Brother Thomas McGahee 80 South Sixth St. Columbus Ohio 43215

HOW IT WORKS: PARALLEL I/O for MOD-8

Since the INPUT routine uses the OUTPUT routine as part of its over-all functioning, I will discuss the software/hardware operation in terms of an INPUT operation.

For the purpose of this discussion I am assuming that the receiving device is a TVT-II and the transmitting device is a parallel output Keyboard.

The software starts off by going into a HALT condition. It will stay in this state until a Keyboard $\overline{\text{STROBE}}$ causes the TTY INT (Interrupt) line to be pulsed HIGH. This interrupt will result in an $\overline{\text{IBS}}$ pulse which will jam an LAA instruction which releases the CPU from the HALT state. Coincident with the original Keyboard $\overline{\text{STROBE}}$ a 74100 latches the Keyboard DATA, which consists of 7 data bits and an MSB that is always HIGH. Software now reads the Keyboard DATA and transfers the DATA to B. (several hundred nanoseconds after the Keyboard $\overline{\text{STROBE}}$ the A and B one-shots cause a Keyboard $\overline{\text{RESET}}$ signal in case the keyboard requires such a signal.)

The INPUT program now calls for an ECHOING of the received character. The software first IDLES the output port. This causes the MSB to be LOW. Then the DATA is sent to the output port, causing the MSB to go HIGH. The software now enters a HALT causing the CPU to HALT. Meanwhile, back at the interface, one-shots C and D have caused a couple hundred nanoseconds of delay and presented a negative-going $\overline{\text{STROBE}}$ signal to the TVT-II. Inside the TVT-II circuitry the DATA is latched and various operations performed until somewhere along the line the latches are finally dumped. We detect the rising edge of this dump command (TVT Input Accepted command, available at J8, 14),and one-shots E and F cause a couple of hundred nanoseconds of delay followed by a negative-going pulse to a 7400 gate causing a positive pulse to be generated on the TTY INT line.

This Interrupt causes $\overline{IBS}$ to go low, resulting in an LAA instruction being jammed into the CPU, releasing it from its HALT condition. Software now causes the output port to be IDLED. DATA is transferred to A and we return to the INPUT routine. Software now strips off the MSB and checks for a CTRL/A. A CTRL/A causes a jump to the Controller. Any other character causes a return to the calling program.

It should be noticed that one-shots A and B protect us against accidental keyboard depressions. If the keyboard did not have this auto-reset feature its strobe could get 'stuck' in the low state on any keyboard depressions occuring when the interface was not under computer control.

Note also that the COLD START software starts off by idling the output port and then feeding in a do-nothing CTRL/X. This doesn't print anything, but it clears all interface and TVT-II circuits.

The COLD START program has one addition over the old Monitor COLD START. When the COLD START is used you will get a carriage-return and line feed followed by

MONITOR-8P --------

This indicates that you are using the PARALLEL I/O OVERLAY. Hitting CTRL/A , and most MONITOR routines that cause a return to 003000,(the Controller), will simply cause the usual -------- to be printed. All routines that call a RST 000, however, will print the MONITOR-8P ------ message.

Since the parallel I/O routines used less space than the original serial routines I have taken the liberty to use some of the freed space to write a NEW TIMER routine.

CAL 000164 gives you the NEW TIMER routine. This routine requires that the calling program provide D and C values. The timing delay can be determined by the following formula:

Time Delay in Microseconds = 65*(D+256*(C))

where C and D are expressed as decimal numbers. For instance, if you LCI 010 , then C= 8 in decimal notation.

A typical calling program would be

```
LCI 005
LDI 100
CAL 00164
```

and would result in 87360 Microseconds delay (.08736 seconds).

Note that low values for delay will be slightly longer than the formula would indicate. This is because the non-loop portions of the program take some time too, and in very short delay programs this small non-recurring time may be a significant part of the total delay. For instance, if both C and D are set for 000 , it will still take some time for the program to Return you to the calling program.

