



**THE ZX  
ASSEMBLER & DISASSEMBLER**

Copyright 1983 by

Bob Berch

Rochester NY 14620

## Contents

- I. Introduction
- II. The Key Commands
- III. The Assembler
- IV. Sample Programs

## Appendices

- A. Error reports
- B. Useful Assembler/Disassembler routines
- C. Useful ROM routines
- D. Useful books



## I. Introduction

Welcome to the ZX Assembler/Disassembler by Bob Berch. The primary purpose of this program is to help you write and debug machine code programs. Included in this one program is an assembler utilizing standard Z80 mnemonics, a disassembler giving hex or decimal output to the screen or printer, a hex or decimal memory editor, a REM generator to make REM statements to save machine code in, and a set of binary SAVE and LOAD routines. The program is a little over 4K long and is supplied on cassette in two versions, back to back:

1. Hi-mem, above RAMTOP, program at 28-32K. Those with only a 16K rampack (the minimum configuration) must use this one.
2. Lo-mem, in the ROM transparent area, program at 12-16K.

To use this version you must have memory in this area as with a 64K Rampack or a Hunter board.

In addition, this program is available on EPROM in the lo-mem version, without the LOAD and SAVE routines, as an exactly 4K program. A plug-in cartridge kit is available to hold the EPROM.

If you can use the lo-mem version, it is recommended you do so as you will find it more out-of-the-way and also more immune to crashes from your generated machine code gone haywire.

This manual will not attempt to teach you machine code programming and you are referred to some of the many Z80 books on the market in appendix D. You are given sample programs in part IV and a fairly complete list of usable ROM routines and their calling sequences in appendix C and a similar list of usable Assembler/Disassembler routines in appendix B.

To load the tape, choose the proper version, and use the name 'ASMB'. It will auto-boot itself in place and the hi-mem version will lower RAMTOP. A title page will be displayed.

Assembly language source code for the assembler is entered under Basic in REM statements; the specifications are given below. The Assembler/Disassembler is called with a RAND USR command and the assembly is started by pressing the Z key. If no errors are found, the object code is deposited in a string named A\$. Here it may be viewed with the disassembler, moved to another part of memory or a REM statement, edited, or saved as a binary file.

The RAND USR command for each version is:

low-mem	RAND USR 13616
hi-mem	RAND USR 30000

Once the RAND USR call has been made, the various functions of the Assembler/Disassembler are called with single letter key commands (letters A to Z). An asterisk in the lower left of the screen signifies a wait for one of these commands.

## II. The Key Commands

First, observe the four display modes illustrated below:

### Disassembly Mode

Hex			Decimal	
00000000	00	OUT FD, A	0	211
00000000	01	LD BC, 7FFF	1	253
00000000	02	JP 0307	10	255
00000000	03	LD HL, (4016)	40	256
00000000	04	JR (4010), HL	34	257
00000000	05	LD A, 00	14	258
00000000	06	AND A	16	259
00000000	07	JP NZ, 07F1	17	260
00000000	08	JP 07F5	19	261
00000000	09	RST 38	20	262
00000000	0A	LD HL, (4016)	33	263
00000000	0B	LD A, (HL)	40	264
00000000	0C	AND A	42	265
00000000	0D	RET NZ	1E	266
00000000	0E	NOP	16	267
00000000	0F	NOP	0	268
00000000	10	NOP	0	269
00000000	11	CALL 00349	0	270
00000000	12	JR 00310	20	271
00000000	13	RST 38	24	272
00000000	14	RST 38	24	273
00000000	15	RST 38	24	274
00000000	16	RST 38	24	275
00000000	17	RST 38	24	276
00000000	18	RST 38	24	277
00000000	19	RST 38	24	278
00000000	1A	RST 38	24	279
00000000	1B	RST 38	24	280
00000000	1C	RST 38	24	281
00000000	1D	RST 38	24	282
00000000	1E	RST 38	24	283
00000000	1F	RST 38	24	284
00000000	20	RST 38	24	285
00000000	21	RST 38	24	286
00000000	22	RST 38	24	287
00000000	23	RST 38	24	288
00000000	24	RST 38	24	289
00000000	25	RST 38	24	290
00000000	26	RST 38	24	291
00000000	27	RST 38	24	292
00000000	28	RST 38	24	293
00000000	29	RST 38	24	294
00000000	2A	RST 38	24	295
00000000	2B	RST 38	24	296
00000000	2C	RST 38	24	297
00000000	2D	RST 38	24	298
00000000	2E	RST 38	24	299
00000000	2F	RST 38	24	300

