```
SERIES-III 8C86/8C87/8088 MACRO ASSEMBLER V1.0 ASSEMBLY OF MODULE CLOCK
OBJECT MODULE PLACED IN :F1:CLOCK.OBJ
INVOCATION LINE CONTROLS:  DEBUG


LOC  OBJ                  LINE     SOURCE

                             1 +1  $TITLE ('KAOS - REAL-TIME AND ALARM CLOCK ROUTINES')
                             2           NAME    CLOCK
                             3
                             4     ;;;   Intel Corporation Proprietary Information. This
                             5     ;     listing is supplied under the terms of a license
                             6     ;     agreement with Intel Corporation and may not be copied
                             7     ;     nor disclosed except in accordance with the terms of
                             8     ;     the agreement.
                             9
                            10
                            11     ;;;   ALARM CONTROL BLOCK DEFINITION
                            12     ;
                            13     ;     FLAG BITS MEAN:
                            14     ;        1 - ALARM TRIGGERED IFF SET.
                            15     ;        2 - ALARM UNKNOWN TO KAOS IFF SET.
                            16
----                        17     ALARM   STRUC
0000                        18     NEXT    DD      0
0004                        19     UTYPE   DB      0
0005                        20     FLAG    DB      0
0006                        21     LAST    DD      0
000A                        22     DLOW    DW      0
000C                        23     DHIGH   DW      0
000E                        24     RMBX    DW      0
----                        25     ALARM   ENDS
                            26
                            27
                            28     ;       MAILBOX HEADER DEFINITION
                            29
----                        30     HEADER  STRUC
0000                        31     HEAD    DW      0
0002                        32     TAIL    DW      0
0004                        33     TYPEF   DW      0
0006                        34     COUNT   DW      0
0008                        35     MHEAD   DD      0
000C                        36     MTAIL   DD      0
----                        37     HEADER  ENDS
                            38
  0015                      39     MBX     EQU     15H             ;MAILBOX TYPE
                            40
                            41
                            42     ;       DATA BASE, EXTERNAL AND GROUP DEFINITONS
                            43
                            44             EXTRN   ISEND:NEAR
                            45
                            46     DGROUP          GROUP   DATA
----                        47     DATA            SEGMENT byte PUBLIC 'DATA'
0000 (1                     48     TIME_HEAD       DD      1 DUP (?)
     ????????
     )
```

```
LOC  OBJ                  LINE     SOURCE

0004 (2                     49                    DB      2 DUP (?)       ;DUMMY BYTES FOR TYPE AND FLAG
     ??
     )
0006 (1                     50                    DD      1 DUP (?)
     ????????
     )
000A (1                     51     TIME_LOW       DW      1 DUP (?)
     ????
     )
000C (1                     52     TIME_HIGH      DW      1 DUP (?)
     ????
     )
----                        53     DATA           ENDS
                            54
                            55     CGROUP  GROUP  CODE
----                        56     CODE    SEGMENT byte PUBLIC 'CODE'
                            57     ASSUME  CS:CGROUP,DS:DGROUP
                            58
                            59             EXTRN  CQHALTANDCATCHFIRE:NEAR
                            60
0000 ----           R       61     DGRP    DW     DGROUP
                            62
                            63 +1  $ EJECT
```

```
LOC  OBJ                  LINE     SCURCE

                            64 +1  $INCLUDE (:F1:PORTS.INC)
                     =1     65
                     =1     66     ;       CONTOL PORTS
                     =1     67
  0000               =1     68     SET_TXSRT          EQU     000H
  0001               =1     69     RESET_TXSRT        EQU     001H
  0002               =1     70     SET_RXAV1          EQU     002H
  0003               =1     71     SET_RXAV2          EQU     003H
  0004               =1     72     SET_RXAV3          EQU     004H
  0005               =1     73     RESET_ERRCR        EQU     005H
  0006               =1     74     RESET_CHANNEL_COUNTER  EQU     006H
  00A0               =1     75     SET_SYS            EQU     0A0H
  00B0               =1     76     SET_LOC            EQU     0B0H
                     =1     77
                     =1     78
                     =1     79     ;       PIT VALUES
                     =1     80
  00D3               =1     81     PITCMD             EQU     0D3H          ;PIT COMMAND PORT
  00D2               =1     82     PIT_BCK            EQU     0D2H          ;PIT BACKOFF TIMER
  00D1               =1     83     PIT_RTC            EQU     0D1H          ;PIT REAL-TIME CLOCK PORT
  00D0               =1     84     PIT_ALC            EQU     0D0H          ;PIT ALARM-CLOCK PORT
                     =1     85
  0000               =1     86     LATCH_RTC          EQU     00H           ;LATCH REAL-TIME CLOCK VALUE
  0040               =1     87     LATCH_ALC          EQU     40H           ;LATCH ALARM-CLOCK VALUE
                     =1     88
                     =1     89
                     =1     90     ;       DMA VALUES
                     =1     91
  00C8               =1     92     DMACMD             EQU     0C8H          ;DMA COMMAND AND STATUS PORT
  00C9               =1     93     DMAREQ             EQU     0C9H          ;DMA REQUEST PORT
  00CA               =1     94     DMAMSKB            EQU     0CAH          ;DMA MASK BIT PORT
  00CB               =1     95     DMAMODE            EQU     0CBH          ;DMA MODE PORT
  00CC               =1     96     DMABPTR            EQU     0CCH          ;DMA BYTE POINTER PORT
  00CD               =1     97     DMACLR             EQU     0CDH          ;? (TMP & CLEAR??)
