Bro. Thomas McGahee 80 South Sixth St. Columbus, Ohio 43215 A

SCIENTIFIC CALCULATOR SOFTWARE

The purpose of this software is to provide a fairly efficient operating system for controlling the SUDING SCIENTIFIC CALCULATOR INTERFACE offered by MINI MICRO MART.

The original software provided by Dr. Suding was too cumbersome for me, and it required several hardware implementations that I didn't want. So I wrote a new version that I think should be able to run on any 8008 based system. The software has been performing for several weeks flawlessly, and if no bugs have crept into this documentation you should be in business.

The following are basic system requirements:
1) 8008 based microcomputer, 500 Khz clock (Mark-8,MIL-MOD,SCELBI etc)

2) Standard ASCII keyboard, (no lower case characters required). Must have standard shift and Control characters.

3) SUDING SCIENTIFIC CALCULATOR INTERFACE. My port assignments are INP 002, OUT 014 , but you may easily change that. (they are located at 012160 and 012172).

4) 2K of RAM, preferably starting at 010000 . NOTE: program may be placed in ROM, since all dynamic storage is external to the program body. ROM version would use 6 1702 PROMS (7 if you don't have Monitor-8). 1K of RAM is for USER's Program.

6) MONITOR-8 . This versatile Monitor is highly recommended! Besides allowing you to run this software without modification, it also makes the writing of your own software a fairly simple thing. MONITOR-8 is available through MINI MICRO MART. I am also going to soon make available a reduced version of Monitor, called MONITOR-S8 which will contain JUST those monitor routines needed to run the Scientific Calculator Software. This MONITOR-S8 will accept parallel input at any speed up to about 400 characters per second, and will deliver a parallel output that is TVT-II compatible (maximum speed 450 characters per second). For those persons who use a TTY instead of a TVT, I will include instructions on changing the timing constants in the program so that the parallel output is reduced to a maximum of slightly less than 10 characters per second. This will allow you to use a UART to assemble a serial output for TTY use. (the same UART will take serial TTY and present parallel input port with parallel data, or an electronic keyboard can be be wired in directly, since it is already parallel.) For users of MONITOR-8 who would like parallel I/O I will be making available a program that can reside in ROM and overlay the first 128 words of MONITOR. Besides parallel I/O , it will allow Cassette routine to be called as a RST 010 (a function that was deleted in the ROM version of MONITOR, but which I have replaced.)

Bro. Thomas McGahee **B**

SYSTEM INITIATION: Starting address is 010200. This will cause an ERROR message to be printed, and then Controller will print the Option Table R/L/P: This gives you three choices 1) RUN the program. 2) LIST the program for verification. 3) PROGRAM. This allows program to be written and corrected.

PROGRAM: To write a program keep the following in mind:
CTRL/A = a return to MONITOR-8
CTRL/R = the same as a simple Carriage Return
CTRL/S = Single-Step. This is for program correction(see below.)
CTRL/X = EXIT. Causes a return to Controller.

CR = A CR (Carriage Return) will cause BOTH a CR and a LF.
LF = A simple Line Feed.

DELETE (also called BACKSPACE or RUBOUT) = Backspace and print \.

If a mathematical function/numeral is entered, it will be printed in PROGRAM and LIST modes, but in RUN mode it will be executed but not printed... that is, unless ! is in use: see below.

! Causes program steps enclosed between !'s to be BOTH Printed AND executed during RUN. (It is sort of like a quotation mark for functions and numerals.. it is extremely useful!).

" Serves to allow the user to insert text into the program. Anything within quotes is simply quoted back during RUN. Note that CTRL/A , and CTRL/X cannot be inside quotes, and neither can & , since these have IMMEDIATE application during PROGRAM mode. Note also that while CR during PROGRAM will give BOTH CR and LF, if it is inside quotes it will ONLY give a CR during run. CTRL/R has NO meaning during RUN if it is INSIDE quotes. ALL other characters/codes are VALID inside quotes. NOTE that quotes MUST be closed BEFORE you type & , since this character indicates the END of your program.

SPACES follow this rule during PROGRAM: They are OPTIONAL and may be freely used and intermixed, EVEN between successive digits of a numerical entry. However, they MUST be present after ANY MULTI-KEY FUNCTION entry is complete. In this special case the space itself is used as an identifier to signal the end of the entry. (This feature allows you to use Mnemonics LONGER than two characters. For instance, SI , SIN , SINE , SINEWAVE , and SIPFQTWITHABANANA are all interpreted by the ALPHA-Decoders as the SIne function. Note that a dash is also OK allowing SINE-OF-THE-NUMBER to be interpreted as SIne. AND, if you are using a TVT with computer cursor control, then you can even get a pseudo-space by plugging in a Cursor-Right command!)

NOTE: In some parts of the program I have caused spaces to be interpreted as calculator NOP's. This prevents their accidentally causing a ruckus during certain rather devious and involved routines .

$ Signifies that you want the program to loop back to the beginning . Combined with M+ and RCL , this can be used to cause incrementing and decrementing runs. ALWAYS follow this character with a &.

& This is the character which indicates the END of the program. EVERYTIME this is encountered you will IMMEDIATELY be returned to Controller. (Note that if it is preceded by $, then during RUN it will NEVER be reached because the program keeps looping back to the beginning. The only way to get out of a loop is via your computer controls. (A panel-induced HLT, or pressing the TTY RESTART pushbutton on MIL-MOD systems.)

? Signifies you desire EXTERNAL INPUT. This allows the entry of variables via the keyboard at various points during the program. Use a ? at EACH place you want variables to be input. (See RUN section for more information on using this option). One note: a SPACE is used as the terminator. Since ALPHABETIC codes such as SINE require a space just to indicate the end of the entry of the particular function, then it is plain that if you are inputting a Multi-Key function ONE SPACE alone will not terminate External Input, but TWO SPACES will. This allows you to CHAIN input complex variables under certain circumstances.

CTRL/ Characters other than CTRL/A and CTRL/X may be freely mixed ANYWHERE in the program, EVEN between successive digit entries of a single number. PLEASE NOTE that a CR IS NOT NOT NOT, I REPEAT, IS NOT a space, and is therefore NOT a valid terminator for multi-key functions. This was done on purpose, to allow a rather free formatting ; but is a pitfall for the unwary, who naturally assumes that if he has completed a line he has terminated a function. NOT SO, my dear friend.

@ Is a rather odd character, and has a correspondingly weird function. Since much of your program may be stuff that will not be echoed during execution, and since ALL CTRL/ characters ARE echoed during execution, how do we avoid CR/LFs being printed during RUN, when we ONLY needed them during PROGRAM and LIST ??? Simple. We present you with @, the pseudo-CR/LF !!! It is printed and then executed as a CR/LF during PROGRAM and LIST, but during RUN it is completely ignored, even if it is between !'s.

NOTE: it is good practice to either include CL CL M+ at the beginning of each program to insure all registers are ZERO, OR (if you are doing an incrementing run where this would mess things up), BEFORE writing the Main program, write CL CL M+ & as a simple program. Run it once, and all registers are cleared. Now you can write your incrementing program secure in the knowledge that it will not start off in some WEIRD place. Of course, a STARTING constant could also be loaded by M+, but don't forget to CLear Clear FIRST. EG. CL CL M+ 45 M+ CL & will clear X and store your starting constant, 45.

CTRL/S This allows you to SINGLE-STEP through your stored program. It does not execute anything (unless a & is encountered, which will cause a return to Controller). Each time CTRL/S is depressed, one stored character will be retrieved and printed, and the Pointer address adjusted. Due to the routines used to implement this function you should not attempt to enter CTRL/S's at a rate faster than 3 or 4 per second. You may think of this as a sort of skip-forward command. For instance, if you BACKSPACE twice and then decide you don't want to change anything, hitting CTRL/S twice will bring you back to where you originally were, with all original data retained. CTRL/S and DELETE together comprise a simple but rather effective means for modifying a program on a one-to-one basis.

NOTE CTRL/S and CTRL/X are NEVER stored in user program, since they are all interpreted IMMEDIATELY.

CTRL/X Causes your IMMEDIATE EXIT from programming. Controller will re-type option list. CTRL/X is IMMEDIATE, and it may be used to get out of the middle of a program after doing a modify to the beginning of a program, for instance. (Internally the software also uses CTRL/X as a do-nothing character. We make this character serve double-duty in this fashion.)

DELETE(Backspace or Rubout) Causes a double-decrement of the Pointer address, and prints as a \ . (If you use a TVT, you may wish to make this actually move the cursor back one space). Delete may be pushed as many times as you want, provided you do not backspace MORE characters than you have in your program! Delete does not really delete, but it DOES move you memory-wise back one slot, allowing you to REPLACE the character already stored there. See CTRL/S details for more information.

CR Will get you both CR and LF. This is for operator convenience.

CTRL/R can be used to get ONLY a CR , if this is desired.

LF will generate a single Line Feed .

The next page will discuss functions available.

FUNCTIONS AVAILABLE: The following functions are available to the user of this software:

(') apostrophe = a calculator NOP.
(left-hand parenthesis. (two levels allowed)
) right-hand parenthesis (two levels allowed)
* multiplication
+ addition
(,) comma = a single-key implementation of EEX
(-) subtraction (do not confuse with a negative SIGN. See CHS)
(.) decimal point
(/) division

(=) Causes answer to be found, AND causes printing of the answer.
(>) DISPLAY the contents of the X register.

The following MULTI-KEY FUNCTIONS MUST consist of AT LEAST the first two characters shown for each. More than two characters are allowed, but all CODING is performed on the first two characters. A SPACE MUST follow the entry of the function code:

E.G. 45 SINE * 2 is 1.414 but 45 SINE*2 is .707 since SINE*2 is seen by software only as SI .