### Data Mode

Hex			Decimal	
00000000	00	PEEK 03	0	211
00000000	01	CLEAR FD	1	253
00000000	02	COPY 01	10	255
00000000	03	03	19	256
00000000	04	04	20	257
00000000	05	05	21	258
00000000	06	06	22	259
00000000	07	07	23	260
00000000	08	08	24	261
00000000	09	09	25	262
00000000	0A	0A	26	263
00000000	0B	0B	27	264
00000000	0C	0C	28	265
00000000	0D	0D	29	266
00000000	0E	0E	30	267
00000000	0F	0F	31	268
00000000	10	10	32	269
00000000	11	11	33	270
00000000	12	12	34	271
00000000	13	13	35	272
00000000	14	14	36	273
00000000	15	15	37	274
00000000	16	16	38	275
00000000	17	17	39	276
00000000	18	18	40	277
00000000	19	19	41	278
00000000	1A	1A	42	279
00000000	1B	1B	43	280
00000000	1C	1C	44	281
00000000	1D	1D	45	282
00000000	1E	1E	46	283
00000000	1F	1F	47	284
00000000	20	20	48	285
00000000	21	21	49	286
00000000	22	22	50	287
00000000	23	23	51	288
00000000	24	24	52	289
00000000	25	25	53	290
00000000	26	26	54	291
00000000	27	27	55	292
00000000	28	28	56	293
00000000	29	29	57	294
00000000	2A	2A	58	295
00000000	2B	2B	59	296
00000000	2C	2C	60	297
00000000	2D	2D	61	298
00000000	2E	2E	62	299
00000000	2F	2F	63	300

In disassembly mode, the leftmost column of numbers is the address, the raw data at those addresses is at center, and the assembly mnemonics are at the right.

In data mode there are five columns: the leftmost and rightmost columns are the addresses (in hex and decimal), next in is the data at each address (also in hex or decimal), and finally in the center is the Sinclair character equivalent of that data.

Now, the key commands in detail:

- A enter address. A > will appear ( or a >> if you are in hex mode) and you enter the digits for the address for the top of the page (TOP). Hex addresses are always four digits and need no return. Decimal addresses always need a return. The delete key deletes the entire entry and you start again.
- B looks backward in memory. If you're in data mode, you go backward 22 addresses. If you're in disassembly mode, you go back 30 addresses. Disassembly backpages will generally overlap, but occasionally will not, as with a field of all NOP's. In addition, disassembly backpages will often land in the middle of a multi-byte instruction and the first instruction or two will be gibberish. Just be aware of it.
- C continue looking forward in memory. In both data and disassembly mode the top of the new screen starts with the byte after the last byte on the present screen.
- D data/disassembly switch. Changes from disassembly to data mode and back again.
- E edit memory. If in disassembly mode, you will be changed to data mode. A cursor will appear at the top address and an input prompt (hex or decimal, depending on what mode you were in last) at the bottom left. Hex data is two digits and needs no return; decimal data always needs a return. Entered data will be poked into memory at the cursor address and the cursor will be moved down the page. The up and down arrow keys can be used to move the cursor over any addresses.
- F set the EOF address, that is the end of file address. This address is used by the L, M, S, T, and X commands.
- G generate a REM. Enter the number of spaces desired within the REM. Makes a 1 REM filled with x's.
- H hex/decimal switch. Changes from hex to decimal mode and back again.

- L load binary. Use thA command to set the first address to be loaded and the F command to set the last. A load will proceed until the last byte is loaded or the break key is pressed.
- M move memory. Moves a block of memory between and including the TOP address (⏏ cursor) and the EOF address to the address entered on the input prompt. The EOF address must be greater than the TOP address. If the block of memory overlaps its future position, this routine intelligently avoids overwriting it by choosing a top-down or bottom-up move.
- Q (shifted) quit. Used to exit any input prompt to get the asterisk or to leave the asterisk to return to Basic.
- S save binary. This is the complement of L, of course. Use the A and F commands to set the limits to save; you save up to and including the EOF byte.
- Besides being used to save files of your assembled code, these commands can be used to load and list 'unlistable' programs or load, edit, and save damaged tapes. No name or header is saved with these files, just raw data.
- During a save, pressing break will return to Basic.
- T type output. Outputs to a Timex or Sinclair printer, in whatever mode you are in, the present and consecutive screens up to the EOF address. Like save, you may break, but will be returned to Basic.
- X clear. Sets to zeros all memory from the TOP address to and including the EOF address.
- Z assemble. Read on.

Note: some commands ask OK? before proceeding. A 'Y' or an enter will proceed. A 'N'; a 'break' or a shifted Q will allow you an escape.

### III. The Assembler

Before calling the assembler (with the Z command), the source code must be entered and present in memory. This is done under Basic, entering each statement in a REM statement. Here are the rules:

1. The operation code must be immediately after the REM followed by one or more blanks or an end of line. The operand follows, parts of the operand separated by a comma, then an optional semi-colon and comments. Blanks are ignored in the operand but must not appear in the op code. A line full of comments starts with a semicolon.

Your REM statements are edited or deleted like any Basic statement.

2. In the operand, variables in the Basic variable area may be used in place of any number. Variables so defined must be at least three characters long and not start with the letter H. Define the variables with LET statements, either in immediate mode or as program statements after a END statement (see below) The second way better documents your program. Remember to 'RUN' your program before calling the assembler to execute your LETs.

