```
1  ************************************************************
2  *  version 1.0   01-May-2020                               *
3  ************************************************************
4  *  monitor source code for 6809 MICROPROCRSSOR KIT 2020    *
5  *  assemble with bs9 assembler                             *
6  *  monitor source code was written by Wichit Sirichote,    *
7  *  wichit.sirichote@gmail.com                              *
8  *                                                          *
9  *  The object file is EPROM/EEPROM image                   *
10 *  more update and technical information:                  *
11 *   kswichit.com/6809/6809.htm                             *
12 ************************************************************
13
14 CPU = 6809
15 STORE $8000,$8000,"newmon.rom"
16 STORE $8000,$8000,"newmon.s19",s19
17
18 GPIO1     = $8000
19 PORT0     = $8001
20 PORT1     = $8002
21 PORT2     = $8003
22
23 LCD_cwr   = $9000
24 LCD_dwr   = $9001
25 LCD_crd   = $9002
26 LCD_drd   = $9003
27
28 ACIAPORT  = $A000
29
30           SETDP 0
31           ORG $0000
32
33 ***********************************
34 * area for saving interrupt stack *
35 ***********************************
36
37 USER_START
38 USER_CC   RMB 1  ;  0
39 USER_A    RMB 1  ;  1
40 USER_B    RMB 1  ;  2
41 USER_DP   RMB 1  ;  3
42 USER_X    RMB 2  ;  4
43 USER_Y    RMB 2  ;  6
44 USER_U    RMB 2  ;  8
45 USER_PC   RMB 2  ;  a
46 USER_END
47 USER_D    = USER_A
48
49 SAVE_SP   RMB 2
50 SAVE_PC   RMB 2
51 CPC       RMB 2  ; current PC
52 SIGN      RMB 1  ; +/- flag
53
54
55 key:      RMB 1
56 hit:      RMB 1
```

```
flag:     RMB 1
tick:     RMB 1
checksum: RMB 1
num:      RMB 2
start:    RMB 2
_end:     RMB 2
state:    RMB 1
buffer:   RMB 6

_virq  = $7ff0
_vnmi  = $7ff3
_vfirq = $7ff6


*  ===========
   ORG   $fff0
*  ===========

   FDB   _reset    ; fff0 : trap on 6309
   FDB   _reset    ; fff2 : SWI3
   FDB   _reset    ; fff4 : SWI2
   FDB   _vfirq    ; fff6 : FIRQ
   FDB   _virq     ; fff8 : IRQ
   FDB   _swi_serv ; fffa : SWI
   FDB   _vnmi     ; fffc : NMI
   FDB   _reset    ; fffe : RESET

*  ===========
   ORG   $c000
*  ===========

*****************
SUBROUTINE _reset
*****************
   lds   #$7ff0      ; init stack pointer
   lda   #$7e        ; store jmp instruction
   ldx   #_irq_serv
   sta   _virq       ; JMP
   stx   _virq+1     ; irq_serv
   lda   #$3b        ; store rti instruction
   ldx   #0
   sta   _vnmi       ; must be modified when testing
   sta   _vfirq      ; with ram vector
   stx   _vnmi+1
   stx   _vfirq+1
   jmp   _main
ENDSUB


********************
SUBROUTINE _irq_serv
********************
   inc tick
   rti
ENDSUB

```

```
113
114  ***********
115  * Stack   *
116  *---------*
117  * a: PC   *
118  * 9:      *
119  * 8: U    *
120  * 7:      *
121  * 6: Y    *
122  * 5:      *
123  * 4: X    *
124  * 3: DP   *
125  * 2: B    *
126  * 1: A    *
127  * 0: CC   *
128  ***********
129
130
131  ********************
132  SUBROUTINE _swi_serv
133  ********************
134   ldx   #USER_START
135  .loop
136   puls  d
137   std   ,x++
138   cmpx  #USER_END
139   bne   .loop
140   std   SAVE_PC
141   lds   SAVE_SP
142   jmp   _key_PC
143  ENDSUB
144
145
146  convert
147   FCB 189,48,155,186,54,174,175,56,191,190
148   FCB 63,167,141,179,143,15
149
150
151  *********************
152  SUBROUTINE _LCD_Ready
153  *********************
154   pshs  b,y
155   ldy   #2000      ; timeout
156  .loop
157   ldb   LCD_crd
158   bpl   .return
159   leay  -1,y
160   bne   .loop
161  .return
162   puls  b,y,pc
163  ENDSUB
164
165
166  *********************
167  SUBROUTINE _LCD_Clear
168  *********************
```

```
 jsr   _LCD_Ready
 ldb   #1
 stb   LCD_cwr
 rts
ENDSUB


LCD_Row_Val
 BYTE  $80,$c0,$94,$d4

**********************
SUBROUTINE _LCD_Row_Col
**********************
* Input: a = row  0-3 *
*        b = col      *
**********************
 jsr   _LCD_Ready
 ldx   #LCD_Row_Val
 addb  a,x
 stb   LCD_cwr
 rts
ENDSUB


********************
SUBROUTINE _LCD_Init
********************
 jsr   _LCD_Ready
 ldb   #$38
 stb   LCD_cwr
 jsr   _LCD_Ready
 ldb   #$0c
 stb   LCD_cwr
 jsr   _LCD_Clear
 clra
 clrb
 bsr   _LCD_Row_Col    ; LCD_XY(0,0);
 ldy   #1000
 jmp   _delay
ENDSUB


********************
SUBROUTINE _LCD_Puts
********************
* Input: x = char* *
********************
 jsr   _LCD_Ready
 ldb   ,x+
 beq   .return
 stb   LCD_dwr
 bra   _LCD_Puts
.return
 rts
ENDSUB

```

```
225
226  ********************
227  SUBROUTINE _LCD_Putc
228  ********************
229  * Input: b = char  *
230  ********************
231   jsr   _LCD_Ready
232   stb   LCD_dwr
233   rts
234  ENDSUB
235
236
237  *****************
238  SUBROUTINE _delay
239  *****************
240  * Input: y = time
241  ***************** ; 6809  6309
242                    ; ----------
243   leay  -1,y       ;    4     4
244   bne   _delay     ;    3     3
245   rts
246
247  ***********************
248  SUBROUTINE _seg_refresh
249  ***********************
250   lda   #%1111 1110 ; select mask
251   ldx   #buffer
252  .loop
253   ldy   #10         ; brightness
254   ldb   ,x+         ; buffer[i]
255   std   PORT1       ; PORT1 = a  PORT2 = b
256   cmpb  #48         ; 1
257   beq   .sega
258   cmpb  #56         ; 7
259   beq   .sega
260   leay  8,y         ; increase brightness
261  .sega
262   bsr   _delay
263   clrb
264   stb   PORT2       ; PORT2 = 0
265   leay  10,y
266   jsr   _delay
267   orcc  #1          ; set carry
268   rola              ; rotate mask
269   cmpa  #%1011 1111
270   bne   .loop
271   rts
272  ENDSUB
273
274
275  ********************
276  SUBROUTINE _kbd_scan
277  ********************
278  * Output: key      *
279  ********************
280   lda   #%1111 1110 ; select mask
```

```
 clr   key
.loopo
 sta   PORT1
 ldb   PORT0
 ldx   #6          ; index
.loopi
 lsrb
 bcc   .return     ; keypress
 inc   key
 leax  -1,x
 bne   .loopi
 orcc  #1          ; set carry
 rola              ; rotate mask
 cmpa  #%1011 1111
 bne   .loopo
 ldb   #-1         ; no key
 lda   PORT0
 bita  #$40
 bne   .kbda
 ldb   #$24        ; GO key
.kbda
 stb   key
.return
 rts
ENDSUB


****************
SUBROUTINE _scan
****************
 pshs  x,y
 bsr   _seg_refresh
 bsr   _kbd_scan
 ldb   key
 puls  x,y,pc
ENDSUB


***********************
SUBROUTINE _dot_address
***********************
 ldd   buffer
 anda  #$bf
 andb  #$bf
 std   buffer
 ldd   buffer+2
 ora   #$40
 orb   #$40
 std   buffer+2
 ldd   buffer+4
 ora   #$40
 orb   #$40
 std   buffer+4
 rts
ENDSUB

```

```
337
338  ********************
339  SUBROUTINE _dot_data
340  ********************
341   ldd   buffer
342   ora   #$40
343   orb   #$40
344   std   buffer
345   ldd   buffer+2
346   anda  #$bf
347   andb  #$bf
348   std   buffer+2
349   ldd   buffer+4
350   anda  #$bf
351   andb  #$bf
352   std   buffer+4
353   rts
354  ENDSUB
355
356
357  ********************
358  SUBROUTINE _byte_seg
359  ********************
360  * Input : b = byte *
361  * Output: d = seg  *
362  ********************
363   ldx   #convert
364   tfr   b,a
365   anda  #15
366   lda   a,x        ; low  nibble
367   lsrb
368   lsrb
369   lsrb
370   lsrb
371   ldb   b,x        ; high nibble
372   rts
373  ENDSUB
374
375
376  ****************
377  SUBROUTINE _hex4
378  ****************
379  * Input: d = word
380  ******************
381   pshs  a          ; save high byte
382   bsr   _byte_seg
383   std   buffer+2
384   puls  b          ; pull high byte
385   bsr   _byte_seg
386   std   buffer+4
387   rts
388  ENDSUB
389
390
391  ***********************
392  SUBROUTINE _read_memory
```

```
393  ***********************
394   ldd   CPC
395   jsr   _hex4
396   ldb   [CPC]
397   bsr   _byte_seg
398   std   buffer
399   jmp   _dot_data
400  ENDSUB
401
402
403  ***********************
404  SUBROUTINE _key_address
405  ***********************
406   clr   hit
407   ldb   #1
408   stb   state
409   jsr   _read_memory
410   jmp   _dot_address
411  ENDSUB
412
413
414  ********************
415  SUBROUTINE _key_data
416  ********************
417   clr   hit
418   ldb   #2
419   stb   state
420   jsr   _read_memory
421   jmp   _dot_data
422  ENDSUB
423
424
425  ********************
426  SUBROUTINE _key_plus
427  ********************
428   ldb   state      ; if (state == 1 || state == 2)
429   beq   .plusa
430   cmpb  #2
431   bhi   .plusa
432   ldd   CPC        ; PC++;
433   addd  #1
434   std   CPC
435   jsr   _read_memory
436   jmp   _key_data
437  .plusa
438   cmpb  #4         ; if (state == 4)
439   bne   .plusb
440   ldd   num        ; start = num;
441   std   start
442   clr   hit        ; hit = 0;
443   clr   SIGN       ; positive
444   rts
445  .plusb
446   cmpb  #5
447   bne   .plusc
448   inc   state      ; state = 6;
```

```
449   ldd   num        ; start = num;
450   std   start
451   clr   hit        ; hit = 0;
452   ldb   #$8f       ; buffer[0]=0x8f;
453   stb   buffer
454   rts
455  .plusc
456   cmpb  #6
457   bne   .return
458   inc   state      ; state = 7;
459   clr   hit        ; hit = 0;
460   ldb   #$b3       ; buffer[0] = 0xb3;
461   stb   buffer
462   ldd   num        ; end = num;
463   std    _end
464   cmpd  start      ; if (end <= start) print_error();
465   bhi   .return
466   jsr _print_error
467  .return
468   rts
469  ENDSUB
470
471
472  *********************
473  SUBROUTINE _key_minus
474  *********************
475   ldb   state      ; if (state == 1 || state == 2)
476   beq   .mina
477   cmpb  #2
478   bhi   .mina
479   ldd   CPC        ; PC--
480   subd  #1
481   std   CPC
482   jsr   _read_memory
483   jmp   _key_data
484  .mina
485   cmpb  #4         ; if (state == 4)
486   bne   .return
487   ldd   num        ; start = num;
488   std   start
489   clr   hit        ; hit = 0;
490   ldb   #-1
491   stb   SIGN       ; negative
492  .return
493   rts
494  ENDSUB
495
496
497  ********************
498  SUBROUTINE _data_hex
499  ********************
500   clrb
501   tst   hit        ; if (hit == 0) x = 0;
502   beq   .hexa
503   ldb   [CPC]      ; x = *PC;
504   lslb             ; x <<= 4;
```

```
 lslb
 lslb
 lslb
.hexa
 orb   key        ; x |= key;
 stb   [CPC]      ; *PC = x;
 ldb   #1
 stb   hit
 jsr   _read_memory
 jmp   _dot_data
ENDSUB


******************
SUBROUTINE _key_PC
******************
 ldd   SAVE_PC
 std   CPC
 jmp   _key_data
ENDSUB


***********************
SUBROUTINE _hex_address
***********************
 clra
 clrb
 tst   hit        ; if (hit == 0) PC = 0;
 bne   .hexa
 std   CPC
.hexa
 incb             ; hit = 1;
 stb   hit
 ldd   CPC        ; PC <<= 4;
 lslb
 rola
 lslb
 rola
 lslb
 rola
 lslb
 rola
 orb   key        ; PC |= key;
 std   CPC
 jsr   _read_memory
 jmp   _dot_address
ENDSUB


***********************
SUBROUTINE _print_error
***********************
 clra
 clrb
 std   buffer
 ldb   #3         ; r
```

```
561   std   buffer+2
562   ldd   #$038f      ; r e
563   std   buffer+4
564   clr   state
565   rts
566
567
568  ******************
569  SUBROUTINE _key_go
570  ******************
571   ldb   state       ; if (state == 1 || state == 2)
572   beq   .goa
573   cmpb  #2
574   bhi   .goa
575   sts   SAVE_SP
576   ldx   USER_X
577   ldy   USER_Y
578   ldu   USER_U
579   ldd   CPC
580   pshs  d
581   ldd   USER_A
582   pshs  d
583   lda   USER_CC
584   tfr   a,cc
585   lda   USER_DP
586   tfr   a,dp
