Mode: All
Left file: D:\Source\github\CSS422_Hardware\Disassembler\Main.L68
Right file: D:\Source\github\CSS422_Hardware\Disassembler\Output.txt

```
00001000 Starting Address                                  <>
Assembler used: EASy68K Editor/Assembler v5.16
» .01
Created On: 12/2/2019 7:22:44 PM
00000000                             1  *=====
» ========================================
» ========================================
» ========================
00000000                             2  * Titl
» e      : Disassembler
00000000                             3  * Writ
» ten by : CSS 422 Best Group
00000000                             4  *
»
00000000                             5  *   Th
» e CSS 422 Best Group is:
00000000                             6  *
»   - Howie Catlin
00000000                             7  *
»   - Kyle Dukart
00000000                             8  *
»   - Colton Sellers
00000000                             9  *
00000000                            10  * Date
»        : 01-Dec-2019
00000000                            11  *
00000000                            12  * Desc
» ription:
00000000                            13  *   Th
» is code contains a method for you to use wit
» h your Disassembler project IO, and a short
» demo of
00000000                            14  *   ho
» w to call this method in the body of the "ST
» ART" code.
00000000                            15  *=====
» ========================================
» ========================================
» ========================
00000000                            16  * Hint
» s and Tips:
00000000                            17  *   -F
» ollow the code commenting convention here fo
» r file & method headers
00000000                            18  *   -D
» on't rename this file name (Main.X68)
00000000                            19  *   -D
» on't reorg the start address ($1000)
00000000                            20  *   -D
» on't rename or edit the IO method I've provi
» ded you with here
00000000                            21  *   -D
```

```
» on't rename the config.cfg file or change th
» e file format:
00000000                            22  *    <
» Long_StartAddress>\CR\LF
00000000                            23  *    <
» Long_EndAddress>\CR\LF<EOF>
00000000                            24  *-----
» ----------------------------------------
» -------------------------
00001000                            25      OR
» G    $1000    *Don't change this; see hints
» and tips above
00001000                            26  *-----
» ----------------------------------------
» -------------------------
00001000                            27
00001000                            28      IN
» CLUDE     'Disassembler_Setup.X68'
00001000                            29
00001000                            30
00001000                            31
00001000                            32      ;
» Configure the stack to exist (current defaul
» t: $0070000)
00001000  4FF9 00070000             33      LE  00001000        LEA             $00070000, A7
» A         STACK_LOCATION,SP
00001006                            34
00001006                            35
00001006                            36      ;
» Read in from config.cfg
00001006  43F9 000021EF             37      LE  00001006        LEA             $000021EF, A1
» A         inFile, A1       ;in file name
0000100C  303C 0033                 38      MO  0000100C        MOVE.W  #$0033, D0
» VE        #51, D0          ;open in file
00001010  4E4F                      39      TR
» AP        #15                                 00001010        JSR             A7
00001012                            40
00001012                            41      ;S
» etup for Read
00001012  343C 0008                 42      MO  00001012        MOVE.W  #$0008, D2
» VE        #8, D2        ;# bytes to read
00001016  303C 0035                 43      MO  00001016        MOVE.W  #$0035, D0
» VE        #53,D0                  ;read from
»  file
0000101A                            44
0000101A                            45      ;
» Read & store begin address in BEGIN_ADDRESS_
» STR
0000101A  43F9 000021FA             46      LE  0000101A        LEA             $000021FA, A1
» A         BEGIN_ADDRESS_STR, A1
00001020  4E4F                      47      TR  00001020        JSR             A7
» AP        #15
```

```
00001022                            48
00001022                            49      ;
» Convert the beginning address to HEX and sto
» re the value in A3
00001022  4EB9 00002174             50      JS      00001022        JSR             $00002174
» R         AsciiToHex
00001028  2647                      51      MO      00001028        MOVEA.L D7, A3
» VE.L      D7, A3
0000102A                            52
0000102A                            53      ;
» Move File position to where the next line sh
» ould be
0000102A  343C 000A                 54      MO      0000102A        MOVE.W  #$000A, D2
» VE        #10,D2
0000102E  303C 0037                 55      MO      0000102E        MOVE.W  #$0037, D0
» VE        #55,D0
00001032  4E4F                      56      TR
» AP        #15
00001034                            57              00001032        JSR             A7
00001034                            58      ;S
» etup for Read again
00001034  343C 0008                 59      MO      00001034        MOVE.W  #$0008, D2
» VE        #8, D2        ;# bytes to read
00001038  303C 0035                 60      MO      00001038        MOVE.W  #$0035, D0
» VE        #53,D0                  ;read from
»  file
0000103C                            61
0000103C                            62      ;
» Read & store end address in END_ADDRESS
0000103C  43F9 00002202             63      LE      0000103C        LEA             $00002202, A1
» A         END_ADDRESS_STR, A1
00001042  4E4F                      64      TR
» AP        #15
00001044                            65              00001042        JSR             A7
00001044                            66      ;
» Convert the end address to hex and store in
» A4
00001044  4EB9 00002174             67      JS      00001044        JSR             $00002174
» R         AsciiToHex
0000104A  2847                      68      MO      0000104A        MOVEA.L D7, A4
» VE.L      D7, A4
0000104C                            69
0000104C                            70      ;
» Clear Current String Length
0000104C  43F9 0000220A             71      LE      0000104C        LEA             $0000220A, A1
» A         CURRENT_STR_LENGTH, A1
00001052  4211                      72      CL      00001052        NEG.B   (A1)
» R.B       (A1)
00001054                            73
00001054                            74
00001054                            75
00001054                            76
```

```
00001054                            77
00001054                            78  ------
» -------------- end include ----------------
» ---
00001054                            79
00001054                            80
00001054                            81  MAIN_L
» OOP:
00001054                            82      ;
» Append String with address
00001054  2C0B                      83      MO
» VE.L    A3, D6
00001056  4EB9 000020C4             84      JS
» R     PrintASCIILong
0000105C  4286                      85      CL
» R.L     D6
0000105E                            86
0000105E  4EB9 00001072             87      JS
» R     Get_Next_Word_D6
00001064                            88
00001064  4EB9 00001084             89      JS
» R     DisassembleOpcode   ; disassemble and
» print word in D6
0000106A                            90
0000106A                            91      ;
» Compare current address to end address
0000106A  B7CC                      92      CM
» P.L       A4,A3           ; Are we done yet?
0000106C  6FE6                      93      BL
» E         MAIN_LOOP       ; Branch if less t
» han or equal too MAIN_LOOP
0000106E                            94
0000106E                            95      ;
» end (print SIMHALT?  close file?)
0000106E                            96
0000106E  FFFF FFFF                 97      SI
» MHALT
00001072                            98
00001072                            99  ******
» ****************************************
» ******************************
00001072                           100  * Meth
» od Name: Get_Next_Word_D6
00001072                           101  * Desc
» ription: This method collects the next word
» at A3 and increments the pointer
00001072                           102  *
»       to the next word.
00001072                           103  * Prec
» onditions: (A3) points to the current addres
» s we are looking at
00001072                           104  * Post
```

```
00001054        MOVE.L  A3, D6

00001056        JSR             $000020C4


0000105C        NEG.L   D6
0000105E        JSR             $00001072

00001064        JSR             $00001084


0000106A        EOR.B   D3, A4


0000106C        DATA    6FE6
0000106E        SIMHALT

00001070        SIMHALT
```

Left	Right
» conditions: D6 contains the new word	
00001072 105 ******	
» **	
» *******************************	
00001072 106 Get_Ne	
» xt_Word_D6:	
00001072 107 ;	
» read word into D6 from current address	
00001072 3C13 108 MO	00001072 MOVE.W (A3), D6
» VE.W (A3), D6	
00001074 109	00001074 ADDQ.W #2, A3
00001074 110 ;	
» Increment Address	
00001074 544B 111 AD	
» D #2,A3 * Increment pointer by	
» two bytes	
00001076 112	
00001076 4E75 113 RT	00001076 RTS
» S	
00001078 114	
00001078 115 ******	
» **	
» *******************************	
00001078 116 * Meth	
» od Name: Get_Next_Word_D7	
00001078 117 * Desc	
» ription: This method collects the next word	
» at A3 and increments the pointer	
00001078 118 *	
» to the next word.	
00001078 119 * Prec	
» onditions: (A3) points to the current addres	
» s we are looking at	
00001078 120 * Post	
» conditions: D7 contains the new word	
00001078 121 ******	
» **	
» *******************************	
00001078 122 Get_Ne	
» xt_Word_D7:	
00001078 123 ;	
» read word into D7 from current address	
00001078 3E13 124 MO	00001078 MOVE.W (A3), D7
» VE.W (A3), D7	
0000107A 125	0000107A ADDQ.W #2, A3
0000107A 126 ;	
» Increment Address	
0000107A 544B 127 AD	
» D #2,A3 * Increment pointer by	
» two bytes	
0000107C 128	
0000107C 4E75 129 RT	0000107C RTS

```
» S
0000107E                            130
0000107E                            131  ******
» ********************************************
» ******************************
0000107E                            132  * Meth
» od Name: Get_Next_Long_D7
0000107E                            133  * Desc
» ription: This method collects the next word
» at A3 and increments the pointer
0000107E                            134  *
»       to the next word.
0000107E                            135  * Prec
» onditions: (A3) points to the current addres
» s we are looking at
0000107E                            136  * Post
» conditions: D7 contains the new word
0000107E                            137  ******
» ********************************************
» ******************************
0000107E                            138  Get_Ne
» xt_Long_D7:
0000107E                            139      ;
» read long into D7 from current address
0000107E  2E13                      140      MO
» VE.L      (A3), D7
00001080                            141
00001080                            142      ;
» Increment Address
00001080  584B                      143      AD
» D         #4,A3        * Increment pointer b
» y four bytes
00001082                            144
00001082  4E75                      145      RT
» S
00001084                            146

00001084                            147
00001084                            148  ******
» ********************************************
» ******************************
00001084                            149  * Meth
» od Name: DisassembleOpcode
00001084                            150  * Desc
» ription: This method will determine which op
» code is in the current word, and
00001084                            151  *   wr
» ite its ASCII assembly code equivalent into
» (A1). It will then call TrapTask13
00001084                            152  *   to
»  print the ASCII assembly code line.
00001084                            153  *
```

```
0000107E        MOVE.L  (A3), D7
00001080        ADDQ.W  #4, A3
00001082        RTS
00001084        MOVEM   .L      A2/1//D6/5//D4
» /3//1//D0/, -(A7)
```

```
00001084                            154  * Prec
» onditions:  D6 contains word-length opcode
00001084                            155  * Post
» conditions: (A1) points to ASCII assembly co
» de ending in 0
00001084                            156  ******
» ****************************************
» ****************************
00001084                            157  Disass
» embleOpcode:
00001084                            158      IN
» CLUDE     'Determine_Opcode.X68'
00001084                            159
00001084                            160  Opcode
» :
00001084  48E7 3808                 161      MO
» VEM.L     D2-D4/A4, -(SP)          ; MOVEM al
» l registers used
00001088                            162
00001088  3406                      163      MO
» VE.W      D6,D2                    ; copy cur
» rent word (D6) into opcode mask register (D2
» )
0000108A  C47C F000                 164      AN
» D         #MASK_OPCODE,D2          ; apply ma
» sk to first 4 bits
0000108E                            165
0000108E  B47C 9000                 166      CM
» P.W       #$9000,D2                ; compare
» with 1001
00001092  6700 023E                 167      BE
» Q         Opcode_SUB               ; if equal
»  jump to SUB
00001096                            168
00001096  B47C 5000                 169      CM
» P.W       #$5000,D2                ; compare
» with 1010
0000109A  6600 0014                 170      BN
» E         SKIP_5000                ; if not e
» qual then skip
0000109E  3606                      171      MO
» VE.W      D6, D3                   ; move cur
» rent word into working register (D3)
000010A0  C67C 0100                 172      AN
» D.W       #MASK_8, D3              ; mask bit
»  8
000010A4  B67C 0000                 173      CM
» P.W       #$0000, D3               ; compare
» with 0
000010A8  6600 0274                 174      BN
» E         Opcode_SUBQ              ; if not e
» qual jump to SUBQ
```

```
00001088        MOVE.W  D6, D2
0000108A        MULS    #$F000, D2
0000108E        CMP.W   #$9000, D2
00001092        DATA    6700
00001094        ORI.B   #$B47C, $50006600
0000109C        ORI.B   #$3606, (A4)
000010A0        MULS    #$0100, D3
000010A4        CMP.W   #$0000, D3
```

```
000010AC  6000 0298                175      BR
» A         Opcode_ADDQ             ; if equal
»  jump to ADDQ
000010B0                           176  Skip_5
» 000:
000010B0                           177
```

```
000010A8        DATA    6600
000010AA        ORI.W   #$6000, #$0298
```

```
000010B0  B47C C000                178      CM
» P.W       #$C000,D2               ; compare
» with 1100
```

```
000010B0        CMP.W   #$C000, D2
```

```
000010B4  6700 02B8                179      BE
» Q         Opcode_MULS             ; if equal
»  jump to MULS
000010B8                           180
```

```
000010B4        DATA    6700
```

```
000010B8  B47C 0000                181      CM
» P.W       #$0000,D2               ; compare
» with 0000
```

```
000010B6        ORI.L   #$B47C0000, $6600
```

```
000010BC  6600 004C                182      BN
» E         Skip_0000               ; if not e
» qual then skip
```

```
000010BE        ORI.W   #$3606, A4
```

```
000010C0  3606                     183      MO
» VE.W      D6,D3                   ; move cur
» rent word into working register (D3)
```

```
000010C2        MULS    #$0DC0, D3
```

```
000010C2  C67C 0DC0                184      AN
» D         #MASK_1110876,D3        ; mask bit
» s 11,10,8,7,6
```

```
000010C6        CMP.W   #$0880, D3
```

```
000010C6                           185
000010C6  B67C 0880                186      CM
» P.W       #$0880,D3               ; compare
» with 10010
```

```
000010CA        DATA    6700
```

```
000010CA  6700 04CC                187      BE
» Q         BCLR_I                  ; if equal
»  jump to BCLR
000010CE  3606                     188      MO
» VE.W      D6,D3                   ; reset cu
» rrent word in D3
000010D0  C67C 0100                189      AN
» D         #MASK_8,D3              ; mask bit
»  8
000010D4                           190
```

```
000010CC        DATA    04CC
000010CE        MOVE.W  D6, D3
000010D0        MULS    #$0100, D3
```

```
000010D4  B67C 0100                191      CM
» P.W       #$0100,D3               ; compare
» with 1
```

```
000010D4        CMP.W   #$0100, D3
```

```
000010D8  6700 049C                192      BE
» Q         Opcode_BCLR             ; if equal
»  jump to BCLR
```

```
000010D8        DATA    6700
```

```
000010DC  3606                     193      MO
» VE.W      D6,D3                   ; reset cu
» rrent word in D3
000010DE  C67C 0800                194      AN
```

```
» D         #MASK_11,D3              ; mask bit
»  11
000010E2                            195
```

```
000010DA        DATA    049C
000010DC        MOVE.W  D6, D3
000010DE        MULS    #$0800, D3
```

```
000010E2  B67C 0800                 196       CM
» P.W       #$0800,D3                ; compare
» with 1
000010E6  6600 0014                 197       BN
» E         Skip_CMPI_EORI           ; if not e
» qual then skip
000010EA  3606                      198       MO
» VE.W      D6, D3                   ; reset cu
» rrent word in D3
000010EC  C67C 0400                 199       AN
» D.W       #MASK_10, D3             ; mask bit
»  10
000010F0  B67C 0000                 200       CM
» P.W       #$0000, D3               ; compare
» with 0
000010F4  6700 03E0                 201       BE
» Q         Opcode_EORI              ; if equal
»  jump to EORI
000010F8  6000 033C                 202       BR
» A         Opcode_CMPI              ; if not e
» qual jump to CMPI
000010FC                            203  Skip_C
» MPI_EORI:
000010FC                            204
000010FC  3606                      205       MO
» VE.W      D6,D3                    ; reset cu
» rrent word in D3
000010FE  C67C 0D00                 206       AN
» D         #MASK_11108,D3           ; mask bit
»  11, 10,8
```

```
000010E2        DATA    6600

000010E8        ORI.B   #$3606, (A4)

000010EC        MULS    #$0400, D3

000010F0        CMP.W   #$0000, D3

000010F4        DATA    6700

000010F6        BCLR    D1, -(A0)
```

```
00001102                            207
```

```
000010F8        BRA     033C
000010FC        MOVE.W  D6, D3

000010FE        MULS    #$0D00, D3
```

```
00001102        CMP.W   #$0000, D3
00001106        DATA    6700
00001108        ORI.L   #$B47CD000, A6
0000110E        DATA    6600
00001110        ORI.B   #$3606, (A4)
00001114        MULS    #$00C0, D3
00001118        CMP.W   #$00C0, D3
```

```
00001102  B67C 0000                 208       CM
» P.W       #$0000,D3                ; compare
» with 000
00001106  6700 028E                 209       BE
» Q         Opcode_ORI               ; if equal
»  jump to ORI
0000110A                            210  Skip_0
» 000:
0000110A                            211
0000110A  B47C D000                 212       CM
» P.W       #$D000,D2                ; compare
```

```
0000111C        DATA    6700

0000111E        DATA    04B2
```

```
» with 1101
0000110E  6600 0014                213      BN
» E         Skip_1101               ; if not e
» qual then skip
00001112  3606                     214      MO
» VE.W      D6,D3                   ; move cur
» rent word into working register (D3)
00001114  C67C 00C0                215      AN
» D         #MASK_76,D3             ; mask bit
» s 7, 6
00001118                           216
00001118  B67C 00C0                217      CM
» P.W       #$00C0,D3               ; compare
» with 11
0000111C  6700 04B2                218      BE
» Q         Opcode_ADDA             ; if equal
»  jump to ADDA
00001120  6600 04D6                219      BN
» E         Opcode_ADD              ; if not e
» qual jump to ADD
00001124                           220
00001124                           221  Skip_1
» 101:
00001124  B47C 8000                222      CM
» P.W       #$8000,D2               ; compare
» with 1000
00001128  6600 0014                223      BN
» E         Skip_1000               ; if not e
» qual then skip
0000112C  3606                     224      MO
» VE.W      D6,D3                   ; move cur
» rent word into working register (D3)
0000112E  C67C 00C0                225      AN
» D         #MASK_76,D3             ; mask bit
» s 7, 6
00001132                           226


00001132  B67C 00C0                227      CM
» P.W       #$00C0,D3               ; compare
» with 11
00001136  6700 050C                228      BE
» Q         Opcode_DIVS             ; if equal
»  jump to DIVS
0000113A  6600 0530                229      BN
» E         Opcode_OR               ; if not e
» qual jump to OR
0000113E                           230
0000113E                           231  Skip_1
» 000:
0000113E  B47C B000                232      CM
» P.W       #$B000,D2               ; compare
```

```
00001120        DATA    6600


00001122        DATA    04D6


00001124        CMP.W   #$8000, D2


00001128        DATA    6600
0000112A        ORI.B   #$3606, (A4)
0000112E        MULS    #$00C0, D3
00001132        CMP.W   #$00C0, D3


00001136        DATA    6700
```

```
» with 1011
00001142  6600 0014                233      BN
» E         Skip_1011               ; if not e
» qual then skip
00001146  3606                     234      MO
» VE.W      D6,D3                   ; move cur
» rent word into working register (D3)
00001148  C67C 0100                235      AN
» D         #MASK_8,D3              ; mask bit
»  8
0000114C                           236
```

```
00001138        BCLR    D2, A4
0000113A        DATA    6600
0000113C        BCLR    D2, $B47C
00001140        CMP.B   D0, D0
00001142        DATA    6600
00001144        ORI.B   #$3606, (A4)
00001148        MULS    #$0100, D3
```

```
0000114C  B67C 0000                237      CM
» P.W       #$0000,D3               ; compare
» with 0
00001150  6700 0566                238      BE
» Q         Opcode_CMP              ; if equal
»  jump to CMP
00001154  6600 058A                239      BN
» E         Opcode_EOR              ; if not e
» qual jump to EOR
00001158                           240
00001158                           241  Skip_1
» 011:
00001158  B47C 1000                242      CM
» P.W       #$1000,D2               ; compare
» with 0001
0000115C  6600 0014                243      BN
» E         Skip_0001               ; if not e
» qual then skip
00001160  3606                     244      MO
» VE.W      D6,D3                   ; move cur
» rent word into working register (D3)
00001162  C67C 01C0                245      AN
» D         #MASK_876,D3            ; mask bit
» s 8, 7, 6
00001166                           246
```

```
0000114C        CMP.W   #$0000, D3

00001150        DATA    6700

00001152        BCLR    D2, -(A6)

00001154        DATA    6600
00001156        BCLR    D2, A2

00001158        CMP.W   #$1000, D2
```

```
0000115C        DATA    6600
0000115E        ORI.B   #$3606, (A4)
00001162        MULS    #$01C0, D3
00001166        CMP.W   #$0040, D3
```

```
00001166  B67C 0040                247      CM
» P.W       #$0040,D3               ; compare
» with 001
0000116A  6700 059C                248      BE
» Q         MOVEA_B                 ; if equal
»  jump to MOVEA_B
0000116E  6600 05C0                249      BN
» E         MOVE_B                  ; if not e
```

```
0000116A        DATA    6700

0000116C        BCLR    D2, (A4)+
```

```
» qual jump to MOVE_B

00001172                            250
00001172                            251  Skip_0
» 001:
00001172  B47C 3000                 252      CM
» P.W       #$3000,D2                ; compare
» with 0011
00001176  6600 0014                 253      BN
» E         Skip_0011                ; if not e
» qual then skip
0000117A  3606                      254      MO
» VE.W      D6,D3                    ; move cur
» rent word into working register (D3)
0000117C  C67C 01C0                 255      AN
» D         #MASK_876,D3             ; mask bit
» s 8, 7, 6
00001180                            256


00001180  B67C 0040                 257      CM
» P.W       #$0040,D3                ; compare
» with 001
00001184  6700 05D2                 258      BE
» Q         MOVEA_W                  ; if equal
»  jump to MOVEA_W
00001188  6600 05F6                 259      BN
» E         MOVE_W                   ; if not e
» qual jump to MOVE_W
0000118C                            260
0000118C                            261  Skip_0
» 011:
0000118C  B47C 2000                 262      CM
» P.W       #$2000,D2                ; compare
» with 0010
00001190  6600 0014                 263      BN
» E         Skip_0010                ; if not e
» qual then skip
00001194  3606                      264      MO
» VE.W      D6,D3                    ; move cur
» rent word into working register (D3)
00001196  C67C 01C0                 265      AN
» D         #MASK_876,D3             ; mask bit
» s 8, 7, 6
0000119A                            266


0000119A  B67C 0040                 267      CM
» P.W       #$0040,D3                ; compare
» with 001
0000119E  6700 0608                 268      BE
» Q         MOVEA_L                  ; if equal
```

```
0000116E        DATA    6600
00001170        BCLR    D2, D0


00001172        CMP.W   #$3000, D2


00001176        DATA    6600
00001178        ORI.B   #$3606, (A4)
0000117C        MULS    #$01C0, D3
00001180        CMP.W   #$0040, D3


00001184        DATA    6700


00001186        BCLR    D2, (A2)
00001188        DATA    6600

0000118A        BCLR    D2, $B47C2000


00001190        DATA    6600
00001192        ORI.B   #$3606, (A4)
00001196        MULS    #$01C0, D3
0000119A        CMP.W   #$0040, D3
```

```
»  jump to MOVEA_L
000011A2  6600 062C                269      BN
» E         MOVE_L                  ; if not e
» qual jump to MOVE_L
000011A6                           270
000011A6                           271  Skip_0
» 010:
```

```

000011A6  B47C E000                272      CM
» P.W       #$E000,D2               ; compare
» with 1110
000011AA  6600 0080                273      BN
» E         Skip_1110               ; if not e
» qual then skip
000011AE  3606                     274      MO
» VE.W      D6,D3                   ; move cur
» rent word into working register (D3)
000011B0  C67C 00C0                275      AN
» D         #MASK_76,D3             ; mask bit
» s 7, 6
000011B4                           276
```

```

000011B4  B67C 00C0                277      CM
» P.W       #$00C0,D3               ; compare
» with 11
000011B8  6600 003C                278      BN
» E         Branch_0110             ; branch t
» o second half of check if not equal
000011BC  3606                     279      MO
» VE.W      D6,D3                   ; reset th
» e word value in D3
000011BE  C67C 0700                280      AN
» D         #MASK_1098,D3           ; mask bit
» s 10, 9, 8
000011C2                           281
000011C2  B67C 0700                282      CM
» P.W       #$0700,D3               ; compare
» with 110
000011C6  6700 0630                283      BE
» Q         Opcode_ROL              ; if equal
»  jump to ROL
000011CA  B67C 0600                284      CM
» P.W       #$0600,D3               ; compare
» with 111
000011CE  6700 063E                285      BE
» Q         Opcode_ROR              ; if equal
»  jump to ROR
000011D2                           286
000011D2  B67C 0300                287      CM
```

```

0000119E        DATA    6700
000011A0        DATA    0608

000011A2        DATA    6600
000011A4        DATA    062C
000011A6        CMP.W   #$E000, D2

000011AA        DATA    6600

000011AC        ORI.L   #$3606C67C, D0

```

```
000011B2        ORI.B   #$B67C, D0
000011B6        ORI.B   #$6600, D0
000011BA        ORI.B   #$3606, #000C
000011C0        BCLR    D3, D0
000011C2        CMP.W   #$0700, D3

000011C6        DATA    6700

```

```

000011C8        DATA    0630

```

```
000011CA        CMP.W   #$0600, D3

000011CE        DATA    6700

000011D0        DATA    063E
000011D2        CMP.W   #$0300, D3
```

```
» P.W       #$0300,D3               ; compare
» with 011
000011D6  6700 064C                 288       BE
» Q         Opcode_LSL              ; if equal
»  jump to LSL
000011DA                            289
000011DA  B67C 0200                 290       CM
» P.W       #$0200,D3               ; compare
» with 010
000011DE  6700 065A                 291       BE
» Q         Opcode_LSR              ; if equal
»  jump to LSR
000011E2                            292
000011E2  B67C 0100                 293       CM
» P.W       #$0100,D3               ; compare
» with 001
000011E6  6700 0668                 294       BE
» Q         Opcode_ASL              ; if equal
»  jump to ASL
000011EA                            295
000011EA  B67C 0000                 296       CM
» P.W       #$0000,D3               ; compare
» with 000
000011EE  6700 0676                 297       BE
» Q         Opcode_ASR              ; if equal
»  jump to ASR
000011F2  6000 0038                 298       BR
» A         Skip_1110               ; no valid
»  opcodes found, skip ahead
000011F6                            299
000011F6                            300  Branch
» _0110:
