

Echo Mode (3 Line Mode)

The burner expects an echo of every byte. A complimented byte received indicates an error has occurred.

Letters used in a Command A, Z, V, E, U, S, L, C, M, P, F, D, I



Port Setup

9600 Baud
No parity
8 Data bits
1 stop bit

CS - ?
DS - ?
CD - ? } - Ignore in 3 line mode.
* You may ignore these 3 lines but you must wire them correctly

LARGEST Prom - 64K (\$FFFF) currently - 16K (\$4000)

NOTES

- ERRORS Are indicated by sending the compliment of the RCVD byte followed by the addr (Hi byte/Low byte), byte received, & Data byte as read

Commands

- B ① Zap - send \$F0 followed by the Data to be Zapped. (binary Data)
- B ② Verify - send \$F5 followed by the Data to be verified with the chip in the Zap socket. (Binary)
- B ③ Verify Erase - send \$F6
- B ④ Upload - send \$F7 then send \$40 for Master or \$80 for clone socket then receive data (Binary)
- B ⑤ Setup changes - send \$F9 the starting address (hi/Lo), then the ending addr (hi/Lo).
- B ⑥ Timing changes - send \$FA the overzap byte, loopmax byte, then the delay time (hi/Lo).
- B ⑦ Module Id Code - send \$FE the receive Module ID, Beginning addr (hi/Lo), & Ending addr (hi/Lo).
- H ⑧ Zap Motorola hex - Send \$53 and your SI Dump (Capital) S
- H ⑨ Zap Mostek hex - Send \$59 and then your Data Y
- H ⑩ Zap Intel hex - Send \$58 and your data X
- B ⑪ Error Return ON - Send \$61 - then command.
- B ⑫ End Binary file - Send \$50 used after Error Return Has been turned on & after the # of preset bytes (Setup parameters) has been sent.
- H ⑬ To Verify Hex Hold down the verify button before power up. & hold until the good Lamp stays on (green). When the verify button is released the green lamp will go out. Now the unit is ready to verify hex

Zread module ID 400
 ZMENU Main Menu 500
 Zout out w/echo 600
 ZAP Zap/Verify 1000
 Zupload UL to MEM 2000
 ZSAVE To Disk 3000
 ZLoad from Disk 3100
 Zchange VARIOUS 3200
 ZIN IN w/echo 700
 ZIN2 No echo 750
 GMENU CLI 130



Zerror - Print error - 700
 msg.

Zend exits program
 with no error
 checking - 4000

1 screen page = 80 col.
 24 rows

Software

NOTES Smart Zap 3line mode Page 1

- ① A OPENPORT for I/O.
- ① B ON Start up the Program sends a \$FF and wait for the echo. ← put in error checking
- ② Then it gets The Module ID (out 1, 1N5 bytes).
- ③ IT then goes directly to the main menu.

- ZMENU - Before you get to ZMENU an ERROR RETURN ON is sent then your selection is processed.
- Zread - Sends a Mod ID command and receives ID STARTING address (hi/lo) and ending address (hi/lo). The calc's size.
- Zout - sends out 1 byte waits for an echo then checks for an error (on error gosub ZERROR)
- ZAP - Sends out ZAP command & data to Zout. When done send ENDBIN
- Zupload - Sends UPLoad command & socket ID command. Then gets data from Zout, Puts into storage in memory.
- ZSave stores memory to a disk filename.
- ZLoad Stores a disk filename into storage in memory
- Zchange Reads Module ID info or memory working range
- ZIN gets a byte from Zapper with echo.
- ZIN2 gets a byte from Zapper without echo.
- Zerror - gets Failed address, data sent, and prom Data.
- Zwaitkey
- ZDir
- ZASCII
- Zfill
- ZClean

```

1: Rem *****
2: Rem *           Public Domain Version.           *
3: Rem *   Sample program on Atari ST for SMART ZAP using GFA BASIC.   *
4: Rem *   Translated by N.Cherry for use on an Atari 520 ST.         *
5: Rem *                                                                 *
6: Rem *           This program is not perfect or really user friendly. *
7: Rem *           But does work as advertised!                       *
8: Rem *** Z2.BAS ***** 03/04/88 *****
9: Rem
10: Rem *****
11: Rem *   This software was written for the SMART ZAP EPROM BURNER   *
12: Rem *           by MICRO KIT                                       *
13: Rem *           6910 Patterson                                       *
14: Rem *           Caledonia, Mich 49816                               *
15: Rem *           (616) 791-9333                                       *
16: Rem *           Price: $99 Kit, $124 assembled and tested          *
17: Rem * This software works better if you have the above, won't work at all if *
18: Rem *           if you don't!                                       *
19: Rem *****
20: Rem
21: Rem *****
22: Rem * This file is not yet complete and future versions may not be exactly *
23: Rem * identical to the first 2 versions. The next release should have the *
24: Rem * variables cleaned up and should allow the files to be placed anywhere *
25: Rem * within the 64k of reserved ram. And the setup and timing changes *
26: Rem * should work well enough so that changing will not require a lot of *
27: Rem * effort on the users part. I may also rewrite the Burners firmware to *
28: Rem * Allow thing like sending a new Module ID code whenever a module has *
29: Rem * changed and a few other ideas I have.                         *
30: Rem * Add Search, Replace, and Change the way you can exit from the print *
31: Rem * menu.                                                           *
32: Rem *****
33: '
34: ' *
35: ' * Memory set up to replace the dim stor$
36: ' *
37: '
38: Resvmem=70*1024
39: Reserve Fre(0)-Resvmem
40: '
41: Deffn Malloc(A)=Gemdos(72,L:A)
42: Deffn Mfree(A)=Gemdos(73,L:A)
43: Deffn Set_port(A,B,C)=Xbios(15,A,B,C,-1,-1,-1)
44: Deffn Current_drive=Gemdos(&H19)
45: '
46: Version=2.1
47: '
48: Promsize=&H10000 ! * 64k + 1
49: Prom_addr=Fn Malloc(Promsize)
50: Gosub Promfill
51: '
52: Hidem ! * some kind of a bug has arisen where the hide/show mouse
53: Showm ! * routine doesn't work until after you've done a hidem
54: Hidem ! * showm and another hidem. Then the routines seem to work fine.