587   puls  d,pc
588
589  .goa
590   cmpb  #4          ; if (state == 4)
591   bne   .goc
592   ldd   start
593   tst   SIGN       ; if (SIGN) start -= num;
594   bpl   .pos
595   subd  num
596   bra   .gob
597  .pos
598   addd  num         ; else start += num;
599  .gob
600   std   start
601   jsr   _hex4        ; _hex4(start)
602   clr   hit
603   rts
604  .goc
605   cmpb  #7          ; if (state == 7)
606   bne   .return
607   ldx   num
608   stx   CPC         ; PC = num
609   ldy   start
610  .loop              ; for (i=0; i<temp; i++)
611   cmpy  _end
612   bhs   .exit
613   ldb   ,y+         ; num[i] = start[i]
614   stb   ,x+
615   bra   .loop
616  .exit
```

```
617   jsr   _read_memory
618   jsr   _dot_data
619   ldb   #2         ; state = 2;
620   stb   state
621  .return
622   rts
623  ENDSUB
624
625
626  ******************
627  SUBROUTINE _key_reg
628  ******************
629   jsr   _clear_buffer
630   ldb   #$ad
631   stb   buffer+3
632   ldd   #$8f03
633   std   buffer+4
634   ldb   #3
635   stb   state
636   rts
637  ENDSUB
638
639
640  ******************
641  SUBROUTINE _reg_a
642  ******************
643   clra
644   ldb   USER_A
645   jsr   _hex4
646   ldd   #$3f00
647   std   buffer
648   clra
649   std   buffer+4
650   rts
651  ENDSUB
652
653
654  ******************
655  SUBROUTINE _reg_b
656  ******************
657   clra
658   ldb   USER_B
659   jsr   _hex4
660   ldd   #$a700
661   std   buffer
662   clra
663   std   buffer+4
664   rts
665  ENDSUB
666
667
668  ******************
669  SUBROUTINE _reg_ab
670  ******************
671   ldd   USER_A
672   jsr   _hex4
```

```
673   ldd   #$a73f
674   std   buffer
675   rts
676  ENDSUB
677
678
679  *****************
680  SUBROUTINE _reg_x
681  *****************
682   ldd   USER_X
683   jsr   _hex4
684   ldd   #$1300
685   std   buffer
686   rts
687  ENDSUB
688
689
690  *****************
691  SUBROUTINE _reg_y
692  *****************
693   ldd   USER_Y
694   jsr   _hex4
695   ldd   #$b600
696   std   buffer
697   rts
698  ENDSUB
699
700
701  *****************
702  SUBROUTINE _reg_u
703  *****************
704   ldd   USER_U
705   jsr   _hex4
706   ldd   #$b500
707   std   buffer
708   rts
709  ENDSUB
710
711
712  *****************
713  SUBROUTINE _reg_s
714  *****************
715   ldd   SAVE_SP
716   jsr   _hex4
717   ldd   #$ae00
718   std   buffer
719   rts
720  ENDSUB
721
722
723  ******************
724  SUBROUTINE _reg_dp
725  ******************
726   ldd   #$1fb3     ; buffer[0] = 0x1F;
727   std   buffer     ; buffer[1] = 0xb3;
728   clra             ; buffer[4] = 0;
```

```
729   clrb              ; buffer[5] = 0;
730   std   buffer+4
731   jsr   _byte_seg
732   std   buffer+2
733   rts
734  ENDSUB
735
736
737  ******************
738  SUBROUTINE _low_cc
739  ******************
740   ldb   USER_CC
741   ldx   #2
742  .loop
743   lda   #$bd        ; 0
744   lsrb
745   bcc   .zero
746   lda   #$30        ; 1
747  .zero
748   sta   buffer,x
749   leax  1,x
750   cmpx  #6
751   bne   .loop
752   ldd   #$858d      ; buffer[0] = 0x85;
753   std   buffer      ; buffer[1] = 0x8d;
754   rts
755  ENDSUB
756
757
758  *****************
759  SUBROUTINE _hi_cc
760  *****************
761   ldb   USER_CC
762   lsrb
763   lsrb
764   lsrb
765   lsrb
766   ldx   #2
767  .loop
768   lda   #$bd        ; 0
769   lsrb
770   bcc   .zero
771   lda   #$30        ; 1
772  .zero
773   sta   buffer,x
774   leax  1,x
775   cmpx  #6
776   bne   .loop
777   ldd   #$378d      ; buffer[0] = 0x37;
778   std   buffer      ; buffer[1] = 0x8d;
779   rts
780  ENDSUB
781
782
783  *********
784  reg_table
```

```
*********
 word  _reg_a  ; 0
 word  _reg_b  ; 1
 word  _reg_ab ; 2
 word  _reg_ab ; 3
 word  _hi_cc  ; 4
 word  _low_cc ; 5
 word  _reg_x  ; 6
 word  _reg_y  ; 7
 word  _reg_dp ; 8
 word  _reg_u  ; 9
 word  _reg_s  ; a

***********************
SUBROUTINE _reg_display
***********************
 ldx   #reg_table
 ldb   key
 cmpb  #10
 bhi   .return
 abx
 abx
 jmp  [,x]
.return
ENDSUB


******************
SUBROUTINE _insert
******************
 ldb   state      ; if (state == 1 || state == 2)
 beq   .return
 cmpb  #2
 bhi   .return
 ldx   CPC        ; for (j=512; j>0; j--)
 leax  512,x
.loop
 ldd   ,--x       ; PC[j] = PC[j-1];
 std   1,x
 cmpx  CPC
 bhi   .loop
 clr   ,x         ; PC[0] = 0;
 jsr   _read_memory
 ldb   #2         ; state = 2;
 stb   state
.return
 rts
ENDSUB


********************
SUBROUTINE _cut_byte
********************
 ldb   state      ; if (state == 1 || state == 2)
 beq   .return
 cmpb  #2
```

```
 bhi   .return
 ldx   CPC         ; for (j=0; j<512; j++)
 ldy   #256
.loop
 ldd   1,x         ; PC[j] = PC[j+1];
 std   ,x++
 leay  -1,y
 bne   .loop
 jsr   _read_memory
 ldb   #2          ; state = 2;
 stb   state
.return
 rts
ENDSUB


********************
SUBROUTINE _key_test
********************
 andcc #$ef        ; enable IRQ
 ldy   #0
 sty   buffer      ; buffer[0] = buffer[1] = 0;
.while
 jsr   _scan
 ldb   tick        ; while (tick < 10)
 cmpb  #10
 blo   .while
 clr   tick        ; tick = 0;
 tfr   y,d
 stb   GPIO1
 jsr   _hex4       ; _hex4(t);
 leay  1,y
 bra   .while
ENDSUB


************************
SUBROUTINE _clear_buffer
************************
 clra
 clrb
 std   buffer
 std   buffer+2
 std   buffer+4
 rts
ENDSUB


********************
SUBROUTINE _key_cal
********************
 ldb   #4             ; state = 4;
 stb   state
 jsr   _clear_buffer
 ldb   #$BD           ; buffer[2] = 0xbd;
 stb   buffer+2
```

```
 clra
 clrb
 std  start          ; start = 0;
 stb  hit            ; hit   = 0;
 rts
ENDSUB


*********************
SUBROUTINE _enter_num
*********************
 clra
 clrb
 tst   hit         ; if (hit == 0) num = 0;
 bne   .enua
 std   num
.enua
 incb              ; hit = 1;
 stb   hit
 ldd   num         ; num <<= 4;
 lslb
 rola
 lslb
 rola
 lslb
 rola
 lslb
 rola
 orb   key         ; num |= key;
 std   num
 jsr   _hex4
 rts
ENDSUB


********************
SUBROUTINE _key_copy
********************
 ldb   #5            ; state = 5;
 stb   state
 clr   hit           ; hit = 0;
 jsr   _clear_buffer
 ldb   #$AE          ; buffer[0] = 0xae;
 stb   buffer
 ldb   #$BD          ; buffer[2] = 0xbd;
 stb   buffer+2
 rts
ENDSUB


*******
key_tab
*******
 word  _key_PC       ; 10
 word  _key_reg      ; 11
 word  _key_data     ; 12
```

```
 953   word  _key_address ; 13
 954   word  _key_ignore  ; 14
 955   word  _key_flag    ; 15
 956   word  _key_minus   ; 16
 957   word  _key_plus    ; 17
 958   word  _insert      ; 18
 959   word  _cut_byte    ; 19
 960   word  _key_test    ; 1a
 961   word  _key_go      ; 1b
 962   word  _key_copy    ; 1c
 963   word  _key_cal     ; 1d
 964   word  _key_dump    ; 1e
 965   word  _key_load    ; 1f
 966
 967  **********************
 968  SUBROUTINE _key_ignore
 969  **********************
 970   rts
 971  ENDSUB
 972
 973
 974  ********************
 975  SUBROUTINE _key_flag
 976  ********************
 977   ldb   flag
 978   eorb  #1
 979   stb   flag
 980   rts
 981  ENDSUB
 982
 983
 984  ******************
 985  SUBROUTINE _key_exe
 986  ******************
 987   tst   flag
 988   bne   .laba
 989   jsr   _beep
 990  .laba
 991   ldb   key
 992   cmpb  #16
 993   blo   .labb
 994   ldx   #key_tab
 995   subb  #16
 996   andb  #15
 997   abx
 998   abx
 999   jmp   [,x]
1000  .labb
1001   ldb   state
1002   decb
1003   lbeq  _hex_address
1004   decb
1005   lbeq  _data_hex
1006   decb
1007   lbeq  _reg_display
1008   jmp   _enter_num
```

```
1009  ENDSUB
1010
1011
1012  ****************
1013  SUBROUTINE _beep
1014  ****************
1015   clr   PORT2
1016   ldx   #60
1017  .loop
1018   lda   #$7f
1019   sta   PORT1
1020   ldb   #50
1021  .dela
1022   decb
1023   bne   .dela
1024   lda   #$ff
1025   sta   PORT1
1026   ldb   #50
1027  .delb
1028   decb
1029   bne   .delb
1030   leax  -1,x
1031   bne   .loop
1032   rts
1033  ENDSUB
1034
1035
1036  *****************
1037  SUBROUTINE _scan1
1038  *****************
1039  .whilea           ; while( _scan() != -1);
1040   jsr   _scan
1041   bpl   .whilea
1042   ldy   #30        ; delay(30);
1043   jsr   _delay
1044  .whileb           ; while( _scan() == -1);
1045   jsr   _scan
1046   bmi   .whileb
1047   ldy   #30        ; delay(30);
1048   jsr   _delay
1049   jsr   _scan      ; key = scan();
1050   ldx   #key_code  ; key = key_code(key);
1051   ldb   b,x
1052   stb   key
1053   jmp   _key_exe
1054  ENDSUB
1055
1056  key_code
1057   BYTE  $18,$19,$1a,$0c, $00,$00,$14,$15,
1058   BYTE  $16,$17,$00,$00, $10,$11,$12,$13,
1059   BYTE  $1f,$00,$0f,$0b, $07,$03,$00,$1e,
1060   BYTE  $0e,$0a,$06,$02, $04,$1d,$0d,$09,
1061   BYTE  $05,$01,$08,$1c, $1b
1062
1063  ********************
1064  SUBROUTINE _initacia
```

```
1065  ********************
1066   ldb   #3          ; reset
1067   stb   ACIAPORT
1068   ldb   #$16        ; baudrate = 19200
1069   stb   ACIAPORT
1070   ldb   ACIAPORT+1 ; clear RBR
1071   rts
1072  ENDSUB
1073
1074
1075  ********************
1076  SUBROUTINE _putchar
1077  ********************
1078  * Input: b = char *
1079  ********************
1080   lda   #2          ; wait on TDRE
1081  .wait
1082   bita  ACIAPORT
1083   beq   .wait
1084   stb   ACIAPORT+1
1085   rts
1086  ENDSUB
1087
1088
1089  ****************
1090  SUBROUTINE _puts
1091  ********************
1092  * Input: x = char *
1093  ********************
1094  .loop
1095   ldb   ,x+
1096   beq   .return
1097   jsr   _putchar
1098   bra   .loop
1099  .return
1100   rts
1101  ENDSUB
1102
1103
1104  ********************
1105  SUBROUTINE _newline
1106  ********************
1107   ldb   #13
1108   jsr   _putchar
1109   ldb   #10
1110   jmp   _putchar
1111  ENDSUB
1112
1113
1114  ********************
1115  SUBROUTINE _send_hex
1116  ********************
1117  * Input: b = byte  *
1118  ********************
1119   pshs  b
1120   lsrb
```

```
1121    lsrb
1122    lsrb
1123    lsrb
1124    addb  #'0'
1125    cmpb  #'9'
1126    bls   .hexh
1127    addb  #7
1128   .hexh
1129    jsr _putchar
1130    puls  b
1131    andb  #15
1132    addb  #'0'
1133    cmpb  #'9'
1134    bls   .hexl
1135    addb  #7
1136   .hexl
1137    jmp _putchar
1138   ENDSUB
1139
1140
1141   ************************
1142   SUBROUTINE _send_word_hex
1143   ************************
1144    ldb   2,s
1145    jsr   _send_hex
1146    ldb   3,s
1147    bra   _send_hex
1148   ENDSUB
1149
1150
1151   ********************
1152   SUBROUTINE _key_dump
1153   ********************
1154    leas  -2,s
1155    ldx   CPC
1156    ldb   #16
1157    stb   ,s          ; for (j=0; j<16; j++)
1158
1159   .loopo
1160    jsr   _newline
1161    pshs  x
1162    jsr   _send_word_hex
1163    leas  2,s
1164    ldb   #':'        ; putchar(':');
1165    jsr   _putchar
1166    ldb   #16
1167    stb   1,s         ; for (p=0; p<16; p++)
1168
1169   .loopi
1170    ldb   ,x+         ; send_hex();
1171    jsr   _send_hex
1172    ldb   #' '        ; putchar(' ');
1173    jsr   _putchar
1174    dec   1,s
1175    bne   .loopi
1176
```

```
1177    ldb   #' '       ; putchar(' ');
1178    jsr   _putchar
1179    leax  -16,x
1180    ldb   #16
1181    stb   1,s         ; for (p=0; p<16; p++)
1182
1183   .loopj
1184    ldb   ,x+         ; putchar();
1185    cmpb  #' '
1186    blo   .dot        ; if (q >= ' ' && q < 0x7F) putchar(q);
1187    cmpb  #$7f
1188    blo   .put
1189   .dot
1190    ldb   #'.'
