ALTAIR 8800 MACHINE/ASSEMBLY LANGUAGE PROGRAM CODING SHEET

Program Name: BASIC Keyboard-ACR Input Routine, #3-12-763 Page 1 of 10

Programmer: Christopher J. Flynn Date: 3/3/76

Address:

Program Length in Bytes: 64 Language: x Machine ___ Assembl

Other Information:

TAG	MNEMONIC	ADDRESS	OCTAL CODE	EXPLANATION
START	PUSHH	XXX, 300	345	Save H,L
	PUSHB	301	305	Save B,C
	LXIH	302	041	Point to "locate CR" switch
		303	334	
		304	XXX	
SENSE	IN	305	333	Read sense switch
		306	377	
	RAL	307	027	Bit 7 to Carry - test A15
	JC	310	332	A15 is set. GO TO ACRIN
		311	335	
		312	XXX	
KBDIN	MVIA	313	076	Set ACC to 1
		314	001	
	MOV	315	167	Set "locate CR" switch to 1
KBDCSW	IN	316	333	Read keyboard status word
		317	000	
	ANI	320	346	examine data ready bit
		321	002	
	JZ	322	312	Not ready. Go to KBDCSW
		323	316	

TAG	MNEMONIC	ADDRESS	OCTAL CODE	EXPLANATION
		XXX, 324	XXX	
	IN	325	333	Read keyboard data
		326	001	
	ANI	327	346	Select bits 0-6
		330	177	
	POPB	331	301	Restore B,C
	POPH	332	341	Restore H,L
	RET	333	311	Pass kbd character to BASIC
SWITCH		334	001	If switch is 1, search for CR
ACRIN	CALL	335	315	Obtain data word from ACR
		336	367	
		337	XXX	
	MOVBA	340	107	Save ACR data word in Reg B
	MOVAM	341	176	Load "locate CR" switch
	RAR	342	037	Bit 0 to carry -test for seek or read
	JNC	343	332	Bit 0 is 0 go to read mode
		344	361	
		345	XXX	
ACRSEEK	MOVAB	346	170	Restore ACC from Reg B
	CPI	347	376	Does ACC contain a CR
		350	015	
	JNZ	351	302	No, keep looking for the 1st CR
		352	335	
		353	XXX	
	SUBA	354	227	Yes, set "locate CR" switch to 0
	MOVMA	355	167	
	JMP	356	303	Now go read data
		357	335	

rogram Name: BASIC Keyboard-ACR Input Routine #3-12-763 Page 3 of 10

TAG	MNEMONIC	ADDRESS	OCTAL CODE	EXPLANATION
		XXX, 360	XXX	
ACRREAD	MOVAB	361	170	Restore ACC from Reg B
	ANI	362	346	Select bits 0-6
		363	177	
	POPB	364	301	Restore B,C
	POPH	365	341	Restore H,L
	RET	366	311	Pass ACR character to BASIC
ACRCSW	IN	367	333	Read ACR status word
		370	006	
	RRC	371	017	Bit 0 to carry - te t for data ready
	JC	372	332	Not ready. GOTO ACRCSW
		373	367	
		374	XXX	
	IN	375	333	Ready. Read ACR character
		376	007	
	RET	377	311	Pass character to caller

ALTAIR 8800 MACHINE/ASSEMBLY LANGUAGE PROGRAM CODING SHEET

Program Name: Modified BASIC Output Routine #1 #3-12-763 Page 4 of

Programmer: Christopher J. Flynn Date: 3/3/76

Address: 2601 Claxton Drive, Herndon, VA 22070

Program Length in Bytes: 13 Language: X Machine ____ Asse

Other Information: Assumes output device baud rate is less than ACR baud rate.

Code applies to unmodified 88-SIO board.