```

*instead of using
RAM for storage use
a tmp file.*



```
55: '  
56: ' *  
57: ' * Set the serial port to 9600 baud and RTS/CTS.  
58: ' *  
59: '  
60: Baud=1 ! * 9600 baud  
61: Cntl=2 ! * RTS/CTS  
62: Ucr=136 ! * 8 data bits, 1 stop , No Parity  
63: Void=Fn Set_port(Baud,Cntl,Ucr)  
64: '  
65: ' *  
66: ' * change to the current drive  
67: ' *  
68: '  
69: Drive=Fn Current_drive  
70: Chdrive Drive+1 ! * current_drive returns drive starting at 0 instead of 1  
71: Chdir '\' ! * change to the root directory  
72: On Drive+1 Gosub A_drive,B_drive,C_drive,D_drive  
73: '  
74: ' *  
75: ' * Initialize the variables and stuff  
76: ' *  
77: '  
78: Cls  
79: Begint=0  
80: Zendt=0  
81: N$='NONE'  
82: O$='NONE'  
83: C$='NONE'  
84: Xx=0  
85: '  
86: ' *  
87: ' * Wait for keyboard input *****  
88: ' *  
89: '  
90: Print  
91: Print Tab(22);'ST is now set for 9600 baud, RTS/CTS.'  
92: Print Tab(19);'Set the DIP switches to : 1-7 ON and 8 OFF.'  
93: Print Tab(14);'Turn on Smart Zap and hit the space bar to continue.'  
94: Box 0,0,639,80  
95: Box 2,2,637,78  
96: '  
97: Gtkey:  
98: Repeat  
99:   A$=Inkey$  
100:   If A$=Chr$(26) ! 'Z will exit the hole program  
101:     Gosub Finish  
102:   Endif  
103: Until A$=' '  
104: '  
105: ' *  
106: ' * Open the port *****  
107: ' *  
108: '  
109: Open 'r',#1,'AUX:'
```

```

110: Open 'r',#2,'CON:'
111: '
112: ' *
113: ' * Clear the port *****
114: ' *
115: '
116: Print
117: Print
118: Print Tab(33);'CLEARING PORT'
119: Do
120:   A=Inp?(1)
121:   Exit If A=0
122:   Void=Inp(1)
123: Loop
124: Print Tab(34);'PORT CLEARED'
125: Print
126: '
127: ' *
128: ' * Send $FF out and wait for the echo *****
129: ' *
130: '
131: A=255
132: Out 1,A
133: Repeat
134:   A=Inp?(1)
135: Until A=-1
136: A=Inp(1)
137: '
138: ' *
139: ' * Start of main routine Menu *****
140: ' *
141: '
142: Gmsub1:
143: Gosub Zread
144: Gmenu:
145: Gosub Zmenu
146: Gosub Gkl
147: Gosub Toupper
148: A=&H61
149: Gosub Zout
150: '
151: ' *
152: ' * CLI *****
153: ' *
154: '
155: ' * Zap
156: '
157: If A#='Z' Then
158:   Command=&HF0
159:   Temp=1
160:   Goto Zap
161: Endif
162: '
163: ' * Verify
164: '

```

clear port

check burner

ZAP

```
165: If A$='V' Then
166:   Command=&HF5
167:   Temp=0
168:   Goto Zap
169: Endif
170: '
171: ' * Erase Check
172: '
173: If A$='E' Then
174:   Print At(11,23);'Checking PROM';
175:   Bad=0
176:   A=&HF6
177:   Gosub Zout
178:   A=&H50
179:   Gosub Zout
180:   If Bad=0
181:     Print At(11,23);'Erase Check - O.K.';
182:     Goto Zwaitkey
183:   Endif
184: Endif
185: '
186: ' * Upload
187: '
188: If A$='U' Then
189:   Goto Upload
190: Endif
191: '
192: ' * Save to disk
193: '
194: If A$='S' Then
195:   Goto Zsave
196: Endif
197: '
198: ' * Load from disk
199: '
200: If A$='L' Then
201:   Goto Zload
202: Endif
203: '
204: ' * Change
205: '
206: If A$='C' Then
207:   Goto Change
208: Endif
209: '
210: ' * Exit program
211: '
212: If A$=Chr$(&H18) ! Total unconditional exit 'X
213:   Showm
214:   Gosub Finish
215: Endif
216: '
217: If A$='X' Then ! * Alert before exit
218:   Goto Zend
219: Endif
```

checkErase

Upload

ZSAVE

ZLOAD

Change

ZEXIT

```
220: '  
221: ' * Print to screen  
222: '  
223: If A$='P' Then  
224:   Goto Zedit  
225: Endif  
226: '  
227: ' * Fill STOR$ with $FF  
228: '  
229: If A$='F'  
230:   Goto Zfill  
231: Endif  
232: '  
233: ' * Directory listing  
234: '  
235: If A$='D'  
236:   Goto Zdir  
237: Endif  
238: '  
239: ' * Information  
240: '  
241: If A$='I'  
242:   Goto Zinfo  
243: Endif  
244: '  
245: ' * Clean EEPROM  
246: '  
247: If A$='A' And Ee=1  
248:   Gosub Zfill2  
249:   Command=&HF0  
250:   Goto Zap  
251: Endif  
252: '  
253: ' * Not a command, ring bell  
254: '  
255: Print Chr$(7)  
256: Goto Gmenu  
257: '  
258: ' *  
259: ' * Command code for ID module read *****  
260: ' *  
261: '  
262: Procedure Zread      z read  
263:   A=&HFB  
264:   Gosub Zout  
265:   Gosub Zin  
266:   Module=A  
267:   If A>15 And A<32  
268:     Ee=1  
269:   Else  
270:     Ee=0  
271:   Endif  
272:   Gosub Zin  
273:   Beginh=A  
274:   Gosub Zin
```

```

275: Begin1=A
276: Gosub Zin
277: Zendh=A
278: Gosub Zin
279: Zendl=A
280: Begin2=Beginh*256+Begin1
281: Zendt=Zendh*256+Zendl
282: Zsize=Zendt-Zbegin2+1
283: Return
284: '
285: ' *
286: ' * Output to zapper and wait for echo *****
287: ' *
288: '
289: Procedure Zout          zout
290:   Out #1,A
291:   B=0
292:   Repeat
293:     B=Inp?(1)
294:   Until B=-1
295:   X1=Inp(1)
296:   '
297:   If A(>)X1
298:     Bad=1
299:     Gosub Zerr
300:   Endif
301:   '
302: Return
303: '
304: ' *
305: ' * Input from zapper then echo back *****
306: ' *
307: '
308: Procedure Zin          zin
309:   Gosub Zin2
310:   Out #1,A
311: Return
312: '
313: ' *
314: ' * An error occured just get the error message *****
315: ' *
316: '
317: Procedure Zin2
318:   A=0
319:   Do
320:     A=Inp?(1)
321:     Exit If A=-1
322:     Ch#=Inkey$
323:     If Ch#=Chr$(&H1A) ! * ^X
324:       Gosub Finish
325:     Endif
326:   Loop
327:   '
328:   A=Inp(1)
329: Return