1191   .put
1192    jsr   _putchar
1193    dec   1,s
1194    bne   .loopj
1195
1196    dec   ,s
1197    bne   .loopo
1198    stx   CPC
1199    jsr   _newline     ; newline();
1200    jsr   _key_address ; key_address();
1201    leas  2,s
1202    rts
1203   ENDSUB
1204
1205
1206   *****
1207   hihex
1208   *****
1209    BYTE $00,$10,$20,$30,$40,$50,$60,$70,$80,$90
1210    BYTE $00,$00,$00,$00,$00,$00,$00
1211    BYTE $a0,$b0,$c0,$d0,$e0,$f0
1212
1213   *****
1214   lohex
1215   *****
1216    BYTE $00,$01,$02,$03,$04,$05,$06,$07,$08,$09
1217    BYTE $00,$00,$00,$00,$00,$00,$00
1218    BYTE $0a,$0b,$0c,$0d,$0e,$0f
1219
1220   ******************
1221   SUBROUTINE _gethex
1222   ******************
1223    jsr   _getchar    ; a = getchar();
1224    ldx   #hihex-$30
1225    ldb   b,x
1226    pshs  b
1227    jsr   _getchar    ; b = getchar();
1228    ldx   #lohex-$30
1229    ldb   b,x
1230    orb   ,s+         ; a = a | b;
1231    pshs  b
1232    addb  checksum    ; checksum += a
```

```
1233    stb   checksum
1234    puls  b,pc
1235   ENDSUB
1236
1237
1238   ********************
1239   SUBROUTINE get16bit
1240   ********************
1241    bsr   _gethex
1242    pshs  b
1243    bsr   _gethex
1244    puls  a,pc
1245   ENDSUB
1246
1247
1248   ************************
1249   SUBROUTINE _read_record1
1250   ************************
1251   * Output: b = checkerr *
1252   ************************
1253    clr   checksum   ; checksum = 0;
1254    bsr   _gethex    ; byte_count = gethex()-3;
1255    subb  #3
1256    pshs  b
1257    bsr   get16bit   ; address16bit = get16bitaddress();
1258    stb   GPIO1
1259    tfr   d,y        ; address16bit;
1260    puls  a          ; byte count
1261   .loop
1262    bsr   _gethex    ; gethex();
1263    stb   ,y+
1264    deca
1265    bne   .loop
1266    ldb   checksum   ; checksum = ~checksum;
1267    comb
1268    pshs  b
1269    bsr   _gethex
1270    subb  ,s+        ; checkerr
1271    rts
1272   ENDSUB
1273
1274
1275   ************************
1276   SUBROUTINE _get_s_record
1277   ************************
1278   .while           ; while (getchar() != 'S');
1279    jsr   _getchar
1280    cmpb  #'S'
1281    bne   .while
1282    jsr   _getchar
1283    cmpb  #'0'       ; case '0': continue
1284    beq   .while
1285    cmpb  #'1'       ; case '1': read_record1();
1286    bne   .exit
1287    jsr   _read_record1
1288    tstb             ; check error
```

```
1289    beq   .while
1290    jsr   _send_hex  ; check sum difference
1291    ldb   #':'
1292    jsr   _putchar
1293    tfr   y,d        ; last used address
1294    jsr   _send_word_hex
1295    ldx   #.MsgErr   ; puts("check sum errors!");
1296    bra   .geta
1297   .exit
1298    ldx   #.MsgOK     ; else puts("OK");
1299   .geta
1300    jsr   _puts
1301    jmp   _key_data  ; key_data();
1302   .MsgErr BYTE ":check sum error!",0
1303   .MsgOK  BYTE "OK",0
1304   ENDSUB
1305
1306
1307   ********************
1308   SUBROUTINE _key_load
1309   ********************
1310
1311    ldx   #.msg
1312    jsr   _puts
1313    jmp   _get_s_record
1314   .msg  BYTE "\r\nLoad Motorola s-record\r\n",0
1315   ENDSUB
1316
1317
1318   ********************
1319   SUBROUTINE _initreg
1320   ********************
1321
1322    ldd  #$0200
1323    std  CPC         ; PC      = 0x0200;
1324    std  SAVE_PC     ; SAVE_PC = 0x0200;
1325    ldd  #$7F00
1326    std  SAVE_SP     ; SAVE_SP = 0x7F00;
1327    std  USER_U      ; USER_U  = 0x7F00;
1328    clr  USER_DP     ; USER_DP = 0;
1329    tfr  CC,A
1330    sta  USER_CC
1331    rts
1332   ENDSUB
1333
1334
1335   ********************
1336   SUBROUTINE _getchar
1337   ********************
1338
1339    ldb   #$16       ; enable receiving
1340    stb   ACIAPORT
1341    ldb   #1
1342   .while            ; while((*acia&1) == 0)
1343    bitb  ACIAPORT
1344    beq   .while
```

```
1345    ldb   #$56       ; stop sending
1346    stb   ACIAPORT
1347    ldb   ACIAPORT+1 ; ch = *(acia+1);
1348    rts              ; return ch;
1349   ENDSUB
1350
1351
1352   ****************
1353   SUBROUTINE _main
1354   ****************
1355
1356    bsr   _initreg   ; initreg();
1357    clr   GPIO1
1358    clr   PORT2
1359    ldb   #$ff
1360    stb   PORT1
1361    clr   flag       ; flag = 0;
1362    jsr   _initacia  ; initacia();
1363    jsr   _newline   ; newline();
1364    ldx   #MSG02
1365    jsr   _puts      ; puts("6809 MICROPROCESSOR KIT 2020");
1366    jsr   _LCD_Init  ; _LCD_Init();
1367    ldx   #MSG03
1368    jsr   _LCD_Puts  ; LCD_Puts("6809 MICROPROCESSOR");
1369    ldd   #$0100
1370    jsr   _LCD_Row_Col ; LCD_Row_Col(1,0);
1371    ldx   #MSG04
1372    jsr   _LCD_Puts  ; LCD_Puts("32kB RAM UART LCD");
1373    jsr   _clear_buffer
1374    ldb   #175
1375    stb   buffer+5   ; buffer[5] = convert[6];
1376    ldb   #191
1377    stb   buffer+4   ; buffer[4] = convert[8];
1378    ldb   #189
1379    stb   buffer+3   ; buffer[3] = convert[0];
1380    ldb   #190
1381    stb   buffer+2   ; buffer[2] = convert[9];
1382   .loop
1383    jsr   _scan1
1384    bra   .loop
1385   ENDSUB
1386
1387   MSG02 BYTE "6809 MICROPROCESSOR KIT 2020\r\n",0
1388   MSG03 BYTE "6809 CPU",0
1389   MSG04 BYTE "32kB RAM LCD",0
1390
1391   END
1392
```

```
                        ********************************************************
                        *  version 1.0   01-May-2020
                        ********************************************************
                        *  monitor source code for 6809 MICROPROCRSSOR KIT 2020
                        *  assemble with bs9 assembler
                        *  monitor source code was written by Wichit Sirichote,
                        *  wichit.sirichote@gmail.com
                        *
                        *  The object file is EPROM/EEPROM image
                        *  more update and technical information:
                        *   kswichit.com/6809/6809.htm
                        ********************************************************

6809                    CPU = 6809
                        STORE $8000,$8000,"newmon.rom"
                        STORE $8000,$8000,"newmon.s19",s19

8000                    GPIO1     = $8000
8001                    PORT0     = $8001
8002                    PORT1     = $8002
8003                    PORT2     = $8003

9000                    LCD_cwr   = $9000
9001                    LCD_dwr   = $9001
9002                    LCD_crd   = $9002
9003                    LCD_drd   = $9003

a000                    ACIAPORT  = $A000

       00                       SETDP 0
0000                            ORG $0000

                        ***********************************
                        * area for saving interrupt stack *
                        ***********************************

0000                    USER_START
0000                    USER_CC   RMB 1  ;  0
0001                    USER_A    RMB 1  ;  1
0002                    USER_B    RMB 1  ;  2
0003                    USER_DP   RMB 1  ;  3
0004                    USER_X    RMB 2  ;  4
0006                    USER_Y    RMB 2  ;  6
0008                    USER_U    RMB 2  ;  8
000a                    USER_PC   RMB 2  ;  a
000c                    USER_END
0001                    USER_D    = USER_A

000c                    SAVE_SP   RMB 2
000e                    SAVE_PC   RMB 2
0010                    CPC       RMB 2  ; current PC
0012                    SIGN      RMB 1  ; +/- flag


0013                    key:      RMB 1
0014                    hit:      RMB 1
```

```
  57  0015                flag:     RMB 1
  58  0016                tick:     RMB 1
  59  0017                checksum: RMB 1
  60  0018                num:      RMB 2
  61  001a                start:    RMB 2
  62  001c                _end:     RMB 2
  63  001e                state:    RMB 1
  64  001f                buffer:   RMB 6
  65
  66  7ff0                _virq  = $7ff0
  67  7ff3                _vnmi  = $7ff3
  68  7ff6                _vfirq = $7ff6
  69
  70
  71                      *  ===========
  72  fff0                   ORG   $fff0
  73                      *  ===========
  74
  75  fff0 c000              FDB   _reset    ; fff0 : trap on 6309
  76  fff2 c000              FDB   _reset    ; fff2 : SWI3
  77  fff4 c000              FDB   _reset    ; fff4 : SWI2
  78  fff6 7ff6              FDB   _vfirq    ; fff6 : FIRQ
  79  fff8 7ff0              FDB   _virq     ; fff8 : IRQ
  80  fffa c026              FDB   _swi_serv ; fffa : SWI
  81  fffc 7ff3              FDB   _vnmi     ; fffc : NMI
  82  fffe c000              FDB   _reset    ; fffe : RESET
  83
  84                      *  ===========
  85  c000                   ORG   $c000
  86                      *  ===========
  87
  88                      *****************
  89  c000                SUBROUTINE _reset
  90                      *****************
  91  c000 10ce    7ff0      lds   #$7ff0      ; init stack pointer
  92  c004   86      7e      lda   #$7e        ; store jmp instruction
  93  c006   8e    c023      ldx   #_irq_serv
  94  c009   b7    7ff0      sta   _virq       ; JMP
  95  c00c   bf    7ff1      stx   _virq+1     ; irq_serv
  96  c00f   86      3b      lda   #$3b        ; store rti instruction
  97  c011   8e    0000      ldx   #0
  98  c014   b7    7ff3      sta   _vnmi       ; must be modified when testing
  99  c017   b7    7ff6      sta   _vfirq      ; with ram vector
 100  c01a   bf    7ff4      stx   _vnmi+1
 101  c01d   bf    7ff7      stx   _vfirq+1
 102  c020   7e    c691      jmp   _main
 103  c023                ENDSUB ;   35 [_reset]
 104
 105
 106                      ********************
 107  c023                SUBROUTINE _irq_serv
 108                      ********************
 109  c023   0c      16      inc tick
 110  c025   3b              rti
 111  c026                ENDSUB ;    3 [_irq_serv]
 112
```

```
                    **********
                    * Stack   *
                    *---------*
                    * a: PC   *
                    * 9:      *
                    * 8: U    *
                    * 7:      *
                    * 6: Y    *
                    * 5:      *
                    * 4: X    *
                    * 3: DP   *
                    * 2: B    *
                    * 1: A    *
                    * 0: CC   *
                    **********


                    ********************
c026                SUBROUTINE _swi_serv
                    ********************
c026   8e    0000   ldx   #USER_START
c029                .loop
c029   35 06        puls  d
c02b   ed 81        std   ,x++
c02d   8c    000c   cmpx  #USER_END
c030   26      f7   bne   .loop
c032   dd      0e   std   SAVE_PC
c034 10de      0c   lds   SAVE_SP
c037   7e    c20f   jmp   _key_PC
c03a                ENDSUB ;   20 [_swi_serv]


c03a                convert
c03a bd 30 9b ba    FCB 189,48,155,186,54,174,175,56,191,190
c044 3f a7 8d b3    FCB 63,167,141,179,143,15


                    ********************
c04a                SUBROUTINE _LCD_Ready
                    ********************
c04a   34 24        pshs  b,y
c04c 108e    07d0   ldy   #2000      ; timeout
c050                .loop
c050   f6    9002   ldb   LCD_crd
c053   2a      04   bpl   .return
c055   31 3f        leay  -1,y
c057   26      f7   bne   .loop
c059                .return
c059   35 a4        puls  b,y,pc
c05b                ENDSUB ;   17 [_LCD_Ready]


                    ********************
c05b                SUBROUTINE _LCD_Clear
                    ********************
```

```
c05b   bd    c04a  jsr   _LCD_Ready
c05e   c6      01  ldb   #1
c060   f7    9000  stb   LCD_cwr
c063   39          rts
c064             ENDSUB ;    9 [_LCD_Clear]


c064             LCD_Row_Val
