| 0570 30 | 925 | DB | 30H |
| 0570 30 | 926 | DB | 30H |
| 0570 30 | 927 | DB | 30H |
| 0581 30 | 929 | DB | 30H |
| 0582 30 | 930 | DB | 30H |
| 0583 30 | 931 | DB | 30H |
| 0584 30 | 932 | DB | 30H |
| 0585 30 | 933 | DB | 30H |
| 0586 30 | 934 | DB | 30H |
| 0587 30 | 935 | DB | 30H |
| 0588 30 | 936 | DB | 30H |
| 0589 30 | 937 | DB | 30H |
| 058A 30 | 938 | DB | 30H |
| 058B 30 | 939 | DB | 30H |
| 058C 30 | 940 | DB | 30H |
| 058D 30 | 941 | DB | 30H |
| 058E 30 | 942 | DB | 30H |
| 058F 30 | 943 | DB | 30H |
| 0590 1A | 944 | DB | 1AH |
| 0591 1B | 945 | DB | 18H |
| 0592 0D | 946 | DB | 0DH |
| 0593 16 | 947 | DB | 16H |
| 0594 30 | 948 | DB | 30H |
| 0595 03 | 949 | DB | 03H |
| 0596 08 | 950 | DB | 08H |
| 0597 02 | 951 | DB | 02H |
| 0598 30 | 952 | DB | 30H |
| 0599 1F | 953 | DB | 1FH |
| 059A 0F | 954 | DB | 0FH |
| 059B 0C | 955 | DB | 0CH |
| 059C 30 | 956 | DB | 30H |
| 059D 30 | 957 | DB | 30H |
| 059E 10 | 958 | DB | 10H |
| 059F 30 | 959 | DB | 30H |
| 05A0 13 | 95A | DB | 13H |
| 05A1 04 | 95B | DB | 04H |
| 05A2 0B | 95C | DB | 0BH |
| 05A3 07 | 95D | DB | 07H |
| 05A4 01 | 95E | DB | 01H |
| 05A5 06 | 95F | DB | 06H |
| 05A6 0A | 960 | DB | 0AH |
| 05A7 05 | 961 | DB | 05H |
| 05A8 17 | 962 | DB | 17H |
| 05A9 05 | 963 | DB | 05H |
| 05AA 09 | 964 | DB | 09H |
| 05AB 14 | 965 | DB | 14H |
| 05AC 11 | 966 | DB | 11H |
| 05AD 12 | 967 | DB | 12H |
| 05AE 15 | 968 | DB | 15H |
| 05AF 10 | 969 | DB | 10H |
| 05B0 30 | 970 | DB | 30H |
| 05B1 30 | 971 | DB | 30H |
| 05B2 30 | 972 | DB | 30H |
| 05B3 30 | 973 | DB | 30H |
| 05B4 30 | 974 | DB | 30H |
| 05B5 30 | 975 | DB | 30H |
| 05B6 30 | 976 | DB | 30H |
| 05B7 30 | 977 | DB | 30H |
| 05B8 30 | 978 | DB | 30H |
| 05B9 30 | 979 | DB | 30H |
| 05BA 30 | 980 | DB | 30H |
| 05BB 30 | 981 | DB | 30H |
| 05BC 30 | 982 | DB | 30H |
| 05BD 30 | 983 | DB | 30H |
| 05BE 30 | 984 | DB | 30H |
| 05BF 30 | 985 | DB | 30H |
| 05C0 30 | 986 | DB | 30H |
| 05C1 30 | 987 | DB | 30H |
| 05C2 30 | 988 | DB | 30H |
| 05C3 30 | 989 | DB | 30H |
| 05C4 30 | 990 | DB | 30H |
| 05C5 30 | 991 | DB | 30H |
| 05C6 30 | 992 | DB | 30H |
| 05C7 30 | 993 | DB | 30H |
| 05C8 30 | 994 | DB | 30H |
| 05C9 30 | 995 | DB | 30H |
| 05CA 30 | 996 | DB | 30H |
| 05CB 30 | 997 | DB | 30H |
| 05CC 30 | 998 | DB | 30H |
| 05CD 30 | 999 | DB | 30H |
| 05CE 30 | 1000 | DB | 30H |
| 05CF 30 | 1001 | DB | 30H |
| 05D0 30 | 1002 | DB | 30H |
| 05D1 30 | 1003 | DB | 30H |
| 05D2 1D | 1004 | DB | 1DH |
| 05D3 30 | 1005 | DB | 30H |
| 05D4 30 | 1006 | DB | 30H |
| 05D5 30 | 1007 | DB | 30H |
| 05D6 30 | 1008 | DB | 30H |
| 05D7 30 | 1009 | DB | 30H |
| 05D8 30 | 1010 | DB | 10H |
| 05D9 30 | 1011 | DB | 30H |
| 05DA 30 | 1012 | DB | 30H |
| 05DB 30 | 1013 | DB | 30H |

SUB Subroutine
CAN Cancel
CR Carriage Return
SYN Synch
FTX Form Text
STX Start of Text
US US
SI SI
FF FF
DLE DLE
DC3 DC3
EOT End Of Text
VT Vertical Tab
BEL Bell
SBH-sub
ACK Acknowledge
LF Line Feed
BS Back Space
ETB End Text Block
ENQ Enquiry
HT Horizontal Tab
DC4 DCS
DC1 DC1
DC2 DC2
NAK NAK
EM EM

; GS
048F 70 711  MOV  A, L ; MOVE LOW BYTE OF CH 2 TC TO A
0490 DA 7D 12  OUT  P2 TC ; OUTPUT CH 2 TC TO 8257
0492 7C 13  MOV  A, H ; MOVE UP BYTE OF CH 2 TC TO A
0493 DA 7E 14  OUT  P2 TC ; OUTPUT CH 2 TC TO 8257
0498 75 15  ;
0499 27 0E 16  LXI  H, 5000H ; LOAD 5000H IN HL
049A DA 7F 17  MOV  A, L ; MOVE LOW BYTE OF CH 3 START ADD TO A
049B 7C 18  MOV  P3 SA ; OUTPUT CH 3 START ADD TO 8257
049C DA 80 19  MOV  A, H ; MOVE UP BYTE OF CH 3 START ADD TO A
049D 7D 20  OUT  P2 TC ; OUTPUT CH 3 START ADD TO 8257
049E 21 CF 07 21  ;
04A1 7D 22  MOV  A, L ; LOAD CH 3 TC VALUE IN HL
04A4 DA 7E 23  MOV  P3 TC ; OUTPUT CH 3 TC TO 8257
04A5 7C 24  MOV  A, H ; MOVE H TO A
04A6 DA 7F 25  OUT  P2 TC ; OUTPUT CH 3 TC TO 8257
04A7 8F 26  HVL  A, DS 57 ; LOAD A WITH MODE SET VALUE
04A8 DA 7E 27  OUT  PM 57 ; OUTPUT MODE SET TO 8257
04A9 7D 28  ;
04A9 2A 8F 29  ; KEYBOARD POLLING ROUTINE
04AB 2A 8F 30  ;
04A9 2A 8F 31  ;
04AB DE 6F 32  ; KPOLL: IN KCOM ; INPUT FIFO STATUS
04AD 07 00 33  ANI  0FH ; MASK STATUS, SAVE BITS 0-2
04AE 2A 50 34  JZ  ZIP ; TEST FOR CHARACTER PRESENT
04B2 2A 60 35  CALL  XMIT ; CALL CHARACTER TRANSMIT ROUTINE
04B5 C9 36  ZIP: RET ; RETURN
04B9 79 37  ; CHARACTER TRANSMIT SUBROUTINE
04B9 79 38  ;
04B9 79 39  XMIT: IN KDAT ; INPUT FIFO CHARACTER
04BB 7D 3A 3A  EED ; INVENT TOP 2 BITS
04BD 10 00 3B 3B 21F004 ; LOAD BASE ADD OF TABLE 3 IN H & L
04BE 07 00 3C 07  LXI  H, 6000H ; LOAD E WITH CHARACTER FROM FIFO
04C0 5F 3D 3D  0000H ; CALCULATE ADD IN LOOKUP TABLE
04C1 19 3E 3E  DAD  D ; CONTAINING ASCII CHAR.
04C2 2F 3F 3F  67F ; CORRESPONDING TO KEY POSITION IN MATRIX
04C4 7C 40 40  DAD  D ; INPUT USB STATUS
04C6 66 41 41  601 ; MASK STATUS, SAVE TRANSMITTER READY BIT
04C8 7C 42 42  DAD  D ; TEST READY BIT
04CA 66 43 43  601 ; MOVE ASCII CHAR TO A
04CC 2F 44 44  67F ; MASK BIT 7
04CE C9 45 45  RET ; OUTPUT CHAR FROM USB
04CF C9 46 46  DUMMY: RET ; RETURN
04DE 78 47 47  ; DUMMY ROUTINE DEFINITION
04D2 78 48 48  ;
04D8 78 49 49  ; TABLE DEFINITION AREA
04D0 CF 04 4A 4A  BSET1: DW DUMMY
04D2 39 01 4B 4B  DW ESCA
04D4 50 01 4C 4C  DW ESCB
04D6 7D 01 4D 4D  DW ESCC
04D8 80 01 4E 4E  DW ESCD
04DA 35 02 4F 4F  DW ESCE
04DC CF 04 50 50  DW DUMMY
04DE CF 04 51 51  DW DUMMY
04E0 7B 01 52 52  DW ESCF
04E2 CF 04 53 53  DW DUMMY
04E4 66 02 54 54  DW ESCG
04E6 FD 01 55 55  DW ESCH
04E8 CF 04 56 56  DW DUMMY
04EA CF 04 57 57  DW DUMMY
04EC CF 04 58 58  DW DUMMY
04EE CF 04 59 59  DW DUMMY
04F0 05 03 5A 5A  BSET2: DW CTRLH
04F2 05 02 5B 5B  DW CTRLA
04F4 PC 02 5C 5C  DW CTRLM
04F6 CF 04 5D 5D  DW DUMMY
04F8 30 5E 5E  BSET3: DB 30H ; DUMMY CHARACTER
04F9 30 5F 5F  DB 30H
04FA 30 60 60  DB 30H
04FB 30 61 61  DB 30H
04FC 30 62 62  DB 30H
04FD 30 63 63  DB 30H
04FE 30 64 64  DB 30H
04FF 30 65 65  DB 30H