Note that spaces are NOT required after single-key functions, so 45SINE *2 is 1.414

ARC (For getting the reciprocal transcendenatl functions)
COS (COSine)
CHS (used for entering negative mantissas and exponents)
CLR (CLears X) (can also clear all registers except Memory).
DGR (selects degrees or radians mode)
EEX (allows entry of signed exponent)

E↑X (e^x)
LOG (Log_{10})

LN (l_n the natural log)

M+ (add to Memory)
NOP (calculator NOP does nothing but idle the calculator)
N! (N Factorial)
PI (3.141592 etc.)
RCL (ReCLaim memory)
REC (RECiprocal 1/x)
SIN (SINe)
STO (STOre X in Memory)

Continued on next page...

SQT (SQuare rooT)
TAN (TANgent)

T↑X (Ten to the X... 10^x)

X←Y (EXChange X and Y)

X↑2 (X^2)

Y↑X (Y^X)

NOTE that no DSP DiSPlay code is given, since this function requires special handling to cause it to print the contents of the X register. It has been implemented as a single-key(>) code.

NUMERALS:

0
1
2
3
4
5
6
7
8
9

NOTE that since spaces may be freely inserted in program, an entry like 1 23 4 56 is interpreted as 123456. A number is terminated, or specified completely only when it is followed by a single-key function or a multi-key function. This means that CTRL characters and spaces do not terminate numerical entries. E.G. 11 1 1* 2 . 00 = 2222. This allows you to give a CR/LF even in the middle of a numerical entry , which is helpful at times.

LIST: this routine will cause the program you have written to be listed exactly as it was entered. This allows you to confirm that it has been correctly programmed and entered. When the & is encountered, there will be a return to controller. (A return to controller means that the option list is printed out).

G

Bro. Thomas McGahee

RUN: This routine will cause your program to be executed. IF the software finds any blatant errors, such as a multi-key function that does not exist, you will get an error message and a return to controller. Thus if software encounters the function GOBARF, it will print ERROR! R/L/P:

More subtle errors on your part will simply result in GARBAGE. Remember the old computer axiom: GARBAGE IN/GARBAGE OUT.

Perhaps the most common errors you will encounter will be:
Not having a space after a multi-key function...
Placing & INSIDE quotation marks...
(An insidious form of this error is only using ONE quotation mark...resulting in the & effectively being in quotation marks!)...
Failing to CLEAR calculator at places where you SHOULD clear it.
Printing stuff you didn't want printed because you forgot to stop echo with a second ! indicator.

Then too, to properly use this software you MUST know the constraints of the MOS Calculator chip itself. READ the calculator manual through thoroughly SEVERAL times. Most complex equations you will wish to solve will have to be re-written in a form the calculator chip can digest. Remember that the software is really a glorified extension of the calculator's usual keyboard. (With extras, such as memorizing all the steps, printing them, and allowing textual messages, of course). If what you are doing would be wrong if done on the calculator keyboard, then don't blame the software when it prints out garbage!

? is a powerful character, allowing external variable input. Note that it allows more than just the entry of simple numbers, though!! for instance, in response to a ? you could input the following: 45SINE SQT REC *2 which will enter the value equal to

$$2\left(\frac{1}{\sqrt{\mathrm{SINE}\ 45^\circ}}\right)$$

You may even request to see the value you are inputting, by simply including = followed by a space as your last input !! E.G. in response to a ? 2*3= (space) would cause 6. to be printed, indicating that you entered 6.

One caution here: how far you can go in entering complex functions in a chained fashion is dependent on what is in the program PRIOR to the ? . Go too far and you may inadvertantly destroy a necessary previous answer stored in the calculator's working registers. You have to know each individual program to know how far you can go before you will create problems. Again, though, this is a factor dependent on the calculator's capacity, and not that of the software.

Bro. Thomas McGahee H

Just a few words concerning the OUTPUT, the ANSWERS generated by the combined Calculator and Software:

Answers can contain UP TO 8 digits in the mantissa, including a decimal point which is floating. In addition there can be a sign preceding the mantissa, a sign preceding the exponent, and a two-digit exponent.

When doing up the software I decided to eliminate leading and trailing zeroes in a manner similar to that followed by the calculator itself. The result is a variable-length answer. For example, the following are typical answers:

```
1.
123.405
22.345 45       Note that a space always separates mantissa and exp.
-24.034
-123.45 -32       here both mantissa and exponent have negative signs.

FO.    the F is an ERROR indicator for overflow/underflow.
```

Because each answer MAY have an exponent printed, it is a good idea to insure that there is adequate spacing between intermediate answers in a program. For example,

```
1.234 2     2.456 3    is much easier to interpret than
1.23422 2.456 3     which is too closely packed.
```

Software insures that EACH mantissa is followed by a printed space. Even where no exponent is found, this space will still be printed. However, there is no built-in safeguard at the end of the exponent, and you might inadvertently run two numbers together: you can spot this sometimes, but it is best to build adequate spacing into your programs as you write them.

One further caution. The program runs at a reasonable speed, however some operations require all sorts of JMPS, CALS, etc., and this coupled with the fact that we have to STALL while entering functions to the rather slow calculator chip can mean that you may over-speed on entering input. The only places that I have actually experienced this is when using CTRL/S , and when entering data in response to a ?. Three or four characters per second is easily handled, but more than this and you may get Garbage. The best thing to do is to find out what IS the maximum speed YOUR system will accept CTRL/S's and External Input, and just be aware of this speed limitation.

Bro. Thomas McGahee

SAMPLE PROGRAMS (Just to show a couple of the functions off to good advantage).

Underlined stuff is Computer-generated response.

```
ERROR!
R/L/P:P
CL CL M+ 10 M+ &
R/L/P:L
CL CL M+ 10 M+ &
R/L/P:R
        (there is a slight pause as this silent program is executed,
R/L/P:P                  setting Memory to 10)
CL 1 M+ CL RCL >" SQUARED IS " X↑2 !=! $ &
R/L/P:R
11. SQUARED IS =121.
12. SQUARED IS =144.
13. SQUARED IS =169.
14. SQUARED IS =196.
15. SQUARED IS =225.
```

and output will continue until interrupted by RESTART signal (in the case of my MIL-MOD-8).

Note that there is an AUTOMATIC CR/LF generated as soon as the $ is encountered. This is built right into the software as part of the initialization routine.

```
ERROR!
R/L/P:P
CL CL " VARIABLE A "?!*!" VARIABLE B "?" IS "!=!&
R/L/P:R
 VARIABLE A 23*2/3> 15.333333* VARIABLE B 2 IS =30.666666
R/L/P:
```

Note that in this example the ? caused the calculator to wait for external input terminated by a space, and in the case of variable A it accepted 23*2/3> , gave us its value, used this as one variable, accepted the second variable (2), and gave us the product of our two variables.

Incidentally, whole gobs of things may be included inside !'s, even things like !CL 2*3 SINE SQT = " HELLO " *5=! which prints as:

```
CL 2*3 SINE SQT    =.3233086 HELLO *5=1.616543
```

The examples I gave are very simple (and even stupid), but I hope they demonstrate some of the principles.

Bro. Thomas McGahee J

CONCERNING THE SOFTWARE ITSELF...

This is not an optimum implementation of my original ideas concerning a complete Scientific Calculator operating system. Among other things, as the program grew I saw places where I could improve certain subroutines so as to provide the operator with greater programming freedom and versatility. In expanding a program which was already pretty far along, I took the easy way out, and instead of re-structuring the whole thing from beginning to end, I simply shuffled some sections around, made a few changes here and there, and threw in a couple of 'patches' where all else failed. Re-writing would not really save much memory...definitely not enough to coax me to taking that course !

On the other hand, I realize how disgusting it is to get your hands on some software only to find that you can't make heads or tails out of what is going on. For this reason I have gone to great pains to clearly and completely document the software itself. In 98% of the software I have placed the functional description next to the instruction it explains. Wherever it was possible I have noted where the program is coming from when it suddenly has another part of the program calling it or jumping to it. All major routines and subroutines are clearly indicated by an asterisk. Further, I am including a listing showing where all LHI and LMI DATA is located, since many persons may want to re-code this software to run in Banks other than that which I have written it for. To TRANSLATE this software to another set of Banks, you have to translate the JMP and CAL Banks (This includes all classes of JMP and CAL instructions, for instance JFZ, JFC, JTZ, JTC, etc...). IN addition those LHI and LMI instructions that are used to set up memory for storing and retrieving data and codes, must also be translated. So here's the information you may need:

Locations 010000 - 010177 contain mostly Octal DATA and cannot be loaded using Symbolic input. Use Monitor LDO routine.

Load symbolics from 010200 - 012370

Starting Address is 010200 or you may use 010211.

MONITOR uses 013350 - 013377

User's program storage area is from 014003 on.

012370 - 013350 is available for user patches to tailor software to their own needs; however, note that my MONITOR-S will use locations 012370- 013250 .

Note that RAM must be available from 013350 on, but all prior software could easily be put in PROM since it never changes.