000011F6  3606                      301       MO
» VE.W      D6,D3                   ; reset th
» e word value in D3
000011F8  C67C 0118                 302       AN
» D         #MASK_843,D3            ; mask bit
» s 8, 4, 3
000011FC                            303


000011FC  B67C 0118                 304       CM
» P.W       #$0118,D3               ; compare
» with 111
00001200  6700 067A                 305       BE
» Q         ROL_I                   ; if equal
»  jump to ROL
00001204                            306
00001204  B67C 0018                 307       CM
» P.W       #$0018,D3               ; compare
» with 011
00001208  6700 06BE                 308       BE
```

```
000011D6        DATA    6700

000011D8        DATA    064C
000011DA        CMP.W   #$0200, D3

000011DE        DATA    6700

000011E0        DATA    065A
000011E2        CMP.W   #$0100, D3

000011E6        DATA    6700

000011E8        DATA    0668
000011EA        CMP.W   #$0000, D3

000011EE        DATA    6700


000011F0        DATA    0676


000011F2        BRA     0038
000011F6        MOVE.W  D6, D3
000011F8        MULS    #$0118, D3
000011FC        CMP.W   #$0118, D3

00001200        DATA    6700

00001202        DATA    067A
00001204        CMP.W   #$0018, D3

00001208        DATA    6700
```

```
» Q         ROR_I                    ; if equal
»  jump to ROR
0000120C                             309
0000120C  B67C 0108                  310       CM
» P.W       #$0108,D3                ; compare
» with 101
00001210  6700 0702                  311       BE
» Q         LSL_I                    ; if equal
»  jump to LSL
00001214                             312
00001214  B67C 0008                  313       CM
» P.W       #$0008,D3                ; compare
» with 001
00001218  6700 0746                  314       BE
» Q         LSR_I                    ; if equal
»  jump to LSR
0000121C                             315
0000121C  B67C 0100                  316       CM
» P.W       #$0100,D3                ; compare
» with 100
00001220  6700 078A                  317       BE
» Q         ASL_I                    ; if equal
»  jump to ASL
00001224                             318
00001224  B67C 0000                  319       CM
» P.W       #$0000,D3                ; compare
» with 000
00001228  6700 07CE                  320       BE
» Q         ASR_I                    ; if equal
»  jump to ASR
0000122C                             321
0000122C                             322  Skip_1
» 110:
0000122C  B47C 6000                  323       CM
» P.W       #$6000,D2                ; compare
» with 0110
00001230  6600 0030                  324       BN
» E         Skip_0110                ; if not e
» qual then skip
00001234  3606                       325       MO
» VE.W      D6,D3                    ; move cur
» rent word into working register (D3)
00001236  C67C 0F00                  326       AN
» D         #MASK_111098,D3          ; mask bit
» s 11, 10, 9, 8
0000123A                             327
»
0000123A  B67C 0000                  328       CM
» P.W       #$0000,D3                ; compare
» with 0000
0000123E  6700 0804                  329       BE
» Q         Opcode_BRA               ; if equal
```

```
0000120A        DATA    06BE
0000120C        CMP.W   #$0108, D3
00001210        DATA    6700
00001212        BCLR    D3, D2
00001214        CMP.W   #$0008, D3
00001218        DATA    6700
0000121A        BCLR    D3, D6
0000121C        CMP.W   #$0100, D3
00001220        DATA    6700
00001222        BCLR    D3, A2
00001224        CMP.W   #$0000, D3
00001228        DATA    6700
```

» jump to BRA	
00001242 330	0000122A BCLR D3, A6
00001242 B67C 0500 331 CM	0000122C CMP.W #$6000, D2
» P.W #$0500,D3 ; compare	
» with 0101	
00001246 6700 0812 332 BE	
» Q Opcode_BCS ; if equal	
» jump to BCS	
0000124A 333	00001230 DATA 6600
0000124A B67C 0C00 334 CM	
» P.W #$0C00,D3 ; compare	
» with 1100	
0000124E 6700 0820 335 BE	
» Q Opcode_BGE ; if equal	
» jump to BGE	
00001252 336	00001232 ORI.B #$3606, $C67C
00001252 B67C 0D00 337 CM	
» P.W #$0D00,D3 ; compare	
» with 1101	
00001256 6700 082E 338 BE	
» Q Opcode_BLT ; if equal	
» jump to BLT	
0000125A 339	00001238 BCLR D7, D0
0000125A B67C 0800 340 CM	0000123A CMP.W #$0000, D3
» P.W #$0800,D3 ; compare	
» with 1000	
0000125E 6700 083C 341 BE	
» Q Opcode_BVC ; if equal	
» jump to BVC	
00001262 342	0000123E DATA 6700
00001262 343 Skip_0	00001240 EORI.B #$B67C, D4
» 110:	
00001262 B47C 4000 344 CM	
» P.W #$4000,D2 ; compare	
» with 0100	
00001266 6600 0048 345 BN	
» E Skip_0100 ; if not e	
» qual then skip	
0000126A 3606 346 MO	
» VE.W D6,D3 ; move cur	
» rent word into working register (D3)	
0000126C 347	00001244 BCLR D2, D0
0000126C B67C 4E71 348 CM	
» P.W #$4E71,D3 ; check ag	
» ainst constant NOP code	
00001270 6700 0840 349 BE	
» Q Opcode_NOP ; if equal	
» jump to NOP	
00001274 350	00001246 DATA 6700
00001274 B67C 4AFC 351 CM	00001248 EORI.B #$B67C, (A2)
» P.W #$4AFC,D3 ; check ag	
» ainst constant ILLEGAL code	

```
00001278  6700 0848                352      BE
» Q         Opcode_ILLEGAL          ; if equal
»  jump to ILLEGAL

0000127C                           353
0000127C  B67C 4E75                354      CM
» P.W       #$4E75,D3               ; check ag
» ainst constant RTS code
00001280  6700 0850                355      BE
» Q         Opcode_RTS              ; if equal
»  jump to RTS
00001284  C67C 0100                356      AN
» D         #MASK_8,D3              ; mask bit
»  8
00001288                           357
00001288                           358
00001288  B67C 0100                359      CM
» P         #$0100,D3               ; compare
» with 1
0000128C  6700 0854                360      BE
» Q         Opcode_LEA              ; if equal
»  jump to LEA
00001290  3606                     361      MO
» VE.W      D6,D3                   ; refresh
» current word into D3
00001292  C67C 0800                362      AN
» D         #MASK_11,D3             ; mask bit
»  11
00001296                           363
00001296  B67C 0000                364      CM
» P         #$0000,D3               ; compare
» with 0
0000129A  6700 0868                365      BE
» Q         Opcode_NEG              ; if equal
»  jump to NEG
0000129E  3606                     366      MO
» VE.W      D6,D3                   ; refresh
» current word into D3
000012A0  C67C 0200                367      AN
» D         #MASK_9,D3              ; mask bit
»  9
000012A4                           368
000012A4  B67C 0000                369      CM
» P         #$0000,D3               ; compare
» with 0
000012A8  6700 0876                370      BE
» Q         Opcode_MOVEM            ; if equal
»  jump to MOVEM
000012AC  6600 08BE                371      BN
» E         Opcode_JSR              ; if not e
» qual jump to JSR
000012B0                           372
```

```
0000124C        CMPI.B  #$6700, D0

00001250        EORI.B  #$B67C, -(A0)
00001254        BCLR    D6, D0

00001256        DATA    6700
00001258        EORI.B  #$B67C, $08006700
00001260        EORI.B  #$B47C, #0000

00001266        DATA    6600

00001268        ORI.W   #$3606, A0
0000126C        CMP.W   #$4E71, D3

00001270        DATA    6700
```

```
000012B0                          373  Skip_0
» 100:
000012B0  BC7C FFFF               374      CM
» P.W       #$FFFF,D6              ; compare
» with FFFF
000012B4  6700 08CC               375      BE
» Q         Opcode_SIMHALT         ; if equal
»  jump to SIMHALT
000012B8                          376
000012B8  6000 0002               377      BR
» A         Opcode_DATA            ; if no op
» codes found jump to DATA
000012BC                          378

000012BC                          379
000012BC                          380
000012BC                          381  ------
» -------------- end include -----------------
» ---
000012BC                          382      IN
» CLUDE     'Breakdown_Opcode.X68'
000012BC                          383
000012BC                          384
000012BC                          385  Opcode
» _DATA:
000012BC  45F9 00002329           386      LE
» A         STR_DATA, A2
000012C2  4EB9 0000203A           387      JS
» R         AppendOutput           ; write th
» e current string in A2 to the output string
000012C8  4EB9 00002066           388      JS
» R         PrintASCIIWord         ; print th
» e input word to output
000012CE  6000 08C2               389      BR
» A         Opcode_Finish
000012D2                          390

000012D2                          391
000012D2                          392  Opcode
» _SUB:
000012D2  45F9 000023F4           393      LE
```

```
00001272        EORI.W  #$B67C, D0

00001276        ILLEGAL

00001278        DATA    6700
0000127A        EORI.W  #$B67C, A0
0000127E        RTS
00001280        DATA    6700
00001282        EORI.W  #$C67C, (A0)

00001286        BCLR    D0, D0
00001288        CMP.W   #$0100, D3
0000128C        DATA    6700

0000128E        EORI.W  #$3606, (A4)

00001292        MULS    #$0800, D3
00001296        CMP.W   #$0000, D3
0000129A        DATA    6700
0000129C        EORI.W  #$3606, $C67C
000012A2        ORI.B   #$B67C, D0
000012A6        ORI.B   #$6700, D0
000012AA        EORI.W  #$6600, $08BEBC7C
000012B2        SIMHALT
000012B4        DATA    6700
000012B6        EORI.B  #$6000, A4
000012BA        ORI.B   #$45F9, D2
000012BE        ORI.B   #$2329, D0

000012C2        JSR             $0000203A
000012C8        JSR             $00002066
000012CE        BRA     08C2
000012D2        LEA             $000023F4, A2
```

```
» A         STR_SUB, A2
000012D8  4EB9 0000203A            394      JS
» R         AppendOutput            ; write th
» e current string in A2 to the output string
000012DE  4EB9 00001D6E            395      JS
» R         Opcode_AppendSizeSuffix
000012E4  3606                     396      MO
» VE.W      D6,D3                   ; move wor
» d value into D3
000012E6  C67C 0100                397      AN
» D         #MASK_8, D3             ; mask bit
»  8
000012EA  B67C 0000                398      CM
» P.W       #$0000,D3              ; compare wi
» th 0
000012EE  6600 0018                399      BN
» E         SUB_EA_DN_EA
000012F2                           400  SUB_DN
» _EA_DN:                  ; case  Dn + <ea>
» -> Dn
000012F2  4EB9 00001BB2            401      JS
» R         EA_AppendMXn
000012F8  4EB9 00001EC6            402      JS
» R         AppendComma
000012FE  4EB9 00001E6E            403      JS
» R         Dn
00001304  6000 088C                404      BR
» A         Opcode_Finish
00001308                           405  SUB_EA
» _DN_EA:                  ; case <ea> + Dn -
» > <ea>
00001308  4EB9 00001E6E            406      JS
» R         Dn
0000130E  4EB9 00001EC6            407      JS
» R         AppendComma
00001314  4EB9 00001BB2            408      JS
» R         EA_AppendMXn
0000131A  6000 0876                409      BR
» A         Opcode_Finish
0000131E                           410
0000131E                           411  Opcode
» _SUBQ:
0000131E  45F9 000023F9            412      LE
» A         STR_SUBQ, A2
00001324  4EB9 0000203A            413      JS
» R         AppendOutput            ; write SU
» BQ to the output string
0000132A  4EB9 00001D6E            414      JS
» R         Opcode_AppendSizeSuffix
00001330  4EB9 00001ED4            415      JS
» R         Data11109
00001336  4EB9 00001EC6            416      JS
```

```
000012D8        JSR             $0000203A

000012DE        JSR             $00001D6E

000012E4        MOVE.W  D6, D3

000012E6        MULS    #$0100, D3

000012EA        CMP.W   #$0000, D3

000012EE        DATA    6600

000012F0        ORI.B   #$4EB9, (A0)+

000012F4        ORI.B   #$1BB2, D0

000012F8        JSR             $00001EC6

000012FE        JSR             $00001E6E

00001304        BRA     088C

00001308        JSR             $00001E6E

0000130E        JSR             $00001EC6

00001314        JSR             $00001BB2

0000131A        BRA     0876

0000131E        LEA             $000023F9, A2

00001324        JSR             $0000203A

0000132A        JSR             $00001D6E

00001330        JSR             $00001ED4

00001336        JSR             $00001EC6
```

```
» R         AppendComma
0000133C  4EB9 00001BB2            417      JS
» R         EA_AppendMXn
00001342  6000 084E                418      BR
» A         Opcode_Finish
00001346                           419
00001346                           420
00001346                           421  Opcode
» _ADDQ:
00001346  45F9 000022E7            422      LE
» A         STR_ADDQ, A2
0000134C  4EB9 0000203A            423      JS
» R         AppendOutput            ; write AD
» DQ to the output string
00001352  4EB9 00001D6E            424      JS
» R         Opcode_AppendSizeSuffix
00001358  4EB9 00001ED4            425      JS
» R         Data11109
0000135E  4EB9 00001EC6            426      JS
» R         AppendComma
00001364  4EB9 00001BB2            427      JS
» R         EA_AppendMXn
0000136A  6000 0826                428      BR
» A         Opcode_Finish
0000136E                           429
0000136E                           430  Opcode
» _MULS:
0000136E  45F9 000023BB            431      LE
» A         STR_MULS, A2
00001374  4EB9 0000203A            432      JS
» R         AppendOutput            ; write MU
» LS to the output string
0000137A  4EB9 00001FBA            433      JS
» R         OpcodeSize_SetToWord
00001380  4EB9 00001BB2            434      JS
» R         EA_AppendMXn
00001386  4EB9 00001EC6            435      JS
» R         AppendComma
0000138C  4EB9 00001E6E            436      JS
» R         Dn
00001392  6000 07FE                437      BR
» A         Opcode_Finish
00001396                           438
00001396                           439
00001396                           440  Opcode
» _ORI:
00001396  45F9 000023D2            441      LE
» A         STR_ORI, A2
0000139C  4EB9 0000203A            442      JS
» R         AppendOutput            ; write th
» e current string in A2 to the output string
000013A2  4EB9 00001D6E            443      JS
```

```
0000133C        JSR             $00001BB2


00001342        BRA     084E


00001346        LEA             $000022E7, A2
0000134C        JSR             $0000203A


00001352        JSR             $00001D6E
00001358        JSR             $00001ED4
0000135E        JSR             $00001EC6
00001364        JSR             $00001BB2


0000136A        BRA     0826


0000136E        LEA             $000023BB, A2
00001374        JSR             $0000203A


0000137A        JSR             $00001FBA
00001380        JSR             $00001BB2
00001386        JSR             $00001EC6
0000138C        JSR             $00001E6E
00001392        BRA     07FE


00001396        LEA             $000023D2, A2
0000139C        JSR             $0000203A


000013A2        JSR             $00001D6E
```

```
» R         Opcode_AppendSizeSuffix
000013A8  B87C 0000                444      CM
» P.W       #$0000, D4
000013AC  6700 002E                445      BE
» Q         ORI_B                   ; if size
» == byte then fetch next byte
000013B0  B87C 0001                446      CM
» P.W       #$0001, D4
000013B4  6700 004A                447      BE
» Q         ORI_W                   ; if size
» == word then fetch next word
000013B8                           448  ORI_L:
»                          ; if size == long
» then fetch next long
000013B8  4EB8 107E                449      JS
» R         Get_Next_Long_D7
000013BC  45F9 000022CB            450      LE
» A         STRINGPOUNDHEX, A2

000013C2  4EB9 0000203A            451      JS
» R         AppendOutput            ; print "#
» $"
000013C8  3806                     452      MO
» VE.W      D6, D4                ; temp store w
» ord in D4
000013CA  2C07                     453      MO
» VE.L      D7, D6                ; move long im
» mediate into D6
000013CC  4EB9 000020C4            454      JS
» R         PrintASCIILong
000013D2  4EB9 00001FCA            455      JS
» R         OpcodeSize_SetToLong
000013D8  6000 004A                456      BR
» A         ORI_END
000013DC                           457  ORI_B:
000013DC  4EB8 1078                458      JS
» R         Get_Next_Word_D7
000013E0  45F9 000022CB            459      LE
» A         STRINGPOUNDHEX, A2
000013E6  4EB9 0000203A            460      JS
» R         AppendOutput            ; print "#
» $"
000013EC  3806                     461      MO
» VE.W      D6, D4                ; temp store w
» ord in D4
000013EE  3C07                     462      MO
» VE.W      D7, D6                ; move word im
» mediate into D6
000013F0  4EB9 00002066            463      JS
» R         PrintASCIIWord
000013F6  4EB9 00001FAA            464      JS
» R         OpcodeSize_SetToByte
```

```
000013A8        CMP.W   #$0000, D4

000013AC        DATA    6700

000013AE        ORI.B   #$B87C, $00016700

000013B6        ORI.W   #$4EB8, A2

000013BA        MOVEA.B $45F90000, A0

000013C0        MOVE.L  A3, (A1)+
000013C2        JSR             $0000203A

000013C8        MOVE.W  D6, D4

000013CA        MOVE.L  D7, D6

000013CC        JSR             $000020C4

000013D2        JSR             $00001FCA

000013D8        BRA     004A

000013DC        JSR             $1078

000013E0        LEA             $000022CB, A2

000013E6        JSR             $0000203A

000013EC        MOVE.W  D6, D4

000013EE        MOVE.W  D7, D6

000013F0        JSR             $00002066

000013F6        JSR             $00001FAA
```

```
000013FC  6000 0026                465      BR
» A         ORI_END
00001400                           466  ORI_W:
00001400  4EB8 1078                467      JS
» R         Get_Next_Word_D7
00001404  45F9 000022CB            468      LE
» A         STRINGPOUNDHEX, A2
0000140A  4EB9 0000203A            469      JS
» R         AppendOutput            ; print "#
» $"
00001410  3806                     470      MO
» VE.W      D6, D4                ; temp store w
» ord in D4
00001412  3C07                     471      MO
» VE.W      D7, D6                ; move word im
» mediate into D6
00001414  4EB9 00002066            472      JS
» R         PrintASCIIWord
0000141A  4EB9 00001FBA            473      JS
» R         OpcodeSize_SetToWord
00001420  6000 0002                474      BR
» A         ORI_END
00001424                           475  ORI_EN
» D:
00001424                           476
00001424  3C04                     477      MO
» VE.W      D4, D6                ; revert word
» to D6
00001426  4EB9 00001EC6            478      JS
» R         AppendComma
0000142C  4EB9 00001BB2            479      JS
» R         EA_AppendMXn
00001432  6000 075E                480      BR
» A         Opcode_Finish
00001436                           481
00001436                           482
00001436                           483  Opcode
» _CMPI:
00001436  45F9 00002323            484      LE
» A         STR_CMPI, A2
0000143C  4EB9 0000203A            485      JS
» R         AppendOutput            ; write th
» e current string in A2 to the output string
00001442  4EB9 00001D6E            486      JS
» R         Opcode_AppendSizeSuffix
00001448  B87C 0000                487      CM
» P.W       #$0000, D4
0000144C  6700 002E                488      BE
» Q         CMPI_B                  ; if size
» == byte then fetch next byte
00001450  B87C 0001                489      CM
» P.W       #$0001, D4
```

```
000013FC        BRA     0026
00001400        JSR             $1078
00001404        LEA             $000022CB, A2
0000140A        JSR             $0000203A
00001410        MOVE.W  D6, D4
00001412        MOVE.W  D7, D6
00001414        JSR             $00002066
0000141A        JSR             $00001FBA
00001420        BRA     0002
00001424        MOVE.W  D4, D6
00001426        JSR             $00001EC6
0000142C        JSR             $00001BB2
00001432        BRA     075E
00001436        LEA             $00002323, A2
0000143C        JSR             $0000203A
00001442        JSR             $00001D6E
00001448        CMP.W   #$0000, D4
0000144C        DATA    6700
0000144E        ORI.B   #$B87C, $00016700
```

```
00001454  6700 004A                490      BE
» Q         CMPI_W                  ; if size
» == word then fetch next word
00001458                           491  CMPI_L
» :                       ; if size == long
» then fetch next long
00001458  4EB8 107E                492      JS
» R         Get_Next_Long_D7
0000145C  45F9 000022CB            493      LE
» A         STRINGPOUNDHEX, A2

00001462  4EB9 0000203A            494      JS
» R         AppendOutput            ; print "#
» $"
00001468  3806                     495      MO
» VE.W      D6, D4              ; temp store w
» ord in D4
0000146A  2C07                     496      MO
» VE.L      D7, D6              ; move long im
» mediate into D6
0000146C  4EB9 000020C4            497      JS
» R         PrintASCIILong
00001472  4EB9 00001FCA            498      JS
» R         OpcodeSize_SetToLong
00001478  6000 004A                499      BR
» A         CMPI_END
0000147C                           500  CMPI_B
» :
0000147C  4EB8 1078                501      JS
» R         Get_Next_Word_D7
00001480  45F9 000022CB            502      LE
» A         STRINGPOUNDHEX, A2
00001486  4EB9 0000203A            503      JS
» R         AppendOutput            ; print "#
» $"
0000148C  3806                     504      MO
» VE.W      D6, D4              ; temp store w
» ord in D4
0000148E  3C07                     505      MO
» VE.W      D7, D6              ; move word im
» mediate into D6
00001490  4EB9 00002066            506      JS
» R         PrintASCIIWord
00001496  4EB9 00001FAA            507      JS
» R         OpcodeSize_SetToByte
0000149C  6000 0026                508      BR
» A         CMPI_END

000014A0                           509  CMPI_W
» :
000014A0  4EB8 1078                510      JS
» R         Get_Next_Word_D7
```

```
00001456        ORI.W   #$4EB8, A2

0000145A        MOVEA.B $45F90000, A0

00001460        MOVE.L  A3, (A1)+
00001462        JSR             $0000203A

00001468        MOVE.W  D6, D4

0000146A        MOVE.L  D7, D6

0000146C        JSR             $000020C4

00001472        JSR             $00001FCA

00001478        BRA     004A

0000147C        JSR             $1078

00001480        LEA             $000022CB, A2

00001486        JSR             $0000203A

0000148C        MOVE.W  D6, D4

0000148E        MOVE.W  D7, D6

00001490        JSR             $00002066

00001496        JSR             $00001FAA
0000149C        BRA     0026

000014A0        JSR             $1078
```

```
000014A4  45F9 000022CB            511      LE
» A         STRINGPOUNDHEX, A2
000014AA  4EB9 0000203A            512      JS
» R         AppendOutput            ; print "#
» $"
000014B0  3806                     513      MO
» VE.W      D6, D4              ; temp store w
» ord in D4
000014B2  3C07                     514      MO
» VE.W      D7, D6              ; move word im
» mediate into D6
000014B4  4EB9 00002066            515      JS
» R         PrintASCIIWord
000014BA  4EB9 00001FBA            516      JS
» R         OpcodeSize_SetToWord
000014C0  6000 0002                517      BR
» A         CMPI_END
000014C4                           518  CMPI_E
» ND:
000014C4  3C04                     519      MO
» VE.W      D4, D6              ; revert word
» to D6
000014C6  4EB9 00001EC6            520      JS
» R         AppendComma
000014CC  4EB9 00001BB2            521      JS
» R         EA_AppendMXn
000014D2  6000 06BE                522      BR
» A         Opcode_Finish
000014D6                           523
000014D6                           524
000014D6                           525
000014D6                           526  Opcode
» _EORI:
000014D6  45F9 0000233C            527      LE
» A         STR_EORI, A2
000014DC  4EB9 0000203A            528      JS
» R         AppendOutput            ; write th
» e current string in A2 to the output string
000014E2  4EB9 00001D6E            529      JS
» R         Opcode_AppendSizeSuffix
000014E8  B87C 0000                530      CM
» P.W       #$0000, D4
000014EC  6700 002E                531      BE
» Q         EORI_B                  ; if size
» == byte then fetch next byte
000014F0  B87C 0001                532      CM
» P.W       #$0001, D4
000014F4  6700 004A                533      BE
» Q         EORI_W                  ; if size
» == word then fetch next word
000014F8                           534  EORI_L
» :                         ; if size == long
```

```
000014A4        LEA             $000022CB, A2

000014AA        JSR             $0000203A

000014B0        MOVE.W  D6, D4

000014B2        MOVE.W  D7, D6

000014B4        JSR             $00002066

000014BA        JSR             $00001FBA

000014C0        BRA     0002

000014C4        MOVE.W  D4, D6

000014C6        JSR             $00001EC6

000014CC        JSR             $00001BB2

000014D2        BRA     06BE

000014D6        LEA             $0000233C, A2

000014DC        JSR             $0000203A

000014E2        JSR             $00001D6E

000014E8        CMP.W   #$0000, D4

000014EC        DATA    6700

000014EE        ORI.B   #$B87C, $00016700

000014F6        ORI.W   #$4EB8, A2
```

```
» then fetch next long
000014F8  4EB8 107E                535       JS
» R         Get_Next_Long_D7
000014FC  45F9 000022CB            536       LE
» A         STRINGPOUNDHEX, A2

00001502  4EB9 0000203A            537       JS
» R         AppendOutput            ; print "#
» $"
00001508  3806                     538       MO
» VE.W      D6, D4                ; temp store w
» ord in D4
0000150A  2C07                     539       MO
» VE.L      D7, D6                ; move long im
» mediate into D6
0000150C  4EB9 000020C4            540       JS
» R         PrintASCIILong
00001512  4EB9 00001FCA            541       JS
» R         OpcodeSize_SetToLong
00001518  6000 004A                542       BR
» A         EORI_END
0000151C                           543   EORI_B
» :
0000151C  4EB8 1078                544       JS
» R         Get_Next_Word_D7
00001520  45F9 000022CB            545       LE
» A         STRINGPOUNDHEX, A2
00001526  4EB9 0000203A            546       JS
» R         AppendOutput            ; print "#
» $"
0000152C  3806                     547       MO
» VE.W      D6, D4                ; temp store w
» ord in D4
0000152E  3C07                     548       MO
» VE.W      D7, D6                ; move word im
» mediate into D6
00001530  4EB9 00002066            549       JS
» R         PrintASCIIWord
00001536  4EB9 00001FAA            550       JS
» R         OpcodeSize_SetToByte
0000153C  6000 0026                551       BR
» A         EORI_END
00001540                           552   EORI_W
» :
00001540  4EB8 1078                553       JS
» R         Get_Next_Word_D7
00001544  45F9 000022CB            554       LE
» A         STRINGPOUNDHEX, A2
0000154A  4EB9 0000203A            555       JS
» R         AppendOutput            ; print "#
» $"
00001550  3806                     556       MO
```

```
000014FA        MOVEA.B $45F90000, A0

00001500        MOVE.L  A3, (A1)+
00001502        JSR             $0000203A

00001508        MOVE.W  D6, D4

0000150A        MOVE.L  D7, D6

0000150C        JSR             $000020C4

00001512        JSR             $00001FCA

00001518        BRA     004A

0000151C        JSR             $1078

00001520        LEA             $000022CB, A2

00001526        JSR             $0000203A

0000152C        MOVE.W  D6, D4

0000152E        MOVE.W  D7, D6

00001530        JSR             $00002066

00001536        JSR             $00001FAA

0000153C        BRA     0026

00001540        JSR             $1078

00001544        LEA             $000022CB, A2

0000154A        JSR             $0000203A

00001550        MOVE.W  D6, D4
```

```
» VE.W      D6, D4              ; temp store w
» ord in D4
00001552  3C07                   557      MO
» VE.W      D7, D6              ; move word im
» mediate into D6
00001554  4EB9 00002066          558      JS
» R         PrintASCIIWord
0000155A  4EB9 00001FBA          559      JS
» R         OpcodeSize_SetToWord
00001560  6000 0002              560      BR
» A         EORI_END
00001564                         561  EORI_E
» ND:
00001564  3C04                   562      MO
» VE.W      D4, D6              ; revert word
» to D6
00001566  4EB9 00001EC6          563      JS
» R         AppendComma
0000156C  4EB9 00001BB2          564      JS
» R         EA_AppendMXn
00001572  6000 061E              565      BR
» A         Opcode_Finish
00001576                         566
00001576                         567
00001576                         568
00001576                         569  Opcode
» _BCLR:
00001576  45F9 000022F9          570      LE
» A         STR_BCLR, A2
0000157C  4EB9 0000203A          571      JS
» R         AppendOutput         ; write th
» e current string in A2 to the output string
00001582  4EB9 00001E6E          572      JS
» R         Dn
00001588  4EB9 00001EC6          573      JS
» R         AppendComma
0000158E  4EB9 00001BB2          574      JS
» R         EA_AppendMXn
00001594  6000 05FC              575      BR
» A         Opcode_Finish
00001598                         576
00001598                         577  BCLR_I
» :
00001598  45F9 000022F9          578      LE
» A         STR_BCLR, A2
0000159E  4EB9 0000203A          579      JS
» R         AppendOutput         ; write th
» e current string in A2 to the output string
000015A4  4EB8 1078              580      JS
» R         Get_Next_Word_D7
000015A8  45F9 000022CB          581      LE
» A         STRINGPOUNDHEX, A2
```

```
00001552        MOVE.W  D7, D6
