```
LOCATION OBJECT CODE LINE     SOURCE LINE

                              1 ^6801^
                              3  NAME ^Rev 00 - DLS^
                              4
                              5 De_MMR_DMA_R MACRO              ;Header Rev. 4
                              6                 .GOTO Ede_MMR_DMA_R
                              7
                              8  Project:      NET, 83-101
                              9
                             10  ##################################################
                             11  ##                                              ##
                             12  ##     MMR_DMA_R                    DLS        ##
                             13  ##                                              ##
                             14  ##################################################
                             15
                             16        Rev History
                             17        Rev.  Date          Name      Change
                             18
                             19         1    12aug1150    MJM       Added stm. just past loc. 0D700H
                             20         0    16jul2330    DLS       Initial Pseudo code AND ENTERED
                             21                                     CODE PRODUCED BY BILL ROSE AND WALTER BANKS
                             22
                             23
                             24
                             25 Ede_MMR_DMA_R  MEND
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      27 Pseudo_code_MMR_DMA_R     MACRO  ;Pseudocode macro area
                      28                       .GOTO Ep_MMR_DMA_R
                      29
                      30
                      31
                      32 Ep_MMR_DMA_R  MEND
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      34 ************************************************************************
                      35 *                                                                      *
                      36 *  MODULE NAME:                                                        *
                      37 *                                                                      *
                      38 *    MMR_DMA_READ                                                      *
                      39 *                                                                      *
                      40 *  INPUTS:                                                             *
                      41 *                                                                      *
                      42 *    REG_X - ADDR TO READ                                              *
                      43 *                                                                      *
                      44 *                                                                      *
                      45 *  FUNCTION(S):                                                        *
                      46 *                                                                      *
                      47 *     1. TO READ A SINGLE BYTE IN THE Z80'S MEM.                       *
                      48 *                                                                      *
                      49 *                                                                      *
                      50 *                                                                      *
                      51 *  OUTPUTS:                                                            *
                      52 *                                                                      *
                      53 *     REG_A - BYTE READ                                                *
                      54 *                                                                      *
                      55 *  CALLS:                                                              *
                      56 *                                                                      *
                      57 *    NONE.                                                             *
                      58 *                                                                      *
                      59 *                                                                      *
                      60 *  CALLED BY:                                                          *
                      61 *                                                                      *
                      62 *    MMR_ACM_RESP                                                      *
                      63 *                                                                      *
                      64 *  NOTES:                                                              *
                      65 *                                                                      *
                      66 *    NONE.                                                             *
                      67 *                                                                      *
                      68 ************************************************************************
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      70 ************************************************************
                      71 *                                                          *
                      72 *  PSEUDO CODE:                                            *
                      73 *     SET DDR'S;                                           *
                      74 *     A=Z80_MEM(X);                                        *
                      75 *     RETURN;                                              *
                      76 *                                                          *
                      77 ************************************************************
                      78
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      80                PROG
                      81                GLB     MMR_DMA_READ
                      82                EXT     STACK_TEMP
                      83 ;                      IMPORTED FROM MMR_DMA_WRITE
                      84 ;
                      85 ;Standard M6801 I/O register definitions
                      86 ;
                      87 ;
(0000)                88 P1DDR          EQU             0  PORT 1 DATA DIRECTION REGISTER
(0001)                89 P2DDR          EQU             1  PORT 2 DATA DIRECTION REGISTER
(0002)                90 P1DATA         EQU             2  PORT 1 DATA REGISTER
(0003)                91 P2DATA         EQU             3  PORT 2 DATA REGISTER
(0004)                92 P3DDR          EQU             4  PORT 3 DATA DIRECTION REGISTER
(0005)                93 P4DDR          EQU             5  PORT 4 DATA DIRECTION REGISTER
(0006)                94 P3DATA         EQU             6  PORT 3 DATA REGISTER
(0007)                95 P4DATA         EQU             7  PORT 4 DATA REGISTER
(000F)                96 P3CSR          EQU            15  PORT 3 CONTROL AND STATUS
                      97 ;
                      98 ;READ DATA  This code fragment will read one byte from
                      99 ;the Z80 memory
                     100 ;
                     101 ;
                     102 ;Enter this subroutine with the following arguments
                     103 ;
                     104 ;       X         address of the location to be read
                     105 ;
                     106 ; Return with
                     107 ;
                     108 ;       A         (Z80X)
                     109 ;       B         contents is lost     } no, they saved 'em
                     110 ;       X         contents is lost     }
                     111 ;
                     112 ;
(0000)               113 MMR_DMA_READ:  EQU             $
0000 3C              114                PSHX
0001 37              115                PSHB
0002 8618            116                LDAA  #018H      SETUP LATCH ENABLE AND OUTPUT
0004 970F            117                STAA  P3CSR      STROBE SELECT
0006 3C              118                PSHX             PUT THE ADDRESS WHER BIA REGISTERS
                     119 *                               CAN GET IT
0007 861E            120                LDAA  #01EH      PORT 1 CONTROL BITS.
0009 9703            121                STAA  P2DATA     GET ALL THE DATA READY
000B 32              122                PULA             GET HIGH ADDRESS
000C 9702            123                STAA  P1DATA     PORT2 HAS HIGH ADDRESS BITS
000E 32              124                PULA             Z80 LOW ADDRESS BYTE
000F 9707            125                STAA  P4DATA
0011 CC15FF          126                LDD   #015FFH    SET A TO FF  B TO 00
0014 CE0010          127                LDX   #0010H     CONSTANTS TO RECONFIGURE FOR INPUTS
0017 BF0000          128                STS   STACK_TEMP SAVE THE STACK USE IT AS AN INDEX
001A 8E0000          129                LDS   #0000H
                     130 *
                     131 *
                     132 *                    ACTUAL DMA DATA READ STARTS HERE
                     133 *
                     134 *
001D 9701            135                STAA  P2DDR      ISSUE THE DMA REQUEST
001F D705            136                STAB  P4DDR      ENABLE LOW  ADDRESS OUTPUTS   } NPD guess 9603.01
```

```
LOCATION OBJECT CODE LINE      SOURCE LINE

    0021 D700          137                 STAB  P1DDR      ENABLE HIGH ADDRESS OUTPUTS
    0023 9606          138                 LDAA  P3DATA     8/12/83   Modification .
    0025 D706          139                 STAB  P3DATA     DUMMY WRITE TO PORT 3
    0027 9F04          140                 STS   P3DDR
    0029 DF00          141                 STX   P1DDR
    002B 9606          142                 LDAA  P3DATA     GET THE DATA LATCHED IN PORT 3
                       143 *
                       144 *
                       145 *                       Z80 READ IS COMPLETE DATA IS IN ACCUMULATOR A
                       146 *
                       147 *
    002D BE0000        148                 LDS   STACK_TEMP  RESTORE THE STACK REGISTER
    0030 33            149                 PULB
    0031 38            150                 PULX
    0032 39            151                 RTS

Errors=    0
```

LINE#	SYMBOL	TYPE	REFERENCES
113	MMR_DMA_READ	P	81
90	P1DATA	A	123
88	P1DDR	A	137,141
91	P2DATA	A	121
89	P2DDR	A	135
96	P3CSR	A	117
94	P3DATA	A	138,139,142
92	P3DDR	A	140
95	P4DATA	A	125
93	P4DDR	A	136
82	STACK_TEMP	E	128,148

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      1 ^6801^
                      3  NAME ^Rev 00 - DLS^
                      4
                      5 De_MMR_DMA_W MACRO              ;Header Rev. 4
                      6                 .GOTO Ede_MMR_DMA_W
                      7
                      8  Project:      NET, 83-101
                      9
                     10  ************************************************************
                     11  *                                                          *
                     12  *     MMR_DMA_W                          DLS               *
                     13  *                                                          *
                     14  ************************************************************
                     15
                     16        Rev History
                     17        Rev.   Date          Name      Change
                     18         2     26jul1228    GFB       STACK_TEMP
                     19         1     26jul1222    GFB       UPDATE AND CORRECT DMA WRITE
                     20         0     16jul1133    DLS       Initial Pseudo code and entered code
                     21                                      produced by Bill Rose and Walter Banks
                     22
                     23
                     24
                     25 Ede_MMR_DMA_W  MEND
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      27 Pseudo_code_MMR_DMA_W    MACRO  ;Pseudocode macro area
                      28                     .GOTO Ep_MMR_DMA_W
                      29
                      30
                      31
                      32 Ep_MMR_DMA_W  MEND
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      34 ************************************************************************
                      35 *                                                                      *
                      36 *  MODULE NAME:                                                        *
                      37 *                                                                      *
                      38 *    MMR_DMA_WRITE                                                     *
                      39 *                                                                      *
                      40 *  INPUTS:                                                             *
                      41 *                                                                      *
                      42 *    REG_A - DATA BYTE TO WRITE.                                       *
                      43 *    REG_X - ADDRESS IN Z80 MEM TO WRITE TO.                           *
                      44 *                                                                      *
                      45 *                                                                      *
                      46 *  FUNCTION(S):                                                        *
                      47 *                                                                      *
                      48 *     1. TO WRITE A SINGLE BYTE TO THE Z80'S MEMORY SPACE.             *
                      49 *                                                                      *
                      50 *  OUTPUTS:                                                            *
                      51 *                                                                      *
                      52 *     NONE.                                                            *
                      53 *                                                                      *
                      54 *  CALLS:                                                              *
                      55 *                                                                      *
                      56 *    NONE.                                                             *
                      57 *                                                                      *
                      58 *                                                                      *
                      59 *  CALLED BY:                                                          *
                      60 *                                                                      *
                      61 *    MMR_ACM_RESP                                                      *
                      62 *                                                                      *
                      63 *  NOTES:                                                              *
                      64 *                                                                      *
                      65 *    NONE.                                                             *
                      66 *                                                                      *
                      67 ************************************************************************
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      69 ************************************************************
                      70 *                                                          *
                      71 *  PSEUDO CODE:                                            *
                      72 *    SET DDR'S                                             *
                      73 *    Z80_MEM(REG_X) = REG_A;                               *
                      74 *   RETURN;                                                *
                      75 ************************************************************
                      76
```

```
TION OBJECT CODE LINE     SOURCE LINE

                    78
                    79                  DATA
                    80                  ORG  008EH
008E                81 STACK_TEMP       RMB  2            SAVES THE STACK POINTER
                    82
                    83                  PROG
                    84                  GLB      MMR_DMA_WRITE
                    85                  GLB      STACK_TEMP
                    86 ;
                    87 ;Standard M6801 I/O register definitions
                    88 ;
                    89 ;
          (0000)    90 P1DDR            EQU              0  PORT 1 DATA DIRECTION REGISTER
          (0001)    91 P2DDR            EQU              1  PORT 2 DATA DIRECTION REGISTER
          (0002)    92 P1DATA           EQU              2  PORT 1 DATA REGISTER
          (0003)    93 P2DATA           EQU              3  PORT 2 DATA REGISTER
          (0004)    94 P3DDR            EQU              4  PORT 3 DATA DIRECTION REGISTER
          (0005)    95 P4DDR            EQU              5  PORT 4 DATA DIRECTION REGISTER
          (0006)    96 P3DATA           EQU              6  PORT 3 DATA REGISTER
          (0007)    97 P4DATA           EQU              7  PORT 4 DATA REGISTER
          (000F)    98 P3CSR            EQU             15  PORT 3 CONTROL AND STATUS
                    99 ;
                   100 ;
                   101 ;WRITE DATA BYTE
                   102 ;
                   103 ;       This subroutine will write a single byte of
                   104 ;       data to the Z80's memory.
                   105 ;
                   106 ;Enter with the following arguments
                   107 ;       X         Address in the Z80's memory
                   108 ;       A         Data byte.
                   109 ;Return
                   110 ;       Contents of A,B,X is lost.
                   111 ;
                   112 ************************************************
          (0000)   113 MMR_DMA_WRITE    EQU   $
    3C             114                   PSHX
    36             115                   PSHA
    37             116                   PSHB
    7F000F         117                   CLR   P3CSR      SET UP P3 CONTROL PORT
    9706           118                   STAA  P3DATA
    3C             119                   PSHX
    32             120                   PULA             GET HIGH ADDRESS BYTE
    9702           121                   STAA  P1DATA     LOAD HIGH ADDRESS REGISTER
    32             122                   PULA             GET LOW ADDRESS BYTE
    9707           123                   STAA  P4DATA
    861A           124                   LDAA  #01AH      SET UP CONTROL BITS FOR WRITE
    9703           125                   STAA  P2DATA
    CC15FF         126                   LDD   #015FFH    CONSTANT TO SET PORT OUTPUTS
    CE0010         127                   LDX   #00010H    CONSTANT TO SET PORT INPUTS
                   128
    9F8E           129                   STS   STACK_TEMP SAVE THE STACK REGISTER
    8E0000         130                   LDS   #00000H    USE STACK AS A WORD WIDE REGISTER
                   131 *
                   132 *             START OF THE ACTUAL TRANSFER
                   133 *
    9701           134                   STAA  P2DDR      SET UP FOR OUTPUTS
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    0020 D704          135                STAB  P3DDR      PORT 3/4 ENABLED OUT
    0022 D705          136                STAB  P4DDR
    0024 D700          137                STAB  P1DDR      SET THE DMA
    0026 9606          138                LDAA  P3DATA     DUMMY READ
    0028 9F04          139                STS   P3DDR
    002A DF00          140                STX   P1DDR      REMOVE ALL REGISTERS FROM THE BUS
                       141 *
                       142 *
    002C 9E8E          143                LDS   STACK_TEMP RESTORE THE STACK
    002E 33            144                PULB
    002F 32            145                PULA
    0030 38            146                PULX
    0031 39            147                RTS

Errors=    0
```

LINE#	SYMBOL	TYPE	REFERENCES
113	MMR_DMA_WRITE	P	84
92	P1DATA	A	121
90	P1DDR	A	137,140
93	P2DATA	A	125
91	P2DDR	A	134
98	P3CSR	A	117
96	P3DATA	A	118,138
94	P3DDR	A	135,139
97	P4DATA	A	123
95	P4DDR	A	136
81	STACK_TEMP	A	85,129,143

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      1 ^6801^
                      3  NAME ^Rev 00 - WRB^
                      4
                      5 De_MASTER_AP MACRO              ;Header Rev. 4
                      6                 .GOTO Ede_MASTER_AP
                      7
                      8  Project:      NET, 03-101
                      9
                     10  ************************************************
                     11  *                                              *
                     12  *     MASTER_AP                      WRB       *
                     13  *                                              *
                     14  ************************************************
                     15
                     16        Rev History
                     17        Rev.  Date          Name      Change
                     18
                     19         5    20jul10141   WRB       RSPND CODE REMOVED THEN FIXED !!!
                     20         4    27jul11725   GFB       RSPND CODE MERGED
                     21         3    26jul11239   GFB       EXTERNALLY LOCATE DMA ROUTINES
                     22         2    23jul12321   GFB       NIM BLOCK CORRECTION/MEM_LENGTH
                     23         1    23jul12214   GFB       Linkage error/common block
                     24         0    23jul11014   WRB       ENTERED THE WHOLE BOWL OF WAX
                     25
                     26 Ede_MASTER_AP  MEND
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     20 ****************************************************************
                     29 *                                                              *
                     30 *  MODULE NAME:                                                *
                     31 *                                                              *
                     32 *    MR. APP                                                   *
                     33 *                                                              *
                     34 *  INPUTS:                                                     *
                     35 *                                                              *
                     36 *    Z80 DATA                                                  *
                     37 *                                                              *
                     38 *  IMPORTS
                     39              EXT  MMR_DMA_WRITE,MMR_DMA_READ
                     40 *                                                              *
                     41 *  FUNCTION(S):                                                *
                     42 *                                                              *
                     43 *     1. TRANSMUTATION OF Z80 REQUESTS INTO NET ACTIVITY.      *
                     44 *                                                              *
                     45 *                                                              *
                     46 *                                                              *
                     47 *  OUTPUTS:                                                    *
                     48 *                                                              *
                     49 *     NET  0.                                                  *
                     50 *                                                              *
                     51 *  CALLS:                                                      *
                     52 *                                                              *
                     53 *    WHOMEVER HE NEEDS.                                        *
                     54 *                                                              *
                     55 *                                                              *
                     56 *  CALLED BY:                                                  *
                     57 *                                                              *
                     58 *    NONE.                                                     *
                     59 *                                                              *
                     60 *  NOTES:                                                      *
                     61 *                                                              *
                     62 *    NONE.                                                     *
                     63 *                                                              *
                     64 ****************************************************************
```

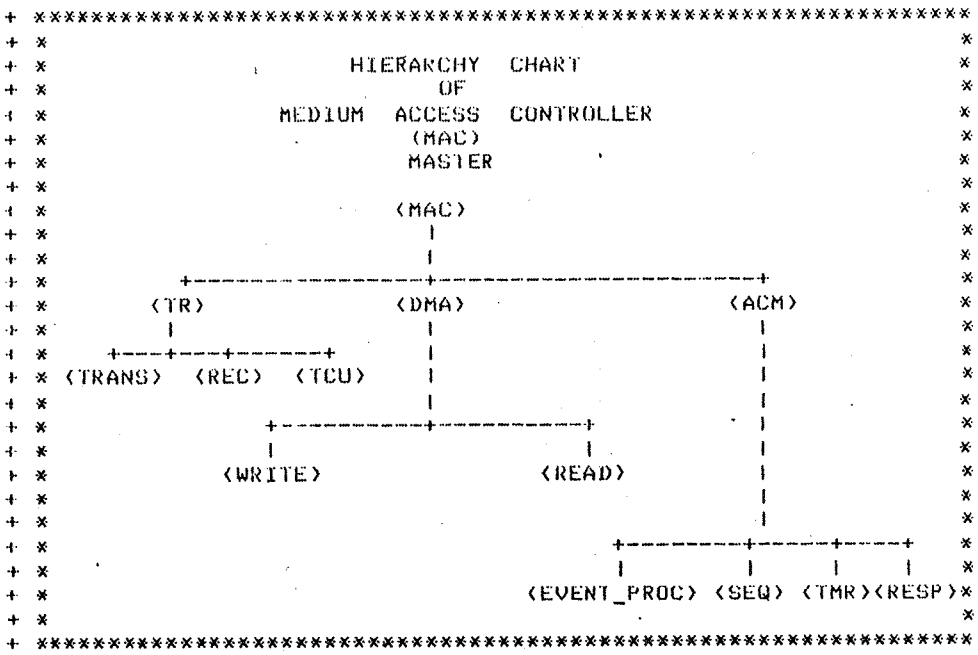
```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      66 ****************************************************************
                      67 *                                                              *
                      68 *  PSEUDO CODE:                                                *
                      69 *                                                              *
                      70 *       MASTER_APP FUNCTION;                                   *
                      71 *       RETURN;                                                *
                      72 *                                                              *
                      73 ****************************************************************
                      74 ************************************************************
                      75 *                                                          *
                      76 *   MASTER_APP FOR ADAMNET                                 *
                      77 *                                                          *
                      78 *          Walter Banks                                    *
                      79 *          North American Technology, Inc                  *
                      80 *          174 Concord Street                              *
                      81 *          Peterborough, NH  03458                         *
                      82 *          (603) 924-6048                                  *
                      83 *          20 July 1983                                    *
                      84 *                                                          *
                      85 ************************************************************
                      86 *
                      87 *               SYSTEM CONSTANTS
                      88 *
                      89                 EXT             MMR_MAC
                      90 *Standard M6801 I/O register definitions ***************************
                      91 *                                                                 *
                      92 *                                                                 *
(0000)                93 P1DDR           EQU             0  PORT 1 DATA DIRECTION REGISTER  *
(0001)                94 P2DDR           EQU             1  PORT 2 DATA DIRECTION REGISTER  *
(0002)                95 P1DATA          EQU             2  PORT 1 DATA REGISTER            *
(0003)                96 P2DATA          EQU             3  PORT 2 DATA REGISTER            *
(0004)                97 P3DDR           EQU             4  PORT 3 DATA DIRECTION REGISTER  *
(0005)                98 P4DDR           EQU             5  PORT 4 DATA DIRECTION REGISTER  *
(0006)                99 P3DATA          EQU             6  PORT 3 DATA REGISTER            *
(0007)               100 P4DATA          EQU             7  PORT 4 DATA REGISTER            *
(000F)               101 P3CSR           EQU            15  PORT 3 CONTROL AND STATUS       *
(0010)               102 SCI_RM          EQU            16  RATE MODE CONTROL               *
(0011)               103 SCI_TR_CS       EQU            17  TRANSMITT RECEIVE CONTROL STATUS*
                     104 *                                                                 *
                     105 *****************************************************************
                     106 *
                     107 *
                     108 *
                     109 *
                     110 *****************************************************************
                     111 *               DCB CONSTANTS                                   *
                     112 *                                                               *
                     113 *                                                               *
(0000)               114 DCB_IDLE        EQU   0
(0001)               115 DCB_STATUS      EQU   1                                       *
(0002)               116 DCB_RESET       EQU   2                                       *
(0003)               117 DCB_WT          EQU   3                                       *
(0004)               118 DCB_RD          EQU   4                                       *
                     119 *                                                             *
                     120 ***********************************  ***************************
                     121 *
                     122 ------------------------------------------------------------------
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                    123 *              DCB OFFSETS                                  *
                    124 *                                                           *
                    125 *                                    DCB                    *
                    126 *                                BLOCK-ORIENTED             *
                    127 *                                ---------------            *
          (0000)    128 DCB_CMD_STAT    EQU   0         |               |           *
          (0001)    129 DCB_BA_LO       EQU   1         |               |           *
          (0002)    130 DCB_BA_HI       EQU   2         |               |           *
          (0003)    131 DCB_BUF_LEN_LO  EQU   3         |               |           *
          (0004)    132 DCB_BUF_LEN_HI  EQU   4         |               |           *
          (0005)    133 DCB_SEC_NUM_0   EQU   5         |               |           *
          (0006)    134 DCB_SEC_NUM_1   EQU   6         |               |           *
          (0007)    135 DCB_SEC_NUM_2   EQU   7         |               |           *
          (0008)    136 DCB_SEC_NUM_3   EQU   8         |               |           *
          (0009)    137 DCB_DEV_NUM     EQU   9         |               |           *
                    138 *reserved       EQU   10        |               |           *
                    139 *reserved       EQU   11        |               |           *
                    140 *reserved       EQU   12        |               |           *
                    141 *reserved       EQU   13        |               |           *
          (000E)    142 DCB_RETRY_LO    EQU   14        |               |           *
          (000F)    143 DCB_RETRY_HI    EQU   15        |               |           *
          (0010)    144 DCB_ADD_CODE    EQU   16        |               |           *
          (0011)    145 DCB_MAXL_LO     EQU   17        |               |           *
          (0012)    146 DCB_MAXL_HI     EQU   18        |               |           *
          (0013)    147 DCB_DEV_TYPE    EQU   19        |               |           *
          (0014)    148 DCB_NODE_TYPE   EQU   20        |               |           *
                    149 *                                ---------------            *
                    150 *                                                           *
                    151 *                                                           *
          (0004)    152 PCB_SIZE        EQU   4         SIZE OF PCB BLOCK (BYTES)
          (0015)    153 DCB_SIZE        EQU   21        SIZE OF DCB BLOCK (BYTES)  *
                    154                                                           *
                    155 ***********************************************************************
                    156 *
                    157
                    158 * IN Z80 MEMORY:
                    159 *
                    160 * ADDR 0-1-(15*DCB_SIZE)-PCB_SIZE:              PCB
                    161 *      0-1-((15-0)*DCB_SIZE)                    DCB 1
                    162 *      0-1-((15-1)*DCB_SIZE)                    DCB 2
                    163 *            ETC
                    164 *      0-1                                      RESERVED
                    165 *
                    166 *
                    167 *               PCB             COMMANDS
                    168 *
        (FFFFFEC0)  169 INIT_PCB_ADDR   EQU  0-1-(15*DCB_SIZE)-PCB_SIZE
                    170
          (0000)    171 PCB_IDLE        EQU  0          IDLE COMMAND
          (0001)    172 PCB_SYNC1       EQU  1
          (0002)    173 PCB_SYNC2       EQU  2
          (0003)    174 PCB_SNA         EQU  3          PCB SET NEW ADDRESS
          (0004)    175 PCB_RESET       EQU  4          PCB RESET M6801
          (0005)    176 PCB_WAIT        EQU  5          PCB WAIT UNTIL FURTHER NOTICE
          (0004)    177 PCB_LENGTH      EQU  PCB_SIZE   LENGTH OF THE PCB BLOCK
          (0003)    178 DCB_CNT_OFFSET  EQU  3          OFFSET TO THE PCB TO THE NUMBER
                    179                 INCLUDE         1_NIM_MR
```