```

```

330: /
331: / *
332: / * Print error message *****
333: / *
334: /
335: Procedure Zerr
336: Cls
337: Print
338: Print Tab(20);'Error in echo back ** maybe the PROM failed'
339: Print Tab(17);'Failed address      Data Sent      Data Prom'
340: Gosub Zin2
341: N=A*256
342: Gosub Zin2
343: N=N+A
344: Print Tab(23);Hex$(N);
345: Gosub Zin2
346: Print Tab(47);Hex$(A);
347: Gosub Zin2
348: Print Tab(62);Hex$(A)
349: Print
350: Print Hex$(Xx),Hex$(Tbyte)
351: Box 0,0,639,76
352: Box 2,2,637,74
353: Gosub Zwait
354: X=Zendt+2
355: Bad=1
356: Return
357: /
358: / *
359: / * Zap and verify *****
360: / *
361: / * Sewritten 03/01/88 *****
362: / *
363: /
364: Zap:
365: Ba=0
366: Xx=0
367: A=Command
368: Gosub Zout
369: Dio
370: Print
371: Print ' BUSY '
372: /
373: If Tenc=1 Then
374:   Print 'ZAPPING'
375: Else
376:   Print 'VERIFYING'
377: Endif
378: /
379: Print
380: Print
381: Box 0,0,639,48
382: Box 2,2,637,46
383: /
384: Print