TAG	MNEMONIC	ADDRESS	OCTAL CODE	EXPLANATION
TTYCSW	IN	XXX, 260	333	Read output device CSW
		261	000	
	ANI	262	346	Select device ready bit
		263	001	
	JZ	264	312	Not ready. GOTO TTYCSW
		265	260	
		266	XXX	
	POPA	267	361	Restore ACC
	OUT	270	323	Send contents of ACC to output device
		271	001	
	OUT	272	323	Send contents of ACC to ACR
		273	007	
	RET	274	311	Back to caller

ALTAIR 8800 MACHINE/ASSEMBLY LANGUAGE PROGRAM CODING SHEET

Program Name: Modified BASIC Output Routine #2 #3-12-763 Page 5 of 10

Programmer: Christopher J. Flynn Date: 3/3/76

Address: 2601 Claxton Drive, Herndon, VA 22070

Program Length in Bytes: 12 Language: X Machine ___ Assembly

Other Information: Assumes output device baud rate is greater than or equal to ACR baud rate.

TAG	MNEMONIC	ADDRESS	OCTAL CODE	EXPLANATION
BEGIN	IN	XXX, 260	333	Read ACR status
		261	006	
	RLC	262	007	Move device ready bit to carry
	JC	263	332	Not ready. GOTO BEGIN
		264	260	
		265	XXX	
	POPA	266	361	Restore ACC
	OUT	267	323	Send contents of ACC to ACR
		270	007	
	OUT	271	323	Send contents of ACC to output device
		272	001	
	RET	273	311	Back to caller

Saving BASIC Programs with the ACR

Introduction

Users of the cassette versions of BASIC and users lacking paper tape equipment have only limited capability to save debugged BASIC programs for later use. 4K users are in a peculiar situation in that there is no provision at all for saving programs short of investing in a paper tape reader/punch. In 8K BASIC (cassette version) there is the capability to save programs (the CSAVE command) on cassette tape and to later reload programs (CLOAD). When programs are stored in this manner, they are written to tape in internal form rather than in source (ASCII character) form. This method may impose a release dependency on stored programs. That is, programs CSAVEd in one version of BASIC may or may not CLOAD in a different version of BASIC. Another difficulty with CSAVE and CLOAD is the potential problem in merging program segments from multiple tapes to create a composite program. Finally, BASIC compatible tapes are not easily created or listed off-line.

This article describes a method of using the ACR cassette interface together with patches to BASIC's input and output subroutines in order to simulate a paper tape reader and punch. Thus, any information which can be displayed on the terminal (programs, subroutines, comments, DATA statements, and so on) can be stored on cassette tape and retrieved at some point later in time

Since the system described herein stores data in source form, several advantages in operational flexibility are immediately obtained. Barring any changes in BASIC syntax, release dependency is minimized. Programs saved under 4K 3.2 BASIC should load and execute properly under future releases. BASIC statements may be loaded from several tapes to create a larger program. For example, a main program may be loaded from one tape, any required subroutines may be loaded from a tape containing a subroutine library, and finally, DATA statements may be read from yet another tape. Using the proposed system, it is also possible to create and list BASIC tapes without bringing up BASIC. Some of the text editting systems now available can be useful for off line data preparation.

The proposed system does have a few disadvantages. At the present time, it is not possible to name files stored on tape. The tape recorder index counter readings must be used to locate files. As will be shown later, the proposed system does not pack characters on the tape as closely as possible. Normally, this will not be of too great significance. The most severe criticism of the proposed method is that there is some question as to the transferability of tapes made in this manner to users with other terminal configurations.

Method

BASIC handles input and output to the terminal by means of input and ourput subroutines which are tailored to the particular I/O interface boards supporting the terminal. These subroutines provide a logical starting place for any attempt to develop effective cassette software.

One subroutine in BASIC is responsible for terminal output. Whenever BASIC attempts to print a character, the output subroutine is invoked. The output subroutine checks the output device status and outputs a character when the device is ready. A simple modification to this subroutine causes a character to be written to the ACR every time a character is printed on the terminal. Thus, if the tape recorder is in the record mode all information (whether typed by the user or printed by BASIC) will be stored on the cassette

Operation

Once BASIC has been loaded and the new input and output routines established, saving and retrieving programs becomes very straightforward.