53
<table>
<thead>
<tr>
<th>Address</th>
<th>Opcode</th>
<th>Instruction</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>042D 19</td>
<td>621</td>
<td>DAD D</td>
<td>ADD 80H-50H TO TOP</td>
</tr>
<tr>
<td>042E 22D687</td>
<td>622</td>
<td>SDL T</td>
<td>STORE MODIFIED T0PAD IN MEMORY</td>
</tr>
<tr>
<td>0431 C9</td>
<td>623</td>
<td>RET</td>
<td>RETURN</td>
</tr>
<tr>
<td>0432 2AE287</td>
<td>627</td>
<td>FILL:</td>
<td></td>
</tr>
<tr>
<td>0435 115000</td>
<td>628</td>
<td>LDL D50H</td>
<td></td>
</tr>
<tr>
<td>0436 19</td>
<td>630</td>
<td>DAD B</td>
<td></td>
</tr>
<tr>
<td>0439 22DC87</td>
<td>631</td>
<td>SLD T</td>
<td></td>
</tr>
<tr>
<td>043B 012020</td>
<td>632</td>
<td>LDL B80H</td>
<td></td>
</tr>
<tr>
<td>043B 100000</td>
<td>633</td>
<td>LDL B5000H</td>
<td></td>
</tr>
<tr>
<td>043C 39</td>
<td>634</td>
<td>DAD SP</td>
<td></td>
</tr>
<tr>
<td>043D 9B</td>
<td>635</td>
<td>XCHG</td>
<td></td>
</tr>
<tr>
<td>043E 24DC87</td>
<td>636</td>
<td>LDL LOC80</td>
<td></td>
</tr>
<tr>
<td>044F 9D</td>
<td>637</td>
<td>PHL</td>
<td></td>
</tr>
<tr>
<td>044F C5</td>
<td>638</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F C6</td>
<td>639</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F C7</td>
<td>640</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F C8</td>
<td>641</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F C9</td>
<td>642</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F CA</td>
<td>643</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F CB</td>
<td>644</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F CC</td>
<td>645</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F CD</td>
<td>646</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F CE</td>
<td>647</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F CF</td>
<td>648</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F D0</td>
<td>649</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F D1</td>
<td>650</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F D2</td>
<td>651</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F D3</td>
<td>652</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F D4</td>
<td>653</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F D5</td>
<td>654</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F D6</td>
<td>655</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F D7</td>
<td>656</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F D8</td>
<td>657</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F D9</td>
<td>658</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F DA</td>
<td>659</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F DB</td>
<td>660</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F DC</td>
<td>661</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F DD</td>
<td>662</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F DE</td>
<td>663</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F DF</td>
<td>664</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F E0</td>
<td>665</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F E1</td>
<td>666</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F E2</td>
<td>667</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F E3</td>
<td>668</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F E4</td>
<td>669</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F E5</td>
<td>670</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F E6</td>
<td>671</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F E7</td>
<td>672</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F E8</td>
<td>673</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F E9</td>
<td>674</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F EA</td>
<td>675</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F EB</td>
<td>676</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F EC</td>
<td>677</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F ED</td>
<td>678</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F EE</td>
<td>679</td>
<td>PUS B</td>
<td></td>
</tr>
<tr>
<td>044F EF</td>
<td>680</td>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>044F F0</td>
<td>681</td>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>044F F1</td>
<td>682</td>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>044F F2</td>
<td>683</td>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>044F F3</td>
<td>684</td>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>044F F4</td>
<td>685</td>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>044F F5</td>
<td>686</td>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>044F F6</td>
<td>687</td>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>044F F7</td>
<td>688</td>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>044F F8</td>
<td>689</td>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>0473 3E0D</td>
<td>590</td>
<td>RT7S:</td>
<td></td>
</tr>
<tr>
<td>0475 D348</td>
<td>591</td>
<td>MVI A3MDC57</td>
<td></td>
</tr>
<tr>
<td>0477 2AD687</td>
<td>592</td>
<td>OUT PMO57</td>
<td></td>
</tr>
<tr>
<td>047A 7D</td>
<td>593</td>
<td>HLD T0PAD</td>
<td></td>
</tr>
<tr>
<td>047B 344</td>
<td>594</td>
<td>MOV A L</td>
<td></td>
</tr>
<tr>
<td>047B 7C</td>
<td>595</td>
<td>OUT PC2SA</td>
<td></td>
</tr>
<tr>
<td>047B 0344</td>
<td>596</td>
<td>MOV A H</td>
<td></td>
</tr>
<tr>
<td>047B 0344</td>
<td>597</td>
<td>OUT PC2SA</td>
<td></td>
</tr>
<tr>
<td>0480 7D</td>
<td>598</td>
<td>MOV A L</td>
<td></td>
</tr>
<tr>
<td>0481 2F</td>
<td>599</td>
<td>CMA</td>
<td></td>
</tr>
<tr>
<td>0482 6F</td>
<td>600</td>
<td>MOV L A</td>
<td></td>
</tr>
<tr>
<td>0483 7C</td>
<td>601</td>
<td>MOV A H</td>
<td></td>
</tr>
<tr>
<td>0483 2F</td>
<td>602</td>
<td>CMA</td>
<td></td>
</tr>
<tr>
<td>0485 67</td>
<td>603</td>
<td>MOV H A</td>
<td></td>
</tr>
<tr>
<td>0485 2F</td>
<td>604</td>
<td>CMA</td>
<td></td>
</tr>
<tr>
<td>0486 23</td>
<td>605</td>
<td>INC H</td>
<td></td>
</tr>
<tr>
<td>0487 11CF87</td>
<td>606</td>
<td>LXI D87CFH</td>
<td></td>
</tr>
<tr>
<td>0488 18</td>
<td>607</td>
<td>DAD D</td>
<td></td>
</tr>
<tr>
<td>0488 D080</td>
<td>608</td>
<td>DAD D</td>
<td></td>
</tr>
<tr>
<td>0488 18</td>
<td>609</td>
<td>DAD D</td>
<td></td>
</tr>
</tbody>
</table>
`CMP H
COMPARE H WITH 87H`

`:JNC NXTCM
IF NO CARRY, CONTINUE COMPARISON`

`:CALL COMRT
IF CARRY, CALL COMPARISON ROUTINE`

`:JUMP TO XSTAD
JUMP TO XSTAD`

`:IF NOT EQNZ, END COMPARISON
IF NOT EQUAL, END COMPARISON`

`:MOV A, OCCH
LOAD OCCH IN A`

`:CMP L, OCCH
COMPARE L WITH OCCH`

`:JNC XSTAD
IF NO CARRY, LOCATION LESS THAN
OR EQUAL TO 87CH`

`:CALL COMRT
CALL COMPARISON ROUTINE`

`:CALL END OF ROW CHAR TEST ROUTINE
CALL END OF ROW CHAR TEST ROUTINE`

`:LOAD UART CHAR ADD IN H&L
LOAD UART CHAR ADDRESS IN H&L`

`:MOVE UART CHAR TO A
MOVE UART CHAR TO A`

`:PHI
MASK OFF UPPER 2 BITS OF CHAR`

`:LOAD LOCATION IN H&L
LOAD LOCATION IN H&L`

`:MOVE CHARACTER TO CHARACTER
MOVE CHARACTER TO CHARACTER`

`:LOCATION IN DISPLAY MEMORY
LOCATION IN DISPLAY MEMORY`

`:RET
RETURN`

`:DISA A, CCTAD
LOAD COLUMN COUNT ADD IN H&L`

`:INCR M
INCREMENT COLUMN COUNT`

`:CALL WP75
CALL LOAD CURSOR POSITION SUBROUTINE`

`:RET
RETURN`

`:DISP:
SUBROUTINE DISP`

`:DISC:
SUBROUTINE DISC`

`:STA A, OCCH
ZERO A`

`:ZERO COLUMN COUNT
ZERO COLUMN COUNT`

`:LHLD RCITAD
LOAD ROWCOUNT IN H&L`

`:LXI D, 50H
LOAD 80D=50H IN D&H`

`:ADD +80 TO ROWCOUNT
ADD +80 TO ROWCOUNT`

`:STORE ROWCOUNT IN MEMORY
STORE ROWCOUNT IN MEMORY`

`:LOAD CURSOR Y POSITION ADDRESS IN H&L
LOAD CURSOR Y POSITION ADDRESS IN H&L`

`:INCR CURSY
INCREMENT CURSOR Y`

`:INCR CURSOR Y POSITION
INCREMENT CURSOR Y POSITION`

`:CALL WP75
CALL LOAD CURSOR POSITION SUBROUTINE`

`:RET
RETURN`

`:DISC:
SUBROUTINE DISC`

`:STA A, OCCH
ZERO A`

`:ZERO COLUMN COUNT
ZERO COLUMN COUNT`

`:LHLD RCITAD
LOAD ROWCOUNT IN H&L`

`:LXI D, 50H
LOAD 80D=50H IN D&H`

`:STORE MODIFIED LOCATION IN MEMORY
STORE MODIFIED LOCATION IN MEMORY`

`:LHLD LOCO1
LOAD LOCATION OF FIRST CHAR
IN ROW IN H&L`

`:LXI D, 0F30H
LOAD COMPENSATION VALUE IN H&L`

`:DAD D
ADD COMPENSATION TO LOCO1
ADD COMPENSATION TO LOCO1`

`:LOAD MODIFIED LOC01 IN MEMORY
STORE MODIFIED LOC01 IN MEMORY`

`:RET
RETURN`

`:END OF ROW TEST ROUTINE
END OF ROW TEST ROUTINE`

`:LHLD LOCO1
LOAD LOCATION OF FIRST CHAR
IN ROW IN H&L`

`:LXI D, 0F30H
LOAD COMPENSATION VALUE IN H&L`

`:DAD D
ADD COMPENSATION TO LOCO1
ADD COMPENSATION TO LOCO1`

`:LOAD MODIFIED LOC01 IN MEMORY
STORE MODIFIED LOC01 IN MEMORY`

`:RET
RETURN`

`:SCROLL SUBROUTINE
SCROLL SUBROUTINE`

`:LOAD TOP IN H&L
LOAD TOP IN H&L`

`:CALL FILL
CALL FILL`

`:CALL FILL WITH SPACE CODES SUBROUTINE
CALL FILL WITH SPACE CODES SUBROUTINE`

`:RET
RETURN`
0308 2A0387 445  :ROWUP SUBROUTINE
0308 1B80FF 446  LHLD  RCTAD  :LOAD ROWCOUNT IN H&L
0308 19 447  LXI  D,OFFBOH  :MOVE -80D=OFFBOH (2'S COMP) TO D&E
030F 22D387 448  DAD  D  :ADD -80D TO ROWCOUNT
030F 22D387 449  SHLD  RCTAD  :STORE RESULT IN ROWCOUNT BUFFER
0312 210587 450  LDA  H,CURSY  :LOAD CURSOR Y POINTER ADDRESS IN H&L
0315 35 451  DCR  M  :DECREMENT CURSOR Y POINTER
0316 CD3C03 452  CALL  WP75  :CALL LOAD CURSOR POSITION SUBROUTINE
0319 C9 453  RET  :RETURN

031A 2A0387 457  :ROWDN SUBROUTINE
031D 175050 458  LHLD  RCTAD  :LOAD ROWCOUNT IN H&L
0320 19 459  DAD  D,50H  :MOVE +50D TO D&E
0321 22D387 460  SHLD  RCTAD  :STORE RESULT IN ROWCOUNT
0324 210587 461  LXI  H,CURSY  :LOAD CURSOR Y POINTER ADDRESS IN H&L
0327 34 462  INR  M  :INCREMENT CURSOR Y POINTER
0328 CD3C03 463  CALL  WP75  :CALL LOAD CURSOR POSITION SUBROUTINE
032B C9 464  RET  :RETURN

032C 210287 467  :COLUMN LEFT SUBROUTINE
032F 35 468  DCR  M  :DECREMENT COLUMN COUNT
0330 CD3C03 470  CALL  WP75  :CALL LOAD CURSOR POSITION SUBROUTINE
0333 C9 471  RET  :RETURN