3. Hexadecimal constants are preceded by an 'H' and are two, three, or four digits long. When only two digits are given for a 16 bit constant, they will be the low order byte.

Decimal data is accepted in the range zero to 65535.

Character data within quotes is allowed, one or two characters long. More than two characters may be given, but only the last two are recognized. Any character code is allowed except the quote (0Bh) and the enter (76h).

examples:

valid constants

invalid constants

H543	(0543h)	+1
'='	(14h or 0014h)	7000
8193	(2001h)	H7
'WXYZ'	(3E3Fh)	-1

4. Branch instructions use the line number or a variable equal to a line number to reference an address, or you may use an absolute address. To reference a line number, put a slash in front of the number. LD instructions may also use a slash in front of a number to reference a line in the program.

examples:

CALL /1200      calls the subroutine at line 1200

JR /80          jumps to line 80

CALL HOA2A        calls the ROM routine at 0A2Ah

LD HL,/TB1        where TB1 is defined by a LET statement  
                  at the end of the program as some line  
                  in the program, HL is loaded with the  
                  address of that line.

6. Instructions are exactly as printed in the ZX81/TS1000 manual appendix with two small exceptions:

1. EX AF,AP' should leave out the prime sign (that is, EX AF,AP ).
2. IM 0, IM 1, and IM 2 are written without spaces (that is, IMO, IM1, and IM2)

Note, this is how the disassembler prints them, too.

6a. As an optional shorthand device, the 'f' symbol may be used in place of '(HL)'.

7. There are four pseudo-ops:

1. ORG specifies the run-time address for the first instruction in your program. If used more than once, only the last ORG is recognized. If no ORG is given, 16514 is assumed.

2. DATA puts data in memory in sequence with the rest of your code. There is no limit to the number of operand parts, each separated by a comma. There are three categories of operand parts allowed:

- a. simple constants or variables. If the value is between 0 and 255, it is stored as one byte. If the value is between 256 and 65535 it is stored as two bytes, low byte first. Note, to store H0001 as two bytes you must use H01,H00 .

- b. string data. In DATA statements only, any number of characters within quotes is stored in entirety. Same rule concerning no quotes or enters.

- c. line number address. You may use a slash in front of a constant or variable to store a line number address, but then make that the only operand part in that DATA statement.

3. RESV puts a specified number of zeros in memory.

ex. RESV 5 is the same as DATA 0,0,0,0,0

4. END is required as the last statement of your source code.

Having entered your source code, call the assembler with a HAND USR xxxx (depending on version) and then Z. If the assembler finds errors, it will print 'ASSEMBLY ERROR' and a report with an error number and line number. The report codes are in appendix A. You will be left in Basic to edit your mistake; do so, and recall the assembler.

If no errors are found, the output code will be deposited in a string named A\$. You will be put in disassembly mode with the top of the page as the first byte of your code and the EOF set to the last. With this, you are set to use the move or save commands. If you move the display, make note of the TOP address so you can find it later.

### Tips

It is considered good programming to be liberal with comments. Especially with this assembler, where line numbers are used over labels for referencing, you have lost a certain readability good comments will restore.

Of course, save your source code before you run your program, for even simple mistakes may crash the system requiring you to reset the machine. If you save your source right after an assembly (with the Basic SAVE), A\$ will be saved along with it.

If you are going to run your code in a 1 REM statement (location 16514 and on), you may want to generate the REM before you assemble. Make it at least as big as you expect the code to be (you can estimate high with no ill effects), and change the first character to a semicolon so the assembler will ignore it. Assemble, move the code to 16514 (4082 hex) (TOP and EOF are already set, but check the length), and try your program.

To save your machine code in a 1 REM without the source code, move the code block to a protected part of memory (8 to 16K or above a lowered RAMTOP) or save your code to tape. Execute a NEW, then generate a REM and reload your code into it.