LOCATION OBJECT CODE LINE SOURCE LINE



LOCATION OBJECT CODE LINE SOURCE LINE

```
+ ********************************************************************
+ *              INTERFACE MODULE DESCRIPTION                        *
+ *              ----------------------------                        *
+ *  NAME:                                                           *
+ *   I_NIM_MR                                                       *
+ *                                                                  *
+ *  FUNCTION:                                                       *
+ *      TO DEFINE THE INTERFACE BETWEEN THE 6801_MASTER_MAC AND     *
+ *      6801_MASTER_APP.                                            *
+ *                                                                  *
+ *  DESCRIPTION:                                                    *
+ *      A BLOCK OF MEMORY WILL BE SHARED BY THE MAC AND APP,        *
+ *      WHEREIN DATA AND CONTROL SIGNALS WILL BE PASSED BACK        *
+ *      AND FORTH BETWEEN THE TWO. A DIAGRAM OF THIS BLOCK          *
+ *      (REFERRED TO AS NIM_BLOCK) FOLLOWS:                         *
+ *                                                                  *
+ *            NIM_DMA_BLK                                           *
+ *          +-+----+------+                                         *
+ *          |S|////|DEV_ID|  A(W/R), M(R/RESET);                    *
+ *          +-+----+------+                                         *
+ *          | MEM_PTR_HI  |  A(W), M(R  );                          *
+ *          +-------------+                                         *
+ *          | MEM_PTR_LO  |    "     "                              *
+ *          +-------------+                                         *
+ *          | MEM_LEN_HI  |  A(W), M(R);                            *
+ *          +-------------+                                         *
+ *          | MEM_LEN_LO  |  A(W), M(R);                            *
+ *          +-------------+                                         *
+ *          | A_SIGNAL    |  A(W), M(R);                            *
+ *          +-------------+                                         *
+ *          | M_SIGNAL    |  A(R/RESET), M(W);                      *
+ *          +-------------+                                         *
+ *                                                                  *
+ *  NOTES:                                                          *
+ *     WHEN THE APP CALLS THE MAC THE FOLLOWING MUST TAKE PLACE:    *
+ *                                                                  *
+ *       1. APP SETS A_SIGNAL TO THE DESIRED FUNCTION.              *
+ *       2. APP SETS M_SIG TO IO_IDLE. (0)                          *
+ *       3. APP SETS MEM_PTR IF APPLICABLE.                         *
+ *       4. APP SETS MEM_LEN IF APPLICABLE.                         *
+ *       5. LASTLY, THE APP SETS DEVICE ID AND                      *
+ *          APP SETS CNFG_WORD.BIT_7.                               *
+ *       6. APP CALLS MAC.                                          *
+ *       7. MAC WILL EXECUTE ORDERS OF MAC.                         *
+ *       8. WHEN MAC IS COMPLETE, IT WILL WRITE TO NIM.M_SIGNAL     *
+ *          AS WELL AS CLR THE CNFG_WORD.BIT_7.                     *
+ *       9. MAC WILL DEPOSIT THE TRUE LENGTH OF DATA WRITTEN        *
+ *          TO Z80_MEM IN MEM_LEN. THUS, APP SHOULD HAVE SAVED      *
+ *          ORIGINAL MEM_LEN BEFORE CALL TO MAC.                    *
+ *      10. THE APP WILL READ THE NIM AND CLEAR A_SIG, M_SIG,       *
+ *          MEM_PTR, MEM_LEN AND CNFG_WORD. (BIT 7=0 ALREADY)       *
+ *                                                                  *
+ ********************************************************************
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                    + ************************************************************************
                    + *                                                                      *
                    + *  DATA ELEMENT DEFINITIONS:                                           *
                    + *                                                                      *
                    + *     CNFG_WORD :                                                      *
                    + *     ----------                                                       *
                    + *       BIT.7:  CLR = APP SHOULD NOT CALL MAC                          *
                    + *               SET = APP SHOULD CALL MAC                              *
                    + *                                                                      *
                    + *     A_SIG:                                                           *
                    + *     -----                                                            *
                    + *        0- NO SIGNAL (IDLE).                                          *
                    + *        1- RESET DEVICE D                                             *
                    + *        2- GET STATUS FROM DEVICE D.                                  *
                    + *        3- GET DATA FROM DEVICE D.                                    *
                    + *        4- PUT DEVICE TO DEVICE TO D.                                 *
                    + *                                                                      *
(0001)              + ASIG_RESET_D    EQU             001H
(0002)              + ASIG_STAT_D     EQU             002H
(0003)              + ASIG_GET_D      EQU             003H
(0004)              + ASIG_PUT_D      EQU             004H
(0004)              + ASIG_MAX_CODE   EQU             ASIG_PUT_D
                    + *     MEM_POINTER                                                      *
                    + *     -----------                                                      *
                    + *                                                                      *
                    + *        (CAN ACCESS UP TO 64K OF Z80 MEMORY)                          *
                    + *                                                                      *
                    + *     A_LEN:                                                           *
                    + *     -----                                                            *
                    + *                                                                      *
                    + *        (CAN RANGE UP TO 64K)                                         *
                    + *                                                                      *
                    + *     M_SIG:                                                           *
                    + *     -----                                                            *
                    + *        0- IO_IDLE                                                    *
                    + *        1- IO_COMPLETE.                                               *
                    + *        2- IO_BUSY                                                    *
                    + *        3- IO_FAILED                                                  *
                    + *                                                                      *
(0000)              + MSIG_IDLE       EQU             000H
(0080)              + MSIG_COMPLETE   EQU             080H
(0002)              + MSIG_TIMEOUT    EQU             002H
(0003)              + MSIG_FAILED     EQU             003H
                    + *                                                                      *
                    + *   NOTES:                                                             *
                    + *     * 1. A:= APPLICATION SIDE OF NODE.                               *
                    + *       2. M:= MAC SIDE OF NODE.                                       *
                    + *       3. x(W):= THIS ELEMENT HAS WRITE ACCESS TO THIS BYTE.          *
                    + *       4. x(R/RESET):= THIS ELEMENT HAS READ ACCESS AS WELL AS        *
                    + *          RESET ACCESS( WRITE ZERO ONLY) TO THIS BYTE.                *
                    + *                                                                      *
                    + ************************************************************************
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      + ********************************************************************
                      + *                                                                  *
                      + *  NOTES TO INSTALLER OF THIS MAC/APP:                             *
                      + *                                                                  *
                      + *   1. THE APP IS RESPONSIBLE FOR INITIALIZING ALL OF RAM.         *
                      + *                                                                  *
                      + *   2. THE APP MUST INITIALIZE THE CONTROL AND STATUS REG AS       *
                      + *      WELL AS THE BAUD RATE AND MODE REGISTER.                    *
                      + *                                                                  *
                      + *   3. THE D1_MODE_WORD MUST BE SET TO ZERO AT PWR UP BY THE       *
                      + *      APP.                                                        *
                      + *                                                                  *
                      + *   4. THE NIM_BLOCK WILL END AT ADDR 255 DECIMAL.                 *
                      + *                                                                  *
                      + ********************************************************************
                       80 *                                  OF DCB'S
                      181 *
                      182 *
                      183 *                MAC INTERFACE CONSTANTS
                      184 *
(0001)                185 MAC_DEV_RESET   EQU  ASIG_RESET_D
(0002)                186 MAC_GET_STAT    EQU  ASIG_STAT_D
(0003)                187 MAC_GET_DATA    EQU  ASIG_GET_D
(0004)                188 MAC_PUT_DATA    EQU  ASIG_PUT_D
                      189 *
                      190 *
                      191 *        M_SIGNAL RETURNS
                      192 *
                      193 * ***** WARNING THIS SHOULD BE SYMBOLIC
(0000)                194 IO_IDLE         EQU  0
(0080)                195 IO_COMPLETE     EQU  80H
(0002)                196 IO_BUSY         EQU  2
(0003)                197 IO_FAILED       EQU  3
                      198 *
                      199 *
                      200 * ERROR NUMBERS TO BE RETURNED TO THE Z80
                      201 *
(00E0)                202 ILLEGAL_ADD     EQU   0E0H      BUFFER MAX DID NOT EQUAL BUFFER SIZE IN
                      203 *                               A BLOCK STRUCTURED DEVICE
(00E1)                204 CH_BAD_SIZE     EQU   0E1H      BUFFERSIZE GREATER THAN MAX COUNT.
(00E2)                205 ILLEGAL_CMD     EQU   0E2H      ILLEGAL COMMAND
                      206 *
                      207 *
                      208 *               ORG     00B0H           START OF APP RAM
                      209 *
                      210                 DATA
                      211                 ORG     97H
(0020)                212 STACKSIZE       EQU     20H
0097                  213 STACK           RMB     STACKSIZE       INITIAL; STACK VALUE
(00B6)                214 STACKSTART      EQU     $-1
00B7                  215 PCB_LOC         RMB     2               Z80 PCB BUFFER
00B9                  216 DCB_COUNT       RMB     1               LOOP POINTER
00BA                  217 DCB_CUR_PNTR    RMB     2               POINTER TO CURRENT DCB BLOCK
00BC                  218 TEMP_D          RMB     2               TEMP STORAGE FOR THE D REG.
00BE                  219 B_START         RMB     2                 ART POINTER FOR CHAR TRANSFERS
00C0                  220 B_END           RMB     2                 .D POINTER FOR CHAR TRANSFERS
00C2                  221 B_IN            RMB     2               IN POINTER FOR CHAR TRANSFERS
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    00C4              222 B_OUT           RMB     2               OUT POINTER FOR CHAR TRANSFERS
    00C6              223 B_LENGTH        RMB     2               DCB MAX LENGTH TEMP STORE
    00C8              224 B_SIZE          RMB     2               SIZE DATA PART OF THE CHAR
                      225 *                                       DATA FILE
                      226
                      227 *
                      228 *       Network interface module area
                      229 *       shared by both the MAC and APP
                      230 *       of a node. This is the common
                      231 *       communication area used to effect
                      232 *       network data tranfers by the MAC and
                      233 *       by the APP
                      234 *
                      235 *
                      236
                      237                 COMN
                      238 *               ORG             00F9H           START ADDRESS OF NIM
    0000              239 NIM_DMA_BLK     RMB             1               |SI////IDEV_ID|
    0001              240 MEM_PTR
    0001              241 MEM_PTR_HI      RMB             1               | MEM_PTR_HI  |
    0002              242 MEM_PTR_LO      RMB             1               | MEM_PTR_LO  |
    0003              243 MEM_LEN
    0003              244 MEM_LEN_HI      RMB             1               | MEM_LEN_HI  |
    0004              245 MEM_LEN_LO      RMB             1               | MEM_LEN-LO  |
    0005              246 A_SIGNAL        RMB             1               | A_SIGNAL    |
    0006              247 M_SIGNAL        RMB             1               | M_SIGNAL    |
                      248
                      249 *
                      250 *               END OF NIM AREA (00FFH)
                      251 *
                      252 *
                      253 *               START OF THE MASTER APPLICATION CODE
                      254 *
                      255 *       This code is used in the master M6801 networking
                      256 *       control to manage the orderly flow of data on the
                      257 *       network.
                      258 *
                      259 *
                      260 ************************************************************
                      261 *                                                          *
                      262 *    ACK_Z80 acknowledge the command in the Z80            *
                      263 *    memory.                                               *
                      264 *    enter with                                            *
                      265 *                                                          *
                      266 *       X      address of the Z_80 command/status location*
                      267 *                                                          *
                      268 ************************************************************
                      269 *
                      270                 PROG
    0000              271 ACK_Z80
    0000 BD0000       272                 JSR   MMR_DMA_READ GET THE CURRENT VALUE
    0003 8A80         273                 ORAA  #80H      SET THE TOP BIT
    0005 BD0000       274                 JSR   MMR_DMA_WRITE  WRITE THE DATA BACK
    0008 39           275                 RTS             (SAVE A BYTE)
                      276 *
                      277 *
                      278 ************************************************************
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     279 *                                                    *
                     280 *     GET_PCB_CMD get pcb command byte               *
                     281 *     enters    (no args)                            *
                     282 *     returns                                        *
                     283 *          A    PCB command                          *
                     284 *          X,B  contents are lost                    *
                     285 *                                                    *
                     286 ******************************************************
                     287
0009                 288 GET_PCB_CMD
0009 DE87            289                 LDX   PCB_LOC   POINT TO THE PCB BUFFER
000B BD0000          290                 JSR   MMR_DMA_READ
000E 39              291                 RTS
                     292
                     293 *
                     294 ******************************************************************
                     295 *                                                                *
                     296 *     GET_NEW_PCB_CMD get pcb command byte                       *
                     297 *       wait for the last command to be replaced                 *
                     298 *       by a new command.                                        *
                     299 *                                                                *
                     300 *     enters    (no args)                                        *
                     301 *     returns                                                    *
                     302 *          A    PCB command                                      *
                     303 *          X,B  contents are lost                                *
                     304 *                                                                *
                     305 ******************************************************************
                     306
000F                 307 GET_NEW_PCB_CMD
                     308
                     309 ***** 8/31/83 START *****
000F CE0053          310                 LDX   #83
0012                 311 D_LAY:
0012 09              312                 DEX
0013 26FD            313                 BNE   D_LAY
                     314 ***** 8/31/83 END   *****
                     315
0015 DE87            316                 LDX   PCB_LOC       POINT TO THE PCB BUFFER
0017 BD0000          317                 JSR   MMR_DMA_READ
001A 4D              318                 TSTA                IS IT PREVIOUS COMMAND
001B 2BF2            319                 BMI   GET_NEW_PCB_CMD
001D 39              320                 RTS
                     321
                     322 ******************************************************************
                     323 *                                                                *
                     324 *    INIT the init routine is responsible for                    *
                     325 *       1) Serial port init including the                        *
                     326 *          port init                                             *
                     327 *                                                                *
                     328 *       2) init all ram variables                                *
                     329 *                                                                *
                     330 *       3) Must execute the SYNC1,SYNC2 startup                  *
                     331 *          sequence.                                             *
                     332 *                                                                *
                     333 *       4) Init PCB location.                                    *
                     334 *                                                                *
                     335 ******************************************************************
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     336
001E 7F0005          337 INIT            CLR   A_SIGNAL          DONT START ANYTHING
0021 86FF            338                 LDAA  #0FFH             SHOW DONE
0023 B70006          339                 STAA  M_SIGNAL
0026 CEFEC0          340                 LDX   #INIT_PCB_ADDR
0029 DFB7            341                 STX   PCB_LOC
                     342 *
                     343 *               Serial port register initialization
                     344 *
002B 8600            345                 LDAA #0
002D 9700            346                 STAA P1DDR
002F 8610            347                 LDAA #10H         SET PORT 2 BIT 4 OUT
0031 9701            348                 STAA P2DDR
0033 8600            349                 LDAA #0
0035 9704            350                 STAA P3DDR
0037 9705            351                 STAA P4DDR
0039 8604            352                 LDAA #04H         INIT RATE AND MODE
003B 9710            353                 STAA SCI_RM       TO 62.5K AND NRZ
003D 860A            354                 LDAA #0AH         TE AND RE ON THE TRCS REG
003F 9711            355                 STAA SCI_TR_CS
                     356
                     357                 GLB  SA
0041                 358 SA:
                     359 *
                     360 *
0041 CE0000          361                 LDX   #0                START UP DELAY WAITING
0044 09              362 INIT_LOOP       DEX                     FOR ALL OF THE Z80
0045 26FD            363                 BNE   INIT_LOOP         HARDWARE TO POWER UP
0047 8D35            364                 BSR   PCB_SYNC          STARTUP SYNC PROTOCOL
                     365 *                                       TO THE Z80.
0049 39              366                 RTS
                     367
                     368 ************************************************************
                     369 *                                                          *
                     370 *     Start of the main line of code.                      *
                     371 *     Initialization followed by the scanner routine.      *
                     372 *                                                          *
                     373 *                                                          *
                     374 *                                                          *
                     375 ************************************************************
                     376
004A 8E00B6          377 START       LDS  #STACKSTART        INIT STACK
                     378
004D 8DCF            379             BSR  INIT               GET THE 6801 INTO A SAFE STATE.
                     380
                     381 *                                   MAIN EXECUTION LOOP OF THE APP
004F 8D54            382 SCANNER     BSR  PROCESS_PCB        CHECK FOR COMMANDS TO PROCESSOR
                     383
0051 DEB7            384             LDX  PCB_LOC            POINT TO PCB
0053 C603            385             LDAB #DCB_CNT_OFFSET
0055 3A              386             ABX
0056 BD0000          387             JSR  MMR_DMA_READ      GET DCB_COUNT
0059 97B9            388             STAA DCB_COUNT          LOOP COUNT
                     389
005B [illegible]7    390             LDX  PCB_LOC            POINT [illegible]CB
005D [illegible]04   391             LDAB #PCB_LENGTH
005F 3A              392             ABX
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    0060 DFBA          393            STX  DCB_CUR_PNTR     POINT TO CURRENT DCB BLOCK
                       394
    0062               395 DCB_LOOP:
    0062 CE0053        396              LDX  #83              500 uSEC
    0065               397 SCAN_1:
    0065 09            398              DEX
    0066 26FD          399              BNE  SCAN_1
                       400
    0068 7D00B9        401            TST  DCB_COUNT        ARE WE DONE
    0068 27E2          402            BEQ  SCANNER          YES
                       403
    006D DEBA          404              LDX  DCB_CUR_PNTR
    006F BD0103        405              JSR  PROCESS_DCB    PROCESS THE BLOCK
                       406
    0072 DEBA          407              LDX  DCB_CUR_PNTR
    0074 C615          408              LDAB #DCB_SIZE      POINT TO NEXT BLOCK
    0076 3A            409              ABX
    0077 DFBA          410              STX  DCB_CUR_PNTR
                       411
    0079 7A00B9        412              DEC  DCB_COUNT      DONE?
    007C 20E4          413            BRA    DCB_LOOP
                       414
                       415 ************************************************************
                       416 *                                                          *
                       417 *    PCB_SYNC  pcb_sync is used to sync the M6801 and      *
                       418 *     the Z80 to prevent the M6801 from executing the      *
                       419 *     that may be found randomly at 8000H on powerup       *
                       420 *                                                          *
                       421 ************************************************************
    007E               422 PCB_SYNC
    007E CE067C        423                  LDX  #83*20         DELAY ~10000 uSEC
    0081               424 SYNC1_DELAY
    0081 09            425                  DEX
    0082 26FD          426                  BNE  SYNC1_DELAY
                       427
    0084 8D83          428                  BSR  GET_PCB_CMD    GET COMMAND BYTE
    0086 8101          429                  CMPA #PCB_SYNC1
    0088 26F4          430                  BNE  PCB_SYNC       MUST GET BYTES IN ORDER
                       431
    008A DEB7          432                  LDX  PCB_LOC        ACKNOWLEDGE THE COMMAND
    008C BD0000        433                  JSR  ACK_Z80
                       434
    008F               435 WAIT_NEW_COMMAND:
    008F CE067C        436                  LDX  #83*20         DELAY ~10000 uSEC
    0092               437 SYNC2_DELAY
    0092 09            438                  DEX
    0093 26FD          439                  BNE  SYNC2_DELAY
                       440
    0095 BD0009        441                  JSR  GET_PCB_CMD    GET COMMAND BYTE
    0098 4D            442                  TSTA                NEW COMMAND YET?
    0099 2BF4          443                  BMI  WAIT_NEW_COMMAND NOPE
                       444
    009B 8102          445                  CMPA #PCB_SYNC2
    009D 26DF          446                  BNE  PCB_SYNC       MUST GET BYTES IN ORDER
                       447
    009F DEB7          448                  LDX  PCB_LOC        A   OWLEDGE THE COMMAND
    00A1 BD0000        449                  JSR  ACK_Z80
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     450
00A4 39              451                  RTS                  BOTH SYNCS HAVE BEEN
                     452 *                                     FOUND IN ORDER
                     453 ************************************************************
                     454 *                                                          *
                     455 *   PROCESS_PCB command. The PCB buffer is used to         *
                     456 *      commands from the Z80 host to the M6801             *
                     457 *      network controller.                                 *
                     458 *                                                          *
                     459 ************************************************************
00A5                 460 PROCESS_PCB
00A5 BD0009          461                  JSR GET_PCB_CMD    GET THE COMMAND
00A8 4D              462                  TSTA               IS IT AN OLD COMMAND
00A9 2B1E            463                  BMI  P_PCB_EXIT
                     464
                     465                  IF   0
                     466
                     467                  CMPA #PCB_IDLE     IS IT A IDLE SEQUENCE
                     468                  BNE  PCB1
                     469                  LDX  #83           IF WE GET AN IDLE SLOW DOWN THE SCAN
                     470 SCAN_2           DEX                DELAY FOR 500 Usec.
                     471                  BNE  SCAN_2
                     472                  LDX  PCB_LOC       ACKNOWLEDGE THE IDLE REQUEST
                     473                  JSR  ACK_Z80
                     474                  BRA  P_PCB_EXIT
                     475
                     476                  ENDIF
                     477
00AB 8101            478 PCB1             CMPA #PCB_SYNC1    IS IT A SYNC
00AD 2604            479                  BNE  PCB2          BRANCH NOT A SYNC1
00AF 8DCD            480                  BSR  PCB_SYNC
00B1 2016            481                  BRA  P_PCB_EXIT
00B3 8103            482 PCB2             CMPA #PCB_SNA      PCB SET NEW ADDRESS
00B5 2604            483                  BNE  PCB3
00B7 8D11            484                  BSR  P_SNA
00B9 200E            485                  BRA  P_PCB_EXIT
00BB 8104            486 PCB3             CMPA #PCB_RESET    PCB RESET?
00BD 2604            487                  BNE  PCB4
00BF 8D19            488                  BSR  P_RESET
00C1 2006            489                  BRA  P_PCB_EXIT
00C3 8105            490 PCB4             CMPA #PCB_WAIT
00C5 2602            491                  BNE  PCB5
00C7 8D2B            492                  BSR  P_WAIT
                     493 *                BRA  P_PCB_EXIT
00C9                 494 PCB5
00C9 39              495 P_PCB_EXIT       RTS
                     496 *
                     497 ************************************************************
                     498 *                                                          *
                     499 *     PCB_SET_NEW_ADDRESS get the new PCB address          *
                     500 *        from the Z80 host computer.                       *
                     501 *                                                          *
                     502 *                                                          *
                     503 ************************************************************
00CA                 504 P_SNA
00CA E6 07           505                  LDX  PCB_LOC     GET CURRENT PCB ADDRESS
00CC 08              506                  INX
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    00CD BD01A6       507                 JSR  GET_Z80_WORD
    00D0 DEB7         508                 LDX  PCB_LOC     ACKNOWLEDGE OLD MOVE COMMAND
    00D2 DDB7         509                 STD  PCB_LOC     NEW PCB ADDRESS
    00D4 BD0000       510                 JSR  ACK_Z80     ACKNOWLEDGE THE COMMAND
    00D7 8DCC         511                 BSR  PROCESS_PCB RECURSIVELY PROCESS NEW PCB
    00D9 39           512                 RTS
                      513 *
                      514 ****************************************************************
                      515 *                                                              *
                      516 *   PCB_RESET  pcb soft reset each of the interfaces           *
                      517 *     on the bus.                                              *
                      518 *                                                              *
                      519 *                                                              *
                      520 ****************************************************************
    00DA              521 P_RESET
    00DA C680         522                 LDAB  #80H           INIT ADDRESS AND S VALUE
    00DC              523 P_RES_LOOP
    00DC 8601         524                 LDAA  #MAC_DEV_RESET
    00DE B70005       525                 STAA  A_SIGNAL       SET UP THE COMMAND
    00E1 5C           526                 INCB                 DEVICE ADDRESS
    00E2 F70000       527                 STAB  NIM_DMA_BLK    SET UP DEVICE NUMBER
    00E5 37           528                 PSHB                 SAVE THE ADDRESS
    00E6 BD0000       529                 JSR   MMR_MAC                CALL "BIG MAC"
    00E9 33           530                 PULB
    00EA C18F         531                 CMPB  #8FH           IS IT DONE
    00EC 26EE         532                 BNE   P_RES_LOOP
    00EE DEB7         533                 LDX  PCB_LOC         ACKNOWLEDGE THE RESET REQUEST
    00F0 BD0000       534                 JSR  ACK_Z80
    00F3 39           535                 RTS                  DONE
                      536 *
                      537 ****************************************************************
                      538 *                                                              *
                      539 *  PCB_WAIT halt all transactions to the Z80 except            *
                      540 *     for periodic checking of the PCB.                        *
                      541 *
                      542 *
                      543 ****************************************************************
    00F4 DEB7         544 P_WAIT          LDX  PCB_LOC          ACKNOWLEDGE THE WAIT
    00F6 BD0000       545                 JSR  ACK_Z80
    00F9 BD000F       546 P_WAIT_LOOP     JSR  GET_NEW_PCB_CMD      SEE IF COMMAND HAS CHANGED
    00FC 8105         547                 CMPA  #PCB_WAIT
    00FE 27F9         548                 BEQ   P_WAIT_LOOP     SLOW DOWN ACCESSES WITH DELAY
                      549 *                                     IN THIS LOOP
    0100 8DA3         550                 BSR   PROCESS_PCB     PROCESS NEW COMMAND
    0102 39           551                 RTS
                      552 *
                      553 ****************************************************************
                      554 *                                                              *
                      555 *    PROCESS_DCB comand block from the Z80                     *
                      556 *      enter with                                              *
                      557 *        X      pointing to the DCB                            *
                      558 *      exit  all registers destroyed                           *
                      559 *                                                              *
                      560 ****************************************************************
    0103              561 PROCESS_DCB
    0103 3C           562                 PSHX                  SAVE DCB ADDRESS
    0104 BD0000       563                 JSR  MMR_DMA_READ     GET THE DCB COMMAND
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    0107 38           564                 PULX                GET ADDRESS
    0108 4D           565                 TSTA                IS THE COMMAND DONE?
    0109 2B2B         566                 BMI  P_DCB_XIT      YES EXIT
    010B 840F         567                 ANDA #0FH           CHECK THE COMMAND
    010D 2602         568                 BNE  P_NON_IDLE     NON IDLE
                      569 *               JSR  ACK_Z80        ACKNOWLEDGE IDLE
    010F 2025         570                 BRA  P_DCB_XIT      DONE
    0111 8102         571 P_NON_IDLE      CMPA #DCB_RESET     IS IT A RESET COMMAND
    0113 2604         572                 BNE  DCB2
    0115 8D4F         573                 BSR  DCB_DEV_RESET
    0117 201D         574                 BRA  P_DCB_XIT
    0119 8101         575 DCB2            CMPA #DCB_STATUS    IS IT LOOKING FOR STATUS
    011B 2604         576                 BNE  DCB3
    011D 8D60         577                 BSR  DCB_DEV_STATUS
    011F 2015         578                 BRA  P_DCB_XIT
    0121 36           579 DCB3            PSHA                SAVE THE COMMAND BYTE
    0122 3C           580                 PSHX                RESAVE THE ADDRESS
    0123 C613         581                 LDAB #DCB_DEV_TYPE  POINT TO DEVICE TYPE
    0125 3A           582                 ABX
    0126 BD0000       583                 JSR  MMR_DMA_READ
    0129 38           584                 PULX                GET BACK THE DCB ADDRESS
    012A 16           585                 TAB                 MOVE RESPONSE TO B
    012B 32           586                 PULA                COMMAND BACK IN A
    012C C501         587                 BITB #1             BLOCK DEVICE ?
    012E 2704         588                 BEQ  P_CHAR_DEV
    0130 8D05         589                 BSR  P_DCB_BLOCK    PROCESS BLOCK DEVICE
    0132 2002         590                 BRA  P_DCB_XIT
    0134 8D18         591 P_CHAR_DEV      BSR  P_DCB_CHAR     PROCESS CHAR DEVICE
    0136 39           592 P_DCB_XIT       RTS                 DONE PROCESS DCB
                      593 *
                      594 *
                      595 *
                      596 ************************************************************
                      597 *READ DATA  This subroutine will read one byte from       *
                      598 *     the Z80 memory                                       *
                      599 *                                                          *
                      600 *                                                          *
                      601 *Enter this subroutine with the following arguments       *
                      602 *                                                          *