0334 210287 474  :COLUMN RIGHT SUBROUTINE
0337 34 475  INR  M  :INCREMENT COLUMN COUNT
0338 CD3C03 477  CALL  WP75  :CALL LOAD CURSOR POSITION SUBROUTINE
033B C9 478  RET  :RETURN

033C 3E80 481  WP75:  MVI  A,80H  ;LOAD A WITH 80H, LOAD CURSOR POSITION COMMAND
033E 3D5F 482  OUT  CACOM  ;LOAD A WITH CURSOR X POSITION
0340 3D287 483  LDA  CCTAD  ;LOAD COLUMN COUNT ADDRESS IN H&L
0344 3D5E 484  OUT  CDRAT  ;LOAD A WITH CURSOR Y POSITION
0344 3D587 485  LDA  CDRAT  ;LOAD A WITH CURSOR Y POSITION
0346 3D5E 486  OUT  CDRAT  ;LOAD A WITH CURSOR Y POSITION
0348 C9 487  RET  :RETURN

034B 3A0387 489  :DISPLAY CHARACTER HANDLING SUBROUTINE
034E 1F4F 490  CPI  4FH  ;COMPARE BYE WITH 4FH=75D
0350 C5A03 491  JZ  CTA  ;IF BYTE=4FH,COLUMN COUNT=LAST CHAR-
0353 CD7E03 492  CALL  DISI  ;ACTER IN ROW
0355 CD6B03 493  CALL  DISA  ;CALL DISA SUBROUTINE
0358 CD7E03 494  RET  :RETURN

035A 2A0387 496  200A  LHLD  RCTAD  :LOAD ROWCOUNT IN H&L
035D 7D 497  MOV  A,L  ;LOAD LOW BYTE OF ROWCOUNT IN H&L
035E 2E80 498  CPI  80H  ;COMPARE BYTE WITH 80H
035F CAR0A3 499  JZ  CTB  ;IF BYTE=80H,CHANGE ROWCOUNT TO COMPARIS-
0363 CD7E03 500  CALL  DISI  ;CALL DISI SUBROUTINE
0365 CDC03 501  CALL  DISB  ;CALL DISB SUBROUTINE
0368 C9 502  RET  :RETURN

036A 7C 503  CTB:  MOV  A,H  ;MOVE UP BYTE OF ROWCOUNT TO H&L
036B 508  CPI  07H  ;COMPARE BYTE WITH 07H
036C CD7703 509  JZ  CTC  ;IF BYTE=07H,END OF DISPLAY COND EXISTS
036D CD7E03 510  CALL  DISI  ;CALL DISI SUBROUTINE
036F CDC03 511  CALL  DISB  ;CALL DISB SUBROUTINE
0372 C9 512  RET  :RETURN

0374 CD7E03 513  CTC:  CALL  DISI  ;CALL DISI SUBROUTINE
0377 CD7C03 514  CALL  DISC  ;CALL DISC SUBROUTINE
037A CDA03 515  RET  :RETURN

037D C9 516  RET  :RETURN

037E 2A0387 519  :SUBROUTINE DISI
0381 EB 520  XAD  :LOAD TOP IN H&L
0382 CDAD7 521  LHLD  RCTAD  :STORE TOP IN D&E
0385 19 522  DAD  D  ;ADD ROWCOUNT=RESULT IN H&L
0387 CD20A87 523  SHLD  LOC01  ;STORE LOCATION OF FIRST CHAR IN ROW
0389 EE 524  XCHG  ;STORE TOP=ROWCOUNT IN D&E
038A 210030 525  LXI  H,0000H  ;ZERO HAL
038D 3A0387 526  LDA  CCTAD  ;LOAD COLUMN COUNT IN A
0390 6F 527  MOV  L,A  ;MOVE COLUMN COUNT TO L
0391 19 528  DAD  D  ;CALL CALCULATE LOCATION=
0392 22D887 529  SHLD  LOCA  ;STORE LOCATION IN MEMORY
0395 3B7 530  MVI  A,87H  ;LOAD 87H IN A
0396 3B7 531  MVI  A,87H  ;LOAD 87H IN A
0271 3E87 353 VMI A,87H ;LOAD 87H IN A
0273 BC 354 CMP H ;COMPAR H WITH 87H
0274 2D1D02 355 JNC VAR ;IF NO CARRY, CONTINUE COMPARISON
0277 CDE02 356 CALL COMRY ;CALL COMPENSATION ROUTINE
0279 C39902 357 JMP FIN ;JUMP TO FIN
027D CB902 358 VAR: JNZ FIN ;IF NOT EQUAL AND COMPARISON
0282 3E8F 359 VMI A,0CFH ;LOAD CFH IN A
0283 D2902 360 CMP L ;COMPARE L WITH CFH
0283 D2902 361 JNC FIN ;IF NO CARRY, COMPARE LESS THAN OR EQ TO 87CFH
0286 CDE02 362 CALL COMRY ;CALL COMPENSATION ROUTINE
0289 2AD687 363 FIN: LHLD TOPAD ;LOAD TOP IN H.L
028C 7D 364 MOV A,L ;MOVE L TO A
028D FED0 365 CPI 00H ;COMPARE BYTE TO 00H
028F C2A02 366 JNZ TROLL ;IF NO COMPARISON, JUMP TO TROLL
0292 7C 367 MOV A,H ;MOVE H TO A
0293 FED0 368 CPI 00H ;COMPARE BYTE TO 00H
0295 C2A02 369 JNZ TROLL ;IF NO COMPARISON, JUMP TO TROLL
029B 219087 372 LXI R,8780H ;IF COMPARISON, SET B0T=8780H
029B 219087 373 SHLD BOTAD ;IF COMPARISON, SET B0T=8780H
029C C3AB02 374 JMP GNOME ;JUMP TO GNOME
02A1 17B0FF 375 TROLL: LXI D,0FF00H ;LOAD -80D=0FF00H IN D&H
02A4 2AD687 376 LHLD TOPAD ;LOAD TOP IN H.L
02A7 19 377 DAR D ;ADD -50D TO TOP
02AB 226687 378 SHLD BOTAD ;LOAD -50D TO TOP
02AB 3E80 379 GNOME: VMI A,0FOH ;LOAD A WITH EOR CHAR (LOOP START)
02AD 2AE087 380 LHLD LOCPR ;LOAD LOCPR IN H.L
02B0 77 381 MOV M,A ;MOVE EOR CHAR TO MEM
02B1 7D 382 MOV A,L ;MOVE L TO A
02B2 FED0 383 CPI 80H ;COMPARE BYTE WITH 80H
02B4 C2D502 384 JNZ WIZAR ;IF NO COMPARISON, JMP TO WIZAR
02B7 7C 385 MOV A,H ;MOVE H TO A
02B8 FED0 386 CPI 80H ;COMPARE BYTE WITH 80H
02B9 C2D502 387 JNZ WIZAR ;IF NO COMPARISON, JMP TO WIZAR
02BD EB 388 GZONK: XCHG ;STORE PRESENT LOC IN D&H
02BE 2AE087 389 LHLD BOTAD ;LOAD BOT IN H.L
02C1 7D 38A MOV A,L ;MOVE L TO A
02C2 BB 38B CMP A,E ;COMPARE E WITH A
02C3 C2CC02 38C JNZ FUN ;IF NO COMP, JUMP TO FUN
02C6 7C 38D MOV A,H ;MOVE H TO A
02C7 BA 38E CMP A,D ;COMPARE D WITH A
02C8 C2CC02 38F JNZ FUN ;IF NO COMP, JUMP TO FUN
02CB C9 390 JMP GAIR ;IF COMPARISON, RETURN
02CC 210080 391 FUN: LXI H,8000H ;LOAD H.L WITH 8000H
02CF 226687 392 SHLD LOCPR ;SET LOCPR = 8000H
02D2 C3AB02 393 JMP GNOME ;JUMP TO GNOME
02D5 EB 394 GONDO: XCHG ;STORE LOCPR IN D&H
02D6 2AE087 395 LHLD BOTAD ;LOAD BOT IN H.L
02D9 7D 396 MOV A,L ;MOVE L TO A
02DA BB 397 CMP A,E ;COMPARE E WITH A
02DB C2E02 398 JNZ NUF ;IF NO COMP, JMP TO NUF
02DE 7C 399 MOV A,H ;MOVE H TO A
02DF BA 39A CMP A,D ;COMPARE D WITH A
02E0 C2E02 39B JNZ NUF ;IF NO COMP, JMP TO NUF
02E3 C9 39C RET ;RETURN
02E4 215000 39D NUF: LXI H,50EH ;LOAD 80D=50H IN H.L
02E7 19 39E DAR D ;ADD 80D TO LOCPR (LOCPR IN D&H)
02E8 226687 39F SHLD LOCPR ;STORE LOCPR IN MEM
02E9 C3AB02 400 JMP GNOME ;JUMP TO GNOME
02E9 C3AB02 401 ;COMPENSATION SUBROUTINE COMRY
02EE 2AE087 402 COMRY: LHLD LOCPR ;LOAD LOCPR IN H.L
02F1 113F08 403 LXI D,0F030H ;LOAD COM VALUE IN D&H
02F2 19 404 DAR D ;ADD COMPENSATION TO LOCPR
02F3 226687 405 SHLD LOCPR ;STORE LOCPR IN MEM
02F8 C9 406 RET ;RETURN
02F9 C35F01 407 ;LINE FEED ROUTINE
02F9 C35F01 408 CTRLJ: JMP ESCB ;JUMP ESCB
02F9 C35F01 409 ;CARRIAGE RETURN ROUTINE
02FC 3E00 410 CTRLM: VMI A,00H ;ZERO A
02FE 32D287 411 STA CCTAD ;SET COLUMN COUNT=0
0301 C39C03 412 CALL WP75 ;CALL LOAD CURSOR POSITION SUBROUTINE
0304 C9 413 RET ;RETURN
0305 C3B001 414 ;BACK SPACE ROUTINE
0305 C3B001 415 CTRLH: JMP ESCD
JMP CCMWB ; JUMP TO CCMWB
LXI H,0780H ; LOAD hxl WITH ROMCOUNT=780H=1920D
MVI A,4FH ; SET COLUMN COUNT=4FH=79D
STL RCTAD ; CALL ROWOP SUBROUTINE
CALL ROWNP ; RETURN

; HOME ROUTINE
LXI H,0000H ; ZERO hxl
STL RCTAD ; SET ROWCOUNT=0
MVI A,'A' ; ZERO A
MVI A,18H ; SET COLUMN COUNT=0
MVI A,26H ; CALL CURSOR Y POINTER=26H=64D
CALL WP75 ; CALL LOAD CURSOR POSITION SUBROUTINE
RET ; RETURN

; ERASE LINE ROUTINE
LXI H,0000H ; LOAD TOP IN hxl
XCHG ; STORE TOP IN DAE
LXI D,RCTAD ; LOAD ROMCOUNT IN hxl
ADD TOP+ROWCOUNT, RESULT IN hxl
STL LockX ; STORE RESULT IN MEM