```

Zerr

ZAP

```

005: Y=0
006: For X=Begin to Zendt
007:   Inc Y
008:   If Y=64 And Temp<>1
009:     Print " *";
010:     Y=0
011:   Else
012:     If Y=62
013:       Print " *";
014:       Y=0
015:   Endif
016:   Error!
017:   A=Pack(Prom_addr+Xx)
018:   Byte=A
019:   Backup Load
020:
021:   If Bad=1
022:     A=A+2e
023:     Print
024:
025:   Next A
026:
027:   Print " *";
028:   In Backup
029:   A=DATA
030:   Home Text
031:   End
032:   Got! Load
033:
034:
035:
036:
037:   Print "Load data from PROM to memory *****"
038:
039:
040:
041:   Print
042:
043:   Print
044:
045:
046:
047:   Print
048:
049:   Print
050:
051:   Print
052:
053:   Print
054:
055:   Print
056:
057:   Print
058:
059:   Print
060:
061:   Print
062:
063:   Print
064:
065:   Print
066:
067:   Print
068:
069:   Print
070:
071:   Print
072:
073:   Print
074:
075:   Print
076:
077:   Print
078:
079:   Print
080:
081:   Print
082:
083:   Print
084:
085:   Print
086:
087:   Print
088:
089:   Print
090:
091:   Print
092:
093:   Print
094:
095:   Print
096:
097:   Print
098:
099:   Print
100:
101:   Print
102:
103:   Print
104:
105:   Print
106:
107:   Print
108:
109:   Print
110:
111:   Print
112:
113:   Print
114:
115:   Print
116:
117:   Print
118:
119:   Print
120:
121:   Print
122:
123:   Print
124:
125:   Print
126:
127:   Print
128:
129:   Print
130:
131:   Print
132:
133:   Print
134:
135:   Print
136:
137:   Print
138:
139:   Print
140:
141:   Print
142:
143:   Print
144:
145:   Print
146:
147:   Print
148:
149:   Print
150:
151:   Print
152:
153:   Print
154:
155:   Print
156:
157:   Print
158:
159:   Print
160:
161:   Print
162:
163:   Print
164:
165:   Print
166:
167:   Print
168:
169:   Print
170:
171:   Print
172:
173:   Print
174:
175:   Print
176:
177:   Print
178:
179:   Print
180:
181:   Print
182:
183:   Print
184:
185:   Print
186:
187:   Print
188:
189:   Print
190:
191:   Print
192:
193:   Print
194:
195:   Print
196:
197:   Print
198:
199:   Print
200:
201:   Print
202:
203:   Print
204:
205:   Print
206:
207:   Print
208:
209:   Print
210:
211:   Print
212:
213:   Print
214:
215:   Print
216:
217:   Print
218:
219:   Print
220:
221:   Print
222:
223:   Print
224:
225:   Print
226:
227:   Print
228:
229:   Print
230:
231:   Print
232:
233:   Print
234:
235:   Print
236:
237:   Print
238:
239:   Print
240:
241:   Print
242:
243:   Print
244:
245:   Print
246:
247:   Print
248:
249:   Print
250:
251:   Print
252:
253:   Print
254:
255:   Print
256:
257:   Print
258:
259:   Print
260:
261:   Print
262:
263:   Print
264:
265:   Print
266:
267:   Print
268:
269:   Print
270:
271:   Print
272:
273:   Print
274:
275:   Print
276:
277:   Print
278:
279:   Print
280:
281:   Print
282:
283:   Print
284:
285:   Print
286:
287:   Print
288:
289:   Print
290:
291:   Print
292:
293:   Print
294:
295:   Print
296:
297:   Print
298:
299:   Print
300:
301:   Print
302:
303:   Print
304:
305:   Print
306:
307:   Print
308:
309:   Print
310:
311:   Print
312:
313:   Print
314:
315:   Print
316:
317:   Print
318:
319:   Print
320:
321:   Print
322:
323:   Print
324:
325:   Print
326:
327:   Print
328:
329:   Print
330:
331:   Print
332:
333:   Print
334:
335:   Print
336:
337:   Print
338:
339:   Print
340:
341:   Print
342:
343:   Print
344:
345:   Print
346:
347:   Print
348:
349:   Print
350:
351:   Print
352:
353:   Print
354:
355:   Print
356:
357:   Print
358:
359:   Print
360:
361:   Print
362:
363:   Print
364:
365:   Print
366:
367:   Print
368:
369:   Print
370:
371:   Print
372:
373:   Print
374:
375:   Print
376:
377:   Print
378:
379:   Print
380:
381:   Print
382:
383:   Print
384:
385:   Print
386:
387:   Print
388:
389:   Print
390:
391:   Print
392:
393:   Print
394:
395:   Print
396:
397:   Print
398:
399:   Print
400:
401:   Print
402:
403:   Print
404:
405:   Print
406:
407:   Print
408:
409:   Print
410:
411:   Print
412:
413:   Print
414:
415:   Print
416:
417:   Print
418:
419:   Print
420:
421:   Print
422:
423:   Print
424:
425:   Print
426:
427:   Print
428:
429:   Print
430:
431:   Print
432:
433:   Print
434:
435:   Print
436:
437:   Print
438:
439:   Print
440:
441:   Print
442:
443:   Print
444:
445:   Print
446:
447:   Print
448:
449:   Print
450:
451:   Print
452:
453:   Print
454:
455:   Print
456:
457:   Print
458:
459:   Print
460:
461:   Print
462:
463:   Print
464:
465:   Print
466:
467:   Print
468:
469:   Print
470:
471:   Print
472:
473:   Print
474:
475:   Print
476:
477:   Print
478:
479:   Print
480:
481:   Print
482:
483:   Print
484:
485:   Print
486:
487:   Print
488:
489:   Print
490:
491:   Print
492:
493:   Print
494:
495:   Print
496:
497:   Print
498:
499:   Print
500:
501:   Print
502:
503:   Print
504:
505:   Print
506:
507:   Print
508:
509:   Print
510:
511:   Print
512:
513:   Print
514:
515:   Print
516:
517:   Print
518:
519:   Print
520:
521:   Print
522:
523:   Print
524:
525:   Print
526:
527:   Print
528:
529:   Print
530:
531:   Print
532:
533:   Print
534:
535:   Print
536:
537:   Print
538:
539:   Print
540:
541:   Print
542:
543:   Print
544:
545:   Print
546:
547:   Print
548:
549:   Print
550:
551:   Print
552:
553:   Print
554:
555:   Print
556:
557:   Print
558:
559:   Print
560:
561:   Print
562:
563:   Print
564:
565:   Print
566:
567:   Print
568:
569:   Print
570:
571:   Print
572:
573:   Print
574:
575:   Print
576:
577:   Print
578:
579:   Print
580:
581:   Print
582:
583:   Print
584:
585:   Print
586:
587:   Print
588:
589:   Print
590:
591:   Print
592:
593:   Print
594:
595:   Print
596:
597:   Print
598:
599:   Print
600:
601:   Print
602:
603:   Print
604:
605:   Print
606:
607:   Print
608:
609:   Print
610:
611:   Print
612:
613:   Print
614:
615:   Print
616:
617:   Print
618:
619:   Print
620:
621:   Print
622:
623:   Print
624:
625:   Print
626:
627:   Print
628:
629:   Print
630:
631:   Print
632:
633:   Print
634:
635:   Print
636:
637:   Print
638:
639:   Print
640:
641:   Print
642:
643:   Print
644:
645:   Print
646:
647:   Print
648:
649:   Print
650:
651:   Print
652:
653:   Print
654:
655:   Print
656:
657:   Print
658:
659:   Print
660:
661:   Print
662:
663:   Print
664:
665:   Print
666:
667:   Print
668:
669:   Print
670:
671:   Print
672:
673:   Print
674:
675:   Print
676:
677:   Print
678:
679:   Print
680:
681:   Print
682:
683:   Print
684:
685:   Print
686:
687:   Print
688:
689:   Print
690:
691:   Print
692:
693:   Print
694:
695:   Print
696:
697:   Print
698:
699:   Print
700:
701:   Print
702:
703:   Print
704:
705:   Print
706:
707:   Print
708:
709:   Print
710:
711:   Print
712:
713:   Print
714:
715:   Print
716:
717:   Print
718:
719:   Print
720:
721:   Print
722:
723:   Print
724:
725:   Print
726:
727:   Print
728:
729:   Print
730:
731:   Print
732:
733:   Print
734:
735:   Print
736:
737:   Print
738:
739:   Print
740:
741:   Print
742:
743:   Print
744:
745:   Print
746:
747:   Print
748:
749:   Print
750:
751:   Print
752:
753:   Print
754:
755:   Print
756:
757:   Print
758:
759:   Print
760:
761:   Print
762:
763:   Print
764:
765:   Print
766:
767:   Print
768:
769:   Print
770:
771:   Print
772:
773:   Print
774:
775:   Print
776:
777:   Print
778:
779:   Print
780:
781:   Print
782:
783:   Print
784:
785:   Print
786:
787:   Print
788:
789:   Print
790:
791:   Print
792:
793:   Print
794:
795:   Print
796:
797:   Print
798:
799:   Print
800:
801:   Print
802:
803:   Print
804:
805:   Print
806:
807:   Print
808:
809:   Print
810:
811:   Print
812:
813:   Print
814:
815:   Print
816:
817:   Print
818:
819:   Print
820:
821:   Print
822:
823:   Print
824:
825:   Print
826:
827:   Print
828:
829:   Print
830:
831:   Print
832:
833:   Print
834:
835:   Print
836:
837:   Print
838:
839:   Print
840:
841:   Print
842:
843:   Print
844:
845:   Print
846:
847:   Print
848:
849:   Print
850:
851:   Print
852:
853:   Print
854:
855:   Print
856:
857:   Print
858:
859:   Print
860:
861:   Print
862:
863:   Print
864:
865:   Print
866:
867:   Print
868:
869:   Print
870:
871:   Print
872:
873:   Print
874:
875:   Print
876:
877:   Print
878:
879:   Print
880:
881:   Print
882:
883:   Print
884:
885:   Print
886:
887:   Print
888:
889:   Print
890:
891:   Print
892:
893:   Print
894:
895:   Print
896:
897:   Print
898:
899:   Print
900:
901:   Print
902:
903:   Print
904:
905:   Print
906:
907:   Print
908:
909:   Print
910:
911:   Print
912:
913:   Print
914:
915:   Print
916:
917:   Print
918:
919:   Print
920:
921:   Print
922:
923:   Print
924:
925:   Print
926:
927:   Print
928:
929:   Print
930:
931:   Print
932:
933:   Print
934:
935:   Print
936:
937:   Print
938:
939:   Print
940:
941:   Print
942:
943:   Print
944:
945:   Print
946:
947:   Print
948:
949:   Print
950:
951:   Print
952:
953:   Print
954:
955:   Print
956:
957:   Print
958:
959:   Print
960:
961:   Print
962:
963:   Print
964:
965:   Print
966:
967:   Print
968:
969:   Print
970:
971:   Print
972:
973:   Print
974:
975:   Print
976:
977:   Print
978:
979:   Print
980:
981:   Print
982:
983:   Print
984:
985:   Print
986:
987:   Print
988:
989:   Print
990:
991:   Print
992:
993:   Print
994:
995:   Print
996:
997:   Print
998:
999:   Print
1000:  Print