INP002 at 012172, & OUT 014 at 012160

Bro. Thomas McGahee K

LHI DATA is stored at the following places:

Address	data	for
010201	010	error message
010212	012	controller message
010301	014	Pointer
010312	014	Pointer
010354	014	Pointer
011070	014	Pointer
011102	014	! Status
011240	010	single-function code storage
011321	014	! Status
012004	010	Alpha-code storage
012150	010	digit decoding
012207	014	restoring H for Increment Pointer routine

LMI data is at only one place: 010310 014 Pointer

MONITOR-8 routines are called as follows:

Monitor address	called from
000020	011372
000016	011053
000013	010040
000013	011046
000013	010275
003217	012007
000332	010355
000030	011357

****

ROUTINES and SUBROUTINES and other major points of interest

* Digit Codes are scattered from 010000 - 010040
* Text of ERROR MESSAGE is stored from 010023 - 010032
-- Carriage Return/Line Feed parch from 010041 - 010046
* Codes for Single-Key Functions are stored from 010047 - 010071
* Multi-Key Function codes are stored from 010072 - 010176
* Routine to call error message is from 010200 - 010210
  (010200 is our usual starting address)
* CONTROLLER is from 010211 - 010255
* TEXT STRING is from 010256 - 010274
  (User may wish to add more specific error messages using this routine to output the text.)
* INITIALIZE POINTER 010275 - 010310
  this routine continues on into next routine
* INCREMENT POINTER 010311 - 010325
* LIST 010326 - 010346

Bro. Thomas McGahee 80 South Sixth St. Columbus, Ohio 43215

BANK	BYTE	OCTAL	MNEMONICS	FUNCTION
010	000	270		Digit code for 8
	001	000	not used	
	002	306		Digit code for (F) ERROR
	003	266		Digit code for 6 / A routine at 012134
	004	271		Digit code for 9 / uses the codes
	005	264		Digit code for 4 / generated BY the
	006	265		Digit code for 5 / calculator chip
	007	000	not used	/ to recover the
	010	262		Digit code for 2 / ASCII codes stored here.
	011	000	not used	
	012	000	not used	
	013	000	not used	
	014	263		Digit code for 3
	015	000	not used	
	016	000	not used	
	017	255		Digit code for (-) negative sign
	020	260		Digit code for (0) zero
	021	000	not used	
	022	306		Digit code for (F) ERROR
	023	215		CR (carriage return) / This is the
	024	212		LF (line feed) / ERROR MESSAGE which
	025	305		E / is printed whenever
	026	322		R / any error is detected
	027	322		R / by program.
	030	317		O / See 010200 for
	031	322		R / more details.
	032	241		!
	033	000	not used	
	034	267		Digit code for 7
	035	261		Digit code for 1
	036	000	not used	
	037	230		Digit code for a BLANK (this is a CTRL/X, which is a non-printing character.)

BANK	BYTE	OCTAL	MNEMONICS	FUNCTION
010	040	000		Calculator NOP (caused by a space during RUN)
	041	106	CAL 000013	/ Called by 011140 in response to
	042	013		/ a CR .. gives both CR and LF
	043	000		
	044	016	LBI 230	/ then B is loaded with a non-printing
	045	230		/ CTRL/X
	046	007	RET	/ and then we return (to 011143).
	047	000		(') Apostrophe is a calculator NOP
	050	050		(Left-hand parenthesis
	051	051		) Right-hand parenthesis
	052	044		* Multiplication
	053	042		+ Addition
	054	054		(,) Comma is the same as EEX
	055	043		(-) Subtraction (NOT a negative SIGN)(See CHS)
	056	041		(.) Decimal point
	057	045		(/) Division
	060	021		0 (Zero) / From 010047 to 010071
	061	022		1 / are stored the SINGLE-KEY
	062	023		2 / FUNCTION CODES.
	063	024		3 / Access to these codes
	064	025		4 / is controlled by the
	065	026		5 / "Single Function Decode"
	066	027		6 / found at 011237.
	067	030		7 / Note that numbers and
	070	031		8 / operations are handled
	071	032		9 / the same way.
	072	301		A / The MULTI-KEY FUNCTIONS
	073	322		R / such as Sine, Cosine, etc.
	074	033		ARC code / are stored from 010072
	075	303		C / to 010176. They are stored
	076	317		O / in a THREE BYTE TABLE format.
	077	062		COS code

Bro. Thomas McGahee 80 South Sixth St. Columbus, Ohio 43215

BANK	BYTE	OCTAL	MNEMONICS	FUNCTION
010	100	303	C	/ Access to these codes
	101	310	H	/ is controlled by the
	102	053	CHS code	/ "Alphabetic Function Decode"
	103	303	C	/ or the "External Alphabetic
	104	314	L	/ Decode" when these call
	105	074	CLR code	/ the "Three BYTE Search"
	106	304	D	/ located at 012001.
	107	307	G	
	110	072	DGR code	/ The first two characters
	111	305	E	/ specify the function. Other
	112	305	E	/ characters do not affect
	113	054	EEX code	/ the coding.
	114	305	E	
	115	336	↑	
	116	104	E↑X code	/ NOTE: ↑ (up-arrow) denotes
	117	314	L	/ exponentiation. On TTY it
	120	317	O	/ prints as an up-arrow,
	121	065	LOG code	/ and on TVT it prints as ^ .
	122	314	L	/ The up-arrow is a SHIFT N
	123	316	N	/ character. IF you prefer
	124	064	LN code	/ you may use ** for denoting
	125	315	M	/ exponentiation. Simply
	126	253	+	/ replace 336 code with
	127	070	M+ code	/ the * 252 code at the
	130	316	N	/ following locations:
	131	317	O	/ 010115 010164 010172 and
	132	000	NOP code	/ 010175.
	133	316	N	
	134	241	!	/ NOTE: the Display code has
	135	105	N! code	/ been implemented as > , a
	136	320	P	/ single-key function, instead
	137	311	I	/ of using a multi-key mnemonic.

(> is a SHIFT PERIOD (.) character)

Bro. Thomas McGahee 80 South Sixth St. Columbus, Ohio 43215

BANK	BYTE	OCTAL	MNEMONICS	FUNCTION
010	140	052	PI code (π)	
	141	322	R	
	142	303	C	
	143	067	RCL code	
	144	322	R	
	145	305	E	
	146	101	REC code (RECiprocal)	
	147	323	S	/ 1/x would not be a proper
	150	311	I	/ alphabetic code, so I
	151	061	SIN code	/ arbitrarily chose REC
	152	323	S	/ as my mnemonic.
	153	324	T	
	154	073	STO code (STOre)	
	155	323	S	
	156	321	Q	
	157	066	SQT code (SQuare rooT)	
	160	324	T	
	161	301	A	
	162	063	TAN code	
	163	324	T	
	164	336	↑	
	165	103	T↑X code (10↑X ...Ten to the X power)	
	166	330	X	/ The left-arrow at 010167
	167	337	←	/ denotes the EXCHANGE function.
	170	071	X←Y code	/ The left-arrow is a SHIFT O
	171	330	X	/ which prints as a (_) on a
	172	336	↑	/ TVT. A good alternate mnemonic
	173	102	X↑2 code	/ would be EXChange which
	174	331	Y	/ would code 305..330..071.
	175	336	↑	
	176	046	Y↑X code	
	177	000	not used	

BANK	BYTE	OCTAL	MNEMONICS	FUNCTION
010	200	056	LHI 010	* ERROR MESSAGE
	201	010		/ We set the memory address to
	202	066	LLI 023	/ 010023 , the start of the
	203	023		/ error message.
	204	046	LEI 032	/ E defines the END of the message
	205	032		/ which is at 010032.
	206	106	CAL 010256	/ Then we call our TEXT STRING
	207	256		/ routine which prints the message
	210	010		/ and then we continue into Controller.
	211	056	LHI 012	* CONTROLLER
	212	012		/ We set the memory address to
	213	066	LLI 224	/ 012224 , the start of the
	214	224		/ Controller message (R/L/P:).
	215	046	LEI 234	/ E defines the END of the message
	216	234		/ which is at 012234.
	217	106	CAL 010256	/ Then we call our TEXT STRING
	220	256		/ routine to print the message.
	221	010		
	222	106	CAL 011357	/ Now get character from Keyboard.
	223	357		
	224	011		
	225	074	CPI 241	/ If the character is a space OR
	226	241		/ ANY CTRL character
	227	140	JTC 010222	/ then go get another input.
	230	222		/ (this allows Home and Erase
	231	010		/ when using a TVT).
	232	300	LAA (NOP)	
	233	300	LAA (NOP)	
	234	074	CPI 320	/ If we have a P
	235	320		
	236	150	JTZ 010371	/ jump to PROGRAMMER at 010371.
	237	371		

BANK	BYTE	OCTAL	MNEMONICS	FUNCTION
010	240	010		
	241	074	CPI 322	/ Of course, if it was an R
	242	322		/ then we would
	243	150	JTZ 011064	/ jump on over to RUN
	244	064		/ which starts at 011064.
	245	011		
	246	074	CPI 314	/ Then again, an L
	247	314		/ would compel us to
	250	150	JTZ 010326	/ visit the LIST routine
	251	326		/ which starts at 010326.
	252	010		
	253	104	JMP 010200	/ ANY other character is a mistake,
	254	200		/ so we jump to ERROR, and from
	255	010		/ there go to CONTROLLER again.
	256	317	LBM	* TEXT STRING
	257	106	CAL 011372	/ Get a stored character into B
	260	372		/ and call the routine to print it.
	261	011		
	262	304	LAE	/ Load A with E (END)
	263	276	CPL	/ and compare this with the low order
	264	053	RTZ	/ memory address. Return if they match.
	265	060	INL	/ If they don't match, increment L
	266	110	JFZ 010256	/ and if there is no carry to worry
	267	256		/ about, then jump back to get
	270	010		/ more characters from storage.
	271	050	INH	/ On a carry we increment H
	272	104	JMP 010256	/ and then jump back to get
	273	256		/ more stored characters.
	274	010		
	275	106	CAL 000013	* INITIALIZE POINTER
	276	013		/ So as not to be slobs, we always
	277	000		/ start with a CR and a LF.