                      603 *      X        address of the location to be read       *
                      604 *                                                          *
                      605 * Return with                                             *
                      606 *                                                          *
                      607 *      A        (Z80X)                                   *
                      608 *      B        contents are saved                       *
                      609 *      X        contents are saved                       *
                      610 *                                                          *
                      611 *                                                          *
                      612 ************************************************************
                      613 *
                      614 ****MMR_DMA_READ   PSHB            SAVE THE B REGISTER
                      615 ****              PSHX            SAVE THE INDEX
                      616
                      617 ****              JSR  MMR_DMA_READ
                      618
                      619 ***                   LDAA  #010H     SE.   LATCH ENABLE AND OUTPUT
                      620 ***                   STAA  P3CSR     STROBE SELECT
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      621 ***                 PSHX            PUT THE ADDRESS WHER BIA REGISTERS
                      622 *** *                               CAN GET IT
                      623 ***                 LDAA  #01EH     PORT 1 CONTROL BITS.
                      624 ***                 STAA  P2DATA    GET ALL THE DATA READY
                      625 ***                 PULA            GET HIGH ADDRESS
                      626 ***                 STAA  P1DATA    PORT2 HAS HIGH ADDRESS BITS
                      627 ***                 PULA            Z80 LOW ADDRESS BYTE
                      628 ***                 STAA  P4DATA
                      629 ***                 LDD   #015FFH   SET A TO FF  B TO 00
                      630 ***                 LDX   #0010H    CONSTANTS TO RECONFIGURE FOR INPUTS
                      631 ***                 STS   STACK_TEMP SAVE THE STACK USE IT AS AN INDEX
                      632 ***                 LDS   #0000H
                      633 *** *
                      634 *** *
                      635 *** *                   ACTUAL DMA DATA READ STARTS HERE
                      636 *** *
                      637 *** *
                      638 ***                 STAA  P2DDR     ISSUE THE DMA REQUEST
                      639 ***                 STAB  P4DDR     ENABLE LOW  ADDRESS OUTPUTS
                      640 ***                 STAB  P1DDR     ENABLE HIGH ADDRESS OUTPUTS
                      641 ***                 STAB  P3DATA    DUMMY WRITE TO PORT 3
                      642 ***                 STS   P3DDR
                      643 ***                 STX   P1DDR
                      644 ***                 LDAA  P3DATA    GET THE DATA LATCHED IN PORT 3
                      645 *** *
                      646 *** *
                      647 *** *                   Z80 READ IS COMPLETE DATA IS IN ACCUMULATOR A
                      648 *** *
                      649 *** *
                      650 ***                 LDS   STACK_TEMP  RESTORE THE STACK REGISTER
                      651
                      652
                      653
                      654 ***             PULX              RESTORE THE INDEX
                      655 ***             PULB              RESTORE B
                      656 ***             RTS               READ IS DONE
                      657 *
                      658 *
                      659 *
                      660 *
                      661 ************************************************************
                      662 *                                                          *
                      663 *WRITE DATA BYTE                                          *
                      664 *                                                          *
                      665 *      This subroutine will write a single byte of         *
                      666 *      data to the Z80's memory.                           *
                      667 *                                                          *
                      668 *Enter with the following arguments                       *
                      669 *      X         Address in the Z80's memory               *
                      670 *      A         Data byte.                                *
                      671 *Return                                                   *
                      672 *      Contents of A,B,X are saved.                        *
                      673 *                                                          *
                      674 ************************************************************
                      675 *
                      676 ***MMR_DMA_WRITE  PSHA             SAVE THE REGISTERS
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     678 ****            PSHX
                     679 ****
                     680 ****            JSR  MMR_DMA_WRITE
                     681
                     682 ***                 CLR   P3CSR      SET UP P3 CONTROL PORT
                     683 ***                 STAA  P3DATA
                     684 ***                 PSHX
                     685 ***                 PULA             GET HIGH ADDRESS BYTE
                     686 ***                 STAA  P1DATA     LOAD HIGH ADDRESS REGISTER
                     687 ***                 PULA             GET LOW ADDRESS BYTE
                     688 ***                 STAA  P4DATA
                     689 ***                 LDAA  #01AH      SET UP CONTROL BITS FOR WRITE
                     690 ***                 STAA  P2DATA
                     691 ***                 LDD   #015FFH    CONSTANT TO SET PORT OUTPUTS
                     692 ***                 LDX   #00010H    CONSTANT TO SET PORT INPUTS
                     693 ***
                     694 ***                 STS   STACK_TEMP SAVE THE STACK REGISTER
                     695 ***                 LDS   #00000H    USE STACK AS A WORD WIDE REGISTER
                     696 *** *
                     697 *** *           START OF THE ACTUAL TRANSFER
                     698 *** *
                     699 ***                 STAA  P2DDR      SET UP FOR OUTPUTS
                     700 ***                 STAB  P3DDR      PORT 3/4 ENABLED OUT
                     701 ***                 STAB  P4DDR
                     702 ***                 STAB  P1DDR      SET THE DMA
                     703 ***                 LDAA  P3DATA     DUMMY READ
                     704 ***                 STS   P3DDR
                     705 ***                 STX   P1DDR      REMOVE ALL REGISTERS FROM THE BUS
                     706 *** *
                     707 *** *
                     708 ***                 LDS   STACK_TEMP RESTORE THE STACK
                     709
                     710
                     711 ***             PULX
                     712 ***             PULB
                     713 ***             PULA             RESTORE ALL REGISTERS
                     714 ***             RTS              DONE
                     715 *
                     716 ************************************************************
                     717 *                                                          *
                     718 *   PROCESS DCB on a block structured device.              *
                     719 *                                                          *
                     720 *                                                          *
                     721 ************************************************************
0137 8103            722 P_DCB_BLOCK     CMPA  #DCB_WT    IS IT A WRITE TO NETWORK
0139 2605            723                 BNE   P_DCB1_BLK
013B BD0230          724                 JSR   DCB_WT_BLK
013E 200D            725                 BRA   P_DCB_EXIT
0140 8104            726 P_DCB1_BLK      CMPA  #DCB_RD    IS IT A READ TO NETWORK
0142 2707            727                 BEQ   P_DCB2_BLK
0144 86E2            728                 LDAA  #ILLEGAL_CMD  UNUSED COMMAND.
0146 BD0000          729                 JSR   MMR_DMA_WRITE WRITE TO THE Z80
0149 2002            730                 BRA   P_DCB_EXIT
014B 9D76            731 P_DCB2_BLK      BSR   DCB_RD_BLK
014I                 732 P_DCB_EXIT      RTS
                     733 *
                     734 ************************************************************
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     735 *                                                     *
                     736 *   PROCESS a DCB on a character oriented device.     *
                     737 *                                                     *
                     738 *                                                     *
                     739 *******************************************************
014E 8104            740 P_DCB_CHAR      CMPA  #DCB_RD       IS IT A DCB READ
0150 2605            741                 BNE   P_DCB1_CHAR
0152 BD029E          742                 JSR   DCB_RD_CHR
0155 200E            743                 BRA   P_DCB_CH_XIT
0157 8103            744 P_DCB1_CHAR     CMPA  #DCB_WT       IS IT A WRITE
0159 2707            745                 BEQ   P_DCB2_CHAR
015B 86E2            746                 LDAA  #ILLEGAL_CMD  UNUSED COMMAND.
015D BD0000          747                 JSR   MMR_DMA_WRITE WRITE TO THE Z80
0160 2003            748                 BRA   P_DCB_CH_XIT
0162 BD02FE          749 P_DCB2_CHAR     JSR   DCB_WT_CHR
0165 39              750 P_DCB_CH_XIT    RTS                 DONE
                     751 *******************************************************
                     752 *                                                     *
                     753 *  DCB RESET reset the device pointed to by the address   *
                     754 *      in the DCB block.                              *
                     755 *                                                     *
                     756 *    enter with                                       *
                     757 *      X     pointing to the beginning of the DCB block   *
                     758 *                                                     *
                     759 *                                                     *
                     760 *******************************************************
                     761 *
0166 3C              762 DCB_DEV_RESET   PSHX                SAVE THE DCB POINTER
0167 C610            763                 LDAB #DCB_ADD_CODE  POINT TO DEVICE ADDRESS
0169 3A              764                 ABX
016A BD0000          765                 JSR  MMR_DMA_READ
016D 8A80            766                 ORAA  #80H          ADDRESS AND S VALUE
016F                 767 DCB_RES_LOOP
016F B70000          768                 STAA  NIM_DMA_BLK   SET UP NETWORK ADDRESS
0172 8601            769                 LDAA  #MAC_DEV_RESET
0174 B70005          770                 STAA  A_SIGNAL      SET UP THE COMMAND
0177 BD0000          771                 JSR   MMR_MAC       CALL "BIG MAC"
017A 38              772                 PULX                ACKNOWLEDGE THE RESET
017B BD034A          773                 JSR  NIM_RESPONSE
017E 39              774                 RTS                 DONE
                     775 *
                     776 *
                     777 *
                     778 *******************************************************
                     779 *                                                     *
                     780 *  DCB STATUS get status block of the device pointed      *
                     781 *      to in the DCB block                            *
                     782 *                                                     *
                     783 *    enter with                                       *
                     784 *      X     pointing to the beginning of the DCB block   *
                     785 *                                                     *
                     786 *                                                     *
                     787 *******************************************************
                     788 *
017F                 789 DCB_DEV_STATUS  PSHX                SAVE THE DCB POINTER
0180 C610            790                 LDAB  #DCB_ADD_CODE PO    TO DEVICE ADDRESS
0182 3A              791                 ABX
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    0183 BD0000       792                 JSR   MMR_DMA_READ
    0186 8A80         793                 ORAA  #80H            ADDRESS AND S VALUE
    0188 B70000       794                 STAA  NIM_DMA_BLK     SET UP NETWORK ADDRESS
    018B 38           795                 PULX                  POINT TO THE STATUS BYTE
    018C 3C           796                 PSHX                  SAVE POINTER
    018D C611         797                 LDAB  #DCB_MAXL_LO    POINT TO DCB_MAXL_LO
    018F 3A           798                 ABX                   CALCULATE THE OFFSET
    0190 FF0001       799                 STX   MEM_PTR_HI      POINT TO Z80 STATUS AREA
    0193 CC0004       800                 LDD   #4              MOVE OVER 4 BYTES
    0196 FD0003       801                 STD   MEM_LEN
    0199 8602         802                 LDAA  #MAC_GET_STAT
    019B B70005       803                 STAA  A_SIGNAL        SET UP THE COMMAND
    019E BD0000       804                 JSR   MMR_MAC        CALL "BIG MAC"
    01A1 38           805                 PULX                  ACKNOWLEDGE THE RESET
    01A2 BD034A       806                 JSR   NIM_RESPONSE
    01A5 39           807                 RTS                   DONE
                      808 *
                      809 *
                      810 ************************************************************
                      811 *    GET_Z80_WORD get a double byte from the Z80            *
                      812 *       X      - address in the Z80                         *
                      813 *     returns                                              *
                      814 *       D      - data in the Z80 memory                     *
                      815 *                                                           *
                      816 ************************************************************
                      817 *                                                           *
    01A6              818 GET_Z80_WORD
    01A6 3C           819                 PSHX              SAVE THE Z80 ADDRESS ARG
    01A7 BD0000       820                 JSR MMR_DMA_READ GET LS BYTE FROM THE Z80
    01AA 36           821                 PSHA              SAVE THE LS BYTE
    01AB 08           822                 INX               POINT TO MS BYTE
    01AC BD0000       823                 JSR MMR_DMA_READ GET MS BYTRE FROM THE Z80
    01AF 33           824                 PULB              LS BYTE
    01B0 38           825                 PULX              RESTORE THE ARG
    01B1 39           826                 RTS
                      827 *
                      828 *                                                         *
                      829 ************************************************************
                      830 *    PUT_Z80_WORD put a double byte into the Z80 memory     *
                      831 *       X      - address in the Z80                         *
                      832 *     returns                                              *
                      833 *       D      - data in the Z80 memory                     *
                      834 *                                                           *
                      835 ************************************************************
                      836 *                                                         *
    01B2              837 PUT_Z80_WORD
    01B2 3C           838                 PSHX              SAVE THE Z80 ADDRESS ARG
    01B3 36           839                 PSHA
    01B4 37           840                 PSHB
    01B5 36           841                 PSHA              SAVE THE MS BYTE
    01B6 17           842                 TBA               TRANSFER THE LS BYTE
    01B7 BD0000       843                 JSR MMR_DMA_WRITE
    01BA 08           844                 INX
    01BB              845                 PULA
    01BC    000       846                 JSR MMR_DMA_WRITE
    01BF 33           847                 PULB
    01C0 32           848                 PULA
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    01C1 38           849                 PULX            RESTORE THE ARG
    01C2 39           850                 RTS
                      851 *
                      852 *
                      853 ****************************************************************
                      854 *  Read data from an external block structured device to  *
                      855 *     the Z80                                             *
                      856 *  Enter                                                  *
                      857 *       X       address of DCB in the Z80.                *
                      858 *                                                         *
                      859 ****************************************************************
                      860 *
    01C3              861 DCB_RD_BLK
    01C3 3C           862                 PSHX                    SAVE THE INDEX REGISTER
    01C4 C605         863                 LDAB  #DCB_SEC_NUM_0 GET OFFSET TO THE SECTOR ADD.
    01C6 3A           864                 ABX
    01C7 FF0001       865                 STX   MEM_PTR           POINT TO Z80 ADDRESS AREA
    01CA CC0005       866                 LDD   #5                MOVE OVER 5 BYTES
    01CD FD0003       867                 STD   MEM_LEN
    01D0 8604         868                 LDAA  #MAC_PUT_DATA
    01D2 B70005       869                 STAA  A_SIGNAL          SET UP THE COMMAND
    01D5 38           870                 PULX
    01D6 3C           871                 PSHX                    SAVE THE DCB POINTER
    01D7 C610         872                 LDAB  #DCB_ADD_CODE POINT TO DEVICE ADDRESS
    01D9 3A           873                 ABX
    01DA BD0000       874                 JSR   MMR_DMA_READ
    01DD 8A80         875                 ORAA  #80H              ADDRESS AND S VALUE
    01DF B70000       876                 STAA  NIM_DMA_BLK       SET UP NETWORK ADDRESS
    01E2 BD0000       877                 JSR   MMR_MAC           CALL "BIG MAC"
    01E5 B60006       878                 LDAA  M_SIGNAL          DID WE GET AN IO_COMPLETE
    01E8 8180         879                 CMPA  #IO_COMPLETE
    01EA 2707         880                 BEQ   RD_2              YES WE ARE DONE.
    01EC 38           881                 PULX
    01ED 3C           882                 PSHX
    01EE BD034A       883                 JSR   NIM_RESPONSE
    01F1 203B         884                 BRA   RD_EXIT
    01F3 38           885 RD_2            PULX
    01F4 3C           886                 PSHX                    SAVE THE INDEX
    01F5 C601         887                 LDAB  #DCB_BA_LO
    01F7 3A           888                 ABX
    01F8 8DAC         889                 BSR   GET_Z80_WORD      GET DATA FROM THE Z80
    01FA FD0001       890                 STD   MEM_PTR           POINT TO Z80 BUFFER AREA
    01FD 38           891                 PULX
    01FE 3C           892                 PSHX
    01FF C611         893                 LDAB  #DCB_MAXL_LO      GET MAX_LENGTH POINTER
    0201 3A           894                 ABX
    0202 BD01A6       895                 JSR   GET_Z80_WORD
    0205 DDC6         896                 STD   B_LENGTH          SAVE MAX LENGTH
    0207 38           897                 PULX
    0208 3C           898                 PSHX
    0209 C603         899                 LDAB  #DCB_BUF_LEN_LO BUFFER LENGTH FROM DCB
    020B 3A           900                 ABX
    020C BD01A6       901                 JSR   GET_Z80_WORD
    020F 93C6         902                 SUBD  B_LENGTH          VALID TRANSFER SIZE?
    0211 09           903                 BEQ   RD_1
    0213              904                 PULX
    0214 3C           905                 PSHX
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    0215 86E0         906                 LDAA  #ILLEGAL_ADD  BUFFER NOT SAME SIZE AS BLOCK_LENGTH
    0217 BD0352       907                 JSR   DCB_RESPONSE
    021A 2012         908                 BRA   RD_EXIT
    021C DCC6         909 RD_1            LDD   B_LENGTH      THIS IS TRANSFER LENGTH
    021E FD0003       910                 STD   MEM_LEN       BUFFER LENGTH CONSIDERATIONS.
    0221 8603         911                 LDAA  #MAC_GET_DATA
    0223 B70005       912                 STAA  A_SIGNAL      SET UP THE COMMAND
    0226 BD0000       913                 JSR   MMR_MAC      CALL "BIG MAC"
    0229 38           914                 PULX
    022A 3C           915                 PSHX                COMMAND DONE KEEP STACK CLEAN
    022B BD034A       916                 JSR  NIM_RESPONSE
    022E 38           917 RD_EXIT         PULX
    022F 39           918                 RTS
                      919 *
                      920 *
                      921 ************************************************************
                      922 *  Write data to an external block structured device from *
                      923 *     the Z80                                             *
                      924 *  Enter                                                  *
                      925 *       X       address of DCB in the Z80.                *
                      926 *                                                         *
                      927 ************************************************************
                      928 *
                      929 *
    0230              930 DCB_WT_BLK
    0230 3C           931                 PSHX                SAVE THE INDEX REGISTER
    0231 C605         932                 LDAB  #DCB_SEC_NUM_0 GET OFFSET TO THE SECTOR ADD.
    0233 3A           933                 ABX
    0234 FF0001       934                 STX   MEM_PTR       POINT TO Z80 ADDRESS AREA
    0237 CC0005       935                 LDD   #5            MOVE OVER 5 BYTES
    023A FD0003       936                 STD   MEM_LEN
    023D 8604         937                 LDAA  #MAC_PUT_DATA
    023F B70005       938                 STAA  A_SIGNAL      SET UP THE COMMAND
    0242 38           939                 PULX
    0243 3C           940                 PSHX                SAVE THE DCB POINTER
    0244 C610         941                 LDAB  #DCB_ADD_CODE POINT TO DEVICE ADDRESS
    0246 3A           942                 ABX
    0247 BD0000       943                 JSR   MMR_DMA_READ
    024A 8A80         944                 ORAA  #80H          ADDRESS AND S VALUE
    024C B70000       945                 STAA  NIM_DMA_BLK   SET UP NETWORK ADDRESS
    024F BD0000       946                 JSR   MMR_MAC      CALL "BIG MAC"
    0252 B60006       947                 LDAA  M_SIGNAL      DID WE GET AN IO_COMPLETE
    0255 8180         948                 CMPA  #IO_COMPLETE
    0257 2707         949                 BEQ   WT_2          YES WE ARE DONE.
    0259 38           950                 PULX
    025A 3C           951                 PSHX
    025B BD034A       952                 JSR   NIM_RESPONSE
    025E 203C         953                 BRA   WT_EXIT
    0260 38           954 WT_2            PULX
    0261 3C           955                 PSHX                SAVE THE INDEX
    0262 C601         956                 LDAB  #DCB_BA_LO
    0264 3A           957                 ABX
    0265 BD01A6       958                 JSR   GET_Z80_WORD  GET DATA FROM THE Z80
    0268 FD0001       959                 STD   MEM_PTR       POINT TO Z80 BUFFER AREA
    02   38           960                 PULX
    026  3C           961                 PSHX
    026D C611         962                 LDAB  #DCB_MAXL_LO  GET MAX_LENGTH POINTER
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    026F 3A           963                 ABX
    0270 BD01A6       964                 JSR   GET_Z80_WORD
    0273 DDC6         965                 STD   B_LENGTH       SAVE MAX LENGTH
    0275 38           966                 PULX
    0276 3C           967                 PSHX
    0277 C603         968                 LDAB  #DCB_BUF_LEN_LO BUFFER LENGTH FROM DCB
    0279 3A           969                 ABX
    027A BD01A6       970                 JSR   GET_Z80_WORD
    027D 93C6         971                 SUBD  B_LENGTH       VALID TRANSFER SIZE?
    027F 2709         972                 BEQ   WT_1
    0281 38           973                 PULX
    0282 3C           974                 PSHX
    0283 86E0         975                 LDAA  #ILLEGAL_ADD  BUFFER NOT SAME SIZE AS BLOCK_LENGTH
    0285 BD0352       976                 JSR   DCB_RESPONSE
    0288 2012         977                 BRA   WT_EXIT
    028A DCC6         978 WT_             LDD   B_LENGTH       THIS IS TRANSFER LENGTH
    028C FD0003       979                 STD   MEM_LEN        BUFFER LENGTH CONSIDERATIONS.
    028F 8604         980                 LDAA  #MAC_PUT_DATA
    0291 B70005       981                 STAA  A_SIGNAL       SET UP THE COMMAND
    0294 BD0000       982                 JSR   MMR_MAC       CALL "BIG MAC"
    0297 38           983                 PULX
    0298 3C           984                 PSHX                 COMMAND DONE KEEP STACK CLEAN
    0299 BD034A       985                 JSR  NIM_RESPONSE
    029C 38           986 WT_EXIT         PULX
    029D 39           987                 RTS
                      988 *
                      989 *
                      990 ************************************************************
                      991 *  READ_CHAR  read a character from a device on the       *
                      992 *   network.                                               *
                      993 *                                                          *
                      994 *                                                          *
                      995 *                                                          *
                      996 ************************************************************
                      997 *
                      998 *
    029E              999 DCB_RD_CHR
    029E 3C          1000                 PSHX                 SAVE THE INDEX
    029F C601        1001                 LDAB  #DCB_BA_LO
    02A1 3A          1002                 ABX
    02A2 BD01A6      1003                 JSR   GET_Z80_WORD  GET DATA FROM THE Z80
    02A5 FD0001      1004                 STD   MEM_PTR        POINT TO Z80 BUFFER AREA
    02A8 38          1005                 PULX
    02A9 3C          1006                 PSHX
    02AA C603        1007                 LDAB  #DCB_BUF_LEN_LO BUFFER LENGTH FROM DCB
    02AC 3A          1008                 ABX
    02AD BD01A6      1009                 JSR   GET_Z80_WORD
    02B0 DDC8        1010                 STD   B_SIZE         TRANSFER SIZE
    02B2 38          1011                 PULX
    02B3 3C          1012                 PSHX
    02B4 C611        1013                 LDAB  #DCB_MAXL_LO  GET MAX_LENGTH POINTER
    02B6 3A          1014                 ABX
    02B7 BD01A6      1015                 JSR   GET_Z80_WORD
    02BA DDC6        1016                 STD   B_LENGTH       SAVE MAX LENGTH
    02BC D0C8        1017                 SUBD  B_SIZE         VALID TRANSFER SIZE?
    02BE 2407        1018                 BPL   RD_CHR_1
    02C0 38          1019                 PULX
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    02C1 3C          1020                 PSHX
    02C2 86E1        1021                 LDAA  #CH_BAD_SIZE  BUFFER NOT SAME SIZE AS BLOCK_LENGTH
    02C4 BD0352      1022                 JSR   DCB_RESPONSE
    02C7 2033        1023                 BRA   RD_CHR_EXIT
    02C9 DCC8        1024 RD_CHR_1        LDD   B_SIZE         THIS IS TRANSFER LENGTH
    02CB FD0003      1025                 STD   MEM_LEN        BUFFER LENGTH CONSIDERATIONS.
    02CE 8603        1026                 LDAA  #MAC_GET_DATA
    02D0 B70005      1027                 STAA  A_SIGNAL       SET UP THE COMMAND
    02D3 38          1028                 PULX
    02D4 3C          1029                 PSHX                 SAVE THE DCB POINTER
    02D5 C610        1030                 LDAB  #DCB_ADD_CODE  POINT TO DEVICE ADDRESS
    02D7 3A          1031                 ABX
    02D8 BD0000      1032                 JSR   MMR_DMA_READ
    02DB 8A80        1033                 ORAA  #80H           ADDRESS AND S VALUE
    02DD B70000      1034                 STAA  NIM_DMA_BLK    SET UP NETWORK ADDRESS
    02E0 BD0000      1035                 JSR   MMR_MAC       CALL "BIG MAC"
    02E3 B60006      1036                 LDAA  M_SIGNAL       DID WE GET AN IO_COMPLETE
    02E6 8180        1037                 CMPA  #IO_COMPLETE
    02E8 260D        1038                 BNE   RD_CHR_3       NO WE ARE DONE.
    02EA 38          1039                 PULX
    02EB 3C          1040                 PSHX
    02EC FC0003      1041                 LDD   MEM_LEN        GET ACTUAL TRANSFER SIZE
    02EF 37          1042                 PSHB                 SAVE B
    02F0 C603        1043                 LDAB  #DCB_BUF_LEN_LO  OFFSET TO OUT POINTER
    02F2 3A          1044                 ABX
    02F3 33          1045                 PULB                 RESTORE B
    02F4 BD01B2      1046                 JSR   PUT_Z80_WORD
    02F7 38          1047 RD_CHR_3        PULX
    02F8 3C          1048                 PSHX                 COMMAND DONE KEEP STACK CLEAN
    02F9 BD034A      1049                 JSR  NIM_RESPONSE
    02FC 38          1050 RD_CHR_EXIT     PULX
    02FD 39          1051                 RTS
                     1052 *
                     1053 *
                     1054 ************************************************************
                     1055 *  WRITE CHAR DATA TO THE NETWORK write data from the     *
                     1056 *    Z80 circular buffer to the network.                   *
                     1057 *                                                          *
                     1058 ************************************************************
                     1059 *
    02FE             1060 DCB_WT_CHR
    02FE 3C          1061                 PSHX                 SAVE THE INDEX
    02FF C601        1062                 LDAB  #DCB_BA_LO
    0301 3A          1063                 ABX
    0302 BD01A6      1064                 JSR   GET_Z80_WORD   GET DATA FROM THE Z80
    0305 FD0001      1065                 STD   MEM_PTR        POINT TO Z80 BUFFER AREA
    0308 38          1066                 PULX
    0309 3C          1067                 PSHX
    030A C603        1068                 LDAB  #DCB_BUF_LEN_LO BUFFER LENGTH FROM DCB
    030C 3A          1069                 ABX
    030D BD01A6      1070                 JSR   GET_Z80_WORD
    0310 DDC8        1071                 STD   B_SIZE         TRANSFER SIZE
    0312 38          1072                 PULX
    0313 3C          1073                 PSHX
    0314    1        1074                 LDAB  #DCB_MAXL_LO   GET MAX_LENGTH POINTER
    0316 3A          1075                 ABX
    0317 BD01A6      1076                 JSR   GET_Z80_WORD
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    031A DDC6        1077                 STD   B_LENGTH       SAVE MAX LENGTH
    031C 93C8        1078                 SUBD  B_SIZE         VALID TRANSFER SIZE?
    031E 2A09        1079                 BPL   WT_CHR_1
    0320 38          1080                 PULX
    0321 3C          1081                 PSHX
    0322 86E1        1082                 LDAA  #CH_BAD_SIZE   BUFFER NOT SAME SIZE AS BLOCK_LENGTH
    0324 BD0352      1083                 JSR   DCB_RESPONSE
    0327 201F        1084                 BRA   WT_CHR_EXIT
    0329 DCC8        1085 WT_CHR_1        LDD   B_SIZE         THIS IS TRANSFER LENGTH
    032B FD0003      1086                 STD   MEM_LEN        BUFFER LENGTH CONSIDERATIONS.
    032E 8604        1087                 LDAA  #MAC_PUT_DATA
    0330 B70005      1088                 STAA  A_SIGNAL       SET UP THE COMMAND
    0333 38          1089                 PULX
    0334 3C          1090                 PSHX                 SAVE THE DCB POINTER
    0335 C610        1091                 LDAB  #DCB_ADD_CODE  POINT TO DEVICE ADDRESS
    0337 3A          1092                 ABX
    0338 BD0000      1093                 JSR   MMR_DMA_READ
    033B 8A80        1094                 ORAA  #80H           ADDRESS AND S VALUE
    033D B70000      1095                 STAA  NIM_DMA_BLK    SET UP NETWORK ADDRESS
    0340 BD0000      1096                 JSR   MMR_MAC       CALL "BIG MAC"
    0343 38          1097                 PULX
    0344 3C          1098                 PSHX                 COMMAND DONE KEEP STACK CLEAN
    0345 BD034A      1099                 JSR  NIM_RESPONSE
    0348 38          1100 WT_CHR_EXIT     PULX
    0349 39          1101                 RTS
                     1102 *
                     1103 *
                     1104 *           PROCESS THE RESPONSE AFTER A CALL TO MAC
                     1105 *           TWO CALLS ONE GETS M_SIGNAL RESPONSE AND
                     1106 *           PASSES THE INFORMATION BACK TO THE Z_80
                     1107 *           THE SECOND (DCB_RESPONSE) PROCESSES INTERNAL
                     1108 *           ERRORS GENERATED WITHIN THE APPLICATION.
                     1109 *
    034A             1110 NIM_RESPONSE
    034A 36          1111                 PSHA                 SAVE CURRENT STATE OF AFFECTED REGISTERS
    034B B60006      1112                 LDAA M_SIGNAL        GET RESULT BACK FROM THE PREVIOUS MAC CALL
    034E 8D02        1113                 BSR  DCB_RESPONSE      PROCESS THE RESULT.
    0350 32          1114                 PULA
    0351 39          1115                 RTS
                     1116 *
                     1117 *
    0352 8180        1118 DCB_RESPONSE    CMPA   #IO_COMPLETE   WAS LAST OPERATION SUCESSFUL
    0354 2727        1119                 BEQ    DCB_DONE
    0356 3C          1120                 PSHX                  SAVE THE REGISTERS
    0357 36          1121                 PSHA
    0358 C60E        1122                 LDAB   #DCB_RETRY_LO  GET THE RETRY COUNT FROM THE DCB
    035A 3A          1123                 ABX
    035B BD01A6      1124                 JSR    GET_Z80_WORD
    035E 830000      1125                 SUBD   #0
    0361 2604        1126                 BNE    DCB_R1
    0363 32          1127                 PULA
    0364 38          1128                 PULX
    0365 2016        1129                 BRA    DCB_DONE
    0367 30001       1130 DCB_R1          ADDD   #1              IT WAS FFFF THEN IT WILL SET Z BIT
    036  .604        1131                 BNE    DCB_RETRY_DEC   IT NO UPDATE WITHOUT SUCCESS
    036C 32          1132                 PULA
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

     036E 2010       1134 *                BRA     DCB_EXIT
     0370 830002     1135 DCB_RETRY_DEC    SUBD    #2              SUBTRACT 1 FROM THE RETRY COUNT
     0373 BD01B2     1136                  JSR     PUT_Z80_WORD    STORE IT AWAY
     0376 830000     1137                  SUBD    #0              TEST FOR 0 RESULT
     0379 32         1138                  PULA                    PULLS DONT AFFECT CONDITION CODES
     037A 38         1139                  PULX
     037B 2603       1140                  BNE     DCB_EXIT
     037D BD0000     1141 DCB_DONE         JSR     MMR_DMA_WRITE   WRITE BACK THE STATUS
     0380 39         1142 DCB_EXIT         RTS
                     1143 *
                     1144 *
                     1145 *
     0381 3B         1146 RET_VECTOR:      RTI                             ;unused vector interrupt service routine
                     1147
                     1148                  ORG     0FFF0H
     FFF0 0381       1149                  FDB     RET_VECTOR              ;Serial i/o interupt vector
     FFF2 0381       1150                  FDB     RET_VECTOR              ;Timer overflow interupt vector
     FFF4 0381       1151                  FDB     RET_VECTOR              ;Output compare interupt vector, i. e. timer interupt
     FFF6 0381       1152                  FDB     RET_VECTOR              ;Input capture interupt vector
     FFF8 0381       1153                  FDB     RET_VECTOR              ;IRQ1 - maskable interupt vector
     FFFA 0381       1154                  FDB     RET_VECTOR              ;Software interupt vector
     FFFC 0381       1155                  FDB     RET_VECTOR              ;Non-maskable interupt vector
     FFFE 004A       1156                  FDB     START                   ;Reset interupt vector

Errors=    0
```