  00CF               =1     98     DMAMASK            EQU     0CFH          ;DMA MASK PORT
                     =1     99
  00C0               =1    100     CH0ADDR            EQU     0C0H          ;CHANNEL 0 ADDRESS PORT
  00C1               =1    101     CH0WC              EQU     0C1H          ;CHANNEL 0 WORD COUNT PORT
  00C2               =1    102     CH1ADDR            EQU     0C2H          ;CHANNEL 1 ADDRESS PORT
  00C3               =1    103     CH1WC              EQU     0C3H          ;CHANNEL 1 WORD COUNT PORT
  00C4               =1    104     CH2ADDR            EQU     0C4H          ;CHANNEL 2 ADDRESS PORT
  00C5               =1    105     CH2WC              EQU     0C5H          ;CHANNEL 2 WORD COUNT PORT
  00C6               =1    106     CH3ADDR            EQU     0C6H          ;CHANNEL 3 ADDRESS PORT
  00C7               =1    107     CH3WC              EQU     0C7H          ;CHANNEL 3 WORD COUNT PORT
                     =1    108
                     =1    109
                     =1    110     ;       PIO VALUES
                     =1    111
  00E3               =1    112     PIOCMD             EQU     0E3H          ;PIO COMMAND PORT (CHECK: F3?)
  00E0               =1    113     PIOA               EQU     0E0H          ;PIO PORT A
  00E1               =1    114     PIOB               EQU     0E1H          ;PIO PORT B
  00E2               =1    115     PIOC               EQU     0E2H          ;PIO PORT C
                     =1    116
  00F3               =1    117     PIOCLR             EQU     11110011B     ;CLEAR SERDES
  002A               =1    118     PIOSEN             EQU     00101010B     ;SERIAL ENABLE
```

```
LOC  OBJ                  LINE     SOURCE

  0022                =1    119     PIOSENP         EQU     00100010B       ;PROMISCUOUS SERIAL ENABLE
  00E0                =1    120     PIOREAD         EQU     11100000B       ;READ ADDRESS COMMAND
                      =1    121
                      =1    122
                      =1    123     ;       PIC VALUES
                      =1    124
  00F0                =1    125     PICCMD          EQU     0F0H            ;PIC COMMAND PORT
  00F0                =1    126     PICDATA         EQU     0F0H            ;PIC DATA PORT
  00F1                =1    127     PICMASK         EQU     0F1H            ;PIC MASK PORT
                      =1    128
  0020                =1    129     ECI_PIC         EQU     20H             ;PIC END-OF-INTERRUPT COMMAND
  0060                =1    130     SEOI_PIC        EQU     60H             ;PIC SELECTIVE EOI COMMAND
  000C                =1    131     POLL_PIC        EQU     0CH             ;POLL PIC COMMAND
  000A                =1    132     READ_IRR        EQU     0AH             ;PIC READ-IRR COMMAND
  0040                =1    133     RTC_DONE        EQU     40H             ;MASK FOR REAL-TIME CLOCK INTERRUPT
  0066                =1    134     RTC_INT_SEOI    EQU     60H+6           ;COMMAND TO EOI RTC INTERRUPT
  0086                =1    135     RTC_INT         EQU     80H+6           ;POLL COMMAND RETURN IF INTERRUPT
                      =1    136
  0001                =1    137     CH1_DONE        EQU     01H             ;RX CHANNEL 1 DONE
  0002                =1    138     CH2_DONE        EQU     02H             ;RX CHANNEL 2 DONE
  0004                =1    139     CH3_DONE        EQU     04H             ;RX CHANNEL 3 DONE
                      =1    140
                      =1    141
                      =1    142     ;       MISC. DEFINITIONS
                      =1    143
  0001                =1    144     MCFLAG          EQU     01H             ;MULTICAST BIT
  0007                =1    145     JUNK_BYTES      EQU     7               ;NUMBER OF RECIEVE JUNK BYTES
                            146
                            147 +1  $ EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           148     ;;;     TIMERINIT - INITIALIZE TIMER DATA STRUCTURES
                           149     ;
                           150     ;          THIS RCUTINE IS CALLED BY "START" TO INITIALIZE TIMER DATA BASES.
                           151
                           152
                           153             PUBLIC  TIMERINIT
0002                       154     TIMERINIT:
0002 BB0000         R      155             MOV     BX,OFFSET DGROUP:TIME_HEAD
0005 891F                  156             MOV     WORD PTR [BX].NEXT,BX   ;INITIALIZE TIME_HEAD
0007 8C5F02                157             MOV     WORD PTR [BX].NEXT+2,DS
000A 895F06                158             MOV     WORD PTR [BX].LAST,BX
000D 8C5F08                159             MOV     WORD PTR [BX].LAST+2,DS
0010 33C0                  160             XOR     AX,AX
0012 A30A00         R      161             MOV     TIME_LOW,AX     ;INITIALIZE REAL-TIME CLOCK
0015 A30C00         R      162             MOV     TIME_HIGH,AX
0018 C3                    163             RET
                           164
                           165 +1  $ EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           166     ;;;     READ$CLOCK (DATA$RTN$0) - READ REAL-TIME CLOCK.
                           167     ;
                           168     ;       PARAMETER:
                           169     ;          DATA$RTN$0 - OFFSET OF A 4 BYTE BLOCK FOR RETURN OF THE
                           170     ;             VALUE OF THE REAL-TIME CLOCK.  THIS VALUE IS STORED
                           171     ;             IN STANDARD FORMAT, I.E., LOW BYTE FIRST.