#### IV. Sample Programs

```

1 REM *****XXXXXXXXXXXXXXXXXXXXX
2 REM **HORIZONTAL LEFT**
3 REM ***** SCROLL *****
4 REM
5 REM LD B,22;LINE COUNTER
6 REM LD HL,(DFIL);1ST LINE
7 REM
8 REM PUSH BC
9 REM LD BC,31
10 REM INC HL;1ST POSITION
11 REM LD A,E;SAVE FOR WRAP
12 REM PUSH HL
13 REM POP DE
14 REM INC HL
15 REM LDIR ;SCROLL 1 LINE
16 REM DEC HL
17 REM LD E,A;FILL LAST POS.
18 REM INC HL
19 REM POP BC
20 REM DJNZ /20 ;LOOP
21 REM
22 REM RET
23 REM END
24 REM
25 REM RET DFIL=16396
26 REM
27 REM
28 REM TEST PROGRAM
29 REM
30 REM FOR I=1 TO 300
31 REM PRINT AT AND*21,RND*31;"*";
32 REM NEXT I
33 REM
34 REM
35 REM LET T=USR 16514
36 REM
37 REM
38 REM
39 REM
40 REM
41 REM
42 REM
43 REM
44 REM
45 REM
46 REM
47 REM
48 REM
49 REM
50 REM
51 REM
52 REM
53 REM
54 REM
55 REM
56 REM
57 REM
58 REM
59 REM
60 REM
61 REM
62 REM
63 REM
64 REM
65 REM
66 REM
67 REM
68 REM
69 REM
70 REM
71 REM
72 REM
73 REM
74 REM
75 REM
76 REM
77 REM
78 REM
79 REM
80 REM
81 REM
82 REM
83 REM
84 REM
85 REM
86 REM
87 REM
88 REM
89 REM
90 REM
91 REM
92 REM
93 REM
94 REM
95 REM
96 REM
97 REM
98 REM
99 REM
100 REM
101 REM
102 REM
103 REM
104 REM
105 REM
106 REM
107 REM
108 REM
109 REM
110 REM
111 REM
112 REM
113 REM
114 REM
115 REM
116 REM
117 REM
118 REM
119 REM
120 REM
121 REM
122 REM
123 REM
124 REM
125 REM
126 REM
127 REM
128 REM
129 REM
130 REM
131 REM
132 REM
133 REM
134 REM
135 REM
136 REM
137 REM
138 REM
139 REM
140 REM
141 REM
142 REM
143 REM
144 REM
145 REM
146 REM
147 REM
148 REM
149 REM
150 REM
151 REM
152 REM
153 REM
154 REM
155 REM
156 REM
157 REM
158 REM
159 REM
160 REM
161 REM
162 REM
163 REM
164 REM
165 REM
166 REM
167 REM
168 REM
169 REM
170 REM
171 REM
172 REM
173 REM
174 REM
175 REM
176 REM
177 REM
178 REM
179 REM
180 REM
181 REM
182 REM
183 REM
184 REM
185 REM
186 REM
187 REM
188 REM
189 REM
190 REM
191 REM
192 REM
193 REM
194 REM
195 REM
196 REM
197 REM
198 REM
199 REM
200 REM
201 REM
202 REM
203 REM
204 REM
205 REM
206 REM
207 REM
208 REM
209 REM
210 REM
211 REM
212 REM
213 REM
214 REM
215 REM
216 REM
217 REM
218 REM
219 REM
220 REM
221 REM
222 REM
223 REM
224 REM
225 REM
226 REM
227 REM
228 REM
229 REM
230 REM
231 REM
232 REM
233 REM
234 REM
235 REM
236 REM
237 REM
238 REM
239 REM
240 REM
241 REM
242 REM
243 REM
244 REM
245 REM
246 REM
247 REM
248 REM
249 REM
250 REM
251 REM
252 REM
253 REM
254 REM
255 REM
256 REM
257 REM
258 REM
259 REM
260 REM
261 REM
262 REM
263 REM
264 REM
265 REM
266 REM
267 REM
268 REM
269 REM
270 REM
271 REM
272 REM
273 REM
274 REM
275 REM
276 REM
277 REM
278 REM
279 REM
280 REM
281 REM
282 REM
283 REM
284 REM
285 REM
286 REM
287 REM
288 REM
289 REM
290 REM
291 REM
292 REM
293 REM
294 REM
295 REM
296 REM
297 REM
298 REM
299 REM
300 REM
301 REM
302 REM
303 REM
304 REM
305 REM
306 REM
307 REM
308 REM
309 REM
310 REM
311 REM
312 REM
313 REM
314 REM
315 REM
316 REM
317 REM
318 REM
319 REM
320 REM
321 REM
322 REM
323 REM
324 REM
325 REM
326 REM
327 REM
328 REM
329 REM
330 REM
331 REM
332 REM
333 REM
334 REM
335 REM
336 REM
337 REM
338 REM
339 REM
340 REM
341 REM
342 REM
343 REM
344 REM
345 REM
346 REM
347 REM
348 REM
349 REM
350 REM
351 REM
352 REM
353 REM
354 REM
355 REM
356 REM
357 REM
358 REM
359 REM
360 REM
361 REM
362 REM
363 REM
364 REM
365 REM
366 REM
367 REM
368 REM
369 REM
370 REM
371 REM
372 REM
373 REM
374 REM
375 REM
376 REM
377 REM
378 REM
379 REM
380 REM
381 REM
382 REM
383 REM
384 REM
385 REM
386 REM
387 REM
388 REM
389 REM
390 REM
391 REM
392 REM
393 REM
394 REM
395 REM
396 REM
397 REM
398 REM
399 REM
400 REM
401 REM
402 REM
403 REM
404 REM
405 REM
406 REM
407 REM
408 REM
409 REM
410 REM
411 REM
412 REM
413 REM
414 REM
415 REM
416 REM
417 REM
418 REM
419 REM
420 REM
421 REM
422 REM
423 REM
424 REM
425 REM
426 REM
427 REM
428 REM
429 REM
430 REM
431 REM
432 REM
433 REM
434 REM
435 REM
436 REM
437 REM
438 REM
439 REM
440 REM
441 REM
442 REM
443 REM
444 REM
445 REM
446 REM
447 REM
448 REM
449 REM
450 REM
451 REM
452 REM
453 REM
454 REM
455 REM
456 REM
457 REM
458 REM
459 REM
460 REM
461 REM
462 REM
463 REM
464 REM
465 REM
466 REM
467 REM
468 REM
469 REM
470 REM
471 REM
472 REM
473 REM
474 REM
475 REM
476 REM
477 REM
478 REM
479 REM
480 REM
481 REM
482 REM