```

upload

clone


```
550:   Print Chr$(7);'Must Be 4 places long'
551:   Goto Change1
552: Endif
553: Gosub Htod
554: I1=H
555: I2=L
556: Change2:
557: Input 'New ending address (Hex input) ';A$
558: If Len(A$)<>4
559:   Print Chr$(7);'Must be 4 places long'
560:   Goto Change2
561: Endif
562: Gosub Htod
563: A=I1
564: Gosub Zout
565: A=I2
566: Gosub Zout
567: Begint=I1*256+I2
568: A=H
569: Gosub Zout
570: A=L
571: Gosub Zout
572: Zendt=H*256+L
573: Goto Gmenu
574: '
575: ' *
576: ' * Change hex to decimal *****
577: ' * My version. This version works the way I want it to. *****
578: ' *
579: '
580: Procedure Htod
581:   I=0
582:   For D=1 To 4
583:     C=Asc(Mid$(A$,D))
584:     If C>=48 And C<=57
585:       C=C-48
586:     Endif
587:     If C>=65 And C<=70
588:       C=C-55
589:     Endif
590:     If C>=97 And C<=102
591:       C=C-87
592:     Endif
593:     I=16*I+C
594:     If D=2
595:       H=I
596:       I=0
597:     Endif
598:   Next D
599:   L=I
600: Return
601: '
602: ' *
603: ' * Fill Malloc with FF *****
604: ' *
```