                           172
                           173
                           174             PUBLIC  READCLOCK
0019                       175     READCLOCK:
0019 5E                    176             POP     SI              ;POP RETURN ADDRESS
001A 5B                    177             POP     BX              ;POP RETURN POINTER
001B 9C                    178             PUSHF                   ;SAVE INTERRUPT FLAG
001C FA                    179             CLI                     ;CLEAR INTERRUPTS
001D B000                  180             MOV     AL,LATCH_RTC
001F E6D3                  181             OUT     PITCMD,AL       ;LATCH UP COUNT
                           182
0021 E4F1                  183             IN      AL,PICMASK      ;SAVE MASK IN CL
0023 8AC8                  184             MOV     CL,AL
0025 B0BF                  185             MOV     AL,NOT RTC_DONE ;OPEN ONLY RTC INTERRUPT LEVEL
0027 E6F1                  186             OUT     PICMASK,AL
0029 B00C                  187             MOV     AL,POLL_PIC     ;POLL TO SEE IF INTERRUPT ASSERTED
002B E6F0                  188             OUT     PICCMD,AL
002D E4F0                  189             IN      AL,PICDATA
002F 86C1                  190             XCHG    AL,CL           ;RESTORE PIC MASK
0031 E6F1                  191             OUT     PICMASK,AL
0033 80F186                192             XOR     CL,RTC_INT      ;TEST IF INTERRUPT ASSERTED
0036 7513                  193             JNZ     READ1
                           194
0038 B066                  195             MOV     AL,RTC_INT_SEOI ;CLEAR INTERRUPT FLAG IN PIC
003A E6F0                  196             OUT     PICCMD,AL
003C B000                  197             MOV     AL,LATCH_RTC    ;RE-LATCH TIME
003E E6D3                  198             OUT     PITCMD,AL
0040 81060A000004    R     199             ADD     TIME_LOW,400H   ;INCREMENT MEMORY PART OF TIMER
0046 83160C0000      R     200             ADC     TIME_HIGH,0
                           201
0048 E4D1                  202     READ1:  IN      AL,PIT_RTC      ;READ REAL-TIME CLOCK
004D 8AD0                  203             MOV     DL,AL
004F E4D1                  204             IN      AL,PIT_RTC
0051 8AF0                  205             MOV     DH,AL
0053 F7D2                  206             NOT     DX              ;COMPLEMENT COUNT -- DOWN COUNTER
0055 81E2FF7F              207             AND     DX,7FFFH        ;MASK OFF BOTTOM 15 BITS
0059 B105                  208             MOV     CL,5            ;SHIFT
005B D3EA                  209             SHR     DX,CL
005D A10A00          R     210             MOV     AX,TIME_LOW     ;ADD RTC TO STORED VALUE
0060 03C2                  211             ADD     AX,DX
0062 8907                  212             MOV     [BX],AX         ;PUT SUM IN RETURN BLOCK
0064 A10C00          R     213             MOV     AX,TIME_HIGH
0067 150000                214             ADC     AX,0
006A 894702                215             MOV     [BX+2],AX
006D 9D                    216             POPF                    ;RETURN
006E FFE6                  217             JMP     SI
                           218
                           219 +1  $ EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           220     ;;      CLOCK$INT - REAL-TIME CLOCK INTERRPUT SERVICE ROUTINE.
                           221     ;
                           222     ;       PARAMETERS:
                           223     ;          NONE.
                           224     ;
                           225     ;       NOTE: THIS ROUTINE CALLS DLLETHERLOAD SO THAT IT MAY COMPUTE
                           226     ;          THE ETHER LOADING.  THIS ROUTINE MUST TAKE INTO ACCOUNT
                           227     ;          THAT ONLY DS AND AX HAVE BEEN SAVED, AND SAVE ANY OTHER
                           228     ;          REGISTERS IT USES.
                           229
                           230
                           231             PUBLIC  CLOCKINT
                           232             EXTRN   DLLETHERLOAD:NEAR
0070                       233     CLOCKINT:
0070 50                    234             PUSH    AX
0071 1E                    235             PUSH    DS
0072 2E8E1E0C00     R      236             MOV     DS,DGRP
0077 81060A0C0004   R      237             ADD     TIME_LOW,400H   ;SERVICE REAL-TIME CLOCK INTERRUPT
007D 83160C0C00     R      238             ADC     TIME_HIGH,0
0082 B020                  239             MOV     AL,EOI_PIC
0084 E6F0                  240             OUT     PICCMD,AL
0086 E80000         E      241             CALL    DLLETHERLOAD
0089 1F                    242             POP     DS
008A 58                    243             POP     AX
008B CF                    244             IRET
                           245
                           246
                           247
                           248
                           249     ;;;     RANDOM - RETURN "RANDOM" NUMBER.
                           250     ;
                           251     ;          THIS ROUTINE RETURNS THE VALUE OF THE REAL-TIME
                           252     ;       CLOCK TIMER IN THE 8253 FOR USE BY THE DATA LINK LAYER
                           253     ;       IN COMPUTING BACK-OFF TIME.
                           254     ;
                           255     ;       RETURNS:
                           256     ;          AX = TIMER VALUE.
                           257
                           258
                           259             PUBLIC  RANDOM
008C 9C                    260     RANDOM: PUSHF                   ;SAVE INTERRUPT FLAG
008D FA                    261             CLI                     ;CLEAR INTERRUPTS
008E B000                  262             MOV     AL,LATCH_RTC
0090 E6D3                  263             OUT     PITCMD,AL       ;LATCH UP COUNT
0092 E4D1                  264             IN      AL,PIT_RTC      ;READ REAL-TIME CLOCK
0094 8AE0                  265             MOV     AH,AL
0096 E4D1                  266             IN      AL,PIT_RTC
0098 86E0                  267             XCHG    AH,AL
009A 9D                    268             POPF
009B C3                    269             RET
                           270
                           271 +1  $ EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           272     ;;;     SETALARM (ALARM$P, MAILBOX$O, DELAY$H, DELAY$L) - SET ALARM CLOCK.