LINE#	SYMBOL	TYPE	REFERENCES
271	ACK_Z80	P	433,449,510,534,545
	ASIG_GET_D	A	187
	ASIG_PUT_D	A	188
	ASIG_RESET_D	A	185
	ASIG_STAT_D	A	186
246	A_SIGNAL	C	337,525,770,803,869,912,938,981,1027,1088
220	B_END	A	
221	B_IN	A	
223	B_LENGTH	A	896,902,909,965,971,978,1016,1077
222	B_OUT	A	
224	B_SIZE	A	1010,1017,1024,1071,1078,1085
219	B_START	A	
204	CH_BAD_SIZE	A	1021,1082
575	DCB2	P	572
579	DCB3	P	576
144	DCB_ADD_CODE	A	763,790,872,941,1030,1091
130	DCB_BA_HI	A	
129	DCB_BA_LO	A	887,956,1001,1062
132	DCB_BUF_LEN_HI	A	
131	DCB_BUF_LEN_LO	A	899,968,1007,1043,1068
128	DCB_CMD_STAT	A	
178	DCB_CNT_OFFSET	A	385
216	DCB_COUNT	A	388,401,412
217	DCB_CUR_PNTR	A	393,404,407,410
137	DCB_DEV_NUM	A	
762	DCB_DEV_RESET	P	573
789	DCB_DEV_STATUS	P	577
147	DCB_DEV_TYPE	A	581
1141	DCB_DONE	P	1119,1129
1142	DCB_EXIT	P	1134,1140
114	DCB_IDLE	A	
395	DCB_LOOP	P	413
146	DCB_MAXL_HI	A	
145	DCB_MAXL_LO	A	797,893,962,1013,1074
148	DCB_NODE_TYPE	A	
1130	DCB_R1	P	1126
118	DCB_RD	A	726,740
861	DCB_RD_BLK	P	731
999	DCB_RD_CHR	P	742
116	DCB_RESET	A	571
1118	DCB_RESPONSE	P	907,976,1022,1083,1113
767	DCB_RES_LOOP	P	
1135	DCB_RETRY_DEC	P	1131
143	DCB_RETRY_HI	A	
142	DCB_RETRY_LO	A	1122
133	DCB_SEC_NUM_0	A	863,932
134	DCB_SEC_NUM_1	A	
135	DCB_SEC_NUM_2	A	
136	DCB_SEC_NUM_3	A	
153	DCB_SIZE	A	169,408
115	DCB_STATUS	A	575
117	DCB_WT	A	722,744
930	DCB_WT_BLK	P	724
1060	DCB_WT_CHR	P	749
311	[illegible]AY	P	313
307	GE_NEW_PCB_CMD	P	319,546
288	GET_PCB_CMD	P	428,441,461

LINE#	SYMBOL	TYPE	REFERENCES
818	GET_Z80_WORD	P	507,889,895,901,958,964,970,1003,1009,1015,1064,1070,1076,1124
202	ILLEGAL_ADD	A	906,975
205	ILLEGAL_CMD	A	728,746
337	INIT	P	379
362	INIT_LOOP	P	363
169	INIT_PCB_ADDR	A	340
196	IO_BUSY	A	
195	IO_COMPLETE	A	879,948,1037,1118
197	IO_FAILED	A	
194	IO_IDLE	A	
185	MAC_DEV_RESET	A	524,769
187	MAC_GET_DATA	A	911,1026
186	MAC_GET_STAT	A	802
188	MAC_PUT_DATA	A	868,937,980,1087
243	MEM_LEN	C	801,867,910,936,979,1025,1041,1086
244	MEM_LEN_HI	C	
245	MEM_LEN_LO	C	
240	MEM_PTR	C	865,890,934,959,1004,1065
241	MEM_PTR_HI	C	799
242	MEM_PTR_LO	C	
39	MMR_DMA_READ	E	272,290,317,387,563,583,765,792,820,823,874,943,1032,1093
39	MMR_DMA_WRITE	E	274,729,747,843,846,1141
89	MMR_MAC	E	529,771,804,877,913,946,982,1035,1096
247	M_SIGNAL	C	339,878,947,1036,1112
239	NIM_DMA_BLK	C	527,760,794,876,945,1034,1095
1110	NIM_RESPONSE	P	773,806,883,916,952,985,1049,1099
95	P1DATA	A	
93	P1DDR	A	346
96	P2DATA	A	
94	P2DDR	A	348
101	P3CSR	A	
99	P3DATA	A	
97	P3DDR	A	350
100	P4DATA	A	
98	P4DDR	A	351
478	PCB1	P	
482	PCB2	P	479
486	PCB3	P	483
490	PCB4	P	487
494	PCB5	P	491
171	PCB_IDLE	A	
177	PCB_LENGTH	A	391
215	PCB_LOC	A	289,316,341,384,390,432,448,505,508,509,533,544
175	PCB_RESET	A	486
152	PCB_SIZE	A	169,177
174	PCB_SNA	A	482
422	PCB_SYNC	P	364,430,446,480
172	PCB_SYNC1	A	429,478
173	PCB_SYNC2	A	445
176	PCB_WAIT	A	490,547
561	PROCESS_DCB	P	405
460	PROCESS_PCB	P	382,511,550
837	PUT_Z80_WORD	P	1046,1136
591	R_CHAR_DEV	P	588
726	R_DCB1_BLK	P	723
744	R_DCB1_CHAR	P	741

LINE#	SYMBOL	TYPE	REFERENCES
749	P_DCB2_CHAR	P	745
722	P_DCB_BLOCK	P	589
740	P_DCB_CHAR	P	591
750	P_DCB_CH_XIT	P	747,748
732	P_DCB_EXIT	P	725,730
592	P_DCB_XIT	P	566,570,574,578,590
571	P_NON_IDLE	P	568
495	P_PCB_EXIT	P	463,481,485,489
521	P_RESET	P	488
523	P_RES_LOOP	P	532
504	P_SNA	P	484
544	P_WAIT	P	492
546	P_WAIT_LOOP	P	548
909	RD_1	P	903
885	RD_2	P	880
1024	RD_CHR_1	P	1018
1047	RD_CHR_3	P	1038
1050	RD_CHR_EXIT	P	1023
917	RD_EXIT	P	884,908
1146	RET_VECTOR	P	1149,1150,1151,1152,1153,1154,1155
358	SA	P	357
382	SCANNER	P	402
397	SCAN_1	P	399
102	SCI_RM	A	353
103	SCI_TR_CS	A	355
213	STACK	A	
212	STACKSIZE	A	213
214	STACKSTART	A	377
377	START	P	1156
424	SYNC1_DELAY	P	426
437	SYNC2_DELAY	P	439
218	TEMP_D	A	
435	WAIT_NEW_COMMAN	P	443
978	WT_1	P	972
954	WT_2	P	949
1085	WT_CHR_1	P	1079
1100	WT_CHR_EXIT	P	1084
986	WT_EXIT	P	953,977

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      1 ^6801^
                      3  NAME ^Rev 00 - RPD^
                      4
                      5 De_MMR_MAC MACRO              ;Header Rev. 4
                      6                .GOTO Ede_MMR_MAC
                      7
                      8  Project:      NET, 83-101
                      9
                     10  ************************************************
                     11  *                                              *
                     12  *     MMR_MAC                  WPC/RPD         *
                     13  *                                              *
                     14  ************************************************
                     15
                     16        Rev History
                     17        Rev.  Date         Name      Change
                     18
                     19         0    24jul130p   RPD       Initial Pseudo code and 6801 code
                     20
                     21 Ede_MMR_MAC  MEND
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      23 ****************************************************************
                      24 *                                                              *
                      25 *  MODULE NAME:                                                *
                      26 *                                                              *
                      27 *    MMR_MAC                                                   *
                      28 *                                                              *
                      29 *  INPUTS:                                                     *
                      30 *                                                              *
                      31 *    NET_BYTE_IN (REG_A)                                       *
                      32 *    NIM_DMA_BLK                                               *
                      33 *                                                              *
                      34 *  FUNCTION(S):                                                *
                      35 *                                                              *
                      36 *     1. TO SEQUENCE THE OPERATION OF MAC.                     *
                      37 *                                                              *
                      38 *     2. TO PROCESS NET_BYTE_IN AS NEEDED.                     *
                      39 *                                                              *
                      40 *     3. TO PROCESS NET_BYTE_IN AS DATA IF D1_MODE_WORD IS     *
                      41 *        NOT SET TO CONTROL.                                   *
                      42 *                                                              *
                      43 *     4. TO ORIGINATE AND ORDER THE SENDING OF DATA TO THE APP.*
                      44 *                                                              *
                      45 *     5. TO DECODE, AND POSSIBLY FORWARD, DATA SENT BY THE APP.*
                      46 *                                                              *
                      47 *  OUTPUTS:                                                    *
                      48 *                                                              *
                      49 *     NET_BYTE_OUT (REG_A)                                     *
                      50 *     NIM_DMA_BLK                                              *
                      51 *                                                              *
                      52 *  CALLS:                                                      *
                      53 *                                                              *
                      54 *    MMR_DMA_WRITE                                             *
                      55 *    MMR_DMA_READ                                              *
                      56 *    MMR_TR_REC                                                *
                      57 *    MMR_TR_TRANS                                              *
                      58 *    MMR_TR_TCU                                                *
                      59 *                                                              *
                      60 *  CALLED BY:                                                  *
                      61 *                                                              *
                      62 *    Z80_MASTER_APP.                                           *
                      63 *                                                              *
                      64 *  INVOKED BY:                                                 *
                      65 *                                                              *
                      66 *    IRQ2 DATA INTERRUPT: THIS VECTOR MUST BE INITIALIZED BY   *
                      67 *    THE APPLICATION!                                          *
                      68 *                                                              *
                      69 *  NOTES:                                                      *
                      70 *                                                              *
                      71 *    NONE.                                                     *
                      72 *                                                              *
                      73 ****************************************************************
```

LOCATION OBJECT CODE LINE SOURCE LINE