c064 80 c0 94 d4   BYTE  $80,$c0,$94,$d4

                 ***********************
c068             SUBROUTINE _LCD_Row_Col
                 ***********************
                 * Input: a = row  0-3 *
                 *        b = col      *
                 ***********************
c068   bd    c04a  jsr   _LCD_Ready
c06b   8e    c064  ldx   #LCD_Row_Val
c06e   eb 86       addb  a,x
c070   f7    9000  stb   LCD_cwr
c073   39          rts
c074             ENDSUB ;   12 [_LCD_Row_Col]


                 ********************
c074             SUBROUTINE _LCD_Init
                 ********************
c074   bd    c04a  jsr   _LCD_Ready
c077   c6      38  ldb   #$38
c079   f7    9000  stb   LCD_cwr
c07c   bd    c04a  jsr   _LCD_Ready
c07f   c6      0c  ldb   #$0c
c081   f7    9000  stb   LCD_cwr
c084   bd    c05b  jsr   _LCD_Clear
c087   4f          clra
c088   5f          clrb
c089   8d      dd  bsr   _LCD_Row_Col    ; LCD_XY(0,0);
c08b 108e    03e8  ldy   #1000
c08f   7e    c0a6  jmp   _delay
c092             ENDSUB ;   30 [_LCD_Init]


                 ********************
c092             SUBROUTINE _LCD_Puts
                 ********************
                 * Input: x = char* *
                 ********************
c092   bd    c04a  jsr   _LCD_Ready
c095   e6 80       ldb   ,x+
c097   27      05  beq   .return
c099   f7    9001  stb   LCD_dwr
c09c   20      f4  bra   _LCD_Puts
c09e             .return
c09e   39          rts
c09f             ENDSUB ;   13 [_LCD_Puts]

```

```
225
226                     ********************
227  c09f               SUBROUTINE _LCD_Putc
228                     ********************
229                     * Input: b = char  *
230                     ********************
231  c09f   bd    c04a  jsr   _LCD_Ready
232  c0a2   f7    9001  stb   LCD_dwr
233  c0a5   39          rts
234  c0a6               ENDSUB ;    7 [_LCD_Putc]
235
236
237                     *****************
238  c0a6               SUBROUTINE _delay
239                     *****************
240                     * Input: y = time
241                     ***************** ; 6809  6309
242                                       ; ----------
243  c0a6   31 3f        leay  -1,y       ;    4     4
244  c0a8   26      fc   bne   _delay     ;    3     3
245  c0aa   39           rts
246
247                     ***********************
248  c0ab               SUBROUTINE _seg_refresh
249                     ***********************
250  c0ab   86      fe   lda   #%1111 1110 ; select mask
251  c0ad   8e    001f   ldx   #buffer
252  c0b0               .loop
253  c0b0 108e    000a   ldy   #10         ; brightness
254  c0b4   e6 80        ldb   ,x+        ; buffer[i]
255  c0b6   fd    8002   std   PORT1       ; PORT1 = a  PORT2 = b
256  c0b9   c1      30   cmpb  #48         ; 1
257  c0bb   27      06   beq   .sega
258  c0bd   c1      38   cmpb  #56         ; 7
259  c0bf   27      02   beq   .sega
260  c0c1   31 28        leay  8,y        ; increase brightness
261  c0c3               .sega
262  c0c3   8d      e1   bsr   _delay
263  c0c5   5f           clrb
264  c0c6   f7    8003   stb   PORT2       ; PORT2 = 0
265  c0c9   31 2a        leay  10,y
266  c0cb   bd    c0a6   jsr   _delay
267  c0ce   1a      01   orcc  #1          ; set carry
268  c0d0   49           rola              ; rotate mask
269  c0d1   81      bf   cmpa  #%1011 1111
270  c0d3   26      db   bne   .loop
271  c0d5   39           rts
272  c0d6               ENDSUB ;   43 [_seg_refresh]
273
274
275                     ********************
276  c0d6               SUBROUTINE _kbd_scan
277                     ********************
278                     * Output: key      *
279                     ********************
280  c0d6   86      fe   lda   #%1111 1110 ; select mask
```

```
281  c0d8   0f      13  clr   key
282  c0da              .loopo
283  c0da   b7    8002  sta   PORT1
284  c0dd   f6    8001  ldb   PORT0
285  c0e0   8e    0006  ldx   #6          ; index
286  c0e3              .loopi
287  c0e3   54          lsrb
288  c0e4   24      1a  bcc   .return     ; keypress
289  c0e6   0c      13  inc   key
290  c0e8   30 1f       leax  -1,x
291  c0ea   26      f7  bne   .loopi
292  c0ec   1a      01  orcc  #1          ; set carry
293  c0ee   49          rola              ; rotate mask
294  c0ef   81      bf  cmpa  #%1011 1111
295  c0f1   26      e7  bne   .loopo
296  c0f3   c6      ff  ldb   #-1         ; no key
297  c0f5   b6    8001  lda   PORT0
298  c0f8   85      40  bita  #$40
299  c0fa   26      02  bne   .kbda
300  c0fc   c6      24  ldb   #$24        ; GO key
301  c0fe              .kbda
302  c0fe   d7      13  stb   key
303  c100              .return
304  c100   39          rts
305  c101              ENDSUB ;   43 [_kbd_scan]
306
307
308                    ****************
309  c101              SUBROUTINE _scan
310                    ****************
311  c101   34 30       pshs  x,y
312  c103   8d      a6  bsr   _seg_refresh
313  c105   8d      cf  bsr   _kbd_scan
314  c107   d6      13  ldb   key
315  c109   35 b0       puls  x,y,pc
316  c10b              ENDSUB ;   10 [_scan]
317
318
319                    ***********************
320  c10b              SUBROUTINE _dot_address
321                    ***********************
322  c10b   dc      1f  ldd   buffer
323  c10d   84      bf  anda  #$bf
324  c10f   c4      bf  andb  #$bf
325  c111   dd      1f  std   buffer
326  c113   dc      21  ldd   buffer+2
327  c115   8a      40  ora   #$40
328  c117   ca      40  orb   #$40
329  c119   dd      21  std   buffer+2
330  c11b   dc      23  ldd   buffer+4
331  c11d   8a      40  ora   #$40
332  c11f   ca      40  orb   #$40
333  c121   dd      23  std   buffer+4
334  c123   39          rts
335  c124              ENDSUB ;   25 [_dot_address]
336
```

```
                        ********************
c124                    SUBROUTINE _dot_data
                        ********************
c124   dc      1f       ldd   buffer
c126   8a      40       ora   #$40
c128   ca      40       orb   #$40
c12a   dd      1f       std   buffer
c12c   dc      21       ldd   buffer+2
c12e   84      bf       anda  #$bf
c130   c4      bf       andb  #$bf
c132   dd      21       std   buffer+2
c134   dc      23       ldd   buffer+4
c136   84      bf       anda  #$bf
c138   c4      bf       andb  #$bf
c13a   dd      23       std   buffer+4
c13c   39                rts
c13d                    ENDSUB ;   25 [_dot_data]


                        ********************
c13d                    SUBROUTINE _byte_seg
                        ********************
                        * Input : b = byte *
                        * Output: d = seg  *
                        ********************
c13d   8e    c03a       ldx   #convert
c140   1f 98             tfr   b,a
c142   84      0f       anda  #15
c144   a6 86             lda   a,x          ; low  nibble
c146   54                lsrb
c147   54                lsrb
c148   54                lsrb
c149   54                lsrb
c14a   e6 85             ldb   b,x          ; high nibble
c14c   39                rts
c14d                    ENDSUB ;   16 [_byte_seg]


                        ****************
c14d                    SUBROUTINE _hex4
                        ****************
                        * Input: d = word
                        ******************
c14d   34 02             pshs  a           ; save high byte
c14f   8d      ec        bsr   _byte_seg
c151   dd      21       std   buffer+2
c153   35 04             puls  b           ; pull high byte
c155   8d      e6        bsr   _byte_seg
c157   dd      23       std   buffer+4
c159   39                rts
c15a                    ENDSUB ;   13 [_hex4]


                        **********************
c15a                    SUBROUTINE _read_memory
```

```
                        **********************
c15a   dc      10  ldd   CPC
c15c   bd    c14d  jsr   _hex4
c15f   e6 9f 0010  ldb   [CPC]
c163   8d      d8  bsr   _byte_seg
c165   dd      1f  std   buffer
c167   7e    c124  jmp   _dot_data
c16a               ENDSUB ;   16 [_read_memory]


                        **********************
c16a               SUBROUTINE _key_address
                        **********************
c16a   0f      14  clr   hit
c16c   c6      01  ldb   #1
c16e   d7      1e  stb   state
c170   bd    c15a  jsr   _read_memory
c173   7e    c10b  jmp   _dot_address
c176               ENDSUB ;   12 [_key_address]


                        ********************
c176               SUBROUTINE _key_data
                        ********************
c176   0f      14  clr   hit
c178   c6      02  ldb   #2
c17a   d7      1e  stb   state
c17c   bd    c15a  jsr   _read_memory
c17f   7e    c124  jmp   _dot_data
c182               ENDSUB ;   12 [_key_data]


                        ********************
c182               SUBROUTINE _key_plus
                        ********************
c182   d6      1e  ldb   state      ; if (state == 1 || state == 2)
c184   27      11  beq   .plusa
c186   c1      02  cmpb  #2
c188   22      0d  bhi   .plusa
c18a   dc      10  ldd   CPC        ; PC++;
c18c   c3    0001  addd  #1
c18f   dd      10  std   CPC
c191   bd    c15a  jsr   _read_memory
c194   7e    c176  jmp   _key_data
c197               .plusa
c197   c1      04  cmpb  #4         ; if (state == 4)
c199   26      09  bne   .plusb
c19b   dc      18  ldd   num        ; start = num;
c19d   dd      1a  std   start
c19f   0f      14  clr   hit        ; hit = 0;
c1a1   0f      12  clr   SIGN       ; positive
c1a3   39          rts
c1a4               .plusb
c1a4   c1      05  cmpb  #5
c1a6   26      0d  bne   .plusc
c1a8   0c      1e  inc   state      ; state = 6;
```

```
c1aa   dc      18  ldd   num        ; start = num;
c1ac   dd      1a  std   start
c1ae   0f      14  clr   hit        ; hit = 0;
c1b0   c6      8f  ldb   #$8f       ; buffer[0]=0x8f;
c1b2   d7      1f  stb   buffer
c1b4   39          rts
c1b5             .plusc
c1b5   c1      06  cmpb  #6
c1b7   26      14  bne   .return
c1b9   0c      1e  inc   state      ; state = 7;
c1bb   0f      14  clr   hit        ; hit = 0;
c1bd   c6      b3  ldb   #$b3       ; buffer[0] = 0xb3;
c1bf   d7      1f  stb   buffer
c1c1   dc      18  ldd   num        ; end = num;
c1c3   dd      1c  std    _end
c1c5 1093      1a  cmpd  start      ; if (end <= start) print_error();
c1c8   22      03  bhi   .return
c1ca   bd    c235  jsr _print_error
c1cd             .return
c1cd   39          rts
c1ce             ENDSUB ;   76 [_key_plus]


                 ********************
c1ce             SUBROUTINE _key_minus
                 ********************
c1ce   d6      1e  ldb   state      ; if (state == 1 || state == 2)
c1d0   27      11  beq   .mina
c1d2   c1      02  cmpb  #2
c1d4   22      0d  bhi   .mina
c1d6   dc      10  ldd   CPC        ; PC--
c1d8   83    0001  subd  #1
c1db   dd      10  std   CPC
c1dd   bd    c15a  jsr   _read_memory
c1e0   7e    c176  jmp   _key_data
c1e3             .mina
c1e3   c1      04  cmpb  #4         ; if (state == 4)
c1e5   26      0a  bne   .return
c1e7   dc      18  ldd   num        ; start = num;
c1e9   dd      1a  std   start
c1eb   0f      14  clr   hit        ; hit = 0;
c1ed   c6      ff  ldb   #-1
c1ef   d7      12  stb   SIGN       ; negative
c1f1             .return
c1f1   39          rts
c1f2             ENDSUB ;   36 [_key_minus]


                 ********************
c1f2             SUBROUTINE _data_hex
                 ********************
c1f2   5f          clrb
c1f3   0d      14  tst   hit        ; if (hit == 0) x = 0;
c1f5   27      08  beq   .hexa
c1f7   e6 9f 0010  ldb   [CPC]      ; x = *PC;
c1fb   58          lslb             ; x <<= 4;
```

```
c1fc   58          lslb