                           273     ;
                           274     ;       PARAMETERS:
                           275     ;          ALARM$P - POINTER TO A 16 BYTE BLOCK OF MEMORY FOR
                           276     ;             THE ALARM CONTROL BLOCK.
                           277     ;          MAILBOX$O - OFFSET OF A MAILBOX TO RETURN THE ALARM CONTROL
                           278     ;             BLOCK AFTER THE ALARM GOES OFF.  IF ZERO,  DO NOT SEND
                           279     ;             TO A MAILBOX.
                           280     ;          DELAY$H, DEALY$L - THE HIGH AND LOW WORDS OF THE TIME DELAY
                           281     ;             IN 800 NS CLOCK TICKS TO WAIT BEFORE THE ALARM GOES OFF.
                           282
                           283
009C B80800                284     SETO:   MOV     AX,8            ; TRIED TO SET AN ALREADY SET ACB
009F 50                    285             PUSH    AX
00A0 E80000          E     286             CALL    CQHALTANDCATCHFIRE
                           287
                           288
                           289             PUBLIC  SETALARM
00A3                       290     SETALARM:
00A3 58                    291             POP     AX              ;POP RETURN ADDRESS
00A4 59                    292             POP     CX              ;CX = DELAY$L
00A5 5F                    293             POP     DI              ;DI = DELAY$H
00A6 5A                    294             POP     DX              ;POP MAILBOX$O
00A7 5E                    295             POP     SI              ;SI = ALARM$O
00A8 07                    296             POP     ES              ;ES = ALARM$B
00A9 26807C0500            297             CMP     ES:[SI].FLAG,0
00AE 74EC                  298             JZ      SETO
00B0 2689540E              299             MOV     ES:[SI].RMBX,DX ;STORE MAILBOX$O INTO CONTROL BLOCK
00B4 50                    300             PUSH    AX              ;RESTORE RETURN ADDRESS
00B5 9C                    301             PUSHF                   ;SAVE FLAGS
00B6 FA                    302             CLI                     ;RUN WITH INTERRUPTS OFF
00B7 C51E0000        R     303             LDS     BX,TIME_HEAD    ;LOAD POINTER TO FIRST ALARM CONTROL BLOCK
                           304
                           305     ;       REGISTER USAGE:
                           306     ;          AX    - TEMPORARY
                           307     ;          DS:BX - CURRENT ALARM CONTROL BLOCK
                           308     ;          DI:CX - DELAY$H:DELAY$L
                           309     ;          DX    - UNUSED
                           310     ;          ES:SI - NEW ALARM CONTOL BLOCK (NOT ACCESSED IN THIS SECTION)
                           311
00BB 81FB0000        R     312             CMP     BX,OFFSET DGROUP:TIME_HEAD      ;TEST FOR NULL LIST
00BF 7507                  313             JNZ     SET1                    ;IF NOT NULL
00C1 8CD8                  314             MOV     AX,DS
00C3 3D----          R     315             CMP     AX,SEG DGROUP:TIME_HEAD
00C6 741D                  316             JZ      SET2                    ;IF NULL
                           317
00C8 B040                  318     SET1:   MOV     AL,LATCH_ALC    ;READ TIMER TO AX
00CA E6D3                  319             OUT     PITCMD,AL
00CC E4D0                  320             IN      AL,PIT_ALC
00CE 8AE0                  321             MOV     AH,AL
00D0 E4D0                  322             IN      AL,PIT_ALC
00D2 86C4                  323             XCHG    AL,AH
00D4 0BFF                  324             OR      DI,DI           ;TEST IF DELAY$H = 0;
00D6 7529                  325             JNZ     SET3            ;IF NOT ZERO
00D8 3BC1                  326             CMP     AX,CX           ;TEST IF TIMER > DELAY
```

```
LOC  OBJ                  LINE     SOURCE

00DA 7625                  327              JNA     SET3            ;IF TIMER <= DELAY
                           328
00DC 2BC1                  329              SUB     AX,CX           ;AX = TIMER - DELAY
00DE 01470A                330              ADD     [BX].DLOW,AX
00E1 83570C00              331              ADC     [BX].DHIGH,0
                           332
                           333      ;       NEW ALARM GOES ON THE HEAD OF THE LIST
                           334
00E5 0BFF                  335      SET2:   OR      DI,DI           ;TEST IF DELAY$H = 0
00E7 750C                  336              JNZ     SET2B           ;IF NOT ZERO
00E9 0BC9                  337              OR      CX,CX           ;TEST IF DELAY$H = 0
00EB 7501                  338              JNZ     SET2A           ;IF NOT ZERO
00ED 41                    339              INC     CX              ;SET TO AS CLOSE TO ZERO AS WE GET
