NEW BASIC 5025 & 5510 COMPILERS (Review by Steve & Phil Browne)

There are now some tape based Basic compilers on sale for the Sharp computers. Most are quite good,but most lack good string and data handling and the abilty to handle decimal numbers.

The Hu 24k Basic compiler(reviewed in suc vol3 no3)was the first compiler to offer good string handling and it was also able to compile nearly all the Hu commands.But all this came at the expense of available memory,with only 4k left to load and compile a program.

The ideal basic compiler would be one that could handle strings,data,decimal and negative numbers,and also be 100% 5025 or 5510 compatable.The compiler should also leave you with enough available memory to load in most of your basic programs and compile them.Well the good news is I have come across just such a compiler.It comes from Germany and there are two versions,one for 5025 basic and one for 5510.For the purpose of this reveiw I will deal with the 5025 version.

COMPILER INSTRUCTIONS

After the compiler has loaded you can load in your 5025 program,you then have the following options.

COMP

This will compile the source program and any error messages will be displayed.If a memory error occurs it means the source program is to long. Programs up to 12k approx should compile ok under this command.After compiling the source program is kept in memory and can be changed and re-compiled.

COMP,D

This command is used if a memory error occurs under the COMP command. Unlike the COMP command the source program is not kept in memory after compiling,thus making extra memory available.Programs upto 20k approx should compile ok under this command.The object program must be saved directly onto tape when using COMP,D.

RUN

This will execute a compiled program.If the program has not been compiled it will compile it first by using the COMP method.A memory error will occur if the sorce program was to large to compile.

COMP,M

This command is used if a memory error occurs under the RUN command.The source program will be compiled and the object program should be saved directly onto tape.If this command is used the run command cannot be used again,so you can only test out your object program by re-loading it from monitor.

COMP"name"

This command compiles program and saves object program to tape.If the COMP,D command has to be used then use COMP,D"name" to save to tape.If the COMP,M has to be used then use COMP,M"name" to save to tape.

It is imperative that when using COMP,D or COMP,M that you save to tape directly,as the source program is not kept and no second chance is given

EXEC

This will load,compile and run.

COMP,R

This will compile and run

The compiler has a interpreter/editor built in,so it is possible to write programs without loading 5025 basic.The interpreter has all the 5025 commands plus a few extra ones to

EXTRA INTERPRETER COMMANDS

DOKE: 16 bit poke
DEEK: 16 bit peek
IF THEN ELSE
AND,OR,XOR

WAIT n: Waits for n seconds
CURSOR x,y
BREAK OFF,BREAK ON
$ML: Entering of Z80 m/c
 $:Entering of hex numbers.eg(sw=$D000)

EDITOR COMMANDS

LOAD: Load in 5025 program
SAVE: Save 5025 source program
VERIFY
LIST,LIST x,LIST x-y,LIST x-,LIST -y
NEW,NEW x,NEW x-y,NEW x-,NEW -y
FINDtext:eg FINDPOKE (This will list all lines with a poke command)
BYE

EXTRA COMPILER INFO

The only commands the compiler appears not to like are certain special 5025 pokes eg:Poke 10167 and pokes to stop breaking and listing ect.These pokes can cause the compiler to crash so it is best to remove them first.Pokes to the video ram and cursor control pokes are ok and so are monitor pokes.

Any pokes or peeks to address 17828 must be changed to 4965.

If you want a very fast screen print then POKE 4967,1

COMPILER TESTS

TEST 1:SHARP DEFENDER 5025 (SIZE=5.1K)

Loaded Sharp defender into the compiler.
FINDPOKE1: Searched for undesirable system pokes and changed POKE 17828,0 to POKE 4965,0
COMP: To compile program.
Program compiled ok and m/c version ran ok
COMP"SHARP DEFENDER M/C": Saved M/C program to tape
SHARP DEFENDER M/C (1200 TO 5E09 SIZE=19.4k)

TEST 2:MAD MAX 5025 (SIZE=12.0K)

Loaded Mad max into the compiler.
Changed POKE 17828,0 to 4965,0
COMP: This command resulted in a Memory Error (program to long)
COMP,D"MAD MAX M/C":Used this command to give me extra memory.As source program will be lost after this i saved directly to tape
MAD MAX M/C (1200 TO 6517 SIZE=21.2K)

TEST 3:SUPER GOLF 5025 (SIZE=19.7K)

Loaded Super golf into the compiler.
COMP,D"SUPER GOLF M/C":As program was so large I did not try the comp command.
Program compiled and ran ok
SUPER GOLF M/C (1200-9796 SIZE=34.1K)

TEST 4:KNIGHTS 36K DEMO (SIZE=20.8K)

Loaded Knights 36k demo into the compiler
COMP,D"DEMO M/C":This resulted in a memory error.
Program to large for compiler.

