIMETER USED
- R1: UNMODIFIED

USES:
- POINT
- CRSTAT

CALLS:
- PSHFPN
- PULFPN
- INPVAL

11.3. ASCFPN

FUNCTION: CONVERTS AN INPUT STRING OF ASCII CHARACTERS INTO A FLOATING POINT NUMBER
FORMAT IS:

(+/-)(DIGITS)(.)(DIGITS)(UPPER CASE E OR LOWER CASE E)(+/-)(DIGITS)
OVERFLOW CAUSES LARGEST POSSIBLE NUMBER TO BE OUTPUT;
NO ERROR OCCURS. UNDERFLOW CAUSES ZERO TO BE OUTPUT.
IF TOO MANY DIGITS ARE SPECIFIED IN THE FRACTION THEY ARE IGNORED.

INPUTS: FIRST CHARACTER ASSUMED IN GLOBAL CHAR.
CALLS GETCHR TO PUT NEXT CHARACTER IN CHAR.

OUTPUTS: THE FLOATING POINT NUMBER IS PLACED ON THE STACK.
SETS THE SYSTEM FLAGS

DIGFLG : 0 - DIGIT SEEN, <> 0 - NO DIGIT SEEN
ITSINT : 0 - INPUT NOT AN INTEGFR <> 0 - INPUT A NON-NEGATIVE INTEGER
TFLGS1 : SEE NOTES
EFLAG : <> O = ONLY AN E SEEN, 0 = ANYTHING ELSE
SIGNS : HIGH ORDER BIT = FRACTION NEGATIVE. LOW ORDER BIT = EXPONENT NEGATIVE.
ERRCD : IS CLEARED WHEN OUTPUT IS NORMAL.
EQU : <> 0 : NOT AN INTEGER

USES: CNT
X0...X5
T1...T4
DEXP,DEXP+1
PLTAB, THE TABLE OF POWERS OF 10
TABPNT

CALLS: GETCHR
PSHRET
TMULT
PSHFPN
NORMR (=PSHRET + NORM)
MAXANS (IN NORM)
ZANS (IN NORM)
FPDIV
FPMUL
A8X

NOTES: IF CNT=1 THEN IF NUMBER IS AN INTEGER THEN CLEAR BITS 2 AND 7 OF TFLGS1
ELSE SET EQU=1

11.4. BACKUP

FUNCTION: THIS ROUTINE USES R- AS A PSEUDO STACK POINTER AND BACKS IT UP ONE STACK ENTRY. R0 HAS THE ADDRESS OF A VALUE TAG=1 FOR INPUT. IF THE BYTE IS NOT A LEGAL TAG SYSERR IS CALLED. IF THE TAG IS THE END OF THE STACK TAG R0 IS NOT ALTERED.

INPUTS: R0 IS THE ADDRESS OF A STACK TAG-1

OUTPUTS: R0 IS UPDATED TO REFLECT THE STACK ENTRY IS PASSED.

USES: R0

11.5. BELCAL

FUNCTION: BEEP THE BELL

11.6. BFRALC

FUNCTION: INITIALIZE BUFFER POINTERS BASED ON PRIMARY ADDRESS.

INPUTS: A.PRIM : PRIMARY ADDRESS
MTSREG : TO DETERMINE IF 256 OR 128 BYTE MAGTAPE RECORDS
A.STAT : TO DETERMINE IF INPUT OR OUTPUT

OUTPUTS: A.STRT : POINTER TO FIRST SLOT
A.END = A.STRT-1
NOTE : A.STRT - 1 IS ALSO CLEARED, THIS IS NECESSARY FOR PUTBYT
A.MAX : POINTER TO LAST SLOT

CALLS: LDXX

11.7. CRTRST

FUNCTION: RESET THE CRT PIA'S AND RESET THE OLD STATUS OF THE DISPLAY AS WELL AS SERVICING ANY PENDING PAGE, COPY, HOME, REWIND REQUESTS. GENERALLY CALLED TO RESTORE CRT OPERATION AFTER MAGTAPE HAS BEEN USED.

CALLS: DSPENL - SERVICES PENDING REQUESTS
ENABLE - TO OFFSET THE DISABLE OF THE MAGTAPE SECTION

11.8. CTLCHR

FUNCTION: THIS ROUTINE SENDS CHARACTERS TO THE CRT BUT WILL WAIT UNTIL THE PAGE IS CLEARED IF THE DISPLAY IS IN FULL STATUS.

CTLCHR - SENDS ALL CHARACTERS TO THE SCREEN

INPUTS: ACC A - ASCII CHARACTER

CALLS: SETBNK
PCHAR
DSPCPY
FULSCN

NOTES: THIS ROUTINE DOES SOME SPECIAL HANDLING FOR THE OPTIONAL DISPLAY HANDLERS LIKE THE OPTION 1 INTERFACE THRU THE 3 BYTE DSPVCT VARIABLE. IT ALSO HANDLES THE SPECIAL PAGE FULL MODES AS WELL AS THE ON FULL CONDITION.

11.9. DEFPNT

FUNCTION: THIS ROUTINE USES R- AS A PSEUDO STACK POINTER AND BACKS IT UP ONE STACK ENTRY. R0 HAS THE ADDRESS OF A VALUE TAG=1 FOR INPUT. IF THE BYTE IS NOT A LEGAL TAG SYSERR IS CALLED. IF THE TAG IS THE END OF THE STACK TAG R0 IS NOT ALTERED.

INPUTS: R0 IS THE ADDRESS OF A STACK TAG=1

OUTPUTS: R0 IS UPDATED TO REFLECT THE STACK ENTRY IS PASSED.

USES: R0