00EE 8BC1                  340      SET2A:  MOV     AX,CX           ;SET TIMER TO LOW PART OF DELAY
00F0 33C9                  341              XOR     CX,CX           ;SET LOW PART OF DELAY = 0
00F2 EB0590                342              JMP     SET2C
00F5 4F                    343      SET2B:  DEC     DI              ;DELAY$H = DELAY$H - 1
00F6 B80000                344              MOV     AX,0            ;SET TIMER TO MAX
00F9 E6D0                  345      SET2C:  OUT     PIT_ALC,AL
00FB 8AC4                  346              MOV     AL,AH
00FD E6D0                  347              OUT     PIT_ALC,AL
00FF EB2A                  348              JMP     SHORT SET7      ;SKIP "ELSE" SECTION
                           349
                           350      ;       NEW ALARM NOT AT THE HEAD OF THE LIST
                           351
0101 2BC8                  352      SET3:   SUB     CX,AX           ;DELAY -= TIMER
0103 83DF00                353              SBB     DI,0
0106 EB02                  354              JMP     SHORT SET5      ;ENTER LOOP
                           355
0108 C51F                  356      SET4:   LDS     BX,[BX].NEXT    ;ADVANCE TO NEXT ENTRY
010A 81FB000C        R     357      SET5:   CMP     BX,OFFSET DGROUP:TIME_HEAD      ;TEST IF END OF LIST
010E 7507                  358              JNZ     SET6            ;IF NOT END OF LIST
0110 8CD8                  359              MOV     AX,DS
0112 3D----          R     360              CMP     AX,SEG DGROUP:TIME_HEAD
0115 7414                  361              JZ      SET7            ;IF END OF LIST
0117 2B4F0A                362      SET6:   SUB     CX,[BX].DLOW    ;DELAY -= BX.DELAY
011A 1B7F0C                363              SBB     DI,[BX].DHIGH
011D 73E9                  364              JNC     SET4            ;LOOP IF NO BORROW
                           365
011F 034F0A                366              ADD     CX,[BX].DLOW    ;PUT IT BACK THE WAY IT WAS
0122 137F0C                367              ADC     DI,[BX].DHIGH
0125 294F0A                368              SUB     [BX].DLOW,CX    ;BX.DELAY -= DELAY
0128 197F0C                369              SBB     [BX].DHIGH,DI
                           370
012B 26894C0A              371      SET7:   MOV     ES:[SI].DLOW,CX ;STORE DELAY
012F 26897C0C              372              MOV     ES:[SI].DHIGH,DI
0133 26C6440500            373              MOV     ES:[SI].FLAG,0  ;SET FLAG TO 'WAITING'
                           374
                           375      ;       REGISTER USAGE WHILE LINKING NEW ACB INTO LIST:
                           376      ;          AX:DI - PREDECESSOR OF CURRENT ALARM CONTROL BLOCK
                           377      ;          DS:BX - CURRENT ALARM CONTROL BLOCK
                           378      ;          CX    - UNUSED
                           379      ;          DX    - UNUSED
                           380      ;          ES:SI - NEW ALARM CONTOL BLOCK
                           381
```

```
LOC  OBJ                  LINE     SOURCE

0138 26891C                382              MOV     WORD PTR ES:[SI].NEXT,BX        ;NEW.NEXT <- CURRENT
013B 268C5C02              383              MOV     WORD PTR ES:[SI].NEXT+2,DS
013F 8B7FC6                384              MOV     DI,WORD PTR [BX].LAST           ;AX:DI <- LAST(CURRENT)
0142 8B4708                385              MOV     AX,WORD PTR [BX].LAST+2
                           386
0145 897706                387              MOV     WORD PTR [BX].LAST,SI           ;CURRENT.LAST <- NEW
0148 8C4708                388              MOV     WORD PTR [BX].LAST+2,ES
                           389
014B 26897C06              390              MOV     WORD PTR ES:[SI].LAST,DI        ;NEW.LAST <- LAST(CURRENT)
014F 26894408              391              MOV     WORD PTR ES:[SI].LAST+2,AX
                           392
0153 8ED8                  393              MOV     DS,AX                           ;DS:DI <- LAST(CURRENT)
0155 8935                  394              MOV     WORD PTR [DI].NEXT,SI;          ;LAST(CURRENT).NEXT <- NEW
0157 8C4502                395              MOV     WORD PTR [DI].NEXT+2,ES
                           396
015A 2E8E1E0000      R     397              MOV     DS,DGRP         ;RESTORE DS
015F 9D                    398              POPF                    ;RESTORE INTERRUPT FLAG
0160 C3                    399              RET                     ;RETURN
                           400
                           401 +1   $ EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           402     ;;      TIMER$INT - ALARM CLOCK TIMER INTERRUPT SERVICE ROUTINE.
                           403     ;
                           404     ;       PARAMETERS:
                           405     ;          NONE.