Bro. Thomas McGahee 80 South Sixth St. Columbus, Ohio 43215

BANK	BYTE	OCTAL	MNEMONICS		FUNCTION
010	300	056	LHI 014		/ The Program Pointer is in RAM:
	301	014			/ the Low order is at 014001 ,
	302	066	LLI 001		/ and the High order is at 014002.
	303	001			/ We initially store adress 014001
	304	076	LMI 001		/ in the Pointer, but it will be
	305	001			/ incremented twice before anything
	306	060	INL		/ is stored: Program storage, then,
	307	076	LMI 014		/ is from RAM 01 003 and on.
	310	014			
	311	056	LHI 014	*	INCREMENT POINTER
	312	014			/ We set the memory to the Pointer
	313	066	LLI 001		/ address (014001), Low order.
	314	001			
	315	106	CAL 000315		/ Then we call the MONITOR-8 routine
	316	315			/ to increment the address stored
	317	000			/ AT the Pointer location(s).
	320	106	CAL 012206		/ Call the patch to restore H & L
	321	206			/ to the Pointer location.
	322	012			/ (I hate patches, but what the heck.)
	323	357	LHM		/ H gets the High order
	324	360	LLA		/ and L gets the Low order.
	325	007	RET		/ Memory is set to address in Pointer.
	326	106	CAL 010275	*	LIST
	327	275			/ Initialize and Increment Pointer.
	330	010			
	331	106	CAL 010311		/ Increment the Pointer (again).
	332	311			
	333	010			
	334	317	LBM		/ B gets the stored character
	335	106	CAL 011372		/ and we call routine to print it.
	336	372			
	337	011			

Bro. Thomas McGahee 80 South Sixth St. Columbus, Ohio 43215

BANK	BYTE	OCTAL	MNEMONICS	FUNCTION
010	340	307	LAM	/ Get the stored character into A
	341	106	CAL 011013	/ and then call CHECKLIST to see if
	342	013		/ it needs special handling.
	343	011		
	344	104	JMP 010331	/ If you get back here,
	345	331		/ there are more characters to get--
	346	010		/ so get busy and do it !
	347	046	LEI 001	* DOUBLE DECREMENT
	350	001		/ Set E=001 (for counting).
	351	334	LDE	/ Set D=001 too
	352	363	LLD	/ Set Low order to 001
	353	056	LHI 014	/ and set High order to 014 :
	354	014		/ This sets us to Pointer address.
	355	106	CAL 000332	/ Now we let MONITOR-8 decrement
	356	332		/ the adress stored IN the Pointer.
	357	000		
	360	041	DCE	/ We decrement E ; the first time
	361	150	JTZ 010352	/ it will go to 000 , and we go back
	362	352		/ and decrement again-- but the
	363	010		/ next time E goes to 377 , and
	364	016	LBI 334	/ we load B with ASCII for a \
	365	334		/ (used for DELETE indicator)
	366	007	RET	/ Then return home.
	367	300	LAA (NOP)	
	370	300	LAA (NOP)	
	371	106	CAL 010275	* PROGRAMMER (loads your program)
	372	275		/ First, Initialize and Increment
	373	010		/ the Pointer.
	374	106	CAL 010311	/ Then increment the Pointer again
	375	311		/ to get ready to store a character.
	376	010		
	377	106	CAL 011357	/ Get a character from Keyboard

Bro. Thomas McGahee 80 South Sixth St. Columbus, Ohio 43215

BANK	BYTE	OCTAL	MNEMONICS	FUNCTION
011	000	357		
	001	011		
	002	300	LAA (NOP)	
	003	300	LAA (NOP)	
	004	106	CAL 011013	/ Then call the CHECKLIST
	005	013		/ to see if we have a special
	006	011		/ character to take care of.
	007	300	LAA (NOP)	
	010	104	JMP 010374	/ Take off in search of more
	011	374		/ Keyboard input !!
	012	010		
	013	074	CPI 377	* CHECKLIST
	014	377		/ First we check for a DELETE,
	015	150	JTZ 011346	/ which requires a Double Decrement
	016	346		/ (and when done, it returns to
	017	011		/ the section that called Checklist).
	020	074	CPI 230	/ IF it is a CTRL/X it means you
	021	230		/ want out, so we
	022	150	JTZ 010211	/ GO to CONTROLLER
	023	211		/ (Farewell Cruel World---).
	024	010		
	025	074	CPI 223	/ IF it is a CTRL/S
	026	223		/ we do something special--
	027	110	JFZ 011036	/ which is skipped otherwise
	030	036		/ by jumping to 011036...
	031	011		
	032	106	CAL 012201	/ That something special is to
	033	201		/ go elsewhere, see what is NOW
	034	012		/ in memory, print it,
	035	307	LAM	/ get it into A for future reference,
	036	370	LMA	/ and in EITHER CASE load the character
	037	074	CPI 246	/ into memory. IF it is (&),

Bro. Thomas McGahee 80 South Sixth St. Columbus, Ohio 43215

BANK	BYTE	OCTAL	MNEMONICS	FUNCTION
011	040	246		
	041	150	JTZ 010211	/ then we know it is the end
	042	211		/ of the user's program,
	043	010		/ and we go home to Mommy.
	044	074	CPI 300	/ However, a (@) will cause us
	045	300		/ to first have MONITOR-8 give us
	046	152	CTZ 000013	/ a CR and a LF.
	047	013		
	050	000		
	051	074	CPI 215	/ IF it is a CR
	052	215		/ we also do a LF.
	053	152	CTZ 000016	/ (MONITOR-8 does it for us).
	054	016		
	055	000		
	056	074	CPI 222	/ IF it is NOT a CTRL/R ,
	057	222		/ then we are done here
	060	013	RFZ	/ so Return...
	061	104	JMP 011370	/ But if it IS a CTRL/R
	062	370		/ we will output a CR and then
	063	011		/ Return (to List or Programmer).
	064	106	CAL 010275	* RUN
	065	275		/ Initialize and Increment Pointer.
	066	010		
	067	056	LHI 014	/ We are setting RAM to store the
	070	014		/ ! STATUS at 014000 , since all
	071	066	LLI 000	/ our registers will be busy.
	072	000		
	073	250	XRA	/ XRA sets A=000 ... then we set
	074	370	LMA	/ the ! STATUS to 000 initially.
	075	106	CAL 010311	/ Increment the Pointer.
	076	311		
	077	010		

Bro. Thomas McGahee 80 South Sixth St. Columbus, Ohio 43215

BANK	BYTE	OCTAL	MNEMONICS	FUNCTION
011	100	307	LAM	/ Get a character from storage
	101	056	LHI 014	/ then set H & L so that the
	102	014		/ memory references the ! STATUS
	103	066	LLI 000	/ which is stored at 014000 .
	104	000		
	105	074	CPI 241	/ IF the character is NOT
	106	241		/ a (!),
	107	110	JFZ 011123	/ then skip to 011123 .
	110	123		
	111	011		/ But if it IS a (!),
	112	317	LBM	/ then load status info into B,
	113	010	INB	/ increment B
	114	301	LAB	/ transfer B to A so we can
	115	044	NDI 001	/ AND-MASK it so that all we save
	116	001		/ is the last (LSB) bit.
	117	370	LMA	/ Then store the NEW ! STATUS,
	120	104	JMP 011075	/ and go back for more characters.
	121	075		
	122	011		
	123	074	CPI 242	/ Perhaps we have a ("),
	124	242		
	125	150	JTZ 012044	/ in which case we go to the
	126	044		/ QUOTE routine at 012044 .
	127	012		
	130	074	CPI 244	/ IF we find a ($)
	131	244		/ we accept the bribe
	132	150	JTZ 011064	/ and endlessly loop back to the
	133	064		/ beginning of the user's program.
	134	011		/ (This could make me dizzy !!).
	135	310	LBA	/ Just in case, store A in B ,
	136	074	CPI 215	/ and check for a CR...
	137	215		

BANK	BYTE	OCTAL	MNEMONICS	FUNCTION
011	140	152	CTZ 010041	/ IF we have a CR, we execute
	141	041		/ BOTH a CR AND a LF, and load B
	142	010		/ with a harmless CTRL/X .
	143	074	CPI 222	/ IF it is NOT a CTRL/R
	144	222		/ we skip over
	145	110	JFZ 011152	/ to 011152.
	146	152		
	147	011		
	150	016	LBI 215	/ but if it IS a CTRL/R
	151	215		/ weload B with the ASCII for a CR.
	152	074	CPI 240	/ ANY CTRL character will make
	153	240		/ the C flag True:
	154	100	JFC 011165	/ Other characters will cause a
	155	165		/ jump to 011165,
	156	011		/ but
	157	106	CAL 011372	/ CTRL characters will be printed.
	160	372		/ and
	161	011		/ then
	162	104	JMP 011075	/ we zip back to 011075 ready to
	163	075		/ get more characters.
	164	011		
	165	074	CPI 300	/ IF it is a (@)
	166	300		/ we do not print it.
	167	150	JTZ 011075	/ (Do not pass GO,
	170	075		/ do not collect $ 200), just
	171	011		/ get back to work !
	172	140	JTC 011203	/ IF it is less than 300,
	173	203		/ then it is non-alphabetic...
	174	011		/ so head for 011203.
	175	106	CAL 011304	/ IF it IS ALPHABETIC, then
	176	304		/ call the ALPHABETIC DECODER
	177	011		/ (it will feed the calculator)

Bro. Thomas McGahee 80 South Sixth St. Columbus, Ohio 43215

BANK	BYTE	OCTAL	MNEMONICS	FUNCTION
011	200	104	JMP 011075	/ (Burp!). After digesting the
	201	075		/ Alphabetic Soup (so to speak),
	202	011		/ we set off for more characters.
	203	074	CPI 246	/ IF we have a (&)
	204	246		/ we know this is the END, so
	205	150	JTZ 010211	/ Farewell Cruel World,
	206	211		/ and back to the Controller.
	207	010		
	210	307	LAM	/ At this time we retrieve
	211	074	CPI 001	/ the ! STATUS from memory,
	212	001		/ and if it is 001 , then
	213	152	CTZ 011372	/ we print the character in B.
	214	372		
	215	011		
	216	301	LAB	/ In any case, make sure A also has
	217	074	CPI 277	/ the character. IF it is a (?)
	220	277		/ then we head for the
	221	150	JTZ 011251	/ EXTERNAL INPUT routine
	222	251		/ at 011251.
	223	011		
	224	074	CPI 275	/ IF we have a (=) or a (>),
	225	275		/ then we call a rather involved
	226	102	CFC 012315	/ routine that eventually loads
	227	315		/ the proper function into calculator,
	230	012		/ AND PRINTS THE ANSWER.
	231	112	CFZ 011237	/ NON-ALPHABETIC characters call
	232	237		/ the SINGLE-FUNCTION DECODER.
	233	011		
	234	104	JMP 011075	/ We is all done here, so
	235	075		/ let's go get more characters!
	236	011		/ (Isn't this FUN ???).
	237	056	LHI 010	* SINGLE (KEY) FUNCTION DECODER