The actual delay constant (the 65 in the formula) will vary slightly from computer to computer, but will be quite stable for a given computer. Using this NEW TIMER you can get delays from 65 microseconds to 4.27 seconds directly. Besides the wide range of delays (available in 65 microsecond steps!), this particular TIMER allows relatively simple computation of the delay.

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

MONITOR-8P PARALLEL I/O

BANK	BYTE	OCTAL	MNEMONICS	FUNCTION
000	000	250	XRA	* RST 000 = COLD START
—	001	133	OUT 015	/ "IDLE" OUTPUT PORT
	002	016	LBI 201	/ LOAD B WITH CTRL/X
	003	201		/ WHICH IS NON-PRINTING
	004	025	RST 020	/ AND INITIALIZE I/O
	005	104	JMP 000105	/ HARDWARE. THEN JUMP TO
	006	105		/ 000105 TO CONTINUE...
	007	000		
	010	104	JMP 007000	* RST 010 = CASSETE
	011	000		/ JUMP TO CASSETTE ROUTINE.
	012	007		
	013	016	LBI 215	* CARRIAGE RET. + LINE FEED
	014	215		/ LOAD B WITH CR
	015	025	RST 020	/ AND DO IT.
	016	016	LBI 212	* LINE FEED
	017	212		/ B HAS LF code CONTINUE...
	020	250	XRA	* RST 020 = PARALLEL OUTPUT
—	021	133	OUT 015	/ "IDLE" OUTPUT PORT,
	022	301	LAB	/ A HAS CHARACTER FROM B
—	023	133	OUT 015	/ SO STUFF IT INTO PORT.
	024	000	HLT	/ WAIT FOR INTERRUPT.
	025	104	JMP 000064	/ Continued at 000064
	026	064		
	027	000		
	030	000	HLT	* RST 030 = PARALLEL INPUT
—	031	113	INP 005	/ WAIT FOR INTERRUPT. THEN
	032	310	LBA	/ GET INPUT, GET IT INTO B...
	033	025	RST 020	/ AND ECHO IT.
	034	104	JMP 000075	/ Continued at 000075
	035	075		
	036	000		
	037	000	HLT (NOT USED)	

— Indicates an IMP 005 or OUT 015 which you might wish to change to an IMP 000 (code 101) and an OUT 012 (code 125).

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

MONITOR-8P

BANK	BYTE	OCTAL	MNEMONICS	FUNCTION
000	040	006	LAI 177	* RST 040 = RUBOUT TEST
	041	177		/ COMPARE B WITH
	042	271	CPB	/ RUBOUT. IF NO MATCH,
	043	013	RFZ	/ RETURN. BUT ON A
	044	016	LBI 337	/ MATCH, LOAD B WITH
	045	337		/ A BACKSLASH
	046	025	RST 020	/ PRINT IT
	047	007	RET	/ AND RETURN.
	050	035	RST 030	* RST 050 = "E" SEARCH
	051	274	CPE	/ GET INPUT & COMPARE
	052	053	RTZ	/ WITH "E". RETURN IF A
	053	104	JMP 000050	/ MATCH - OTHERWISE TRY
	054	050		/ AGAIN.
	055	000		
	056	000	HLT (NOT USED)	
	057	000	HLT (NOT USED)	
	060	104	JMP 000103	* RST 060 = BREAKPOINT
	061	103		/ JUMP TO BREAKPOINT.
	062	000		
	063	000	not used	
	064	250	XRA	-- continued from 000025
	065	133	OUT 015	/ NOW IDLE PORT
	066	301	LAB	/ GET B INTO A
	067	007	RET	/ AND GO HOME
	070	030	IND	* RST 070 = TIMING LOOP
	071	110	JFZ 000070	/ Increment D UNTIL IT
	072	070		/ REACHES 000,
	073	000		/ THEN
	074	007	RET	/ RETURN HOME.
	075	044	NDI 177	-- Continued from 000034
	076	177		/ STRIP OFF MSB
	077	074	CPI 001	/ AND CHECK FOR CTRL/A