                           406
                           407
                           408             PUBLIC  TIMERINT
0161                       409     TIMERINT:
0161 50                    410             PUSH    AX              ;SAVE REGISTERS
0162 53                    411             PUSH    BX
0163 51                    412             PUSH    CX
0164 52                    413             PUSH    DX
0165 56                    414             PUSH    SI
0166 57                    415             PUSH    DI
0167 1E                    416             PUSH    DS
0168 06                    417             PUSH    ES
                           418
                           419     ;       FIXED REGISTER USAGE:
                           420     ;          ES    - DGROUP
                           421     ;       SHORT TERM REGISTER USAGE:
                           422     ;          AX    - TEMPORARY
                           423     ;          DS:BX - CURRENT ALARM CONTROL BLOCK
                           424
0169 B020                  425             MOV     AL,EOI_PIC      ;ACKNOWLEDGE INTERRUPT
016B E6F0                  426             OUT     PICCMD,AL
016D 2E8E060C00      R     427             MOV     ES,DGRP
0172 26C51E0000      R     428             LDS     BX,ES:DGROUP:TIME_HEAD
0177 81FB000C        R     429             CMP     BX,OFFSET DGROUP:TIME_HEAD
017B 7507                  430             JNZ     TINT1           ;IF LIST NOT NULL
017D 8CD8                  431             MOV     AX,DS
017F 3D----          R     432             CMP     AX,DGROUP
0182 7474                  433             JZ      TINT6           ;IF LIST IS NULL, EXIT
                           434
0184 8B470A                435     TINT1:  MOV     AX,[BX].DLOW    ;TEST IF NO MORE TIME ON ALARM
0187 0B470C                436             OR      AX,[BX].DHIGH
018A 7545                  437             JNZ     TINT4           ;IF ADDITIONAL TIME ON ALARM
                           438
                           439     ;       REGISTER USAGE IN THIS SECTION:
                           440     ;          AX    - MAILBOX OFFSET
                           441     ;          DS:BX - HEAD ALARM CONTROL BLOCK IN FIRST PART
                           442     ;          DX:BX - HEAD ALARM CONTROL BLOCK IN SECOND PART
                           443     ;          DS:SI - NEXT ALARM CONTROL BLOCK IN SECOND PART
                           444
018C 8B470E                445     TINT2:  MOV     AX,[BX].RMBX    ;READ MAILBOX TO SIGNAL
018F C6470501              446             MOV     [BX].FLAG,1     ;SET FLAG TO 'TRIGGERED'
0193 8CDA                  447             MOV     DX,DS           ;SAVE BASE OF CURRENT ACB
0195 C537                  448             LDS     SI,[BX].NEXT    ;UNLINK HEAD OF LIST
0197 2689360C00      R     449             MOV     WORD PTR ES:DGROUP:TIME_HEAD,SI
019C 268C1E0200      R     450             MOV     WORD PTR ES:DGROUP:TIME_HEAD+2,DS
01A1 C744060C00      R     451             MOV     WORD PTR [SI].LAST,OFFSET DGROUP:TIME_HEAD
01A6 C74408----      R     452             MOV     WORD PTR [SI].LAST+2,DGROUP
01AB 0BC0                  453             OR      AX,AX           ;TEST IF ZERO MAILBOX ADDRESS
01AD 7410                  454             JZ      TINT3           ;IF ZERO, DO NOT SEND
01AF 50                    455             PUSH    AX              ;PUSH MAILBOX OFFSET
01B0 52                    456             PUSH    DX              ;PUSH ACB BASE
```

```
LOC  OBJ                  LINE     SOURCE

01B1 53                    457              PUSH    BX                      ;PUSH ACB OFFSET
01B2 2E8E1E0C00      R     458              MOV     DS,DGRP                 ;SET UP DS TO CONFORM TO 'COMPACT' MODEL
01B7 E80000          E     459              CALL    ISEND                   ;SEND TIMER CONTROL BLOCK TO MAILBOX
01BA 2E8E060C00      R     460              MOV     ES,DGRP                 ;RESTORE ES
                           461
                           462      ;       REGISTER USAGE RETURNS TO ORIGINAL
                           463
01BF 26C51E0C00      R     464      TINT3:  LDS     BX,ES:DGROUP:TIME_HEAD  ;RESTORE POINTER IN DS:BX
01C4 81FB0000        R     465              CMP     BX,OFFSET DGROUP:TIME_HEAD      ;TEST IF AT END OF LIST
01C8 7507                  466              JNZ     TINT4                   ;IF NOT AT END OF LIST
01CA 8CD8                  467              MOV     AX,DS
01CC 3D----          R     468              CMP     AX,DGROUP
01CF 7427                  469              JZ      TINT6                   ;IF AT END OF LIST
                           470
01D1 837F0C0C              471      TINT4:  CMP     [BX].DHIGH,0
01D5 740E                  472              JZ      TINT5
01D7 836F0C01              473              SUB     [BX].DHIGH,1
01DB B000                  474              MOV     AL,LOW 0H               ;SET TIMER ('0H' ADJUSTABLE FOR PROC. TIME)
01DD E6D0                  475              OUT     PIT_ALC,AL
01DF B000                  476              MOV     AL,HIGH 0H
01E1 E6D0                  477              OUT     PIT_ALC,AL
01E3 EB13                  478              JMP     SHORT TINT6             ;EXIT
                           479
01E5 8B470A                480      TINT5:  MOV     AX,[BX].DLOW
01E8 3D1400                481              CMP     AX,20
01EB 729F                  482              JC      TINT2                   ;IF LITTLE ADDITIONAL TIME, TRIGGER THIS ALARM
01ED E6D0                  483              OUT     PIT_ALC,AL              ;SET TIMER
01EF 8AC4                  484              MOV     AL,AH
01F1 E6D0                  485              OUT     PIT_ALC,AL
01F3 C7470A0C00            486              MOV     [BX].DLOW,0             ;CLEAR DELAY
                           487
                           488
01F8 07                    489      TINT6:  POP     ES                      ;RESTORE REGISTERS
01F9 1F                    490              POP     DS
01FA 5F                    491              POP     DI
01FB 5E                    492              POP     SI
01FC 5A                    493              POP     DX
01FD 59                    494              POP     CX
01FE 5B                    495              POP     BX
01FF 58                    496              POP     AX
0200 CF                    497              IRET
                           498
                           499 +1   $ EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           500     ;;;     CREATE$ALARM (ALARM$P) - CREATE AN ALARM CONTROL BLOCK.
                           501     ;
                           502     ;          THIS ROUTINE FORMATS RAW MEMORY IN SUCH A WAY THAT
                           503     ;       "CLEAR$ALARM"S MAY LEGALLY BE PERFORMED ON IT.