```
                    75               INCLUDE I_NIM_MR
                     + ************************************************************
                     + *                                                          *
                     + *                      HIERARCHY  CHART                    *
                     + *                           OF                             *
                     + *                  MEDIUM  ACCESS  CONTROLLER              *
                     + *                           (MAC)                          *
                     + *                           MASTER                         *
                     + *                                                          *
                     + *                          (MAC)                           *
                     + *                            |                             *
                     + *                            |                             *
                     + *          +-----------------+-------------------------+   *
                     + *        (TR)                (DMA)                    (ACM)  *
                     + *         |                    |                       |    *
                     + *     +---+---+-------+        |                       |    *
                     + *  (TRANS)  (REC)  (TCU)       |                       |    *
                     + *                              |                       |    *
                     + *                 +------------+-----------+           |    *
                     + *                 |                        |           |    *
                     + *              (WRITE)                   (READ)        |    *
                     + *                                                      |    *
                     + *                                                      |    *
                     + *                                                      |    *
                     + *                                          +---------+-----+-----+    *
                     + *                                          |         |     |     |    *
                     + *                                     (EVENT_PROC) (SEQ) (TMR)(RESP)*
                     + *                                                          *
                     + ************************************************************
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                    + ****************************************************************
                    + *              INTERFACE MODULE DESCRIPTION                    *
                    + *              ----------------------------                    *
                    + *  NAME:                                                       *
                    + *   1_NIM_MR                                                   *
                    + *                                                              *
                    + *  FUNCTION:                                                   *
                    + *      TO DEFINE THE INTERFACE BETWEEN THE 6801_MASTER_MAC AND *
                    + *      6801_MASTER_APP.                                        *
                    + *                                                              *
                    + *  DESCRIPTION:                                                *
                    + *      A BLOCK OF MEMORY WILL BE SHARED BY THE MAC AND APP,    *
                    + *      WHEREIN DATA AND CONTROL SIGNALS WILL BE PASSED BACK    *
                    + *      AND FORTH BETWEEN THE TWO. A DIAGRAM OF THIS BLOCK      *
                    + *      (REFERRED TO AS NIM_BLOCK) FOLLOWS:                     *
                    + *                                                              *
                    + *           NIM_DMA_BLK                                        *
                    + *           +-+----+------+                                    *
                    + *           |S|////|DEV_ID|  A(W/R), M(R/RESET);               *
                    + *           +-+----+------+                                    *
                    + *           | MEM_PTR_HI  |  A(W), M(R  );                     *
                    + *           +-------------+                                    *
                    + *           | MEM_PTR_LO  |    "     "                         *
                    + *           +-------------+                                    *
                    + *           | MEM_LEN_HI  |  A(W), M(R);                       *
                    + *           +-------------+                                    *
                    + *           | MEM_LEN_LO  |  A(W), M(R);                       *
                    + *           +-------------+                                    *
                    + *           | A_SIGNAL    |  A(W), M(R);                       *
                    + *           +-------------+                                    *
                    + *           | M_SIGNAL    |  A(R/RESET), M(W);                 *
                    + *           +-------------+                                    *
                    + *                                                              *
                    + *  NOTES:                                                      *
                    + *     WHEN THE APP CALLS THE MAC THE FOLLOWING MUST TAKE PLACE: *
                    + *                                                              *
                    + *       1. APP SETS A_SIGNAL TO THE DESIRED FUNCTION.          *
                    + *       2. APP SETS M_SIG TO IO_IDLE. (0)                      *
                    + *       3. APP SETS MEM_PTR IF APPLICABLE.                     *
                    + *       4. APP SETS MEM_LEN IF APPLICABLE.                     *
                    + *       5. LASTLY, THE APP SETS DEVICE ID AND                  *
                    + *          APP SETS CNFG_WORD.BIT_7.                           *
                    + *       6. APP CALLS MAC.                                      *
                    + *       7. MAC WILL EXECUTE ORDERS OF MAC.                     *
                    + *       8. WHEN MAC IS COMPLETE, IT WILL WRITE TO NIM.M_SIGNAL *
                    + *          AS WELL AS CLR THE CNFG_WORD.BIT_7.                 *
                    + *       9. MAC WILL DEPOSIT THE TRUE LENGTH OF DATA WRITTEN    *
                    + *          TO Z80_MEM IN MEM_LEN. THUS, APP SHOULD HAVE SAVED  *
                    + *          ORIGINAL MEM_LEN BEFORE CALL TO MAC.                *
                    + *      10. THE APP WILL READ THE NIM AND CLEAR A_SIG, M_SIG,   *
                    + *          MEM_PTR, MEM_LEN AND CNFG_WORD. (BIT 7=0 ALREADY)   *
                    + *                                                              *
                    + ****************************************************************
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                    + ****************************************************************
                    + *                                                              *
                    + *  DATA ELEMENT DEFINITIONS:                                   *
                    + *                                                              *
                    + *     CNFG_WORD :                                              *
                    + *     ----------                                               *
                    + *       BIT.7:  CLR = APP SHOULD NOT CALL MAC                  *
                    + *               SET = APP SHOULD CALL MAC                      *
                    + *                                                              *
                    + *     A_SIG:                                                   *
                    + *     ------                                                   *
                    + *        0- NO SIGNAL (IDLE).                                  *
                    + *        1- RESET DEVICE D                                     *
                    + *        2- GET STATUS FROM DEVICE D.                          *
                    + *        3- GET DATA FROM DEVICE D.                            *
                    + *        4- PUT DEVICE TO DEVICE TO D.                         *
                    + *                                                              *
(0001)              + ASIG_RESET_D    EQU             001H
(0002)              + ASIG_STAT_D     EQU             002H
(0003)              + ASIG_GET_D      EQU             003H
(0004)              + ASIG_PUT_D      EQU             004H
(0004)              + ASIG_MAX_CODE   EQU             ASIG_PUT_D
                    + *     MEM_POINTER                                              *
                    + *     -----------                                              *
                    + *                                                              *
                    + *        (CAN ACCESS UP TO 64K OF Z80 MEMORY)                  *
                    + *                                                              *
                    + *     A_LEN:                                                   *
                    + *     ------                                                   *
                    + *                                                              *
                    + *        (CAN RANGE UP TO 64K)                                 *
                    + *                                                              *
                    + *     M_SIG:                                                   *
                    + *     ------                                                   *
                    + *        0- IO_IDLE                                            *
                    + *        1- IO_COMPLETE.                                       *
                    + *        2- IO_BUSY                                            *
                    + *        3- IO_FAILED                                          *
                    + *                                                              *
(0000)              + MSIG_IDLE       EQU             000H
(0080)              + MSIG_COMPLETE   EQU             080H
(0002)              + MSIG_TIMEOUT    EQU             002H
(0003)              + MSIG_FAILED     EQU             003H
                    + *                                                              *
                    + *   NOTES:                                                     *
                    + *     * 1. A:= APPLICATION SIDE OF NODE.                       *
                    + *       2. M:= MAC SIDE OF NODE.                               *
                    + *       3. x(W):= THIS ELEMENT HAS WRITE ACCESS TO THIS BYTE.  *
                    + *       4. x(R/RESET):= THIS ELEMENT HAS READ ACCESS AS WELL AS *
                    + *          RESET ACCESS( WRITE ZERO ONLY) TO THIS BYTE.        *
                    + *                                                              *
                    + ****************************************************************
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                    + ************************************************************
                    + *                                                          *
                    + *  NOTES TO INSTALLER OF THIS MAC/APP:                     *
                    + *                                                          *
                    + *   1. THE APP IS RESPONSIBLE FOR INITIALIZING ALL OF RAM. *
                    + *                                                          *
                    + *   2. THE APP MUST INITIALIZE THE CONTROL AND STATUS REG AS*
                    + *      WELL AS THE BAUD RATE AND MODE REGISTER.            *
                    + *                                                          *
                    + *   3. THE D1_MODE_WORD MUST BE SET TO ZERO AT PWR UP BY THE*
                    + *      APP.                                                *
                    + *                                                          *
                    + *   4. THE NIM_BLOCK WILL END AT ADDR 255 DECIMAL.         *
                    + *                                                          *
                    + ************************************************************
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      77 ;
                      78 ; local equates
                      79 ;
                      80 ; general equates
                      81
(000F)                82 NODE_MASK       EQU     00FH                    ;node address mask
(00AA)                83 SOME_ERROR      EQU     0AAH                    ;some error code (to be defined)
                      84
                      85 ;
                      86 ;NIM block equates
                      87 ;
(00F9)                88 CNFG_WORD       EQU     0F9H                    ;NIM.CNFG_WORD
(00FA)                89 MEM_PTR         EQU     0FAH
(00FA)                90 MEM_PTR_HI      EQU     MEM_PTR+0               ;NIM.MEM_PTR_HI
(00FB)                91 MEM_PTR_LO      EQU     MEM_PTR+1               ;NIM.MEM_PTR_LO
(00FC)                92 MEM_LEN         EQU     0FCH
(00FC)                93 MEM_LEN_HI      EQU     MEM_LEN+0               ;NIM.MEM_LEN_HI
(00FD)                94 MEM_LEN_LO      EQU     MEM_LEN+1               ;NIM.MEM_LEN_LO
(00FE)                95 A_SIG           EQU     0FEH                    ;NIM.A_SIGNAL
(00FF)                96 M_SIG           EQU     0FFH                    ;NIM.M_SIGNAL
                      97
                      98 ;
                      99 ; message scenarios equates
                     100 ;
(0000)               101 MN_RESET        EQU     000H*16                 ;command.control (reset)
(0010)               102 MN_STATUS       EQU     001H*16                 ;command.control (status)
(0020)               103 MN_ACK          EQU     002H*16                 ;command.control (ack)
(0030)               104 MN_CLR          EQU     003H*16                 ;command.control (clr)
(0040)               105 MN_RECEIVE      EQU     004H*16                 ;command.control (receive)
(0050)               106 MN_CANCEL       EQU     005H*16                 ;command.control (cancel)
(0060)               107 MN_SEND         EQU     006H*16                 ;command.data (send)
(0070)               108 MN_NACK         EQU     007H*16                 ;command.control (nack)
(00D0)               109 MN_READY        EQU     00DH*16                 ;command.control (ready)
                     110
(0008)               111 NM_STATUS       EQU     008H                    ;response.control (status)
(0009)               112 NM_ACK          EQU     009H                    ;response.control (ack)
(000A)               113 NM_CANCEL       EQU     00AH                    ;response.control (cancel)
(000B)               114 NM_SEND         EQU     00BH                    ;response.data (send)
(000C)               115 NM_NACK         EQU     00CH                    ;response.control (nack)
                     116
(0080)               117 ERR_CODES       EQU     80H
                     118
(0081)               119 READY_TMOUT     EQU     ERR_CODES+1
(0082)               120 RTS_COM_ERR     EQU     ERR_CODES+2
(0083)               121 SEND_TMOUT      EQU     ERR_CODES+3
(0084)               122 SEND_COM_ERR    EQU     ERR_CODES+4
(0085)               123 SEND_DATA_BR    EQU     ERR_CODES+5
(0086)               124 READY_NACK      EQU     ERR_CODES+6
(0087)               125 READY_B_R       EQU     ERR_CODES+7
(0088)               126 SEND_DATA_NAK   EQU     ERR_CODES+8
                     127
(0089)               128 RECV_TMOUT      EQU     ERR_CODES+9
(008A)               129 RECV_BR         EQU     ERR_CODES+10
(008B)               130 RECV_DATA_BR    EQU     ERR_CODES+11
(008C)               131 RECV_NACK       EQU     ERR_CODES+12
(008D)               132 CLR_TMOUT       EQU     ERR_CODES+13
(008E)               133 CLR_ERR         EQU     ERR_CODES+14
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

             (008F)   134 CLR_BR          EQU     ERR_CODES+15
             (0090)   135 DATA_IN_ERROR   EQU     ERR_CODES+16
             (0091)   136 DATAIN_TIMEOUT  EQU     ERR_CODES+17
                      137
             (0092)   138 BAD_ASIG        EQU     ERR_CODES+18
                      139
             (0093)   140 STAT_TMOUT      EQU     ERR_CODES+19
             (0094)   141 STAT_BR         EQU     ERR_CODES+20
             (0095)   142 STATUS_BR       EQU     ERR_CODES+21
                      143                 GLB     TCU_ERR1
             (0096)   144 TCU_ERR1        EQU     ERR_CODES+22
                      145                 GLB     TCU_ERR2
             (0097)   146 TCU_ERR2        EQU     ERR_CODES+23
                      147                 GLB     TRANS_TO1
             (0098)   148 TRANS_TO1       EQU     ERR_CODES+24          ;SET IN TRANS
                      149                 GLB     TRANS_TO2
             (0099)   150 TRANS_TO2       EQU     ERR_CODES+25
                      151                 GLB     HAVE_ORFE
             (009A)   152 HAVE_ORFE       EQU     ERR_CODES+26          ;SET IN TRANS
                      153                 GLB     TIME_OUT_ERR
             (009B)   154 TIME_OUT_ERR    EQU     ERR_CODES+27
             (009C)   155 BAD_RDRF        EQU     ERR_CODES+28
             (009D)   156 BAD_TDRE        EQU     ERR_CODES+29
             (009E)   157 BAD_ORFE        EQU     ERR_CODES+30
                      158
                      159 *
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     161 ;
                     162 ;  pseudocode for MMR_MAC
                     163 ;
                     164 ;1 begin
                     165 ;1   EXPECTED_NODE = NIM.CNFG_WORD .AND. node_mask
                     166 ;1   (init other variables)
                     167 ;1   case NIM.A_SIG of
                     168 ;2     idle        : begin
                     169 ;3                     nop
                     170 ;2                   end
                     171 ;2     reset       : begin
                     172 ;3                     reset_device
                     173 ;2                   end
                     174 ;2     status      : begin
                     175 ;3                     get_status_from_device
                     176 ;2                   end
                     177 ;2     get         : begin
                     178 ;3                     get_data_from_device
                     179 ;2                   end
                     180 ;2     put         : begin
                     181 ;3                     put_data_to_device
                     182 ;2                   end
                     183 ;2     all_others  : begin
                     184 ;3                     NIM.M_SIG = some_error_indicator
                     185 ;2                   end
                     186 ;1   endcase
                     187 ;1 end
                     188 ;
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     190                PROG
                     191
                     192                EXT     MMR_TR_TRANS,MMR_TR_TCU
                     193                EXT     MMR_DMA_READ,MMR_DMA_WRITE
                     194                EXT     MMR_TR_REC,VERIFY_CMND,ISOLATE_CMND
                     195                EXT     EXPECTED_NODE
                     196
                     197                GLB     MMR_MAC
0000                 198 MMR_MAC:
                     199 ;
                     200 ; MAKE SURE INITIAL CONDITIONS OF UART ARE ACCEPTABLE
                     201 ;
                     202
0000 9611            203                LDAA    011H,D                  ; READ STATUS REGISTER
0002 9612            204                LDAA    012H,D                  ; READ DATA-IN REGISTER
                     205
0004 8600            206                LDAA    #00000000B              ; SHUT OFF RD/WR ENABLES
0006 9711            207                STAA    011H,D                  ;
                     208
0008 8614            209                LDAA    #20                     ; SOME DELAY
000A                 210 LOOP:
000A 4A              211                DECA
000B 26FD            212                BNE     LOOP
                     213
000D 860A            214                LDAA    #00001010B              ; ENABLE RD/WR
000F 9711            215                STAA    011H,D                  ;
                     216
0011 9611            217                LDAA    011H,D                  ; READ STATUS REG
0013 9612            218                LDAA    012H,D                  ; READ DATA-IN REG
                     219
0015 BD01CD          220                JSR     LINE_TURN_DELAY         ; DELAY > 1 BYTE TIME
                     221
0018 9611            222                LDAA    011H,D                  ; CHECK STATUS BITS
001A D612            223                LDAB    012H,D                  ; DUMMY READ
                     224
001C 9611            225                LDAA    011H,D
001E D612            226                LDAB    012H,D
                     227
0020 8520            228                BITA    #00100000B              ; CHECK TDRE
0022 2605            229                BNE     TDRE_OK                 ; TDRE OKAY
                     230
0024 869D            231                LDAA    #BAD_TDRE               ; OOPS ...
0026 97FF            232                STAA    M_SIG,D                 ;
                     233
0028 39              234                RTS                             ; ALL DONE
                     235
0029                 236 TDRE_OK:
0029 8580            237                BITA    #10000000B              ; CHECK RDRF
002B 2705            238                BEQ     RDRF_OK                 ;
                     239
002D 869C            240                LDAA    #BAD_RDRF               ;
002F 97FF            241                STAA    M_SIG,D                 ;
                     242
0031 39              243                RTS                             ;
                     244
0032                 245 RDRF_OK:
0032 8540            246                BITA    #01000000B              ; CHECK ORFE
```