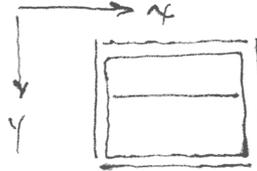
```
605: '  
606: Zfill:  
607: Gosub Zfill2  
608: Goto Gmenu  
609: '  
610: Procedure Zfill2  
611:   Cls  
612:   Print At(3,2);"Filling Reserved memory area."  
613:   Box 0,0,639,48  
614:   Box 2,2,637,46  
615:   Yy=0  
616:   Print At(0,5);  
617:   For X=Begin To Zendt  
618:     Inc Yy  
619:     Poke (Prom_addr+X),&HFF  
620:     If Yy=63  
621:       Print ' *';  
622:       Yy=0  
623:     Endif  
624:   Next X  
625: Return  
626: '  
627: ' *  
628: ' * Directory command *****  
629: ' *  
630: '  
631: Zdir: ZDIR  
632: Old$=N$  
633: Showm  
634: Fileselect D$,N$,0$  
635: If N$=""  
636:   N$=Old$  
637: Endif  
638: Hidem  
639: Goto Gmenu  
640: '  
641: ' *  
642: ' * Information about program. *****  
643: ' *  
644: '  
645: Zinfo: ZINFO  
646: Cls  
647: Box 0,0,639,399  
648: Box 2,2,637,397  
649: Print At(5,4);"SmartZap EPROM Burner By Micro Kit Electronics."  
650: Print At(5,5);"Zap-ST written By N.Cherry 03/01/88 Version ";Version;"B."  
651: Print At(5,6);"For use with the Atari 520ST. (or a 1040ST)"  
652: Print At(5,7);"BASIC Version will now handle any prom up to a 27512 or equivalent."  
653: Print At(5,8);"C Version should also handle the same proms, but be more portable."  
654: Print At(10,10);"Set for : 9600 baud Done in the program with software."  
655: Print At(20,11);"1 Start bit"  
656: Print At(20,12);"8 data bits"  
657: Print At(20,13);"1 Stop bit"  
658: Print At(20,14);"No parity"  
659: Print At(20,15);"RTS/CTS on, Xon/Xoff off."
```

```
660: Print At(5,17);'The largest size PROM that SmartZap can currently handle: 64K 27512.'
661: Print At(5,18);'Version ';Version;' software can handle 64K 27512 PROMS'
662: Print At(5,19);'The Letter after a version number indicates what language I wrote it in.'
663: Print At(5,20);'      A - Assembly          B - GFA BASIC          C - 'C''
664: Goto Zwaitkey
665: '
666: ' *
667: ' * Wait for a key to be hit. *****
668: ' *
669: '
670: Zwaitkey:
671: Print
672: Gosub Zwait
673: Goto Gmenu
674: '
675: Procedure Zwait
676:   Print At(26,24);'Please hit any key to continue.'
677:   Z=Inp(2)
678: Return
679: '
680: ' *
681: ' * print Malloc'd area
682: ' *
683: '
684: Zedit:
685: T$='Print'
686: T=0
687: Pos=0
688: B=0
689: Cls
690: '
691: Box 0,0,639,399 ! * main box
692: Box 2,2,637,38 ! * Title box
693: Box 2,40,68,70 ! * Addr box
694: Box 70,40,468,70 ! * addr of low byte
695: Box 470,40,637,70 !* ASCII addr box
696: Box 2,72,68,342 ! * Hi byte addr box
697: Box 70,72,468,342 !* byte box
698: Box 470,72,637,342!* ASCII box
699: Box 2,344,637,397 !* Message box
700: '
701: Posit=1
702: Print At(3,Posit+1);' Smart Zap ';T$;' Screen'
703: Print At(4,Posit+3);'ADDR';
704: Print At(11,Posit+3);'00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F'
705: Print At(62,Posit+3);'0123456789ABCDEF';
706: Repeat
707:   ' Zupdate:
708:   Posit=Posit+4
709:   For X=0 To 15
710:     Inc Posit
711:     Print At(2,Posit);' $';Hex$(Pos);
712:     Print At(10,Posit);
713:     For Y=0 To 15
714:       Inc Z
```

```

715:     Gosub Wtob(Pos+Y)
716:     If Byte<16
717:         Print " 0";Hex$(Byte);
718:     Else
719:         Print " ";Hex$(Byte);
720:     Endif
721: Next Y
722: Print At(61,Posit);"!";
723: For Y=0 To 15
724:     Inc B
725:     Gosub Wtob(Pos+Y)
726:     If Byte=>32 And Byte<=126
727:         Print Chr$(Byte);
728:     Else
729:         Print ".";
730:     Endif
731: Next Y
732: Print At(78,Posit);"!";
733: Pos=Pos+16
734: Next X
735: Print At(4,Posit+2);"Please hit any
736: A=Inp(2)
737: Posit=1
738: Until Pos>Zendt Or A=#H1A
739: Goto Gmenu
740: '
741: ' *
742: ' * Smart Zap command selection menu
743: ' *
744: '
745: Procedure Zmenu
746:     Posit=0
747:     B$=Hex$(Begint)
748:     E$=Hex$(Zendt)
749:     M$=Hex$(Module)
750:     F=Fre(0)
751:     Cls
752:     Box 0,0,639,399 ! Large outer box
753:     Box 2,2,637,38 ! used as inner frame
754:     Box 2,40,637,124 ! used to box information
755:     Box 2,126,637,397 ! used to box selection
756:     '
757:     Print At(12,Posit+2);"Atari ST EPROM BURNER PROGRAM Version ";Version;"B By N.Cherry"
758:     '
759:     Print At(11,Posit+4);"SMART ZAP";Tab(31);"MODULE ID           ";M$
760:     Print At(11,Posit+5);"MEMORY AVAILABLE ";F;
761:     Print Tab(31);"BEGIN AREA FOR ZAPPER ";Begint
762:     Print At(11,Posit+6);"PROM SIZE           ";Zsize;
763:     Print Tab(31);"ENDING AREA           ";Zendt
764:     Print At(11,Posit+7);"FILE NAME           ";O$
765:     Print At(11,Posit+10);"Z) ZAP FROM MEMORY";
766:     Print Tab(31);"U) UPLOAD PROM TO MEMORY"
767:     Print At(11,Posit+12);"E) VERIFY ERASE";
768:     Print Tab(31);"V) VERIFY FROM MEMORY"
769:     Print At(11,Posit+14);"S) SAVE MEMORY TO DISK";

```



640 = 80c
400 = 16c ?

Title

Info

Menu

```
770: Print Tab(31);'L)  LOAD FROM DISK TO MEMORY'
771: Print At(11,Posit+16);'F)  FILL STORED DATA AREA';
772: Print Tab(31);'P)  PRINT UPLOADED STRING'
773: Print At(11,Posit+18);'D)  DIRECTORY OF ';D#;
774: Print Tab(31);'I)  GENERAL INFORMATION'
775: Print At(11,Posit+20);'C)  CHANGE ID MODULE';
776: If Ee=1
777:   Print Tab(31);'A)  CLEAN EEPROM'
778:   Print At(11,Posit+22);'X)  EXIT FROM PROGRAM'
779: Else
780:   Print Tab(31);'X)  EXIT FROM PROGRAM'
781: Endif
782: Print At(26,Posit+24);'Command => ';
783: Return
784: '
785: ' *
786: ' * Get a key
787: ' *
788: '
789: Procedure Gkl
790:   Repeat
791:     A=Inp?(2)
792:     Until A<>0
793:     A=Inp(2)
794:   Return
795: '
796: ' *
797: ' * Push character to upper case
798: ' *
799: '
800: Procedure Toupper
801:   If A>96 And A<123
802:     A=A-32
803:   Endif
804:   A$=Chr$(A)
805:   Print Chr$(A)
806: Return
807: '
808: ' *
809: ' * unknown error handler
810: ' *
811: '
812: Zerror:
813: Showm
814: B$=' OK '
815: S$='Unknown System error'
816: Alert 2,S$,1,B$,A
817: Gosub Finish
818: '
819: ' *
820: ' * Not enough to memory to continue
821: ' *
822: '
823: Procedure Nomen
824:   Showm
```

```
825:   B$='EXIT'
826:   Ms$='You'll have to elimante a fewlaccessories or ramdisk. As youldo not have enough memory toIcontinue, sorry!!'
827:   Print Chr$(7)
828:   Alert 3,Ms$,1,B$,A
829:   Print 'Error #';Err
830:   Gosub Zwait
831:   Gosub Finish
832: Return
833: '
834: ' *
835: ' * File_selector
836: ' *
837: '
838: Procedure F_select
839:   Old$=N$
840:   Showm
841:   Repeat
842:     Fileselect D$,N$,O$
843:     N$=Mid$(O$,4)
844:     Until N$(<)' ' Or O$=''
845:   Hidem
846: Return
847: '
848: '
849: ' *
850: ' * End of program *****
851: ' *
852: '
853: Zend:
854: Showm
855: B$=' EXIT IRETURN'
856: Ms$=' Do you want toIRETURN to ZAP-ST or! Exit to Desktop?'
857: Print Chr$(7)
858: Alert 0,Ms$,1,B$,A
859: If A=2
860:   Hidem
861:   Goto Gmenu
862: Endif
863: Gosub Finish
864: '
865: ' * this helps with the malloc routine
866: '
867: Procedure Finish
868:   Void=Fn Mfree(Prom_addr)
869:   Reserve Fre(0)+Resvmem-255
870:   End
871: Return
872: '
873: ' * End of progam *****
874: '
875: Procedure Promfill
876:   For X=0 To Promsize Step 4
877:     Lpoke Prom_addr+X,-1
878:   Next X
879: Return
```

```
880: '  
881: ' *  
882: ' * Word to byte function  
883: ' *  
884: '  
885: Procedure Wtob(Address)  
886:   Byte=Peek(Prom_addr+Address)  
887: Return  
888: '  
889: Procedure A_drive  
890:   D$='A:\*. *'  
891:   D1$='A:\'  
892: Return  
893: '  
894: Procedure B_drive  
895:   D$='B:\*. *'  
896:   D1$='B:\'  
897: Return  
898: '  
899: Procedure C_drive  
900:   D$='C:\*. *'  
901:   D1$='C:\'  
902: Return  
903: '  
904: Procedure D_drive  
905:   D$='D:\*. *'  
906:   D1$='D:\'  
907: Return  
908:
```