00001554        JSR             $00002066
0000155A        JSR             $00001FBA
00001560        BRA     0002
00001564        MOVE.W  D4, D6
00001566        JSR             $00001EC6
0000156C        JSR             $00001BB2
00001572        BRA     061E
0001576        LEA             $000022F9, A2
0000157C        JSR             $0000203A
00001582        JSR             $00001E6E
00001588        JSR             $00001EC6
0000158E        JSR             $00001BB2
00001594        BRA     05FC
00001598        LEA             $000022F9, A2
0000159E        JSR             $0000203A
000015A4        JSR             $1078
000015A8        LEA             $000022CB, A2
```

```
000015AE  4EB9 0000203A            582      JS      |  000015AE        JSR             $0000203A
» R         AppendOutput            ; print "#       |
» $"                                                |
000015B4  3806                     583      MO      |  000015B4        MOVE.W  D6, D4
» VE.W      D6, D4                ; temp store w     |
» ord in D4                                         |
000015B6  3C07                     584      MO      |  000015B6        MOVE.W  D7, D6
» VE.W      D7, D6                ; move word im     |
» mediate into D6                                   |
000015B8  4EB9 00002066            585      JS      |  000015B8        JSR             $00002066
» R         PrintASCIIWord                          |
000015BE  3C04                     586      MO      |  000015BE        MOVE.W  D4, D6
» VE.W      D4, D6                ; revert word      |
» to D6                                             |
000015C0  4EB9 00001EC6            587      JS      |  000015C0        JSR             $00001EC6
» R         AppendComma                             |
000015C6  4EB9 00001BB2            588      JS      |  000015C6        JSR             $00001BB2
» R         EA_AppendMXn                            |
000015CC  6000 05C4                589      BR      |
» A         Opcode_Finish                           |
000015D0                           590              |  000015CC        BRA     05C4
000015D0                           591              |
000015D0                           592  Opcode      |
» _ADDA:                                            |
000015D0  45F9 000022E1            593      LE      |  000015D0        LEA             $000022E1, A2
» A         STR_ADDA, A2                            |
000015D6  4EB9 0000203A            594      JS      |  000015D6        JSR             $0000203A
» R         AppendOutput            ; write AD      |
» DA to the output string                           |
000015DC  4EB9 00001DD4            595      JS      |  000015DC        JSR             $00001DD4
» R         Size8                                   |
000015E2  4EB9 00001BB2            596      JS      |  000015E2        JSR             $00001BB2
» R         EA_AppendMXn                            |
000015E8  4EB9 00001EC6            597      JS      |  000015E8        JSR             $00001EC6
» R         AppendComma                             |
000015EE  4EB9 00001EA8            598      JS      |  000015EE        JSR             $00001EA8
» R         An                                      |
000015F4  6000 059C                599      BR      |
» A         Opcode_Finish                           |
000015F8                           600              |  000015F4        BRA     059C
000015F8                           601              |
000015F8                           602              |
000015F8                           603  Opcode      |
» _ADD:                                             |
000015F8  45F9 000022DC            604      LE      |  000015F8        LEA             $000022DC, A2
» A         STR_ADD, A2                             |
000015FE  4EB9 0000203A            605      JS      |  000015FE        JSR             $0000203A
» R         AppendOutput            ; write AD      |
» D to the output string                            |
00001604  4EB9 00001D6E            606      JS      |  00001604        JSR             $00001D6E
» R         Opcode_AppendSizeSuffix                 |
0000160A  3606                     607      MO      |  0000160A        MOVE.W  D6, D3
```

```
» VE.W      D6,D3                    ; move wor
» d value into D3
0000160C  C67C 0100                608      AN
» D         #MASK_8, D3              ; mask bit
»  8
00001610  B67C 0000                609      CM
» P.W       #$0000,D3                ; compare
» with 0
00001614  6600 0018                610      BN
» E         ADD_EA_DN_EA
00001618                           611  ADD_DN
» _EA_DN:                  ; case  Dn + <ea>
» -> Dn
00001618  4EB9 00001BB2            612      JS
» R         EA_AppendMXn
0000161E  4EB9 00001EC6            613      JS
» R         AppendComma
00001624  4EB9 00001E6E            614      JS
» R         Dn
0000162A  6000 0566                615      BR
» A         Opcode_Finish
0000162E                           616  ADD_EA
» _DN_EA:                  ; case <ea> + Dn -
» > <ea>
0000162E  4EB9 00001E6E            617      JS
» R         Dn
00001634  4EB9 00001EC6            618      JS
» R         AppendComma
0000163A  4EB9 00001BB2            619      JS
» R         EA_AppendMXn
00001640  6000 0550                620      BR
» A         Opcode_Finish
00001644                           621
00001644                           622
00001644                           623
00001644                           624  Opcode
» _DIVS:
00001644  45F9 00002330            625      LE
» A         STR_DIVS, A2
0000164A  4EB9 0000203A            626      JS
» R         AppendOutput                ; writ
» e DIVS to the output string
00001650  4EB9 00001FBA            627      JS
» R         OpcodeSize_SetToWord        ; set
» the flag for .W
00001656  4EB9 00001BB2            628      JS
» R         EA_AppendMXn
0000165C  4EB9 00001EC6            629      JS
» R         AppendComma
00001662  4EB9 00001E6E            630      JS
» R         Dn
00001668  6000 0528                631      BR
```

```
0000160C        MULS    #$0100, D3

00001610        CMP.W   #$0000, D3

00001614        DATA    6600

00001616        ORI.B   #$4EB9, (A0)+

0000161A        ORI.B   #$1BB2, D0

0000161E        JSR             $00001EC6

00001624        JSR             $00001E6E

0000162A        BRA     0566

0000162E        JSR             $00001E6E

00001634        JSR             $00001EC6

0000163A        JSR             $00001BB2

00001640        BRA     0550

00001644        LEA             $00002330, A2

0000164A        JSR             $0000203A

00001650        JSR             $00001FBA

00001656        JSR             $00001BB2

0000165C        JSR             $00001EC6

00001662        JSR             $00001E6E
```

```
» A         Opcode_Finish
0000166C                          632
0000166C                          633
0000166C                          634  Opcode
» _OR:
0000166C  45F9 000023CE           635      LE
» A STR_OR, A2
00001672  4EB9 0000203A           636      JS
» R AppendOutput            ; write OR to the
» output string
00001678  4EB9 00001D6E           637      JS
» R Opcode_AppendSizeSuffix
0000167E  3606                    638      MO
» VE.W  D6,D3               ; move word value
» into D3
00001680  C67C 0100               639      AN
» D #MASK_8, D3             ; mask bit 8
00001684  B67C 0000               640      CM
» P.W #$0000,D3             ; compare with 0
00001688  6600 0018               641      BN
» E OR_EA_DN_EA
0000168C                          642  OR_DN_
» EA_DN:                    ; case  Dn + <ea>
» -> Dn
0000168C  4EB9 00001BB2           643      JS
» R EA_AppendMXn
00001692  4EB9 00001EC6           644      JS
» R AppendComma
00001698  4EB9 00001E6E           645      JS
» R Dn
0000169E  6000 04F2               646      BR
» A Opcode_Finish
000016A2                          647  OR_EA_
» DN_EA:                    ; case <ea> + Dn -
» > <ea>
000016A2  4EB9 00001E6E           648      JS
» R Dn
000016A8  4EB9 00001EC6           649      JS
» R AppendComma
000016AE  4EB9 00001BB2           650      JS
» R EA_AppendMXn
000016B4  6000 04DC               651      BR
» A Opcode_Finish
000016B8                          652
000016B8                          653
000016B8                          654
000016B8                          655  Opcode
» _CMP:
000016B8  45F9 0000231E           656      LE
» A         STR_CMP, A2
000016BE  4EB9 0000203A           657      JS
» R         AppendOutput           ; write CM
```

```
00001668        BRA     0528

0000166C        LEA             $000023CE, A2

00001672        JSR             $0000203A

00001678        JSR             $00001D6E

0000167E        MOVE.W  D6, D3

00001680        MULS    #$0100, D3

00001684        CMP.W   #$0000, D3

00001688        DATA    6600

0000168A        ORI.B   #$4EB9, (A0)+

0000168E        ORI.B   #$1BB2, D0

00001692        JSR             $00001EC6

00001698        JSR             $00001E6E

0000169E        BRA     04F2

000016A2        JSR             $00001E6E

000016A8        JSR             $00001EC6

000016AE        JSR             $00001BB2

000016B4        BRA     04DC

000016B8        LEA             $0000231E, A2
```

```
» P to the output string
000016C4  4EB9 00001D6E            658      JS
» R         Opcode_AppendSizeSuffix
000016CA  4EB9 00001BB2            659      JS
» R         EA_AppendMXn
000016D0  4EB9 00001EC6            660      JS
» R         AppendComma

000016D6  4EB9 00001E6E            661      JS
» R         Dn
000016DC  6000 04B4                662      BR
» A         Opcode_Finish
000016E0                           663
000016E0                           664
000016E0                           665
000016E0                           666  Opcode
» _EOR:
000016E0  45F9 00002337            667      LE
» A         STR_EOR, A2
000016E6  4EB9 0000203A            668      JS
» R         AppendOutput            ; write EO
» R to the output string
000016EC  4EB9 00001D6E            669      JS
» R         Opcode_AppendSizeSuffix
000016F2  4EB9 00001E6E            670      JS
» R         Dn
000016F8  4EB9 00001EC6            671      JS
» R         AppendComma
000016FE  4EB9 00001BB2            672      JS
» R         EA_AppendMXn
00001704  6000 048C                673      BR
» A         Opcode_Finish
00001708                           674
00001708                           675
00001708                           676
00001708                           677  MOVEA_
» B:
00001708  45F9 00002366            678      LE
» A         STR_MOVEA_B, A2
0000170E  4EB9 0000203A            679      JS
» R         AppendOutput            ; write MO
» VEA.B to the output string
00001714  4EB9 00001FAA            680      JS
» R         OpcodeSize_SetToByte
0000171A  4EB9 00001BB2            681      JS
» R         EA_AppendMXn
00001720  4EB9 00001EC6            682      JS
» R         AppendComma
00001726  4EB9 00001EA8            683      JS
» R         An
0000172C  6000 0464                684      BR
» A         Opcode_Finish
```

```
000016BE        JSR             $0000203A

000016C4        JSR             $00001D6E

000016CA        JSR             $00001BB2
000016D0        JSR             $00001EC6

000016D6        JSR             $00001E6E
000016DC        BRA     04B4

000016E0        LEA             $00002337, A2
000016E6        JSR             $0000203A

000016EC        JSR             $00001D6E
000016F2        JSR             $00001E6E
000016F8        JSR             $00001EC6
000016FE        JSR             $00001BB2

00001704        BRA     048C

00001708        LEA             $00002366, A2
0000170E        JSR             $0000203A

00001714        JSR             $00001FAA
0000171A        JSR             $00001BB2
00001720        JSR             $00001EC6
00001726        JSR             $00001EA8
```

```
00001730                          685
00001730                          686
00001730                          687
00001730                          688  MOVE_B
» :
00001730  45F9 000023A0           689      LE
» A         STR_MOVE_B, A2
00001736  4EB9 0000203A           690      JS
» R         AppendOutput           ; write MO
» VE.B to the output string
0000173C  4EB9 00001FAA           691      JS
» R         OpcodeSize_SetToByte
00001742  4EB9 00001BB2           692      JS
» R         EA_AppendMXn
00001748  4EB9 00001EC6           693      JS
» R         AppendComma
0000174E  4EB9 00001BC0           694      JS
» R         EA_AppendXnM
00001754  6000 043C               695      BR
» A         Opcode_Finish
00001758                          696
00001758                          697
00001758                          698  MOVEA_
» W:
00001758  45F9 0000237A           699      LE
» A         STR_MOVEA_W, A2
0000175E  4EB9 0000203A           700      JS
» R         AppendOutput           ; write MO
» VEA.W to the output string
00001764  4EB9 00001FBA           701      JS
» R         OpcodeSize_SetToWord
0000176A  4EB9 00001BB2           702      JS
» R         EA_AppendMXn
00001770  4EB9 00001EC6           703      JS
» R         AppendComma
00001776  4EB9 00001EA8           704      JS
» R         An
0000177C  6000 0414               705      BR
» A         Opcode_Finish
00001780                          706
00001780                          707
00001780                          708
00001780                          709  MOVE_W
» :
00001780  45F9 000023B2           710      LE
» A         STR_MOVE_W, A2
00001786  4EB9 0000203A           711      JS
» R         AppendOutput           ; write MO
» VE.W to the output string
0000178C  4EB9 00001FBA           712      JS
» R         OpcodeSize_SetToWord
00001792  4EB9 00001BB2           713      JS
```

```
0000172C        BRA     0464
00001730        LEA             $000023A0, A2
00001736        JSR             $0000203A
0000173C        JSR             $00001FAA
00001742        JSR             $00001BB2
00001748        JSR             $00001EC6
0000174E        JSR             $00001BC0
00001754        BRA     043C
00001758        LEA             $0000237A, A2
0000175E        JSR             $0000203A
00001764        JSR             $00001FBA
0000176A        JSR             $00001BB2
00001770        JSR             $00001EC6
00001776        JSR             $00001EA8
0000177C        BRA     0414
00001780        LEA             $000023B2, A2
00001786        JSR             $0000203A
0000178C        JSR             $00001FBA
00001792        JSR             $00001BB2
```

```
» R         EA_AppendMXn
00001798  4EB9 00001EC6            714      JS
» R         AppendComma
0000179E  4EB9 00001BC0            715      JS
» R         EA_AppendXnM
000017A4  6000 03EC                716      BR
» A         Opcode_Finish
000017A8                           717
000017A8                           718
000017A8                           719
000017A8                           720  MOVEA_
» L:
000017A8  45F9 00002370            721      LE
» A         STR_MOVEA_L, A2
000017AE  4EB9 0000203A            722      JS
» R         AppendOutput            ; write MO
» VEA.L to the output string
000017B4  4EB9 00001FCA            723      JS
» R         OpcodeSize_SetToLong
000017BA  4EB9 00001BB2            724      JS
» R         EA_AppendMXn
000017C0  4EB9 00001EC6            725      JS
» R         AppendComma
000017C6  4EB9 00001EA8            726      JS
» R         An
000017CC  6000 03C4                727      BR
» A         Opcode_Finish
000017D0                           728
000017D0                           729
000017D0                           730
000017D0                           731  MOVE_L
» :
000017D0  45F9 000023A9            732      LE
» A         STR_MOVE_L, A2
000017D6  4EB9 0000203A            733      JS
» R         AppendOutput            ; write MO
» VE.L to the output string
000017DC  4EB9 00001FCA            734      JS
» R         OpcodeSize_SetToLong
000017E2  4EB9 00001BB2            735      JS
» R         EA_AppendMXn
000017E8  4EB9 00001EC6            736      JS
» R         AppendComma
000017EE  4EB9 00001BC0            737      JS
» R         EA_AppendXnM
000017F4  6000 039C                738      BR
» A         Opcode_Finish
000017F8                           739
000017F8                           740
000017F8                           741
000017F8                           742  Opcode
» _ROL:
```

```
000017A8        LEA             $00002370, A2
000017AE        JSR             $0000203A
000017B4        JSR             $00001FCA
000017BA        JSR             $00001BB2
000017C0        JSR             $00001EC6
000017C6        JSR             $00001EA8
000017CC        BRA     03C4
000017D0        LEA             $000023A9, A2
000017D6        JSR             $0000203A
000017DC        JSR             $00001FCA
000017E2        JSR             $00001BB2
000017E8        JSR             $00001EC6
000017EE        JSR             $00001BC0
000017F4        BRA     039C
```

```
000017F8  45F9 000023D7           743      LE
» A         STR_ROL, A2
000017FE  4EB9 0000203A           744      JS
» R         AppendOutput           ; write RO
» L to the output string
00001804  4EB9 00001BB2           745      JS
» R         EA_AppendMXn
0000180A  6000 0386               746      BR
» A         Opcode_Finish
0000180E                          747
0000180E                          748
0000180E                          749
0000180E                          750  Opcode
» _ROR:
0000180E  45F9 000023DD           751      LE
» A         STR_ROR, A2
00001814  4EB9 0000203A           752      JS
» R         AppendOutput           ; write RO
» R to the output string
0000181A  4EB9 00001BB2           753      JS
» R         EA_AppendMXn
00001820  6000 0370               754      BR
» A         Opcode_Finish
00001824                          755
00001824                          756
00001824                          757
00001824                          758  Opcode
» _LSL:
00001824  45F9 0000235A           759      LE
» A         STR_LSL, A2
0000182A  4EB9 0000203A           760      JS
» R         AppendOutput           ; write LS
» L to the output string
00001830  4EB9 00001BB2           761      JS
» R         EA_AppendMXn
00001836  6000 035A               762      BR
» A         Opcode_Finish
0000183A                          763
0000183A                          764
0000183A                          765
0000183A                          766  Opcode
» _LSR:
0000183A  45F9 00002360           767      LE
» A         STR_LSR, A2
00001840  4EB9 0000203A           768      JS
» R         AppendOutput           ; write LS
» R to the output string
00001846  4EB9 00001BB2           769      JS
» R         EA_AppendMXn
0000184C  6000 0344               770      BR
» A         Opcode_Finish
00001850                          771
```

```
000017F8        LEA             $000023D7, A2

000017FE        JSR             $0000203A

00001804        JSR             $00001BB2

0000180A        BRA     0386

0000180E        LEA             $000023DD, A2

00001814        JSR             $0000203A

0000181A        JSR             $00001BB2

00001820        BRA     0370

00001824        LEA             $0000235A, A2

0000182A        JSR             $0000203A

00001830        JSR             $00001BB2

00001836        BRA     035A

0000183A        LEA             $00002360, A2

00001840        JSR             $0000203A

00001846        JSR             $00001BB2

0000184C        BRA     0344
```

```
00001850                            772
00001850                            773  Opcode
» _ASL:
00001850  45F9 000022ED             774       LE
» A         STR_ASL, A2
00001856  4EB9 0000203A             775       JS
» R         AppendOutput             ; write AS
» L to the output string
0000185C  4EB9 00001BB2             776       JS
» R         EA_AppendMXn
00001862  6000 032E                 777       BR
» A         Opcode_Finish
00001866                            778
00001866                            779
00001866                            780  Opcode
» _ASR:
00001866  45F9 000022F3             781       LE
» A         STR_ASR, A2
0000186C  4EB9 0000203A             782       JS
» R         AppendOutput             ; write AS
» R to the output string
00001872  4EB9 00001BB2             783       JS
» R         EA_AppendMXn
00001878  6000 0318                 784       BR
» A         Opcode_Finish
0000187C                            785
0000187C                            786
0000187C                            787  ROL_I:
0000187C  45F9 000023D7             788       LE
» A         STR_ROL, A2
00001882  4EB9 0000203A             789       JS
» R         AppendOutput             ; write RO
» L to the output string
00001888  4EB9 00001D6E             790       JS
» R         Opcode_AppendSizeSuffix
0000188E  3606                      791       MO
» VE.W      D6,D3                ; move word va
» lue into working register D3
00001890  C67C 0040                 792       AN
» D         #MASK_5, D3              ; mask bit
»  5
00001894  B67C 0000                 793       CM
» P.W       #$0000,D3               ; compare wi
» th 0

00001898  6600 0018                 794       BN
» E         ROL_I_DN
0000189C                            795  ROL_I_
» I:                        ; case of immediat
» e input
0000189C  4EB9 00001ED4             796       JS
```

```
00001850        LEA             $000022ED, A2

00001856        JSR             $0000203A

0000185C        JSR             $00001BB2

00001862        BRA     032E

00001866        LEA             $000022F3, A2

0000186C        JSR             $0000203A

00001872        JSR             $00001BB2

00001878        BRA     0318

0000187C        LEA             $000023D7, A2

00001882        JSR             $0000203A

00001888        JSR             $00001D6E

0000188E        MOVE.W  D6, D3

00001890        MULS    #$0040, D3

00001894        CMP.W   #$0000, D3

00001898        DATA    6600
0000189A        ORI.B   #$4EB9, (A0)+
0000189E        ORI.B   #$1ED4, D0
```

```
» R         Data11109
000018A2  4EB9 00001EC6            797      JS
» R         AppendComma
000018A8  4EB9 00001E8C            798      JS
» R         Dn210
000018AE  6000 02E2                799      BR
» A         Opcode_Finish
000018B2                           800  ROL_I_
» DN:                       ; case of data reg
» ister input
000018B2  4EB9 00001E6E            801      JS
» R         Dn
000018B8  4EB9 00001EC6            802      JS
» R         AppendComma
000018BE  4EB9 00001E8C            803      JS
» R         Dn210
000018C4  6000 02CC                804      BR
» A         Opcode_Finish
000018C8                           805
000018C8                           806
000018C8                           807  ROR_I:
000018C8  45F9 000023DD            808      LE
» A         STR_ROR, A2
000018CE  4EB9 0000203A            809      JS
» R         AppendOutput            ; write RO
» R to the output string
000018D4  4EB9 00001D6E            810      JS
» R         Opcode_AppendSizeSuffix
000018DA  3606                     811      MO
» VE.W      D6,D3               ; move word va
» lue into working register D3
000018DC  C67C 0040                812      AN
» D         #MASK_5, D3             ; mask bit
»  5
000018E0  B67C 0000                813      CM
» P.W       #$0000,D3             ; compare wi
» th 0


000018E4  6600 0018                814      BN
» E         ROR_I_DN
000018E8                           815  ROR_I_
» I:                        ; case of immediat
» e input
000018E8  4EB9 00001ED4            816      JS
» R         Data11109
000018EE  4EB9 00001EC6            817      JS
» R         AppendComma
000018F4  4EB9 00001E8C            818      JS
» R         Dn210
000018FA  6000 0296                819      BR
» A         Opcode_Finish
```

```
000018A2        JSR             $00001EC6

000018A8        JSR             $00001E8C

000018AE        BRA     02E2


000018B2        JSR             $00001E6E

000018B8        JSR             $00001EC6

000018BE        JSR             $00001E8C


000018C4        BRA     02CC


000018C8        LEA             $000023DD, A2

000018CE        JSR             $0000203A


000018D4        JSR             $00001D6E

000018DA        MOVE.W  D6, D3


000018DC        MULS    #$0040, D3


000018E0        CMP.W   #$0000, D3


000018E4        DATA    6600
000018E6        ORI.B   #$4EB9, (A0)+
000018EA        ORI.B   #$1ED4, D0


000018EE        JSR             $00001EC6

000018F4        JSR             $00001E8C

000018FA        BRA     0296
```

```
000018FE                            820  ROR_I_
» DN:                          ; case of data reg
» ister input
000018FE  4EB9 00001E6E             821      JS
» R         Dn
00001904  4EB9 00001EC6             822      JS
» R         AppendComma
0000190A  4EB9 00001E8C             823      JS
» R         Dn210
00001910  6000 0280                 824      BR
» A         Opcode_Finish
00001914                            825
00001914                            826
00001914                            827  LSL_I:
00001914  45F9 0000235A             828      LE
» A         STR_LSL, A2
0000191A  4EB9 0000203A             829      JS
» R         AppendOutput             ; write LS
» L to the output string
00001920  4EB9 00001D6E             830      JS
» R         Opcode_AppendSizeSuffix
00001926  3606                      831      MO
» VE.W      D6,D3                ; move word va
» lue into working register D3
00001928  C67C 0040                 832      AN
» D         #MASK_5, D3              ; mask bit
»  5
0000192C  B67C 0000                 833      CM
» P.W       #$0000,D3              ; compare wi
» th 0
00001930  6600 0018                 834      BN
» E         LSL_I_DN
00001934                            835  LSL_I_
» I:                           ; case of immediat
» e input
00001934  4EB9 00001ED4             836      JS
» R         Data11109
0000193A  4EB9 00001EC6             837      JS
» R         AppendComma
00001940  4EB9 00001E8C             838      JS
» R         Dn210
00001946  6000 024A                 839      BR
» A         Opcode_Finish
0000194A                            840  LSL_I_
» DN:                          ; case of data reg
» ister input
0000194A  4EB9 00001E6E             841      JS
» R         Dn
00001950  4EB9 00001EC6             842      JS
» R         AppendComma
00001956  4EB9 00001E8C             843      JS
» R         Dn210
```

```
000018FE        JSR             $00001E6E
00001904        JSR             $00001EC6
0000190A        JSR             $00001E8C

00001910        BRA     0280

00001914        LEA             $0000235A, A2
0000191A        JSR             $0000203A

00001920        JSR             $00001D6E
00001926        MOVE.W  D6, D3
00001928        MULS    #$0040, D3

0000192C        CMP.W   #$0000, D3
00001930        DATA    6600
00001932        ORI.B   #$4EB9, (A0)+
00001936        ORI.B   #$1ED4, D0
0000193A        JSR             $00001EC6
00001940        JSR             $00001E8C
00001946        BRA     024A

0000194A        JSR             $00001E6E
00001950        JSR             $00001EC6
00001956        JSR             $00001E8C
```

```
0000195C  6000 0234                844      BR
» A         Opcode_Finish
00001960                           845
00001960                           846
00001960                           847  LSR_I:
00001960  45F9 00002360            848      LE
» A         STR_LSR, A2
00001966  4EB9 0000203A            849      JS
» R         AppendOutput                ; writ
» e LSR to the output string
0000196C  4EB9 00001D6E            850      JS
» R         Opcode_AppendSizeSuffix
00001972  3606                     851      MO
» VE.W      D6,D3                       ; move
»  word value into working register D3
00001974  C67C 0040                852      AN
» D         #MASK_5, D3                 ; mask
»  bit 5
00001978  B67C 0000                853      CM
» P.W       #$0000,D3                   ; comp
» are with 0
0000197C  6600 0018                854      BN
» E         LSR_I_DN
00001980                           855  LSR_I_
» I:                       ; case of immediat
» e input
00001980  4EB9 00001ED4            856      JS
» R         Data11109
00001986  4EB9 00001EC6            857      JS
» R         AppendComma
0000198C  4EB9 00001E8C            858      JS
» R         Dn210
00001992  6000 01FE                859      BR
» A         Opcode_Finish
00001996                           860  LSR_I_
» DN:                      ; case of data reg
» ister input
00001996  4EB9 00001E6E            861      JS
» R         Dn
0000199C  4EB9 00001EC6            862      JS
» R         AppendComma
000019A2  4EB9 00001E8C            863      JS
» R         Dn210
000019A8  6000 01E8                864      BR
» A         Opcode_Finish
000019AC                           865
000019AC                           866
000019AC                           867  ASL_I:
000019AC  45F9 000022ED            868      LE
» A         STR_ASL, A2
000019B2  4EB9 0000203A            869      JS
» R         AppendOutput                ; writ
```

```
0000195C        BRA     0234
00001960        LEA             $00002360, A2
00001966        JSR             $0000203A
0000196C        JSR             $00001D6E
00001972        MOVE.W  D6, D3
00001974        MULS    #$0040, D3
00001978        CMP.W   #$0000, D3
0000197C        DATA    6600
0000197E        ORI.B   #$4EB9, (A0)+
00001982        ORI.B   #$1ED4, D0
00001986        JSR             $00001EC6
0000198C        JSR             $00001E8C
00001992        BRA     01FE
00001996        JSR             $00001E6E
0000199C        JSR             $00001EC6
000019A2        JSR             $00001E8C
000019A8        BRA     01E8