c1fd   58          lslb
c1fe   58          lslb
c1ff             .hexa
c1ff   da      13  orb   key         ; x |= key;
c201   e7 9f 0010  stb   [CPC]       ; *PC = x;
c205   c6      01  ldb   #1
c207   d7      14  stb   hit
c209   bd    c15a  jsr   _read_memory
c20c   7e    c124  jmp   _dot_data
c20f             ENDSUB ;   29 [_data_hex]


                  ******************
c20f              SUBROUTINE _key_PC
                  ******************
c20f   dc      0e  ldd   SAVE_PC
c211   dd      10  std   CPC
c213   7e    c176  jmp   _key_data
c216             ENDSUB ;    7 [_key_PC]


                  ***********************
c216              SUBROUTINE _hex_address
                  ***********************
c216   4f          clra
c217   5f          clrb
c218   0d      14  tst   hit         ; if (hit == 0) PC = 0;
c21a   26      02  bne   .hexa
c21c   dd      10  std   CPC
c21e             .hexa
c21e   5c          incb              ; hit = 1;
c21f   d7      14  stb   hit
c221   dc      10  ldd   CPC         ; PC <<= 4;
c223   58          lslb
c224   49          rola
c225   58          lslb
c226   49          rola
c227   58          lslb
c228   49          rola
c229   58          lslb
c22a   49          rola
c22b   da      13  orb   key         ; PC |= key;
c22d   dd      10  std   CPC
c22f   bd    c15a  jsr   _read_memory
c232   7e    c10b  jmp   _dot_address
c235             ENDSUB ;   31 [_hex_address]


                  ***********************
c235              SUBROUTINE _print_error
                  ***********************
c235   4f          clra
c236   5f          clrb
c237   dd      1f  std   buffer
c239   c6      03  ldb   #3          ; r
```

```
561  c23b   dd      21  std   buffer+2
562  c23d   cc    038f  ldd   #$038f     ; r e
563  c240   dd      23  std   buffer+4
564  c242   0f      1e  clr   state
565  c244   39          rts
566
567
568                     ******************
569  c245               SUBROUTINE _key_go
570                     ******************
571  c245   d6      1e  ldb   state      ; if (state == 1 || state == 2)
572  c247   27      20  beq   .goa
573  c249   c1      02  cmpb  #2
574  c24b   22      1c  bhi   .goa
575  c24d 10df      0c  sts   SAVE_SP
576  c250   9e      04  ldx   USER_X
577  c252 109e      06  ldy   USER_Y
578  c255   de      08  ldu   USER_U
579  c257   dc      10  ldd   CPC
580  c259   34 06       pshs  d
581  c25b   dc      01  ldd   USER_A
582  c25d   34 06       pshs  d
583  c25f   96      00  lda   USER_CC
584  c261   1f 8a       tfr   a,cc
585  c263   96      03  lda   USER_DP
586  c265   1f 8b       tfr   a,dp
587  c267   35 86       puls  d,pc
588
589  c269               .goa
590  c269   c1      04  cmpb  #4          ; if (state == 4)
591  c26b   26      14  bne   .goc
592  c26d   dc      1a  ldd   start
593  c26f   0d      12  tst   SIGN       ; if (SIGN) start -= num;
594  c271   2a      04  bpl   .pos
595  c273   93      18  subd  num
596  c275   20      02  bra   .gob
597  c277               .pos
598  c277   d3      18  addd  num         ; else start += num;
599  c279               .gob
600  c279   dd      1a  std   start
601  c27b   bd    c14d  jsr   _hex4        ; _hex4(start)
602  c27e   0f      14  clr   hit
603  c280   39          rts
604  c281               .goc
605  c281   c1      07  cmpb  #7          ; if (state == 7)
606  c283   26      1c  bne   .return
607  c285   9e      18  ldx   num
608  c287   9f      10  stx   CPC         ; PC = num
609  c289 109e      1a  ldy   start
610  c28c               .loop             ; for (i=0; i<temp; i++)
611  c28c 109c      1c  cmpy  _end
612  c28f   24      06  bhs   .exit
613  c291   e6 a0       ldb   ,y+         ; num[i] = start[i]
614  c293   e7 80       stb   ,x+
615  c295   20      f5  bra   .loop
616  c297               .exit
```

```
c297   bd    c15a  jsr   _read_memory
c29a   bd    c124  jsr   _dot_data
c29d   c6      02  ldb   #2         ; state = 2;
c29f   d7      1e  stb   state
c2a1               .return
c2a1   39           rts
c2a2               ENDSUB ;   93 [_key_go]


                   ******************
c2a2               SUBROUTINE _key_reg
                   ******************
c2a2   bd    c3d4  jsr   _clear_buffer
c2a5   c6      ad  ldb   #$ad
c2a7   d7      22  stb   buffer+3
c2a9   cc    8f03  ldd   #$8f03
c2ac   dd      23  std   buffer+4
c2ae   c6      03  ldb   #3
c2b0   d7      1e  stb   state
c2b2   39           rts
c2b3               ENDSUB ;   17 [_key_reg]


                   ******************
c2b3               SUBROUTINE _reg_a
                   ******************
c2b3   4f           clra
c2b4   d6      01  ldb   USER_A
c2b6   bd    c14d  jsr   _hex4
c2b9   cc    3f00  ldd   #$3f00
c2bc   dd      1f  std   buffer
c2be   4f           clra
c2bf   dd      23  std   buffer+4
c2c1   39           rts
c2c2               ENDSUB ;   15 [_reg_a]


                   ******************
c2c2               SUBROUTINE _reg_b
                   ******************
c2c2   4f           clra
c2c3   d6      02  ldb   USER_B
c2c5   bd    c14d  jsr   _hex4
c2c8   cc    a700  ldd   #$a700
c2cb   dd      1f  std   buffer
c2cd   4f           clra
c2ce   dd      23  std   buffer+4
c2d0   39           rts
c2d1               ENDSUB ;   15 [_reg_b]


                   ******************
c2d1               SUBROUTINE _reg_ab
                   ******************
c2d1   dc      01  ldd   USER_A
c2d3   bd    c14d  jsr   _hex4
```

```
673  c2d6   cc    a73f  ldd   #$a73f
674  c2d9   dd      1f  std   buffer
675  c2db   39         rts
676  c2dc             ENDSUB ;   11 [_reg_ab]
677
678
679                   *****************
680  c2dc             SUBROUTINE _reg_x
681                   *****************
682  c2dc   dc      04  ldd   USER_X
683  c2de   bd    c14d  jsr   _hex4
684  c2e1   cc    1300  ldd   #$1300
685  c2e4   dd      1f  std   buffer
686  c2e6   39         rts
687  c2e7             ENDSUB ;   11 [_reg_x]
688
689
690                   *****************
691  c2e7             SUBROUTINE _reg_y
692                   *****************
693  c2e7   dc      06  ldd   USER_Y
694  c2e9   bd    c14d  jsr   _hex4
695  c2ec   cc    b600  ldd   #$b600
696  c2ef   dd      1f  std   buffer
697  c2f1   39         rts
698  c2f2             ENDSUB ;   11 [_reg_y]
699
700
701                   *****************
702  c2f2             SUBROUTINE _reg_u
703                   *****************
704  c2f2   dc      08  ldd   USER_U
705  c2f4   bd    c14d  jsr   _hex4
706  c2f7   cc    b500  ldd   #$b500
707  c2fa   dd      1f  std   buffer
708  c2fc   39         rts
709  c2fd             ENDSUB ;   11 [_reg_u]
710
711
712                   *****************
713  c2fd             SUBROUTINE _reg_s
714                   *****************
715  c2fd   dc      0c  ldd   SAVE_SP
716  c2ff   bd    c14d  jsr   _hex4
717  c302   cc    ae00  ldd   #$ae00
718  c305   dd      1f  std   buffer
719  c307   39         rts
720  c308             ENDSUB ;   11 [_reg_s]
721
722
723                   *****************
724  c308             SUBROUTINE _reg_dp
725                   *****************
726  c308   cc    1fb3  ldd   #$1fb3     ; buffer[0] = 0x1F;
727  c30b   dd      1f  std   buffer     ; buffer[1] = 0xb3;
728  c30d   4f         clra              ; buffer[4] = 0;
```

```
c30e   5f          clrb            ; buffer[5] = 0;
c30f   dd      23  std   buffer+4
c311   bd    c13d  jsr   _byte_seg
c314   dd      21  std   buffer+2
c316   39          rts
c317             ENDSUB ;   15 [_reg_dp]


                 ******************
c317             SUBROUTINE _low_cc
                 ******************
c317   d6      00  ldb   USER_CC
c319   8e    0002  ldx   #2
c31c             .loop
c31c   86      bd  lda   #$bd       ; 0
c31e   54          lsrb
c31f   24      02  bcc   .zero
c321   86      30  lda   #$30       ; 1
c323             .zero
c323   a7 88   1f  sta   buffer,x
c326   30 01       leax  1,x
c328   8c    0006  cmpx  #6
c32b   26      ef  bne   .loop
c32d   cc    858d  ldd   #$858d     ; buffer[0] = 0x85;
c330   dd      1f  std   buffer     ; buffer[1] = 0x8d;
c332   39          rts
c333             ENDSUB ;   28 [_low_cc]


                 *****************
c333             SUBROUTINE _hi_cc
                 *****************
c333   d6      00  ldb   USER_CC
c335   54          lsrb
c336   54          lsrb
c337   54          lsrb
c338   54          lsrb
c339   8e    0002  ldx   #2
c33c             .loop
c33c   86      bd  lda   #$bd       ; 0
c33e   54          lsrb
c33f   24      02  bcc   .zero
c341   86      30  lda   #$30       ; 1
c343             .zero
c343   a7 88   1f  sta   buffer,x
c346   30 01       leax  1,x
c348   8c    0006  cmpx  #6
c34b   26      ef  bne   .loop
c34d   cc    378d  ldd   #$378d     ; buffer[0] = 0x37;
c350   dd      1f  std   buffer     ; buffer[1] = 0x8d;
c352   39          rts
c353             ENDSUB ;   32 [_hi_cc]


                 *********
c353             reg_table
```

```
                    *********
c353 c2b3            word  _reg_a  ; 0
c355 c2c2            word  _reg_b  ; 1
c357 c2d1            word  _reg_ab ; 2
c359 c2d1            word  _reg_ab ; 3
c35b c333            word  _hi_cc  ; 4
c35d c317            word  _low_cc ; 5
c35f c2dc            word  _reg_x  ; 6
c361 c2e7            word  _reg_y  ; 7
c363 c308            word  _reg_dp ; 8
c365 c2f2            word  _reg_u  ; 9
c367 c2fd            word  _reg_s  ; a

                    ************************
c369                SUBROUTINE _reg_display
                    ************************
c369   8e    c353   ldx   #reg_table
c36c   d6      13   ldb   key
c36e   c1      0a   cmpb  #10
c370   22      04   bhi   .return
c372   3a           abx
c373   3a           abx
c374   6e 94        jmp  [,x]
c376                .return
c376                ENDSUB ;   13 [_reg_display]


                    ******************
c376                SUBROUTINE _insert
                    ******************
c376   d6      1e   ldb   state      ; if (state == 1 || state == 2)
c378   27      1b   beq   .return
c37a   c1      02   cmpb  #2
c37c   22      17   bhi   .return
c37e   9e      10   ldx   CPC        ; for (j=512; j>0; j--)
c380   30 89 0200   leax  512,x
c384                .loop
c384   ec 83        ldd   ,--x       ; PC[j] = PC[j-1];
c386   ed 01        std   1,x
c388   9c      10   cmpx  CPC
c38a   22      f8   bhi   .loop
c38c   6f 84        clr   ,x         ; PC[0] = 0;
c38e   bd    c15a   jsr   _read_memory
c391   c6      02   ldb   #2         ; state = 2;
c393   d7      1e   stb   state
c395                .return
c395   39           rts
c396                ENDSUB ;   32 [_reg_display.return]


                    ********************
c396                SUBROUTINE _cut_byte
                    ********************
c396   d6      1e   ldb   state      ; if (state == 1 || state == 2)
c398   27      19   beq   .return
c39a   c1      02   cmpb  #2
```

```
841  c39c   22      15  bhi   .return
842  c39e   9e      10  ldx   CPC        ; for (j=0; j<512; j++)
843  c3a0 108e    0100  ldy   #256
844  c3a4              .loop
845  c3a4   ec 01       ldd   1,x        ; PC[j] = PC[j+1];
846  c3a6   ed 81       std   ,x++
847  c3a8   31 3f       leay  -1,y
848  c3aa   26      f8  bne   .loop
849  c3ac   bd    c15a  jsr   _read_memory
850  c3af   c6      02  ldb   #2         ; state = 2;
851  c3b1   d7      1e  stb   state
852  c3b3              .return
853  c3b3   39          rts
854  c3b4              ENDSUB ;   30 [_cut_byte]
855
856
857                    ********************
858  c3b4              SUBROUTINE _key_test
859                    ********************
860  c3b4   1c      ef  andcc #$ef       ; enable IRQ
861  c3b6 108e    0000  ldy   #0