Bro. Thomas McGahee 80 South Sixth St. Columbus, Ohio 43215

BANK	BYTE	OCTAL	MNEMONICS	FUNCTION
011	240	010		/ Set H to 010 (Bank address),
	241	024	SUI 200	/ strip the character in A
	242	200		/ of its MSB
	243	360	LLA	/ and use this as the BYTE address.
	244	307	LAM	/ Recover the stored code
	245	106	CAL 012026	/ and cram it down the
	246	026		/ calculator's throat.
	247	012		
	250	007	RET	/ Return to get more characters.
	251	106	CAL 011357	* EXTERNAL INPUT
	252	357		/ GET a character from the Keyboard
	253	011		/ and get it all fixed up properly,
	254	300	LAA (NOP)	/ and check to see if it is a space.
	255	300	LAA (NOP)	
	256	300	LAA (NOP)	
	257	300	LAA (NOP)	
	260	150	JTZ 011075	/ IF it is a space we know we have
	261	075		/ finished here, so go back to
	262	011		/ the RUN program.
	263	074	CPI 300	/ IF we have a NON-ALPHABETIC
	264	300		/ character
	265	140	JTC 012300	/ we go to 012300 to take a
	266	300		/ closer look at it.
	267	012		
	270	106	CAL 012257	/ But if it IS ALPHABETIC, we call
	271	257		/ the EXTERNAL ALPHABETIC DECODER
	272	012		/ which decodes and stuffs the calc.
	273	104	JMP 011251	/ Then we check for more
	274	251		/ Keyboard input.
	275	011		
	276	106	CAL 011237	/ The SINGLE-KEY FUNCTION DECODER
	277	237		/ handles NON-ALPHABETIC characters.

Bro. Thomas McGahee 80 South Sixth St. Columbus, Ohio 43215

BANK	BYTE	OCTAL	MNEMONICS	FUNCTION
011	300	011		
	301	104	JMP 011251	/ Upon our return we go looking
	302	251		/ for more Keyboard input.
	303	011		
	304	330	LDA	* ALPHABETIC DECODER
	305	106	CAL 010311	/ LDA stores FIRST character in D.
	306	311		/ then we increment the pointer
	307	010		
	310	347	LEM	/ and store 2nd character in E.
	311	106	CAL 012001	/ Perform a THREE BYTE SEARCH
	312	001		/ which includes decoding and
	313	012		/ stuffing code into calculator.
	314	106	CAL 010347	/ Then call DOUBLE DECREMENT
	315	347		/ which backs us up to where our
	316	010		/ first ALPHA character was stored.
	317	317	LBM	/ Place stored character in B.
	320	056	LHI 014	/ Set up memory to recover the
	321	014		/ ! STATUS at 014000 .
	322	066	LLI 000	
	323	000		
	324	307	LAM	/ Get ! STATUS into A, and compare
	325	276	CPL	/ it with L, which is 000 .
	326	150	JTZ 011334	/ If ! STATUS is 000, then there is
	327	334		/ NO echo; skip to 011334 .
	330	011		
	331	106	CAL 011372	/ IF echo was desired, print character
	332	372		/ in B.
	333	011		
	334	301	LAB	/ Make sure A also has character.
	335	074	CPI 240	/ IF it is a space,
	336	240		
	337	053	RTZ	/ then we is done--Begone !

BANK	BYTE	OCTAL	MNEMONICS	FUNCTION
011	340	106	CAL 010311	/ IF it is NOT a space, then
	341	311		/ increment the Pointer to set up
	342	010		/ the next stored character
	343	104	JMP 011317	/ and then loop back for more!
	344	317		
	345	011		
	346	106	CAL 010347	/ --Jumped to here from 011015.
	347	347		/ For DELETE we call DOUBLE DECREMENT,
	350	010		/ and upon our return we
	351	104	JMP 012215	/ go to 012215 to print \ and
	352	215		/ then get sent back home.
	353	012		/ (you're right, this IS a patch!).
	354	300	LAA (NOP)	/ (I bet the NOP's gave me away).
	355	300	LAA (NOP)	
	356	300	LAA (NOP)	
	357	106	CAL 000030	* KEYBOARD INPUT
	360	030		/ MONITOR-8 gets the Keyboard input
	361	000		
	362	004	ADI 200	/ Replace the MSB since we need it
	363	200		/ in output routine.
	364	074	CPI 240	/ IF it is a space, set Z flag True
	365	240		/ for possible branch instructions.
	366	007	RET	/ Finished--go home.
	367	300	LAA (NOP)	
	370	016	LBI 215	/ --011061 wants CTRL/R to be changed
	371	215		/ to a CR. continue .....
	372	106	CAL 000020	* TTY/TVT OUTPUT
	373	020		/ MONITOR-8 outputs character in B
	374	000		
	375	301	LAB	/ We restore character to A
	376	074	CPI 240	/ and if it is a space, we set the
	377	240		/ Z flag True

Bro. Thomas McGahee 80 South Sixth St. Columbus, Ohio 43215

BANK	BYTE	OCTAL	MNEMONICS	FUNCTION
012	000	007	RET	/ and then head for home.
	001	026	LCI 030	* THREE BYTE SEARCH (Alpha-decode)
	002	030		/ If you got here, you must have
	003	056	LHI 010	/ two characters (in D & E),
	004	010		/ so set C (counter) to 030
	005	066	LLI 072	/ and set memory address at 010072,
	006	072		/ the start of the stored codes.
	007	106	CAL 003217	/ Call for MONITOR-8 to execute
	010	217		/ a Three Byte Search.
	011	003		
	012	317	LBM	/ Load recovered code into B
	013	302	LAC	/ and load C (counter) into A.
	014	CPI	000	/ IF C had 000 it indicates that
	015	000		/ it was an INVALID CODE,
	016	150	JTZ 010200	/ so we PANIC and give an ERROR
	017	200		/ message and go to Controller.
	020	010		
	021	301	LAB	/ Load valid codes into A
	022	106	CAL 012026	/ and stuff it into the calculator.
	023	026		
	024	012		
	025	007	RET	/ That's all--go home now.
	026	074	CPI 275	* OUTPUT FUNCTIONS (stuff calculator!)
	027	275		/ If it is NOT a non-coded (=),
	030	110	JFZ 012035	/ skip to 012035.
	031	035		
	032	012		
	033	006	LAI 047	/ If it is an (=), code it
	034	047		
	035	106	CAL 012160	/ In any case, stuff the code
	036	160		/ into the calculator.
	037	012		

Bro. Thomas McGahee 80 South Sixth St. Columbus, Ohio 43215

BANK	BYTE	OCTAL	MNEMONICS	FUNCTION
012	040	106	CAL 012157	/ "IDLE" the calculator.
	041	157		
	042	012		
	043	007	RET	/ Go back where you came from.
	044	106	CAL 010311	* QUOTE (Jumped to from 011124)
	045	311		/ Increment the Pointer
	046	010		
	047	307	LAM	/ Get the character into A.
	050	074	CPI 242	/ If we find the closing ("),
	051	242		/ it means we are finished quoting,
	052	150	JTZ 011075	/ so go back to the RUN routine.
	053	075		
	054	011		
	055	310	LBA	/ If we are still quoting, load B
	056	106	CAL 011372	/ with the character, and then
	057	372		/ print it.
	060	011		
	061	104	JMP 012044	/ Now loop back for more!
	062	044		
	063	012		
	064	006	LAI 211	* DIGIT SELECT
	065	211		/ Load A with digit #9 code (211),
	066	106	CAL 012160	/ and cram it into calculator.
	067	160		
	070	012		
	071	104	JMP 012242	/ Initially we set E=200 and wait
	072	242		/ until calculator has valid output.
	073	012		
	074	040	INE	/ Now increment E and check to see
	075	006	LAI 215	/ if we have done all the digits
	076	215		/ (201-214).
	077	274	CPE	/ IF E= 215, then we are done

BANK	BYTE	OCTAL	MNEMONICS	FUNCTION
012	100	053	RTZ	/ so go on home.
	101	006	LAI 212	/ If we are not finished, check
	102	212		/ and see if this is digit 10 (212).
	103	274	CPE	/ If it <u>is</u> digit 10,
	104	150	JTZ 012124	/ jump to 012124.
	105	124		
	106	012		
	107	304	LAE	/ Load A with current digit code
	110	106	CAL 012160	/ and stuff our digit request code
	111	160		/ into the calculator.
	112	012		
	113	106	CAL 012172	/ Wait for a VALID calculator
	114	172		/ output.
	115	012		
	116	106	CAL 012134	/ Decode the answer and then
	117	134		/ print the answer.
	120	012		
	121	104	JMP 012074	/ Now loop back and check for
	122	074		/ the next digit.
	123	012		
	124	016	LBI 240	/ When it is digit 10 (beginning of
	125	240		/ exponent), we load B with ASCII
	126	106	CAL 011372	/ for a space. We print a space
	127	372		/ so that things are kept neat
	130	011		/ and unambiguous.
	131	104	JMP 012107	/ Then we go back and continue
	132	107		/ digit output as usual.
	133	012		
	134	016	LBI 240	* DIGIT DECODER
	135	240		/ Check: numbers smaller than 240
	136	271	CPB	/ indicate there is a decimal point,
	137	140	JTC 012326	/ and we handle these at 012326 .