000019AC        LEA             $000022ED, A2
000019B2        JSR             $0000203A
```

```
» e ASL to the output string
000019B8  4EB9 00001D6E            870      JS
» R         Opcode_AppendSizeSuffix
000019BE  3606                     871      MO
» VE.W      D6,D3                        ; move
»  word value into working register D3
000019C0  C67C 0040                872      AN
» D         #MASK_5, D3                  ; mask
»  bit 5
000019C4  B67C 0000                873      CM
» P.W       #$0000,D3                    ; comp
» are with 0
000019C8  6600 0018                874      BN
» E         ASL_I_DN
000019CC                           875  ASL_I_
» I:                       ; case of immediat
» e input
000019CC  4EB9 0000 1ED4           876      JS
» R         Data11109
000019D2  4EB9 00001EC6            877      JS
» R         AppendComma
000019D8  4EB9 00001E8C            878      JS
» R         Dn210
000019DE  6000 01B2                879      BR
» A         Opcode_Finish
000019E2                           880  ASL_I_
» DN:                      ; case of data reg
» ister input
000019E2  4EB9 00001E6E            881      JS
» R         Dn
000019E8  4EB9 00001EC6            882      JS
» R         AppendComma
000019EE  4EB9 00001E8C            883      JS
» R         Dn210
000019F4  6000 019C                884      BR
» A         Opcode_Finish
000019F8                           885
000019F8                           886  ASR_I:
000019F8  45F9 000022F3            887      LE
» A         STR_ASR, A2
000019FE  4EB9 0000203A            888      JS
» R         AppendOutput                 ; writ
» e ASR to the output string
00001A04  4EB9 00001D6E            889      JS
» R         Opcode_AppendSizeSuffix
00001A0A  3606                     890      MO
» VE.W      D6,D3                        ; move
»  word value into working register D3
00001A0C  C67C 0040                891      AN
» D         #MASK_5, D3                  ; mask
»  bit 5
00001A10  B67C 0000                892      CM
```

```
000019B8        JSR             $00001D6E

000019BE        MOVE.W  D6, D3

000019C0        MULS    #$0040, D3

000019C4        CMP.W   #$0000, D3

000019C8        DATA    6600

000019CA        ORI.B   #$4EB9, (A0)+

000019CE        ORI.B   #$1ED4, D0

000019D2        JSR             $00001EC6

000019D8        JSR             $00001E8C

000019DE        BRA     01B2


000019E2        JSR             $00001E6E

000019E8        JSR             $00001EC6

000019EE        JSR             $00001E8C

000019F4        BRA     019C


000019F8        LEA             $000022F3, A2

000019FE        JSR             $0000203A

00001A04        JSR             $00001D6E

00001A0A        MOVE.W  D6, D3

00001A0C        MULS    #$0040, D3

00001A10        CMP.W   #$0000, D3
```

```
» P.W       #$0000,D3                   ; comp
» are with 0
00001A14  6600 0018               893       BN
» E         ASR_I_DN
00001A18                          894  ASR_I_
» I:                        ; case of immediat
» e input
00001A18  4EB9 00001ED4           895       JS
» R         Data11109
00001A1E  4EB9 00001EC6           896       JS
» R         AppendComma
00001A24  4EB9 00001E8C           897       JS
» R         Dn210
00001A2A  6000 0166               898       BR
» A         Opcode_Finish
00001A2E                          899  ASR_I_
» DN:                       ; case of data reg
» ister input
00001A2E  4EB9 00001E6E           900       JS
» R         Dn
00001A34  4EB9 00001EC6           901       JS
» R         AppendComma
00001A3A  4EB9 00001E8C           902       JS
» R         Dn210
00001A40  6000 0150               903       BR
» A         Opcode_Finish
00001A44                          904
00001A44                          905
00001A44                          906  Opcode
» _BRA:
00001A44  45F9 00002312           907       LE
» A         STR_BRA, A2
00001A4A  4EB9 0000203A           908       JS
» R         AppendOutput             ; write th
» e current string in A2 to the output string
00001A50  4EB9 00001E40           909       JS
» R         Displacement
00001A56  6000 013A               910       BR
» A         Opcode_Finish
00001A5A                          911
00001A5A                          912  Opcode
» _BCS:
00001A5A  45F9 00002300           913       LE
» A         STR_BCS, A2
00001A60  4EB9 0000203A           914       JS
» R         AppendOutput             ; write th
» e current string in A2 to the output string
00001A66  4EB9 00001E40           915       JS
» R         Displacement
00001A6C  6000 0124               916       BR
» A         Opcode_Finish
00001A70                          917
```

```
00001A14        DATA    6600
00001A16        ORI.B   #$4EB9, (A0)+
00001A1A        ORI.B   #$1ED4, D0
00001A1E        JSR             $00001EC6
00001A24        JSR             $00001E8C
00001A2A        BRA     0166
00001A2E        JSR             $00001E6E
00001A34        JSR             $00001EC6
00001A3A        JSR             $00001E8C
00001A40        BRA     0150
00001A44        LEA             $00002312, A2
00001A4A        JSR             $0000203A
00001A50        JSR             $00001E40
00001A56        BRA     013A
00001A5A        LEA             $00002300, A2
00001A60        JSR             $0000203A
00001A66        JSR             $00001E40
00001A6C        BRA     0124
```

```
00001A70                          918
00001A70                          919  Opcode
» _BGE:
00001A70  45F9 00002306           920       LE      00001A70        LEA             $00002306, A2
» A         STR_BGE, A2
00001A76  4EB9 0000203A           921       JS      00001A76        JSR             $0000203A
» R         AppendOutput           ; write th
» e current string in A2 to the output string
00001A7C  4EB9 00001E40           922       JS      00001A7C        JSR             $00001E40
» R         Displacement
00001A82  6000 010E               923       BR
» A         Opcode_Finish
00001A86                          924               00001A82        BRA     010E
00001A86                          925  Opcode
» _BLT:
00001A86  45F9 0000230C           926       LE      00001A86        LEA             $0000230C, A2
» A         STR_BLT, A2
00001A8C  4EB9 0000203A           927       JS      00001A8C        JSR             $0000203A
» R         AppendOutput           ; write th
» e current string in A2 to the output string
00001A92  4EB9 00001E40           928       JS      00001A92        JSR             $00001E40
» R         Displacement
00001A98  6000 00F8               929       BR
» A         Opcode_Finish
00001A9C                          930               00001A98        BRA     00F8
00001A9C                          931
00001A9C                          932  Opcode
» _BVC:
00001A9C  45F9 00002318           933       LE      00001A9C        LEA             $00002318, A2
» A         STR_BVC, A2
00001AA2  4EB9 0000203A           934       JS      00001AA2        JSR             $0000203A
» R         AppendOutput                   ;
» write the current string in A2 to the output
»  string
00001AA8  4EB9 00001E40           935       JS      00001AA8        JSR             $00001E40
» R         Displacement
00001AAE  6000 00E2               936       BR
» A         Opcode_Finish
00001AB2                          937               00001AAE        BRA     00E2
00001AB2                          938  Opcode
» _NOP:
00001AB2  45F9 000023C7           939       LE      00001AB2        LEA             $000023C7, A2
» A         STR_NOP, A2
00001AB8  4EB9 0000203A           940       JS      00001AB8        JSR             $0000203A
» R         AppendOutput                   ;
» write the current string in A2 to the output
»  string
00001ABE  6000 00D2               941       BR      00001ABE        BRA     00D2
» A         Opcode_Finish
00001AC2                          942
00001AC2                          943  Opcode
» _ILLEGAL:
```

00001AC2 45F9 00002342 944 LE	00001AC2 LEA $00002342, A2
» A STR_ILLEGAL, A2	
00001AC8 4EB9 0000203A 945 JS	00001AC8 JSR $0000203A
» R AppendOutput ;	
» write the current string in A2 to the output	
» string	
00001ACE 6000 00C2 946 BR	00001ACE BRA 00C2
» A Opcode_Finish	
00001AD2 947	
00001AD2 948 Opcode	
» _RTS:	
00001AD2 45F9 000023E3 949 LE	00001AD2 LEA $000023E3, A2
» A STR_RTS, A2	
00001AD8 4EB9 0000203A 950 JS	00001AD8 JSR $0000203A
» R AppendOutput ;	
» write the current string in A2 to the output	
» string	
00001ADE 6000 00B2 951 BR	
» A Opcode_Finish	
00001AE2 952	00001ADE BRA 00B2
00001AE2 953	
00001AE2 954 Opcode	
» _LEA:	
00001AE2 45F9 00002353 955 LE	00001AE2 LEA $00002353, A2
» A STR_LEA, A2	
00001AE8 4EB9 0000203A 956 JS	00001AE8 JSR $0000203A
» R AppendOutput ;	
» write the current string in A2 to the output	
» string	
00001AEE 4EB9 00001BB2 957 JS	00001AEE JSR $00001BB2
» R EA_AppendMXn	
00001AF4 4EB9 00001EC6 958 JS	00001AF4 JSR $00001EC6
» R AppendComma	
00001AFA 4EB9 00001EA8 959 JS	00001AFA JSR $00001EA8
» R An	
00001B00 6000 0090 960 BR	
» A Opcode_Finish	
00001B04 961	00001B00 BRA 0090
00001B04 962 Opcode	
» _NEG:	
00001B04 45F9 000023C2 963 LE	00001B04 LEA $000023C2, A2
» A STR_NEG, A2	
00001B0A 4EB9 0000203A 964 JS	00001B0A JSR $0000203A
» R AppendOutput ;	
» write the current string in A2 to the output	
» string	
00001B10 4EB9 00001D6E 965 JS	00001B10 JSR $00001D6E
» R Opcode_AppendSizeSuffix	
00001B16 4EB9 00001BB2 966 JS	00001B16 JSR $00001BB2
» R EA_AppendMXn	
00001B1C 6000 0074 967 BR	00001B1C BRA 0074
» A Opcode_Finish	

```
00001B20                          968
00001B20                          969  Opcode
» _MOVEM:
00001B20  45F9 00002384           970      LE
» A         STR_MOVEM, A2
00001B26  4EB9 0000203A           971      JS
» R         AppendOutput                    ;
» write the current string in A2
00001B2C  4EB9 00001E0A           972      JS
» R         Size6
00001B32  3606                    973      MO
» VE.W      D6,D3                           ;
» move word value into working register D3
00001B34  C67C 0400               974      AN
» D         #MASK_10, D3                    ;
» mask bit 5
00001B38  B67C 0000               975      CM
» P.W       #$0000,D3                       ;
» compare with 0
00001B3C  6600 0018               976      BN
» E         MOVEM_EA_RL
00001B40                          977  MOVEM_
» RL_EA:                  ; case of register
» s to memory
00001B40  4EB9 00001F08           978      JS
» R         RegisterList_Predecrement
00001B46  4EB9 00001EC6           979      JS
» R         AppendComma
00001B4C  4EB9 00001BB2           980      JS
» R         EA_AppendMXn
00001B52  6000 003E               981      BR
» A         Opcode_Finish
00001B56                          982  MOVEM_
» EA_RL:                  ; case of memory t
» o registers
00001B56  4EB9 00001BB2           983      JS
» R         EA_AppendMXn
00001B5C  4EB9 00001EC6           984      JS
» R         AppendComma
00001B62  4EB9 00001F58           985      JS
» R         RegisterList_PostIncrement
00001B68  6000 0028               986      BR
» A         Opcode_Finish
00001B6C                          987
00001B6C                          988  Opcode
» _JSR:
00001B6C  45F9 0000234C           989      LE
» A         STR_JSR, A2
00001B72  4EB9 0000203A           990      JS
» R         AppendOutput
00001B78  4EB9 00001BB2           991      JS
» R         EA_AppendMXn                    ;
```

```
00001B20        LEA             $00002384, A2

00001B26        JSR             $0000203A

00001B2C        JSR             $00001E0A

00001B32        MOVE.W  D6, D3

00001B34        MULS    #$0400, D3

00001B38        CMP.W   #$0000, D3

00001B3C        DATA    6600

00001B3E        ORI.B   #$4EB9, (A0)+

00001B42        ORI.B   #$1F08, D0

00001B46        JSR             $00001EC6

00001B4C        JSR             $00001BB2

00001B52        BRA     003E

00001B56        JSR             $00001BB2

00001B5C        JSR             $00001EC6

00001B62        JSR             $00001F58

00001B68        BRA     0028

00001B6C        LEA             $0000234C, A2

00001B72        JSR             $0000203A

00001B78        JSR             $00001BB2
```

```
» Sets D4 & 5 to Src Name & Value; Sets D6 & 7
»  to Dest Name & Value
00001B7E  6000 0012                992      BR      00001B7E        BRA     0012
» A         Opcode_Finish
00001B82                           993
00001B82                           994  Opcode
» _SIMHALT:
00001B82  45F9 000023EA            995      LE      00001B82        LEA             $000023EA, A2
» A         STR_SIMHALT, A2
00001B88  4EB9 0000203A            996      JS      00001B88        JSR             $0000203A
» R         AppendOutput                       ;
» write the current string in A2 to the output
»  string
00001B8E  6000 0002                997      BR
» A         Opcode_Finish
00001B92                           998              00001B8E        BRA     0002
00001B92                           999  Opcode
» _Finish:
00001B92  43F9 0000220B           1000      LE
» A         OUTPUT, A1                         ;
» Load Output into A1
00001B98  4EB9 000020DE           1001      JS
» R         TrapTask13                         ;
» Send it to the printer
00001B9E  45F9 0000220A           1002      LE
» A         CURRENT_STR_LENGTH, A2
00001BA4  4212                    1003      CL
» R.B       (A2)                               ;
» Set Current String Length Back to Zero
00001BA6  4EB9 00001FDA           1004      JS
» R         OpcodeSize_SetToElse               ;
» reset Opcode Size
00001BAC  4CDF 101C               1005      MO
» VEM.L     (SP)+, D2-D4/A4                    ;
» return registers to their previous state
00001BB0  4E75                    1006      RT
» S                                            ;
» return from function
00001BB2                          1007              00001B92        LEA             $0000220B, A1
00001BB2                          1008              00001B98        JSR             $000020DE
00001BB2                          1009  ------      00001B9E        LEA             $0000220A, A2
» -------------- end include -----------------
» ---
00001BB2                          1010      IN
» CLUDE     'Addressing.X68'
00001BB2                          1011              00001BA4        NEG.B   (A2)
00001BB2                          1012              00001BA6        JSR             $00001FDA
                                                    00001BAC        MOVEM   .W      (A7)+,
                                                    00001BAE        MOVE.B  (A4)+, D0
00001BB2                          1013              00001BB0        RTS
00001BB2                          1014  EA_App
» endMXn:
```

```
00001BB2  4EB9 00001F76           1015      JS      |  00001BB2        JSR             $00001F76
» R         EA_SetIsMXn                             |
00001BB8  4EB9 00001BD4           1016      JS      |  00001BB8        JSR             $00001BD4
» R         EA_AppendModeRegister                   |
00001BBE  4E75                    1017      RT      |  00001BBE        RTS
» S                                                 |
00001BC0                          1018              |  00001BC0        JSR             $00001F90
00001BC0                          1019              |  00001BC6        JSR             $00001BD4
00001BC0                          1020  EA_App      |
» endXnM:                                           |
00001BC0  4EB9 00001F90           1021      JS      |
» R         EA_SetIsXnM                             |
00001BC6  4EB9 00001BD4           1022      JS      |
» R         EA_AppendModeRegister                   |
00001BCC  4EB9 00001F76           1023      JS      |  00001BCC        JSR             $00001F76
» R         EA_SetIsMXn                             |
00001BD2  4E75                    1024      RT      |  00001BD2        RTS
» S                                                 |
00001BD4                          1025              |
00001BD4                          1026              |
00001BD4                          1027              |
00001BD4                          1028  EA_App      |
» endModeRegister:                                  |
00001BD4  48E7 FC40               1029      MO      |  00001BD4        MOVEM   .L      D6/5//3//D2/1/
» VEM.L     D0-D5/A1,-(SP)                          |  » /D0/, -(A7)
00001BD8  2F06                    1030      MO      |  00001BD8        MOVE.L  D6, -(A7)
» VE.L      D6,-(SP)                                |
00001BDA  2606                    1031      MO      |  00001BDA        MOVE.L  D6, D3
» VE.L      D6,D3                           ;       |
» copy the current address into d3 for process      |
» ing                                               |
00001BDC                          1032              |  00001BDC        JSR             $00001D1E
00001BDC  4EB9 00001D1E           1033      JS      |
» R         EA_GetStandardModeInD3                  |
00001BE2                          1034              |  00001BE2        CMP.W   #$0000, D3
00001BE2  B67C 0000               1035      CM      |
» P         #MODE_Dn,D3                     ;       |
» check if mode = Dn, data reg direct               |
00001BE6  6700 0026               1036      BE      |
» Q         EA_SetStrArray_Dn                       |
00001BEA                          1037              |  00001BE6        DATA    6700
00001BEA  B67C 0008               1038      CM      |  00001BE8        ORI.B   #$B67C, -(A6)
» P         #Mode_An,D3                     ;       |
» check if mode = An, address reg direct            |
00001BEE  6700 0028               1039      BE      |  00001BEC        ORI.B   #$6700, A0
» Q         EA_SetStrArray_An                       |
00001BF2                          1040              |  00001BF0        ORI.B   #$B67C, $0010
00001BF2  B67C 0010               1041      CM      |
» P         #Mode_AnInd,D3                  ;       |
» check if mode = (An), address reg indirect        |
00001BF6  6700 002A               1042      BE      |
» Q         EA_SetStrArray_AnInd                    |
```

```
00001BFA                            1043
00001BFA  B67C 0018                 1044      CM
» P         #Mode_AnPostInc,D3                ;
» check if mode = -(An), address reg post incr
» ement
00001BFE  6700 0036                 1045      BE
» Q         EA_SetStrArray_AnPostInc
```

```
00001BF6        DATA    6700
00001BF8        ORI.B   #$B67C, $00186700
```

```
00001C00        ORI.B   #$B67C, $00206700
```

```
00001C08        ORI.B   #$6000, -(A4)
00001C0C        ORI.W   #$45F9, A4
00001C10        ORI.B   #$24BE, D0
```

```
00001C02                            1046
»
00001C02  B67C 0020                 1047      CM
» P         #Mode_AnPreDec,D3                 ;
» chec if mode = (An)+, address reg pre decrem
» ent
00001C06  6700 0024                 1048      BE
» Q         EA_SetStrArray_AnPreDec
00001C0A                            1049
00001C0A  6000 004C                 1050      BR
» A         EA_ProcessElse
00001C0E                            1051
00001C0E                            1052
00001C0E                            1053  EA_Set
» StrArray_Dn
00001C0E  45F9 000024BE             1054      LE
» A         EA_StrArray_Dn,A2
00001C14  6000 002A                 1055      BR
» A         EA_AppendRegisterName
00001C18                            1056
00001C18                            1057  EA_Set
» StrArray_An
00001C18  45F9 000024CE             1058      LE
» A         EA_StrArray_An,A2
00001C1E  6000 0020                 1059      BR
» A         EA_AppendRegisterName
00001C22                            1060
00001C22                            1061  EA_Set
» StrArray_AnInd
00001C22  45F9 000024DE             1062      LE
» A         EA_StrArray_AnInd,A2
00001C28  6000 0016                 1063      BR
» A         EA_AppendRegisterName
00001C2C                            1064
00001C2C                            1065  EA_Set
» StrArray_AnPreDec
00001C2C  45F9 000024FE             1066      LE
» A         EA_StrArray_AnPreDec,A2
00001C32  6000 000C                 1067      BR
» A         EA_AppendRegisterName
00001C36                            1068
00001C36                            1069  EA_Set
» StrArray_AnPostInc
```

```
00001C14        BRA     002A
```

```
00001C18        LEA             $000024CE, A2
00001C1E        BRA     0020
```

```
00001C22        LEA             $000024DE, A2
```

```
00001C28        BRA     0016
```

```
00001C2C        LEA             $000024FE, A2
```

```
00001C32        BRA     000C
```

```
00001C36        LEA             $000024EE, A2
```

```
00001C3C        BRA     0002
```

```
00001C36  45F9 000024EE           1070      LE
» A         EA_StrArray_AnPostInc,A2
00001C3C  6000 0002               1071      BR
» A         EA_AppendRegisterName
00001C40                          1072
00001C40                          1073
00001C40                          1074  EA_App
» endRegisterName
00001C40  4EB9 00001D46           1075      JS
» R         EA_GetStandardRegInD3
00001C46  C6FC 0002               1076      MU
» LU        #ARRAY_ELEMENT_WIDTH,D3         ;
» multiply by 2 because the string arrays are
» word-ordered
00001C4A  3472 3000               1077      MO
» VE.W      (A2,D3),A2                      ;
» A2 stores destination EA String
00001C4E  4EB9 0000203A           1078      JS
» R         AppendOutput                    ;
» AppendOutput prints the MXn result
00001C54  6000 00C0               1079      BR
» A         EA_Return
00001C58                          1080
00001C58                          1081
00001C58                          1082  EA_Pro
» cessElse
00001C58  4EB9 00001D46           1083      JS
» R         EA_GetStandardRegInD3
00001C5E                          1084
00001C5E  B67C 0004               1085      CM
» P         #MODE_Imm,D3
00001C62  6700 000E               1086      BE
» Q         EA_ProcessImmediate
00001C66                          1087


00001C66  B67C 0000               1088      CM
» P         #MODE_AbsWord,D3
00001C6A  6700 0074               1089      BE
» Q         EA_ProcessAbsoluteWord
00001C6E                          1090
»
00001C6E  6000 008A               1091      BR
» A         EA_ProcessAbsoluteLong
00001C72                          1092
00001C72                          1093  EA_Pro
» cessImmediate
00001C72  4EB9 00001FEA           1094      JS
» R         OpcodeSize_GetSize
00001C78                          1095
00001C78  B07C 0000               1096      CM
» P         #OPCODESIZE_BYTE,D0
```

```
00001C40        JSR             $00001D46
00001C46        MULS    #$0002, D3
00001C4A        MOVEA.W $30004EB9, A2

00001C50        ORI.B   #$203A, D0


00001C54        BRA     00C0


00001C58        JSR             $00001D46
00001C5E        CMP.W   #$0004, D3


00001C62        DATA    6700
00001C64        ORI.B   #$B67C, A6
00001C68        ORI.B   #$6700, D0
00001C6C        ORI.W   #$6000, #$008A


00001C72        JSR             $00001FEA
```

```
00001C7C  6700 000E               1097      BE
» Q         EA_ProcessImmediateByte
00001C80                          1098
00001C80  B07C 0001               1099      CM
» P         #OPCODESIZE_WORD,D0
00001C84  6700 0024               1100      BE
» Q         EA_ProcessImmediateWord
00001C88                          1101
00001C88  6000 003A               1102      BR
» A         EA_ProcessImmediateLong
00001C8C                          1103
                                                                  00001C78        CMP.W   #$0000, D0
                                                                  00001C7C        DATA    6700
                                                                  00001C7E        ORI.B   #$B07C, A6
                                                                  00001C82        ORI.B   #$6700, D1
                                                                  00001C86        ORI.B   #$6000, -(A4)
00001C8C                          1104
00001C8C                          1105  EA_Pro
» cessImmediateByte
00001C8C  45F9 000024B7           1106      LE                    00001C8A        ORI.B   #$45F9, $000024B7
» A         EA_Str_Hash,A2                  ;
» move the '#' value into A2
00001C92  4EB9 0000203A           1107      JS                    00001C92        JSR             $0000203A
» R         AppendOutput
00001C98                          1108                            00001C98        JSR             $1072
00001C98  4EB8 1072               1109      JS                    00001C9C        MULS    #$000F, D6
» R         Get_Next_Word_D6                ;
» pull the next word into D3
00001C9C  CC7C 000F               1110      AN
» D.W       #$0F,D6                         ;
» mask the word in D6 to a byte
00001CA0  4EB9 00002066           1111      JS
» R         PrintASCIIWord                  ;
» PrintAsciiWord uses d6 as input
00001CA6  6000 006E               1112      BR                    00001CA0        JSR             $00002066
» A         EA_Return
00001CAA                          1113                            00001CA6        BRA     006E
00001CAA                          1114  EA_Pro                    00001CAA        LEA             $000024B9, A2
» cessImmediateWord
00001CAA  45F9 000024B9           1115      LE
» A         EA_Str_HashDollar,A2
»     ; move the '#' value into A2
00001CB0  4EB9 0000203A           1116      JS                    00001CB0        JSR             $0000203A
» R         AppendOutput
00001CB6                          1117                            00001CB6        JSR             $1072
00001CB6  4EB8 1072               1118      JS
» R         Get_Next_Word_D6                ;
» pull the next word into D3
00001CBA  4EB9 00002066           1119      JS
» R         PrintASCIIWord                  ;
» PrintAsciiWord uses d6 as input
00001CC0  6000 0054               1120      BR                    00001CBA        JSR             $00002066
» A         EA_Return
00001CC4                          1121                            00001CC0        BRA     0054
00001CC4                          1122  EA_Pro                    00001CC4        LEA             $000024B9, A2
» cessImmediateLong
```

```
00001CC4  45F9 000024B9           1123      LE
» A         EA_Str_HashDollar,A2
»     ; move the '#' value into A2
00001CCA  4EB9 0000203A           1124      JS      00001CCA        JSR             $0000203A
» R         AppendOutput
00001CD0                          1125              00001CD0        JSR             $107E
00001CD0  4EB8 107E               1126      JS
» R         Get_Next_Long_D7                ;
» pull the next word into D3
00001CD4  2C07                    1127      MO      00001CD4        MOVE.L  D7, D6
» VE.L      D7,D6
00001CD6  4EB9 00002066           1128      JS
» R         PrintASCIIWord                  ;
» PrintAsciiWord uses d6 as input
00001CDC  6000 0038               1129      BR      00001CD6        JSR             $00002066
» A         EA_Return
00001CE0                          1130              00001CDC        BRA     0038
00001CE0                          1131
00001CE0                          1132              00001CE0        LEA             $000024BC, A2
00001CE0                          1133  EA_Pro
» cessAbsoluteWord
00001CE0  45F9 000024BC           1134      LE
» A         EA_Str_Dollar,A2            ; move
»  the '#$' value into A2
00001CE6  4EB9 0000203A           1135      JS      00001CE6        JSR             $0000203A
» R         AppendOutput
00001CEC                          1136              00001CEC        JSR             $1072
00001CEC  4EB8 1072               1137      JS
» R         Get_Next_Word_D6                ;
» pull the next word into D3
00001CF0  4EB9 00002066           1138      JS
» R         PrintASCIIWord                  ;
» PrintAsciiWord uses d6 as input
00001CF6  6000 001E               1139      BR      00001CF0        JSR             $00002066
» A         EA_Return
00001CFA                          1140              00001CF6        BRA     001E
00001CFA                          1141              00001CFA        LEA             $000024BC, A2
00001CFA                          1142  EA_Pro
» cessAbsoluteLong
00001CFA  45F9 000024BC           1143      LE
» A         EA_Str_Dollar,A2            ; move
»  the '#$' value into A2
00001D00  4EB9 0000203A           1144      JS      00001D00        JSR             $0000203A
» R         AppendOutput
00001D06                          1145              00001D06        JSR             $107E
00001D06  4EB8 107E               1146      JS
» R         Get_Next_Long_D7                ;
» pull the next word into D3
00001D0A  2C07                    1147      MO      00001D0A        MOVE.L  D7, D6
» VE.L      D7,D6
00001D0C  4EB9 000020C4           1148      JS
» R         PrintASCIILong                  ;
```

```
» PrintAsciiWord uses d6 as input
00001D12  6000 0002               1149      BR
» A         EA_Return
00001D16                          1150
00001D16                          1151
00001D16                          1152  EA_Ret
» urn
00001D16  2C1F                    1153      MO
» VE.L      (SP)+,D6
00001D18  4CDF 023F               1154      MO
» VEM.L     (SP)+,D0-D5/A1                  ;
» clean up the other stack storage
00001D1C  4E75                    1155      RT
» S
00001D1E                          1156
00001D1E                          1157

00001D1E                          1158
00001D1E                          1159
00001D1E                          1160
00001D1E                          1161  EA_Get
» StandardModeInD3




00001D1E  2F04                    1162      MO
» VE.L      D4,-(SP)
00001D20  2606                    1163      MO
» VE.L      D6,D3                           ;