                           504     ;
                           505     ;       PARAMETER:
                           506     ;          ALARM$P - POINTER TO A 16-BYTE BLOCK OF MEMORY
                           507     ;             TO FORMAT INTO AN ALARM CONTROL BLOCK.
                           508
                           509
                           510             PUBLIC  CREATEALARM
0201                       511     CREATEALARM:
0201 5A                    512             POP     DX              ;POP RETURN ADDRESS
0202 5B                    513             POP     BX              ;POP ALARM$P
0203 07                    514             POP     ES
0204 26C6470502            515             MOV     ES:[BX].FLAG,2  ;SET FLAG TO 'UNKNOWN'
0209 FFE2                  516             JMP     DX              ;RETURN
                           517
                           518
                           519
                           520
                           521     ;;;     CHECK$ALARM (ALARM$P) - CHECK IF ALARM HAS GONE OFF.
                           522     ;
                           523     ;       PARAMETER:
                           524     ;          ALARM$P - POINTER TO AN ALARM CONTROL BLOCK FOR A PENDING ALARM.
                           525     ;
                           526     ;       RETURNS:
                           527     ;          TRUE - THE ALARM HAS GONE OFF.
                           528     ;          FALSE - THE ALARM HAS NOT GONE OFF.
                           529
                           530
                           531             PUBLIC  CHECKALARM
020B                       532     CHECKALARM:
020B 5A                    533             POP     DX              ;POP RETURN ADDRESS
020C 5B                    534             POP     BX              ;POP ALARM POINTER
020D 07                    535             POP     ES
020E 268A4705              536             MOV     AL,ES:[BX].FLAG ;READ FLAG
0212 250100                537             AND     AX,1            ;MASK OUT 'TRIGGERED' BIT
0215 FFE2                  538             JMP     DX              ;RETURN
                           539
                           540 +1  $ EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           541     ;;;     CLEAR$ALARM (ALARM$O) - CLEAR ALARM FROM KAOS STRUCTURES.
                           542     ;
                           543     ;       PARAMETER:
                           544     ;          ALARM$O - OFFSET OF AN ALARM CONTROL BLOCK.
                           545
                           546
                           547             PUBLIC  CLEARALARM
0217 B80700                548     CLEAR0: MOV     AX,7                    ;SET AN ERROR
021A 50                    549             PUSH    AX
021B E8A500                550             CALL    HALTANDCATCHFIRE
                           551
021E                       552     CLEARALARM:
021E 5A                    553             POP     DX                      ;POP RETURN ADDRESS
021F 5B                    554             POP     BX                      ;POP ALARM POINTER
0220 07                    555             POP     ES
0221 52                    556             PUSH    DX                      ;RESTORE RETURN ADDRESS
0222 9C                    557             PUSHF                           ;SAVE FLAGS
0223 FA                    558             CLI                             ;RUN WITH INTERRPUTS OFF
0224 268A4705              559             MOV     AL,ES:[BX].FLAG         ;READ FLAG
0228 26C6470502            560             MOV     ES:[BX].FLAG,2          ;CLEAR FLAG TO 'UNKNOWN'
022D 3C01                  561             CMP     AL,1                    ;CHECK 'TRIGGERED' BIT
022F 743B                  562             JZ      CLEAR3                  ;IF TRIGGERED
0231 3C02                  563             CMP     AL,2                    ;CHECK 'UNKNOWN' BIT
0233 7435                  564             JZ      CLEARX                  ;IF UNKNOWN
0235 3C00                  565             CMP     AL,0                    ;CHECK IF CURRENTLY ACTIVE
0237 75DE                  566             JNZ     CLEAR0                  ;IF AN ERROR
                           567
0239 1E                    568             PUSH    DS                      ;SAVE DGROUP
023A 26C537                569             LDS     SI,ES:[BX].NEXT         ;FIND NEXT TIMER IN LIST
023D 81FE0000        R     570             CMP     SI,OFFSET DGROUP:TIME_HEAD      ;CHECK IF NO NEXT ALARM
0241 7507                  571             JNZ     CLEAR1                  ;IF NOT AT END OF LIST
0243 8CD8                  572             MOV     AX,DS
0245 3D----          R     573             CMP     AX,DGROUP
0248 740E                  574             JZ      CLEAR2                  ;IF AT END OF LIST
024A 268B4F0A              575     CLEAR1: MOV     CX,ES:[BX].DLOW         ;ADD CURRENT DELAY TO NEXT DELAY
024E 014C0A                576             ADD     [SI].DLOW,CX
0251 268B4F0C              577             MOV     CX,ES:[BX].DHIGH
0255 114C0C                578             ADC     [SI].DHIGH,CX
0258 26C47F06              579     CLEAR2: LES     DI,ES:[BX].LAST         ;CLOSE UP LINKS
025C 268935                580             MOV     WORD PTR ES:[DI].NEXT,SI
025F 268C5D02              581             MOV     WORD PTR ES:[DI].NEXT+2,DS
0263 897C06                582             MOV     WORD PTR [SI].LAST,DI
0266 8C4408                583             MOV     WORD PTR [SI].LAST+2,ES
0269 1F                    584             POP     DS                      ;RESTORE DGROUP TO DS
                           585
026A 9D                    586     CLEARX: POPF                            ;RETURN
026B C3                    587             RET
                           588
026C 268B7F0E              589     CLEAR3: MOV     DI,ES:[BX].RMBX         ;READ MAILBOX OFFSET
0270 8CC2                  590             MOV     DX,ES