483 REM
484 REM
485 REM
486 REM
487 REM
488 REM
489 REM
490 REM
491 REM
492 REM
493 REM
494 REM
495 REM
496 REM
497 REM
498 REM
499 REM
500 REM
501 REM
502 REM
503 REM
504 REM
505 REM
506 REM
507 REM
508 REM
509 REM
510 REM
511 REM
512 REM
513 REM
514 REM
515 REM
516 REM
517 REM
518 REM
519 REM
520 REM
521 REM
522 REM
523 REM
524 REM
525 REM
526 REM
527 REM
528 REM
529 REM
530 REM
531 REM
532 REM
533 REM
534 REM
535 REM
536 REM
537 REM
538 REM
539 REM
540 REM
541 REM
542 REM
543 REM
544 REM
545 REM
546 REM
547 REM
548 REM
549 REM
550 REM
551 REM
552 REM
553 REM
554 REM
555 REM
556 REM
557 REM
558 REM
559 REM
560 REM
561 REM
562 REM
563 REM
564 REM
565 REM
566 REM
567 REM
568 REM
569 REM
570 REM
571 REM
572 REM
573 REM
574 REM
575 REM
576 REM
577 REM
578 REM
579 REM
580 REM
581 REM
582 REM
583 REM
584 REM
585 REM
586 REM
587 REM
588 REM
589 REM
590 REM
591 REM
592 REM
593 REM
594 REM
595 REM
596 REM
597 REM
598 REM
599 REM
600 REM
601 REM
602 REM
603 REM
604 REM
605 REM
606 REM
607 REM
608 REM
609 REM
610 REM
611 REM
612 REM
613 REM
614 REM
615 REM
616 REM
617 REM
618 REM
619 REM
620 REM
621 REM
622 REM
623 REM
624 REM
625 REM
626 REM
627 REM
628 REM
629 REM
630 REM
631 REM
632 REM
633 REM
634 REM
635 REM
636 REM
637 REM
638 REM
639 REM
640 REM
641 REM
642 REM
643 REM
644 REM
645 REM
646 REM
647 REM
648 REM
649 REM
650 REM
651 REM
652 REM
653 REM
654 REM
655 REM
656 REM
657 REM
658 REM
659 REM
660 REM
661 REM
662 REM
663 REM
664 REM
665 REM
666 REM
667 REM
668 REM
669 REM
670 REM
671 REM
672 REM
673 REM
674 REM
675 REM
676 REM
677 REM
678 REM
679 REM
680 REM
681 REM
682 REM
683 REM
684 REM
685 REM
686 REM
687 REM
688 REM
689 REM
690 REM
691 REM
692 REM
693 REM
694 REM
695 REM
696 REM
697 REM
698 REM
699 REM
700 REM
701 REM
702 REM
703 REM
704 REM
705 REM
706 REM
707 REM
708 REM
709 REM
710 REM
711 REM
712 REM
713 REM
714 REM
715 REM
716 REM
717 REM
718 REM
719 REM
720 REM
721 REM
722 REM
723 REM
724 REM
725 REM
726 REM
727 REM
728 REM
729 REM
730 REM
731 REM
732 REM
733 REM
734 REM
735 REM
736 REM
737 REM
738 REM
739 REM
740 REM
741 REM
742 REM
743 REM
744 REM
745 REM
746 REM
747 REM
748 REM
749 REM
750 REM
751 REM
752 REM
753 REM
754 REM
755 REM
756 REM
757 REM
758 REM
759 REM
760 REM
761 REM
762 REM
763 REM
764 REM
765 REM
766 REM
767 REM
768 REM
769 REM
770 REM
771 REM
772 REM
773 REM
774 REM
775 REM
776 REM
777 REM
778 REM
779 REM
780 REM
781 REM
782 REM
783 REM
784 REM
785 REM
786 REM
787 REM
788 REM
789 REM
790 REM
791 REM
792 REM
793 REM
794 REM
795 REM
796 REM
797 REM
798 REM
799 REM
800 REM
801 REM
802 REM
803 REM
804 REM
805 REM
806 REM
807 REM
808 REM
809 REM
810 REM
811 REM
812 REM
813 REM
814 REM
815 REM
816 REM
817 REM
818 REM
819 REM
820 REM
821 REM
822 REM
823 REM
824 REM
825 REM
826 REM
827 REM
828 REM
829 REM
830 REM
831 REM
832 REM
833 REM
834 REM
835 REM
836 REM
837 REM
838 REM
839 REM
840 REM
841 REM
842 REM
843 REM
844 REM
845 REM
846 REM
847 REM
848 REM
849 REM
850 REM
851 REM
852 REM
853 REM
854 REM
855 REM
856 REM
857 REM
858 REM
859 REM
860 REM
861 REM
862 REM
863 REM
864 REM
865 REM
866 REM
867 REM
868 REM
869 REM
870 REM
871 REM
872 REM
873 REM
874 REM
875 REM
876 REM
877 REM
878 REM
879 REM
880 REM
881 REM
882 REM
883 REM
884 REM
885 REM
886 REM
887 REM
888 REM
889 REM
890 REM
891 REM
892 REM
893 REM
894 REM
895 REM
896 REM
897 REM
898 REM
899 REM
900 REM
901 REM
902 REM
903 REM
904 REM
905 REM
906 REM
907 REM
908 REM
909 REM
910 REM
911 REM
912 REM
913 REM
914 REM
915 REM
916 REM
917 REM
918 REM
919 REM
920 REM
921 REM
922 REM
923 REM
924 REM
925 REM
926 REM
927 REM
928 REM
929 REM
930 REM
931 REM
932 REM
933 REM
934 REM
935 REM
936 REM
937 REM
938 REM
939 REM
940 REM
941 REM
942 REM
943 REM
944 REM
945 REM
946 REM
947 REM
948 REM
949 REM
950 REM
951 REM
952 REM
953 REM
954 REM
955 REM
956 REM
957 REM
958 REM
959 REM
960 REM
961 REM
962 REM
963 REM
964 REM
965 REM
966 REM
967 REM
968 REM
969 REM
970 REM
971 REM
972 REM
973 REM
974 REM
975 REM
976 REM
977 REM
978 REM
979 REM
980 REM
981 REM
982 REM
983 REM
984 REM
985 REM
986 REM
987 REM
988 REM
989 REM
990 REM
991 REM
992 REM
993 REM
994 REM
995 REM
996 REM
997 REM
998 REM
999 REM
1000 REM