» copy the current address into d3 for process
» ing
00001D22  4EB9 00001F68           1164      JS
» R         EA_GetIsXnM                     ;
» determine if this is MXn, or XnM,
00001D28                          1165
»                                           ;
» which determines which bits we inspect
00001D28  B07C 0001               1166      CM
» P         #IsTrue,D0                      ;
» if MXn, no bit shifting is required
00001D2C  6700 0006               1167      BE
» Q         EA_ShiftXnM_Mode
00001D30  6600 000A               1168      BN
» E         EA_Mask_Mode             ; if XnM,
»  shift over 3 bits so bits 8/7/6 can be trea
» ted like 5/4/3
00001D34                          1169
00001D34                          1170  EA_Shi
» ftXnM_Mode
00001D34  283C 00000003           1171      MO
```

```
00001D0C        JSR             $000020C4

00001D12        BRA     0002



00001D16        MOVE.L  (A7)+, D6



00001D18        MOVEM   .W      (A7)+,
00001D1A        ORI.B   #$4E75, $2F042606
00001D22        JSR             $00001F68
00001D28        CMP.W   #$0001, D0
00001D2C        DATA    6700
00001D2E        ORI.B   #$6600, D6

00001D32        ORI.B   #$283C, A2
00001D36        ORI.B   #$0003, D0
00001D3A        ASR     .L      #4, D3
00001D3C        MULS    #$0000, D3
00001D40        ORI.B   #$281F, $4E75
00001D46        MOVE.L  D4, -(A7)
```

```
» VE.L      #EA_MODE_ASRL_DISTANCE,D4
00001D3A  E8A3                    1172      AS
» R.L       D4,D3                           ;
» shift to the right so the MODE masks align
00001D3C                          1173
00001D3C                          1174  EA_Mas
» k_Mode
00001D3C  C6BC 00000038           1175      AN
» D.L       #MASK_543,D3
00001D42  281F                    1176      MO
» VE.L      (SP)+,D4
00001D44  4E75                    1177      RT
» S
00001D46                          1178
```

```
00001D48        MOVE.L  D6, D3
```

```
00001D4A        JSR             $00001F68
00001D50        CMP.W   #$0001, D0
00001D54        DATA    6700
00001D56        ORI.B   #$6600, D6
```

```
00001D46                          1179
00001D46                          1180  EA_Get
» StandardRegInD3
00001D46  2F04                    1181      MO
» VE.L      D4,-(SP)
00001D48  2606                    1182      MO
» VE.L      D6,D3                           ;
» copy the current address into d3 for process
» ing
00001D4A  4EB9 00001F68           1183      JS
» R         EA_GetIsXnM                     ;
» determine if this is MXn, or XnM,
00001D50                          1184
»                                           ;
» which determines which bits we inspect
00001D50  B07C 0001               1185      CM
» P         #IsTrue,D0                      ;
» if MXn, no bit shifting is required
00001D54  6700 0006               1186      BE
» Q         EA_ShiftXnM_Register            ;
» if XnM, shift over 3 bits so bits 8/7/6 can
» be treated like 5/4/3
00001D58  6600 000A               1187      BN
» E         EA_Mask_Register
00001D5C                          1188
00001D5C                          1189  EA_Shi
» ftXnM_Register
00001D5C  283C 00000009           1190      MO
» VE.L      #EA_REGISTER_ASRL_DISTANCE,D4
00001D62  E8A3                    1191      AS
» R.L       D4,D3                           ;
» shift to 543 for use with the standar MODE m
» asks
00001D64                          1192
00001D64                          1193  EA_Mas
» k_Register
00001D64  C6BC 00000007           1194      AN
```

```
00001D5A        ORI.B   #$283C, A2
00001D5E        ORI.B   #$0009, D0
```

```
00001D62        ASR     .L      #4, D3
```

```
00001D64        MULS    #$0000, D3
```

```
» D.L       #MASK_210,D3
00001D6A  281F                    1195      MO
» VE.L      (SP)+,D4
00001D6C  4E75                    1196      RT
» S
00001D6E                          1197
00001D6E                          1198
00001D6E                          1199
00001D6E                          1200  Opcode
» _AppendSizeSuffix:
00001D6E  3606                    1201      MO
» VE.W      D6, D3
»   ; move current word into working register
00001D70  C67C 00C0               1202      AN
» D.W       #MASK_76, D3
»   ; mask bits 7, 6
00001D74                          1203
00001D74  B67C 0000               1204      CM
» P.W       #$0000, D3
»   ; compare with 00
00001D78  6700 0012               1205      BE
» Q         Opcode_AppendSizeSuffix_B
»   ; if equal goto B
00001D7C                          1206
00001D7C  B67C 0040               1207      CM
» P.W       #$0040, D3
»   ; compare with 01
00001D80  6700 0022               1208      BE
» Q         Opcode_AppendSizeSuffix_W
»   ; if equal goto W
00001D84                          1209
00001D84  B67C 0080               1210      CM
» P.W       #$0080, D3
»   ; compare with 10
00001D88  6700 0032               1211      BE
» Q         Opcode_AppendSizeSuffix_L
»   ; if equal goto L
00001D8C                          1212
»
»   ; if bytes are 11 there is an error, not h
» andling this currently
00001D8C                          1213  Opcode
» _AppendSizeSuffix_B:
00001D8C  45F9 000022D0           1214      LE
» A         STRING_B, A2
00001D92  4EB9 0000203A           1215      JS
» R         AppendOutput
»   ; write '.B' to the output string
00001D98  383C 0000               1216      MO
» VE.W      #$0000, D4
»   ; set D4 to 00
00001D9C  4EB9 00001FAA           1217      JS
```

```
00001D68        ORI.B   #$281F, D7

00001D6C        RTS

00001D6E        MOVE.W  D6, D3

00001D70        MULS    #$00C0, D3
00001D74        CMP.W   #$0000, D3

00001D78        DATA    6700
00001D7A        ORI.B   #$B67C, (A2)

00001D7E        ORI.W   #$6700, D0

00001D82        ORI.B   #$B67C, -(A2)
00001D86        ORI.L   #$67000032, D0

00001D8C        LEA             $000022D0, A2

00001D92        JSR             $0000203A

00001D98        MOVE.W  #$0000, D4

00001D9C        JSR             $00001FAA
```

```
» R         OpcodeSize_SetToByte
00001DA2  4E75                    1218      RT
» S
00001DA4                          1219
00001DA4                          1220  Opcode
» _AppendSizeSuffix_W:
00001DA4  45F9 000022D4           1221      LE
» A         STRING_W, A2
00001DAA  4EB9 0000203A           1222      JS
» R         AppendOutput
»   ; write '.W' to the output string
00001DB0  383C 0001               1223      MO
» VE.W      #$0001, D4
»   ; set D4 to 01
00001DB4  4EB9 00001FBA           1224      JS
» R         OpcodeSize_SetToWord
00001DBA  4E75                    1225      RT
» S
00001DBC                          1226
00001DBC                          1227  Opcode
» _AppendSizeSuffix_L:
00001DBC  45F9 000022D8           1228      LE
» A         STRING_L, A2
00001DC2  4EB9 0000203A           1229      JS
» R         AppendOutput
»   ; write '.L' to the output string
00001DC8  383C 0002               1230      MO
» VE.W      #$0002, D4
»   ; set D4 to 10
00001DCC  4EB9 00001FCA           1231      JS
» R         OpcodeSize_SetToLong
00001DD2  4E75                    1232      RT
» S
00001DD4                          1233
00001DD4                          1234
00001DD4                          1235  Size8:
00001DD4  3606                    1236      MO
» VE.W      D6, D3              ; move current
»  word into working register
00001DD6  C67C 0100               1237      AN
» D.W       #MASK_8, D3         ; mask bit 8
00001DDA  B67C 0000               1238      CM
» P.W       #$0000, D3          ; compare with
»  D3
00001DDE  6700 0016               1239      BE
» Q         Size8_W             ; if equal got
» o W
00001DE2                          1240  Size8_
» L:                       ; if not equal fal
» l through to L
```

```
00001DA2        RTS

00001DA4        LEA             $000022D4, A2

00001DAA        JSR             $0000203A

00001DB0        MOVE.W  #$0001, D4

00001DB4        JSR             $00001FBA

00001DBA        RTS

00001DBC        LEA             $000022D8, A2

00001DC2        JSR             $0000203A

00001DC8        MOVE.W  #$0002, D4

00001DCC        JSR             $00001FCA

00001DD2        RTS

00001DD4        MOVE.W  D6, D3
00001DD6        MULS    #$0100, D3

00001DDA        CMP.W   #$0000, D3

00001DDE        DATA    6700

00001DE0        ORI.B   #$45F9, (A6)

00001DE4        ORI.B   #$22D8, D0
00001DE8        JSR             $0000203A
```

```
00001DE2  45F9 000022D8           1241      LE
» A         STRING_L, A2
00001DE8  4EB9 0000203A           1242      JS
» R         AppendOutput            ; write '.
» L' to the output string
00001DEE  4EB9 00001FCA           1243      JS
» R         OpcodeSize_SetToLong
00001DF4  4E75                    1244      RT
» S
00001DF6                          1245  Size8_
» W:
00001DF6  45F9 000022D4           1246      LE
» A         STRING_W, A2
00001DFC  4EB9 0000203A           1247      JS
» R         AppendOutput            ; write '.
» W' to the output string
00001E02  4EB9 00001FBA           1248      JS
» R         OpcodeSize_SetToWord
00001E08  4E75                    1249      RT
» S
```

```
00001DEE        JSR             $00001FCA
```

```
00001DF4        RTS
```

```
00001DF6        LEA             $000022D4, A2

00001DFC        JSR             $0000203A

00001E02        JSR             $00001FBA

00001E08        RTS
```

```
00001E0A  3606                    1252      MO
» VE.W      D6, D3                  ; move cur
» rent word into working register
00001E0C  C67C 0020               1253      AN
» D.W       #MASK_6, D3             ; mask bit
»  8
00001E10  B67C 0000               1254      CM
» P.W       #$0000, D3              ; compare
» with D3
00001E14  6700 0016               1255      BE
» Q         Size6_W                 ; if equal
»  goto W
00001E18                          1256  Size6_
» L:                                ; if not e
» qual fall through to L
00001E18  45F9 000022D8           1257      LE
» A         STRING_L, A2
00001E1E  4EB9 0000203A           1258      JS
» R         AppendOutput            ; write '.
» L' to the output string
00001E24  4EB9 00001FCA           1259      JS
» R         OpcodeSize_SetToLong
00001E2A  4E75                    1260      RT
» S
00001E2C                          1261  Size6_
```

```
00001E0A        MOVE.W  D6, D3
00001E0C        MULS    #$0020, D3
00001E10        CMP.W   #$0000, D3
00001E14        DATA    6700
00001E16        ORI.B   #$45F9, (A6)
00001E1A        ORI.B   #$22D8, D0
00001E1E        JSR             $0000203A
```

```
00001E24        JSR             $00001FCA
```

```
00001E2A        RTS
```

```
» W:
00001E2C  45F9 000022D4           1262      LE
» A         STRING_W, A2
00001E32  4EB9 0000203A           1263      JS
» R         AppendOutput            ; write '.
» W' to the output string
00001E38  4EB9 00001FBA           1264      JS
» R         OpcodeSize_SetToWord
00001E3E  4E75                    1265      RT
» S
00001E40                          1266


00001E40                          1267
00001E40                          1268  Displa
» cement:
00001E40  3606                    1269      MO
» VE.W      D6, D3                ; move current
»  word into working register
00001E42  C67C 00FF               1270      AN
» D.W       #MASK_76543210, D3  ; mask so only
»  the bottom byte is available
00001E46  B67C 0000               1271      CM
» P.W       #$0000, D3          ; compare with
»  0
00001E4A  6700 0010               1272      BE
» Q         Displacement_Fetch_Word
00001E4E  2F06                    1273      MO
» VE.L      D6,-(SP)        ; move D6 to stack
»  for preservation
00001E50  3C03                    1274      MO
» VE.W      D3, D6          ; move masked valu
» e into D6 for printing
00001E52  4EB9 00002066           1275      JS
» R         PrintASCIIWord
00001E58  2C1F                    1276      MO
» VE.L      (SP)+,D6        ; pop D6 from stac
» k
00001E5A  4E75                    1277      RT
» S
00001E5C                          1278  Displa
» cement_Fetch_Word:
00001E5C  4EB8 1078               1279      JS
» R         Get_Next_Word_D7
00001E60  3F06                    1280      MO
» VE.W      D6,-(SP)        ; move D6 to stack
»  for preservation
00001E62  3C07                    1281      MO
» VE.W      D7, D6          ; move value into
» D6 for printing
00001E64  4EB9 00002066           1282      JS
» R         PrintASCIIWord
```

```
00001E2C        LEA             $000022D4, A2

00001E32        JSR             $0000203A


00001E38        JSR             $00001FBA

00001E3E        RTS

00001E40        MOVE.W  D6, D3
00001E42        MULS    #$00FF, D3
00001E46        CMP.W   #$0000, D3
00001E4A        DATA    6700
00001E4C        ORI.B   #$2F06, (A0)

00001E50        MOVE.W  D3, D6


00001E52        JSR             $00002066

00001E58        MOVE.L  (A7)+, D6


00001E5A        RTS
```

```
00001E6A  3C1F                    1283      MO
» VE.W      (SP)+,D6        ; pop D6 from stac
» k
00001E6C  4E75                    1284      RT
» S

00001E6E                          1285
00001E6E                          1286

00001E6E                          1287  Dn:
00001E6E  3606                    1288      MO
» VE.W      D6,D3               ; move word in
» to D3
00001E70  C67C 0E00               1289      AN
» D.W       #MASK_11109, D3     ; mask bits 11
» , 10, 9
00001E74  EF5B                    1290      RO
» L.W       #7,D3               ; rotate so bi
» ts are in least significant spot
00001E76  C6FC 0002               1291      MU
» LU.W      #2,D3               ; multiply by
» 2 so offset is measured in words not bytes
00001E7A  49F9 000024BE           1292      LE
» A         EA_StrArray_Dn, A4
00001E80  3474 3000               1293      MO
» VE.W      (A4,D3),A2          ; move string
» value from StrArray_Dn with offset D3 into A
» 2
00001E84  4EB9 0000203A           1294      JS
» R         AppendOutput        ; print string
»  in A2
00001E8A  4E75                    1295      RT
» S
00001E8C                          1296
00001E8C                          1297
00001E8C                          1298  Dn210:
00001E8C  3606                    1299      MO
» VE.W      D6,D3               ; move word in
» to D3
00001E8E  C67C 0007               1300      AN
» D.W       #MASK_210, D3       ; mask bits 2,
»  1, 0
00001E92  C6FC 0002               1301      MU
» LU.W      #2,D3               ; multiply by
» 2 so offset is measured in words not bytes
00001E96  49F9 000024BE           1302      LE
» A         EA_StrArray_Dn, A4
00001E9C  3474 3000               1303      MO
» VE.W      (A4,D3),A2          ; move string
» value from StrArray_Dn with offset D3 into A
» 2
00001EA0  4EB9 0000203A           1304      JS
```

```
00001E5C        JSR             $1078

00001E60        MOVE.W  D6, -(A7)
00001E62        MOVE.W  D7, D6
00001E64        JSR             $000020666
00001E6A        MOVE.W  (A7)+, D6
00001E6C        RTS
00001E6E        MOVE.W  D6, D3

00001E70        MULS    #$0E00, D3

00001E74        ROL     .W      D7, D3

00001E76        MULS    #$0002, D3

00001E7A        LEA             $000024BE, A4

00001E80        MOVEA.W #$3000, A2

00001E84        JSR             $0000203A

00001E8A        RTS

00001E8C        MOVE.W  D6, D3
00001E8E        MULS    #$0007, D3
00001E92        MULS    #$0002, D3

00001E96        LEA             $000024BE, A4

00001E9C        MOVEA.W #$3000, A2

00001EA0        JSR             $0000203A
```

```
» R         AppendOutput        ; print string
»  in A2
00001EA6  4E75                    1305      RT
» S
00001EA8                          1306

00001EA8                          1307
00001EA8                          1308  An:
00001EA8  3606                    1309      MO
» VE.W      D6,D3                 ; move word in
» to D3
00001EAA  C67C 0E00               1310      AN
» D.W       #MASK_11109, D3       ; mask bits 11
» , 10, 9
00001EAE  EF5B                    1311      RO
» L.W       #7,D3                 ; rotate so bi
» ts are in least significant spot
00001EB0  C6FC 0002               1312      MU
» LU.W      #2,D3                 ; multiply by
» 2 so offset is measured in words not bytes
00001EB4  49F9 000024CE           1313      LE
» A         EA_StrArray_An, A4
00001EBA  3474 3000               1314      MO
» VE.W      (A4,D3),A2            ; move string
» value from StrArray_An with offset D3 into A
» 2
00001EBE  4EB9 0000203A           1315      JS
» R         AppendOutput        ; print string
»  in A2
00001EC4  4E75                    1316      RT
» S
00001EC6                          1317
00001EC6                          1318
00001EC6                          1319  Append
» Comma:
00001EC6  45F9 000022C6           1320      LE
» A         STRINGCOMMA, A2       ; move string
» comma into A2
00001ECC  4EB9 0000203A           1321      JS
» R         AppendOutput        ; print string
»  in A2
00001ED2  4E75                    1322      RT
» S
00001ED4                          1323


00001ED4                          1324
00001ED4                          1325  Data11
» 109:
00001ED4  3606                    1326      MO
» VE.W  D6, D3              ; move current wor
» d into working register
```

```
00001EA6        RTS

00001EA8        MOVE.W  D6, D3
00001EAA        MULS    #$0E00, D3
00001EAE        ROL     .W      D7, D3
00001EB0        MULS    #$0002, D3

00001EB4        LEA             $000024CE, A4

00001EBA        MOVEA.W #$3000, A2

00001EBE        JSR             $0000203A

00001EC4        RTS

00001EC6        LEA             $000022C6, A2
00001ECC        JSR             $0000203A

00001ED2        RTS

00001ED4        MOVE.W  D6, D3
00001ED6        MULS    #$0E00, D3
00001EDA        MOVEA.W $00002584, A4
00001EE0        ROL     .W      D7, D3
```

```
00001ED6  C67C 0E00              1327      AN
» D.W   #MASK_11109, D3     ; mask bits 11, 10
» , 9
00001EDA  3879 00002584          1328      MO
» VEA   Hex_StrArray, A4    ; Move the base ar
» ray into A4
00001EE0  EF5B                   1329      RO
» L     #7, D3              ; rotate left 7 so
»  bits are in least significant position
00001EE2  B67C 0000              1330      CM
» P.W   #0, D3              ; check if D3 is 0
00001EE6  6600 0004              1331      BN
» E     Data11109_NOT_ZERO  ; if D3 is not zer
» o, jump ahead
00001EEA  5043                   1332      AD
» D.W   #8, D3              ; if D3 is zero, a
» dd 8
00001EEC                         1333  Data11
» 109_NOT_ZERO:
00001EEC  C7FC 0002              1334      MU
» LS    #2, D3              ; multiply by 2 fo
» r correct array offset
00001EF0  45F9 000022C9          1335      LE
» A     STRINGPOUND, A2     ; move string # in
» to A2
00001EF6  4EB9 0000203A          1336      JS
» R     AppendOutput        ; print string # i
» n A2
00001EFC  45F4 3000              1337      LE
» A     (A4,D3), A2         ; move ASCII immed
» iate into A2
00001F00  4EB9 0000203A          1338      JS
» R     AppendOutput


00001F06  4E75                   1339      RT
» S
00001F08                         1340

00001F08                         1341

00001F08                         1342
00001F08                         1343  Regist
» erList_Predecrement:
00001F08  4EB8 1078              1344      JS
» R         Get_Next_Word_D7        ; get next
»  word into D7
00001F0C  3879 000024BE          1345      MO
» VEA       EA_StrArray_Dn, A4  ; move the bas
» e array into A4
00001F12  383C 0010              1346      MO
» VE.W      #16, D4             ; move counter
```

```
00001EE2        CMP.W   #$0000, D3

00001EE6        DATA    6600


00001EE8        ORI.B   #$5043, D4


00001EEC        MULS    #$0002, D3

00001EF0        LEA             $000022C9, A2


00001EF6        JSR             $0000203A

00001EFC        LEA             #$4EB9, A2
00001F02        ORI.B   #$203A, D0
00001F06        RTS

00001F08        JSR             $1078
00001F0C        MOVEA.W $000024BE, A4
00001F12        MOVE.W  #$0010, D4
00001F16        CMP.W   #$0000, D4
00001F1A        DATA    6700
```

```
»  into D4 and set to 15
00001F16                          1347  RL_LOO
» P:
00001F16  B87C 0000               1348      CM
» P.W       #0, D4              ; check for en
» d of loop
00001F1A  6700 0028               1349      BE
» Q         RL_LOOP_END
00001F1E  5344                    1350      SU
» BQ        #1, D4                ; decrement
»  D4
00001F20  0907                    1351      BT
» ST        D4, D7                ; test bit
» in position D4
00001F22  66F2                    1352      BN
» E         RL_LOOP
00001F24  C8FC 0002               1353      MU
» LU        #2, D4              ; multiply D4
» by 2 for offset
00001F28  45F4 4000               1354      LE
» A         (A4, D4), A2
00001F2C  4EB9 0000203A           1355      JS
» R         AppendOutput            ; print re
» gister
00001F32  89FC 0002               1356      DI
» VS        #2, D4                  ; divide D
» 4 by 2 for counter

00001F36  45F9 000022CE           1357      LE
» A         STRINGSLASH, A2
00001F3C  4EB9 0000203A           1358      JS
» R         AppendOutput            ; print '/
» '
00001F42  60D2                    1359      BR
» A         RL_LOOP
00001F44                          1360  RL_LOO
» P_END:
00001F44  43F9 0000220B           1361      LE
» A         OUTPUT, A1
00001F4A  45F9 0000220A           1362      LE
» A         CURRENT_STR_LENGTH, A2
00001F50  D2D2                    1363      AD
» D         (A2), A1
00001F52  12BC 0000               1364      MO
» VE.B      #00, (A1)
00001F56  4E75                    1365      RT
» S
00001F58                          1366
00001F58                          1367
00001F58                          1368  Regist
» erList_Postincrement:
00001F58                          1369      ;t
```

```
00001F1C        ORI.B   #$5344, $0907


00001F22        DATA    66F2


00001F24        MULS    #$0002, D4

00001F28        LEA             #$4EB9, A2

00001F2E        ORI.B   #$203A, D0

00001F32        DIVS    #$0002, D4
00001F36        LEA             $000022CE, A2

00001F3C        JSR             $0000203A

00001F42        BRA     00D2


00001F44        LEA             $0000220B, A1

00001F4A        LEA             $0000220A, A2

00001F50        ADDA.W  (A2), A1

00001F52        MOVE.B  #0000, (A1)

00001F56        RTS
```

```
» odo
00001F58  4E75                    1370      RT
» S
00001F5A                          1371
00001F5A                          1372
00001F5A                          1373
00001F5A                          1374
00001F5A                          1375
00001F5A                          1376
00001F5A                          1377   *====
» ========================================
» ========================================
» ========================
00001F5A                          1378   * Eff
» ective Addressing Helper Functions
00001F5A                          1379   *====
» ========================================
» ========================================
» ========================
00001F5A                          1380  EA_Get
» IsMXn
00001F5A  2F08                    1381      MO
» VE.L      A0,-(SP)           ; backup A0
00001F5C  41F9 000021ED           1382      LE
» A         EA_IsMXn,A0        ; A0 -> the 'is
» MXn' flag
00001F62  1010                    1383      MO
» VE.B      (A0),D0            ; copy the 'isM
» Xn' flag into D0
00001F64  205F                    1384      MO
» VE.L      (SP)+,A0           ; restore A0
00001F66  4E75                    1385      RT
» S
00001F68                          1386
00001F68                          1387  EA_Get
» IsXnM
00001F68  2F08                    1388      MO
» VE.L      A0,-(SP)           ; backup A0
00001F6A  41F9 000021EE           1389      LE
» A         EA_IsXnM,A0        ; A0 -> the 'is
» XnM' flag
00001F70  1010                    1390      MO
» VE.B      (A0),D0            ; copy the 'isX
» nM' flag into D0
00001F72  205F                    1391      MO
» VE.L      (SP)+,A0           ; restore A0
00001F74  4E75                    1392      RT
» S
00001F76                          1393
00001F76                          1394
00001F76                          1395  EA_Set
» IsMXn
```

```
00001F58        RTS

00001F5A        MOVE.L  A0, -(A7)
00001F5C        LEA             $000021ED, A0

00001F62        MOVE.B  (A0), D0

00001F64        MOVEA.L (A7)+, A0

00001F66        RTS

00001F68        MOVE.L  A0, -(A7)

00001F6A        LEA             $000021EE, A0

00001F70        MOVE.B  (A0), D0

00001F72        MOVEA.L (A7)+, A0

00001F74        RTS
```

```
00001F76  2F08                    1396      MO
» VE.L      A0,-(SP)            ; backup A0
00001F78  41F9 000021ED           1397      LE
» A         EA_IsMXn,A0         ; A0 -> isMXn
» flag
00001F7E  10BC 0001               1398      MO
» VE.B      #IsTrue,(A0)        ; set isMXn =
» to isTrue
00001F82  41F9 000021EE           1399      LE
» A         EA_IsXnM,A0         ; A0 -> isXnM
00001F88  10BC 0000               1400      MO
» VE.B      #IsFalse,(A0)       ; set isXnM =
» isFalse
00001F8C  205F                    1401      MO
» VE.L      (SP)+,A0            ; restore A0
00001F8E  4E75                    1402      RT
» S
00001F90                          1403
00001F90                          1404
00001F90                          1405  EA_Set
» IsXnM
00001F90  2F08                    1406      MO
» VE.L      A0,-(SP)            ; backup A0
00001F92  41F9 000021EE           1407      LE
» A         EA_IsXnM,A0         ; A0 -> XnM
00001F98  10BC 0001               1408      MO
» VE.B      #IsTrue,(A0)        ; XnM set to i
» sTrue
00001F9C  41F9 000021ED           1409      LE
» A         EA_IsMXn,A0         ; A0 -> MXn
00001FA2  10BC 0000               1410      MO
» VE.B      #IsFalse,(A0)       ; MXn set to i
» sFalse
00001FA6  205F                    1411      MO
» VE.L      (SP)+,A0            ; restore A0
00001FA8  4E75                    1412      RT
» S
00001FAA                          1413
00001FAA                          1414
00001FAA                          1415  Opcode
» Size_SetToByte
00001FAA  2F08                    1416      MO
» VE.L      A0,-(SP)
00001FAC  41F9 000021EC           1417      LE
» A         OpcodeSize_Current,A0
00001FB2  10BC 0000               1418      MO
» VE.B      #OPCODESIZE_BYTE,(A0)
00001FB6  205F                    1419      MO
» VE.L      (SP)+,A0
00001FB8  4E75                    1420      RT
» S
00001FBA                          1421
```

```
00001F76        MOVE.L  A0, -(A7)

00001F78        LEA             $000021ED, A0

00001F7E        MOVE.B  #0001, (A0)

00001F82        LEA             $000021EE, A0

00001F88        MOVE.B  #0000, (A0)

00001F8C        MOVEA.L (A7)+, A0

00001F8E        RTS

00001F90        MOVE.L  A0, -(A7)

00001F92        LEA             $000021EE, A0

00001F98        MOVE.B  #0001, (A0)

00001F9C        LEA             $000021ED, A0

00001FA2        MOVE.B  #0000, (A0)

00001FA6        MOVEA.L (A7)+, A0

00001FA8        RTS

00001FAA        MOVE.L  A0, -(A7)

00001FAC        LEA             $000021EC, A0

00001FB2        MOVE.B  #0000, (A0)

00001FB6        MOVEA.L (A7)+, A0

00001FB8        RTS
```

```
00001FBA                          1422
00001FBA                          1423  Opcode
» Size_SetToWord
00001FBA  2F08                    1424       MO    00001FBA        MOVE.L  A0, -(A7)
» VE.L      A0,-(SP)
00001FBC  41F9 000021EC           1425       LE    00001FBC        LEA             $000021EC, A0
» A         OpcodeSize_Current,A0
00001FC2  10BC 0001               1426       MO    00001FC2        MOVE.B  #0001, (A0)
» VE.B      #OPCODESIZE_WORD,(A0)
00001FC6  205F                    1427       MO    00001FC6        MOVEA.L (A7)+, A0
» VE.L      (SP)+,A0
00001FC8  4E75                    1428       RT    00001FC8        RTS
» S
00001FCA                          1429
00001FCA                          1430
00001FCA                          1431
00001FCA                          1432  Opcode
» Size_SetToLong
00001FCA  2F08                    1433       MO    00001FCA        MOVE.L  A0, -(A7)
» VE.L      A0,-(SP)
00001FCC  41F9 000021EC           1434       LE    00001FCC        LEA             $000021EC, A0
» A         OpcodeSize_Current,A0
00001FD2  10BC 0002               1435       MO    00001FD2        MOVE.B  #0002, (A0)
» VE.B      #OPCODESIZE_LONG,(A0)
00001FD6  205F                    1436       MO    00001FD6        MOVEA.L (A7)+, A0
» VE.L      (SP)+,A0
00001FD8  4E75                    1437       RT    00001FD8        RTS
» S
00001FDA                          1438
00001FDA                          1439  Opcode
» Size_SetToElse
00001FDA  2F08                    1440       MO    00001FDA        MOVE.L  A0, -(A7)
» VE.L      A0,-(SP)
00001FDC  41F9 000021EC           1441       LE    00001FDC        LEA             $000021EC, A0
» A         OpcodeSize_Current,A0
00001FE2  10BC 0003               1442       MO    00001FE2        MOVE.B  #0003, (A0)
» VE.B      #OPCODESIZE_ELSE,(A0)
00001FE6  205F                    1443       MO    00001FE6        MOVEA.L (A7)+, A0
» VE.L      (SP)+,A0
00001FE8  4E75                    1444       RT    00001FE8        RTS
» S
00001FEA                          1445
00001FEA                          1446
00001FEA                          1447
00001FEA                          1448
00001FEA                          1449  Opcode
» Size_GetSize
00001FEA  2F08                    1450       MO    00001FEA        MOVE.L  A0, -(A7)
» VE.L      A0,-(SP)
00001FEC  41F9 000021EC           1451       LE    00001FEC        LEA             $000021EC, A0
» A         OpcodeSize_Current,A0
00001FF2  1010                    1452       MO    00001FF2        MOVE.B  (A0), D0
```

```
» VE.B      (A0),D0
00001FF4  205F                    1453      MO    00001FF4        MOVEA.L (A7)+, A0
» VE.L      (SP)+,A0