FINAL SUMMARY

I never thought a compiler as good as this would exist for the Sharp,and it is hard to find fault with it.All 5025 commands appear to compile ok.This means that Sharp owners with little or no knowledge of machine code can write any program in basic and produce a m/c version.For MZ-80A owners there is a 5510 version available.

Unforunately the compiler is not for sale in this country,but i can order them from Germany.The cost is 76DM,which is £20 approx

Basic Compiler for MZ80K.

This Basic compiler can be found on S U C disk volume 103 with the file name of "BASIC COMPILER". I was intrigued to find out what it was because the little documentation I have for S U C volume 103 didn't mention anything about the compiler.

This is a "true" compiler and not an interpreter in disguise but it includes an interpreter.

I have taken the compiler off disk using the S-DOM operating system and have used it from cassette. I have also used S-Dom to make it into a master disk and have run it from disk in that form.

The compiler does not support disk input/output, it follows the cassette Basic characteristics.

The compiler program has apparently come from Germany although all the output is in English.

The compiler announces itself as :- "EDITOR-COMPILER FOR SHARP BASIC PROGRAMS COPYRIGHT 1983 BY UWE MAXIM"

The compiler will compile and/or interpret Basic programs and it will also create a stand alone compiled version on cassette.

Programs may be entered and tested using the interpreter before compilation. This makes the process of program development a lot easier. Programs may be freely interchanged between the compiler and SP-5025 Basic provided the advanced features of the compiler such as DEEK are not used.

The Interpreter error messages are :-

SYNTAX ERROR
MISMATCH ERROR
LINE NOT FOUND ERROR
MEMORY ERROR
LINE TOO LONG ERROR
NAME TOO LONG ERROR
DATA ERROR
READ ERROR
OUT OF DATA ERROR
PRINTER ERROR
NEXT WITHOUT FOR ERROR
FOR ERROR
RETURN WITHOUT GOSUB
16TH FOR ERROR
UNDEFINED FUNCTION ERROR
OUT OF MEMORY ERROR
DIMENSION ERROR
FILE ERROR

The compiler keywords and equivalent tokens in hexadecimal are :-

Token	Keyword	Token	Keyword	Token	Keyword	Token	Keyword
80	REM	81	DATA	82	LIST *	83	RUN *
84	NEW *	85	PRINT	86	LET	87	FOR
88	IF	89	GOTO	8A	READ	8B	GOSUB
8C	RETURN	8D	NEXT	8E	STOP	8F	END
90	ON	91	LOAD *	92	SAVE *	93	VERIFY *
94	POKE	95	DIM	96	DEF FN	97	INPUT
98	RESTORE	99	CLR	9A	MUSIC	9B	TEMPO
9C	USR(	9D	WOPEN	9E	ROPEN	9F	CLOSE
A0	BYE	A1	LIMIT	A2	COMP *§	A3	SET
A4	RESET	A5	GET	A6	INP#	A7	OUT#
A8		A9		AA	DOKE §	AB	
AC	ELSE §	AD	THEN	AE	TO	AF	STEP
B0	><	B1	<>	B2	=<	B3	<=
B4	=>	B5	>=	B6	=	B7	>
B8	<	B9	AND §	BA	OR §	BB	XOR §
BC	+	BD	-	BE	*	BF	/
C0	LEFT$(	C1	RIGHT$(	C2	MID$(	C3	LEN(
C4	CHR$(	C5	STR$(	C6	ASC(	C7	VAL(
C8	PEEK(	C9	TAB(	CA	SPC(	CB	SIZE
CC		CD		CE		CF	↑
D0	RND(	D1	SIN(	D2	COS(	D3	TAN(
D4	ATN(	D5	EXP(	D6	GAU(§	D7	LOG(
D8	LN(	D9	ABS(	DA	SGN(	DB	SQR(
DC		DD	INT(	DE		DF	$ML §
E0		E1		E2		E3	VTAB(§
E4	DEEK(§	E5		E6		E7	
E8		E9		EA		EB	
EC		ED		EE		EF	
F0		F1		F2	CURSOR §	F3	
F4		F5		F6		F7	WAIT §
F7	OFF §	F8		F9	BREAK §	FB	
FC	FIND *§	FD	EXEC *§	FE		FF	