```

This 23 byte program moves everything in the display to the left one position and fills in the last column with the contents of the first. It assumes a fully expanded display file.

Note a .1 REM has been generated (and the first character changed to a semi-colon) to move the code after assembly.

The test program puts 300 random asterisks on the screen then continually calls our scroller.

Replace line 26 with:  
26 REM SUB A  
for a non-wrap-around scroll.

After assembling, move the code to the 1 REM and call the test program with a GOTO 200.

```

1 REM ;XXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X
5 REM ;**BIG CHARACTERS**
6 REM ;** PROGRAM **
7 REM ;CHAR AT LOC. 16507
8 REM ;SCREEN POS. ON LINE 34
10 REM ;
20 REM LD A,(16507)
22 REM LD DE,H1E00 ;CODE*8 +
24 REM LD L,A ; 1E00 =
26 REM LD H,E ;CHAR. TABLE
27 REM ADD HL,HL ;POS.
28 REM ADD HL,HL
29 REM ADD HL,HL
30 REM ADD HL,DE
32 REM LD E,8 ;LINE CTR.
34 REM LD BC,H080C ;PRINT POS.
36 REM ;
40 REM PUSH BC
42 REM PUSH DE
44 REM PUSH HL
46 REM CALL H08F5 ;PRINT AT
48 REM POP HL
49 REM ;
50 REM LD C,2 ;P/U BYTE
52 REM LD B,8 ;BIT CTR.
60 REM RL C ;TEST BIT
62 REM LD A,"█"
64 REM JR C,/70
66 REM SUB A
70 REM RST H10 ; PRINT
72 REM DJNZ 60
73 REM ;
80 REM POP DE
82 REM POP BC
84 REM INC HL ; NEXT BYTE
86 REM INC B ; NEXT LINE
88 REM DEC E
90 REM JR NZ,/40
99 REM RET
100 REM END
199 REM
200 REM TEST PROGRAM
210 LET A=CODE INKEY$
220 IF A=0 THEN GOTO 210
230 POKE 16507,A
240 RAND USR 16514
250 GOTO 210

```

Another graphics program, here playing with the character table in the upper 512 bytes of ROM. Code to be displayed is passed through the unused system byte at 16507. Two ROM routines are called: Print AT and print a character. Two loops are set up to process the 8 lines and the 8 bits per line.