; COMPENSATION SUBROUTINE COMRX
LXI D,OP30H ; LOAD LOCXX IN hxl
LXI D,OP30H ; LOAD COMPENSATION VALUE IN DAE
STL LockX ; ADD DAE+TO hxl
MVI A,OP30H ; STORE RESULT IN LockX

; CLEAR SCREEN ROUTINE
MVI A,OP30H ; MOVE CURSOR TO A
MVI A,B,9FH ; MOVE LOOP CTR START VALUE =19H=25D TO B
MVI A,D,50H ; MOVE 50D=50H TO DAE
LXI H,8000H ; MOVE 8000H TO hxl

; MOVE CURSOR TO MEM
MVI A,M,A ; ADD B0D=50H TO ADDRESS IN hxl
DAD D ; DECREMENT B
DCR B ; CONTINUE LOOPING IF B NOT ZERO
JNZ LOADX ; MOVE CURSOR TO ADDRESS IN hxl

; ERASE TO END OF SCREEN ROUTINE
LXI H,0000H ; ZERO hxl
STL RCTAD ; ZERO ROWCOUNT
LXI H,8000H ; LOAD TOP IN hxl
LXI H,8000H ; STORE TOP IN DAE
ADD TOP+ROWCOUNT, RESULT IN hxl
FIRST CHAR IN PRESENT ROW ; STORE LOCATION IN MEM

; CALL LOAD CURSOR POSITION SUBROUTINE
LOAD USR CHAR IN A
MASK CHAN SAVE BITS 2-3
LOAD PASE ADD TO TUPLE 2 IN H/L
CLEAR DAE
LOAD OFFSET IN E
ADD OFFSET TO PASE, RESULT IN H/L
MOVE LOW BYTE OF ROUTINE ADD TO E
INCREMENT COMPUTED ADDRESS
MOVE UP BYTE OF ROUTINE ADD TO D
EXCHANGE DAE WITH H/L
LOAD PC WITH ROUTINE ADD, JMP TO ROUTINE

CURSOR UP ROUTINE

LOAD ROWCOUNT IN H/L
MOVE LOW BYTE OF ROWCOUNT TO A
COMPARE BYTE WITH 00H
IF BYTE=0 CONTINUE COMPARISON
CALL ROWUP SUBROUTINE
RETURN

MOVE UP BYTE OF ROWCOUNT TO A
COMPARE BYTE WITH 00H
IF BYTE=0,ROWCOUNT=FIRST ROW
CALL ROWUP SUBROUTINE
RETURN

BETA: LXI H,0780H
STORE 0780H IN ROWCOUNT BUFFER
LOAD A WITH CURSOR Y POS LAST ROW VALUE (19200)
CALL WP75
CALL LOAD CURSOR POSITION SUBROUTINE

CURSOR DOWN ROUTINE

LOAD ROWCOUNT IN H/L
MOVE LOW BYTE OF ROWCOUNT TO A
COMPARE BYTE WITH 00H
IF BYTE=0,CONTINUE COMPARISON
CALL ROWDOWN SUBROUTINE
RETURN

MOVE UP BYTE OF ROWCOUNT TO A
COMPARE BYTE WITH 07H
IF BYTE=07H,ROWCOUNT=LAST ROW
CALL ROWDOWN SUBROUTINE
RETURN

CURSOR RIGHT ROUTINE

LOAD COLUMN COUNT IN A
COMPARE BYTE WITH 4FH
IF BYTE=4FH,COLUMN COUNT =LAST CHARACTER POS IN ROW
CALL COLUMN RIGHT SUBROUTINE
RETURN

CALL CORT
RETURN

CURSOR LEFT ROUTINE

LOAD COLUMN COUNT IN A
COMPARE BYTE WITH 00H
IF BYTE=0,COLUMN COUNT =FIRST CHAR POS IN ROW
CALL COLUMN LEFT SUBROUTINE
RETURN

LOAD ROWCOUNT IN H/L
MOVE LOW BYTE OF ROWCOUNT TO A
COMPARE BYTE WITH 00H
IF BYTE=0,CONTINUE COMPARISON
CALL WP75
CALL LOAD CURSOR POSITION SUBROUTINE
CALL SCROLL SUBROUTINE
RETURN
Appendix 5.6
SOFTWARE LISTINGS

LOC OBJ SEQ SOURCE STATEMENT

00F8 5 CNCTIL EQU 0F8H ; 8251 CONTROL ADDRESS
00FA 6 CNIN EQU 0FAH ; 8251 INPUT DATA ADD
00FB 7 CNOUT EQU 0FAH ; 8251 OUTPUT DATA ADD
00FF 8 KCOM EQU 6FH ; 8279 COMMAND ADDRESS
006E 9 KDAT EQU 6EH ; 8279 DATA ADDRESS
005E 10 CRCOM EQU 5FH ; 8275 COMMAND ADDRESS
005E 11 CRDAI EQU 4EH ; 8275 DATA ADDRESS
0045 12 PC52ST EQU 45H ; CH 2 START ADD PORT
0046 13 PC52TC EQU 46H ; CH 2 TERM COUNT PORT
0047 14 PC52EA EQU 46H ; CH 2 STARTING ADD PORT
0047 15 PC52ER EQU 47H ; CH 2 TERM COUNT PORT
0000 16 MD57 EQU 00H ; MODE CLEAR
0084 17 MD57 EQU 88H ; MODE SET (AUTOLOAD, CH 2 ENABLED)
0048 18 PMD57 EQU 46H ; MODE SET PORT

19 ; SYSTEM INITIALIZATION ROUTINES

0000 C34000 0000 JMP CRITGO ; JUMP TO START OF MAIN ROUTINE
0038 0038H ORG 0038H
0038 C3900 0000 JMP POLL ; JUMP TO START OF INT SERVICE ROUTINE
0040 090H ORG 0040H
0041 31FF87 32 CRITGO: DI ; DISABLE INTERRUPTS
0041 31FF87 33 LXI SP, 87FFH ; LOAD STACK POINTER
0041 31FF87 34 45 ; MEMORY CLEAR ROUTINE
0044 210000 30 LXI H,8000H ; LOAD HL WITH START ADD OF DISPLAY MEM
0047 3E20 31 THETA: MVI A, 00H ; LOAD A WITH SPACE CHAR CODE
0049 77 32 MOV M,A ; LOAD SPACE CHAR IN MEM
0049 7D 33 MOV A,00H ; MOVE LOW ADD BYTE TO A
004B FE9F 40 CPI OCFH ; COMPARE WITH OCFH
004D CA5400 41 JZ NXX1 ; IF COMPARISON JMP TO NXX1
0051 C34700 42 INX H ; INCREMENT H,L
0054 7C 43 JMP THETA ; JMP TO THETA, CONT LOADING MEMORY
0055 FE87 44 NXX1: MOV A,H ; MOVE UP ADD BYTE TO A
0057 CA5200 45 CPI 87H ; COMPARE WITH 87H
0057 CA5200 46 JZ NXX2 ; IF COMPARISON ADD LAST DISPLAY
005A 23 47 INX H ; INCREMENT H,L
005B C34700 48 JMP THETA ; JMP TO THETA, CONT LOADING MEMORY

50 ; POINTER/BUFFER CLEAR ROUTINE

005E 210000 53 NXX2: LXI H,0000H ; ZERO H,L
0061 22D387 54 SHLD RCTAD ; ZERO ROW COUNT
0064 22D887 55 SHLD LOCUP ; ZERO BUFFER
0067 22D887 56 SHLD LOCAD ; ZERO CHARACTER LOCATION
006A 22DA87 57 SHLD LOCO1 ; ZERO LOC OF 1ST CHAR IN ROW
006D 22DC87 58 SHLD LOCO0 ; ZERO LOC OF OTHER CHAR IN ROW
0070 22DE87 59 SHLD LOCX ; ZERO PRESENT LOC OF 1ST CHAR IN ROW
0073 22D287 60 SHLD LOCPR ; ZERO PRESENT LOC OF 1ST CHAR IN ROW
0076 210080 61 LXI H,8000H ; LOAD H,L WITH 8000H
0079 22D687 62 SHLD TOPAD ; SET TOP = 8000H
007C 218087 63 LXI H,8780H ; LOAD H,L WITH 8780H
007F 22D687 64 SHLD BOTAD ; SET BOT = 8780H
0082 3E00 65 MVI A,00H ; ZERO A
0082 3E00 66 STA CCTAD ; ZERO COLUMN COUNT
0087 32D287 67 STA CURSY ; ZERO CURSOR Y POINTER
0087 32D587 68 STA XPLS ; ZERO ESC SEQ FLAG
008D 32EB87 69 STA USCHR ; ZERO USART CHAR BUFFER

70 ; 8251 INITIALIZATION ROUTINE

0090 3E48 71 MVI A,4FH ; MODE SET VALUE TO A
0092 3DFB 74 OUT CNCTIL ; OUTPUT VALUE
0094 3E27 72 MVI A,27H ; COMMAND WORD TO A
0096 3DFB 75 OUT CNCTIL ; OUTPUT VALUE

77 ; 8279 INITIALIZATION ROUTINE

0098 3E35 79 MVI A,35H ; OUTPUT PROG CLOCK, DIV BY 21
009A 3D6F 81 OUT KCOM

84 ; 8275 INITIALIZATION ROUTINE

45
ROW TIME

1st CHARACTER ARRIVES

2nd CHARACTER ARRIVES (2083 μsec)

DMA AVAILABLE CPU DMA AVAILABLE CPU DMA AVAILABLE CPU

TIME BETWEEN CHARACTERS = 2.083 ms = 2083 μsec

TOTAL CPU TIME AVAILABLE BETWEEN CHARACTERS = (457 μsec x 3) + (2082 - 2011 μsec) = 1443 μsec

BAUD RATE = 4800 BAUD
10 BITS/CHARACTER

Figure 5-3. CPU Availability/Character at 4800 Baud (DMA Active)

Figure 5-4. Typical Character Attribute Logic
**START**
INITIALIZE
ROUTINES

[Flowchart diagram]

* UNDER NORMAL OPERATING CONDITIONS, 427 CLOCK CYCLES REPRESENTS
  THE WORST CASE EXECUTION TIME FOR THIS ROUTINE

** IT IS NECESSARY FOR THE REMOTE DEVICE TO WAIT APPROXIMATELY 2.5 ms
  FOLLOWING THE TRANSMISSION OF AN ESCJ CHARACTER BEFORE RESUMING
  TRANSMISSION.