MONITOR-8P

BANK	BYTE	OCTAL	MNEMONICS	FUNCTION
000	100	001		
	101	013	RFZ	/ RETURN IF NOT CTRL/A.
	102	104	JMP 003000	/ CTRL/A CAUSES A JUMP
	103	000		/ TO CONTROLLER.
	104	003		
	105	106	CAL 000013	-- Continued from 000005
	106	013		/ GET A CR+LF.
	107	000		
	110	056	LHI 000	/ SET MEMORY TO 000130,
	111	000		
	112	066	LLI 130	/ THE START OF OUR
	113	130		/ COLD START MESSAGE.
	114	317	LBM	/ RETRIEVE A CHARACTER
	115	025	RST 020	/ AND PRINT IT.
	116	060	INL	/ INCREMENT MEMORY LOW ORDER.
	117	006	LAI 145	/ COMPARE WITH 145.
	120	145		
	121	276	CPL	
	122	110	JFZ 000114	/ IF NOT 145, THEN GET
	123	114		/ NEXT CHARACTER.
	124	000		
	125	104	JMP 003003	/ WHEN ALL DONE GO TO
	126	003		/ CONTROLLER TO PRINT 8 DASHES
	127	003		/ AND ALL THAT STUFF.
	130	240	SPACE	
	131	240	SPACE	/ 000130 - 000144 CONTAIN
	132	315	M	/ COLD START MESSAGE
	133	317	O	/ " MONITOR-8P ".
	134	316	N	
	135	311	I	
	136	324	T	
	137	317	O	

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

MONITOR-8P

BANK	BYTE	OCTAL	MNEMONICS	FUNCTION
000	140	322		R
	141	255		—
	142	270		8
	143	320		P
	144	240		SPACE
	145	000		/ 000145 - 000163
	146	000		/ are not USED
	147	000		
	150	000		
	151	000		THE FORMULA FOR THE
	152	000		NEW TIMER IS
	153	000		Time in microsec. = 65*(D + 256*(C))
	154	000		↑
	155	000		THIS CONSTANT MAY VARY
	156	000		SLIGHTLY FROM COMPUTER-
	157	000		TO-COMPUTER.
	160	000		
	161	000		
	162	000		
	163	000		
	164	030	IND	* NEW TIMER
	165	020	INC	/ INCREMENT C and D
	166	031	DCD	/ DECREMENT D
	167	110	JFZ 000 166	/ LOOP IF NOT 000
	170	166		
	171	000		
	172	021	DCC	/ DECREMENT C and if
	173	110	JFZ 000 166	/ not 000, do D LOOP
	174	166		/ 256 TIMES
	175	000		
	176	007	RET	/ RETURN WHEN DONE.
	177	035	RST 030	* FETCH & TEST FOR OCTAL

THE REST IS IDENTICAL TO REGULAR MONITOR SOFTWARE (See MOD 8 MANUAL)

PLEASE NOTE that my software listings only show the first 128 words of the OVERLAY. The rest of the PROM must be loaded with the regular MONITOR program, just as it is found in the original MONITOR software listing.

Since PROMS are enabled by banks, consisting of 256 words, you must use a PROM with at least 256 words.

ALSO, the user must include circuitry that will DISABLE the MONITOR 8316 ROM whenever the OVERLAY is selected. (this can usually be done with no extra logic by tying one of the 8316's CHIP SELECT inputs to the same $\overline{\text{SELECT}}$ line that will be used to enable the PROM with the OVERLAY.

Following is a detailed set of instructions for connecting up the 'handshaking' and data lines of the interface.

Proposed C-Mod 8-2P.
combines RESTART hardware and Parallel I/O hardware (with handshaking ability).