## Assembler Error Reports

## Appendix A

The error report is given similar to Basic errors, that is, error number - slash - line number. These error numbers apply:

- 1 NON REM STATEMENT or NO END STATEMENT. Only REMs are allowed before the REM END and the REM END is required.
- 2 INVALID OP CODE. Neither a Zilog mnemonic nor a ZX Assembler pseudo op has been found. Remember, blanks are not allowed before or within the op code.
- 3 BAD OPERAND SYNTAX. This is the largest category and could be many things. Check for correct number of operand parts, spelling, punctuations, or the use of parenthesis.
- 4 VARIABLE NOT FOUND. You have specified a variable which can't be found in the Basic variable area. Perhaps you forgot to 'RUN' through your LETs.
- 5 DATA OUT OF RANGE. You have given a value greater than 255 where 8 bits is required or 65535 where 16 bits is required.
- 6 PROGRAM TOO LARGE. The assembler has encountered insufficient head room to process the assembly. You must either raise RAMTOP or assemble your program in two or more segments.
- 7 JUMP DISPLACEMENT TOO LARGE. You have asked for a JR or DJNZ instruction to jump over 128 bytes away. Change to a JP instruction.
- 8 INVALID LINE NUMBER. You have referenced a line number beyond the end of your program.

The Assembler/Disassembler contains some machine code routines which you might want to use or maybe look at with the disassembler and glean some interesting techniques. They are given here with the address for the hi-mem version. Subtract 4000h to get the low-mem address.

address	my name	description
7FEA	INKY	reads the keyboard returning the character value in the A reg. and the L reg.. No key pressed returns zero, the break key 255, and the shift key 254.
7PDF	INKW	waits for a no key condition, then calls INKY.
7F6C	PRTN	prints the contents of HL. Location 16417 (4021h) is used as a control byte as follows: bit 0 reset - print decimal set - print hexadecimal bit 1 reset - signed decimal set - unsigned decimal bit 7 reset - 4 hex digits set - 2 hex digits
7FD1	PC1	prints the character or token of the code in the A reg.
7FB0	PRTC	prints the characters following the call until a 70h byte is reached.
7FC6	TAB	tabs the print position to the column specified by the L reg., setting to blanks everything on the way.
7F4A	INPUT	inputs a number to HL, prompting as described for the assembler/disassembler. Location 4021h is used as described for PRTN. A shifted Q will set the carry.
771E	SCRDN	scrolls the screen down and sets the print position at 0,0.
770B	SCRUP	scrolls the screen up and sets the print position at 21,0.

The table of 26 addresses for each key command starts at 74FC. You may further explore the assembler/disassembler by tracking down those addresses. Unassigned keys are routed to 754E. You may want to change unassigned keys to jump to your own routines.

Here are described many of the useful ROM routines. There are more, but they are often difficult to utilize. For in-depth explanations, consult Dr. Ian Logan's disassemblies.

address

- 0000 RST 0. Not really useful, but you should know this is the cold start for the system. Clears RAM, sets up a display file, etc.
- 0008 RST 8. The error restart. Prints the Basic error message. The byte following this call is incremented and taken as the error number. Line number is taken from PPC (4007h).
- 0010 RST 16. Print a character. Code is taken from the A reg. and must be 0-3Fh, 80h-BFh, or 76h, else you will crash. AF, BC, DE and HL are all saved and restored. If bit 1 of FLAGS is set, output will be sent to the printer.
- 0028 RST 40. The floating-point calculator. This fascinating routine takes up about one-third of the ROM and performs all the Basic arithmetic functions and operations plus a number of special internal ones. The bytes following this call are treated as instructions to manipulate data (five bytes each entry) on the calculator stack. The end of this instruction string is signaled by a 34h byte. Not just numbers, but strings are manipulated by this calculator, their 5 byte entries containing address and length pointers. The list of instruction codes is given at the end of this section. RST 40 does not save and restore most registers.
- 02BB Scan keyboard. Returns in HL the 16 bit 'key value'.
- 031E Save one byte. Outputs to the cassette port the byte pointed to by HL. Unfortunately, there is no simple similar routine for loading.
- 07BD Decode keyboard. Takes 16 bit 'key value' from BC and returns with HL pointing to the proper character in the tables 007E to 0110.
- 0869 Copy screen. Just like the Basic command.
- 08F5 Print AT. Enter with line no. in B reg. and column no. in C reg. Computes new DF-CC with error checking.

099E 'Make room'. Used by the system to insert lines, variables, expand the display file, etc. Updates the system pointers when through. From (HL), makes BC spaces. CALL 099B makes one space.

09D8 Finds the address of a line number in a Basic program. Enter with line no. in HL; exit with address in HL.

0A1F Clears the bottom screen and does a Print AT 23,0.

0A2A Clears total screen, just like Basic command.

0A60 Reclaim memory. Used by the system to delete lines, variables, etc. At HL, deletes BC spaces, updating the necessary system pointers.

0A98 outputs a decimal number, up to 9999, taken from BC reg.