Total lines printed: 907

Total characters counted: 16712

SMART ZAP

SMART ZAP OPERATION

External mode, Serial or Parallel (Dip switch 3 on, 8 off)

When Smart Zap is powered up in the external mode it will wait in the command mode, indicated by the yellow lamp being on, for a valid command to be received. When a command is received it will be executed immediately, and upon successful completion the program will return to the command mode. Following are the valid commands:

Command Word		Mode
Hex	Decimal	
F0H	240 (or Zap button)	Zap
F5H	245 (or Vfy button)	Verify
F6H	246	Verify erase only
F7H	247 (or both Zap & Vfy)	Upload
F9H	249	Setup Changes
FAH	250	Timing Changes
FBH	251	Module Read
FFH	255	Null
61H	97	Error Return on
53H	83	Motorola Hex ←
50H	80 (error return mode only)	End Binary File ←
3BH	59	Mostek Hex ←
3AH	58	Intel Hex ←

Verifies in Binary

Proper execution of the chosen mode will be indicated by a green lamp. While the routine is being performed the green and red lamps will alternately flash. The lamps alternate each time 32 characters have been received. If you wish to return to the command mode, while in one of the above modes, simply press the Zap and Verify buttons at the same time and release them. This should give you a yellow lamp, and put you back in the command mode.

If an I/O input failure has occurred while executing a mode, the red lamp will flash on and off. The indicated failures are:
2 blinks, unknown command received,
4 blinks, checksum failure (hex format only),
5 blinks, bad hex format,
6 blinks, input buffer overflow,
7 blinks, no stop bit received.

ERROR RETURN MODE:

In the 3 line echo mode an error is indicated by echoing the complement of the received byte, then the following information will be sent: the address where the error occurred, in high byte low byte order, the data byte as received by the Zapper and the data byte as read from the programmed device (ie EPROM, EEPROM).

In the 5 line mode, "F" (46H) is sent to the host computer if a failure occurs, then the Zapper sends the same error data as in the 3 line mode. When the end of file is reached the Zapper will send a "P" (50H) if the command has not failed.

SMART ZAP

The following are examples of how to use the different commands:

Zap (binary data): *send BINARY*

Send F0H (240) followed by the data to be zapped. Or you can push the ZAP button followed by the zapping data.

Verify (binary data):

Send F5H (245) followed by the data to be verified with the chip in the zap socket, or push the Verify button followed by the zapping data.

Verify Erase:

Send F6H (246) only.

F6 Receive back
F7 send F7

Upload (binary data):

Send F7H (247) or push the Zap and Verify button at the same time. Now send 40H (64) or push the Zap button, if the data to be sent out is from the master socket, or send 80H (128) or push the verify button, if the data to be sent out is from the clone socket.

Setup Changes:

Send F9H (249) then the start address high byte, the start address low, the end address high, and then the end address low. The setup change can be used for zapping, verifying, or uploading from one location thru all locations.

Timing Changes:

Send FAH (250) then the overzap byte, the loopmax byte, then the delay time highbyte then lowbyte. Overzap byte is used in the fast zapping mode (dip switch 2=on) Overzap pulse equals the overzap byte times the loop count. The loopmax is the number of times the program will try to zap a location. Overzap byte and loopmax byte are standard Intel designations, set to that standard unless changed. See an Intel manual for a complete explanation. The delay time bytes are a value used to generate the 1 millisecond delay when zapping, it is normally set to 0133H (307) but can be made smaller if necessary.

Module ID Code:

get Hi Hi
Lo Lo
Send FBH (251) then receive back the module ID code followed by the Beginning Address (high then low byte) and Ending Address of range to be zapped.

Null:

Sending FFH (255) does nothing in the command mode. This command is used to compensate for software discrepancies in some computers in the area of the transmit buffer.

Error Return on:

Send 61H (97). This command enables an enhanced error response mode for the first operation which follows it only. When an error occurs it causes the address where the error occurred, in high byte low byte order, to be sent back to the host computer, followed by the data as received from the host computer then the data as read from the programmed device (ie EPROM, EEPROM).

Zap Motorola Hex:

The first byte in a Motorola Hex file is always 53H (80), when Smart Zap receives this command it will assume that you want to use the following data to zap a device unless you have performed the Verify Hex sequence.

End Binary File:

This command is only used if Error Return mode has been enabled. Send 50H (80) to indicate the end of a binary file has been reached. This command can't be used to end a file before the preset number of bytes have been sent. The preset number of bytes is determined either by the default number of bytes in the device being programmed or by the range allowed by use of the Setup Changes command.

Zap Mostek Hex:

Same as Motorola hex except the first byte is always 3BH (59).