00001FF6  4E75                    1454      RT    00001FF6        RTS
» S
00001FF8                          1455            00001FF8        JSR             $1FEA
00001FF8                          1456
00001FF8                          1457  Opcode
» Size_IsByte
00001FF8  4EB8 1FEA               1458      JS
» R         OpcodeSize_GetSize
00001FFC  B03C 0000               1459      CM    00001FFC        CMP.B   #0000, D0
» P.B       #OPCODESIZE_BYTE,D0
00002000  6700 002E               1460      BE
» Q         Flag_IsTrue
00002004  6000 0022               1461      BR
» A         Flag_IsFalse
00002008                          1462            00002000        DATA    6700
»
00002008                          1463            00002002        ORI.B   #$6000, $00224EB8
00002008                          1464
00002008                          1465  Opcode
» Size_IsWord
00002008  4EB8 1FEA               1466      JS
» R         OpcodeSize_GetSize
0000200C  B03C 0001               1467      CM    0000200A        MOVE.B  $B03C0001, $6700001E
» P.B       #OPCODESIZE_WORD,D0
00002010  6700 001E               1468      BE
» Q         Flag_IsTrue
00002014  6000 0012               1469      BR
» A         Flag_IsFalse
00002018                          1470            00002014        BRA     0012
00002018                          1471            00002018        JSR             $1FEA
00002018                          1472
00002018                          1473  Opcode
» Size_IsLong
00002018  4EB8 1FEA               1474      JS
» R         OpcodeSize_GetSize
0000201C  B03C 0002               1475      CM    0000201C        CMP.B   #0002, D0
» P.B       #OPCODESIZE_LONG,D0
00002020  6700 000E               1476      BE
» Q         Flag_IsTrue
00002024  6000 0002               1477      BR
» A         Flag_IsFalse
00002028                          1478            00002020        DATA    6700
00002028                          1479            00002022        ORI.B   #$6000, A6
00002028                          1480
00002028                          1481  Flag_I
» sFalse
00002028  4280                    1482      CL    00002026        ORI.B   #$4280, D2
» R.L       D0
0000202A  103C 0000               1483      MO    0000202A        MOVE.B  #0000, D0
```

```
» VE.B      #IsFalse,D0
0000202E  4E75                    1484      RT    0000202E        RTS
» S
00002030                          1485
00002030                          1486
00002030                          1487
00002030                          1488  Flag_I
» sTrue
00002030  4280                    1489      CL    00002030        NEG.L   D0
» R.L       D0
00002032  103C 0001               1490      MO    00002032        MOVE.B  #0001, D0
» VE.B      #IsTrue,D0
00002036  4E75                    1491      RT    00002036        RTS
» S
00002038                          1492
00002038                          1493
00002038                          1494
00002038                          1495  ------
» -------------- end include -----------------
» ---
00002038                          1496      ;I
» NCLUDE    'OPCODE_OR.x68'
00002038  4E75                    1497      RT    00002038        RTS
» S
0000203A                          1498
0000203A                          1499
0000203A                          1500  ******
» ****************************************
» ******************************
0000203A                          1501  * Meth
» od Name:  AppendOutput
0000203A                          1502  * Desc
» ription:  This file will be included in Main
» .X68 disassembler program
0000203A                          1503  *
»   When opcodes is being broken down we will
» be actively appending the
0000203A                          1504  *
»   output string to prepare for TRAPTask13
0000203A                          1505  *
0000203A                          1506  * Prec
» onditions: Appending String is in A2 and mus
» t end in null
0000203A                          1507  *
»            string constants are in Gbl_CONST
» .X68
0000203A                          1508  *
0000203A                          1509  ******
» ****************************************
» ******************************
0000203A                          1510  Append
» Output:
```

```
0000203A                          1511      IN
» CLUDE     'AppendOutput.X68'
0000203A                          1512  SaveIt
» Fam REG A1-A3/D1-D2
0000203A                          1513      ;S
» ave context
0000203A  48E7 6070               1514      MO
» VEM.L SaveItFam, -(SP)
0000203E                          1515
0000203E                          1516
»   ;Clear the registers I plan on using
0000203E  4241                    1517      CL
» R     D1
00002040  4242                    1518      CL
» R     D2
00002042                          1519
00002042                          1520
00002042                          1521      ;L
» oad Current Output into A1
00002042  43F9 0000220B           1522      LE
» A OUTPUT, A1
00002048                          1523
00002048                          1524      ;C
» ounter D1, Must be set to -1 to deal with nu
» ll termination of strings
00002048  123C 00FF               1525      MO
» VE.B    #-1, D1
0000204C                          1526
0000204C                          1527      ;S
» et D2 as Current String Length
0000204C  47F9 0000220A           1528      LE
» A      CURRENT_STR_LENGTH, A3
00002052  1413                    1529      MO
» VE.B   (A3), D2
00002054                          1530
00002054                          1531      ;G
» et to the current position of the string
00002054  D2C2                    1532      AD
» D     D2, A1

00002056                          1533
00002056                          1534      ;N
» ow that we are in position, start appending
» the string in A2
00002056                          1535  APPEND
» LOOP:
00002056  5201                    1536      AD
» DI.B  #1, D1                      ;Increment
»  Current Length by 1
00002058  12DA                    1537      MO
» VE.B  (A2)+, (A1)+                ;Move the
» Byte in A2 to our Output
```

```
0000203A        MOVEM   .L      A2/A0/7//D6/5/
» /D2/1//D0/, -(A7)
0000203E        NEG.W   D1

00002040        NEG.W   D2

00002042        LEA             $0000220B, A1

00002048        MOVE.B  #000F, D1

0000204C        LEA             $0000220A, A3

00002052        MOVE.B  (A3), D2

00002054        ADDA.W  D2, A1

00002056        ADDQ.B  #1, D1

00002058        MOVE.B  (A2)+, (A1)+
0000205A        DATA    66FA
```

```
0000205A  66FA                    1538      BN
» E     APPENDLOOP                  ; If zero
» has not been reached, Loop Back
0000205C                          1539
0000205C                          1540
0000205C                          1541      ;
» Update the current string length with the co
» unter
0000205C  D313                    1542      AD
» D.B     D1, (A3)
0000205E                          1543
0000205E                          1544      ;
» restore context
0000205E  4CDF 0E06               1545      MO
» VEM.L (SP)+, SaveItFam
00002062                          1546
00002062  4E75                    1547      RT
» S


00002064                          1548
00002064                          1549
00002064                          1550
00002064                          1551  ------
» -------------- end include -----------------
» ---
00002064  4E75                    1552      RT
» S
00002066                          1553
00002066                          1554  ******
» ****************************************
» ******************************
00002066                          1555  * Meth
» od Name:  PrintASCIIWord
00002066                          1556  * Desc
» ription:  This function will take a 16 bit w
» ord as input and convert it to
00002066                          1557  *
»   ASCII format.  It will then call AppendOut
» put to add it to the print buffer.
00002066                          1558  *
00002066                          1559  * Prec
» onditions: Current word is in D6
00002066                          1560  *
00002066                          1561  ******
» ****************************************
» ******************************
00002066                          1562  PrintA
» SCIIWord:
00002066  48E7 3008               1563      MO
» VEM.L     D2-D3/A4, -(SP)     ; MOVEM all re
» gisters used
```

```
0000205C        ADD.B   D1, (A3)

0000205E        MOVEM   .W      (A7)+,

00002060        CMPI.B  #$4E75, D6
00002064        RTS

00002066        MOVEM   .L      A2/1/7//D6/5//
» D4/3//1//D0/, -(A7)
0000206A        MOVEA.W $00002584, A4

00002070        MOVE.W  D6, D3
00002072        MULS    #$F000, D3


00002076        ROL     .W      D4, D3
```

```
0000206A  3879 00002584           1564      MO
» VEA       Hex_StrArray, A4    ; Move the bas
» e array into A4
00002070                          1565
00002070  3606                    1566      MO
» VE.W      D6,D3                ; move word i
» nto temp register
00002072  C67C F000               1567      AN
» D         #MASK_OPCODE, D3       ; mask fir
» st 4 bits
00002076  E95B                    1568      RO
» L         #4, D3                 ; rotate l
» eft 4 so bits are in least significant posit
» ion
00002078  C7FC 0002               1569      MU
» LS        #2, D3               ; multiply by
» 2 for correct array offset
0000207C  45F4 3000               1570      LE
» A         (A4,D3), A2       ; move ASCII cha
» r into A2
00002080  4EB8 203A               1571      JS
» R         AppendOutput
00002084                          1572
00002084  3606                    1573      MO
» VE.W      D6, D3               ; move word in
» to temp register
00002086  C67C 0F00               1574      AN
» D         #MASK_111098, D3       ; mask sec
» ond 4 bits
0000208A  E15B                    1575      RO
» L         #8, D3               ; rotate right
»  8 so bits are in least significant position
0000208C  C7FC 0002               1576      MU
» LS        #2, D3               ; multiply by
» 2 for correct array offset
00002090  45F4 3000               1577      LE
» A         (A4,D3), A2         ; move ASCII c
» har into A2
00002094  4EB8 203A               1578      JS
» R         AppendOutput
00002098                          1579
00002098  3606                    1580      MO
» VE.W      D6, D3               ; move word i
» nto temp register
0000209A  C67C 00F0               1581      AN
» D         #MASK_7654, D3       ; mask third
» 4 bits
0000209E  E85B                    1582      RO
» R         #4, D3               ; rotate right
»  4 so bits are in least significant position
000020A0  C7FC 0002               1583      MU
» LS        #2, D3               ; multiply by
```

```
00002078        MULS    #$0002, D3
0000207C        LEA             #$4EB8, A2
00002082        MOVE.L  $3606C67C, D0
00002088        BCLR    D7, D0
```

```
» 2 for correct array offset
000020A4  45F4 3000               1584      LE
» A         (A4,D3), A2       ; move ASCII cha
» r into A2
000020A8  4EB8 203A               1585      JS
» R         AppendOutput
000020AC                          1586
000020AC  3606                    1587      MO
» VE.W      D6, D3               ; move word i
» nto temp register
000020AE  C67C 000F               1588      AN
» D         #MASK_3210, D3       ; mask last 4
»  bits
000020B2  C7FC 0002               1589      MU
» LS        #2, D3              ; multiply by
» 2 for correct array offset
000020B6  45F4 3000               1590      LE
» A         (A4,D3), A2       ; move ASCII cha
» r into A2
000020BA  4EB8 203A               1591      JS
» R         AppendOutput
000020BE                          1592
000020BE  4CDF 100C               1593      MO
» VEM.L     (SP)+, D2-D3/A4     ; return regis
» ters to their previous state
000020C2  4E75                    1594      RT
» S                                ; exit
000020C4                          1595
000020C4                          1596
000020C4                          1597  ******
» ****************************************
» ******************************
000020C4                          1598  * Meth
» od Name:  PrintASCIILong
000020C4                          1599  * Desc
» ription:  This function will take a 32 bit w
» ord as input and convert it to
000020C4                          1600  *
»   ASCII format.  It will then call AppendOut
» put to add it to the print buffer.
000020C4                          1601  *
000020C4                          1602  * Prec
» onditions: Current word is in D6
000020C4                          1603  *
000020C4                          1604  ******
» ****************************************
» ******************************
000020C4                          1605
000020C4                          1606  PrintA
» SCIILong:
000020C4  48E7 0300               1607      MO
» VEM.L     D6-D7, -(SP)        ; MOVEM all re
```

```
0000208A        ROL     .W      D0, D3
0000208C        MULS    #$0002, D3

00002090        LEA             #$4EB8, A2
00002096        MOVE.L  $3606C67C, D0

0000209C        ORI.B   #$E85B, $C7FC

000020A2        ORI.B   #$45F4, D2

000020A6        MOVE.W  D0, D0

000020A8        JSR             $203A

000020AC        MOVE.W  D6, D3
000020AE        MULS    #$000F, D3

000020B2        MULS    #$0002, D3
```

```
» gisters used
000020C8  1E3C 0010              1608      MO
» VE.B      #16, D7             ; move 16 into
»  D7 for rotation
000020CC  EFBE                   1609      RO
» L.L       D7, D6              ; rotate D6 by
»  16 bits
000020CE  4EB8 2066              1610      JS
» R         PrintASCIIWord
000020D2  EFBE                   1611      RO
» L.L       D7, D6
000020D4  4EB8 2066              1612      JS
» R         PrintASCIIWord
000020D8  4CDF 00C0              1613      MO
» VEM.L     (SP)+, D6-D7        ; MOVEM all re
» gisters used
000020DC  4E75                   1614      RT
» S
```

```
000020B6        LEA             #$4EB8, A2
```

```
000020BC        MOVE.L  $4CDF100C, D0
```

```
000020C2        RTS

000020C4        MOVEM   .L      A2/1//A0/7///D
» 4/3//D2/1//D0/, -(A7)
000020C8        MOVE.B  #0000, D7
000020CC        ROL     .L      #7, D6
```

```
000020DE                         1615
000020DE                         1616  ******
» ****************************************
» ****************************
000020DE                         1617  * Meth
» od Name: TrapTask13
000020DE                         1618  * Desc
» ription: Creates a file if none exists, and
» appends bytes to that file
000020DE                         1619  *   wh
» ile also echoing the written bytes to the sc
» reen.  You shouldn't need to
000020DE                         1620  *   ch
» ange this code.
000020DE                         1621  *
000020DE                         1622  * Call
» ing Convention: Callee-Saved
000020DE                         1623  *
000020DE                         1624  * Prec
» onditions & Method Input:
000020DE                         1625  *   A1
»  points to the null-terminated buffer to wri
» te (newline will be added for you)
000020DE                         1626  *
000020DE                         1627  * Post
» conditions & Output:
000020DE                         1628  *   AL
» L files that were previously open will be CL
» OSED (FileIDs will be invalid)
000020DE                         1629  *   Se
» e 'Output.txt' in directory for the results,
```

```
000020CE        JSR             $2066
```

```
000020D2        ROL     .L      #7, D6
```

```
000020D4        JSR             $2066
000020D8        MOVEM   .W      (A7)+,
```

```
»  also piped to the console
000020DE                          1630  *
000020DE                          1631  *

000020DE                          1632  *  A2
» holds a pointer to null terminated string to
»  write (input)
000020DE                          1633  *  A3
» points to the null-terminated file name
000020DE                          1634  *  D3
» holds the number of bytes already in the fil
» e to write
000020DE                          1635  *
000020DE                          1636  *  D5
» holds number of bytes to write
000020DE                          1637  ******
» ****************************************
» ****************************
000020DE                          1638  toSave
»  REG D0-D5/A2-A3
000020DE                          1639  TrapTa
» sk13:
000020DE                          1640      **
» ****************************************
» ********************
000020DE                          1641      *
» Method initialization, regsiter spilling, pa
» rameter saving, etc.
000020DE                          1642      **
» ****************************************
» **********************
000020DE  48E7 FC30               1643      MO
» VEM.L toSave, -(SP)   ;  Callee-Saved, so sa
» ve and restore
000020E2                          1644
000020E2  2449                    1645      MO
» VEA.L A1, A2 ; save this buffer to write
000020E4  47F9 000021D8           1646      LE
» A outFilename, A3  ; save this for later, to
» o
000020EA                          1647
000020EA  303C 0032               1648      MO
» VE #50,d0
000020EE  4E4F                    1649      TR
» AP #15 ; close all files, suggested to begin
»  any IO
000020F0                          1650      **
» ****************************************
» ********************
000020F0                          1651      *
» End Method Init
000020F0                          1652      **
```

```
000020DA        ORI.B   #$4E75, D0
000020DE        MOVEM   .L      D6/5//D4/D2/1/
» /D0/, -(A7)

000020E2        MOVEA.L A1, A2

000020E4        LEA             $000021D8, A3
000020EA        MOVE.W  #$0032, D0

000020EE        JSR             A7
```

```
» ****************************************
» ********************
000020F0                        1653
000020F0                        1654      **
» ****************************************
» ****************************************
000020F0                        1655      *
» Calculate the number of bytes to write by se
» arching for the null in the target buffer A0
000020F0                        1656      **
» ****************************************
» ****************************************
000020F0  4285                  1657      CL
» R.L D5 *D5 is now the number of bytes to wri
» te
000020F2                        1658  nullLo
» op:
000020F2  1019                  1659      MO
» VE.B (A1)+, D0
000020F4  0C00 0000             1660      CM
» PI.B #0,D0  * compare to null
000020F8  6700 0006             1661      BE
» Q findNullLoopDone
000020FC  5245                  1662      AD
» DI.W #1, D5
000020FE  60F2                  1663      BR
» A nullLoop
00002100                        1664
00002100                        1665  findNu
» llLoopDone:
00002100  224B                  1666      MO
» VEA.L A3, A1 * reset A1 so it points to the
» file to write to (to open, next)
00002102                        1667
00002102                        1668      ;c
» heck if file exists, and open with task 51 i
» f so, otherwise 52
00002102                        1669      ;(
» precondition here is A1 points to the null-t
» erminated filename )
00002102  103C 0033             1670      MO
» VE.B #51, D0      ;open file (task 51 is exi
» sting, 52 is new)
00002106  4E4F                  1671      TR
» AP #15
00002108                        1672
00002108                        1673      if
» .w D0 <NE> #0 then.s         ; if file error
»  (404, not found)
00002108  B07C 0000             1674s     CM
» P.W   #0,D0
0000210C  6706                  1675s     BE

000020F0        NEG.L   D5
000020F2        MOVE.B  (A1)+, D0
000020F4        CMPI.B  #$0000, D0
000020F8        DATA    6700
000020FA        ORI.B   #$5245, D6
000020FE        BRA     00F2
00002100        MOVEA.L A3, A1
00002102        MOVE.B  #0003, D0
00002106        JSR             A7
00002108        CMP.W   #$0000, D0
0000210C        DATA    6706
0000210E        MOVE.B  #0004, D0
```

```
» Q.S   _00000000
0000210E  103C 0034               1676
»   MOVE.B #52, D0             ; open new file
»  (52 is new)
00002112  4E4F                    1677          00002112        JSR             A7
»   TRAP #15
00002114                          1678      en  00002114        NEG.L   D3
» di
00002114                          1679s _00000
» 000
00002114                          1680          00002116        DATA    7401
00002114                          1681      **
» ****************************************
» ****************************************
» ***************
00002114                          1682      *
» Seek to END of FILE by counting the number o
» f bytes, closing, reopening, then seeking.
00002114                          1683      *
»    (first, count number of bytes already in
» the file to obtain seek position)
00002114                          1684      **
» ****************************************
» ****************************************
» ***************
00002114  4283                    1685      Cl
» r.L D3   ;TODO: reg save, D3 is now our coun
» t of bytes read
00002116  7401                    1686      MO
» VE.L #1, D2 ; read one byte at a time
00002118  43F9 000021E3           1687      LE  00002118        LEA             $000021E3, A1
» A byteRead, A1
0000211E                          1688          0000211E        MOVE.B  #0005, D0
0000211E                          1689  countL
» oop:
0000211E  103C 0035               1690      MO
» VE.B #53, D0 ; try to read one byte (TODO: M
» OVE out of loop)
00002122  4E4F                    1691      TR
» AP #15
00002124                          1692          00002122        JSR             A7
00002124  0C40 0001               1693      CM  00002124        CMPI.W  #$0001, D0
» PI.W #1,D0  ;1 == EOF
00002128  6700 0006               1694      BE  00002128        DATA    6700
» Q countDone
0000212C  5243                    1695      AD  0000212A        ORI.B   #$5243, D6
» DI #1, D3
0000212E  60EE                    1696      BR  0000212E        BRA     00EE
» A countLoop
00002130                          1697
00002130                          1698  countD
» one:
```

00002130		1699	*		
» close this file					
00002130	303C 0038	1700	M	00002130	MOVE.W #$0038, D0
» OVE #56,d0					
00002134	4E4F	1701	T		
» RAP #15					
00002136		1702		00002134	JSR A7
00002136		1703	*		
» reopen the target file					
00002136	224B	1704	M	00002136	MOVEA.L A3, A1
» OVE.L A3,A1					
00002138	303C 0033	1705	M	00002138	MOVE.W #$0033, D0
» OVE #51, D0					
0000213C	4E4F	1706	T	0000213C	JSR A7
» RAP #15					
				0000213E	MOVE.L D3, D2
0000213E		1707		00002140	MOVE.W #$0037, D0
0000213E		1708	*		
» seek to right position, then continue with w					
» riting					
0000213E	2403	1709	MO		
» VE.L D3, D2 ; MOVE the number of bytes found					
» in the file to D2					
00002140	303C 0037	1710	MO		
» VE #55, D0 ; position file task					
00002144	4E4F	1711	TR	00002144	JSR A7
» AP #15					
00002146		1712		00002146	MOVE.L D5, D2
00002146		1713	**		
» **********					
» **********					
00002146		1714	*		
» Actually write the buffer to the file, after					
» caculating the number of bytes					
00002146		1715	*		
» to write and after seeking to the right loc					
» ation in the file for append					
00002146		1716	**		
» **********					
» **********					
00002146		1717		00002148	MOVEA.L A2, A1
00002146	2405	1718	MO	0000214A	MOVE.B #0006, D0
» VE.L D5, D2 ; restore this for the actually					
» writing the buffer					
00002148		1719	;		
» assumes A0 hasnt changed since handed to thi					
» s method					
00002148	224A	1720	MO		
» VEA.L A2, A1 ; load the address of the buffe					
» r we want to write to disk					
0000214A		1721	;		
» assumes file ID is still stored in D1.L					

```
0000214A  103C 0036              1722      MO
» VE.B #54, D0 ; subtask 54 is write to open f
» ile (append, or?), assumes D2 holds # of byt
» es
0000214E  4E4F                   1723      TR
» AP #15
00002150                         1724
00002150                         1725      ;
» add a newline to the file output
00002150  43F9 000021BC          1726      LE
» A NEWLINE, A1
00002156  103C 0036              1727      MO
» VE.B #54, D0
0000215A  143C 0002              1728      MO
» VE.B #2,D2  ; kills # of bytes to write from
»  input param
0000215E  4E4F                   1729      TR
» AP #15
00002160                         1730
00002160                         1731      ;
» finally, close only this file
00002160  103C 0038              1732      MO
» VE.B #56, D0 ; close file task
00002164  4E4F                   1733      TR
» AP #15
00002166                         1734
00002166                         1735      ;
» report to screen
00002166  224A                   1736      MO
» VEA.L A2, A1 ; load the address of the buffe
» r we want to write to disk & screen
00002168  103C 000D              1737      MO
» VE.B #13, D0
0000216C  4E4F                   1738      TR
» AP #15
0000216E                         1739
0000216E                         1740      ;
» restore context
0000216E  4CDF 0C3F              1741      MO
» VEM.L (SP)+, toSave
00002172                         1742
00002172  4E75                   1743      RT
» S
00002174                         1744
00002174                         1745
00002174                         1746  *-----
» ----------------------------------------
» --------------------------------
00002174                         1747  * Meth
» od Name: AsciiToHex
00002174                         1748  * Writ
» ten by : Berger, Modified by Nash
```

```
0000214E        JSR             A7
00002150        LEA             $000021BC, A1
00002156        MOVE.B  #0006, D0
0000215A        MOVE.B  #0002, D2
0000215E        JSR             A7
00002160        MOVE.B  #0008, D0
00002164        JSR             A7
00002166        MOVEA.L A2, A1
00002168        MOVE.B  #000D, D0
0000216C        JSR             A7
0000216E        MOVEM   .W      (A7)+,
00002170        CMPI.B  #$4E75, $48E78000
00002178        NEG.L   D7
```

```
00002174                            1749  * Date
»        : 3/1/2019
00002174                            1750  * Desc
» ription: Converts chars '0'-'9' and 'a'-'f'
» to 0-9,a-F
00002174                            1751  *
»          Transforms/unpacks 8 chars (8b each
» ) pointed to by A1 into
00002174                            1752  *
»          its (4b each) equivalent hex value
00002174                            1753  *
00002174                            1754  *  Pre
» conditions & Input
00002174                            1755  *
»   A1 (input) points to a memory buffer holdi
» ng 8 ascii chars (not null-terminated)
00002174                            1756  *
»   This function calls another function (stri
» p_ascii)
00002174                            1757  *
00002174                            1758  *  Pos
» tconditions & Output
00002174                            1759  *
»   D7 (output) holds the converted value
00002174                            1760  *
»   Caller-Saved : D0 is temp, D6 is a loop va
» r
00002174                            1761  *-----
» ----------------------------------------
» --------------------------------
00002174                            1762  AsciiT
» oHexRegList REG D0,D6
00002174                            1763  AsciiT
» oHex
00002174  48E7 8000                 1764      MO
» VEM.L asciiToHexRegList, -(SP)  *save contex
» t
00002178  4287                      1765      CL
» R.L D7 * cLEAr our return value
0000217A  7C08                      1766      MO
» VE.L #8, D6 ; and set up our loop counter
0000217C                            1767
0000217C                            1768  chrLoo
» p
0000217C  1019                      1769      MO
» VE.B (A1)+,D0 * Get the first byte
0000217E  4EB9 00002196             1770      js
» r strip_ascii * Get rid of the ascii code
»
00002184  8E40                      1771      OR
» .W D0,D7 * Load the bits into D7
00002186                            1772
```

```
0000217A        DATA    7C08
0000217C        MOVE.B  (A1)+, D0
0000217E        JSR             $00002196
```

```
00002186  5306                    1773      su
» bI.B #1,D6  *decrement our loop variable
00002188  6700 0006               1774      BE
» Q chrDone   *skip shifting if we are done
0000218C                          1775
0000218C  E987                    1776      AS
» L.L #4,D7 * shift left 4 bits to prepare for
»  next byte
0000218E  60EC                    1777      BR
» A chrLoop
00002190                          1778
00002190                          1779  chrDon
» e
00002190  4CDF 0001               1780      MO
» VEM.L (SP)+,asciiToHexRegList
00002194  4E75                    1781      RT
» S
00002196                          1782
00002196                          1783
00002196                          1784  ******
» ****************************************
» ********************
00002196                          1785  * SUBR
» OUTINE: strip_ascii
00002196                          1786  * reMO
» VE the ascii code from the digits 0-9,a-f, o
» r A-F
00002196                          1787  * Inpu
» t Parameters: <D0> = ascii code
00002196                          1788  *
00002196                          1789  * Retu
» rn parameters: D0.B = number 0...F, returned
»  as 00...0F
00002196                          1790  * Regi
» sters used internally: D0
00002196                          1791  * Assu
» mptions: D0 contains $30-$39, $41-$46 or $61
» -66
00002196                          1792  *
00002196                          1793  ******
» ****************************************
» **********************
00002196                          1794  strip_
» ascii
00002196  B03C 0039               1795
» CMP.B #$39,D0 * Is it in range of 0-9?