0B6B prints a string of characters, including tokens, pointed to by DE with length BC.

0EB2 Plot, unplot. Enter with B reg. holding Y coordinate, C reg. holding X. To plot, poke T-ADDR (4030h) with A0h; to unplot, poke it with 0.

0C0E Scroll, just like Basic.

0F23 Fast, just like Basic.

0F2B Slow, just like Basic.

0F46 Tests break key and carry is set if not pressed.

0F4B Debounce keyboard. Call after a keyboard read.

111C Finds variables in the variable area. To call, load CH\_ADD (4016 & 7) with the address of a string of the name of the variable you wish to find. Also, bit 7 of the C reg. must be reset. On return, the carry will be set if no name was found, or the carry will be reset and HL will point to the last letter of the name in the variable area. Increment HL and you may pickup the value of the variable, or if an array or string, its length parameters.

1308 16 bit multiply. HL = HL \* DE.

149A Clears the variable area.

1520 BC to floating point. Converts the number in the BC reg. to a 5 byte f.p. number on the f.p. stack.

158A F.P. to BC. Opposite of BC to f.p. Carry is set if result is greater than 65535.

## Calculator instructions

code		code	
00	jump if stack top is true, displacement byte follows	2C	NOT function
01	exchange top two values	2D	duplicates top entry
02	delete top value	2E	divides, returns int. quotient and remainder
03	subtracts top value from 2nd value.	2F	unconditional jump, displacement follows
04	multiplies top two values	30	stack a number, data follows
05	divides 2nd entry by top	31	DJNZ (using BERG), disp. follows
06	2nd entry to power of 1st	32	less than 0 test
07	ORs top two numbers	33	gr. than 0 test
08	ANDs top two numbers	34	end calculation
09	less than or equal operation	35	used by trig. functions
0A	greater than or equal op.	36	integer truncation
0B	not equal op.	37	execute f.p. op. in A reg.
0C	greater than op.	38	converts E-format to floating-point
0D	less than op.	86,	calls series generator
0E	equal to op.	88,	for trig. and log.
0F	addition op.	& 8C	functions
10	ANDs a string and a number	A0	stacks a 0
11	string <= op.	A1	stacks a 1
12	string => op.	A2	stacks $\frac{1}{2}$
13	string <> op.	A3	stacks $\pi/2$
14	string > op.	A4	stacks 10 decimal
15	string < op.	C0	stores top f.p.
16	string = op.	through	entry in system
17	adds two strings	C5	MEMBOT area
18	negates top number	E0	recalls MEMBOT
19	CODE function	through	entry to the
1A	VAL function	E5	f.p. stack
1B	LEN function		
1C	SIN function		
1D	COS function		
1E	TAN function		
1F	ASN function		
20	ACS function		
21	ATN function		
22	LN function		
23	EXP function		
24	INT function		
25	SQR function		
26	SGN function		
27	ABS function		
28	PEEK function		
29	USR function		
2A	STR $\frac{1}{2}$ function		
2B	CHR $\frac{1}{2}$ function		



It is recommended you purchase two 'Bibles':

1. A manual on the Z80 chip. If possible, go to a bookstore or book department of a computer store and choose one whose style you like. In the field are:

The Z80 Assembly Language Programming Manual and The Z80 Technical Manual by Zilog

Z80 Assembly Language Programming by Lance Leventhal, Osborne/McGraw Hill

How to Program the Z80 by Rodney Zaks, SYBEX

The last one is available at Radio Shack stores.

2. A manual on the 8K ROM. There is only one and it is good:

Sinclair ZX81 ROM Disassembly Parts A & B by Dr. Ian Logan, Melbourne House

In addition, there are a few books out specifically on the ZX81 and machine code:

Mastering Machine Code on the ZX81 by Toni Baker, Reston

Machine Language Programming Made Simple by Michael Roberts, Melbourne House

Understanding Your ZX81 ROM by Dr. Ian Logan, Melbourne House

Mastering Machine Code is quite good. It explains the Z80 chip fully, but spends less time on this than the other books, instead delving right into many meaty sample programs, ways to save machine code, and even using the floating-point calculator. Programming Made Simple and Understanding the ROM are primarily tutorials on the Z80 chip and its instructions. Made Simple quickly describes only one full-blown sample program (checkers). Understanding uses bits of the ROM as examples, but only has one thin chapter on its overall organization.

## V1.0 Notes

In the assembler, JR and DJNZ instructions cannot reference absolute addresses. In fact, the slash is actually optional; they will always reference line nos.

On indexed instructions utilizing a displacement, that displacement may only be a decimal number, not hex. The acceptable range on that number is -128 to +127. However, during disassembly in decimal mode, the displacement is printed as a decimal number 0 to 255, 255 being equivalent to -1, etc.

The instruction in the form LD (XY+d),N (indexed 36h instruction) is mis-disassembled as a three byte instruction instead of four, the third byte being used twice. Examples in the ROM of this bug start at 0644h.