Only the keywords marked "*" can be used in direct mode.

Note those keywords marked "§" do not have any meaning in SP-5025 Basic.

Also unlike SP-5025 Basic the compiler allows true string comparisons such as IF A$ > "START" THEN 100.

On "INPUT" the compiler beeps when each character is entered.

The Basic program text is stored from B2FFH downwards.

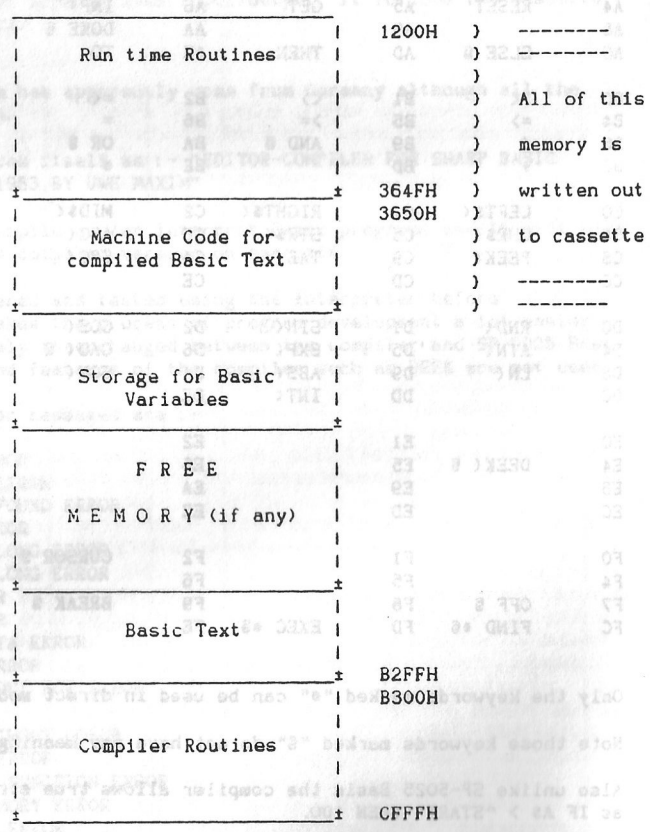
A tokenised line consists of :-

Line number - in binary two bytes
..... - tokenised line
'cr' - end of line

The program text ends with a line number of 0, and a zero in the tokenised text first position.

Memory Map.

The memory map after loading has taken place is as follows :-



The Commands.

The commands described are those which differ from Sharp SP-5025 Basic.

1 COMP - Compile text in memory and out put to cassette tape.
Options are N, R, M or D. The Basic text in memory is compiled and output to cassette. The form is COMP "SAMPLE", N, R

The object program tape loads and executes from 1200H. The generated machine code is located from 3650H. A sample is appended.

2 RUN - Run the program in memory from the interpreter.
A line number cannot be specified. The form is RUN

3 EXEC - Load in a Basic program from cassette and run it.
A program in Basic is loaded from tape and automatically run. The form of the command is EXEC "SAMPLE"

4 $ML - Execute machine code inline.
Machine code in hexadecimal is coded and will be executed at the point encountered. Beware this feature should be used with caution. To jump to the Sharp Monitor code $ML C3,00,00 (JP 0). Note that two digits must be coded and they should be separated by commas. A line number may be coded, this consists of the line number preceded by a "#" eg to jump to line 1234 code $ML C3 #1234.

5 WAIT - Pause execution for a number of seconds.
The execution halts depending on the value specified, for example WAIT 10 pauses execution for 10 seconds.