862  c3ba 109f      1f  sty   buffer     ; buffer[0] = buffer[1] = 0;
863  c3bd              .while
864  c3bd   bd    c101  jsr   _scan
865  c3c0   d6      16  ldb   tick       ; while (tick < 10)
866  c3c2   c1      0a  cmpb  #10
867  c3c4   25      f7  blo   .while
868  c3c6   0f      16  clr   tick       ; tick = 0;
869  c3c8   1f 20       tfr   y,d
870  c3ca   f7    8000  stb   GPIO1
871  c3cd   bd    c14d  jsr   _hex4      ; _hex4(t);
872  c3d0   31 21       leay  1,y
873  c3d2   20      e9  bra   .while
874  c3d4              ENDSUB ;   32 [_key_test]
875
876
877                    ************************
878  c3d4              SUBROUTINE _clear_buffer
879                    ************************
880  c3d4   4f          clra
881  c3d5   5f          clrb
882  c3d6   dd      1f  std   buffer
883  c3d8   dd      21  std   buffer+2
884  c3da   dd      23  std   buffer+4
885  c3dc   39          rts
886  c3dd              ENDSUB ;    9 [_clear_buffer]
887
888
889                    ******************
890  c3dd              SUBROUTINE _key_cal
891                    ******************
892  c3dd   c6      04  ldb   #4             ; state = 4;
893  c3df   d7      1e  stb   state
894  c3e1   bd    c3d4  jsr   _clear_buffer
895  c3e4   c6      bd  ldb   #$BD          ; buffer[2] = 0xbd;
896  c3e6   d7      21  stb   buffer+2
```

```
c3e8   4f          clra
c3e9   5f          clrb
c3ea   dd      1a  std  start          ; start = 0;
c3ec   d7      14  stb  hit            ; hit   = 0;
c3ee   39          rts
c3ef             ENDSUB ;   18 [_key_cal]


                 *********************
c3ef             SUBROUTINE _enter_num
                 *********************
c3ef   4f          clra
c3f0   5f          clrb
c3f1   0d      14  tst   hit          ; if (hit == 0) num = 0;
c3f3   26      02  bne   .enua
c3f5   dd      18  std   num
c3f7             .enua
c3f7   5c          incb               ; hit = 1;
c3f8   d7      14  stb   hit
c3fa   dc      18  ldd   num          ; num <<= 4;
c3fc   58          lslb
c3fd   49          rola
c3fe   58          lslb
c3ff   49          rola
c400   58          lslb
c401   49          rola
c402   58          lslb
c403   49          rola
c404   da      13  orb   key          ; num |= key;
c406   dd      18  std   num
c408   bd    c14d  jsr   _hex4
c40b   39          rts
c40c             ENDSUB ;   29 [_enter_num]


                 *********************
c40c             SUBROUTINE _key_copy
                 *********************
c40c   c6      05  ldb   #5            ; state = 5;
c40e   d7      1e  stb   state
c410   0f      14  clr   hit           ; hit = 0;
c412   bd    c3d4  jsr   _clear_buffer
c415   c6      ae  ldb   #$AE          ; buffer[0] = 0xae;
c417   d7      1f  stb   buffer
c419   c6      bd  ldb   #$BD          ; buffer[2] = 0xbd;
c41b   d7      21  stb   buffer+2
c41d   39          rts
c41e             ENDSUB ;   18 [_key_copy]


                 *******
c41e             key_tab
                 *******
c41e c20f          word  _key_PC       ; 10
c420 c2a2          word  _key_reg      ; 11
c422 c176          word  _key_data     ; 12
```

```
 953  c424 c16a          word  _key_address ; 13
 954  c426 c43e          word  _key_ignore  ; 14
 955  c428 c43f          word  _key_flag    ; 15
 956  c42a c1ce          word  _key_minus   ; 16
 957  c42c c182          word  _key_plus    ; 17
 958  c42e c376          word  _insert      ; 18
 959  c430 c396          word  _cut_byte    ; 19
 960  c432 c3b4          word  _key_test    ; 1a
 961  c434 c245          word  _key_go      ; 1b
 962  c436 c40c          word  _key_copy    ; 1c
 963  c438 c3dd          word  _key_cal     ; 1d
 964  c43a c531          word  _key_dump    ; 1e
 965  c43c c643          word  _key_load    ; 1f
 966
 967                     **********************
 968  c43e               SUBROUTINE _key_ignore
 969                     **********************
 970  c43e   39           rts
 971  c43f               ENDSUB ;    1 [_key_ignore]
 972
 973
 974                     ********************
 975  c43f               SUBROUTINE _key_flag
 976                     ********************
 977  c43f   d6      15  ldb   flag
 978  c441   c8      01  eorb  #1
 979  c443   d7      15  stb   flag
 980  c445   39          rts
 981  c446               ENDSUB ;    7 [_key_flag]
 982
 983
 984                     ******************
 985  c446               SUBROUTINE _key_exe
 986                     ******************
 987  c446   0d      15  tst   flag
 988  c448   26      03  bne   .laba
 989  c44a   bd    c472  jsr   _beep
 990  c44d               .laba
 991  c44d   d6      13  ldb   key
 992  c44f   c1      10  cmpb  #16
 993  c451   25      0b  blo   .labb
 994  c453   8e    c41e  ldx   #key_tab
 995  c456   c0      10  subb  #16
 996  c458   c4      0f  andb  #15
 997  c45a   3a          abx
 998  c45b   3a          abx
 999  c45c   6e 94       jmp   [,x]
1000  c45e               .labb
1001  c45e   d6      1e  ldb   state
1002  c460   5a          decb
1003  c461 1027    fdb1  lbeq  _hex_address
1004  c465   5a          decb
1005  c466 1027    fd88  lbeq  _data_hex
1006  c46a   5a          decb
1007  c46b 1027    fefa  lbeq  _reg_display
1008  c46f   7e    c3ef  jmp   _enter_num
```

```
c472                ENDSUB ;   44 [_key_exe]


                    ***************
c472                SUBROUTINE _beep
                    ***************
c472   7f    8003   clr   PORT2
c475   8e    003c   ldx   #60
c478                .loop
c478   86      7f   lda   #$7f
c47a   b7    8002   sta   PORT1
c47d   c6      32   ldb   #50
c47f                .dela
c47f   5a           decb
c480   26      fd   bne   .dela
c482   86      ff   lda   #$ff
c484   b7    8002   sta   PORT1
c487   c6      32   ldb   #50
c489                .delb
c489   5a           decb
c48a   26      fd   bne   .delb
c48c   30 1f        leax  -1,x
c48e   26      e8   bne   .loop
c490   39           rts
c491                ENDSUB ;   31 [_beep]


                    *****************
c491                SUBROUTINE _scan1
                    *****************
c491                .whilea            ; while( _scan() != -1);
c491   bd    c101   jsr   _scan
c494   2a      fb   bpl   .whilea
c496 108e    001e   ldy   #30          ; delay(30);
c49a   bd    c0a6   jsr   _delay
c49d                .whileb            ; while( _scan() == -1);
c49d   bd    c101   jsr   _scan
c4a0   2b      fb   bmi   .whileb
c4a2 108e    001e   ldy   #30          ; delay(30);
c4a6   bd    c0a6   jsr   _delay
c4a9   bd    c101   jsr   _scan        ; key = scan();
c4ac   8e    c4b6   ldx   #key_code    ; key = key_code(key);
c4af   e6 85        ldb   b,x
c4b1   d7      13   stb   key
c4b3   7e    c446   jmp   _key_exe
c4b6                ENDSUB ;   37 [_scan1]

c4b6                key_code
c4b6 18 19 1a 0c     BYTE  $18,$19,$1a,$0c, $00,$00,$14,$15,
c4be 16 17 00 00     BYTE  $16,$17,$00,$00, $10,$11,$12,$13,
c4c6 1f 00 0f 0b     BYTE  $1f,$00,$0f,$0b, $07,$03,$00,$1e,
c4ce 0e 0a 06 02     BYTE  $0e,$0a,$06,$02, $04,$1d,$0d,$09,
c4d6 05 01 08 1c     BYTE  $05,$01,$08,$1c, $1b

                    ********************
c4db                SUBROUTINE _initacia
```

```
                        ********************
c4db   c6      03        ldb   #3          ; reset
c4dd   f7    a000        stb   ACIAPORT
c4e0   c6      16        ldb   #$16        ; baudrate = 19200
c4e2   f7    a000        stb   ACIAPORT
c4e5   f6    a001        ldb   ACIAPORT+1 ; clear RBR
c4e8   39                rts
c4e9                   ENDSUB ;   14 [_initacia]


                        ********************
c4e9                   SUBROUTINE _putchar
                        ********************
                        * Input: b = char *
                        ********************
c4e9   86      02        lda   #2          ; wait on TDRE
c4eb                   .wait
c4eb   b5    a000        bita  ACIAPORT
c4ee   27      fb        beq   .wait
c4f0   f7    a001        stb   ACIAPORT+1
c4f3   39                rts
c4f4                   ENDSUB ;   11 [_putchar]


                        ****************
c4f4                   SUBROUTINE _puts
                        ********************
                        * Input: x = char *
                        ********************
c4f4                   .loop
c4f4   e6 80             ldb   ,x+
c4f6   27      05        beq   .return
c4f8   bd    c4e9        jsr   _putchar
c4fb   20      f7        bra   .loop
c4fd                   .return
c4fd   39                rts
c4fe                   ENDSUB ;   10 [_puts]


                        ********************
c4fe                   SUBROUTINE _newline
                        ********************
c4fe   c6      0d        ldb   #13
c500   bd    c4e9        jsr   _putchar
c503   c6      0a        ldb   #10
c505   7e    c4e9        jmp   _putchar
c508                   ENDSUB ;   10 [_newline]


                        ********************
c508                   SUBROUTINE _send_hex
                        ********************
                        * Input: b = byte  *
                        ********************
c508   34 04             pshs  b
c50a   54                lsrb
```

```
1121  c50b   54          lsrb
1122  c50c   54          lsrb
1123  c50d   54          lsrb
1124  c50e   cb      30  addb  #'0'
1125  c510   c1      39  cmpb  #'9'
1126  c512   23      02  bls   .hexh
1127  c514   cb      07  addb  #7
1128  c516              .hexh
1129  c516   bd    c4e9  jsr _putchar
1130  c519   35 04       puls  b
1131  c51b   c4      0f  andb  #15
1132  c51d   cb      30  addb  #'0'
1133  c51f   c1      39  cmpb  #'9'
1134  c521   23      02  bls   .hexl
1135  c523   cb      07  addb  #7
1136  c525              .hexl
1137  c525   7e    c4e9  jmp _putchar
1138  c528              ENDSUB ;   32 [_send_hex]
1139
1140
1141                    ************************
1142  c528              SUBROUTINE _send_word_hex
1143                    ************************
1144  c528   e6 62       ldb   2,s
1145  c52a   bd    c508  jsr   _send_hex
1146  c52d   e6 63       ldb   3,s
1147  c52f   20      d7  bra   _send_hex
1148  c531              ENDSUB ;    9 [_send_word_hex]
1149
1150
1151                    ********************
1152  c531              SUBROUTINE _key_dump
1153                    ********************
1154  c531   32 7e       leas  -2,s
1155  c533   9e      10  ldx   CPC
1156  c535   c6      10  ldb   #16
1157  c537   e7 e4       stb   ,s          ; for (j=0; j<16; j++)
1158
1159  c539              .loopo
1160  c539   bd    c4fe  jsr   _newline
1161  c53c   34 10       pshs  x
1162  c53e   bd    c528  jsr   _send_word_hex
1163  c541   32 62       leas  2,s
1164  c543   c6      3a  ldb   #':'        ; putchar(':');
1165  c545   bd    c4e9  jsr   _putchar
1166  c548   c6      10  ldb   #16
1167  c54a   e7 61       stb   1,s         ; for (p=0; p<16; p++)
1168
1169  c54c              .loopi
1170  c54c   e6 80       ldb   ,x+         ; send_hex();
1171  c54e   bd    c508  jsr   _send_hex
1172  c551   c6      20  ldb   #' '        ; putchar(' ');
1173  c553   bd    c4e9  jsr   _putchar
1174  c556   6a 61       dec   1,s
1175  c558   26      f2  bne   .loopi
1176
```

```
c55a   c6      20  ldb   #' '       ; putchar(' ');
c55c   bd    c4e9  jsr   _putchar
c55f   30 10       leax  -16,x
c561   c6      10  ldb   #16
c563   e7 61       stb   1,s         ; for (p=0; p<16; p++)

c565              .loopj
c565   e6 80       ldb   ,x+         ; putchar();
c567   c1      20  cmpb  #' '
c569   25      04  blo   .dot        ; if (q >= ' ' && q < 0x7F) putchar(q)
c56b   c1      7f  cmpb  #$7f
c56d   25      02  blo   .put
c56f              .dot
c56f   c6      2e  ldb   #'.'