0000219A  6F00 001A               1796
» BLE sub30 * Its a number
0000219E  B03C 0046               1797
» CMP.B #$46,D0 * Is is A...F?
000021A2  6F00 000A               1798
» BLE sub37 * Its A...F
```

```
00002184        OR.W    D0, D7
00002186        SUBQ.B  #1, D6
00002188        DATA    6700
0000218A        ORI.B   #$E987, D6
0000218E        BRA     00EC
00002190        MOVEM   .W      (A7)+,
00002192        ORI.B   #$4E75, D1
00002196        CMP.B   #0009, D0
```

```
000021A6  0400 0057               1799
» SUB.B #$57,D0 * Its a...f
000021AA  6000 000E               1800
» BRA ret_sa * Go back
000021AE  0400 0037               1801  sub37
» SUB.B #$37,D0 * Strip 37
000021B2  6000 0006               1802
» BRA ret_sa * Go back
000021B6  0400 0030               1803  sub30
» SUB.B #$30,D0 * Strip 30
000021BA  4E75                    1804  ret_sa
»  RTS * Go back
000021BC                          1805
```

```
0000219A        DATA    6F00
0000219C        ORI.B   #$B03C, (A2)+
000021A0        ORI.W   #$6F00, D6
000021A4        ORI.B   #$0400, A2
000021A8        ORI.W   #$6000, (A7)
000021AC        ORI.B   #$0400, A6
```

```
000021BC                          1806
000021BC                          1807  * Requ
» ired variables and constants go here for you
» r Disassembler
000021BC  =0000000D               1808  CR
»               EQU $0D
000021BC  =0000000A               1809  LF
»               EQU $0A
000021BC= 0D 0A 00                1810  NEWLIN
» E             DC.B CR,LF,0
000021BF= 4D 4F 56 45 2E 4C 20    1811  MSG1
»               DC.B 'MOVE.L D4,D5',0
          44 34 2C 44 35 00
000021CC= 41 44 44 2E 42 20 44    1813  MSG2
»               DC.B 'ADD.B D0,D1',0
          30 2C 44 31 00
000021D8= 4F 75 74 70 75 74 2E    1815  outFil
» ename         DC.B 'Output.txt',0
          74 78 74 00
000021E3                          1817  byteRe
» ad            DS.B 1
000021E4= 31 41 30 30 31 41 30    1818  ascii_
» val           DC.B $31,$41,$30,$30,$31,$41,$
» 30,$30 * Test value $1A001A00
          30
000021EC                          1820      IN
» CLUDE 'Gbl_CONST.X68'
000021EC                          1821
000021EC                          1822
000021EC                          1823
000021EC                          1824
000021EC  =00000002               1825  ARRAY_
» ELEMENT_WIDTH     EQU     2
000021EC                          1826
000021EC                          1827
```

```
000021B0        ORI.B   #$6000, $00060400
000021B8        ORI.B   #$4E75, $0D0A
000021BE        ORI.W   #$4F56, A5
000021C2        LEA             $4C204434, A2
000021C8        MOVEA.L D4, A6
000021CA        MOVE.W  D0, -(A2)
000021CC        LEA             D4, A0
```

```
000021EC  =00000001               1828  IsTrue
»       EQU     1
000021EC  =00000000               1829  IsFals
» e     EQU     0
000021EC                          1830
000021EC= 01                      1831  Opcode
» Size_Current DC.B    1
000021ED                          1832
000021ED  =00000000               1833  OPCODE
» SIZE_BYTE    EQU 0
000021ED  =00000001               1834  OPCODE
» SIZE_WORD    EQU 1
000021ED  =00000002               1835  OPCODE
» SIZE_LONG    EQU 2
000021ED  =00000003               1836  OPCODE
» SIZE_ELSE    EQU 3
000021ED                          1837

000021ED                          1838
000021ED                          1839  EA_IsM
» Xn            DS.B    1
000021EE                          1840  EA_IsX
» nM            DS.B    1
000021EF                          1841
000021EF  =00000009               1842  EA_REG
» ISTER_ASRL_DISTANCE  EQU 9
000021EF  =00000003               1843  EA_MOD
» E_ASRL_DISTANCE      EQU 3
000021EF                          1844
000021EF                          1845
000021EF= 43 6F 6E 66 69 67 2E    1846  inFile
»           DC.B 'Config.cfg',0
          63 66 67 00
000021FA                          1848  BEGIN_
» ADDRESS_STR   DS.B 8
00002202                          1849  END_AD
» DRESS_STR     DS.B 8
0000220A                          1850
```

```
000021CE        NEG.B   $42204430

000021D4        MOVEA.L D4, A6
000021D6        MOVE.W  D0, -(A0)

000021D8        LEA             $74707574, A7
000021DE        MOVEA.L #$000A, A7
000021E4        MOVE.W  D1, $3030
000021E8        MOVE.W  D1, $3030

000021EC        BCLR    D1, D1
000021EE        ORI.W   #$6F6E, D3

000021F2        DATA    6669
000021F4        DATA    672E

000021F6        DATA    6366

000021F8        DATA    6700
000021FA        MOVE.W  $3030, D0
000021FE        MOVE.W  $3030, -(A0)
00002202        MOVE.W  $3030, D0
00002206        MOVE.W  $41301130, D1
0000220C        MOVE.W  $3032, D0
00002210        MOVE.W  $30094D4F, D1
00002216        ADDQ.W  #3, D5
00002218        MOVEA.L (A7), A7
0000221A        MOVEM   .W      A1,
0000221C        BCLR    D4, D1
0000221E        MOVE.W  D0, -(A0)
00002220        MOVEA.L D1, A4
00002222        ORI.B   #$0020, $44310030
0000222A        MOVE.W  D0, D0
```

```
                                                  1851
0000220A                                          1852
0000220A  =00070000                               1853  STACK_
» LOCATION      EQU     $0070000
0000220A                                          1854

0000220A                                          1855




0000220A                                          1856
0000220A  =00000000                               1857  EMPTY_
» ADDRESS   EQU    $00000000
0000220A                                          1858
0000220A  =00000001                               1859  BYTE_L
» ENGTH     EQU     1
0000220A  =00000002                               1860  WORD_L
» ENGTH     EQU     2
0000220A  =00000004                               1861  LONG_L
» ENGTH     EQU     4




0000220A                                          1862
0000220A  =00000009                               1863  TAB
»   EQU     $09
0000220A                                          1864
0000220A  =00000400                               1865  FILE_D
» EFAULT_READ_BYTES  EQU 1024

0000220A                                          1866
0000220A                                          1867  CURREN
» T_STR_LENGTH  DS.B    1
0000220B  =00000028                               1868  OUTPUT
» _LENGTH       EQU     40
0000220B                                          1869  OUTPUT
»               DS.B    OUTPUT_LENGTH
00002233                                          1870  OUTPUT
» _TEMP         DS.B    OUTPUT_LENGTH
0000225B                                          1871
0000225C= 5375 6363 6573 7300     1872  ERROR_
» CODE_FILE_0  DC.W    'Success',0
```

```
0000222C        ORI.B   #$2F2C, $202D
00002232        MOVEA.L D1, A4
00002234        MOVE.W  $00302F2C, -(A3)
0000223A        MOVE.L  $28413729, D0
00002240        ORI.B   #$FFFF, $FFFFFFFF
00002248        SIMHALT
0000224A        SIMHALT
0000224C        SIMHALT
0000224E        SIMHALT
00002250        SIMHALT
00002252        SIMHALT

00002254        SIMHALT
00002256        SIMHALT
00002258        SIMHALT
0000225A        SIMHALT
0000225C        SUBQ.W  #1, $63636573
00002262        DATA    7300
00002264        ORI.B   #$454F, D0
00002268        NEG.B   -(A0)
0000226A        LEA             $636F756E, A2

00002270        DATA    7465
00002272        DATA    7265

00002274        DATA    6400

00002276        ORI.B   #$4572, D0

0000227A        DATA    726F
0000227C        DATA    7200
0000227E        ORI.B   #$4669, D0
00002282        BGE     0065
00002284        MOVEA.L $73205265, A0

0000228A        DATA    6164
0000228C        MOVEA.L $6E6C7900, A0

00002292        ORI.B   #$5375, D0
00002296        DATA    6363




00002298        BCS     0073
```

```
          0000
00002266= 454F 4620 456E 636F     1874  ERROR_
» CODE_FILE_1  DC.W    'EOF Encountered',0
          756E 7465 7265 6400
          0000
00002278= 4572 726F 7200 0000     1877  ERROR_
» CODE_FILE_2  DC.W    'Error',0
00002280= 4669 6C65 2069 7320     1878  ERROR_
» CODE_FILE_3  DC.W    'File is Read only',0
          5265 6164 206F 6E6C
          7900 0000
00002294                          1881
00002294= 5375 6363 6573 7300     1882  FILE_E
» RROR_ARRAY   DC.W    'Success','EOF Encounte
» red','Error','File is Read only',0
          454F 4620 456E 636F
          756E 7465 7265 6400
          4572 726F 7200 4669

          6C65 2069 7320 5265
          6164 206F 6E6C 7900
          0000

000022C6                          1889
000022C6= 2C 20 00                1890  STRING
» COMMA     DC.B    ', ',0
000022C9= 23 00                   1891  STRING
» POUND     DC.B    '#',0
000022CB= 23 24 00                1892  STRING
» POUNDHEX  DC.B    '#$',0
000022CE= 2F 00                   1893  STRING
» SLASH     DC.B    '/',0

000022D0                          1894
000022D0= 2E 42 09 00             1895  STRING
» _B        DC.B    '.B',TAB,0
000022D4= 2E 57 09 00             1896  STRING
» _W        DC.B    '.W',TAB,0
000022D8= 2E 4C 09 00             1897  STRING
» _L        DC.B    '.L',TAB,0
```

```
0000229A        DATA    7300
0000229C        LEA             A7, A2
0000229E        NEG.B   -(A0)

000022A0        LEA             $636F756E, A2

000022A6        DATA    7465
000022A8        DATA    7265
000022AA        DATA    6400
000022AC        LEA             $726F7200, A2
000022B2        NEG.W   $6C652069

000022B8        DATA    7320
000022BA        ADDQ.W  #1, -(A5)
000022BC        DATA    6164
000022BE        MOVEA.L $6E6C7900, A0


000022C4        ORI.B   #$2C20, D0

000022C8        ORI.B   #$0023, -(A3)

000022CC        MOVE.L  D0, D2
000022CE        MOVE.L  D0, -(A7)
000022D0        MOVEA.L D2, A7
000022D2        BCLR    D4, D0
000022D4        MOVEA.L (A7), A7
000022D6        BCLR    D4, D0
000022D8        MOVEA.L A4, A7
000022DA        BCLR    D4, D0
000022DC        BCLR    D4, D1
000022DE        NEG.W   D4
000022E0        ORI.B   #$4144, A1
000022E4        NEG.W   D1
000022E6        ORI.B   #$4144, A1

000022EA        NEG.W   (A1)

000022EC        ORI.B   #$4153, A1
```

```
000022DC                          1898
000022DC= 09 41 44 44 00          1899  STR_AD
» D         DC.B    TAB,'ADD',0
000022E1= 09 41 44 44 41 00       1900  STR_AD
» DA        DC.B    TAB,'ADDA',0
000022E7= 09 41 44 44 51 00       1901  STR_AD
» DQ        DC.B    TAB,'ADDQ',0
000022ED= 09 41 53 4C 09 00       1902  STR_AS
» L         DC.B    TAB,'ASL',TAB,0
000022F3= 09 41 53 52 09 00       1903  STR_AS
» R         DC.B    TAB,'ASR',TAB,0
000022F9= 09 42 43 4C 52 09 00    1904  STR_BC
» LR        DC.B    TAB,'BCLR',TAB,0
00002300= 09 42 43 53 09 00       1905  STR_BC
» S         DC.B    TAB,'BCS',TAB,0
00002306= 09 42 47 45 09 00       1906  STR_BG
» E         DC.B    TAB,'BGE',TAB,0
0000230C= 09 42 4C 54 09 00       1907  STR_BL
» T         DC.B    TAB,'BLT',TAB,0
00002312= 09 42 52 41 09 00       1908  STR_BR
» A         DC.B    TAB,'BRA',TAB,0
00002318= 09 42 56 43 09 00       1909  STR_BV
» C         DC.B    TAB,'BVC',TAB,0
0000231E= 09 43 4D 50 00          1910  STR_CM
» P         DC.B    TAB,'CMP',0
00002323= 09 43 4D 50 49 00       1911  STR_CM
» PI        DC.B    TAB,'CMPI',0
00002329= 09 44 41 54 41 09 00    1912  STR_DA
» TA        DC.B    TAB,'DATA',TAB,0
00002330= 09 44 49 56 53 09 00    1913  STR_DI
» VS        DC.B    TAB,'DIVS',TAB,0
00002337= 09 45 4F 52 00          1914  STR_EO
» R         DC.B    TAB,'EOR',0
0000233C= 09 45 4F 52 49 00       1915  STR_EO
```

```
000022F0        MOVEM   .W      A1,
000022F2        ORI.B   #$4153, A1
000022F6        ADDQ.B  #1, A1
000022F8        ORI.B   #$4243, A1
000022FC        MOVEM   .W      (A2),
000022FE        BCLR    D4, D0
00002300        BCLR    D4, D2
00002302        LEA             (A3), A1
00002304        BCLR    D4, D0
00002306        BCLR    D4, D2
00002308        LEA             D5, A3
0000230A        BCLR    D4, D0
0000230C        BCLR    D4, D2
0000230E        MOVEM   .W      (A4),
00002310        BCLR    D4, D0
00002312        BCLR    D4, D2
00002314        ADDQ.W  #1, D1
00002316        BCLR    D4, D0
00002318        BCLR    D4, D2
0000231A        ADDQ.W  #3, D3

0000231C        BCLR    D4, D0

0000231E        BCLR    D4, D3

00002320        LEA             (A0), A6

00002322        ORI.B   #$434D, A1

00002326        ADDQ.W  #8, A1

00002328        ORI.B   #$4441, A1

0000232C        ADDQ.W  #2, D1

0000232E        BCLR    D4, D0

00002330        BCLR    D4, D4

00002332        LEA             (A6), A4

00002334        SUBQ.B  #1, A1

00002336        ORI.B   #$454F, A1

0000233A        ADDQ.B  #1, D0

0000233C        BCLR    D4, D5

0000233E        LEA             (A2), A7

00002340        LEA             D0, A4
```

```
» RI        DC.B    TAB,'EORI',0
00002342= 09 49 4C 4C 45 47 41    1916  STR_IL
» LEGAL     DC.B    TAB,'ILLEGAL',TAB,0
          4C 09 00
0000234C= 09 4A 53 52 09 09 00    1918  STR_JS
» R         DC.B    TAB,'JSR',TAB,TAB,0
00002353= 09 4C 45 41 09 09 00    1919  STR_LE
» A         DC.B    TAB,'LEA',TAB,TAB,0
0000235A= 09 4C 53 4C 09 00       1920  STR_LS
» L         DC.B    TAB,'LSL',TAB,0
00002360= 09 4C 53 52 09 00       1921  STR_LS
» R         DC.B    TAB,'LSR',TAB,0
00002366= 09 4D 4F 56 45 41 2E    1922  STR_MO
» VEA_B     DC.B    TAB,'MOVEA.B',TAB,0
          42 09 00
00002370= 09 4D 4F 56 45 41 2E    1924  STR_MO
» VEA_L     DC.B    TAB,'MOVEA.L',TAB,0
          4C 09 00
0000237A= 09 4D 4F 56 45 41 2E    1926  STR_MO
» VEA_W     DC.B    TAB,'MOVEA.W',TAB,0
          57 09 00
00002384= 09 4D 4F 56 45 4D 09    1928  STR_MO
» VEM       DC.B    TAB,'MOVEM',TAB,0
          00
0000238C= 09 4D 4F 56 45 4D 2E    1930  STR_MO
» VEM_L     DC.B    TAB,'MOVEM.L',TAB,0
          4C 09 00
00002396= 09 4D 4F 56 45 4D 2E    1932  STR_MO
» VEM_W     DC.B    TAB,'MOVEM.W',TAB,0
          57 09 00
000023A0= 09 4D 4F 56 45 2E 42    1934  STR_MO
» VE_B      DC.B    TAB,'MOVE.B',TAB,0
          09 00
000023A9= 09 4D 4F 56 45 2E 4C    1936  STR_MO
» VE_L      DC.B    TAB,'MOVE.L',TAB,0
```

```
00002342        BCLR    D4, A1
00002344        MOVEM   .W      A4,

00002346        LEA             D7, A2

00002348        LEA             A4, A0
0000234A        BCLR    D4, D0

0000234C        BCLR    D4, A2
0000234E        SUBQ.W  #1, (A2)

00002350        BCLR    D4, A1
00002352        ORI.B   #$4C45, A1
00002356        LEA             A1, A0
00002358        BCLR    D4, D0
0000235A        BCLR    D4, A4
0000235C        SUBQ.W  #1, A4
0000235E        BCLR    D4, D0
00002360        BCLR    D4, A4
00002362        SUBQ.W  #1, (A2)
00002364        BCLR    D4, D0
00002366        BCLR    D4, A5
00002368        LEA             (A6), A7
0000236A        LEA             D1, A2
0000236C        MOVEA.L D2, A7
0000236E        BCLR    D4, D0
00002370        BCLR    D4, A5
00002372        LEA             (A6), A7

00002374        LEA             D1, A2
00002376        MOVEA.L A4, A7

00002378        BCLR    D4, D0

0000237A        BCLR    D4, A5
0000237C        LEA             (A6), A7

0000237E        LEA             D1, A2
00002380        MOVEA.L (A7), A7

00002382        BCLR    D4, D0
00002384        BCLR    D4, A5
```

```
          09 00
000023B2= 09 4D 4F 56 45 2E 57    1938  STR_MO
» VE_W      DC.B    TAB,'MOVE.W',TAB,0
          09 00
000023BB= 09 4D 55 4C 53 09 00    1940  STR_MU
» LS        DC.B    TAB,'MULS',TAB,0
000023C2= 09 4E 45 47 00          1941  STR_NE
» G         DC.B    TAB,'NEG',0
000023C7= 09 4E 4F 50 09 09 00    1942  STR_NO
» P         DC.B    TAB,'NOP',TAB,TAB,0
000023CE= 09 4F 52 00             1943  STR_OR
»           DC.B    TAB,'OR',0
000023D2= 09 4F 52 49 00          1944  STR_OR
» I         DC.B    TAB,'ORI',0
000023D7= 09 52 4F 4C 09 00       1945  STR_RO
» L         DC.B    TAB,'ROL',TAB,0
000023DD= 09 52 4F 52 09 00       1946  STR_RO
» R         DC.B    TAB,'ROR',TAB,0
000023E3= 09 52 54 53 09 09 00    1947  STR_RT
» S         DC.B    TAB,'RTS',TAB,TAB,0
000023EA= 09 53 49 4D 48 41 4C    1948  STR_SI
» MHALT     DC.B    TAB,'SIMHALT',TAB,0
          54 09 00
000023F4= 09 53 55 42 00          1950  STR_SU
» B         DC.B    TAB,'SUB',0
000023F9= 09 53 55 42 51 00       1951  STR_SU
» BQ        DC.B    TAB,'SUBQ',0
000023FF                          1952
000023FF  =0000F000               1953  MASK_O
» PCODE     EQU     $F000
000023FF  =00000F00               1954  MASK_1
» 11098     EQU     $0F00
000023FF  =00000DC0               1955  MASK_1
» 110876    EQU     $0DC0
000023FF  =00000C00               1956  MASK_1
» 110       EQU     $0C00
000023FF  =00000E00               1957  MASK_1
» 1109      EQU     $0E00
000023FF  =00000D00               1958  MASK_1
» 1108      EQU     $0D00
000023FF  =00000800               1959  MASK_1
» 1         EQU     $0800
000023FF  =00000400               1960  MASK_1
» 0         EQU     $0400
000023FF  =00000700               1961  MASK_1
» 098       EQU     $0700
000023FF  =00000200               1962  MASK_9
```

```
00002386        LEA             (A6), A7
00002388        LEA             A5, A2
0000238A        BCLR    D4, D0
0000238C        BCLR    D4, A5
0000238E        LEA             (A6), A7
00002390        LEA             A5, A2
00002392        MOVEA.L A4, A7

00002394        BCLR    D4, D0


00002396        BCLR    D4, A5
00002398        LEA             (A6), A7


0000239A        LEA             A5, A2
0000239C        MOVEA.L (A7), A7

0000239E        BCLR    D4, D0

000023A0        BCLR    D4, A5

000023A2        LEA             (A6), A7


000023A4        LEA             $420900009, A2
```

```
»           EQU     $0200
000023FF  =000001C0               1963  MASK_8
» 76        EQU     $01C0
000023FF  =00000118               1964  MASK_8
» 43        EQU     $0118
000023FF  =00000100               1965  MASK_8
»           EQU     $0100
000023FF  =000000FF               1966  MASK_7
» 6543210   EQU     $00FF
000023FF  =000000F0               1967  MASK_7
» 654       EQU     $00F0
000023FF  =000000C0               1968  MASK_7
» 6         EQU     $00C0
000023FF  =00000020               1969  MASK_6
»           EQU     $0020
000023FF  =00000040               1970  MASK_5
»           EQU     $0040
000023FF  =00000018               1971  MASK_4
» 3         EQU     $0018
000023FF  =0000000F               1972  MASK_3
» 210       EQU     $000F
000023FF  =00000007               1973  MASK_2
» 10        EQU     $0007
000023FF  =00000038               1974  MASK_5
» 43        EQU     $0038
```

```
000023FF                          1975
000023FF  =00000000               1976  MODE_D
» n         EQU     $0000
000023FF  =00000008               1977  MODE_A
» n         EQU     $0008
000023FF  =00000010               1978  MODE_A
» nInd      EQU     $0010
000023FF  =00000018               1979  MODE_A
» nPostInc  EQU     $0018
000023FF  =00000020               1980  MODE_A
» nPreDec   EQU     $0020
000023FF  =00000038               1981  MODE_E
» lse       EQU     $0038
000023FF  =00000004               1982  MODE_I
» mm        EQU     $0004
000023FF  =00000001               1983  MODE_A
» bsLong    EQU     $0001
000023FF  =00000000               1984  MODE_A
» bsWord    EQU     $0000
000023FF                          1985
```

```
000023AA        LEA             A7, A6

000023AC        ADDQ.W  #3, D5

000023AE        MOVEA.L A4, A7

000023B0        BCLR    D4, D0

000023B2        BCLR    D4, A5

000023B4        LEA             (A6), A7

000023B6        LEA             $570900009, A2

000023BC        LEA             (A5), A6

000023BE        MOVEM   .W      (A3),

000023C0        BCLR    D4, D0

000023C2        BCLR    D4, A6

000023C4        LEA             D7, A2

000023C6        ORI.B   #$4E4F, A1
000023CA        ADDQ.B  #8, A1
000023CC        BCLR    D4, D0
000023CE        BCLR    D4, A7
000023D0        ADDQ.B  #1, D0
000023D2        BCLR    D4, A7
000023D4        ADDQ.W  #1, A1
000023D6        ORI.B   #$524F, A1
000023DA        MOVEM   .W      A1,

000023DC        ORI.B   #$524F, A1

000023E0        ADDQ.B  #1, A1

000023E2        ORI.B   #$5254, A1

000023E6        SUBQ.B  #1, A1

000023E8        BCLR    D4, D0

000023EA        BCLR    D4, (A3)
```

```
000023EC        LEA             A5, A4
```

```
000023FF                          1986
000023FF                          1987
000023FF                          1988


00002400= 44 30 00                1989  EA_Str
» _D0       DC.B    'D0',0
00002402= 44 31 00                1990  EA_Str
» _D1       DC.B    'D1',0
00002405= 44 32 00                1991  EA_Str
» _D2       DC.B    'D2',0


00002408= 44 33 00                1992  EA_Str
» _D3       DC.B    'D3',0
0000240B= 44 34 00                1993  EA_Str
» _D4       DC.B    'D4',0
0000240E= 44 35 00                1994  EA_Str
» _D5       DC.B    'D5',0
```

```
000023EE        MOVEM   .W      A2//A0/D6/5//3
» /D2//D0/, D1
000023F2        BCLR    D4, D0
000023F4        BCLR    D4, (A3)
000023F6        SUBQ.W  #2, D2
000023F8        ORI.B   #$5355, A1
000023FC        NEG.W   (A1)
000023FE        ORI.W   #$3000, D4
00002402        NEG.B   $00443200
00002408        NEG.B   $00443400

0000240E        NEG.B   $00443600
00002414        NEG.B   $00413000

0000241A        LEA             $00413200, A0

00002420        LEA             $00413400, A0
00002426        LEA             $00413600, A0
0000242C        LEA             $00284130, A0

00002432        MOVE.L  D0, -(A4)

00002434        MOVEA.L D1, A4

00002436        MOVE.W  $00284132, -(A0)
0000243C        MOVE.L  D0, -(A4)
0000243E        MOVEA.L D1, A4
00002440        MOVE.W  $00284134, -(A1)
00002446        MOVE.L  D0, -(A4)
00002448        MOVEA.L D1, A4
0000244A        MOVE.W  $00284136, -(A2)
00002450        MOVE.L  D0, -(A4)
00002452        MOVEA.L D1, A4
00002454        MOVE.W  $00284130, -(A3)
0000245A        MOVE.L  $00284131, -(A4)