Zap Intel Hex:

Same as Motorola hex except the first byte is always 3AH (58).

Verify Hex:

Hold down the Verify button before powering up the Smart Zap unit, continue to hold the button down until the Good lamp stays on, about 2 seconds. When the Verify button is released the Green lamp will go off, and the unit is ready to verify from a hex file.

***NOTE:** If echo mode is on then Smart Zap will expect an echo of all words it sends, and it will echo all words it receives. If a failure occurs while in echo mode the data will be complemented before it is echoed back, this is a software signal to the host computer that a failure has occurred.

* ----- *

- 1) SMARTZAP.TOS - A terminal interface to the SMARTZAP EPROM burner, Allows uploading and down loading from EPROM in various formats. ✓
- 2) SIMPLASH.TTP - This program will assemble programs for various Micro-processors (68xx, 65xx, 658xx, 80xx, Z80). ✓
- 3) DISASSEM.TTP - This program disassembles the above assembled programs allowing them to be sent to various devices. ✓
- 4) SPLIT.TTP - This program will split 68000 programs into even and odd files so they be loaded into EPROM. ✓
- 5) BSRX10.PR6 - This program will be used as an interface to the 6802 BRS microcontroller. This unit will have an on board clock, non-volatile SRAM, a serial port and an optional printer port. Other upgrades allow the use of a 6809 uP and upload and download capabilities of events. } K
- 6) SEARCH.TTP - This program will search all directories for the occurrence of the input string. Should include a wildcard for search. ✓ FIND
- 7) CPRINT.TTP - This program will take a file and send it to the printer, options should include no header and no line numbers. ✓

* ----- *

EQUIPMENT NEEDED

* ----- *

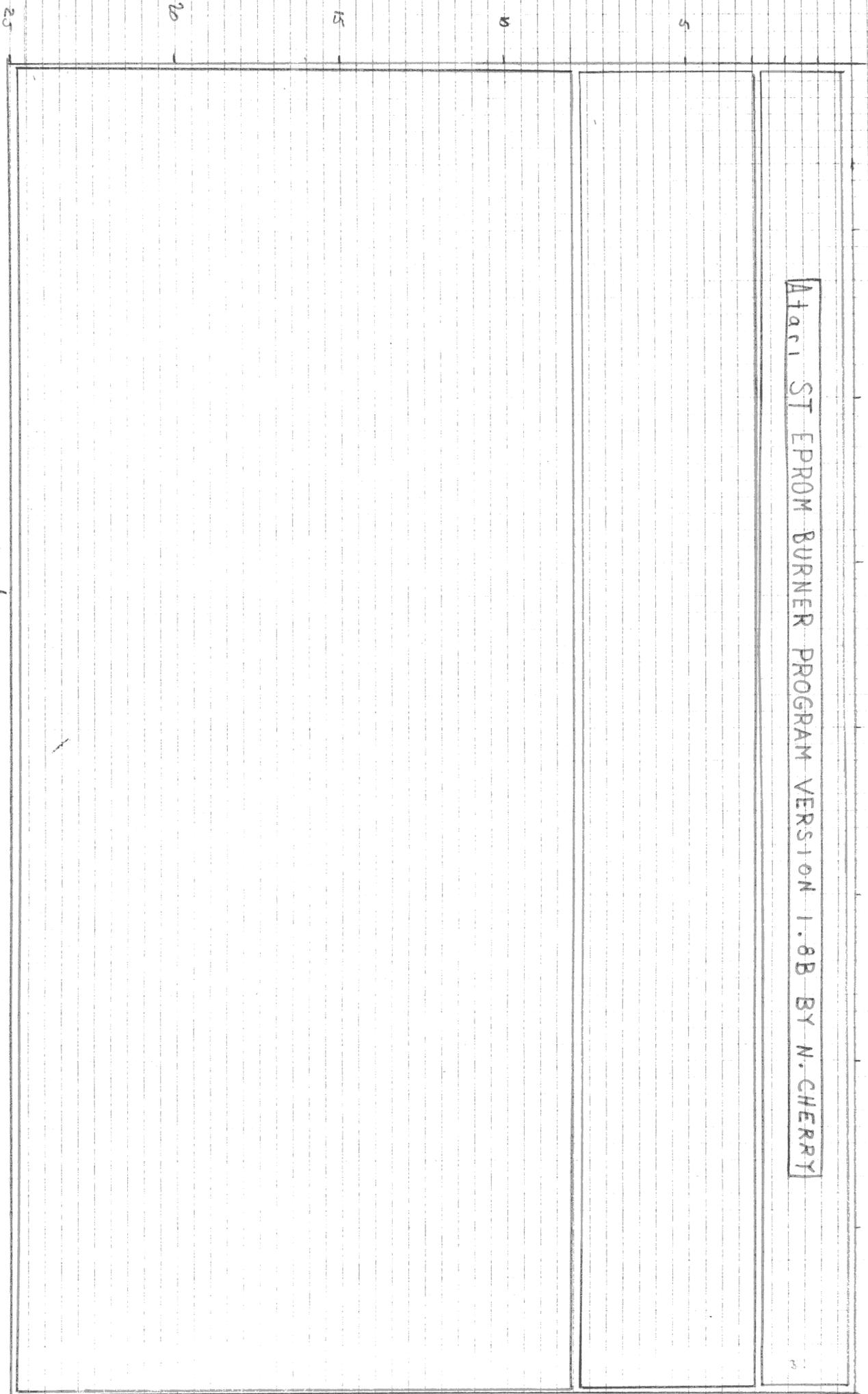
Atari ST, 1-4 meg upgrade, 2 floppy drives, Hard drive, 3/12/24 modem, printer, C-compiler, Resource Construction Set, Disassembler, Assembler, Editor, and debugger. For work on the Atari 8 bit series the Atari 8 bit emulator, IBM emulator, and Apple emulator.

For the X10 project:

X10 controllers, control units, uProcessor boards, sensors, and various hardware } K

* ----- *

Atari ST EPPROM BURNER PROGRAM VERSION 1.8B BY N. CHERRY



8 PIXELS

16 PIXELS



1 char

400

25 X



IBM PC MONO & TERMINALS

24