c571              .put
c571   bd    c4e9  jsr   _putchar
c574   6a 61       dec   1,s
c576   26      ed  bne   .loopj

c578   6a e4       dec   ,s
c57a   26      bd  bne   .loopo
c57c   9f      10  stx   CPC
c57e   bd    c4fe  jsr   _newline     ; newline();
c581   bd    c16a  jsr   _key_address ; key_address();
c584   32 62       leas  2,s
c586   39          rts
c587              ENDSUB ;   86 [_key_dump]


                  *****
c587              hihex
                  *****
c587 00 10 20 30   BYTE $00,$10,$20,$30,$40,$50,$60,$70,$80,$90
c591 00 00 00 00   BYTE $00,$00,$00,$00,$00,$00,$00
c598 a0 b0 c0 d0   BYTE $a0,$b0,$c0,$d0,$e0,$f0

                  *****
c59e              lohex
                  *****
c59e 00 01 02 03   BYTE $00,$01,$02,$03,$04,$05,$06,$07,$08,$09
c5a8 00 00 00 00   BYTE $00,$00,$00,$00,$00,$00,$00
c5af 0a 0b 0c 0d   BYTE $0a,$0b,$0c,$0d,$0e,$0f

                  ******************
c5b5              SUBROUTINE _gethex
                  ******************
c5b5   bd    c67c  jsr   _getchar    ; a = getchar();
c5b8   8e    c557  ldx   #hihex-$30
c5bb   e6 85       ldb   b,x
c5bd   34 04       pshs  b
c5bf   bd    c67c  jsr   _getchar    ; b = getchar();
c5c2   8e    c56e  ldx   #lohex-$30
c5c5   e6 85       ldb   b,x
c5c7   ea e0       orb   ,s+         ; a = a | b;
c5c9   34 04       pshs  b
c5cb   db      17  addb  checksum    ; checksum += a
```

```
1233  c5cd   d7      17  stb   checksum
1234  c5cf   35 84       puls  b,pc
1235  c5d1              ENDSUB ;   28 [_gethex]
1236
1237
1238                    ******************
1239  c5d1              SUBROUTINE get16bit
1240                    ******************
1241  c5d1   8d      e2  bsr   _gethex
1242  c5d3   34 04       pshs  b
1243  c5d5   8d      de  bsr   _gethex
1244  c5d7   35 82       puls  a,pc
1245  c5d9              ENDSUB ;    8 [get16bit]
1246
1247
1248                    ************************
1249  c5d9              SUBROUTINE _read_record1
1250                    ************************
1251                    * Output: b = checkerr *
1252                    ************************
1253  c5d9   0f      17  clr   checksum   ; checksum = 0;
1254  c5db   8d      d8  bsr   _gethex    ; byte_count = gethex()-3;
1255  c5dd   c0      03  subb  #3
1256  c5df   34 04       pshs  b
1257  c5e1   8d      ee  bsr   get16bit   ; address16bit = get16bitaddress();
1258  c5e3   f7    8000  stb   GPIO1
1259  c5e6   1f 02       tfr   d,y        ; address16bit;
1260  c5e8   35 02       puls  a          ; byte count
1261  c5ea              .loop
1262  c5ea   8d      c9  bsr   _gethex    ; gethex();
1263  c5ec   e7 a0       stb   ,y+
1264  c5ee   4a          deca
1265  c5ef   26      f9  bne   .loop
1266  c5f1   d6      17  ldb   checksum   ; checksum = ~checksum;
1267  c5f3   53          comb
1268  c5f4   34 04       pshs  b
1269  c5f6   8d      bd  bsr   _gethex
1270  c5f8   e0 e0       subb  ,s+        ; checkerr
1271  c5fa   39          rts
1272  c5fb              ENDSUB ;   34 [_read_record1]
1273
1274
1275                    ************************
1276  c5fb              SUBROUTINE _get_s_record
1277                    ************************
1278  c5fb              .while           ; while (getchar() != 'S');
1279  c5fb   bd    c67c  jsr   _getchar
1280  c5fe   c1      53  cmpb  #'S'
1281  c600   26      f9  bne   .while
1282  c602   bd    c67c  jsr   _getchar
1283  c605   c1      30  cmpb  #'0'       ; case '0': continue
1284  c607   27      f2  beq   .while
1285  c609   c1      31  cmpb  #'1'       ; case '1': read_record1();
1286  c60b   26      18  bne   .exit
1287  c60d   bd    c5d9  jsr   _read_record1
1288  c610   5d          tstb             ; check error
```

```
c611   27      e8  beq   .while
c613   bd    c508  jsr   _send_hex  ; check sum difference
c616   c6      3a  ldb   #':'
c618   bd    c4e9  jsr   _putchar
c61b   1f 20        tfr   y,d        ; last used address
c61d   bd    c528  jsr   _send_word_hex
c620   8e    c62e  ldx   #.MsgErr   ; puts("check sum errors!");
c623   20      03  bra   .geta
c625              .exit
c625   8e    c640  ldx   #.MsgOK     ; else puts("OK");
c628              .geta
c628   bd    c4f4  jsr   _puts
c62b   7e    c176  jmp   _key_data  ; key_data();
c62e 3a 63 68 65  .MsgErr BYTE ":check sum error!",0
c640 4f 4b 00     .MsgOK  BYTE "OK",0
c643              ENDSUB ;   72 [_get_s_record]


                  ********************
c643              SUBROUTINE _key_load
                  ********************

c643   8e    c64c  ldx   #.msg
c646   bd    c4f4  jsr   _puts
c649   7e    c5fb  jmp   _get_s_record
c64c 0d 0a 4c 6f  .msg  BYTE "\r\nLoad Motorola s-record\r\n",0
c667              ENDSUB ;   36 [_key_load]


                  ********************
c667              SUBROUTINE _initreg
                  ********************

c667   cc    0200  ldd  #$0200
c66a   dd      10  std  CPC         ; PC      = 0x0200;
c66c   dd      0e  std  SAVE_PC     ; SAVE_PC = 0x0200;
c66e   cc    7f00  ldd  #$7F00
c671   dd      0c  std  SAVE_SP     ; SAVE_SP = 0x7F00;
c673   dd      08  std  USER_U      ; USER_U  = 0x7F00;
c675   0f      03  clr  USER_DP     ; USER_DP = 0;
c677   1f a8        tfr  CC,A
c679   97      00  sta  USER_CC
c67b   39           rts
c67c              ENDSUB ;   21 [_initreg]


                  ********************
c67c              SUBROUTINE _getchar
                  ********************

c67c   c6      16  ldb   #$16       ; enable receiving
c67e   f7    a000  stb   ACIAPORT
c681   c6      01  ldb   #1
c683              .while             ; while((*acia&1) == 0)
c683   f5    a000  bitb  ACIAPORT
c686   27      fb  beq   .while
```

```
1345  c688   c6      56  ldb   #$56       ; stop sending
1346  c68a   f7    a000  stb   ACIAPORT
1347  c68d   f6    a001  ldb   ACIAPORT+1 ; ch = *(acia+1);
1348  c690   39          rts               ; return ch;
1349  c691              ENDSUB ;   21 [_getchar]
1350
1351
1352                    ****************
1353  c691              SUBROUTINE _main
1354                    ****************
1355
1356  c691   8d      d4  bsr   _initreg   ; initreg();
1357  c693   7f    8000  clr   GPIO1
1358  c696   7f    8003  clr   PORT2
1359  c699   c6      ff  ldb   #$ff
1360  c69b   f7    8002  stb   PORT1
1361  c69e   0f      15  clr   flag       ; flag = 0;
1362  c6a0   bd    c4db  jsr   _initacia  ; initacia();
1363  c6a3   bd    c4fe  jsr   _newline   ; newline();
1364  c6a6   8e    c6d9  ldx   #MSG02
1365  c6a9   bd    c4f4  jsr   _puts      ; puts("6809 MICROPROCESSOR KIT 2020")
1366  c6ac   bd    c074  jsr   _LCD_Init  ; _LCD_Init();
1367  c6af   8e    c6f8  ldx   #MSG03
1368  c6b2   bd    c092  jsr   _LCD_Puts  ; LCD_Puts("6809 MICROPROCESSOR");
1369  c6b5   cc    0100  ldd   #$0100
1370  c6b8   bd    c068  jsr   _LCD_Row_Col ; LCD_Row_Col(1,0);
1371  c6bb   8e    c701  ldx   #MSG04
1372  c6be   bd    c092  jsr   _LCD_Puts  ; LCD_Puts("32kB RAM UART LCD");
1373  c6c1   bd    c3d4  jsr   _clear_buffer
1374  c6c4   c6      af  ldb   #175
1375  c6c6   d7      24  stb   buffer+5   ; buffer[5] = convert[6];
1376  c6c8   c6      bf  ldb   #191
1377  c6ca   d7      23  stb   buffer+4   ; buffer[4] = convert[8];
1378  c6cc   c6      bd  ldb   #189
1379  c6ce   d7      22  stb   buffer+3   ; buffer[3] = convert[0];
1380  c6d0   c6      be  ldb   #190
1381  c6d2   d7      21  stb   buffer+2   ; buffer[2] = convert[9];
1382  c6d4              .loop
1383  c6d4   bd    c491  jsr   _scan1
1384  c6d7   20      fb  bra   .loop
1385  c6d9              ENDSUB ;   72 [_main]
1386
1387  c6d9 36 38 30 39  MSG02 BYTE "6809 MICROPROCESSOR KIT 2020\r\n",0
1388  c6f8 36 38 30 39  MSG03 BYTE "6809 CPU",0
1389  c701 33 32 6b 42  MSG04 BYTE "32kB RAM LCD",0
1390
1391                    END
1392
1393
1394    175 Symbols
1395  -------------
1396  USER_START                    $0000    37D   134
1397  USER_CC                       $0000    38D   583    740    761   1330
1398  USER_A                        $0001    39D    47    581    644    671
1399  USER_D                        $0001    47
1400  USER_B                        $0002    40D   658
```

```
1401  USER_DP                  $0003    41D   585   1328
1402  USER_X                   $0004    42D   576    682
1403  USER_Y                   $0006    43D   577    693
1404  USER_U                   $0008    44D   578    704   1327
1405  USER_PC                  $000a    45
1406  USER_END                 $000c    46D   138
1407  SAVE_SP                  $000c    49D   141    575    715   1326
1408  SAVE_PC                  $000e    50D   140    521   1324
1409  CPC                      $0010    51D   394    396    432    434
1410                                    479   481    503    510    522
1411                                    534   538    548    579    608
1412                                    819   824    842   1155   1198
1413                                   1323
1414  SIGN                     $0012    52D   443    491    593
1415  key                      $0013    55D   281    289    302    314
1416                                    509   547    802    925    991
1417                                   1052
1418  hit                      $0014    56D   406    417    442    451
1419                                    459   489    501    512    532
1420                                    537   602    900    910    915
1421                                    937
1422  flag                     $0015    57D   977    979    987   1361
1423  tick                     $0016    58D   109    865    868
1424  checksum                 $0017    59D  1232   1233   1253   1266
1425  num                      $0018    60D   440    449    462    487
1426                                    595   598    607    912    916
1427                                    926
1428  start                    $001a    61D   441    450    464    488
1429                                    592   600    609    899
1430  _end                     $001c    62D   463    611
1431  state                    $001e    63D   408    419    428    448
1432                                    458   475    564    571    620
1433                                    635   815    829    838    851
1434                                    893   936   1001
1435  buffer                   $001f    64D   251    322    325    326
1436                                    329   330    333    341    344
1437                                    345   348    349    352    383
1438                                    386   398    453    461    559
1439                                    561   563    631    633    647
1440                                    649   661    663    674    685
1441                                    696   707    718    727    730
1442                                    732   748    753    773    778
1443                                    862   882    883    884    896
1444                                    940   942   1375   1377   1379
1445                                   1381
1446  _virq                    $7ff0    66D    79     94     95
1447  _vnmi                    $7ff3    67D    81     98    100
1448  _vfirq                   $7ff6    68D    78     99    101
1449  GPIO1                    $8000    18D   870   1258   1357
1450  PORT0                    $8001    19D   284    297
1451  PORT1                    $8002    20D   255    283   1019   1025
1452                                   1360
1453  PORT2                    $8003    21D   264   1015   1358
1454  LCD_cwr                  $9000    23D   171    188    198    201
1455  LCD_dwr                  $9001    24D   219    232
1456  LCD_crd                  $9002    25D   157
```

```
1457  LCD_drd                    $9003    26
1458  ACIAPORT                   $a000    28D  1067   1069   1070   1082
1459                                      1084   1340   1343   1346   1347
1460  _reset                     $c000    89D    75     76     77     82
1461  _irq_serv                  $c023   107D    93
1462  _swi_serv                  $c026   132D    80
1463  _swi_serv.loop             $c029   135D   139
1464  convert                    $c03a   146D   363
1465  _LCD_Ready                 $c04a   152D   169    185    196    199
1466                                      216    231
1467  _LCD_Ready.loop            $c050   156D   160
1468  _LCD_Ready.return          $c059   161D   158
1469  _LCD_Clear                 $c05b   167D   202
1470  LCD_Row_Val                $c064   176D   186
1471  _LCD_Row_Col               $c068   180D   205   1370
1472  _LCD_Init                  $c074   194D  1366
1473  _LCD_Puts                  $c092   212D   220   1368   1372
1474  _LCD_Puts.return           $c09e   221D   218
1475  _LCD_Putc                  $c09f   227
1476  _delay                     $c0a6   238D   207    244    262    266
1477                                     1043  1048
1478  _seg_refresh               $c0ab   248D   312
1479  _seg_refresh.loop          $c0b0   252D   270
1480  _seg_refresh.sega          $c0c3   261D   257    259