A. Saving a BASIC program

To store a program on tape follow the steps enumerated below:

1. Type in the program. Test it to make sure it is fully operational.
2. Add a dummy line at the end of the program (e.g. 999 REM. . .).
3. Type LIST, but do not hit RETURN.
4. Set up the recorder in the record mode. Write down the footage meter reading.
5. Record at least 10-15 seconds of leader.
6. Now type RETURN. The program will be printed on the terminal and recorded on the tape recorder. Note that carriage return marks the beginning of the tape file.
7. After the program has finished printing, allow the ACR to write at least 10-15 seconds of trailer.
8. Stop the tape recorder.
9. Write down the final footage meter reading.

B. Retrieving a BASIC program from tape

To access a program which is stored on tape, perform the following steps:

1. Prepare the tape recorder for playback operation.
2. Using the footage meter, locate the desired file and stop the recorder in the leader.
3. At this point you may wish to type NEW at the terminal if you are loading a main program. Otherwise, the program statements read from tape will be merged with the BASIC program currently in memory.
4. Turn on the A15 sense switch to signify that system input will come from the ACR.
5. Type RETURN on the keyboard - this completes the changeover from t keyboard read routine to the ACR read routine. The keyboard should now be insensitive to further input. Furthermore, the INP console light should be on indicating that BASIC is expecting ACR input.
6. Start the tape recorder making sure that the tape is positioned in leader. BASIC will scan the tape until the first carriage return is encountered which signifies the beginning of the tape file.
7. After the beginning of the file has been located, BASIC statements will be read from tape and printed or displayed on the output terminal.
8. Watch for the dummy statement (999REM...) at the end of the program. When this is encountered, turn off A15 as soon as possible in order to switch BASIC back to keyboard entry.
9. If A15 is not turned off in time, BASIC will be "stuck" in the ACR read mode. If this should happen, keep A15 in the off posistion,

Saving BASIC programs is then a matter of selectively (and manually) turning on and off the tap recorder.

The modification to the output subroutine presumes no alteration of ACR adjustments; the ACR remains set at 300 baud. If the output device is a teletype machine operation at less than 300 baud, the extra write instruction in the output subroutine will not substantially degrade printing speed. Indeed, the mismatch in baud rates is the reason that this method does not achieve optimum packing of characters on the cassette tape. Depending on the exact baud rates, there will be a delay of several milliseconds after the ACR character has been written and before the Teletype has finished printing the character. This delay, however, insures that during playback the ACR will not overrun the Teletype. If, on the other hand, an output device is used which is significantly faster than 300 baud (e.g.a TV typewriter using a paral I/O board), then the output routine, modified as above, will limit the data transfer rate from the computer to the output device. If the degradation is too severe, it may be possible to selectively enable and disable the ACR output logic in a manner similar to the input routine discussed below.

Depending on the version of BASIC, there may be several places in the interpreter where a check for terminal input is made. Only one of these routines, however, is used for accepting terminal data. The other routines ar Control C checks used to interrupt a running program.

BASIC's input routine is similar to the output routine. The device statu is checked. If the device is ready, a character is read from the device and passed to BASIC for processing. Otherwise, BASIC waits until an input signal is sensed.

The modifications to BASIC's input routine are more involved that the output routine. Essentially, however, the modifications consist of checking a sense switch on the CPU front panel and then reading from the keyboard or ACR depending on the sense switch setting. To retrieve data from tape then, the only action that is required is to turn on the sense switch and to start the tape recorder. Note that since the ACR has replaced the keyboard as the input device (as long as the sense switch is set) all characters stored on tape will appear on the output device as though they were input from the keyboard.

The timing considerations discussed earlier also apply during playback. Tapes recorded and played back on the same system should be processed properly. A potential problem exists, however, with trying to play back a tape created on another user's system if the other user employs a different speed te terminal. For example, a tape made on parallel I/O board TV typewriter system will most likely not have the several millisecond delay between ASCII character Attempting to print such a tape on a slower Teletype will cause the ACR to overrun the Teletype. To remedy such a situation where there is a timing mismatch, simply NOP the output device status checking code, read in the proble tape, ignore the gibberish that is printed, and restore the output routine. Most probably, BASIC will have read the tape properly even though the character could not be printed.

rewind the tape back slightly into the data, and play the tape again. As soon as one character is read from the tape, BASIC will revert back to the keyboard entry mode.