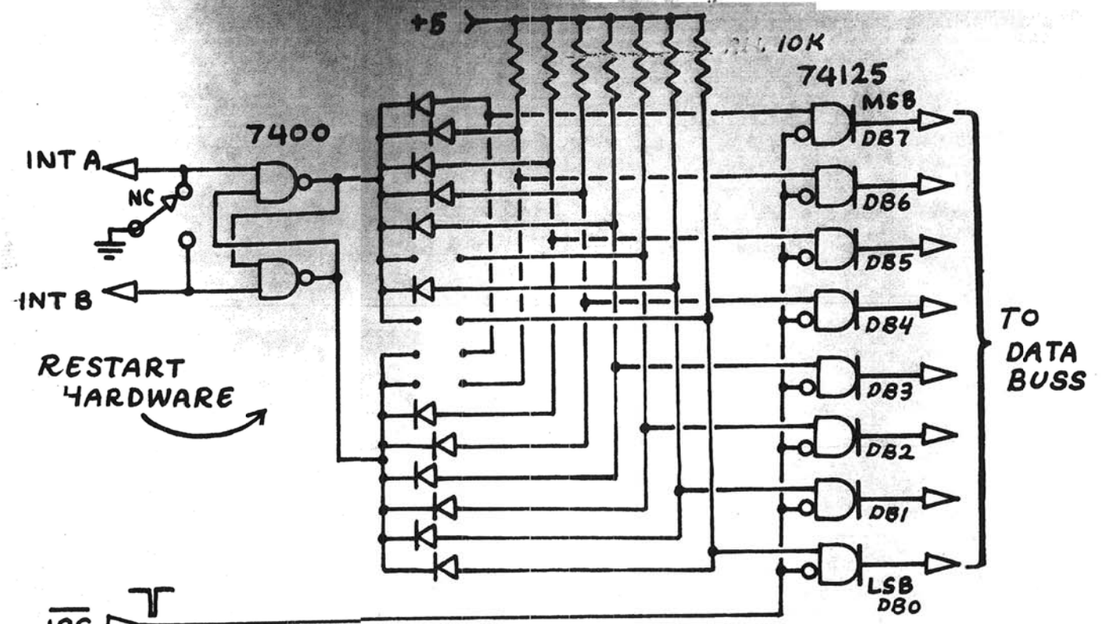
By BRO. Thomas McGahee
February, 1976

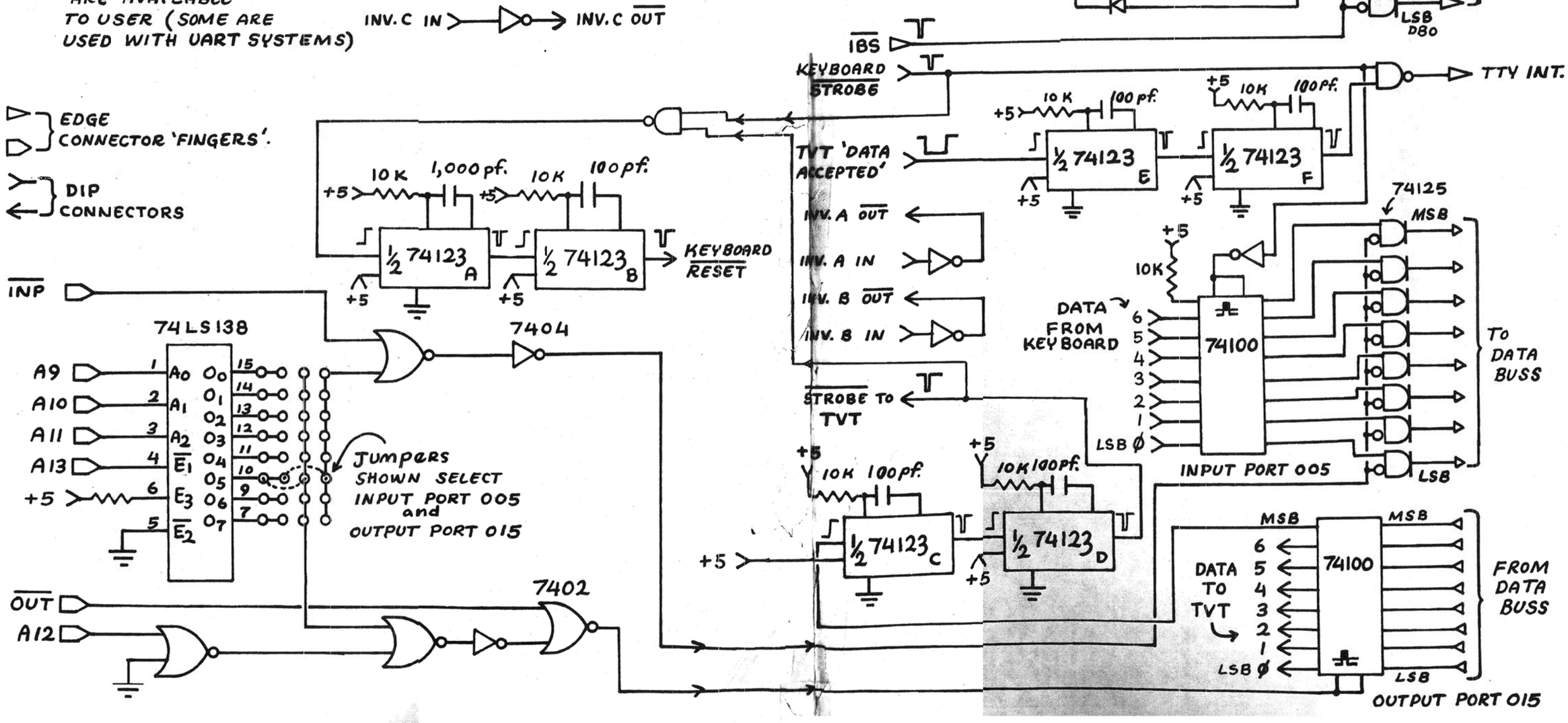
UNUSED GATES ARE AVAILABLE TO USER (SOME ARE USED WITH UART SYSTEMS)

INV. C IN → INV. C $\overline{\text{OUT}}$

EDGE CONNECTOR 'FINGERS'.

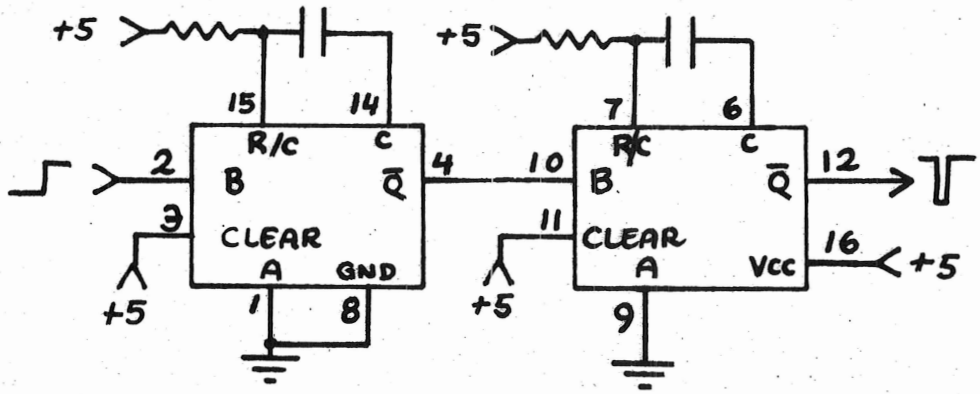
DIP CONNECTORS





Note: with the exception of the capacitor values used, each pair of 74123 one-shot functions is wired identically.

They are set up to trigger on the rising edge and generate a negative-going pulse. Each pair of one-shot functions is contained within a single IC package.



USING THE C-MOD 8 2P WITH A TVT-II & ELECTRONIC KEYBOARD

NOTE: Inverters are available to the user to invert any signals that may not be compatible with his Keyboard.

KEYBOARD $\overline{\text{STROBE}}$: This is a negative-going signal sent by the Keyboard to indicate that a key has been depressed.