00002460        MOVE.L  $00284132, -(A4)
00002466        MOVE.L  $00284133, -(A4)
0000246C        MOVE.L  $00284134, -(A4)
00002472        MOVE.L  $00284135, -(A4)
00002478        MOVE.L  $00284136, -(A4)
0000247E        MOVE.L  $00284137, -(A4)
00002484        MOVE.L  $002D2841, -(A4)
0000248A        MOVE.W  $002D2841, D0
00002490        MOVE.W  $002D2841, -(A0)
00002496        MOVE.W  $002D2841, D1
0000249C        MOVE.W  $002D2841, -(A1)
000024A2        MOVE.W  $002D2841, D2
000024A8        MOVE.W  $002D2841, -(A2)
000024AE        MOVE.W  $002D2841, D3
000024B4        MOVE.W  $00230023, -(A3)
000024BA        MOVE.L  D0, D2
000024BC        MOVE.L  D0, D2
```

```
00002411= 44 36 00                1995  EA_Str
» _D6       DC.B    'D6',0
00002414= 44 37 00                1996  EA_Str
» _D7       DC.B    'D7',0
00002417                          1997
00002417= 41 30 00                1998  EA_Str
» _A0       DC.B    'A0',0
0000241A= 41 31 00                1999  EA_Str
» _A1       DC.B    'A1',0
0000241D= 41 32 00                2000  EA_Str
» _A2       DC.B    'A2',0
00002420= 41 33 00                2001  EA_Str
» _A3       DC.B    'A3',0
00002423= 41 34 00                2002  EA_Str
» _A4       DC.B    'A4',0
00002426= 41 35 00                2003  EA_Str
» _A5       DC.B    'A5',0
00002429= 41 36 00                2004  EA_Str
» _A6       DC.B    'A6',0
0000242C= 41 37 00                2005  EA_Str
» _A7       DC.B    'A7',0
0000242F                          2006
0000242F= 28 41 30 29 00          2007  EA_Str
» _AInd0       DC.B    '(A0)',0
```

```
000024BE        MOVE.L  $24022405, $2408240B
000024C8        MOVE.L  A6, D2
000024CA        MOVE.L  (A1), D2
000024CC        MOVE.L  (A4), D2
000024CE        MOVE.L  (A7), D2
000024D0        MOVE.L  (A2)+, D2
000024D2        MOVE.L  (A5)+, D2
000024D4        MOVE.L  -(A0), D2
000024D6        MOVE.L  -(A3), D2
000024D8        MOVE.L  -(A6), D2
000024DA        MOVE.L  $242C242F, D2
000024E0        MOVE.L  #$243E, D2
000024E6        MOVEA.L D3, A2
000024E8        MOVEA.L A0, A2
000024EA        MOVEA.L A5, A2
000024EC        MOVEA.L (A2), A2
000024EE        MOVEA.L (A7), A2
000024F0        MOVEA.L (A5)+, A2
000024F2        MOVEA.L -(A3), A2
000024F4        MOVEA.L $246F2475, A2
000024FA        MOVEA.L $24812487, A2
00002500        MOVE.L  A5, (A2)
00002502        MOVE.L  (A3), (A2)
00002504        MOVE.L  (A1)+, (A2)
00002506        MOVE.L  (A7)+, (A2)
00002508        MOVE.L  -(A5), (A2)
0000250A        MOVE.L  $24B10000, (A2)
00002510        ORI.B   #$0002, D1
00002514        ORI.B   #$0004, D3

00002518        ORI.B   #$0006, D5

0000251C        ORI.B   #$0008, D7
```

```
00002434= 28 41 31 29 00          2008  EA_Str
» _AInd1       DC.B    '(A1)',0
00002439= 28 41 32 29 00          2009  EA_Str
» _AInd2       DC.B    '(A2)',0
0000243E= 28 41 33 29 00          2010  EA_Str
» _AInd3       DC.B    '(A3)',0
00002443= 28 41 34 29 00          2011  EA_Str
» _AInd4       DC.B    '(A4)',0
00002448= 28 41 35 29 00          2012  EA_Str
» _AInd5       DC.B    '(A5)',0
0000244D= 28 41 36 29 00          2013  EA_Str
» _AInd6       DC.B    '(A6)',0
00002452= 28 41 37 29 00          2014  EA_Str
» _AInd7       DC.B    '(A7)',0
00002457                          2015          00002520        ORI.B   #$000A, A1
00002457= 28 41 30 29 2B 00       2016  EA_Str
» _APostInc0       DC.B    '(A0)+',0
0000245D= 28 41 31 29 2B 00       2017  EA_Str
» _APostInc1       DC.B    '(A1)+',0
00002463= 28 41 32 29 2B 00       2018  EA_Str
» _APostInc2       DC.B    '(A2)+',0
00002469= 28 41 33 29 2B 00       2019  EA_Str
» _APostInc3       DC.B    '(A3)+',0
0000246F= 28 41 34 29 2B 00       2020  EA_Str
» _APostInc4       DC.B    '(A4)+',0
00002475= 28 41 35 29 2B 00       2021  EA_Str
» _APostInc5       DC.B    '(A5)+',0
0000247B= 28 41 36 29 2B 00       2022  EA_Str
» _APostInc6       DC.B    '(A6)+',0
00002481= 28 41 37 29 2B 00       2023  EA_Str
» _APostInc7       DC.B    '(A7)+',0
00002487                          2024
00002487= 2D 28 41 30 29 00       2025  EA_Str
» _APreDec0       DC.B    '-(A0)',0
0000248D= 2D 28 41 31 29 00       2026  EA_Str
» _APreDec1       DC.B    '-(A1)',0
00002493= 2D 28 41 32 29 00       2027  EA_Str
» _APreDec2       DC.B    '-(A2)',0
00002499= 2D 28 41 33 29 00       2028  EA_Str
» _APreDec3       DC.B    '-(A3)',0
0000249F= 2D 28 41 34 29 00       2029  EA_Str
» _APreDec4       DC.B    '-(A4)',0
000024A5= 2D 28 41 35 29 00       2030  EA_Str
» _APreDec5       DC.B    '-(A5)',0
000024AB= 2D 28 41 36 29 00       2031  EA_Str
» _APreDec6       DC.B    '-(A6)',0
000024B1= 2D 28 41 37 29 00       2032  EA_Str
» _APreDec7       DC.B    '-(A7)',0
000024B7                          2033          00002524        ORI.B   #$000C, A3
000024B7                          2034
000024B7= 23 00                   2035  EA_Str
» _Hash             DC.B    '#',0
```

```
000024B9= 23 24 00                2036  EA_Str
» _HashDollar       DC.B    '#$',0
000024BC= 24 00                   2037  EA_Str
» _Dollar           DC.B    '$',0
000024BE                          2038
000024BE                          2039
000024BE= 23FF 2402 2405 2408     2040  EA_Str
» Array_Dn          DC.W    EA_Str_D0,
» EA_Str_D1,        EA_Str_D2,        EA_Str_D
» 3,        EA_Str_D4,        EA_Str_D5,
»   EA_Str_D6,        EA_Str_D7
          240B 240E 2411 2414
000024CE= 2417 241A 241D 2420     2042  EA_Str
» Array_An          DC.W    EA_Str_A0,
» EA_Str_A1,        EA_Str_A2,        EA_Str_A
» 3,        EA_Str_A4,        EA_Str_A5,
»   EA_Str_A6,        EA_Str_A7
          2423 2426 2429 242C
000024DE= 242F 2434 2439 243E     2044  EA_Str
» Array_AnInd       DC.W    EA_Str_AInd0,
» EA_Str_AInd1,     EA_Str_AInd2,     EA_Str_A
» Ind3,     EA_Str_AInd4,     EA_Str_AInd5,
»   EA_Str_AInd6,     EA_Str_AInd7
          2443 2448 244D 2452
000024EE= 2457 245D 2463 2469     2046  EA_Str
» Array_AnPostInc   DC.W    EA_Str_APostInc0,
» EA_Str_APostInc1, EA_Str_APostInc2, EA_Str_A
» PostInc3, EA_Str_APostInc4, EA_Str_APostInc5
» , EA_Str_APostInc6, EA_Str_APostInc7
          246F 2475 247B 2481
000024FE= 2487 248D 2493 2499     2048  EA_Str
» Array_AnPreDec    DC.W    EA_Str_APreDec0,
» EA_Str_APreDec1,  EA_Str_APreDec2,  EA_Str_A
» PreDec3,  EA_Str_APreDec4,  EA_Str_APreDec5,
»   EA_Str_APreDec6,  EA_Str_APreDec7
          249F 24A5 24AB 24B1
0000250E                          2050
0000250E                          2051
0000250E= 0000 0001 0002 0003     2052  EA_Mod
» eArray_Dn         DC.W    $00,$01,$02,$03,$0
» 4,$05,$06,$07
          0004 0005 0006 0007
0000251E= 0008 0009 000A 000B     2054  EA_Mod
» eArray_AnArray    DC.W    $08,$09,$0A,$0B,$0
» C,$0D,$0E,$0F
          000C 000D 000E 000F
0000252E= 0010 0011 0012 0013     2056  EA_Mod
» eArray_AnInd      DC.W    $10,$11,$12,$13,$1
» 4,$15,$16,$17
          0014 0015 0016 0017
0000253E= 0018 0019 001A 001B     2058  EA_Mod
» eArray_AnPostInc  DC.W    $18,$19,$1A,$1B,$1
```

```
00002528        ORI.B   #$000E, A5
0000252C        ORI.B   #$0010, A7
00002530        ORI.B   #$0012, (A1)

00002534        ORI.B   #$0014, (A3)
00002538        ORI.B   #$0016, (A5)

0000253C        ORI.B   #$0018, (A7)

00002540        ORI.B   #$001A, (A1)+
00002544        ORI.B   #$001C, (A3)+
```

```
» C,$1D,$1E,$1F
          001C 001D 001E 001F                                  | 00002548         ORI.B   #$001E, (A5)+
0000254E= 0020 0021 0022 0023     2060  EA_Mod                 |
» eArray_AnPreDec   DC.W    $20,$21,$22,$23,$2                 |
» 4,$25,$26,$27                                                |
          0024 0025 0026 0027                                  | 0000254C         ORI.B   #$0020, (A7)+
0000255E= 003C 0039 0038          2062  EA_Mod                 | 00002550         ORI.B   #$0022, -(A1)
» eArray_Else       DC.W    $3C,$39,$38                        |
00002564                          2063                         | 00002554         ORI.B   #$0024, -(A3)
00002564                          2064                         | 00002558         ORI.B   #$0026, -(A5)
00002564= 30 00                   2065  Hex_St                 | 0000255C         ORI.B   #$003C, -(A7)
» r_00              DC.B    '0',0                              |
00002566= 31 00                   2066  Hex_St                 |
» r_01              DC.B    '1',0                              |
00002568= 32 00                   2067  Hex_St                 |
» r_02              DC.B    '2',0                              |
0000256A= 33 00                   2068  Hex_St                 | 00002560         ORI.B   #$0038, $30003100
» r_03              DC.B    '3',0                              |
0000256C= 34 00                   2069  Hex_St                 |
» r_04              DC.B    '4',0                              |
0000256E= 35 00                   2070  Hex_St                 |
» r_05              DC.B    '5',0                              |
00002570= 36 00                   2071  Hex_St                 |
» r_06              DC.B    '6',0                              |
00002572= 37 00                   2072  Hex_St                 |
» r_07              DC.B    '7',0                              |
00002574= 38 00                   2073  Hex_St                 |
» r_08              DC.B    '8',0                              |
00002576= 39 00                   2074  Hex_St                 |
» r_09              DC.B    '9',0                              |
00002578= 41 00                   2075  Hex_St                 |
» r_10              DC.B    'A',0                              |
0000257A= 42 00                   2076  Hex_St                 |
» r_11              DC.B    'B',0                              |
0000257C= 43 00                   2077  Hex_St                 |
» r_12              DC.B    'C',0                              |
0000257E= 44 00                   2078  Hex_St                 |
» r_13              DC.B    'D',0                              |
00002580= 45 00                   2079  Hex_St                 |
» r_14              DC.B    'E',0                              |
00002582= 46 00                   2080  Hex_St                 |
» r_15              DC.B    'F',0                              |
00002584                          2081                         | 00002568         MOVE.W  D0, D1
00002584= 2564 2566 2568 256A     2082  Hex_St                 | 0000256A         MOVE.W  D0, -(A1)
» rArray         DC.W       Hex_Str_00, Hex_St                 |
» r_01, Hex_Str_02, Hex_Str_03, Hex_Str_04, He                 |
» x_Str_05, Hex_Str_06, Hex_Str_07                             |
          256C 256E 2570 2572                                  |
00002594= 2574 2576 2578 257A     2084                         |
»                DC.W       Hex_Str_08, Hex_St                 |
» r_09, Hex_Str_10, Hex_Str_11, Hex_Str_12, He                 |
» x_Str_13, Hex_Str_14, Hex_Str_15                             |
```

```
          257C 257E 2580 2582
000025A4                          2086
000025A4                          2087
000025A4                          2088
000025A4                          2089
000025A4                          2090
000025A4                          2091
000025A4                          2092
000025A4                          2093
000025A4                          2094
000025A4                          2095
000025A4                          2096  ------
» -------------- end include -----------------
» ---
000025A4                          2097
000025A4                          2098
000025A4                          2099      ;O
» RG $00111112
000025A4                          2100
000025A4                          2101
000025A4                          2102      ;I
» NCLUDE '.\UnitTests\TEST_002.x68'
000025A4                          2103
000025A4                          2104
000025A4                          2105
000025A4                          2106
000025A4                          2107
000025A4                          2108     END
»     $1000        ; last line of source

No errors detected
No warnings generated


SYMBOL TABLE INFORMATION
Symbol-name         Value
-------------------------
ADD_DN_EA_DN        1618
ADD_EA_DN_EA        162E
AN                  1EA8
APPENDCOMMA         1EC6
APPENDLOOP          2056
APPENDOUTPUT        203A
ARRAY_ELEMENT_WIDTH  2
ASCIITOHEX          2174
ASCIITOHEXREGLIST   1
ASCII_VAL           21E4
ASL_I               19AC
ASL_I_DN            19E2
ASL_I_I             19CC
ASR_I               19F8
ASR_I_DN            1A2E
```

```
0000256C        MOVE.W  D0, D2
0000256E        MOVE.W  D0, -(A2)

00002570        MOVE.W  D0, D3

00002572        MOVE.W  D0, -(A3)
00002574        MOVE.W  D0, D4

00002576        MOVE.W  D0, -(A4)
00002578        LEA             D0, A0

0000257A        NEG.B   D0
0000257C        LEA             D0, A1

0000257E        NEG.B   D0
00002580        LEA             D0, A2
00002582        NEG.B   D0
00002584        MOVE.L  -(A4), $25662568

0000258A        MOVE.L  $256C256E, $25702572

00002594        MOVE.L  #$2578, $257A257C
0000259E        MOVE.L  $25802582, $FFFFFFFF
```

```
ASR_I_I             1A18
BCLR_I              1598
BEGIN_ADDRESS_STR   21FA
BRANCH_0110         11F6
BYTEREAD            21E3
BYTE_LENGTH         1
CHRDONE             2190
CHRLOOP             217C
CMPI_B              147C
CMPI_END            14C4
CMPI_L              1458
CMPI_W              14A0
COUNTDONE           2130
COUNTLOOP           211E
CR                  D
CURRENT_STR_LENGTH  220A
DATA11109           1ED4
DATA11109_NOT_ZERO  1EEC
DISASSEMBLEOPCODE   1084
DISPLACEMENT        1E40
DISPLACEMENT_FETCH_WORD  1E5C
DN                  1E6E
DN210               1E8C
EA_APPENDMODEREGISTER  1BD4
EA_APPENDMXN        1BB2
EA_APPENDREGISTERNAME  1C40
EA_APPENDXNM        1BC0
EA_GETISMXN         1F5A
EA_GETISXNM         1F68
EA_GETSTANDARDMODEIND3  1D1E
EA_GETSTANDARDREGIND3  1D46
EA_ISMXN            21ED
EA_ISXNM            21EE
EA_MASK_MODE        1D3C
EA_MASK_REGISTER    1D64
EA_MODEARRAY_ANARRAY  251E
EA_MODEARRAY_ANIND  252E
EA_MODEARRAY_ANPOSTINC  253E
EA_MODEARRAY_ANPREDEC  254E
EA_MODEARRAY_DN     250E
EA_MODEARRAY_ELSE   255E
EA_MODE_ASRL_DISTANCE  3
EA_PROCESSABSOLUTELONG  1CFA
EA_PROCESSABSOLUTEWORD  1CE0
EA_PROCESSELSE      1C58
EA_PROCESSIMMEDIATE  1C72
EA_PROCESSIMMEDIATEBYTE  1C8C
EA_PROCESSIMMEDIATELONG  1CC4
EA_PROCESSIMMEDIATEWORD  1CAA
EA_REGISTER_ASRL_DISTANCE  9
EA_RETURN           1D16
EA_SETISMXN         1F76
```

```
EA_SETISXNM         1F90
EA_SETSTRARRAY_AN   1C18
EA_SETSTRARRAY_ANIND  1C22
EA_SETSTRARRAY_ANPOSTINC  1C36
EA_SETSTRARRAY_ANPREDEC  1C2C
EA_SETSTRARRAY_DN   1C0E
EA_SHIFTXNM_MODE    1D34
EA_SHIFTXNM_REGISTER  1D5C
EA_STRARRAY_AN      24CE
EA_STRARRAY_ANIND   24DE
EA_STRARRAY_ANPOSTINC  24EE
EA_STRARRAY_ANPREDEC  24FE
EA_STRARRAY_DN      24BE
EA_STR_A0           2417
EA_STR_A1           241A
EA_STR_A2           241D
EA_STR_A3           2420
EA_STR_A4           2423
EA_STR_A5           2426
EA_STR_A6           2429
EA_STR_A7           242C
EA_STR_AIND0        242F
EA_STR_AIND1        2434
EA_STR_AIND2        2439
EA_STR_AIND3        243E
EA_STR_AIND4        2443
EA_STR_AIND5        2448
EA_STR_AIND6        244D
EA_STR_AIND7        2452
EA_STR_APOSTINC0    2457
EA_STR_APOSTINC1    245D
EA_STR_APOSTINC2    2463
EA_STR_APOSTINC3    2469
EA_STR_APOSTINC4    246F
EA_STR_APOSTINC5    2475
EA_STR_APOSTINC6    247B
EA_STR_APOSTINC7    2481
EA_STR_APREDEC0     2487
EA_STR_APREDEC1     248D
EA_STR_APREDEC2     2493
EA_STR_APREDEC3     2499
EA_STR_APREDEC4     249F
EA_STR_APREDEC5     24A5
EA_STR_APREDEC6     24AB
EA_STR_APREDEC7     24B1
EA_STR_D0           23FF
EA_STR_D1           2402
EA_STR_D2           2405
EA_STR_D3           2408
EA_STR_D4           240B
EA_STR_D5           240E
EA_STR_D6           2411
```

```
EA_STR_D7           2414
EA_STR_DOLLAR       24BC
EA_STR_HASH         24B7
EA_STR_HASHDOLLAR   24B9
EMPTY_ADDRESS       0
END_ADDRESS_STR     2202
EORI_B              151C
EORI_END            1564
EORI_L              14F8
EORI_W              1540
ERROR_CODE_FILE_0   225C
ERROR_CODE_FILE_1   2266
ERROR_CODE_FILE_2   2278
ERROR_CODE_FILE_3   2280
FILE_DEFAULT_READ_BYTES  400
FILE_ERROR_ARRAY    2294
FINDNULLLOOPDONE    2100
FLAG_ISFALSE        2028
FLAG_ISTRUE         2030
GET_NEXT_LONG_D7    107E
GET_NEXT_WORD_D6    1072
GET_NEXT_WORD_D7    1078
HEX_STRARRAY        2584
HEX_STR_00          2564
HEX_STR_01          2566
HEX_STR_02          2568
HEX_STR_03          256A
HEX_STR_04          256C
HEX_STR_05          256E
HEX_STR_06          2570
HEX_STR_07          2572
HEX_STR_08          2574
HEX_STR_09          2576
HEX_STR_10          2578
HEX_STR_11          257A
HEX_STR_12          257C
HEX_STR_13          257E
HEX_STR_14          2580
HEX_STR_15          2582
INFILE              21EF
ISFALSE             0
ISTRUE              1
LF                  A
LONG_LENGTH         4
LSL_I               1914
LSL_I_DN            194A
LSL_I_I             1934
LSR_I               1960
LSR_I_DN            1996
LSR_I_I             1980
MAIN_LOOP           1054
MASK_10             400
```

```
MASK_1098           700
MASK_11             800
MASK_1110           C00
MASK_11108          D00
MASK_1110876        DC0
MASK_11109          E00
MASK_111098         F00
MASK_210            7
MASK_3210           F
MASK_43             18
MASK_5              40
MASK_543            38
MASK_6              20
MASK_76             C0
MASK_7654           F0
MASK_76543210       FF
MASK_8              100
MASK_843            118
MASK_876            1C0
MASK_9              200
MASK_OPCODE         F000
MODE_ABSLONG        1
MODE_ABSWORD        0
MODE_AN             8
MODE_ANIND          10
MODE_ANPOSTINC      18
MODE_ANPREDEC       20
MODE_DN             0
MODE_ELSE           38
MODE_IMM            4
MOVEA_B             1708
MOVEA_L             17A8
MOVEA_W             1758
MOVEM_EA_RL         1B56
MOVEM_RL_EA         1B40
MOVE_B              1730
MOVE_L              17D0
MOVE_W              1780
MSG1                21BF
MSG2                21CC
NEWLINE             21BC
NULLLOOP            20F2
OPCODE              1084
OPCODESIZE_BYTE     0
OPCODESIZE_CURRENT  21EC
OPCODESIZE_ELSE     3
OPCODESIZE_GETSIZE  1FEA
OPCODESIZE_ISBYTE   1FF8
OPCODESIZE_ISLONG   2018
OPCODESIZE_ISWORD   2008
OPCODESIZE_LONG     2
OPCODESIZE_SETTOBYTE  1FAA
```

```
OPCODESIZE_SETTOELSE  1FDA
OPCODESIZE_SETTOLONG  1FCA
OPCODESIZE_SETTOWORD  1FBA
OPCODESIZE_WORD     1
OPCODE_ADD          15F8
OPCODE_ADDA         15D0
OPCODE_ADDQ         1346
OPCODE_APPENDSIZESUFFIX  1D6E
OPCODE_APPENDSIZESUFFIX_B  1D8C
OPCODE_APPENDSIZESUFFIX_L  1DBC
OPCODE_APPENDSIZESUFFIX_W  1DA4
OPCODE_ASL          1850
OPCODE_ASR          1866
OPCODE_BCLR         1576
OPCODE_BCS          1A5A
OPCODE_BGE          1A70
OPCODE_BLT          1A86
OPCODE_BRA          1A44
OPCODE_BVC          1A9C
OPCODE_CMP          16B8
OPCODE_CMPI         1436
OPCODE_DATA         12BC
OPCODE_DIVS         1644
OPCODE_EOR          16E0
OPCODE_EORI         14D6
OPCODE_FINISH       1B92
OPCODE_ILLEGAL      1AC2
OPCODE_JSR          1B6C
OPCODE_LEA          1AE2
OPCODE_LSL          1824
OPCODE_LSR          183A
OPCODE_MOVEM        1B20
OPCODE_MULS         136E
OPCODE_NEG          1B04
OPCODE_NOP          1AB2
OPCODE_OR           166C
OPCODE_ORI          1396
OPCODE_ROL          17F8
OPCODE_ROR          180E
OPCODE_RTS          1AD2
OPCODE_SIMHALT      1B82
OPCODE_SUB          12D2
OPCODE_SUBQ         131E
ORI_B               13DC
ORI_END             1424
ORI_L               13B8
ORI_W               1400
OR_DN_EA_DN         168C
OR_EA_DN_EA         16A2
OUTFILENAME         21D8
OUTPUT              220B
OUTPUT_LENGTH       28
```

```
OUTPUT_TEMP          2233
PRINTASCIILONG       20C4
PRINTASCIIWORD       2066
REGISTERLIST_POSTINCREMENT  1F58
REGISTERLIST_PREDECREMENT  1F08
RET_SA               21BA
RL_LOOP              1F16
RL_LOOP_END          1F44
ROL_I                187C
ROL_I_DN             18B2
ROL_I_I              189C
ROR_I                18C8
ROR_I_DN             18FE
ROR_I_I              18E8
SAVEITFAM            E06
SIZE6                1E0A
SIZE6_L              1E18
SIZE6_W              1E2C
SIZE8                1DD4
SIZE8_L              1DE2
SIZE8_W              1DF6
SKIP_0000            110A
SKIP_0001            1172
SKIP_0010            11A6
SKIP_0011            118C
SKIP_0100            12B0
SKIP_0110            1262
SKIP_1000            113E
SKIP_1011            1158
SKIP_1101            1124
SKIP_1110            122C
SKIP_5000            10B0
SKIP_CMPI_EORI       10FC
STACK_LOCATION       70000
STRINGCOMMA          22C6
STRINGPOUND          22C9
STRINGPOUNDHEX       22CB
STRINGSLASH          22CE
STRING_B             22D0
STRING_L             22D8
STRING_W             22D4
STRIP_ASCII          2196
STR_ADD              22DC
STR_ADDA             22E1
STR_ADDQ             22E7
STR_ASL              22ED
STR_ASR              22F3
STR_BCLR             22F9
STR_BCS              2300
STR_BGE              2306
STR_BLT              230C
STR_BRA              2312
```

STR_BVC	2318			
STR_CMP	231E			
STR_CMPI	2323			
STR_DATA	2329			
STR_DIVS	2330			
STR_EOR	2337			
STR_EORI	233C			
STR_ILLEGAL	2342			
STR_JSR	234C			
STR_LEA	2353			
STR_LSL	235A			
STR_LSR	2360			
STR_MOVEA_B	2366			
STR_MOVEA_L	2370			
STR_MOVEA_W	237A			
STR_MOVEM	2384			
STR_MOVEM_L	238C			
STR_MOVEM_W	2396			
STR_MOVE_B	23A0			
STR_MOVE_L	23A9			
STR_MOVE_W	23B2			
STR_MULS	23BB			
STR_NEG	23C2			
STR_NOP	23C7			
STR_OR	23CE			
STR_ORI	23D2			
STR_ROL	23D7			
STR_ROR	23DD			
STR_RTS	23E3			
STR_SIMHALT	23EA			
STR_SUB	23F4			
STR_SUBQ	23F9			
SUB30	21B6			
SUB37	21AE			
SUB_DN_EA_DN	12F2			
SUB_EA_DN_EA	1308			
TAB	9			
TOSAVE	C3F			
TRAPTASK13	20DE			
WORD_LENGTH	2			
_00000000	2114			