0272 0BFF                  591             OR      DI,DI                   ;CHECK FOR ZERO (NO MAILOX)
0274 74F4                  592             JZ      CLEARX                  ;IF ZERO
0276 837D0415              593             CMP     [DI].TYPEF,MBX          ;MAKE SURE IT IS A MAILBOX
027A 759B                  594             JNZ     CLEAR0                  ;BLOW UP IF NOT A MAILBOX
027C 1E                    595             PUSH    DS                      ;INITIALIZE REGS FOR LOOP
```

```
LOC  OBJ                  LINE     SOURCE

027D 07                    596             POP     ES
027E 8D7508                597             LEA     SI,[DI].MHEAD
0281 EB02                  598             JMP     SHORT CLEAR5            ;ENTER LOOP
                           599
0283 58                    600     CLEAR4: POP     AX                      ;POP LAST MESSAGE ADDRESS FROM STACK
0284 58                    601             POP     AX
0285 06                    602     CLEAR5: PUSH    ES                      ;SAVE LAST MESSAGE ADDRESS
0286 56                    603             PUSH    SI
0287 26C434                604             LES     SI,DWORD PTR ES:[SI]    ;READ NEXT MESSAGE
028A 81FEFFFF              605             CMP     SI,0FFFFH               ;CHECK IF END OF LIST
028E 7504                  606             JNZ     CLEAR6                  ;IF NOT END OF LIST
0290 58                    607             POP     AX                      ;CLEAR STACK AND EXIT
0291 58                    608             POP     AX
0292 EBD6                  609             JMP     CLEARX
0294 3BF3                  610     CLEAR6: CMP     SI,BX                   ;CHECK IF NEXT MESSAGE IS THIS ACB
0296 75EB                  611             JNZ     CLEAR4                  ;IF NOT
0298 8CC0                  612             MOV     AX,ES
029A 3BC2                  613             CMP     AX,DX
029C 75E5                  614             JNZ     CLEAR4                  ;IF NOT A MATCH, LOOP
                           615
029E 26C434                616             LES     SI,DWORD PTR ES:[SI]    ;MESSAGE FOLLOWING ACB
02A1 8CC0                  617             MOV     AX,ES
02A3 5B                    618             POP     BX                      ;MESSAGE PRECEDING ACB
02A4 07                    619             POP     ES
02A5 268937                620             MOV     WORD PTR ES:[BX],SI     ;CLOSE LINKS
02A8 26894702              621             MOV     WORD PTR ES:[BX]+2,AX
                           622
02AC 81FEFFFF              623             CMP     SI,0FFFFH               ;CHECK IF ACB WAS LAST ITEM ON MAILBOX
02B0 7506                  624             JNZ     CLEAR7                  ;IF NOT LAST ITEM
02B2 895D0C                625             MOV     WORD PTR [DI].MTAIL,BX  ;UPDATE LAST POINTER IN MAILBOX HEADER
02B5 8C450E                626             MOV     WORD PTR [DI].MTAIL+2,ES
                           627
02B8 837D0600              628     CLEAR7: CMP     [DI].COUNT,0            ;CHECK SIGN
02BC 7DAC                  629             JGE     CLEARX                  ;DO NOT INCREMENT IF >= 0
02BE FF4506                630             INC     [DI].COUNT              ;ONE MESSAGE REMOVED FROM MAILBOX
02C1 EBA7                  631             JMP     CLEARX
                           632
----                       633     CODE    ENDS
                           634
                           635
                           636 +1  $TITLE ('KAOS - ABORT PROCESSING')
                           637 +1  $eject
```

```
LOC  OBJ                  LINE     SOURCE

                           638
----                       639     DATA    SEGMENT BYTE PUBLIC 'DATA'
                           640             PUBLIC  HCF_STACK,ERRORCODE
0008 (1                    641     HCF_STACK       DW      1 DUP (?)
     ????
     )
0010 0000                  642     ERRORCODE       DW      0
----                       643     DATA    ENDS
                           644
----                       645     CODE    SEGMENT BYTE PUBLIC 'CODE'
                           646             ASSUME  CS:CGROUP,DS:DGROUP
                           647
                           648
                           649     ;;;     HALT$AND$CATCH$FIRE (ERROR$CODE) -- ABORT
                           650     ;
                           651     ;          THIS ROUTINE SAVES AWAY THE ERROR CODE,
                           652     ;       THEN DISABLES INTERRUPTS AND HALTS.
                           653     ;
                           654     ;       PARAMETER:
                           655     ;          ERROR$CODE - 16-BIT CODE FOR THE ERROR THAT OCCURED.
                           656
                           657             PUBLIC  HALTANDCATCHFIRE
02C3                       658     HALTANDCATCHFIRE:
02C3 5F                    659             POP     DI      ; CALLING ROUTINE
02C4 5E                    660             POP     SI      ; ERROR CODE
02C5 89361000        R     661             MOV     ERRORCODE,SI
02C9 0E                    662             PUSH    CS              ; SAVE ALL THE REGISTERS
02CA 57                    663             PUSH    DI
02CB 1E                    664             PUSH    DS
02CC 06                    665             PUSH    ES
02CD 50                    666             PUSH    AX
02CE 53                    667             PUSH    BX
02CF 51                    668             PUSH    CX
02D0 52                    669             PUSH    DX
02D1 89260E00        R     670             MOV     HCF_STACK,SP
02D5 FA                    671     HCF1:   CLI
02D6 B007                  672             MOV     AL,7            ; TURN ON LED
02D8 E6E2                  673             OUT     0E2H,AL
02DA F4                    674             HLT
02DB EBF8                  675             JMP     HCF1            ;LOOP IF RESTARTED BY ICE
                           676
----                       677     CODE    ENDS
                           678             END

ASSEMBLY COMPLETE, NO ERRORS FOUND
```