Bro. Thomas McGahee 80 South Sixth St. Columbus, Ohio 43215

BANK	BYTE	OCTAL	MNEMONICS	FUNCTION
012	140	326		
	141	012		
	142	016	LBI 040	/ Non-decimal point numbers
	143	040		/ are bigger by 40, so we
	144	221	SUB	/ subtract 40 (this cuts coding in half)
	145	024	SUI 200	/ Both types of numbers are reduced
	146	200		/ by 200. Register A has new code.
	147	056	LHI 010	/ Bank is 010, and BYTE address is
	150	010		/ set equal to
	151	360	LLA	/ the new code in A.
	152	317	LBM	/ GET the ASCII from memory into B
	153	106	CAL 011372	/ and print the answer.
	154	372		
	155	011		
	156	007	RET	/ Now you can go home.
	157	250	XRA	* IDLE CALCULATOR / XRA sets A=000,
	160	131	OUT 014	* STUFF CALCULATOR (with code in A).
	161	026	LCI 375	/ Calculator is on Output Port 024
	162	375		/ in my system. C and D are loaded
	163	036	LDI 150	/ with constants for delay routine,
	164	150		
	165	104	JMP 012360	/ Then we go to our timing loop.
	166	360		/ (from there we Return home).
	167	012		
	170	046	LEI 200	* INITIALIZE DIGIT COUNT (in E register).
	171	200		/ Initially digit counter is 200.
	172	105	INP 002	* WAIT FOR VALID CALCULATOR DATA
	173	074	CPI 200	/ READ calculator. If the data out
	174	200		/ is less than 200, it is invalid,
	175	140	JTC 012172	/ so here we go loop-the-loop
	176	172		/ until we DO get a valid READ data.
	177	012		

Bro. Thomas McGahee 80 South Sixth St. Columbus, Ohio 43215

BANK	BYTE	OCTAL	MNEMONICS	FUNCTION
012	200	007	RET	/ With a valid code you go home.
	201	317	LBM	* CTRL/S ROUTINE (called from 011032)
	202	106	CAL 011372	/ Get a character from memory into B,
	203	372		/ and print it.
	204	011		
	205	007	RET	/ Return (to 011035).
	206	056	LHI 014	* PATCH FOR RESTORING MEM. TO POINTER
	207	014		/ Called from 010320, this patch
	210	066	LLI 001	/ sets memory to Pointer (014001),
	211	001		/ then sets memory to address <u>stored</u>
	212	307	LAM	/ in Pointer! Register A gets the
	213	060	INL	/ Low order. Incrementing L sets
	214	007	RET	/ memory to High Order. RETURN (010323).
	215	106	CAL 011372	* MORE DELETE PATCHING
	216	372		/ Called from 011351, this causes
	217	011		/ the \ to be printed
	220	104	JMP 010374	/ Then we jump to 010374
	221	374		/ to continue Programming routine.
	222	010		
	223	300	LAA (NOP)	
	224	215	CR	* CONTROLLER MESSAGE (R/L/P:)
	225	212	LF	/ Controller (010211) prints this
	226	322	R	/ simple message. It may be
	227	257	/	/ expanded by up to five more
	230	314	L	/ characters since this message is
	231	257	/	/ followed by 5 LAA (NOP)'s.
	232	320	P	/ Just change the E (END) value
	233	272	:	/ at 010216.
	234	207	CTRL/G	/ CTRL/G rings the Bell if you
	235	300	LAA (NOP)	/ have one.
	236	300	LAA (NOP)	
	237	300	LAA (NOP)	

Bro. Thomas McGahee 80 South Sixth St. Columbus, Ohio 43215

BANK	BYTE	OCTAL	MNEMONICS	FUNCTION
012	240	300	LAA (NOP)	
	241	300	LAA (NOP)	
	242	106	CAL 012170	* CHECK FOR END OF CALCULATOR BLANKING
	243	170		/ Jumped to from 012071 , this
	244	012		/ initializes E to 200, waits for
	245	044	NDI 007	/ calculator output greater than 200 ,
	246	007		/ AND-MASKS last 3 Bits and
	247	074	CPI 007	/ IF the recovered code ends in 7,
	250	007		/ it indicates BLANKING is still in
	251	150	JTZ 012242	/ progress, so we loop back and
	252	242		/ continue looping until we get a
	253	012		/ code greater than 200 that does
	254	104	JMP 012074	/ NOT end in 7. THEN we jump
	255	074		/ to 012074 to start assembling
	256	012		/ our Answer one digit at a time.
	257	340	LEA	* EXTERNAL ALPHABETIC DECODER
	260	106	CAL 011357	/ Called from 011270... Load first
	261	357		/ character into E (temporary store).
	262	011		/ Input another Keyboard character.
	263	334	LDE	/ Move 1st character (E) into D,
	264	340	LEA	/ and 2nd (A) character into E
	265	106	CAL 012001	/ THEN call a THREE BYTE SEARCH,
	266	001		/ and include a stuffing of the
	267	012		/ code into the calculator.
	270	106	CAL 011357	/ Coding and stuffing are finished,
	271	357		/ but more than two characters
	272	011		/ are allowed, so accept more
	273	110	JFZ 012270	/ Keyboard input, and if it is NOT
	274	270		/ a space, we go back and accept
	275	012		/ more Keyboard input.
	276	007	RET	/ A space causes us to Return.
	277	300	LAA (NOP)	

BANK	BYTE	OCTAL	MNEMONICS	FUNCTION
012	300	074	CPI 240	* CHECK FOR SPECIAL CASES (= and >)
	301	240		/--We jumped here from 011265.
	302	140	JTC 011251	/ ANY CTRL character is not a
	303	251		/ function, so go back to 011251
	304	011		/ for more characters.
	305	074	CPI 275	/ IF we have (= or >), then we
	306	275		/ will have to
	307	102	CFC 012315	/ call on 012315 for help.
	310	315		
	311	012		
	312	104	JMP 011276	/ When finished, or if it was not
	313	276		/ a special case, then jump back
	314	011		/ to EXTERNAL routine for morework!
	315	106	CAL 012350	/ AHA! we HAVE (= or >), so we
	316	350		/ convert it to proper code, stuff
	317	012		/ the calculator with the function,
	320	106	CAL 012064	/ Then print the answer by calling
	321	064		/ DIGIT SELECT routine.
	322	012		
	323	006	LAI 247	/ Load A with a ('), which will
	324	247		/ decode as a calculator NOP,
	325	007	RET	/ and head off into the sunset!
	326	106	CAL 012145	* DECIMAL POINT OUTPUT
	327	145		/ Called from 012166, this routine
	330	012		/ first prints the digit itself,
	331	015	LBI 256	/ then we load B with ASCII for a
	332	256		/ decimal point,
	333	106	CAL 011372	/ and then we print the
	334	372		/ decimal point.
	335	011		/ We then Return to 012150 to continue
	336	007	RET	/ for more if needed.
	337	300	LAA (NOP)	/ The following nine NOP's are for possible future expansion.

Bro. Thomas McGahee 80 South Sixth St. Columbus, Ohio 43215

BANK	BYTE	OCTAL	MNEMONICS	FUNCTION
012	340	300	LAA (NOP)	/ NOP's are for user expansion
	341	300	LAA (NOP)	/ of software.
	342	300	LAA (NOP)	
	343	300	LAA (NOP)	
	344	300	LAA (NOP)	
	345	300	LAA (NOP)	
	346	300	LAA (NOP)	
	347	300	LAA (NOP)	
	350	150	JTZ 012026	/--Called from 012315, this routine
	351	026		/ gives non-coded (=) to
	352	012		/ decoder which codes and stuffs it.
	353	006	LAI 034	/ But if it is (>), then we give A
	354	034		/ the DISPLAY code (034),
	355	104	JMP 012026	/ Then we stuff it. (Stuffing will
	356	026		/ include printing 'answer').
	357	012		
	360	030	IND	* TIMING LOOP
	361	110	JFZ 012360	/ Keep incrementing D until it goes
	362	360		/ to 000
	363	012		
	364	020	INC	/ Increment C , and if it has not
	365	110	JFZ 012360	/ yet reached 000 , then increment
	366	360		/ D 377 times (octal).
	367	012		
	370	007	RET	/ Timing is finished...go on home.
	371	300	LAA (NOP)	
	372	300	LAA (NOP)	/ These NOP's are available for
	373	300	LAA (NOP)	/ user expansion of software.
	374	300	LAA (NOP)	
	375	300	LAA (NOP)	
	376	300	LAA (NOP)	
	377	300	LAA (NOP)	/ And THAT'S ALL, FOLKS !

/

Bro. Thomas McGahee 80 South Sixth St. Columbus, Ohio 43215

MINIMUM MONITOR: FOR SCIENTIFIC CALCULATOR

This is a collection of the absolute minimum of software routines which are needed to support my software for the SUDING SCIENTIFIC CALCULATOR INTERFACE. I was able to reduce it a bit further than I originally thought, and the result is 89 instructions/words located from 012371 to 013121. Since my routines are minimal, they reduce the overhead of RAM needed to support the Monitor routines to ZERO. This means that the user now has 013122-013377 free for his own use. You may write expanded versions of some of my software there, or add patches of your own to do such things as add extended error messages and the like.

The input/output routines are for parallel operation. If you are using a serial device, then a UART can easily perform the conversion to/from parallel. Note thatlocations 013005 and 013007 require the user to plug in the values for the time delay. This allows you to tailor the system I/O for slow (110 baud, 10 characters per second) or high speed (can go as high as 14,000 baud...but most sytems cannot handle that high). For TVT use I suggest a rate of 450 characters per second max. This is well under the maximum acceptance speed of the TVT, and allows for rather wide variations in the system timing.

If you already have a parallel input and output port available, then you should be able to implement the hardware necessary for under $2 . I strongly suggest using the 74123 approach since this always ensures proper settling times for the data.

The hardware/software functions as follows: When told to get an input character, the software keeps looping and searching for the MSB (Most Significant Bit) being HIGH. As soon as it detects this it "echoes" the character to the output port. Since the MSB is normally kept LOW, and ALL characters have the MSB HIGH, there will always be a low-to-high transition at the MSB. This is detected and a negative pulse developed (in most cases after an additional 100 ns. delay) which informs the TVT or UART that data is ready to be given to the recieving device. This same signal resets the MSB of the input port, and may be used to reset the keyboard,(if the keyboard requires a reset signal). Meanwhile the software causes a delay (this is in the output routine, which is called by the input routine). This delay is user-selectable, and determines the maximum speed at which characters can be printed...and therefore the maximum speed at which they can be input also.