KEYBOARD $\overline{\text{RESET}}$: This is a delayed signal developed by the interface to give a negative pulse to reset keyboards that need a reset signal.

DATA FROM KEYBOARD : These seven lines accept Keyboard Data and send it to the 74100 Data Latch which holds the data after the keyboard is reset. (Note that SWTP numbers their lines from 1-7 instead of 0-6).

$\overline{\text{STROBE}}$ TO TVT : This is a negative going pulse that is applied to J9, 10 to tell TVT to accept Data. (TVT must have its strobe jumper set up for negative going pulse.)

TVT DATA ACCEPTED : This connects to TVT-II J8, 14 . When the TVT generates a signal to dump its internal latches it develops a signal on J8, 14 which goes low and then returns to its high position. When this signal returns high it triggers one-shots E and F which create a delay and then generate a TTY INT signal to inform CPU that the data has been processed. You may wish to increase the value for the capacitor on the E one-shot if you experience any difficulty with computer-operated cursor controls: The TVT-II circuitry dumps Control characters earlier than other characters, since they are not stored in TVT memory !

DATA TO TVT : These seven lines provide the proper data to the TVT. Note that SWTP TVT-II numbers their lines 1-7 but we have followed the more common convention of numbering from 0-6.

No parallel interface board should be required on the TVT-II itself, since all the handshaking hardware has been provided on the MOD-8 2P board. By using handshaking the Computer and the TVT can communicate data at a speed that is just a little slower than the maximum theoretical speed of the TVT-II. Actual speed of data transfer is a function of several factors, both hardware and software related.

****

USING THE C-MOD 8 2P WITH UARTS (pin #s are for COM 2502 etc.)

Note that Uart clock speed is independant of the software and hardware, since the Uart provides the necessary 'handshaking' signals needed to provide full communication between UART and computer.

KEYBOARD $\overline{\text{STROBE}}$: UART pin # 19 provides positive-going strobe indicating data has been assembled in parallel form in the receiver section of the UART. INVERT this signal from pin # 19 using one of the inverters available and connect the output of this inverter to the KEYBOARD $\overline{\text{STROBE}}$ input.

KEYBOARD $\overline{\text{RESET}}$: This connects to UART pin # 18 and is used to reset the receiver of the UART once the data has been accepted by the interface.

DATA FROM KEYBOARD : UART pins # 6-12 connect here. numbering of bits may vary from those used in my diagrams, but you will get them connected right if you start by connecting the LSB to the port LSB and work your way up until you have connected seven data lines.

$\overline{\text{STROBE}}$ TO TVT : This connects to UART pin # 23, and it goes low temporarily to indicate to UART that parallel data is ready to be transmitted.

TVT DATA ACCEPTED : Connect this to UART pin # 24. As soon as the UART finishes the transmission of a character this will go HIGH triggering one-shots E and F into creating a short delay and then a pulse that will result in a TTY INT being generated.
DATA TO TVT : tie UART data lines to output port in proper order.

IF USING AN ELECTRONIC KEYBOARD FOR SENDING IN PARALLEL, and a UART for receiving, then wire keyboard up as explained in section on TVT use, and wire the part of the UART that is being used to transmit to TTY or MODEM according to the instructions for the UART.

In a similar manner you can also transmit data in serial to a UART, feed UART's parallel output to interface, and send parallel data out to a TVT or similar parallel device. Naturally in such a hybrid approach ECHOING will proceed at a pace equal to the speed of the slower device. However, if a slow speed input is used, this will not slow down the output when it is NOT echoing, but is simply under computer control. Thus you may use a slow speed TTY to get data in at say 10 characters per second, and it will echo at this slow speed on the TVT, but when the computer takes over you can fill an entire screen (512) characters in a few seconds or less.

When using a UART, you may change the BAUD rate without having to make any adjustments in the interface presented here. (Note also that transmitter and receiver clocks of UART may operate at different speed with no problem since the 'hand-shaking' will create 'stalls' if needed !)