6 BREAK - Enable/disable SHIFT/BREAK key.
BREAK ON enables the S/B, BREAK OFF disables the S/B key so that execution CANNOT be interrupted until complete.

7 CURSOR - Position the cursor on the screen.
The form is CURSOR x,y where x is between 0 and 39 and y is between 0 and 24. Example CURSOR 1,4 positions the cursor at column 2 on line 5.

8 VTAB -

9 FIND - locate a string in the Basic text.
Finds the designated string and outputs the line in which it is found to the screen. Form is FINDx, note the expression or string x MUST be placed right next to the D of FIND ie no gap.

10 GAU -

11 LIST - List the program in memory.
It should be noted that the program cannot be listed to the printer.

12 AND, OR and XOR - do bit manipulation on numeric fields.
2 AND 3 gives 2.
eg C=(A AND B)

13 DEEK and DOKE - 16 bit versions of PEEK and POKE.

14 ELSE - the otherwise for an IF.
IF A$ = "YES" THEN 100 ELSE PRINT "Error"

The rest of the commands are the same as in SP-5025 Basic as supplied by Sharp.

When running a compiled program, pressing SHIFT/BREAK will cause the message "RESTART ?" to appear, if the answer is "Y" the execution will start at the beginning of the program else there will be a return to Sharp monitor.

Whenever an execution error occurs there will be an error message such as "DIMENSION ERROR" and this will be followed by the restart message.

If you accidentally leave the compiled program then the program may be restarted from monitor by typing in "GOTO$1200".

It should be noted that the pokes which are familiar to SP-5025 and SP-5060 Basic will not work with the compiler, although it will compile the text, when run the pokes could have a disastrous effect.

The good news for S U C members is that the compiler will both interpret and compile programs written in SP-5025 Basic. Compiled Basic programs require no further support and will run from the Sharp Monitor.

Appendix.

Sample Program Compilation and Output.

```
10 FOR I= 1 TO 10
20 PRINT I;
30 NEXT I
40 END
```

This gives output as follows :-

```
3650          CALL 2F74H              ; Initialise
3653          CALL 33E6H
3656          DB   0C1H, 0, 0, 0, 80H ; decimal 1
365B          LD   HL, 3688H          ; I
365E          CALL 2E66H
3661          CALL 33E6H
3664          DB   0C4H, 0, 0, 0, 0A0H ; decimal 10
3669          CALL 2ECAH
366C          CALL 2ED5H
366F          LD   HL, 3688           ; I
3672          CALL 1F8CH              ; Print number
3675          CALL 2CE8H
3678          LD   HL, 3688H          ; I
367B          CALL 2EFCH
367E          CALL 6                  ; new line
3681          CALL 3506H              ; RESTART? subroutine
```

```
3684            CALL 3506H              ;  "    "    "    "
3687            DB   0
3688            DS   5                  ; Storage for I
```

This sample :-

```
10 A=4
20 B=36
30 C=A+B
40 PRINT "A+B=";C
50 END
```

Gives output as follows :-

```
3650            CALL 2F74H              ; Initialize
3653            LD   HL,369FH           ; A
3656            PUSH HL
3657            CALL 33E6H
365A            DB   03H,0,0,0,80H      ; decimal 4
365F            CALL 1F7EH
3662            LD   HL,36A4H           ; B
3665            PUSH HL
3666            CALL 33E6H
3669            DB   06H,0,0,0,90H      ; decimal 34
366E            CALL 1F7EH
3671            LD   HL,36A9H           ; C
3674            PUSH HL
3675            LD   HL,369FH           ; A
3678            CALL 283CH
367B            LD   HL,36A4H           ; B
367E            CALL 281EH
3681            CALL 1F7EH
3684            CALL 2C29H              ; Print string
3687            DB   'A+B=',13          ; String
368C            CALL 2D87H
368F            LD   HL,36A9H           ; C
3692            CALL 1F8CH              ; Print number
3695            CALL 27E8H
3698            CALL 3506H              ; RESTART? subroutine
369B            CALL 3506H              ;  "  "  "  "  "
369E            DB   0,
369F            DS   5                  ; Variable A
36A4            DS   5                  ; Variable B
36A9            DS   5                  ; Variable C
```

It can be seen that this is a true compiler, ie that which produces machine code. The machine code is organised such that the run time routines occupy 1200H to 364FH and the compiled Basic text starts at 3650H, variables are stored above the program.