Figure 5-1. Subroutine Execution Times Flowchart

### DMA ACTIVE

<table>
<thead>
<tr>
<th>1st ROW TIME</th>
<th>24th ROW TIME</th>
<th>25th ROW TIME</th>
<th>1st VERT RETRACE ROW</th>
<th>2nd VERT RETRACE ROW</th>
</tr>
</thead>
<tbody>
<tr>
<td>617 ( \mu \text{sec} )</td>
<td>617 ( \mu \text{sec} )</td>
<td>617 ( \mu \text{sec} )</td>
<td>617 ( \mu \text{sec} )</td>
<td>617 ( \mu \text{sec} )</td>
</tr>
<tr>
<td>160 ( \mu \text{sec} )</td>
<td>457 ( \mu \text{sec} )</td>
<td>160 ( \mu \text{sec} )</td>
<td>457 ( \mu \text{sec} )</td>
<td>160 ( \mu \text{sec} )</td>
</tr>
</tbody>
</table>

**DMA AVAILABLE**

### DMA INACTIVE

**CPU AVAILABLE**

**CPU AVAILABLE**

1234 \( \mu \text{sec} \)

### DMA ACTIVE

**CPU AVAILABLE**

**DMA AVAILABLE**

Figure 5-2. CPU Availability
Appendix 5.4
SOFTWARE TIMING

Subroutine execution times are summarized in the flowchart provided in Figure 5-1. The values shown represent the number of clock cycles required for the execution of a given routine. The actual routine execution time is obtained by multiplying the number of clock cycles/routine by the time/clock cycle. For a 2.048 MHz system clock, the time/clock cycle is 0.4883 µsec. It should be noted that the values indicated represent worst-case execution times. In order to appreciate the meaning of the subroutine execution times, it is necessary to consider two factors:

1. The time available for the CPU to execute instructions between DMA operations.
2. The maximum rate at which data characters are presented to the CPU for processing.

CPU availability during a complete display frame is illustrated in Figure 5-2. Available CPU processing time, per character, at 4800 baud, during the DMA active portion of the display frame, is illustrated in Figure 5-3. It can be seen from Figure 5-3 that 1443 µsec are available for processing each character during the DMA active portion of the frame. Total CPU processing time during the DMA inactive portion of the frame may be seen from Figure 5-2 to be 1234 µsec. This value encompasses the time to process the 8275 interrupt and perform character handling functions.

Using the information contained in Figure 5-1, the maximum execution time* for a given character handling routine is 802 µsec. Since this value is less than 1,443 msec, proper timing is assured. Using the maximum character handling routine execution time and the time required for 8275 interrupt processing, the maximum CPU availability requirement during the DMA inactive portion of the frame may be calculated. This value corresponds to 802 µsec + 253 µsec (8275 interrupt processing) or 1055 µsec. Since this value is less than 1234 µsec, proper timing is assured.

*see notes, Figure 5-1.

Appendix 5.5
VISUAL ATTRIBUTE IMPLEMENTATION CONSIDERATIONS

In order to utilize the visual attribute features of the 8275, it is necessary to modify the CRT system hardware and software functions accordingly. Hardware modifications necessary to implement character attributes are illustrated in Figure 5-4. The attribute outputs LA0–LA1 selectively control the data transferred to the output shift register.

The software memory management scheme presented in the Application Note must be modified in order to accommodate attribute features. An outline of the software considerations involved when using the attribute features is presented as follows:

1. Attributes, as described in the 8275 Data Sheet, occupy character locations in display memory. Since the number of attributes per display row may be variable, the linear mapping relationship between character position on the screen and memory pointers Top Row Count, and Column Count no longer exists. It is necessary to keep track of the number of attribute characters in each row and their specific location when modifying pointer values.

2. The increased number of character locations required will force the user to incorporate additional display RAM.

3. Since the total number of characters in display memory may be variable when attributes are utilized, it is necessary to modify the starting address and terminal count values for the DMA channels as required.

4. Character insertion and deletion operations may be handled through block transfer operations or through the use of extended display memory row segments.
### Appendix 5.2
#### ESCAPE/CONTROL/DISPLAY CHARACTER SUMMARY

<table>
<thead>
<tr>
<th>BIT</th>
<th>0_0</th>
<th>0_1</th>
<th>0_10</th>
<th>1_0</th>
<th>1_01</th>
<th>1_10</th>
<th>1_11</th>
<th>0_10</th>
<th>0_11</th>
<th>1_0</th>
<th>1_1</th>
<th>1_11</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>NUL</td>
<td>DLE</td>
<td>SP</td>
<td>@</td>
<td>#</td>
<td>$</td>
<td>%</td>
<td>&amp;</td>
<td>'</td>
<td>(</td>
<td>)</td>
<td>\</td>
</tr>
<tr>
<td>0001</td>
<td>SOH</td>
<td>DC1</td>
<td>O</td>
<td>1</td>
<td>A</td>
<td>Q</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0010</td>
<td>STX</td>
<td>DC2</td>
<td>R</td>
<td>2</td>
<td>B</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0011</td>
<td>ETX</td>
<td>DC3</td>
<td>S</td>
<td>3</td>
<td>C</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0100</td>
<td>EOT</td>
<td>DC4</td>
<td>T</td>
<td>4</td>
<td>D</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0101</td>
<td>ENQ</td>
<td>NAK</td>
<td>U</td>
<td>5</td>
<td>E</td>
<td>U</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0110</td>
<td>ACK</td>
<td>SYN</td>
<td>V</td>
<td>6</td>
<td>F</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0111</td>
<td>BEL</td>
<td>ETS</td>
<td>W</td>
<td>7</td>
<td>G</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>BS</td>
<td>CAN</td>
<td>X</td>
<td>H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1001</td>
<td>HT</td>
<td>EM</td>
<td>Y</td>
<td>9</td>
<td>I</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1010</td>
<td>LF</td>
<td>SUB</td>
<td>Z</td>
<td>J</td>
<td>Z</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1011</td>
<td>VT</td>
<td>ESC</td>
<td></td>
<td>K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1100</td>
<td>FF</td>
<td>FS</td>
<td></td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1101</td>
<td>CR</td>
<td>GS</td>
<td></td>
<td>M</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1110</td>
<td>SO</td>
<td>RS</td>
<td></td>
<td>N</td>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1111</td>
<td>SI</td>
<td>US</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**NOTE:** Shaded blocks indicate characters that are ambiguous or may not be present in all terminals. Others can be generated but are ignored upon receipt.
Scroll Subroutine (SCROL)

The scroll subroutine, Figure 4-37, fills the row in display memory pointed to by TOP with space characters via the fill subroutine, then modifies the value of TOP. TOP is utilized by the 8275 service subroutine in re-initializing the 8257 DMA controller.

Fill Subroutine (FILL)

The fill subroutine, Figure 4-38, calculates the location of the last character in the current display row, plus one character position, by adding 80D = 50H to the location of the first character in the current display row. The current stack pointer value is saved, then the stack pointer is loaded with the location of the last character in the current display row, plus one character position. The B and C registers of the CPU are loaded with space characters and 40 PUSH B operations performed. This technique provides a rapid means (275 μsec) of filling a given row with space codes.

Load Cursor Position Subroutine (WP 75)

The load cursor position subroutine, Figure 4-39, transfers the contents of the Column Count and cursor Y position pointers to the 8275 cursor X position and cursor Y position registers, respectively.
Carriage Return Routine (CTRLM)
The carriage return routine, Figure 4-32, clears the column count and updates the 8275 cursor position registers.

Row Up, Row Down Subroutines (ROW UP, ROW DOWN)
The row up subroutine, Figure 4-33, subtracts 80D from the Row Count value, decrements the Cursor Y Position pointer, and updates the 8275 Cursor Position registers. The row down subroutine, Figure 4-34, differs in that 80D is added to Row Count.

Column Right, Column Left Subroutines (COLRT, COLLT)
The column right subroutine, Figure 4-35, increments the Column Count pointer and updates the 8275 cursor position registers. The column left subroutine, Figure 4-36, differs in that the Column Count is decremented.
The End-of-Row character (EOR) recognized by the 8275 allows the clear screen feature to be executed in a considerably shorter time span. During the clear screen routine, Figure 4-28, EOR characters are placed in the first character position of each row in display memory. Since the EOR character blanks the entire display row when placed in the first character position of the row, the use of EOR characters in each row blanks the entire screen. All pointers are cleared during the clear screen operation.

**Erase to End of Screen Routine (ESCI)**

The erase to end of screen routine, Figure 4-30, inserts End of Row characters (EOR) in display memory in the same fashion as the clear screen routine. The fundamental difference between the routines is that the erase to end of screen routine must insert EOR characters selectively. Only rows from the present display row until the last display row, pointed to by BOTTOM, receive EOR characters. It should be noted that the pointer BOTTOM changes dynamically with scrolling operations.

**Home Routine (ESCH)**

The home routine, Figure 4-29, resets the Row Count, Column Count and Cursor Y Position buffers to zero, but does not affect the value of TOP.

**Erase Line Routine (ESCK)**

The erase line routine, Figure 4-31, calculates the location of the first character in the current display row, stores the location in buffer memory, and calls the fill subroutine, which fills the row with space codes.

**Backspace Routine (CTRLH)**

See cursor left routine.

**Line Feed Routine (CTRLJ)**

See cursor down routine.
Cursor Down Routine (ESCB)

The cursor down routine, Figure 4-25, determines if the cursor is located in the last display row. If it is, the scroll subroutine is called. No modification of cursor position is called for. If the cursor is not located in the last display row, the row down subroutine is called.

Cursor Right Routine (ESCC)

The cursor right routine tests the cursor location and moves the cursor as described in Figure 4-26. If the cursor is in the last display position, a scrolling operation occurs. 8275 Cursor X and Y Position registers are updated accordingly.

Cursor Left Routine (ESCD)

The cursor left routine tests the cursor location and moves the cursor as described in Figure 4-27.

Clear Screen Routine (ESCE)

Several possibilities existed for implementing the clear screen function. The simplest of these techniques involves filling the display memory with space codes. This technique, although conceptually simple, requires several milliseconds to implement.
Display Subroutine One (DIS1)

Display subroutine one, Figure 4-22, calculates the location in memory at which the display character is to be inserted. If the location calculation results in an address outside of the display memory bounds, appropriate compensation action is taken. Prior to inserting the display character in memory, the first character position in the row in which the character will be located is examined. If an End of Row character (EOR) is found, the row in question will be blanked by the 8275. It is necessary to clear the row by filling it with space codes (Fill Subroutine), then insert the display character in the desired location. If no EOR character is found, insertion proceeds without further software intervention.

Figure 4-22. Display Subroutine 1 (DIS1)

Display Subroutines A, B, C (DISA, DISB, DISC)

Display subroutines A, B, and C, Figure 4-23, modify the appropriate display memory pointers. The modifications are based on the present cursor location, as determined by subroutine DISPL. The resulting cursor position data is transferred to the 8275 Cursor X and Y Position registers. If DISC is called, a scrolling operation occurs.

Figure 4-23. Display Subroutines — A (DISA), B (DISB), C (DISC)

Cursor Up Routine (ESCA)

The cursor up routine, Figure 4-24, determines if the cursor is located in the first display row. If it is, the Row Count and Column Count values are modified, and the cursor is moved to the last display row with no change in X position. If the cursor is not in the top row, the row up subroutine is called.
Control Code Subroutine (CNTRL)

The control code subroutine, Figure 4-20, involves, conceptually, the same procedures executed by the escape sequence subroutine. A summary of control code functions is given in Appendix 5.2.

Display Character Handling Subroutine (DISPL)

The display character handling subroutine, Figure 4-21, determines if the cursor is located in the last column of the row, the last display position, or elsewhere and calls the appropriate subroutines.
**USART Read/Store Subroutine (RDF 51)**

The read/store USART character subroutine, Figure 4-17, moves a character from the USART to the CPU, masks off the upper-most bit, and stores the character in system buffer memory.