```
LOCATION OBJECT CODE LINE   SOURCE LINE

    0034 2705          247                 BEQ     ORFE_OK                 ;
                       248
    0036 869E          249                 LDAA    #BAD_ORFE               ;
    0038 97FF          250                 STAA    M_SIG,D                 ;
                       251
    003A 39            252                 RTS                             ;
                       253
    003B               254 ORFE_OK:
                       255                                                 ;1   EXPECTED_NODE = NIM.CNFG_WORD .AND. node_mask
    003B D6F9          256                 LDAB    CNFG_WORD,D             ;      get the value in CNFG_WORD
    003D C40F          257                 ANDB    #NODE_MASK              ;      mask in the node address
    003F D700          258                 STAB    EXPECTED_NODE,D         ;      put the node address into EXPECTED_NODE
                       259                                                 ;1   (init other variables)
                       260
                       261                                                 ;1   case NIM.A_SIG of
    0041 D6FE          262                 LDAB    A_SIG,D                 ;      get the value in A_SIG
    0043 C104          263                 CMPB    #ASIG_MAX_CODE          ; INVALID COMMAND ?
    0045 2209          264                 BHI     BAD_ASIG_COMMAND        ; NOPE
    0047 58            265                 LSLB                            ;      A_SIG value = A_SIG value * 2
    0048 CE0056        266                 LDX     #CASE_TABLE             ;      point to the case address table
    004B 3A            267                 ABX                             ;      index into the table
    004C EE00          268                 LDX     0,X                     ;      get the address of the desired case
    004E 6E00          269                 JMP     0,X                     ;      execute the desired case
                       270
    0050               271 BAD_ASIG_COMMAND:                               ;2     all_others : begin
                       272                                                 ;3                    NIM.M_SIG = some_error_indicator
    0050 C692          273                 LDAB    #BAD_ASIG               ;                     get the error constant
    0052 D7FF          274                 STAB    M_SIG,D
    0054 2021          275                 BRA     ENDCASE                 ;2                  end
                       276 ;
                       277 ;  case address table
                       278 ;
    0056               279 CASE_TABLE:
    0056 0060          280                 FDB     IDLE                    ;address of idle case
    0058 0063          281                 FDB     RESET                   ;address of reset case
    005A 0068          282                 FDB     STATUS                  ;address of status case
    005C 006D          283                 FDB     GET                     ;address of get case
    005E 0072          284                 FDB     PUT                     ;address of put case
                       285 ;
    0060               286 IDLE:                                           ;2     idle       : begin
    0060 01            287                 NOP                             ;3                    nop
    0061 2014          288                 BRA     ENDCASE                 ;2                  end
    0063               289 RESET:                                          ;2     reset      : begin
    0063 BD0078        290                 JSR     RESET_DEVICE            ;3                    reset_device
    0066 200F          291                 BRA     ENDCASE                 ;2                  end
    0068               292 STATUS:                                         ;2     status     : begin
    0068 BD008E        293                 JSR     GET_STATUS              ;3                    get_status_from_device
    006B 200A          294                 BRA     ENDCASE                 ;2                  end
    006D               295 GET:                                            ;2     get        : begin
    006D BD00C7        296                 JSR     GET_DATA                ;3                    get_data_from_device
    0070 2005          297                 BRA     ENDCASE                 ;2                  end
    0072               298 PUT:                                            ;2     put        : begin
    0072 BD01D4        299                 JSR     PUT_DATA                ;3                    put_data_to_device
    0075 2002          300                 BRA     ENDCASE                 ;2                  end
    0077               301 ENDCASE:                                        ;1   endcase
    0077 39            302                 RTS
                       303
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    0078              304 SKIP
                      305 ;
                      306 ;  pseudocode for RESET_DEVICE
                      307 ;
                      308 ;1 begin
                      309 ;1   nop
                      310 ;1 end
                      311 ;
    0078              312 RESET_DEVICE:
    0078 8600         313                 LDAA    #MN_RESET               ; SOFT-RESET COMMAND
    007A 9A00         314                 ORAA    EXPECTED_NODE,D         ; NODE ADDR
    007C BD0000       315                 JSR     MMR_TR_TRANS            ; OUTPUT IT
    007F 240C         316                 BCC     RS_EXIT
                      317
    0081 BD0000       318                 JSR     MMR_TR_TCU              ;
    0084 2407         319                 BCC     RS_EXIT
                      320
    0086 BD01CD       321                 JSR     LINE_TURN_DELAY
                      322
    0089 8680         323                 LDAA    #MSIG_COMPLETE          ; TELL APP WE ARE DONE
    008B 97FF         324                 STAA    M_SIG,D                 ;
                      325
    008D              326 RS_EXIT
    008D 39           327                 RTS
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     329 ;
                     330 ;  pseudocode for GET_STATUS
                     331 ;
                     332 ;1 begin
                     333 ;1   nop
                     334 ;1 end
                     335 ;
                     336
          (0004)     337 STAT_PKT_SIZE  EQU     4
                     338
    008E             339 GET_STATUS:
    008E 8610        340                LDAA    #MN_STATUS              ; REQUEST STATUS FROM DEVICE
    0090 9A00        341                ORAA    EXPECTED_NODE,D         ; NODE ADDR
    0092 BD0000      342                JSR     MMR_TR_TRANS            ; OUT TO NODE
    0095 241A        343                BCC     E_W_R_ERR5
                     344
    0097 BD0000      345                JSR     MMR_TR_TCU              ;
    009A 2415        346                BCC     E_W_R_ERR5
                     347
    009C BD0000      348                JSR     MMR_TR_REC
    009F 2410        349                BCC     E_W_R_ERR5              ;if no net byte available
                     350
    00A1 BD0000      351                JSR     VERIFY_CMND
    00A4 2506        352                BCS     R_N_R5                  ;if node = expected node
                     353
    00A6 C694        354                LDAB    #STAT_BR                ;  NIM.M_SIG = rts_com_err
    00A8 D7FF        355                STAB    M_SIG,D
                     356
    00AA 2005        357                BRA     E_W_R_ERR5
                     358
    00AC             359 R_N_R5:
    00AC BD0000      360                JSR     ISOLATE_CMND            ;put command into lower nibble
    00AF 2003        361                BRA     E_W_R_OK5
                     362
    00B1             363 E_W_R_ERR5:
    00B1 7E01C9      364                JMP     END_GET_STATUS
                     365
    00B4             366 E_W_R_OK5:                                     ;1   case ready_response of
    00B4 8108        367                CMPA    #NM_STATUS              ; RIGHT COMMAND BACK ?
    00B6 2707        368                BEQ     GOT_NM_STATUS           ; YES ...
                     369
    00B8 C695        370                LDAB    #STATUS_BR              ; ERROR
    00BA D7FF        371                STAB    M_SIG,D                 ;
                     372
    00BC 7E01C9      373                JMP     END_GET_STATUS
                     374
    00BF             375 GOT_NM_STATUS:
    00BF CE0004      376                LDX     #STAT_PKT_SIZE          ; SIZE OF STATUS PACKET
    00C2 DF00        377                STX     BYTE_COUNT,D            ;
                     378
    00C4 7E014C      379                JMP     DO_REC_STATUS           ; USE RECEIVE_DATA LOGIC
                     380                                                ;  TO GET STATUS IN
                     381
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     383 ;
                     384 ;  pseudocode for GET_DATA
                     385 ;
                     386 ;1 begin
                     387 ;1   nop
                     388 ;1 end
                     389 ;
    00C7             390 GET_DATA:
    00C7 8640        391                 LDAA    #MN_RECEIVE
    00C9 9A00        392                 ORAA    EXPECTED_NODE,D
    00CB BD0000      393                 JSR     MMR_TR_TRANS
    00CE 241A        394                 BCC     E_W_R_ERR3
                     395
    00D0 BD0000      396                 JSR     MMR_TR_TCU
    00D3 2415        397                 BCC     E_W_R_ERR3
                     398
    00D5 BD0000      399                 JSR     MMR_TR_REC
    00D8 2410        400                 BCC     E_W_R_ERR3          ;if no net byte available
                     401
    00DA BD0000      402                 JSR     VERIFY_CMND
    00DD 2506        403                 BCS     R_N_R3              ;if node = expected node
                     404
    00DF C68A        405                 LDAB    #RECV_BR            ;  NIM.M_SIG = rts_com_err
    00E1 D7FF        406                 STAB    M_SIG,D
                     407
    00E3 2005        408                 BRA     E_W_R_ERR3
                     409
    00E5             410 R_N_R3:
    00E5 BD0000      411                 JSR     ISOLATE_CMND        ;put command into lower nibble
    00E8 2003        412                 BRA     E_W_R_OK3
                     413
    00EA             414 E_W_R_ERR3:
    00EA 7E01C9      415                 JMP     END_GET_DATA
                     416
    00ED             417 E_W_R_OK3:                                  ;1   case ready_response of
    00ED 8109        418                 CMPA    #NM_ACK
    00EF 2712        419                 BEQ     RECEIVE_ACK
    00F1 810C        420                 CMPA    #NM_NACK
    00F3 2707        421                 BEQ     RECEIVE_NACK
                     422
    00F5 C68B        423                 LDAB    #RECV_DATA_BR
    00F7 D7FF        424                 STAB    M_SIG,D
                     425
    00F9 7E01C9      426                 JMP     END_GET_DATA
                     427
    00FC             428 RECEIVE_NACK:
    00FC C68C        429                 LDAB    #RECV_NACK
    00FE D7FF        430                 STAB    M_SIG,D
                     431
    0100 7E01C9      432                 JMP     END_GET_DATA
                     433
    0103             434 RECEIVE_ACK:
    0103 8630        435                 LDAA    #MN_CLR
    0105 9A00        436                 ORAA    EXPECTED_NODE,D
    0107 BD0000      437                 JSR     MMR_TR_TRANS
    010A             438                 BCC     E_W_R_ERR4
                     439
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    010C BD0000        440                 JSR     MMR_TR_TCU
    010F 2415          441                 BCC     E_W_R_ERR4
                       442
    0111 BD0000        443                 JSR     MMR_TR_REC
    0114 2410          444                 BCC     E_W_R_ERR4              ;if no net byte available
                       445
    0116 BD0000        446                 JSR     VERIFY_CMND
    0119 2506          447                 BCS     R_N_R4                  ;if node = expected node
                       448
    011B C68E          449                 LDAB    #CLR_ERR            ;  NIM.M_SIG = rts_com_err
    011D D7FF          450                 STAB    M_SIG,D
                       451
    011F 2005          452                 BRA     E_W_R_ERR4
                       453
    0121               454 R_N_R4:
    0121 BD0000        455                 JSR     ISOLATE_CMND            ;put command into lower nibble
    0124 2003          456                 BRA     E_W_R_OK4
                       457
    0126               458 E_W_R_ERR4:
    0126 7E01C9        459                 JMP     END_GET_DATA
                       460
    0129               461 E_W_R_OK4:                                      ;1    case ready_response of
    0129 810B          462                 CMPA    #NM_SEND
    012B 2707          463                 BEQ     CLR_SEND
                       464
    012D 868F          465                 LDAA    #CLR_BR
    012F 97FF          466                 STAA    M_SIG,D
                       467
    0131 7E01C9        468                 JMP     END_GET_DATA
                       469
    0134               470 CLR_SEND:
    0134 BD0000        471                 JSR     MMR_TR_REC
    0137 2503          472                 BCS     N_E_G_D1
                       473
    0139 7E01C9        474                 JMP     END_GET_DATA
                       475
    013C               476 N_E_G_D1:
    013C 97FC          477                 STAA    MEM_LEN+0,D
    013E 9700          478                 STAA    BYTE_COUNT+0,D
                       479
    0140 BD0000        480                 JSR     MMR_TR_REC
    0143 2503          481                 BCS     N_E_G_D2
                       482
    0145 7E01C9        483                 JMP     END_GET_DATA
                       484
    0148               485 N_E_G_D2:
    0148 97FD          486                 STAA    MEM_LEN+1,D
    014A 9701          487                 STAA    BYTE_COUNT+1,D
                       488
    014C               489 DO_REC_STATUS:
    014C 7F0002        490                 CLR     CHK_SUM,D
                       491
    014F               492 DATA_IN_LOOP:
    014F BD0000        493                 JSR     MMR_TR_REC
    0152 2-            494                 BCC     END_GET_DATA
                       495
    0154 DEFA          496                 LDX     MEM_PTR_H1,D
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     497
                     498 ;***************JSR     MMR_DMA_WRITE**********************************
                     499
                     500                 EXT     STACK_TEMP
                     501 ;
                     502 ;Standard M6801 I/O register definitions
                     503 ;
                     504 ;
          (0000)     505 P1DDR           EQU           0  PORT 1 DATA DIRECTION REGISTER
          (0001)     506 P2DDR           EQU           1  PORT 2 DATA DIRECTION REGISTER
          (0002)     507 P1DATA          EQU           2  PORT 1 DATA REGISTER
          (0003)     508 P2DATA          EQU           3  PORT 2 DATA REGISTER
          (0004)     509 P3DDR           EQU           4  PORT 3 DATA DIRECTION REGISTER
          (0005)     510 P4DDR           EQU           5  PORT 4 DATA DIRECTION REGISTER
          (0006)     511 P3DATA          EQU           6  PORT 3 DATA REGISTER
          (0007)     512 P4DATA          EQU           7  PORT 4 DATA REGISTER
          (000F)     513 P3CSR           EQU          15  PORT 3 CONTROL AND STATUS
                     514 ;
                     515 ;
                     516 ;WRITE DATA BYTE
                     517 ;
                     518 ;       This subroutine will write a single byte of
                     519 ;       data to the Z80's memory.
                     520 ;
                     521 ;Enter with the following arguments
                     522 ;       X       Address in the Z80's memory
                     523 ;       A       Data byte.
                     524 ;Return
                     525 ;       Contents of A is lost.
                     526 ;
                     527 ******************************************************
                     528
0156 36              529                 PSHA            SAVE DATA BYTE OUT
                     530
0157 7F000F          531                 CLR   P3CSR     SET UP P3 CONTROL PORT
015A 9706            532                 STAA  P3DATA
015C 3C              533                 PSHX
015D 32              534                 PULA            GET HIGH ADDRESS BYTE
015E 9702            535                 STAA  P1DATA    LOAD HIGH ADDRESS REGISTER
0160 32              536                 PULA            GET LOW ADDRESS BYTE
0161 9707            537                 STAA  P4DATA
0163 861A            538                 LDAA  #01AH     SET UP CONTROL BITS FOR WRITE
0165 9703            539                 STAA  P2DATA
0167 CC15FF          540                 LDD   #015FFH   CONSTANT TO SET PORT OUTPUTS
016A CE0010          541                 LDX   #00010H   CONSTANT TO SET PORT INPUTS
                     542
016D BF0000          543                 STS   STACK_TEMP SAVE THE STACK REGISTER
0170 8E0000          544                 LDS   #00000H    USE STACK AS A WORD WIDE REGISTER
                     545 *
                     546 *           START OF THE ACTUAL TRANSFER
                     547 *
0173 9701            548                 STAA  P2DDR     SET UP FOR OUTPUTS
0175 D704            549                 STAB  P3DDR     PORT 3/4 ENABLED OUT
0177 D705            550                 STAB  P4DDR
0179 D700            551                 STAB  P1DDR     SET THE DMA
017B 9606            552                 LDAA  P3DATA    DUMMY R
017D 9F04            553                 STS   P3DDR
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    017F DF00          554                 STX   P1DDR     REMOVE ALL REGISTERS FROM THE BUS
                       555 *
                       556 *
    0181 BE0000        557                 LDS   STACK_TEMP RESTORE THE STACK
                       558
    0184 32            559                 PULA
                       560
                       561 ; ***************************************************************
                       562
    0185 9802          563                 EORA  CHK_SUM,D
    0187 9702          564                 STAA  CHK_SUM,D
                       565
    0189 DEFA          566                 LDX   MEM_PTR_HI,D
    018B 08            567                 INX
    018C DFFA          568                 STX   MEM_PTR_HI,D
                       569
    018E DE00          570                 LDX   BYTE_COUNT,D
    0190 09            571                 DEX
    0191 DF00          572                 STX   BYTE_COUNT,D
                       573
    0193 26BA          574                 BNE   DATA_IN_LOOP
                       575
    0195 BD0000        576                 JSR   MMR_TR_REC
    0198 242F          577                 BCC   END_GET_DATA
    019A 9102          578                 CMPA  CHK_SUM,D
    019C 2715          579                 BEQ   DATA_IN_CS_OK
                       580
    019E C690          581                 LDAB  #DATA_IN_ERROR
    01A0 D7FF          582                 STAB  M_SIG,D
                       583
    01A2 BD01CD        584                 JSR   LINE_TURN_DELAY
                       585
    01A5 8670          586                 LDAA  #MN_NACK
    01A7 9A00          587                 ORAA  EXPECTED_NODE,D
    01A9 BD0000        588                 JSR   MMR_TR_TRANS
    01AC 241B          589                 BCC   END_GET_DATA
    01AE BD0000        590                 JSR   MMR_TR_TCU
    01B1 2016          591                 BRA   END_GET_DATA
                       592
    01B3               593 DATA_IN_CS_OK:
    01B3 C680          594                 LDAB  #MSIG_COMPLETE
    01B5 D7FF          595                 STAB  M_SIG,D
                       596
    01B7 BD01CD        597                 JSR   LINE_TURN_DELAY
                       598
    01BA 8620          599                 LDAA  #MN_ACK
    01BC BA0000        600                 ORAA  EXPECTED_NODE
    01BF BD0000        601                 JSR   MMR_TR_TRANS
    01C2 2405          602                 BCC   END_GET_DATA
    01C4 BD0000        603                 JSR   MMR_TR_TCU
                       604
    01C7 2000          605                 BRA   END_GET_DATA
                       606
    01C9               607 END_GET_DATA:
    01C9               608 END_GET_STATUS:
    01C9 B    CD       609                 JSR   LINE_TURN_DELAY
                       610
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    01CC 39           611                 RTS
                      612
    01CD              613 LINE_TURN_DELAY:
    01CD CE0021       614                 LDX     #200/6              ; # USEC TO LET DEVICE
                      615                                             ;  WAKEUP
                      616
    01D0              617 L_T_D_LOOP:
    01D0 09           618                 DEX
    01D1 26FD         619                 BNE     L_T_D_LOOP
                      620
    01D3 39           621                 RTS
                      622
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     624 ;
                     625 ;  pseudocode for PUT_DATA
                     626 ;
                     627 ;1 begin
                     628 ;1   MMR_TR_TRANS (MN_READY)
                     629 ;1   MMR_TR_TCU
                     630 ;1   ready_response = get_response
                     631 ;1   case ready_response of
                     632 ;2     rdy_timeout : begin
                     633 ;3                     NIM.M_SIG = ready_time_out
                     634 ;2                   end
                     635 ;2     rdy_ack     : begin
                     636 ;3                     MMR_TR_TRANS (MN_SEND)
                     637 ;3                     MMR_TR_TRANS (NIM.MEM_LEN_HI)
                     638 ;3                     MMR_TR_TRANS (NIM.MEM_LEN_LO)
                     639 ;3                     byte_count = 0
                     640 ;3                     chk_sum = 0
                     641 ;3                     repeat
                     642 ;4                       MMR_TR_TRANS (NIM.MEM_PTR [byte_count])
                     643 ;4                       chk_sum = chk_sum .XOR. NIM.MEM_PTR [byte_count]
                     644 ;4                       byte_count = byte_count + 1
                     645 ;3                     until byte_count = NIM.MEM_LEN
                     646 ;3                     MMR_TR_TRANS (chk_sum)
                     647 ;3                     MMR_TR_TCU
                     648 ;3                     send_response = get_response
                     649 ;3                     case send_response of
                     650 ;4                       snd_timeout : begin
                     651 ;5                                       NIM.M_SIG = send_data_timeout
                     652 ;4                                     end
                     653 ;4                       snd_ack     : begin
                     654 ;5                                       MMR_TR_TRANS (MN_ACK)
                     655 ;5                                       MMR_TR_TCU
                     656 ;5                                       NIM.M_SIG = io_complete
                     657 ;4                                     end
                     658 ;4                       snd_nack    : begin
                     659 ;5                                       MMR_TR_TRANS (MN_ACK)
                     660 ;5                                       MMR_TR_TCU
                     661 ;5                                       NIM.M_SIG = send_data_nack
                     662 ;4                                     end
                     663 ;4                     endcase (send_response)
                     664 ;2                   end
                     665 ;2     rdy_nack    : begin
                     666 ;3                     NIM.M_SIG = ready_nack
                     667 ;2                   end
                     668 ;1   endcase (ready_response)
                     669 ;1 end
                     670 ;
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     672 ;
                     673 ; actual code for PUT_DATA
                     674 ;
    01D4             675 PUT_DATA:
    01D4 86D0        676                 LDAA    #MN_READY               ;1   MMR_TR_TRANS (MN_READY)
    01D6 9A00        677                 ORAA    EXPECTED_NODE,D
    01D8 BD0000      678                 JSR     MMR_TR_TRANS
    01DB 241A        679                 BCC     E_W_R_ERR1
    01DD BD0000      680                 JSR     MMR_TR_TCU              ;1   MMR_TR_TCU
    01E0 2415        681                 BCC     E_W_R_ERR1
                     682
                     683
                     684                                                 ;1   ready_response = get_response
    01E2 BD0000      685                 JSR     MMR_TR_REC
    01E5 2410        686                 BCC     E_W_R_ERR1
                     687
    01E7 BD0000      688                 JSR     VERIFY_CMND
    01EA 2506        689                 BCS     R_N_R1.                 ;if node = expected node
                     690
    01EC C682        691                 LDAB    #RTS_COM_ERR            ;  NIM.M_SIG = rts_com_err
    01EE D7FF        692                 STAB    M_SIG,D
                     693
    01F0 2005        694                 BRA     E_W_R_ERR1
                     695
    01F2             696 R_N_R1:
    01F2 BD0000      697                 JSR     ISOLATE_CMND            ;put command into lower nibble
    01F5 2003        698                 BRA     E_W_R_OK1
                     699
    01F7             700 E_W_R_ERR1:
    01F7 7E02D6      701                 JMP     END_PUT_DATA
                     702
    01FA             703 E_W_R_OK1:                                      ;1   case ready_response of
    01FA 8109        704                 CMPA    #NM_ACK
    01FC 270A        705                 BEQ     RDY_ACK
    01FE 810C        706                 CMPA    #NM_NACK
    0200 2603        707                 BNE     NOT_RDY_NACK
    0202 7E02CC      708                 JMP     RDY_NACK
    0205             709 NOT_RDY_NACK:
    0205 7E02D2      710                 JMP     OTHER_CMND
                     711
    0208             712 RDY_ACK:                                        ;2     rdy_ack      : begin
    0208 8660        713                 LDAA    #MN_SEND                ;3                     MMR_TR_TRANS (MN_SEND)
    020A 9A00        714                 ORAA    EXPECTED_NODE,D
    020C BD0000      715                 JSR     MMR_TR_TRANS
    020F 240E        716                 BCC     GOTO_E_W_R_ERR2
                     717
    0211 96FC        718                 LDAA    MEM_LEN_HI,D            ;3                     MMR_TR_TRANS (NIM.MEM_LEN_HI)
    0213 BD0000      719                 JSR     MMR_TR_TRANS
    0216 2407        720                 BCC     GOTO_E_W_R_ERR2
                     721
    0218 96FD        722                 LDAA    MEM_LEN_LO,D            ;3                     MMR_TR_TRANS (NIM.MEM_LEN_LO)
    021A BD0000      723                 JSR     MMR_TR_TRANS
    021D 2503        724                 BCS     NO_E_W_R_ERR2
                     725
    021F             726 GOTO_E_W_R_ERR2:
    021F 7E02AE      727                 JMP     E_W_R_ERR2
                     728
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    0222              729 NO_E_W_R_ERR2:
                      730
    0222 CE0000       731                 LDX     #00000H                 ;3                  byte_count = 0
    0225 DF00         732                 STX     BYTE_COUNT,D
    0227 7F0002       733                 CLR     CHK_SUM,D               ;3                  chk_sum = 0
    022A              734 REPEAT1:                                ;[      ;3                  repeat
    022A DEFA         735                 LDX     MEM_PTR_HI,D     4
                      736 ;               JSR     MMR_DMA_READ     6      ;4                    MMR_TR_TRANS (NIM.MEM_PTR [byte_count]
                      737
                      738 *****
                      739
    022C 8610         740                  LDAA  #010H      SETUP LATCH ENABLE AND OUTPUT
    022E 970F         741                  STAA  P3CSR      STROBE SELECT
    0230 3C           742                  PSHX             PUT THE ADDRESS WHER BIA REGISTERS
                      743 *                                 CAN GET IT .
    0231 861E         744                  LDAA  #01EH      PORT 1 CONTROL BITS.
    0233 9703         745                  STAA  P2DATA     GET ALL THE DATA READY
    0235 32           746                  PULA             GET HIGH ADDRESS
    0236 9702         747                  STAA  P1DATA     PORT2 HAS HIGH ADDRESS BITS
    0238 32           748                  PULA             Z80 LOW ADDRESS BYTE
    0239 9707         749                  STAA  P4DATA
    023B CC15FF       750                  LDD   #015FFH    SET A TO FF  B TO 00
    023E CE0010       751                  LDX   #0010H     CONSTANTS TO RECONFIGURE FOR INPUTS
    0241 BF0000       752                  STS   STACK_TEMP SAVE THE STACK USE IT AS AN INDEX
    0244 8E0000       753                  LDS   #0000H
                      754 *
                      755 *
                      756 *                       ACTUAL DMA DATA READ STARTS HERE .
                      757 *
                      758 *
    0247 9701         759                  STAA  P2DDR      ISSUE THE DMA REQUEST
    0249 D705         760                  STAB  P4DDR      ENABLE LOW  ADDRESS OUTPUTS
    024B D700         761                  STAB  P1DDR      ENABLE HIGH ADDRESS OUTPUTS
    024D 9606         762                  LDAA  P3DATA     8/12/83   Modification
    024F D706         763                  STAB  P3DATA     DUMMY WRITE TO PORT 3
    0251 9F04         764                  STS   P3DDR
    0253 DF00         765                  STX   P1DDR
    0255 9606         766                  LDAA  P3DATA     GET THE DATA LATCHED IN PORT 3
                      767 *
                      768 *
                      769 *                       Z80 READ IS COMPLETE DATA IS IN ACCUMULATOR A
                      770 *
                      771 *
    0257 8E0000       772                  LDS   STACK_TEMP  RESTORE THE STACK REGISTER
                      773
                      774 *****
                      775
    025A DEFA         776                 LDX     MEM_PTR_HI,D     4
    025C 08           777                 INX                      3
    025D DFFA         778                 STX     MEM_PTR_HI,D     4
    025F 36           779                 PSHA                     3
                      780 ;               JSR     MMR_TR_TRANS     6
                      781
                      782 *****
                      783
                      784 ; local equates
                      785 ;
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

            (0020)   786 TDRE_MASK      EQU     020H                    ;"transmit_data_register_empty" mask
            (0660)   787 TIMER          EQU     0660H                   ;10MS
                     788
    0260 CE0660      789                 LDX     #TIMER          3
    0263             790 REPEAT:                                         ;1 REPEAT
                     791                                                 ;2   TDRE = SC1_TR_CS .AND. TDRE_MASK
    0263 09          792                 DEX                     3
    0264 270B        793                 BEQ     TO1             3
                     794 *
    0266 D611        795                 LDAB    011H,D          3       ;       get the control/status byte
    0268 C420        796                 ANDB    #TDRE_MASK      2       ;       mask in the TDRE bit
    026A 27F7        797                 BEQ     REPEAT          3       ;1 UNTIL TDRE = TRUE
                     798
    026C 9713        799                 STAA    013H,D          3       ;1 SC1_TX = NEXT_BYTE_OUT
                     800
    026E 0D          801                 SEC                      2
    026F 200C        802                 BRA     END_TR           3
                     803                         ;*TOTAL    25
                     804
            (0271)   805 TO1            EQU     $
    0271 8698        806                 LDAA    #TRANS_TO1  ;ERR CODE FOR TRANS TIME OUT
    0273 97FF        807                 STAA    M_SIG,D
    0275 0C          808                 CLC
    0276 2005        809                 BRA     END_TR
                     810
    0278             811 TO2:
    0278 8699        812                 LDAA    #TRANS_TO2
    027A 97FF        813                 STAA    M_SIG,D
    027C 0C          814                 CLC
                     815
    027D             816 END_TR:
    027D 242F        817                 BCC     E_W_R_ERR2      3
    027F 32          818                 PULA                    4
    0280 9802        819                 EORA    CHK_SUM,D       3       ;4                      chk_sum = chk_sum .XOR. NIM.MEM_PTR lb
    0282 9702        820                 STAA    CHK_SUM,D       3
                     821                                                 ;4                      byte_count = byte_count + 1
    0284 DE00        822                 LDX     BYTE_COUNT,D    4
    0286 08          823                 INX                     3
    0287 DF00        824                 STX     BYTE_COUNT,D    4
    0289 9CFC        825                 CPX     MEM_LEN_H1,D    5
    028B 269D        826                 BNE     REPEAT1         3       ;3                  until byte_count = NIM.MEM_LEN
                     827                         ;*TOTAL    58
    028D 9602        828                 LDAA    CHK_SUM,D               ;3                  MMR_TR_TRANS (chk_sum)
    028F BD0000      829                 JSR     MMR_TR_TRANS
    0292 241A        830                 BCC     E_W_R_ERR2
    0294 BD0000      831                 JSR     MMR_TR_TCU              ;3                  MMR_TR_TCU
    0297 2415        832                 BCC     E_W_R_ERR2
                     833
                     834                                                 ;3                  send_response = get_response
    0299 BD0000      835                 JSR     MMR_TR_REC
    029C 2410        836                 BCC     E_W_R_ERR2
                     837
    029E BD0000      838                 JSR     VERIFY_CMND
    02A1 2506        839                 BCS     R_N_R2                  ;if node = expected node
                     840
    02A3 C604        841                 LDAB    #SEND_COM_ERR           ;  NIM.M_SIG = rts_com_err
    02A5 D7FF        842                 STAB    M_SIG,D
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     843
    02A7 2005        844                 BRA     E_W_R_ERR2
                     845
    02A9             846 R_N_R2:
    02A9 BD0000      847                 JSR     ISOLATE_CMND            ;put command into lower nibble
    02AC 2002        848                 BRA     E_W_R_OK2
                     849
    02AE             850 E_W_R_ERR2:
    02AE 2026        851                 BRA     END_PUT_DATA
                     852
    02B0             853 E_W_R_OK2:                                      ;1   case ready_response of
                     854                                                ;3                     case send_response of
                     855                                                ;4                       snd_timeout : begin
                     856                                                ;5                                     NIM.M_SIG = send_data_
                     857                                                ;4                                   end
                     858                                                ;4                       snd_ack.    : begin
                     859                                                ;5                                     MMR_TR_TRANS (MN_ACK)
                     860                                                ;5                                     MMR_TR_TCU
                     861                                                ;5                                     NIM.M_SIG = io_complet
                     862                                                ;4                                   end
                     863                                                ;4                       snd_nack    : begin
                     864                                                ;5                                     MMR_TR_TRANS (MN_ACK)
                     865                                                ;5                                     MMR_TR_TCU
                     866                                                ;5                                     NIM.M_SIG = send_data_
                     867                                                ;4                                   end
                     868                                                ;4                     endcase (send_response)
    02B0 8109        869                 CMPA    #NM_ACK
    02B2 2706        870                 BEQ     SEND_DATA_ACK
                     871
    02B4 810C        872                 CMPA    #NM_NACK
    02B6 2708        873                 BEQ     SEND_DATA_NACK
                     874
    02B8 200C        875                 BRA     OTHER_CMND2
                     876
    02BA             877 SEND_DATA_ACK:
                     878 ;               LDAA    #MN_ACK
                     879 ;               ORAA    EXPECTED_NODE,D
                     880 ;               JSR     MMR_TR_TRANS
                     881 ;               JSR     MMR_TR_TCU
                     882
    02BA C680        883                 LDAB    #MSIG_COMPLETE
    02BC D7FF        884                 STAB    M_SIG,D
                     885
    02BE 2016        886                 BRA     END_PUT_DATA
                     887
    02C0             888 SEND_DATA_NACK:
                     889 ;               LDAA    #MN_ACK
                     890 ;               ORAA    EXPECTED_NODE,D
                     891 ;               JSR     MMR_TR_TRANS
                     892 ;               JSR     MMR_TR_TCU
                     893
    02C0 C688        894                 LDAB    #SEND_DATA_NAK
    02C2 D7FF        895                 STAB    M_SIG,D
                     896
    02C4 2           897                 BRA     END_PUT_DATA
                     898
    02C6             899 OTHER_CMND2:
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    02C6 C685          900                 LDAB    #SEND_DATA_BR
    02C8 D7FF          901                 STAB    M_SIG,D
                       902
    02CA 200A          903                 BRA     END_PUT_DATA
                       904
    02CC               905 RDY_NACK:                                   ;2      rdy_nack     : begin
    02CC C686          906                 LDAB    #READY_NACK         ;3                       NIM.M_SIG = ready_nack
    02CE D7FF          907                 STAB    M_SIG,D
    02D0 2004          908                 BRA     ENDCASE_RDY         ;2                     end
    02D2               909 OTHER_CMND:
    02D2 C687          910                 LDAB    #READY_B_R
    02D4 D7FF          911                 STAB    M_SIG,D
                       912
                       913 ;               BRA     ENDCASE_RDY
    02D6               914 ENDCASE_RDY:                                ;1   endcase (ready_response)
    02D6               915 END_PUT_DATA:
    02D6 39            916                 RTS
                       917
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     919 ;
                     920 ; local data areas defined
                     921 ;
                     922                 DATA
                     923
    0000             924 BYTE_COUNT      RMB     2
    0002             925 CHK_SUM         RMB     1