1481  _kbd_scan                  $c0d6   276D   313
1482  _kbd_scan.loopo            $c0da   282D   295
1483  _kbd_scan.loopi            $c0e3   286D   291
1484  _kbd_scan.kbda             $c0fe   301D   299
1485  _kbd_scan.return           $c100   303D   288
1486  _scan                      $c101   309D   864   1040   1045   1049
1487  _dot_address               $c10b   320D   410    550
1488  _dot_data                  $c124   339D   399    421    514    618
1489  _byte_seg                  $c13d   358D   382    385    397    731
1490  _hex4                      $c14d   377D   395    601    645    659
1491                                      672    683    694    705    716
1492                                      871    927
1493  _read_memory               $c15a   392D   409    420    435    482
1494                                      513    549    617    827    849
1495  _key_address               $c16a   404D   953   1200
1496  _key_data                  $c176   415D   436    483    523    952
1497                                     1301
1498  _key_plus                  $c182   426D   957
1499  _key_plus.plusa            $c197   437D   429    431
1500  _key_plus.plusb            $c1a4   445D   439
1501  _key_plus.plusc            $c1b5   455D   447
1502  _key_plus.return           $c1cd   467D   457    465
1503  _key_minus                 $c1ce   473D   956
1504  _key_minus.mina            $c1e3   484D   476    478
1505  _key_minus.return          $c1f1   492D   486
1506  _data_hex                  $c1f2   498D  1005
1507  _data_hex.hexa             $c1ff   508D   502
1508  _key_PC                    $c20f   519D   142    950
1509  _hex_address               $c216   528D  1003
1510  _hex_address.hexa          $c21e   535D   533
1511  _print_error               $c235   555D   466
1512  _key_go                    $c245   569D   961
```

```
_key_go.goa                $c269   589D   572    574
_key_go.pos                $c277   597D   594
_key_go.gob                $c279   599D   596
_key_go.goc                $c281   604D   591
_key_go.loop               $c28c   610D   615
_key_go.exit               $c297   616D   612
_key_go.return             $c2a1   621D   606
_key_reg                   $c2a2   627D   951
_reg_a                     $c2b3   641D   786
_reg_b                     $c2c2   655D   787
_reg_ab                    $c2d1   669D   788    789
_reg_x                     $c2dc   680D   792
_reg_y                     $c2e7   691D   793
_reg_u                     $c2f2   702D   795
_reg_s                     $c2fd   713D   796
_reg_dp                    $c308   724D   794
_low_cc                    $c317   738D   791
_low_cc.loop               $c31c   742D   751
_low_cc.zero               $c323   747D   745
_hi_cc                     $c333   759D   790
_hi_cc.loop                $c33c   767D   776
_hi_cc.zero                $c343   772D   770
reg_table                  $c353   784D   801
_reg_display               $c369   799D  1007
_reg_display.return        $c376   808D   804
_insert                    $c376   813D   958
_insert.loop               $c384   821D   825
_insert.return             $c395   830D   816    818
_cut_byte                  $c396   836D   959
_cut_byte.loop             $c3a4   844D   848
_cut_byte.return           $c3b3   852D   839    841
_key_test                  $c3b4   858D   960
_key_test.while            $c3bd   863D   867    873
_clear_buffer              $c3d4   878D   629    894    938   1373
_key_cal                   $c3dd   890D   963
_enter_num                 $c3ef   906D  1008
_enter_num.enua            $c3f7   913D   911
_key_copy                  $c40c   933D   962
key_tab                    $c41e   948D   994
_key_ignore                $c43e   968D   954
_key_flag                  $c43f   975D   955
_key_exe                   $c446   985D  1053
_key_exe.laba              $c44d   990D   988
_key_exe.labb              $c45e  1000D   993
_beep                      $c472  1013D   989
_beep.loop                 $c478  1017D  1031
_beep.dela                 $c47f  1021D  1023
_beep.delb                 $c489  1027D  1029
_scan1.whilea              $c491  1039D  1041
_scan1                     $c491  1037D  1383
_scan1.whileb              $c49d  1044D  1046
key_code                   $c4b6  1056D  1050
_initacia                  $c4db  1064D  1362
_putchar                   $c4e9  1076D  1097   1108   1110   1129
                                  1137   1165   1173   1178   1192
                                  1292
```

```
_putchar.wait                   $c4eb  1081D  1083
_puts.loop                      $c4f4  1094D  1098
_puts                           $c4f4  1090D  1300   1312   1365
_puts.return                    $c4fd  1099D  1096
_newline                        $c4fe  1105D  1160   1199   1363
_send_hex                       $c508  1115D  1145   1147   1171   1290
_send_hex.hexh                  $c516  1128D  1126
_send_hex.hexl                  $c525  1136D  1134
_send_word_hex                  $c528  1142D  1162   1294
_key_dump                       $c531  1152D   964
_key_dump.loopo                 $c539  1159D  1197
_key_dump.loopi                 $c54c  1169D  1175
_key_dump.loopj                 $c565  1183D  1194
_key_dump.dot                   $c56f  1189D  1186
_key_dump.put                   $c571  1191D  1188
hihex                           $c587  1207D  1224
lohex                           $c59e  1214D  1228
_gethex                         $c5b5  1221D  1241   1243   1254   1262
                                       1269
get16bit                        $c5d1  1239D  1257
_read_record1                   $c5d9  1249D  1287
_read_record1.loop              $c5ea  1261D  1265
_get_s_record                   $c5fb  1276D  1313
_get_s_record.while             $c5fb  1278D  1281   1284   1289
_get_s_record.exit              $c625  1297D  1286
_get_s_record.geta              $c628  1299D  1296
_get_s_record.MsgErr            $c62e  1302D  1295
_get_s_record.MsgOK             $c640  1303D  1298
_key_load                       $c643  1308D   965
_key_load.msg                   $c64c  1314D  1311
_initreg                        $c667  1319D  1356
_getchar                        $c67c  1336D  1223   1227   1279   1282
_getchar.while                  $c683  1342D  1344
_main                           $c691  1353D   102
_main.loop                      $c6d4  1382D  1384
MSG02                           $c6d9  1387D  1364
MSG03                           $c6f8  1388D  1367
MSG04                           $c701  1389D  1371
buffer                          $001f    64D   251    322    325    326
                                        329    330    333    341    344
                                        345    348    349    352    383
                                        386    398    453    461    559
                                        561    563    631    633    647
                                        649    661    663    674    685
                                        696    707    718    727    730
                                        732    748    753    773    778
                                        862    882    883    884    896
                                        940    942   1375   1377   1379
                                       1381
CPC                             $0010    51D   394    396    432    434
                                        479    481    503    510    522
                                        534    538    548    579    608
                                        819    824    842   1155   1198
                                       1323
state                           $001e    63D   408    419    428    448
                                        458    475    564    571    620
```

```
                                        635    815    829    838    851
                                        893    936   1001
hit                             $0014    56D   406    417    442    451
                                        459    489    501    512    532
                                        537    602    900    910    915
                                        937
num                             $0018    60D   440    449    462    487
                                        595    598    607    912    916
                                        926
key                             $0013    55D   281    289    302    314
                                        509    547    802    925    991
                                       1052
start                           $001a    61D   441    450    464    488
                                        592    600    609    899
checksum                        $0017    59D  1232   1233   1253   1266
flag                            $0015    57D   977    979    987   1361
SAVE_SP                         $000c    49D   141    575    715   1326
USER_A                          $0001    39D    47    581    644    671
USER_CC                         $0000    38D   583    740    761   1330
tick                            $0016    58D   109    865    868
SIGN                            $0012    52D   443    491    593
SAVE_PC                         $000e    50D   140    521   1324
USER_U                          $0008    44D   578    704   1327
_end                            $001c    62D   463    611
USER_Y                          $0006    43D   577    693
USER_X                          $0004    42D   576    682
USER_DP                         $0003    41D   585   1328
USER_END                        $000c    46D   138
USER_B                          $0002    40D   658
USER_START                      $0000    37D   134
USER_PC                         $000a    45
USER_D                          $0001    47
buffer                          $001f    64D   251    322    325    326
                                        329    330    333    341    344
                                        345    348    349    352    383
                                        386    398    453    461    559
                                        561    563    631    633    647
                                        649    661    663    674    685
                                        696    707    718    727    730
                                        732    748    753    773    778
                                        862    882    883    884    896
                                        940    942   1375   1377   1379
                                       1381
CPC                             $0010    51D   394    396    432    434
                                        479    481    503    510    522
                                        534    538    548    579    608
                                        819    824    842   1155   1198
                                       1323
state                           $001e    63D   408    419    428    448
                                        458    475    564    571    620
                                        635    815    829    838    851
                                        893    936   1001
hit                             $0014    56D   406    417    442    451
                                        459    489    501    512    532
                                        537    602    900    910    915
                                        937
```

```
num                             $0018    60D   440    449    462    487
                                         595   598    607    912    916
                                         926
key                             $0013    55D   281    289    302    314
                                         509   547    802    925    991
                                        1052
start                           $001a    61D   441    450    464    488
                                         592   600    609    899
checksum                        $0017    59D  1232   1233   1253   1266
flag                            $0015    57D   977    979    987   1361
SAVE_SP                         $000c    49D   141    575    715   1326
USER_A                          $0001    39D    47    581    644    671
USER_CC                         $0000    38D   583    740    761   1330
tick                            $0016    58D   109    865    868
SIGN                            $0012    52D   443    491    593
SAVE_PC                         $000e    50D   140    521   1324
USER_U                          $0008    44D   578    704   1327
_end                            $001c    62D   463    611
USER_Y                          $0006    43D   577    693
USER_X                          $0004    42D   576    682
USER_DP                         $0003    41D   585   1328
USER_END                        $000c    46D   138
USER_B                          $0002    40D   658
USER_START                      $0000    37D   134
USER_PC                         $000a    45
USER_D                          $0001    47
```