**Escape Sequence Subroutine (ESREC)**

The escape sequence subroutine, Figure 4-19, performs a masking operation on the USART character, shifts the result by one bit position, and adds this value to the base address of the escape sequence lookup table, BSETI. The lookup table contains starting addresses for each of the escape sequence routines. This address is jammed into the program counter and the routine executed. A summary of escape sequence functions is given in Appendix 5.2.
Interrupt Polling Subroutine (Poll)

The interrupt polling subroutine, Figure 4-15, tests to determine the source of the interrupt. If the interrupt originated with the 8275, the 8275 interrupt service subroutine is called. Following completion of the subroutine, interrupts are re-enabled, and a return executed. An interrupt issued from the 8251 forces subroutine calls to the read/store USART character subroutine and the character recognition/handling subroutine. Interrupts are re-enabled at the completion of the character recognition/handling routine. A return operation follows.

8275 Interrupt Service Subroutine (RT 75)

The 8275 interrupt service subroutine, Figure 4-16, re-initializes the 8257 DMA Controller, then tests the 8279 FIFO status. If a character has been transmitted from the keyboard to the Keyboard Controller, a table lookup operation is performed to obtain the correct ASCII code for the character, and the character is transmitted.
4.4.4 System Subroutines

System Initialization Routine (CRTGO)

The system initialization routine, Figure 4-14, establishes a starting point for system operation. The 8251 USART is initialized to transmit to and receive characters from an external device. The 8279 Keyboard Controller, at system reset, comes up in the two-key rollover mode. It is therefore only necessary to set up the Keyboard Controller internal operating frequency during initialization. Assuming a desired internal operating frequency of approximately 100 kHz and a 2.048 MHz system clock, the frequency divider chain is programmed to divide by 21. The 8275 initialization parameters are determined from the original CRT system specifications and vertical retrace Row Count/Horizontal Retrace Character Count calculations previously performed. The delayed line number feature allows the use of only 3 line count outputs to determine which of 10 possible lines in a character row will be displayed. Given that the underline placement position is set to the ninth row, the top and bottom lines of the character are automatically blanked, leaving, effectively, 8 unique lines for display. The 8275 cursor position registers are initialized to zero, forcing the cursor to the upper left-hand corner of the display. The preset counters command resets all 8275 counters to zero and stops the 8275 counters until another command is issued. The 8275 is then started by a start display command. An interrupt will be generated from the 8275 approximately 15 ms later. Interrupts are enabled following the 8275 start command. Interrupts were disabled prior to this time to insure that the central processor did not react to erroneous interrupts from the 8275 generated prior to 8275 initialization. The processor, following initialization, waits in a loop until the arrival of an interrupt from the 8275 or 8251.
Table 4-2
SCREEN POINTER/8275 CURSOR X,Y POSITION REGISTER RELATIONSHIP

<table>
<thead>
<tr>
<th>ROW</th>
<th>ROW COUNT VALUE</th>
<th>CURSOR Y POSITION REGISTER VALUE</th>
<th>COLUMN</th>
<th>COLUMN COUNT VALUE</th>
<th>CURSOR X POSITION REGISTER VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0000H</td>
<td>00H</td>
<td>1</td>
<td>00H</td>
<td>00H</td>
</tr>
<tr>
<td>2</td>
<td>0050H</td>
<td>01H</td>
<td>2</td>
<td>01H</td>
<td>01H</td>
</tr>
<tr>
<td>3</td>
<td>00A0H</td>
<td>02H</td>
<td>3</td>
<td>02H</td>
<td>02H</td>
</tr>
<tr>
<td>4</td>
<td>00F0H</td>
<td>03H</td>
<td>4</td>
<td>03H</td>
<td>03H</td>
</tr>
<tr>
<td>25</td>
<td>0780H = 1920D</td>
<td>18H = 24D</td>
<td>80</td>
<td>4FH = 79D</td>
<td>4FH = 79D</td>
</tr>
</tbody>
</table>

Scrolling

Scrolling is implemented in the CRT system design by shifting the entire display up by 1 row when a scrolling condition occurs. Scrolling will occur when certain cursor manipulation functions are exercised or when a character is entered in the last CRT display position, indicating a full memory page condition exists. Character entry will be used as the vehicle for explaining scrolling in the following discussion.

Characters are normally entered sequentially in display memory. When the 2000th character has been entered, display memory capacity has been attained; i.e., a full page condition exists. At this point, scrolling will take place. For scrolling to take place, DMA channel 2, the channel used to extract characters from display memory, must be re-initialized to the appropriate starting address and terminal count values. The memory pointer TOP will be used to establish the starting address for channel 2. Prior to scrolling, TOP = 8000H, the starting address of display memory. Each scrolling operation causes 80D (50H) to be added to TOP, moving the pointer, as shown in Figure 4-13b, to the beginning of the following row in display memory. It should be recalled that TOP, in conjunction with Row Count and Column Count determines the insertion address for incoming display characters. The net effect of modifying TOP is to shift the information being displayed on the CRT up by 1 row; i.e., scrolling is accomplished. Prior to scrolling, the terminal count value for DMA channel 2 is equal in magnitude to the display memory length -1 or 87CFH - 8000H. The actual value sent to the terminal count register is 87CFH - 8000H + 8000H. The addition of 8000H sets bit 14 in the terminal count register to a 1, indicating a DMA read operation. If scrolling is to be implemented, the terminal count value must be modified to 87CFH - TOP + 8000H. Characters transferred by channel 2 include those characters located from the address specified by TOP to the end of display memory. In order to transfer the characters from the beginning of display memory through the address immediately prior to TOP, the autoload feature of the 8257 DMA controller is utilized. When DMA channel 2 reaches terminal count, following the transfer of characters from TOP to the end of display memory, the starting address and terminal count parameters stored in the DMA channel 3 registers are loaded into channel 2. DMA operations resume in channel 2 using the channel 3 parameters. To accomplish the desired channel 3 operations, it is only necessary to re-initialize the channel 3 starting address to the beginning address of display memory, and the terminal count value to 87CFH, the maximum terminal count for a 2000-byte display memory space. These processes are performed during DMA re-initialization following an 8275 interrupt. New text entry following scrolling is illustrated in Figure 4-13. BOTTOM, a parameter corresponding to the address of the first character in the last row to be displayed, is utilized during clear to end of screen operations.
System Memory Organization

System memory organization is shown in Figure 4-11. It should be noted that an additional 2K block of RAM was utilized for program memory (rather than PROM) during the software development/debug phase of system design.

![Figure 4-11. System Memory Organization](image)

Character Position/Screen Pointer Relationships

To define the location of a character on the screen, two pointers, Row Count and Column Count, were created in memory. The relationship between character location on the screen and the two pointers is illustrated in Figure 4-12. Row Count and Column Count are stored in memory locations RCTAD and CCTAD, respectively. Row Count represents the position of the first character in a given row. For the first row, Row Count = 0000H. For the second row, Row Count = 0050H. Column Count represents the specific column in which the character is located. Character position on the screen may be calculated by adding the Row Count to the Column Count; e.g., the highlighted character in Figure 4-12 is located at AOH + 03H = A3H.

![Figure 4-12. Character Location/Pointer Relationship](image)

Memory Pointer/8275 Cursor Position Register Relationship

It was necessary to establish a relationship between Row Count and Column Count pointers and the 8275 Cursor X and Y Position registers for the cursor generated by the 8275 to be loaded at the appropriate position on the screen. This relationship is summarized in Table 4-2.

The value transferred to the 8275 for the Cursor X Position is identical to the Column Count. A new parameter, Cursor Y Position, stored at memory location CURSY, was also established. For a given Row Count value, a value for Cursor Y Position is defined. This value is transferred to the 8275 Cursor Y Position register.

It is necessary to introduce an additional parameter, Top, which will be used in conjunction with Row Count and Column Count to determine the location in display memory at which an incoming display character will be stored. The location at which a given character will be stored (assuming no more than 2000 characters have been entered since initialization) is calculated by adding TOP + Row Count + Column Count, where TOP is assumed to be 8000H, the starting location of display memory shown in Figure 4-11. Following system initialization, characters will be entered in display memory starting at memory location 8000H. The 2000th character will be entered at location 87CFH. Upon entering the 2001st character, a scrolling condition exists and TOP will be modified to point to memory address 8050H. An in-depth discussion of scrolling is presented in the next section.
system reset, the central processor interrupt system is disabled, the program counter is set to zero, and peripheral reset functions are carried out. Following reset, the system software initializes all peripherals, clears buffer memory, initializes special buffer locations, fills display memory with space codes, and enables interrupts. The processor then loops until an interrupt arrives from the 8275 or 8251. When the processor detects the occurrence of an interrupt, the instruction being executed is completed, an RST 7 vector is placed on the system data bus, and the RST 7 call instruction is executed, forcing a jump to the starting address of the 8275/8251 interrupt polling routine. Once the polling routine establishes the source of the interrupt, program flow continues along one of the two possible paths shown in Figure 4-10. An 8275 interrupt causes the 8257 DMA Controller to be reinitialized, the 8279 Keyboard Controller to be serviced, and, if a key depression has occurred, a character to be transmitted to the terminal output. An interrupt from the 8251 will first cause the USART character to be read and stored in memory. The system software then examines the character to determine whether it is a displayable character, a control code, or the first or second character in an escape sequence. After determining the nature of the character, an appropriate subroutine is called. Following the completion of the routines associated with an 8275/8251 interrupt, interrupts are re-enabled and a return instruction executed. The CPU then loops until the receipt of an interrupt. In order to appreciate the operation of the system software in detail, it is necessary to consider the following items:

1. System memory organization.
2. The relationship between character position on the screen and screen pointers Row Count, Column Count, and memory pointer Top.
4. Scrolling concepts, including the relation between scrolling, display memory, and the memory pointer Top.
4.4 SYSTEM SOFTWARE DESIGN

4.4.1 General Considerations

The approach taken in presenting the system software design is as follows: First, the software development process will be outlined. A discussion of system software operation will then be undertaken. Software operation will be followed by a detailed presentation of system subroutines.

4.4.2 Software Development

Software development was accomplished using the following tools:

1. Intel® MDS microcomputer development system
2. Intel® dual floppy disc system
3. Intel® ICE-80 In-Circuit Emulator
4. Intel® ISIS II disc operating system

The MDS was utilized in conjunction with the dual floppy disc system for program editing, assembly, relocation, and loading functions.

The ICE module was used extensively for loading assembled routines into the prototype system RAM and debugging program errors. While in the emulation mode, the ICE processor controlled the operation of the CRT system. During debugging, emulation proceeded normally until certain user specified break conditions occurred, at which time ICE entered the interrogation mode. During interrogation mode all processor functions, including DMA, ceased, allowing the user to access and display CPU register contents, status, and up to 44 previous machine cycles, system memory contents, and I/O device data.