Besides the I/O routines, other routines are contained in the software. All of these routines replace a specific Monitor routine:

ADDRESS	EQUIVALENT MONITOR ADDRESS	FUNCTION OF ROUTINE
012371	000013	OUTPUT A CARRIAGE RET. + LINE FEED
012376	000016	OUTPUT A LINE FEED
013000	000020	PARALLEL OUTPUT
013023	000030	PARALLEL INPUT with ECHO
013045	000070	TIMER
013052	000315	INCREMENT ADDRESS
013075	003217	THREE BYTE TABLE SEARCH
013063	000332	DECREMENT ADDRESS

You will find where these routines should be called from on page K.

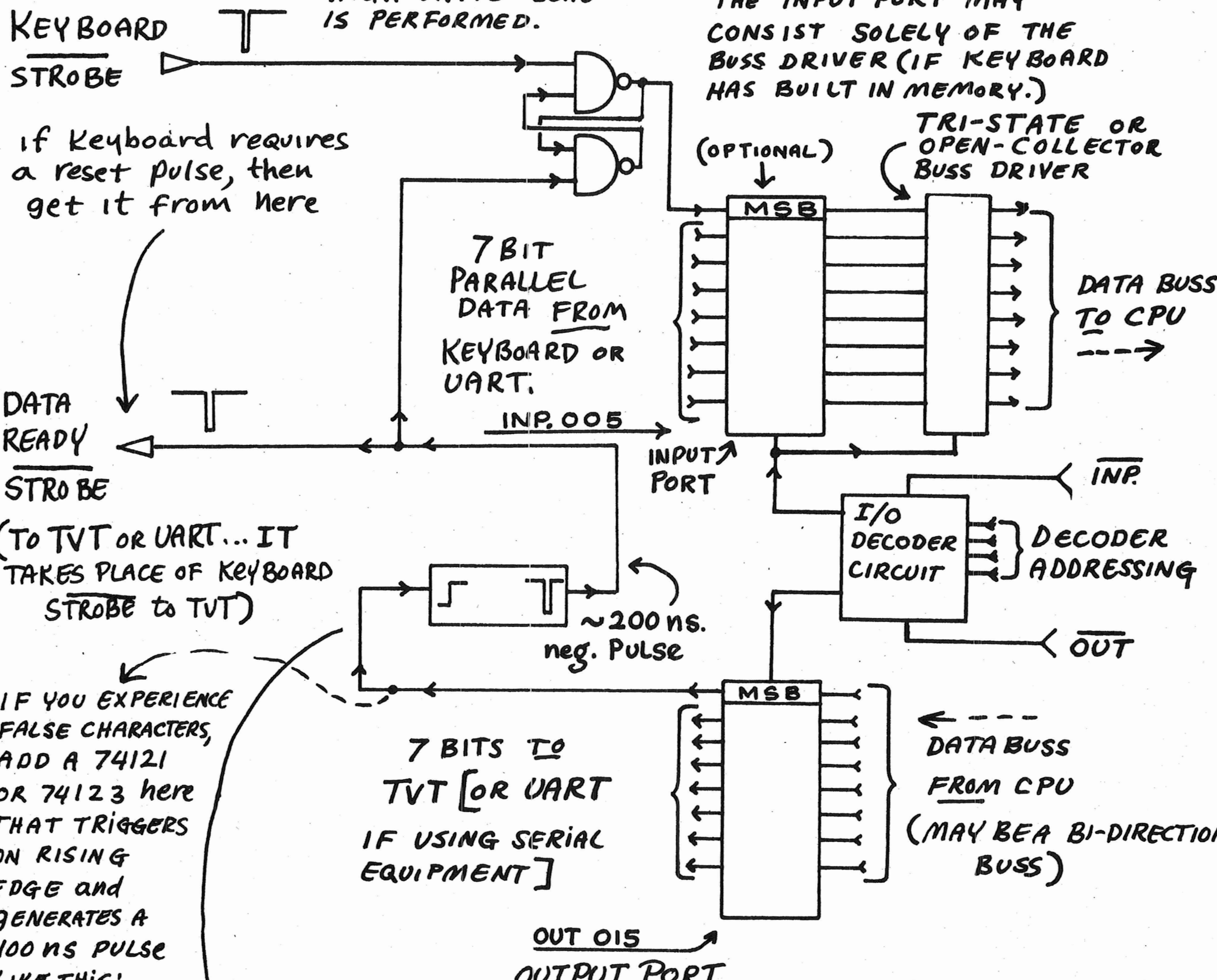
Parallel I/O Hardware

By Bro. Thomas McGahee
for Mini Micro Mart

THIS RS FLIP FLOP CAUSES MSB TO GO HIGH AND STAY HIGH UNTIL "ECHO" IS PERFORMED.

IN MANY SYSTEMS THE INPUT PORT MAY CONSIST SOLELY OF THE BUSS DRIVER (IF KEYBOARD HAS BUILT IN MEMORY.)

if keyboard requires a reset pulse, then get it from here

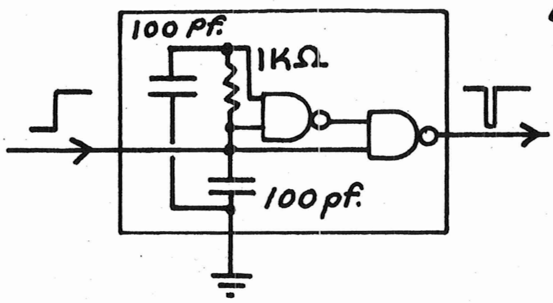


IF YOU EXPERIENCE FALSE CHARACTERS, ADD A 74121 OR 74123 here THAT TRIGGERS ON RISING EDGE and GENERATES A 100 ns PULSE LIKE THIS:

THIS will allow extra time FOR DATA TO "SETTLE" BEFORE TVT accepts IT.

A SINGLE 74123 can Provide BOTH ONE-SHOT FUNCTIONS FOR ABOUT $1.50. This is BEST.

To GET 200 ns. pulse you can use a 74121 or This cheaper alternative which is just as good in this application



THIS DETECTS THE RISING Edge of MSB and SIGNALS TVT or UART to accept Data.

Bro. Thomas McGahee
80 South Sixth St.
Columbus, Ohio
43215

PARALLEL I/O
Program: MINIMUM Monitor
REV. FOR SCIENTIFIC CALCULATOR

ANK	BYTE	CODE	MNEMONICS	FUNCTIONAL DESCRIPTION
012	40			
	41			
	42			
	43			
	44			
	45			
	46			
	47			
	50			
	51			
	52			
	53			
	54			
	55			
	56			
	57			
	60			
	61			
	62			
	63			
	64			
	65		MINIMUM MONITOR	
	66		OCCUPIES 012 371 — 013121	
	67			
	70			
012	371	016	LBI 215	* CR+LF (old ~~000~~ 000013)
	372	215		
	373	106	CAL 013000	/ PRINT CR
	374	000		
	375	013		
	376	016	LBI 212	* LF (old 000016)
	377	212		

Bro. Thomas McGahee 80 South Sixth St. Columbus, Ohio 43215

BANK	BYTE	OCTAL	MNEMONICS	FUNCTION
013	000	250	XRA	* PARallel OUTPUT (old 000020)
	001	133	OUT 015	/ idle Parallel output Port
	002	301	LAB	
	003	133	OUT 015	/ STUFF CHARACTER into PORT
	004	026	LCI XXX →	(001 for TUT, 014 for 110 BAUD TTY)
	005			
	006	036	LDI XXX →	(342 for TUT, 060 FOR 110 BAUD TTY)
	007			
	010	106	CAL 013 045	/ CALL TIMER
	011	045		
	012	013		
	013	061	DCL	/ Decrement C and if NOT
	014	[redacted]	JFZ 013 006	/ 000, LOOP
	015	006	~~006~~	
	016	013		
	017	250	XRA	/ [redacted] idle OUTPUT PORT
	020	133	OUT 015	
	021	301	LAB	/ get character into A
	022	007	RET	/ and RETURN
	023	113	INP 005	* Parallel INPUT W/ECHO (old 000030)
	024	310	LBA	/ KEEP CHECKING
	025	074	CPI 200	/ INPUT FOR
	026	200		/ MSB High
	027	140	JTC 013 023	/ BY LOOPING,
	030	023		
	031	013		/ WHEN IT IS HIGH,
	032	106	~~BST~~ CAL 013 000	/ ECHO the CHARACTER
	033	000		
	034	013		
	035	044	~~CPI 001~~ NDI 177	/ STRIP off MSB
	036	[redacted]	177	
	037	074	CPI 001	/ check for CTRL/A

BANK	BYTE	OCTAL	MNEMONICS	FUNCTION
013	040	001		
	041	013	RFZ	/ Return if not CTRL/A.
	042	104	JMP 010200	/ a CTRL A ~~calls a~~ causes
	043	211		/ a return to Controller.
	044	010		
	045	030	IND	* TIMER (old 000070)
	046	110	JFZ 013045	/ KEEP INCREMENTING D
	047	045		/ UNTIL IT = 000
	050	013		
	051	007	RET	/ THEN GO HOME
	052	317	LBM	* INCrement Address (old 000315)
	053	010	INB	/ GET LOW ORDER FROM
	054	371	LMB	/ MEMORY INTO B, INCREMENT
	055	013	RFZ	/ AND PLACE IN MEMORY.
	056	060	INL	/ RETURN UNLESS WE NEED
	057	317	LBM	/ A CARRY. TO CARRY GET
	060	010	INB	/ HIGH ORDER INTO B, INCREMENT,
	061	371	LMB	/ RESTORE IT TO MEMORY
	062	007	RET	/ AND RETURN.
	063	317	LBM	* Decrement Address (old 000332)
	064	011	DCB	/ THIS IS THE SAME AS
	065	371	LMB	/ THE "INCREMENT" ROUTINE
	066	060	INL	/ EXCEPT IT DECREMENTS.
	067	010	INB	
	070	013	RFZ	
	071	317	LBM	
	072	011	DCB	
	073	371	LMB	
	074	007	RET	/ RETURN WHEN DONE
	075	021	DCC	* 3 BYTE TABLE SEARCH (old 003217)
	076	013	RTZ	/ DECREMENT C & IF = 000, RETURN.
	077	307	LAM	/ OTHERWISE GET A CHARACTER

BANK	BYTE	OCTAL	MNEMONICS	FUNCTION
013	100	060	INC	/ get Ready for NEXT CHARACTER,
	101	273	CPD	/ See if 1st CHARACTER MATCHES.
	102	110	~~JMP~~ JFZ 013115	/ IF NOT, JMP to NEXT 3BYTE
	103	115		/ Block
	104	013		
	105	307	LAM	/ IF 1ST MATCHed, get
	106	274	CPE	/ ANOTHER CHARACTER &
	107	110	JFZ 013115	/ IF NO MATCH JMP to
	110	115		/ NEXT 3BYTE Block
	111	013		
	112	060	INC	/ if BOTH MATCHED THEN
	113	307	LAM	/ Go To NEXT Location and
	114	007	RET	/ GET DATA and Go HOME.
	115	060	INC	/ go to NEXT 3BYTE
	116	060	INC	/ Block
	117	104	JMP 013075	/ AND TRY AGAIN.
	120	075		
	121	013		
	122			Rest is free for user.
	123			
	124			
	125			
	126			
	127			
	130			
	131			
	132			
	133			
	134			
	135			
	136			
	137			

HARDWARE/SOFTWARE WITH "HANDSHAKING" FOR MAXIMUM SPEED.