I haven't had time to do any timing comparisons between the compiled and SP-5025 interpreted Basic. It should be interesting to compare the speeds.

MZ-80K Section
Edited by Maurice Hawes
6 Belle Vue, The Esplanade, Weymouth
Telephone 0305 783518

MZ80K News

Apart from the very welcome Library lists, there is not a lot of specific MZ-80K Material this time - but most of the General, CP/M and Hardware sections are applicable to the "K", Peter Tuffs has added more on the newly-discovered compiler, and Andrew Ferguson has given us a very useful chart of PEEKs, POKEs and USRs.

BASIC COMPILER FOR THE MZ-80K - FOLLOW-UP by Peter Tuffs

Readers will recall that gaps were left in my article in Vol.8 No.3 pp.24-29, because I was unable to determine the functions of certain commands. John Edwards has sent me further information which was passed to him by Alan Bunting, and this information is given below, where it amplifies or completes the earlier article.

SP-5025 programs can be compiled, run, and edited repeatedly without the use of a conversion program. The complete list of SP-5025 programmable commands can be used, as well as:-

1. All direct commands except for LIST,CONT,SAVE,VERIFY and AUTO.

2. New commands DOKE,DEEK,IF.THEN.ELSE,AND,OR,XOR,WAIT,CURSOR,BREAK

3. Also allowed are hexadecimal constants such as $D000 (except in DATA statements), and hexadecimal Z80 machine-code preceded by $ML.

The Editor is essentially the same as that of SP-5025 and is used in the same manner. Multiple direct statements such as LOAD:LIST may not be used. NEW has several variations:-

NEW	Clears the current programme
NEW x	Deletes line x
NEW x,y	Deletes from line x to line y
NEW x-	Deletes from line x onwards
NEW -y	Deletes up to line y

The Compiler commands are as follows:-

RUN	Runs the compiled program; if it has not been compiled, COMP will be run first, automatically.
EXEC	Equivalent to LOAD:COMP:RUN
COMP	Compiles a program; switches are:- ,R compiles and runs ,N uses additional memory space ,D uses space taken up by the source program ,M uses space taken up by compiler and source "NAME" compiles and saves object code to tape

COMP switches may be combined e.g. COMP,R,N"NAME".

GAU(x) function is apparently equivalent to INT(x).

My thanks to John Edwards for sending me the information, and to Alan Bunting for passing it to John in the first place. ***

GERMAN TAPE-BASED COMPILER FOR THE MZ-80K AND MZ-80A

Update by Greg Chapman and Maurice Hawes

EDITORIAL NOTE

In his main MZ-80A article on Compilers, elsewhere in this issue, Greg Chapman refers to the original German Manual for Uwe Maxim's SA-5510 Compiler, and its English translation. Our first idea was to print the English SA-5510 Manual in the 'A' section, but when we compared it with its SP-5025 counterpart, we found that the two Manuals were virtually identical, and also realised that most of the 'K' Manual had been published in Vol.9 No.1 (though at that time your Editor did not recognise it as such!). The existence of Uwe Maxim's 'A' Compiler was noted by Steve & Phil Browne in their review of the 'K' Compiler in Vol.4 No.1, but later S.U.C articles refer only to the 'K' Compiler. In Vol.4 No.3, Andy Binfield gave comparative timings and showed how to make a security copy; and in Vol.8 No.3, P. Tuffs revealed a great deal of information which was not given in any of the original German and English Manuals.

Vol.8 No.3 is still available, but all the other sources are out of print, so we decided to summarise the literature, covering both versions of the Compiler, and present it in the 'General' section of this Magazine. Our overview, below, is based on the English Manuals, but has additions which reflect practical experience.

INTRODUCTION

Standard tape Basic programs can be compiled, run, and edited repeatedly with this Compiler, without using a conversion program or disk Compiler (a dig at FDOS ? - Ed.) Most Basic commands are allowed; the 'K' Compiler keywords with their 1-byte tokens are listed in Vol.8 No.3; the 'A' Compiler keywords are similar, but their tokens are mixed 1-byte and 2-byte, as in SA-5510. There are also several 'extras' - auto-repeating keys for the MZ-80K, proper string comparisons, and several new commands (see next paragraph).