Errors=    0
```

LINE#	SYMBOL	TYPE	REFERENCES
	ASIG_MAX_CODE	A	263
95	A_SIG	A	262
138	BAD_ASIG	A	273
271	BAD_ASIG_COMMAN	P	264
157	BAD_ORFE	A	249
155	BAD_RDRF	A	240
156	BAD_TDRE	A	231
924	BYTE_COUNT	D	377,478,487,570,572,732,822,824
279	CASE_TABLE	P	266
925	CHK_SUM	D	490,563,564,578,733,819,820,828
134	CLR_BR	A	465
133	CLR_ERR	A	449
470	CLR_SEND	P	463
132	CLR_TMOUT	A	
88	CNFG_WORD	A	256
136	DATAIN_TIMEOUT	A	
593	DATA_IN_CS_OK	P	579
135	DATA_IN_ERROR	A	581
492	DATA_IN_LOOP	P	574
489	DO_REC_STATUS	P	379
301	ENDCASE	P	275,288,291,294,297,300
914	ENDCASE_RDY	P	908
607	END_GET_DATA	P	415,426,432,459,468,474,483,494,577,589,591,602,605
608	END_GET_STATUS	P	364,373
915	END_PUT_DATA	P	701,851,886,897,903
816	END_TR	P	802,809
117	ERR_CODES	A	119,120,121,122,123,124,125,126,128,129,130,131,132,133,134,135,136,138,140,141,142,144,146,148,150,152 154,155,156,157
195	EXPECTED_NODE	E	258,314,341,392,436,587,600,677,714
700	E_W_R_ERR1	P	679,681,686,694
850	E_W_R_ERR2	P	727,817,830,832,836,844
414	E_W_R_ERR3	P	394,397,400,408
458	E_W_R_ERR4	P	438,441,444,452
363	E_W_R_ERR5	P	343,346,349,357
703	E_W_R_OK1	P	698
853	E_W_R_OK2	P	848
417	E_W_R_OK3	P	412
461	E_W_R_OK4	P	456
366	E_W_R_OK5	P	361
295	GET	P	283
390	GET_DATA	P	296
339	GET_STATUS	P	293
726	GOTO_E_W_R_ERR2	P	716,720
375	GOT_NM_STATUS	P	368
152	HAVE_ORFE	A	151
286	IDLE	P	280
194	ISOLATE_CMND	E	360,411,455,697,847
613	LINE_TURN_DELAY	P	220,321,584,597,609
210	LOOP	P	212
617	L_T_D_LOOP	P	619
92	MEM_LEN	A	93,94,472,486
93	MEM_LEN_HI	A	718,825
94	MEM_LEN_LO	A	722
89	MEM_PTR	A	90,91
90	MEM_PTR_HI	A	496,566,568,735,776,778
91	MEM_PTR_LO	A	
193	MMR_DMA_READ	F	

LINE#	SYMBOL	TYPE	REFERENCES
193	MMR_DMA_WRITE	E	
198	MMR_MAC	P	197
194	MMR_TR_REC	E	348,399,443,471,480,493,576,685,835
192	MMR_TR_TCU	E	318,345,396,440,590,603,680,831
192	MMR_TR_TRANS	E	315,342,393,437,588,601,678,715,719,723,829
103	MN_ACK	A	599
106	MN_CANCEL	A	
104	MN_CLR	A	435
108	MN_NACK	A	586
109	MN_READY	A	676
105	MN_RECEIVE	A	391
101	MN_RESET	A	313
107	MN_SEND	A	713
102	MN_STATUS	A	340
	MSIG_COMPLETE	A	323,594,883
96	M_SIG	A	232,241,250,274,324,355,371,406,424,430,450,466,582,595,692,807,813,842,884,895,901,907,911
112	NM_ACK	A	418,704,869
113	NM_CANCEL	A	
115	NM_NACK	A	420,706,872
114	NM_SEND	A	462
111	NM_STATUS	A	367
82	NODE_MASK	A	257
709	NOT_RDY_NACK	P	707
729	NO_E_W_R_ERR2	P	724
476	N_E_G_D1	P	472
485	N_E_G_D2	P	481
254	ORFE_OK	P	247
909	OTHER_CMND	P	710
899	OTHER_CMND2	P	875
507	P1DATA	A	535,747
505	P1DDR	A	551,554,761,765
508	P2DATA	A	539,745
506	P2DDR	A	548,759
513	P3CSR	A	531,741
511	P3DATA	A	532,552,762,763,766
509	P3DDR	A	549,553,764
512	P4DATA	A	537,749
510	P4DDR	A	550,760
298	PUT	P	284
675	PUT_DATA	P	299
245	RDRF_OK	P	238
712	RDY_ACK	P	705
905	RDY_NACK	P	708
125	READY_B_R	A	910
124	READY_NACK	A	906
119	READY_TMOUT	A	
434	RECEIVE_ACK	P	419
428	RECEIVE_NACK	P	421
129	RECV_BR	A	405
130	RECV_DATA_BR	A	423
131	RECV_NACK	A	429
128	RECV_TMOUT	A	
790	REPEAT	P	797
734	REPEAT1	P	826
289	RES	P	281
312	RESL_DEVICE	P	290
326	RS_EXIT	P	316,319

LINE#	SYMBOL	TYPE	REFERENCES
120	RTS_COM_ERR	A	691
696	R_N_R1	P	689
846	R_N_R2	P	839
410	R_N_R3	P	403
454	R_N_R4	P	447
359	R_N_R5	P	352
122	SEND_COM_ERR	A	841
877	SEND_DATA_ACK	P	870
123	SEND_DATA_BR	A	900
888	SEND_DATA_NACK	P	873
126	SEND_DATA_NAK	A	894
121	SEND_TMOUT	A	
304	SKIP	P	
83	SOME_ERROR	A	
500	STACK_TEMP	E	543,557,752,772
292	STATUS	P	282
142	STATUS_BR	A	370
141	STAT_BR	A	354
337	STAT_PKT_SIZE	A	376
140	STAT_TMOUT	A	
144	TCU_ERR1	A	143
146	TCU_ERR2	A	145
786	TDRE_MASK	A	796
236	TDRE_OK	P	229
787	TIMER	A	789
154	TIME_OUT_ERR	A	153
805	TO1	P	793
811	TO2	P	
148	TRANS_TO1	A	147,806
150	TRANS_TO2	A	149,812
194	VERIFY_CMND	E	351,402,446,688,838

LOCATION OBJECT CODE LINE SOURCE LINE

```
                    1 ^S801^
                    3  NAME ^Rev 02 - RPD^
                    4
                    5 De_D_MMR MACRO              ;Header Rev. 4
                    6                  .GOTO Ede_D_MMR
                    7
                    8  Project:      NET, 83-101
                    9
                   10  ********************************************************
                   11  *                                                      *
                   12  *     D_MMR                          DLS               *
                   13  *                                                      *
                   14  ********************************************************
                   15
                   16        Rev History
                   17        Rev.   Date          Name      Change
                   18         4     23jul2323    GFB       MODIFY NIM BLOCK/MASTER_AP COMP.
                   19         3     23jul2314    GFB       COMMON BLOCK DEF FOR NIM
                   20         2     20jul815p    RPD       added ORG for NIM
                   21         1     14jul0210    DLS       CUSTOMIZED FOR MASTER NODE
                   22         0     13jul1015    DLS       Initial Pseudo code
                   23
                   24 Ede_D_MMR  MEND
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      26 ****************************************************************
                      27 *                                                              *
                      28 *  MODULE NAME:                                                *
                      29 *                                                              *
                      30 *    D_MMR                                                     *
                      31 *                                                              *
                      32 *  FUNCTION(S):                                                *
                      33 *                                                              *
                      34 *     1.  TO DECLARE THE DATA AREA "NIM_BLOCK."                *
                      35 *     2.  TO DECLARE THE D1_MODE_WORD.                         *
                      36 *                                                              *
                      37 *  NOTES:                                                      *
                      38 *                                                              *
                      39 * 1. NIM_BLOCK IS USED AS THE INTERFACE BETWEEN THE            *
                      40 *    MEDIUM ACCESS CONTROLLER AND THE RESIDENT APPLICATION     *
                      41 *    PROGRAM.                                                  *
                      42 *                                                              *
                      43 * 2. THE INSTALLER IS RESPONSIBLE FOR LOCATING THIS DATA       *
                      44 *    MODULE SO THAT THE LAST BYTE ENDS AT LOCATION 127 (DEC).  *
                      45 *                                                              *
                      46 ****************************************************************
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      48                GLB     D_MMR
                      49                GLB     D1_MODE_WORD
                      50                GLB     NIM_DMA_BLK
                      51                GLB     CNFG_WORD
                      52                GLB     A_SIGNAL
                      53                GLB     A_SIG
                      54                GLB     MEM_PTR
                      55                GLB     MEM_PTR_HI
                      56                GLB     MEM_PTR_LO
                      57                GLB     MEM_LEN
                      58                GLB     MEM_LEN_HI
                      59                GLB     MEM_LEN_LO
                      60                GLB     MEM_SIGNAL
                      61                GLB     MEM_SIG
                      62                GLB     EXPECTED_NODE
                      63                GLB     NXT_STATE
                      64                GLB     TIMEFORRESPONSE
                      65                GLB     INDEXTOMEM
                      66                GLB     TIME
                      67                ORG     83H
0083                  68 D_MMR:
                      69 ***************************************************************
                      70 *                                                             *
                      71 *  DATA_WORD:                                                 *
                      72 *                                                             *
                      73 *    D1_MODE_WORD                                             *
                      74 *                                                             *
                      75 *   FUNCTION:                                                 *
                      76 *                                                             *
                      77 *    CONTAINS THE STATE OF SEQUENCER PROCESSING               *
                      78 *                                                             *
                      79 ***************************************************************
                      80
                      81                DATA
                      82                ORG     90H
0090                  83 D1_MODE_WORD   RMB     1
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      85 ************************************************************************
                      86 *                                                                      *
                      87 *  DATA WORD:                                                          *
                      88 *                                                                      *
                      89 *    EXPECTED_NODE                                                     *
                      90 *                                                                      *
                      91 *   FUNCTION:                                                          *
                      92 *                                                                      *
                      93 *    CONTAINS THE NODE ADDR OF THE CURRENT TOKEN HOLDER.               *
                      94 *                                                                      *
                      95 *                                                                      *
                      96 *                                                                      *
                      97 ************************************************************************
0091                  98 EXPECTED_NODE  RMB     1         ;CURRENT TOKEN HOLDER
0092                  99 NXT_STATE      RMB     1         ;NXT STATE OF EP PROCESSING
0093                 100 TIMEFORRESPONSE RMB     1         ;TRIGGERS TIMING ON DATA TRANSMISSION
0094                 101 TIMER           RMB     2         ;USED TO SET REG_X AS INPUT TO TMR MODULE
0096                 102 INDEXTOMEM      RMB     1         ;POINTER INTO Z80 MEMORY
```

LOCATION OBJECT CODE LINE SOURCE LINE

```
104                 INCLUDE I_NIM_MR
  + ************************************************************************
  + *                                                                      *
  + *                        HIERARCHY  CHART                              *
  + *                             OF                                       *
  + *                   MEDIUM  ACCESS  CONTROLLER                         *
  + *                            (MAC)                                     *
  + *                            MASTER                                    *
  + *                                                                      *
  + *                           (MAC)                                      *
  + *                             |                                        *
  + *                             |                                        *
  + *          +------------------+-----------------------+                *
  + *        (TR)                (DMA)                  (ACM)             *
  + *          |                   |                      |                *
  + *      +---+---+-------+       |                      |                *
  + * (TRANS)  (REC)  (TCU)       |                      |                *
  + *                             |                      |                *
  + *                  +----------+----------+           |                *
  + *                  |                     |           |                *
  + *              (WRITE)               (READ)          |                *
  + *                                                    |                *
  + *                                          +---------+-----+-----+    *
  + *                                          |               |     |    *
  + *                                       (EVENT_PROC)    (SEQ) (TMR)(RESP)*
  + *                                                                      *
  + ************************************************************************
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                    1 ********************************************************************
                    2 *                  INTERFACE MODULE DESCRIPTION                    *
                    3 *                  ----------------------------                    *
                    4 *  NAME:                                                           *
                    5 *   I_NIM_MR                                                       *
                    6 *                                                                  *
                    7 *  FUNCTION:                                                       *
                    8 *      TO DEFINE THE INTERFACE BETWEEN THE 6801_MASTER_MAC AND     *
                    9 *      6801_MASTER_APP.                                            *
                   10 *                                                                  *
                   11 *  DESCRIPTION:                                                    *
                   12 *      A BLOCK OF MEMORY WILL BE SHARED BY THE MAC AND APP,        *
                   13 *      WHEREIN DATA AND CONTROL SIGNALS WILL BE PASSED BACK        *
                   14 *      AND FORTH BETWEEN THE TWO. A DIAGRAM OF THIS BLOCK          *
                   15 *      (REFERRED TO AS NIM_BLOCK) FOLLOWS:                         *
                   16 *                                                                  *
                   17 *            NIM_DMA_BLK                                           *
                   18 *           +-+----+------+                                       *
                   19 *           |S|////|DEV_ID|  A(W/R), M(R/RESET);                  *
                   20 *           +-+----+------+                                       *
                   21 *           | MEM_PTR_HI  |  A(W), M(R  );                        *
                   22 *           +-------------+                                       *
                   23 *           | MEM_PTR_LO  |    "     "                            *
                   24 *           +-------------+                                       *
                   25 *           | MEM_LEN_HI  |  A(W), M(R);                          *
                   26 *           +-------------+                                       *
                   27 *           | MEM_LEN_LO  |  A(W), M(R);                          *
                   28 *           +-------------+                                       *
                   29 *           | A_SIGNAL    |  A(W), M(R);                          *
                   30 *           +-------------+                                       *
                   31 *           | M_SIGNAL    |  A(R/RESET), M(W);                    *
                   32 *           +-------------+                                       *
                   33 *                                                                  *
                   34 *  NOTES:                                                          *
                   35 *     WHEN THE APP CALLS THE MAC THE FOLLOWING MUST TAKE PLACE:    *
                   36 *                                                                  *
                   37 *       1. APP SETS A_SIGNAL TO THE DESIRED FUNCTION.              *
                   38 *       2. APP SETS M_SIG TO IO_IDLE. (0)                          *
                   39 *       3. APP SETS MEM_PTR IF APPLICABLE.                         *
                   40 *       4. APP SETS MEM_LEN IF APPLICABLE.                         *
                   41 *       5. LASTLY, THE APP SETS DEVICE ID AND                      *
                   42 *          APP SETS CNFG_WORD.BIT_7.                               *
                   43 *       6. APP CALLS MAC.                                          *
                   44 *       7. MAC WILL EXECUTE ORDERS OF MAC.                         *
                   45 *       8. WHEN MAC IS COMPLETE, IT WILL WRITE TO NIM.M_SIGNAL     *
                   46 *          AS WELL AS CLR THE CNFG_WORD.BIT_7.                     *
                   47 *       9. MAC WILL DEPOSIT THE TRUE LENGTH OF DATA WRITTEN        *
                   48 *          TO Z80_MEM IN MEM_LEN. THUS, APP SHOULD HAVE SAVED      *
                   49 *          ORIGINAL MEM_LEN BEFORE CALL TO MAC.                    *
                   50 *      10. THE APP WILL READ THE NIM AND CLEAR A_SIG, M_SIG,       *
                   51 *          MEM_PTR, MEM_LEN AND CNFG_WORD. (BIT 7=0 ALREADY)       *
                   52 *                                                                  *
                   53 ********************************************************************
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                    + ****************************************************************
                    + *                                                              *
                    + *  DATA ELEMENT DEFINITIONS:                                   *
                    + *                                                              *
                    + *     CNFG_WORD :                                              *
                    + *     ----------                                               *
                    + *       BIT.7:  CLR = APP SHOULD NOT CALL MAC                  *
                    + *               SET = APP SHOULD CALL MAC                      *
                    + *                                                              *
                    + *     A_SIG:                                                   *
                    + *     -----                                                    *
                    + *        0- NO SIGNAL (IDLE).                                  *
                    + *        1- RESET DEVICE D                                     *
                    + *        2- GET STATUS FROM DEVICE D.                          *
                    + *        3- GET DATA FROM DEVICE D.                            *
                    + *        4- PUT DEVICE TO DEVICE TO D.                         *
                    + *                                                              *
(0001)              + ASIG_RESET_D    EQU             001H
(0002)              + ASIG_STAT_D     EQU             002H
(0003)              + ASIG_GET_D      EQU             003H
(0004)              + ASIG_PUT_D      EQU             004H
(0004)              + ASIG_MAX_CODE   EQU             ASIG_PUT_D
                    + *     MEM_POINTER                                              *
                    + *     -----------                                              *
                    + *                                                              *
                    + *        (CAN ACCESS UP TO 64K OF Z80 MEMORY)                  *
                    + *                                                              *
                    + *     A_LEN:                                                   *
                    + *     -----                                                    *
                    + *                                                              *
                    + *        (CAN RANGE UP TO 64K)                                 *
                    + *                                                              *
                    + *     M_SIG:                                                   *
                    + *     -----                                                    *
                    + *        0- IO_IDLE                                            *
                    + *        1- IO_COMPLETE.                                       *
                    + *        2- IO_BUSY                                            *
                    + *        3- IO_FAILED                                          *
                    + *                                                              *
(0000)              + MSIG_IDLE       EQU             000H
(0080)              + MSIG_COMPLETE   EQU             080H
(0002)              + MSIG_TIMEOUT    EQU             002H
(0003)              + MSIG_FAILED     EQU             003H
                    + *                                                              *
                    + *   NOTES:                                                     *
                    + *     * 1. A:= APPLICATION SIDE OF NODE.                       *
                    + *       2. M:= MAC SIDE OF NODE.                               *
                    + *       3. x(W):= THIS ELEMENT HAS WRITE ACCESS TO THIS BYTE.  *
                    + *       4. x(R/RESET):= THIS ELEMENT HAS READ ACCESS AS WELL AS *
                    + *          RESET ACCESS( WRITE ZERO ONLY) TO THIS BYTE.        *
                    + *                                                              *
                    + ****************************************************************
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                    + ********************************************************************
                    + *                                                                  *
                    + *  NOTES TO INSTALLER OF THIS MAC/APP:                             *
                    + *                                                                  *
                    + *   1. THE APP IS RESPONSIBLE FOR INITIALIZING ALL OF RAM.         *
                    + *                                                                  *
                    + *   2. THE APP MUST INITIALIZE THE CONTROL AND STATUS REG.AS       *
                    + *      WELL AS THE BAUD RATE AND MODE REGISTER.                    *
                    + *                                                                  *
                    + *   3. THE D1_MODE_WORD MUST BE SET TO ZERO AT PWR UP BY THE       *
                    + *      APP.                                                        *
                    + *                                                                  *
                    + *   4. THE NIM_BLOCK WILL END AT ADDR 255 DECIMAL.                 *
                    + *                                                                  *
                    + ********************************************************************
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     106 ;
                     107 ; master NIM block data area (NOTE: this data block must end at 00FFH)
                     108 ;
                     109                    COMN
                     110 *                  ORG              00F9H          START ADDRESS OF NIM
     0000            111 CNFG_WORD
     0000            112 NIM_DMA_BLK        RMB              1              !SI////IDEV_ID!
     0001            113 MEM_PTR
     0001            114 MEM_PTR_HI         RMB              1              ! MEM_PTR_HI   !
     0002            115 MEM_PTR_LO         RMB              1              ! MEM_PTR_LO   !
     0003            116 MEM_LEN
     0003            117 MEM_LEN_HI         RMB              1              ! MEM_LEN_HI   !
     0004            118 MEM_LEN_LO         RMB              1              ! MEM_LEN-LO   !
     0005            119 A_SIG
     0005            120 A_SIGNAL           RMB              1              ! A_SIGNAL     !
     0006            121 M_SIG
     0006            122 M_SIGNAL           RMB              1              ! M_SIGNAL     !
                     123 *
                     124 *                  END OF NIM AREA (00FFH)
                     125 *
                     126 *

Errors=     0
```

LINE#	SYMBOL	TYPE	REFERENCES
119	A_SIG	C	53
120	A_SIGNAL	C	52
111	CNFG_WORD	C	51
83	D1_MODE_WORD	A	49
68	D_MMR	D	48
98	EXPECTED_NODE	A	62
102	INDEXTOMEM	A	65
116	MEM_LEN	C	57
117	MEM_LEN_HI	C	58
118	MEM_LEN_LO	C	59
113	MEM_PTR	C	54
114	MEM_PTR_HI	C	55
115	MEM_PTR_LO	C	56
121	M_SIG	C	61
122	M_SIGNAL	C	60
112	NIM_DMA_BLK	C	50
99	NXT_STATE	A	63
100	TIMEFORRESPONSE	A	64
101	TIMER	A	66

LOCATION OBJECT CODE LINE   SOURCE LINE

```
^6801^
 NAME ^Rev 03 - RPD^

De_MMR_TR_REC MACRO              ;Header Rev. 4
               .GOTO Ede_MMR_TR_REC

 Project:       NET, 83-101

 ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
 ▒                            ▒
 ▒    MMR_TR_REC          DLS ▒
 ▒                            ▒
 ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒

      Rev History
      Rev.  Date         Name     Change
       4    24JUL420P    WPC      TAKE OUT VALIDITY CHECKS
       3    20jul1605p   RPD      added read of control/status flag to reset RDRF
       2    13jul1750a   DLS      CUSTOMIZED FOR MASTER NODE.
       1    13jul1750a   RPD      converted pseudo code to 6801 code
       0    12JUL1305    DLS      Initial Pseudo code