Modifying BASIC

The modification required to BASIC consist of adding a new input subroutine and a new output subroutine and modifying BASIC's existing I/O routines to CALL these new routines. Accompanying sheets contain the machine language code for the new routines and patches for the cassette version of 3.2 4K BASIC.

Refer to the code for new I/O routines. The sections of code labelled KBDCSW and TTYCSW handle the terminal input and output devices respectively. In the example shown, keyboard input is accepted via an 88-PIO parallel I/O board. Terminal output, on the other hand, is performed via an early version serial board. The important point is that the KBDCSW and TTYCSW routines must be tailored to the specific devices being used. Any doubt about the I/O programming can be resolved by loading BASIC and examining its terminal I/O routines.

Note that two output routines have been included in the documentation. Choose one of them according to the baud rate of the output terminal device. The new output routine is designed to capitalize on speed difference between the ACR and terminal. By outputting to the slower device first and by performing status checking on the slower device, the assumption can be made that the faster device will always be ready to output. Therefore, status checking code for the faster device can be eliminated. If, for some reason, satisfactory results are not achieved, modify the new output routine to check the status of both the ACR and the terminal before writing.

As shown on the accompanying documentation, BASIC's I/O routines are replaced with CALL instructions to the new routines. Teh locations shown are applicable to 4K BASIC Version 3.2. A recent issue of "Computer Notes" suggested a method for locating these I/O routines. An easy way to find BASIC's I/O routines consists of loading BASIC and then stopping BASIC while it is printing and stopping it again while it is waiting for terminal input. In each case, note the locations and memory contents when BASIC is stopped. Then, using the EXAMINE switch, find the device status checking IN instruction for each routine. These are the locations that will be replaced by CALLs to the new routines.

Listed below are stops to be followed in order to bring up BASIC and apply the necessary modifications:

1. Toggle in or load from tape the new I/O routines. Locate these routines in a high page of memory and above the area used by the bootstrap loader.
2. Load BASIC according to normal procedure.
3. Stop BASIC as soon as the initialization dialogue begins. Note the location where BASIC was stopped.
4. Replace BASIC's I/O routines with CALLs to the new I/O routines just loaded.
5. Restart BASIC from the location where it was stopped. If BASIC was in the old output routine, restart it from the newly inserted CALL

statement.

6. Complete the initialization dialogue. Do not allocate all of the memory to BASIC or the new I/O routines will be overlaid.

Conclusion

This article has described a simple software interface to BASIC which effectively simulates a paper tape reader and punch with the result that BASIC' capability in the area of off=line data storage is greatly enhanced.

Although the system was originally intended to provide a source program storage facility, other applications suggest themselves since any data that can be entered via a keyboard can also be entered via tape. Consider the following BASIC program.

```
10   FOR I = 1 to 10
15   PRINT 900 + I; "DATA"; 3.14159*I
20   NEXT I
```

This program prints a series of DATA statements. If the DATA statements are stored on cassette tape, they can be accessed later by another BASIC progr The ACR, then, may serve as a convenient work file for communicating temporary results between programs.

An advanced user may carry the work file principle a step further. With the string capabilities of 8K BASIC, it is possible to write a single compiler. Instead of generating machine code, the compiler could generate BASIC statements and save them on tape for later execution.

There are, in the end, a potentially unlimited number of uses for the ACR data storage system presented in this article.

MODIFICATIONS TO 4K 3.2 BASIC

The following patches to BASIC are made after BASIC has been loaded and started and before the initialization dialogue has been completed. Do not apply these patches and then start BASIC from location zero or the patches will be overlaid.

Output Routine

Location	Old Contents	New Contents	
003, 167	333	315	Call new output rtn
, 170	000	260	
, 171	346	XXX	
, 172	001	311	Back to BASIC

Input Routine

Location	Old Contents	New Contents	
003, 202	333	315	Call new input rtn
, 203	000	300	
, 204	346	XXX	
, 205	002	311	Back to BASIC