PROGRAMMABLE COMMANDS

All SP-5025 and SA-5510 commands except for LIST, CONT, SAVE, VERIFY and AUTO, plus the following NEW commands:

DOKE address (16-bit number)	- 16-bit POKE
DEEK (16-bit number)	- 16-bit PEEK
IF..THEN..ELSE	- upgraded from IF...THEN
AND OR NOT	- Boolean
WAIT n	- waits n seconds (n<256)
BREAK OFF/ON	- disables/enables BREAK key
GAU(x)	- (nearly ?) the same as INT(x)
VTAB(n) (MZ-80K only)	- appears to do nothing (?)

In addition, good facilities are provided for links to machine-code; hexadecimal constants are allowed throughout (except in DATA statements), and Z80 machine-code 'in-line' instructions may be introduced by $ML. It is also possible to refer directly to line numbers within the source code. Examples of hexadecimal constants and 'in-line' machine-code are given on the next page.

EXAMPLES OF HEX CONSTANTS AND LINKAGE TO MACHINE-CODE

```
VR=$D000          - set variable VR to $D000

$ML CD,3E,00      - call the Monitor routine at $003E

$ML C3,00,00      - jump to $0000

$ML C3 #1500      - jump to the address of LINE 1500

$ML 21 #2000      - load HL with the address of LINE 2000
```

In general, #nn represents the 16-bit address of line number nn, and is NOT preceded by a 'comma'. In the last example, LINE 2000 could contain machine-code data, introduced by $ML.

Note that when using $ML to pass a multi-byte instruction, it must all be on the same line, e.g. you CANNOT write 10 $ML,CD followed by 20 $ML 3E,00; it must all be on one line e.g. 10 $ML CD,3E,00.

EDITOR COMMANDS

The editor of the Basic Compiler is essentially the same as that of SP-5025 and SA-5510, and is used in the same manner (i.e. it allows full-screen editing). However, the following ADDITIONAL editor commands can be used in direct mode (but note that multiple commands such as LOAD:LIST may not be used):-

LOAD Loads a SHARP Basic program

SAVE Saves a SHARP Basic program held in memory (the program is readable by the normal Basic interpreter).

VERIFY Compares the saved program with that held in memory

```
LIST lists the whole program
LIST x lists line x only
LIST x-y lists lines x-y inclusive
LIST x- lists from line x to the end.
LIST -y lists from the start up to line y.

NEW deletes ALL the current program
NEW x deletes line x only
NEW x-y deletes from line x to line y.
NEW x- deletes from line x to end
NEW -y deletes up to line y
```

FINDtext looks for specified text, and displays the line number(s) where it is found. The specified text may be a keyword e.g. FINDPOKE finds all occurrences of the POKE command. (N.B. no gap is allowed between 'FIND' and 'text' !!)

BYE or MON (as appropiate) jumps to the ROM Monitor.

The commands LOAD, SAVE, and VERIFY allow (optional) 16-character filenames, exactly as in BASIC.

COMPILER COMMANDS

RUN Runs the compiled object program; if it has not been compiled, or if lines have been changed since the last compilation, COMP is invoked automatically before RUN.

EXEC Equivalent to the command sequence LOAD:COMP:RUN.

COMP Compiles a program without destroying the source; suitable for BASIC programs up to about 12K, gives MEMORY ERROR if the program is too big. Error lines are indicated (if LINE NOT FOUND ERRORs come up, an object program is still produced, because the compiler run is not broken).

COMP,R As for COMP, but also runs the program.

COMP,N Produces an object program willy/nilly; if there are errors, line numbers are indicated. Uses additional memory.

COMP,D Used if COMP, RUN or EXEC give MEMORY ERROR. Programs up to about 20k long can be compiled, but the source code is destroyed. If errors are found in lines containing DATA or REM statements, the following text is lost. In any case, compiling time is longer with DATA and REM statements.

COMP,M (includes COMP,D) is used if COMP,D gives MEMORY ERROR. It creates the maximum memory capacity for variables and strings, by destroying the source program AND the Compiler.