Ede_MMR_TR_REC  MEND
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      26 ************************************************************************
                      27 *                                                                      *
                      28 *  MODULE NAME:                                                        *
                      29 *                                                                      *
                      30 *    MMR_TR_REC                                                        *
                      31 *                                                                      *
                      32 *  INPUTS:                                                             *
                      33 *                                                                      *
                      34 *    NET_BYTE_IN (LOCATION 12)                                         *
                      35 *    D1_MODE_WORD (DECLARED IN D_MMR)                                  *
                      36 *                                                                      *
                      37 *  FUNCTION(S):                                                        *
                      38 *                                                                      *
                      39 *     1. TO GET A BYTE FROM THE NETWORK.                               *
                      40 *                                                                      *
                      41 *  OUTPUTS:                                                            *
                      42 *                                                                      *
                      43 *     NET_BYTE_IN (REG_A)                                              *
                      44 *     TOKEN :  CARRY SET = BYTE FOR THIS NODE.                         *
                      45 *              CARRY CLR = NOT FOR THIS NODE.                          *
                      46 *                                                                      *
                      47 *  CALLS:                                                              *
                      48 *                                                                      *
                      49 *    NONE.                                                             *
                      50 *                                                                      *
                      51 *  CALLED BY:                                                          *
                      52 *                                                                      *
                      53 *    MMR_ACM_SEQ                                                       *
                      54 *                                                                      *
                      55 *  NOTES:                                                              *
                      56 *                                                                      *
                      57 *    NONE.                                                             *
                      58 *                                                                      *
                      59 ************************************************************************
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      61 ************************************************************************
                      62 *                                                                      *
                      63 *  PSEUDO CODE:                                                        *
                      64 *                                                                      *
                      65 *    MMR_TR_REC:                                                       *
                      66 *                                                                      *
                      67 *    CARRY=SET;                                                        *
                      68 *    REG_A=MEM(12);                                                    *
                      69 *    IF D1_MODE_WORD() CONTROL                                         *
                      70 *        THEN                                                          *
                      71 *          GOTO RT_REC;   /* RECEIVING DATA MODE */                    *
                      72 *    ENDIF;                                                            *
                      73 *                                                                      *
                      74 *       SAVE_REG_A = REG_A;                                            *
                      75 *       REG_A = $0F.AND.REG_A; /* LOWER HALF = ADDR */                 *
                      76 *       IF NODE_ADDR () REG_A                                          *
                      77 *          THEN                                                        *
                      78 *            CARRY=0;                                                  *
                      79 *            GOTO RT_REC;                                              *
                      80 *       ENDIF;                                                         *
                      81 *       REG_A=$F0.AND.SAVE_REG_A;/* UPPER HALF = CMND */            *
                      82 *       SHIFT REG_A TO LOWER NIBBLE;                                   *
                      83 *  RT_REC: RETURN;                                                     *
                      84 *                                                                      *
                      85 ************************************************************************
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     97                INCLUDE PG0_EQU
                      + ;
                      + ; 6801 internal register equates (page 0)
                      + ;
(0000)                + P1_DIR          EQU     000H                    ;port 1 data direction register
(0002)                + P1_DATA         EQU     002H                    ;port 1 data register
                      +
(0001)                + P2_DIR          EQU     001H                    ;port 2 data direction register
(0003)                + P2_DATA         EQU     003H                    ;port 2 data register
                      +
(0004)                + P3_DIR          EQU     004H                    ;port 3 data direction register
(0006)                + P3_DATA         EQU     006H                    ;port 3 data register
                      +
(0005)                + P4_DIR          EQU     005H                    ;port 4 data direction register
(0007)                + P4_DATA         EQU     007H                    ;port 4 data register
                      +
(0008)                + T_CNTLSTAT      EQU     008H                    ;timer control and status register
(0009)                + T_CNTRHGH       EQU     009H                    ;counter high byte
(000A)                + T_CNTRLOW       EQU     00AH                    ;counter low byte
(000B)                + T_OCMPHGH       EQU     00BH                    ;output compare register high byte
(000C)                + T_OCMPLOW       EQU     00CH                    ;output compare register low byte
(000D)                + T_ICAPHGH       EQU     00DH                    ;input capture register high byte
(000E)                + T_ICAPLOW       EQU     00EH                    ;input capture register low byte
                      +
(000F)                + P3_CNTLSTAT     EQU     00FH                    ;port 3 control and status register
                      +
(0010)                + SCI_RM          EQU     010H                    ;rate and mode control register
(0011)                + SCI_TR_CS       EQU     011H                    ;transmit/receive control and status register
(0012)                + SCI_RX          EQU     012H                    ;receive data register
(0013)                + SCI_TX          EQU     013H                    ;transmit data register
                      +
(0014)                + RAM_CNTL        EQU     014H                    ;RAM control register
                     88
                     89 ;
                     90 ; local equates
                     91 ;
(000F)               92 ADDR_MASK       EQU     00FH
(00F0)               93 RESP_MASK       EQU     0F0H
                     94
                     95                 EXT     EXPECTED_NODE
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      97                PROG
                      98
                      99 ; WAIT FOR READER_DATA_REGISTER_FULL FLAG = TRUE
                     100 ; READ NET_BYTE_IN
                     101
                     102 ; INPUTS: NONE
                     103 ; OUTPUTS: A = NET_BYTE_IN
                     104 ;          CARRY FLAG SET: NO TIME-OUT WHILE WAITING FOR A BYTE
                     105 ;                     CLR: TIME-OUT OCCURRED WHILE WAITING FOR A BYTE IN
                     106 ;          B = ERROR CODE
                     107 ;              1= TIME_OUT ERR
                     108 ;              2= FRAMING_ERR
                     109
                     110                GLB     MMR_TR_REC
          (0021)     111 START_TIMER    EQU     500/15                  ;TIME OUT = 500us
                     112                                                ;LOOP TIME = 15 CYCLES
0000                 113 MMR_TR_REC:
0000 CE0021          114                LDX     #START_TIMER
          (0003)     115 LOOP:          EQU     $
0003 09              116                DEX                             ;COUNT DOWN
0004 2715            117                BEQ     TIME_OUT                ;BRANCH IF TIME RUNS OUT
                     118 *
                     119 *   CHECK TO SEE IF THE RECEIVE SHIFT REGISTER HAS BEEN DUMPED
                     120 *    INTO THE RECEIVE REGISTER.
                     121 *
0006 9611            122                LDAA    SCI_TR_CS,D
0008 2B02            123                BMI     GOT_DATA                ;BRANCH IF NEW DATA ARRIVED
                     124 *
000A 20F7            125                BRA     LOOP                    ;GO BACK FOR MORE
                     126 *
          (000C)     127 GOT_DATA:      EQU     $
000C D612            128                LDAB    SCI_RX,D                ;GET DATA
000E 8440            129                ANDA    #01000000B              ;LOOK AT BIT 6, CHECK OVERRUN CONDITION
0010 2706            130                BEQ     GOOD_DATA
                     131 *
                     132                EXT     HAVE_ORFE,M_SIG
                     133
0012 C600            134                LDAB    #HAVE_ORFE
0014 D700            135                STAB    M_SIG,D
                     136
0016 0C              137                CLC                             ;INDICATE ERROR
0017 39              138                RTS
                     139 *
          (0018)     140 GOOD_DATA:     EQU     $
0018 17              141                TBA                             ;PLACE DATA IN A
0019 0D              142                SEC
001A 39              143                RTS
                     144 *
          (001B)     145 TIME_OUT:      EQU     $
                     146
                     147                EXT     TIME_OUT_ERR
                     148
001B C600            149                LDAB    #TIME_OUT_ERR
001D D700            150                STAB    M_SIG,D
                     151
001F 0C              152                CLC                             ;INDICATE ERROR
0020 39              153                RTS
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     154
                     155 ; TAKE A NET_BYTE_IN THAT IS EXPECTED TO BE A COMMAND BYTE
                     156 ;  AND VERIFY THAT THE CORRECT NODE IS RESPONDING
                     157
                     158 ; INPUTS: A = NET_BYTE_IN
                     159 ; OUTPUTS: CARRY FLAG SET: VALID COMMAND BYTE
                     160 ;                     CLR: INVALID COMMAND BYTE
                     161
                     162                 GLB     VERIFY_CMND
                     163
    0021             164 VERIFY_CMND:
    0021 36          165                 PSHA                            ;2   SAVE_NBI = NET_BYTE_IN                    /* NOT REC
    0022 840F        166                 ANDA    #ADDR_MASK              ;2   ADDRESS = NET_BYTE_IN .AND. ADDR_MASK     /* LOWER H
    0024 9100        167                 CMPA    EXPECTED_NODE,D         ;2   IF ADDRESS = NODE_ADDR
    0026 2603        168                 BNE     WRONG_NODE
                     169                                                 ;3     NET_BYTE_IN = SAVE_NBI .AND. CMND_MASK   /* UPPER H
                     170
    0028 0D          171                 SEC                             ; RIGHT NODE
                     172
    0029 2001        173                 BRA     END_V_C_B               ; DONE
                     174
    002B             175 WRONG_NODE:
    002B 0C          176                 CLC                             ; BAD NODE OR GARBLED BYTE
                     177
    002C             178 END_V_C_B:
    002C 32          179                 PULA                            ; RESTORE SAVED NET_BYTE_IN
                     180
    002D 39          181                 RTS
                     182
                     183 ; INPUTS: A = NET_BYTE_IN (HOPEFULLY VERIFIED AS VALID)
                     184 ; OUTPUTS: A = COMMAND CODE FROM NET_BYTE_IN
                     185
                     186                 GLB     ISOLATE_CMND
                     187
    002E             188 ISOLATE_CMND:
    002E 84F0        189                 ANDA    #RESP_MASK              ;       mask in the command nibble
                     190                                                 ;3    SHIFT NET_BYTE_IN COMMAND INTO LOWER NIBBLE
    0030 44          191                 LSRA
    0031 44          192                 LSRA
    0032 44          193                 LSRA
    0033 44          194                 LSRA
                     195
    0034 39          196                 RTS

Errors=    0
```

LINE#	SYMBOL	TYPE	REFERENCES
92	ADDR_MASK	A	166
178	END_V_C_B	P	173
95	EXPECTED_NODE	E	167
140	GOOD_DATA	P	130
127	GOT_DATA	P	123
132	HAVE_ORFE	E	134
188	ISOLATE_CMND	P	186
115	LOOP	P	125
113	MMR_TR_REC	P	110
132	M_SIG	E	135,150
93	RESP_MASK	A	189
	SCI_RX	A	128
	SCI_TR_CS	A	122
111	START_TIMER	A	114
145	TIME_OUT	P	117
147	TIME_OUT_ERR	E	149
164	VERIFY_CMND	P	162
175	WRONG_NODE	P	168

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      1 ^6801^
                      3  NAME ^Rev 04 - RPD^
                      4
                      5 De_MMR_TR_TRANS MACRO              ;Header Rev. 4
                      6                 .GOTO Ede_MMR_TR_TRANS
                      7
                      8  Project:      NET, 83-101
                      9
                     10  ************************************************************
                     11  *                                                          *
                     12  *     MMR_TR_TRANS                    DLS                  *
                     13  *                                                          *
                     14  ************************************************************
                     15
                     16        Rev History
                     17        Rev.  Date          Name      Change
                     18         4    19jul1930p    RPD       removed clear RDRF, use MMR_TR_TCU instead
                     19         3    18jul1945p    RPD       added code to clear received data while xmitting
                     20         2    13jul0234     DLS       CUSTOMIZED FOR MASTER
                     21         1    13jul0835a    RPD       converted pseudo code to 6801 code
                     22         0    12JUL1236     DLS       Initial Pseudo code
                     23
                     24 Ede_MMR_TR_TRANS  MEND
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      26 ****************************************************************************
                      27 *                                                                          *
                      28 *  MODULE NAME:                                                            *
                      29 *                                                                          *
                      30 *    MMR_TR_TRANS                                                          *
                      31 *                                                                          *
                      32 *  INPUTS:                                                                 *
                      33 *                                                                          *
                      34 *    NET_BYTE_OUT (REG_A)                                                  *
                      35 *                                                                          *
                      36 *  FUNCTION(S):                                                            *
                      37 *                                                                          *
                      38 *     1. TO SEND A BYTE OUT OVER THE NETWORK.                              *
                      39 *                                                                          *
                      40 *  OUTPUTS:                                                                *
                      41 *                                                                          *
                      42 *     NET_BYTE_OUT (LOCATION 13)                                           *
                      43 *     CARRY SET = OK CLR = TIMEOUT                                         *
                      44 *  CALLS:                                                                  *
                      45 *                                                                          *
                      46 *    NONE.                                                                 *
                      47 *                                                                          *
                      48 *  CALLED BY:                                                              *
                      49 *                                                                          *
                      50 *    MMR_ACM_SEQ                                                           *
                      51 *    MMR_NIM_READ                                                          *
                      52 *                                                                          *
                      53 *  NOTES:                                                                  *
                      54 *                                                                          *
                      55 *    NONE.                                                                 *
                      56 *                                                                          *
                      57 ****************************************************************************
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      59 ************************************************************************
                      60 *                                                                      *
                      61 *  PSEUDO CODE:                                                        *
                      62 *                                                                      *
                      63 *       MMR_TR_TRANS:                                                  *
                      64 *                                                                      *
                      65 *         REPEAT_UNTIL_SET:                                            *
                      66 *                                                                      *
                      67 *              IF MEM(11).5=0 THEN GOTO REPEAT_UNTIL_SET;              *
                      68 *              ENDIF;                                                  *
                      69 *                                                                      *
                      70 *              MEM(13)=REG_A;                                          *
                      71 *                                                                      *
                      72 *        RETURN;                                                       *
                      73 *                                                                      *
                      74 ************************************************************************
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     76                 INCLUDE PG0_EQU
                      + ;
                      + ; 6801 internal register equates (page 0)
                      + ;
(0000)                + P1_DIR           EQU     000H                 ;port 1 data direction register
(0002)                + P1_DATA          EQU     002H                 ;port 1 data register
                      +
(0001)                + P2_DIR           EQU     001H                 ;port 2 data direction register
(0003)                + P2_DATA          EQU     003H                 ;port 2 data register
                      +
(0004)                + P3_DIR           EQU     004H                 ;port 3 data direction register
(0006)                + P3_DATA          EQU     006H                 ;port 3 data register
                      +
(0005)                + P4_DIR           EQU     005H                 ;port 4 data direction register
(0007)                + P4_DATA          EQU     007H                 ;port 4 data register
                      +
(0008)                + T_CNTLSTAT       EQU     008H                 ;timer control and status register
(0009)                + T_CNTRHGH        EQU     009H                 ;counter high byte
(000A)                + T_CNTRLOW        EQU     00AH                 ;counter low byte
(000B)                + T_OCMPHGH        EQU     00BH                 ;output compare register high byte
(000C)                + T_OCMPLOW        EQU     00CH                 ;output compare register low byte
(000D)                + T_ICAPHGH        EQU     00DH                 ;input capture register high byte
(000E)                + T_ICAPLOW        EQU     00EH                 ;input capture register low byte
                      +
(000F)                + P3_CNTLSTAT      EQU     00FH                 ;port 3 control and status register
                      +
(0010)                + SCI_RM           EQU     010H                 ;rate and mode control register
(0011)                + SCI_TR_CS        EQU     011H                 ;transmit/receive control and status register
(0012)                + SCI_RX           EQU     012H                 ;receive data register
(0013)                + SCI_TX           EQU     013H                 ;transmit data register
                      +
(0014)                + RAM_CNTL         EQU     014H                 ;RAM control register
                     77
                     78 ;
                     79 ; local equates
                     80 ;
(0020)               81 TDRE_MASK       EQU     020H                 ;"transmit_data_register_empty" mask
(0660)               82 TIMER           EQU     0660H                ;10MS
                     83
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      85                PROG
                      86                EXT     M_SIG
                      87                GLB     MMR_TR_TRANS
     0000             88 MMR_TR_TRANS:                        ;T
     0000 CE0660      89                LDX     #TIMER          3
     0003             90 REPEAT:                                      ;1 REPEAT
                      91                                              ;2   TDRE = SCI_TR_CS .AND. TDRE_MASK
     0003 09          92                DEX                     3
     0004 270A        93                BEQ     T01             3
                      94 *
     0006 D611        95                LDAB    SCI_TR_CS,D     3       ;       get the control/status byte
     0008 C420        96                ANDB    #TDRE_MASK      2       ;       mask in the TDRE bit
     000A 27F7        97                BEQ     REPEAT          3       ;1 UNTIL TDRE = TRUE
                      98
     000C 9713        99                STAA    SCI_TX,D        3       ;1 SCI_TX = NEXT_BYTE_OUT
                     100
                     101                IF      0
                     102
                     103                LDX     #TIMER
                     104 LOOP:
                     105                DEX
                     106                BEQ     T02
                     107 *
                     108                LDAB    SCI_TR_CS,D
                     109                ANDB    #TDRE_MASK
                     110                BNE     LOOP
                     111
                     112                ENDIF
                     113
     000E 0D         114                SEC                     2
     000F 39         115                RTS                     5
                     116                        ;*TOTAL  27
          (0010)     117 T01           EQU     $
                     118                EXT     TRANS_T01
     0010 8600       119                LDAA    #TRANS_T01  ;ERR CODE FOR TRANS TIME OUT
     0012 9700       120                STAA    M_SIG,D
     0014 0C         121                CLC
     0015 39         122                RTS
                     123
     0016            124 T02:
                     125                EXT     TRANS_T02
     0016 8600       126                LDAA    #TRANS_T02
     0018 9700       127                STAA    M_SIG,D
     001A 0C         128                CLC
     001B 39         129                RTS
                     130

Errors=    0
```

LINE#	SYMBOL	TYPE	REFERENCES
88	MMR_TR_TRANS	P	87
86	M_SIG	E	120,127
90	REPEAT	P	97
	SCI_TR_CS	A	95
	SCI_TX	A	99
81	TDRE_MASK	A	96
82	TIMER	A	89
117	T01	P	93
124	T02	P	
118	TRANS_T01	E	119
125	TRANS_T02	E	126

LOCATION OBJECT CODE LINE SOURCE LINE

```
^6801^
 NAME ^Rev 00 - RPD^

De_MMR_TR_TCU MACRO               ;Header Rev. 4
                .GOTO Ede_MMR_TR_TCU

 Project:       NET, 83-101

 ******************************
 *                            *
 *    MMR_TR_TCU          RPD *
 *                            *
 ******************************

       Rev History
       Rev.  Date        Name      Change

        0    19jul535p   RPD       Initial Pseudo code and code

Ede_MMR_TR_TCU  MEND
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      23 ****************************************************************
                      24 *                                                              *
                      25 *  MODULE NAME:                                                *
                      26 *                                                              *
                      27 *    MMR_TR_TCU (transmit clean up)                            *
                      28 *                                                              *
                      29 *  INPUTS:                                                     *
                      30 *                                                              *
                      31 *    none                                                      *
                      32 *                                                              *
                      33 *                                                              *
                      34 *  FUNCTION(S):                                                *
                      35 *                                                              *
                      36 *     1. Clears the "receive data register full" flag of the   *
                      37 *        6801 SCI after a transmission sequence (1 or more     *
                      38 *        bytes). The flag is set as a result of sending a byte *
                      39 *        out and receiving the same byte in on the common NET  *
                      40 *        line used for sending and receiving.                  *
                      41 *                                                              *
                      42 *  OUTPUTS:                                                    *
                      43 *                                                              *
                      44 *     SCI control/status register bit 7 = 0                    *
                      45 *                                                              *
                      46 *  CALLS:                                                      *
                      47 *                                                              *
                      48 *    none                                                      *
                      49 *                                                              *
                      50 *  CALLED BY:                                                  *
                      51 *                                                              *
                      52 *    MMR_ACM_R                                                 *
                      53 *    Call routines calling MMR_TR_TRANS)                       *
                      54 *                                                              *
                      55 *  NOTES:                                                      *
                      56 *     1 - This sequence follows the procedure described in     *
                      57 *         hardware manuals for clearing the flag. Which is:    *
                      58 *           step 1) read the SCI control status register       *
                      59 *           step 2) read the SCI receive data register         *
                      60 *     2 - The MAC modules are responsible for calling this     *
                      61 *         module after doing a transmit function to avoid      *
                      62 *         reading itself when other data is expected.          *
                      63 *                                                              *
                      64 ****************************************************************
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      66 ************************************************************
                      67 *                                                          *
                      68 *  PSEUDO CODE:                                            *
                      69 *                                                          *
                      70 *    begin                                                 *
                      71 *      wait for TDRE = 1                                   *
                      72 *      clear RDRF (from 2nd to the last byte)              *
                      73 *      wait for RDRF = 1                                   *
                      74 *      read in the received byte (from very last byte)     *
                      75 *    end                                                   *
                      76 *                                                          *
                      77 ************************************************************
                      78
                      79 ;
                      80 ; local equate
                      81 ;
           (0011)     82 SCI_TR_CS      EQU     011H
           (0012)     83 SCI_RX         EQU     012H
           (0020)     84 TDRE_MASK      EQU     020H
           (0294)     85 TIMER          EQU     0660      ;10MS
                      86
                      87                PROG
                      88                GLB     MMR_TR_TCU
0000                  89 MMR_TR_TCU:
                      90
0000 CE0294           91                LDX     #TIMER
0003                  92 REPEAT:
0003 09               93                DEX
0004 2716             94                BEQ     TO1
                      95 *
0006 D611             96                LDAB    SCI_TR_CS,D
0008 C420             97                ANDB    #TDRE_MASK
000A 27F7             98                BEQ     REPEAT
                      99
000C D612            100                LDAB    SCI_RX,D              ;reset RDRF from 2nd to last byte
                     101
000E CE0294          102                LDX     #TIMER
0011                 103 REPEAT1:
0011 09              104                DEX
0012 270E            105                BEQ     TO2
                     106 *
0014 D611            107                LDAB    SCI_TR_CS,D           ;1 WAIT FOR RECEIVE DATA REGISTER FULL
0016 2AF9            108                BPL     REPEAT1
                     109
0018 D612            110                LDAB    SCI_RX,D              ;1 EMPTY RECEIVED DATA REGISTER AND CLEAR RDRF BIT
                     111                                              ;reset RDRF from last byte
001A 0D              112                SEC
001B 39              113                RTS
                     114 *
           (001C)    115 TO1           EQU     $
                     116
                     117                EXT     TCU_ERR1,M_SIG
                     118
001C 8600            119                LDAA    #TCU_ERR1
001E    10           120                STAA    M_SIG,D
                     121
0020 0C              122                CLC
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    0021 39           123               RTS
                      124
    0022              125 T02:
                      126               EXT     TCU_ERR2
                      127
    0022 8600         128               LDAA    #TCU_ERR2
    0024 9700         129               STAA    M_SIG,D
                      130
    0026 0C           131               CLC
    0027 39           132               RTS
                      133

Errors=    0
```

LINE#	SYMBOL	TYPE	REFERENCES
89	MMR_TR_TCU	P	88
117	M_SIG	E	120,129
92	REPEAT	P	98
103	REPEAT1	P	108
83	SCI_RX	A	100,110
82	SCI_TR_CS	A	96,107
117	TCU_ERR1	E	119
126	TCU_ERR2	E	128
84	TDRE_MASK	A	97
85	TIMER	A	91,102
115	T01	P	94
125	T02	P	105

FILE/PROG NAME	PROGRAM	DATA	COMMON	ABSOLUTE	DATE	TIME	COMMENTS
MMR_DMA_R:pADAMM	F800				Thu, 23 Jun 1983,	0:28	Rev 00 - DLS
MMR_DMA_W:pADAMM	F833			008E-008F	Thu, 23 Jun 1983,	0:29	Rev 00 - DLS
next address	F865						
MASTER_AP:pADAMM	F865		00F9	0097-00C9 FFF0-FFFF	Thu, 23 Jun 1983,	0:17	Rev 00 - WRB
MMR_MAC:pADAMM	FBE7	0080			Thu, 23 Jun 1983,	0:31	Rev 00 - RPD
D_MMR:pADAMM			00F9	0090-0096	Thu, 23 Jun 1983,	0:15	Rev 02 - RPD
MMR_TR_RE:pADAMM	FEBE				Thu, 23 Jun 1983,	0:39	Rev 03 - RPD
MMR_TR_TR:pADAMM	FEF3				Thu, 23 Jun 1983,	0:43	Rev 04 - RPD
MMR_TR_TC:pADAMM	FF0F				Thu, 23 Jun 1983,	0:41	Rev 00 - RPD
next address	FF37	0003	0100				

XFER address= 0000 Defined by DEFAULT
absolute & link_com file name=MAA:pADAMM
Total# of bytes loaded= 078D

SYMBOL	R	VALUE	DEF BY	REFERENCES
A_SIG	C	00FE	D_MMR:pADAMM	
A_SIGNAL	C	00FE	D_MMR:pADAMM	
CNFG_WORD	C	00F9	D_MMR:pADAMM	
D1_MODE_WORD	A	0090	D_MMR:pADAMM	
D_MMR	D	0083	D_MMR:pADAMM	
EXPECTED_NODE	A	0091	D_MMR:pADAMM	MMR_TR_RE:pADAMM MMR_MAC:pADAMM
HAVE_ORFE	A	009A	MMR_MAC:pADAMM	MMR_TR_RE:pADAMM
INDEXTOMEM	A	0096	D_MMR:pADAMM	
ISOLATE_CMND	P	FEEC	MMR_TR_RE:pADAMM	MMR_MAC:pADAMM
MEM_LEN	C	00FC	D_MMR:pADAMM	
MEM_LEN_HI	C	00FC	D_MMR:pADAMM	
MEM_LEN_LO	C	00FD	D_MMR:pADAMM	
MEM_PTR	C	00FA	D_MMR:pADAMM	
MEM_PTR_HI	C	00FA	D_MMR:pADAMM	
MEM_PTR_LO	C	00FB	D_MMR:pADAMM	
MMR_DMA_READ	P	F800	MMR_DMA_R:pADAMM	MMR_MAC:pADAMM MASTER_AP:pADAMM
MMR_DMA_WRITE	P	F833	MMR_DMA_W:pADAMM	MMR_MAC:pADAMM MASTER_AP:pADAMM
MMR_MAC	P	FBE7	MMR_MAC:pADAMM	MASTER_AP:pADAMM
MMR_TR_REC	P	FEBE	MMR_TR_RE:pADAMM	MMR_MAC:pADAMM
MMR_TR_TCU	P	FF0F	MMR_TR_TC:pADAMM	MMR_MAC:pADAMM
MMR_TR_TRANS	P	FEF3	MMR_TR_TR:pADAMM	MMR_MAC:pADAMM
M_SIG	C	00FF	D_MMR:pADAMM	MMR_TR_TC:pADAMM MMR_TR_TR:pADAMM MMR_TR_RE:pADAMM
M_SIGNAL	C	00FF	D_MMR:pADAMM	
NIM_DMA_BLK	C	00F9	D_MMR:pADAMM	
NXT_STATE	A	0092	D_MMR:pADAMM	
SA	P	F8A6	MASTER_AP:pADAMM	
STACK_TEMP	A	00BE	MMR_DMA_W:pADAMM	MMR_MAC:pADAMM MMR_DMA_R:pADAMM
TCU_ERR1	A	0096	MMR_MAC:pADAMM	MMR_TR_TC:pADAMM
TCU_ERR2	A	0097	MMR_MAC:pADAMM	MMR_TR_TC:pADAMM
TIMEFORRESPONSE	A	0093	D_MMR:pADAMM	
TIMER	A	0094	D_MMR:pADAMM	
TIME_OUT_ERR	A	009B	MMR_MAC:pADAMM	MMR_TR_RE:pADAMM
TRANS_TO1	A	0098	MMR_MAC:pADAMM	MMR_TR_TR:pADAMM
TRANS_TO2	A	0099	MMR_MAC:pADAMM	MMR_TR_TR:pADAMM
VERIFY_CMND	P	FEDF	MMR_TR_RE:pADAMM	MMR_MAC:pADAMM