This hardware/software combination is meant to allow the output device to operate at maximum possible speed. Instead of using timing loops to pre-determine the delay, this system lets the receiving apparatus signal when it is ready to accept new data. This 'handshaking' is preceeded and followed by a delay of at least 100 nanoseconds to insure proper 'settling' of data being transferred.

Since Input routine uses the Output routine, we will explain the Input routine in detail:

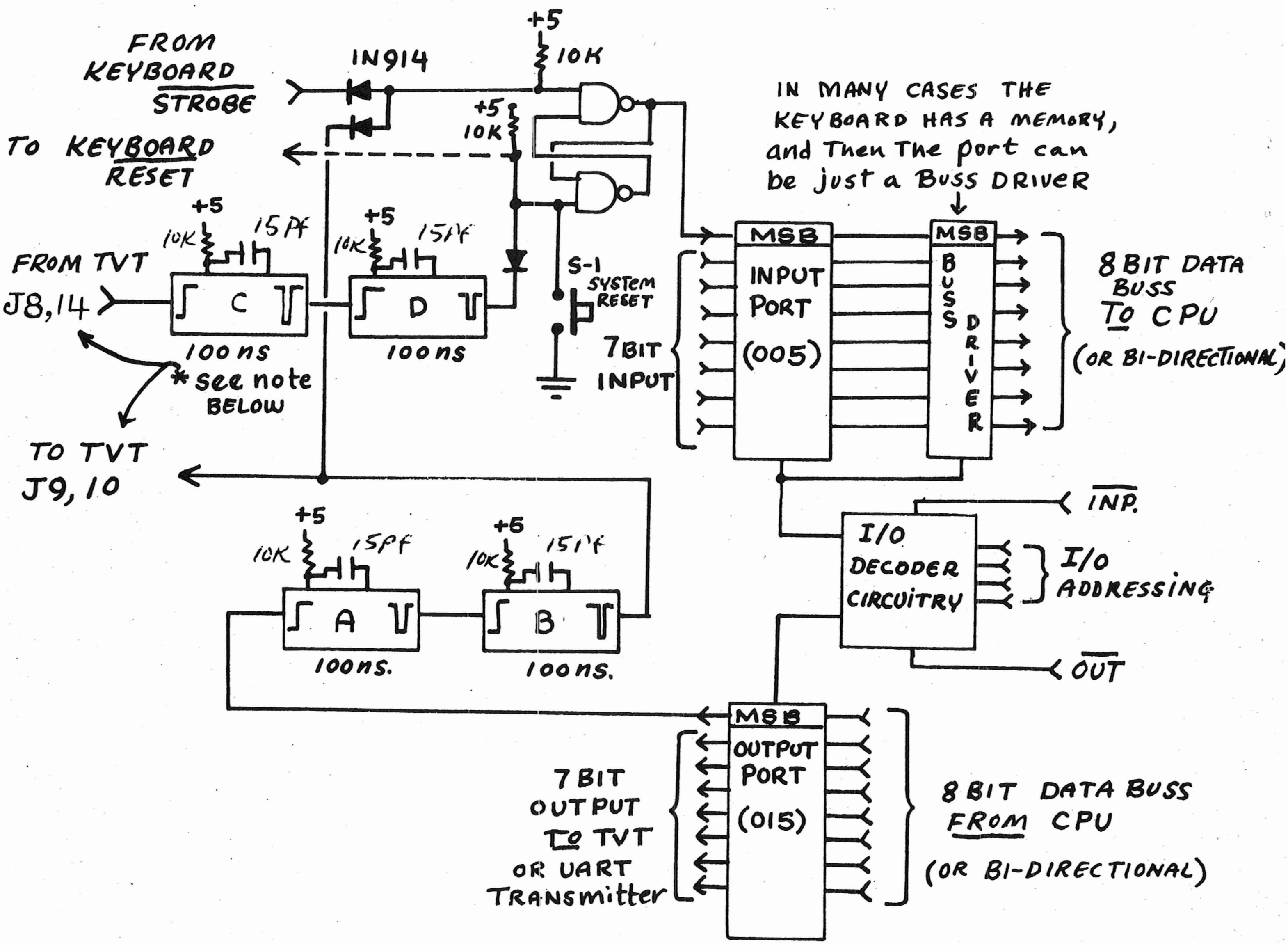
A loop causes the computer to keep searching INPUT port until the MSB goes HIGH, indicating input data is ready. The software immediately starts to 'echo' the input by calling the OUTPUT routine. This routine starts by 'idling' the output port (insuring that MSB starts out LOW). Then the character is output, which will always cause the MSB to go HIGH. Hardware detects the MSB low -to- high transistion, causes a 100 nanosecond delay, and then delivers a LOW pulse to receiving device, causing data on output port to be loaded into the device. At the same time we insure that MSB of input port is high by coupling this pulse to the RS flip flop. (This is done since output can occur from computer-generated data as well as from keyboard data, and we use the MSB of the input port to tell us when we are done). When receiving device has processed information and is ready to accept a new input a low-to-high transition is sent to the C one-shot which creates a 100 nanosecond delay. At the end of this delay the D one-shot creates a RESET pulse for the INPUT port MSB. All this time the software has been looping, waiting for the MSB to go low. When the software detects that this has occured the program flow RETURNS...in this case to the INPUT routine, which then strips off MSB from data and checks for a CTRL/A . (a CTRL/A will send you to the Controller).

Note the following as regards the software: We have kept the addresses where the Input and Output routines start the same as those used in the simpler I/O routine. Further, the software from 012371 - 012377 and from 013045 - 013121 remains unchanged, so consult the software for Minimum Monitor for these sections. See page A-2 for a listing of what Monitor routines this software replaces.

PARALLEL I/O HARDWARE FOR TVT OR UART

BY BRO. THOMAS McGanee
FOR MINI MICRO MART

(Includes 'handshaking' for maximum SYSTEM SPEED.)



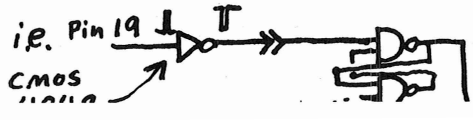
UART = AY-5-1012, Com 2502, 2536 etc..

* If USING A UART, CONNECTIONS ARE AS FOLLOWS:

PIN 19 ⟶
(old J8,14)

Pin 23 ⟵
(TDS)
(old J9,10)

in Addition A UART REQUIRES THE Following:
Connect PIN 18 (RDAR) to "KEYBOARD RESET"
and connect PIN 19 to "KEYBOARD STROBE" INPUT via an INVERTER.



Bro. Thomas McGahee 80 South Sixth St. Columbus, Ohio 43215

Minimum Monitor w/ handshaking

BANK	BYTE	OCTAL	MNEMONICS	FUNCTION
013	000	250	XRA	* OUTPUT ASCII
	001	133	OUT 015	idle PORT
	002	301	LAB	
	003	133	OUT 015	PRINT B
	004	113	INP 005	
	005	074	CPI 200	IF MSB of INPUT PORT
	006	200		IS High, Loop
	007	100	JFC 013004	
	010	004		
	011	013		
	012	250	XRA	
	013	133	OUT 015	idle PORT
	014	301	LAB	/ set up B
	015	007	RET	/ GO HOME
	016	013	RFZ	-- ~~INPUT~~ INPUT continued from 013042
	017	104	JMP 010211	CTRL/A causes a Return
	020	211		To CALCULATOR Controller.
	021	010		
	022	000	not used (HLT)	
	023	113	INP 005	* INPUT ASCII
	024	310	LBA	
	025	074	CPI 200	LOOP UNTIL DATA
	026	200		IS VALid
	027	150	JTC 013023	(MSB High)
	030	023		
	031	013		
	032	106	CAL 013000	ECHO CHARACTER
	033	000		
	034	013		
	035	044	NDI 177	STRIP OFF MSB
	036	177		
	037	310	LBA	B and A HAVE DATA

Bro. Thomas McGahee 80 South Sixth St. Columbus, Ohio 43215

BANK	BYTE	OCTAL	MNEMONICS	FUNCTION
013	040	074	CP I 001	
	041	001		IS IT A CTRL/A
	042	104	JMP 013 016	continued elsewhere.
	043	016		
	044	013		
	045			
	046		the REST IS THE SAME	
	047		AS THE OTHER	
	050		MINIMUM SOFTWARE	
	051		ROUTINES.	
	052			
	053			
	054			
	055			
	056			
	057			
	060			
	061			
	062			
	063			
	064			
	065			
	066			
	067			
	070			
	071			
	072			
	073			
	074			
	075			
	076			
	077			