COMP"name" is used to compile a program and then save it to tape. If the program is properly debugged, it will subsequently run without the Compiler, loading at $1200 (from the Monitor) and thus having maximum memory available for data.

It is also possible to use different compiler modes in conjunction with one another, using comma separators e.g. COMP,N,R,"name".

WARM STARTS, POKES, AND PRINTER SOFTWARE

Compiler warm start - GOTO$B300 or JB300
Compiled program warm start - GOTO$1200 or J1200
Most if not all of the common BASIC POKEs will crash the system! The printer routine is at $2737 on the 'K', or $2746 on the 'A'; you may need to modify this routine for a non-Sharp printer.

POINTS TO WATCH

When using COMP,D or COMP,M it is advisable to SAVE the program, because the source code is destroyed. Rational recursive programs are permitted. When writing nested FOR..NEXT loops, especially when sub-routines are involved, the appropriate loop variable should be specified after every NEXT (if this is not done, errors may occur on RUN). A line such as 10 GOSUB10 should not be used (the program is destroyed). 'DATA' with nothing after it implies an empty string; 'DATA,' means two empty strings, one before and one after the ','. In borderline cases GAU(x) and INT(x) may return different values. VTAB(x) is a mystery; it is accepted if x is in the range 0-24 (vertical TAB ?) but all it does is 'CR'. ***

1. GERMAN SA-5510 COMPILER (by Uwe Maxim)

This is undeniably the easiest of the compilers to use. At first sight, the package appears and behaves like standard SA-5510 with which everyone will be familiar. Then, some differences start to emerge. For instance, only the direct commands given in the Compiler manual work. These do not include AUTO, to generate program line numbers automatically. Eventually you realise that the sign-on screen which announced the program as an 'Editor-Compiler' was right. That is just what it is.

Originally the package came with two manuals, one in the original German, the second, an English translation. A slightly edited version of the original English manual is reprinted elsewhere in this issue, from which you can see that the Germanic flavour of the original is retained! Those who find the manual a little too brief should read Peter Tuff's article in Vol.8 No.3 pp.24-29, and Andy Binfield's timings of BASIC and compiled sample programs in Vol.4 No.3 p.7. Although both these articles describe Uwe Maxim's SP-5025 Compiler, virtually everthing in them holds true for the SA-5510 Compiler, including upper-case error messages; this shows that the SA-5510 Compiler was derived from the earlier SP-5025 Compiler. However, there are differences between the two Compilers; for instance, the SA-5510 Compiler does not come protected in the way that Andy Binfield describes. The original manual also confirms the Compiler's MZ-80K origins; it refers to CURSORx,y as being an 'extended' command, and tells you to use the MZ-80K Monitor command 'GOTO$XXXX', instead of the correct MZ-80A command 'JXXXX'.

It is certainly worth experimenting to see how your favourite programs work with this Compiler. The manual does point out that it is a little more fussy in some aspects of syntax than SA-5510 itself; some of my programs needed slight adjustment to their FOR..NEXT loops to get them to run under the Compiler. A negative aspect, which surprised me, is that a compiled PRINT statement is no quicker than under the interpreter. Also, the POKE given in Vol.4 No.1 on pp.42-43 does not work with the SA-5510 Compiler (but I strongly suspect that the location has been moved, rather than omitted altogether, in the MZ-80A version of the program).

For some users, the additional facilities of the Compiler, such as IF..THEN..ELSE, DOKE and DEEK, WAITn(seconds), the Editor command FINDxxx, and the extension of NEW to allow block deletion of lines, may make the Compiler more attractive than the Interpreter. The command, EXEC, which will load, compile and run a BASIC source file, is also welcome. Source programs longer than about 20k must be saved to tape when compiled, and then run from the ROM Monitor as machine-code programs, because they overwrite both the Editor and the Compiler. Be warned, however, my orignal CASH FLOW program took over six minutes to load when compiled in this way. The run-times are substantial, as they must be with any compiler that will compile BASIC's comprehensive string handling and floating point arithmetic statements. As a result I cannot recommend using this Compiler to produce machine-code versions of short programs. It is better to load the Compiler and use the EXEC command to load and run the original source code.