4.4.3 Operation

The fundamental operations performed by the CRT system software are presented in Figure 4-10. Extensive use of subroutines in implementing major software functions resulted in readily understandable software. Debugging operations were also simplified as a result of the software structure. At
4.3.5 Keyboard Interface Design

The keyboard interface, Figure 4-8, consists of the 8279 Keyboard Controller and the decoding logic necessary for scanning the keyboard matrix. The 8279 SL0–SL2 output lines are decoded by the 74S138 decoder. The eight output lines from the decoder select 1 of 8 keyboard matrix rows for testing by the 8279. The keyboard matrix column output lines are connected to the 8279 return lines, RL0–RL7. Open collector outputs presented by individual keys within the matrix eliminate the need for isolation diodes when two keys in a given column are depressed. Two-key rollover was chosen as the operating mode for the 8279.

4.3.6 System Memory Design

The system memory, illustrated in Figure 4-9, consists of one 2716 EPROM used for program storage and four 2114 RAMs used for display memory, buffer memory, and system stack. The 2114 4K static RAM was chosen for the design because of its 1K X 4 organization, ease of use, and availability. Buffering between RAM memory and the system data bus was used to minimize bus loading.
LTEN. The delay is accomplished using a two-stage shift register constructed with edge triggered D flip-flops (74175). The system dot clock (11.34 MHz) is obtained by dividing the 22.68 MHz output from the 8224 clock generator by two. The dot clock is utilized to clock the 74166 output shift register and is divided by 7, using a 74S163 counter, to produce the system character clock. It should be noted that the use of a bipolar character generator PROM such as the Intel® 3604 or 3608 will reduce the external dot timing logic package count due to the reduced access time.

Figure 4.6. Dot Timing Logic
The Horizontal Blanking Time (HRTC) is calculated as follows:

\[
HRTC = 20 \times (617.284 \text{ ns})
\]
\[
= 12,346 \mu\text{sec (nominal value 11 \mu\text{sec})}
\]

The 8275 will be programmed for a Horizontal Retrace Count of 20. Since the specifications call for a Horizontal Drive Pulselwidth of 25–30 \mu\text{sec}, an external one-shot is required. The one-shot is triggered by the leading edge of HRTC.

Using the value for the Character Time/Line, the Dot Clock Rate may be established. It should be noted that the clock is used to shift data from the parallel in-serial out shift register (contained in the dot timing logic) to the CRT video input. The system character clock is also derived from the Dot Clock.

The dot clock is calculated as follows:

\[
\left( \frac{\text{Dot Time}}{\text{line}} \right) = \left( \frac{\text{Character Time}}{\text{line}} \right) \times \left( \frac{\# \text{ dots/character}}{7} \right)
\]
\[
= 6,17284 \times 10^{-7} \times \frac{1}{7}
\]
\[
= 8.8183 \times 10^{-8} \text{sec}
\]
\[
= 88.183 \text{ ns}
\]

Dot Clock Frequency \[
= \frac{1}{\text{Dot Time/Line}} = 11.34 \text{ MHz}
\]

The Horizontal Oscillator Repetition Rate may be calculated as follows:

\[
f_{\text{Horiz}} = \frac{1}{\text{Total Line Time}} = \frac{1}{61.7284 \times 10^{-6} \text{ sec}}
\]
\[
= 16,200 \text{ Hz}
\]

This value falls within the system specification of 15,750 ±500 pps.

4.3.4 Dot Timing Logic

The primary function of the dot timing logic, illustrated in Figure 4-6, is to transfer the output of the character generator ROM to the video input of the CRT. Due to the high data transfer rate (11.34 MHz), logic external to the 8275 is required for this function. The data transfer operation is accomplished as follows: The character generator ROM output is applied to the parallel input lines of the 74166 shift register, the shift register is loaded synchronously with respect to the positive-going edge of the character clock, and data is clocked out of the 74166 serial input at the dot clock frequency. The 74166 output is applied, through appropriate gating logic, to the CRT video input. In addition to the previously described functions, the dot timing logic provides the timing signals required for transferring characters from the 8275 character code and line count outputs to the character generator ROM, implements the video suppress and light enable gating functions, and generates the system dot and character clocks.

In order to understand the dot timing logic design process, it is necessary to refer to Figure 4-6 and Figure 4-7.

It can be seen from the timing waveforms of Figure 4-7 that the character code output from the 8275 will be valid 150 ns (worst case) after the negative-going edge of the character clock. The character generator ROM output will be valid, assuming a direct connection between the 8275 and the ROM, 450 ns (worst case) after the character code appears at the address inputs. Total delay from the negative-going edge of the character clock until ROM output data becomes available is then 600 ns. Given the character clock width of 617 ns and external logic propagation delays and setup times, it becomes difficult to latch the ROM output for the first display character during the first character clock period. In order to alleviate this situation, a data pipelining technique is utilized. The timing for this technique is shown in Figure 4-7. A latch, introduced between the 8275 and the character generator ROM as shown in Figure 4-6, samples character code and line count data from the 8275 1/2 dot clock (45 ns) after the positive-going edge of the character clock. Data from the latch is applied to the character generator ROM address lines yielding, after a 450 ns delay (worst case), the appropriate 7-bit code at the ROM output. ROM data is loaded into the 74166 shift register on the next positive-going edge of the character clock. This technique effectively delays the video output from the shift register by 1/2 character clocks, but eliminates the difficulties in sampling the ROM data within the first character clock period. Due to the video delay associated with this technique, it is also necessary to delay all signals affecting the video output and CRT timing. These signals include HRTC, VRTC, VSP, and
Table 4-1

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>RANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vertical Blanking Time (VRTC)</td>
<td>900 ( \mu )sec nominal</td>
</tr>
<tr>
<td>Vertical Drive Pulselength</td>
<td>300 ( \mu )sec ( \leq ) PW ( \leq ) 1.4 ms</td>
</tr>
<tr>
<td>Horizontal Blanking Time (HRTC)</td>
<td>11 ( \mu )sec nominal</td>
</tr>
<tr>
<td>Horizontal Drive Pulselength</td>
<td>25 ( \mu )sec ( \leq ) PW ( \leq ) 30 ( \mu )sec</td>
</tr>
<tr>
<td>Horizontal Repetition Rate</td>
<td>15,750 ±500 pps</td>
</tr>
</tbody>
</table>

Given the constraints in Table 4-1 and the Refresh Rate specification of 60 Hz, the Vertical Retrace Row Count and Horizontal Retrace Character Count parameters required by the 8275 CRT Controller may be calculated:

\[
\text{Total Screen Time} = \frac{1}{\text{Refresh rate}} = \frac{1}{60 \text{ Hz}} = 0.01667 \text{ sec}
\]

Also,

\[
\text{Total Screen Time} = \text{Row Time} \times (\# \text{ of Display Rows}) + \text{Vertical Blanking Time (VRTC)}
\]

Vertical Blanking Time (VRTC) must be an integral number of Row Times (between 1 and 4). Therefore,

\[
0.01667 \text{ sec} = \text{Row Time} \times (25) + \text{VRTC}
\]

If \( N \) is selected to be 2, the following result is obtained:

\[
\text{Row Time} = 6.17284 \times 10^{-4} \text{ sec}
\]

Therefore,

\[
\text{VRTC} = (2)(\text{Row Time}) = 12.3457 \times 10^{-4} \text{ sec} = 1.23457 \text{ ms}
\]

Since the Vertical Blanking Time, nominally 900 \( \mu \)sec, falls within the constraints for the Vertical Drive Pulselwidth, the VRTC output from the 8275 may be used directly for the Vertical Drive Pulse. The 8275 will be programmed for a Vertical Retrace Row Count of 2.

In order to calculate the Horizontal Retrace Character Count, it is necessary to consider the row format as defined in the specifications. Figure 4-5 shows three adjacent characters in a row. The row, as shown, is composed of 10 Lines/Row and 7 Dots/Line/Character. Given that the Row Time is 617.284 \( \mu \)sec, the Total Line Time may be calculated as follows:

\[
\text{Total Line Time} = \frac{\text{Row Time}}{\# \text{Lines/Row}} = \frac{617.284 \times 10^{-6} \text{sec}}{10} = 61.7284 \times 10^{-6} \text{ sec} = 61.7284 \mu \text{sec}
\]

The Total Line Time is composed of the display portion of the line plus the Horizontal Blanking Time (HRTC).

\[
\text{Total Line Time} = 61.7284 \times 10^{-6} \text{ sec} = 80 \left( \frac{\text{Character Time}}{\text{line}} \right) + \text{HRTC}
\]

Horizontal Blanking Time (HRTC) must be an integral number of Character Times/Line. Then

\[
61.7284 \times 10^{-6} \text{ sec} = 80 \left( \frac{\text{Character Time}}{\text{line}} \right) + M \left( \frac{\text{Character Time}}{\text{line}} \right)
\]

If \( M \) is selected to be 20, the following result is obtained:

\[
\frac{\text{Character Time}}{\text{line}} = \frac{61.7284 \times 10^{-6}}{80 + 20} = 6.1728 \times 10^{-7} \text{ sec} = 617.284 \text{ ns}
\]

This value defines the period of the 8275 character clock.

---

**Figure 4-5. Row Format**
rate necessary for CRT refreshing. Display characters are then transferred from the 8275 row buffers to the character code outputs CC0–CC5. The character code outputs are applied to the character generator address lines A3–A8 (Figure 4-3). Line count outputs LC0–LC2 from the 8275 are applied to character generator address lines A0–A2. It should be noted that the 8275 displays character rows one line at a time. The line count outputs are utilized to determine which line of the character selected by A3–A8 will be displayed. Following the transfer of the first line to the dot timing logic, the line count is incremented and the second line of the character row is selected. The process continues until the last line of the row under consideration is transferred to the dot timing logic.

The dot timing logic latches the 6-bit character code and 3-bit line count from the 8275 on positive transitions of the character clock and transfers this information to the character generator ROM. In systems requiring a greater number of lines/character, the fourth line count output would also be used. The 7-bit ROM output corresponds to the 7 dots which make up a line segment for a particular character. The ROM output is loaded into a parallel input-serial output shift register. The shift register is clocked at the dot clock rate (11.34 MHz) continuously. The shift register output constitutes the video input to the CRT. The character code outputs select the character to be displayed at a given character position in the display row. The character set consists of 2⁶=64 ASCII upper case alphanumeric characters.

The row by row transfer of character data from display memory to the 8275 continues until the beginning of the last display row. At this time the 8275 issues an interrupt to the CPU. The CPU polls both the 8275 and 8251. Having determined that the interrupt originated with the 8275, the CPU calls the 8275 interrupt subroutine. The 8275 interrupt subroutine re-initializes the 8257 DMA Controller starting address and terminal count parameters and polls the 8279 Keyboard Controller to determine if a key depression has occurred. If a key has been depressed, the CPU reads the key position data from the 8279, performs a table lookup, and transmits the appropriate ASCII character to the CRT data output via the 8251 USART. It should be noted that interrupts are generated by the 8275 every 16.67 ms for a 60 Hz screen refresh rate.

**Figure 4-3. Character Generator/Dot Timing Logic Block Diagram**

### 4.3.3 System Timing

The CRT terminal display raster is shown in Figure 4-4. It can be seen from the figure that a display row is composed of 10 lines. The Total Line Time consists of the display portion of the line plus the Horizontal Blanking Time. Row Time is equal to the number of lines per row multiplied by the Total Line Time. The Total Screen Time (1/Refresh Rate) is equal to the Row Time multiplied by the number of display rows plus the Row Time intervals associated with vertical blanking. Specifications for the BALL BROS. monitor show that there are constraints on the Vertical Blanking Time, Horizontal Blanking Time, and Horizontal Oscillator Repetition Rate. These constraints are summarized in Table 4-1.

**Figure 4-4. CRT Display Raster**
4.3.2 Operation

The 8080A CPU initializes each peripheral to the appropriate mode of operation following system reset. Upon receiving a character from a remote device, the 8251 USART issues an interrupt to the CPU. The CPU calls the interrupt service subroutine, which polls both the 8275 and 8251 to determine the source of the interrupt. Having determined that the 8251 issued the interrupt, the CPU calls the READ/STORE USART character subroutine, reads the USART character, and stores the character in buffer memory. The character recognition subroutine is called next. This routine determines whether the character is a displayable character, a control character, or a character in an escape sequence. Assuming the character is a displayable character, the CPU places the character in display memory at the location corresponding to the present cursor position, advances the cursor, modifies the display memory pointers, and, if required, performs the operations necessary for scrolling. If the received character is a control character or escape sequence character requiring cursor and display memory pointer changes, these functions are carried out. Escape sequences which involve erasing a portion of the display are also handled via the appropriate subroutines.

In order to place characters contained in display memory on the CRT display screen, the 8275 CRT Controller must first transfer the display characters, via the 8257 DMA Controller, to the 8275's row buffers. It should be noted that the 8257 DMA Controller is required to achieve the data transfer
4.2 SYSTEM SPECIFICATIONS

The specifications for the CRT terminal design are as follows:

Display Format
- 80 characters/display row
- 25 display rows

Character Format (Figure 4-1)
- 5×7 character contained within a 7×10 matrix, 1st and 10th lines blanked, 1st and 7th columns blanked, 9th line cursor position, blinking underline cursor.

Characters Recognized
- Displayable characters: 64 ASCII upper-case alphanumeric characters
- Control characters:
  - Line feed, Control J
  - Carriage return, Control M
  - Back space, Control H
- Escape Sequences:
  - Cursor up, ESC, A
  - Cursor down, ESC, B
  - Cursor right, ESC, C
  - Cursor left, ESC, D
  - Clear screen, ESC, E
  - Home, ESC, H
  - Erase to end of screen, ESC, J
  - Erase line, ESC, K

Characters Transmitted
- 64 ASCII upper-case alphanumeric characters
- ASCII Control Character set
- ASCII Escape Sequence set

Program Memory
- 2K bytes, 2716 EPROM

Display/Buffer/Stack Memory
- 2K bytes, 2114 static RAM

Data Rate
- 4800 BAUD maximum using 8080A

CRT Monitor
- Ball Bros TV-12, 12 MHz B.W.

Keyboard
- Microswitch hall effect keyboard, open collector outputs

Scrolling Capability
- Scroll up feature implemented with 8257 DMA Controller

4.3 SYSTEM HARDWARE DESIGN

4.3.1 General Considerations

A block diagram of the CRT terminal is presented in Figure 4-2. The diagram includes only essential system features. A detailed schematic of the CRT terminal is contained in the appendix. The terminal was constructed using an Intel® SDK-80 microcomputer kit and an Intel® SBC 905 prototyping board. The standard 8080 bus structure incorporated in the SDK-80 kit allowed the CRT terminal to be implemented with minimum buffering.

In the ensuing discussion of CRT terminal operation, it will be assumed that the terminal normally communicates with a remote device, such as an Intel® MDS microcomputer development system. Communication will take place in the full duplex mode. The CRT terminal, upon transmitting a character to the remote device, will remain idle until a character is received from the external device. Transmission of a character to the remote device is initiated by depressing a key on the keyboard. Character transmission to the CRT terminal from the remote device is assumed to be asynchronous with respect to terminal operation.
The primary functions of the 8279 in the CRT system application include scanning the 64 key keyboard, determining if a key has been depressed, and, when polled by the system processor, transmitting the address of the key in the keyboard matrix to the master processor. Alternately, the interrupt line from the 8279 may be used to inform the CPU of a key depression. A block diagram of the 8279 interface, as implemented in the CRT system design example, is provided in Figure 3-7. The keyboard controller initiates the keyboard scanning process by transmitting keyboard scan line selection information over output lines SL0–SL2. The data may be encoded or decoded depending on the mode programmed. Assuming encoded mode is selected, the SL0–SL2 lines are connected to the input of a 3-line to 8-line decoder as shown in Figure 3-7. The decoder outputs are connected to the keyboard row inputs. Only one decoder output will be enabled for a given set of input conditions. The keyboard column outputs are connected to the 8279 return line inputs RL0–RL7. The eight return lines are buffered and latched by the 8279. These lines are scanned by the internal logic of the 8279, looking for a key depression in the selected row. If the debounce circuit detects a key depression, it waits approximately 10 ms to determine if the key remains down. If it does, the address of the key in the matrix plus the status of the shift and control lines are transferred to the 8279 FIFO. The FIFO data format is shown in Figure 3-8. The FIFO will hold up to eight data bytes; that is, up to eight key depressions may occur prior to a CPU initiated read operation. The number of characters entered into the FIFO is indicated by the character count contained within the FIFO status word. When a key depression is detected, the 8279 interrupt line goes high, and the FIFO status is modified to reflect the number of characters contained in the FIFO. The CPU may determine the occurrence of a key depression in one of two ways: The 8279 interrupt line may be connected to the interrupt input line of the CPU, forcing the CPU to call an interrupt service routine which reads the FIFO character. An alternate approach requires the CPU to periodically poll the 8279, reading the FIFO status word. If the FIFO character count is non-zero, indicating that at least one character is present in the FIFO, the CPU then reads the FIFO contents. This approach will be utilized in the CRT design example. A read operation places the contents of the FIFO on the system data bus and decrements the FIFO character count, contained within the FIFO status word, by one.

![Figure 3-7. 8279 Interface](image)

![Figure 3-8. FIFO Data Byte Format](image)

4. CRT SYSTEM DESIGN EXAMPLE

4.1 SCOPE OF THE PROJECT

A fully operational, microcomputer-based CRT terminal was designed and constructed utilizing the 8275 CRT Controller and 8279 Keyboard Controller as the basic system elements. The terminal incorporates the majority of the functions found in existing dedicated computer terminals. An Intel® 8080A microprocessor was utilized as the CPU in the design. The recently announced Intel® 8085 microprocessor constitutes an ideal processor for future CRT terminal designs. LSI devices were utilized in the design whenever possible in order to minimize component count.
3.2 8279

The 8279 Programmable Keyboard/Display Interface block diagram and pin configuration are shown in Figure 3-6. The 8279 will be utilized in the CRT design example for performing keyboard scanning, key debounce, and data bus interface functions. Only features associated with these functions will be described in this section. The reader is referred to the 8279 data sheet for information on display control, sensor matrix mode operation, and strobed input mode operation. A detailed description of the 8279 keyboard scanning, debounce, and data bus interface functions follows.

---

**Figure 3-5. Field Attribute Examples**

---

**Figure 3-6. 8279 Pin Configuration and Block Diagram**
Character attributes were designed to produce the following graphics:

<table>
<thead>
<tr>
<th>CHARACTER ATTRIBUTE CODE “CCCC”</th>
<th>OUTPUTS</th>
<th>SYMBOL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000 Above Underline</td>
<td>0 0 1 0</td>
<td></td>
<td>Top Left Corner</td>
</tr>
<tr>
<td>0000 Underline</td>
<td>1 0 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000 Below Underline</td>
<td>0 1 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0001 Above Underline</td>
<td>0 1 0 0</td>
<td></td>
<td>Top Right Corner</td>
</tr>
<tr>
<td>0001 Underline</td>
<td>1 0 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0001 Below Underline</td>
<td>0 1 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0010 Above Underline</td>
<td>0 0 1 0</td>
<td></td>
<td>Bottom Left Corner</td>
</tr>
<tr>
<td>0010 Underline</td>
<td>1 0 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0010 Below Underline</td>
<td>0 0 1 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0011 Above Underline</td>
<td>0 1 0 0</td>
<td></td>
<td>Bottom Right Corner</td>
</tr>
<tr>
<td>0011 Underline</td>
<td>1 1 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0011 Below Underline</td>
<td>0 1 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0100 Above Underline</td>
<td>0 0 1 0</td>
<td></td>
<td>Top Intersect</td>
</tr>
<tr>
<td>0100 Underline</td>
<td>0 0 0 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0100 Below Underline</td>
<td>0 1 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0101 Above Underline</td>
<td>0 1 0 0</td>
<td></td>
<td>Right Intersect</td>
</tr>
<tr>
<td>0101 Underline</td>
<td>1 1 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0101 Below Underline</td>
<td>0 1 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0110 Above Underline</td>
<td>0 1 0 0</td>
<td></td>
<td>Left Intersect</td>
</tr>
<tr>
<td>0110 Underline</td>
<td>1 0 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0110 Below Underline</td>
<td>0 1 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0111 Above Underline</td>
<td>0 1 0 0</td>
<td></td>
<td>Bottom Intersect</td>
</tr>
<tr>
<td>0111 Underline</td>
<td>0 0 0 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0111 Below Underline</td>
<td>0 0 1 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000 Above Underline</td>
<td>0 0 1 0</td>
<td></td>
<td>Horizontal Line</td>
</tr>
<tr>
<td>1000 Underline</td>
<td>0 0 0 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000 Below Underline</td>
<td>0 0 1 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1001 Above Underline</td>
<td>0 1 0 0</td>
<td></td>
<td>Vertical Line</td>
</tr>
<tr>
<td>1001 Underline</td>
<td>0 1 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1001 Below Underline</td>
<td>0 1 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1010 Above Underline</td>
<td>0 1 0 0</td>
<td></td>
<td>Crossed Lines</td>
</tr>
<tr>
<td>1010 Underline</td>
<td>0 0 0 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1010 Below Underline</td>
<td>0 1 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1011 Above Underline</td>
<td>0 0 0 0</td>
<td></td>
<td>Not Recommended</td>
</tr>
<tr>
<td>1011 Underline</td>
<td>0 0 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1011 Below Underline</td>
<td>0 0 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1100 Above Underline</td>
<td>0 0 1 0</td>
<td></td>
<td>Special Codes</td>
</tr>
<tr>
<td>1100 Underline</td>
<td>0 0 1 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1100 Below Underline</td>
<td>0 0 1 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1101 Above Underline</td>
<td>0 0 1 0</td>
<td></td>
<td>Illegal</td>
</tr>
<tr>
<td>1101 Underline</td>
<td>Undefined</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1101 Below Underline</td>
<td>Undefined</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1110 Above Underline</td>
<td>0 0 1 0</td>
<td></td>
<td>Illegal</td>
</tr>
<tr>
<td>1110 Underline</td>
<td>Undefined</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1110 Below Underline</td>
<td>Undefined</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1111 Above Underline</td>
<td>0 0 1 0</td>
<td></td>
<td>Illegal</td>
</tr>
<tr>
<td>1111 Underline</td>
<td>Undefined</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1111 Below Underline</td>
<td>Undefined</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Character Attribute Code 1011 is not recommended for normal operation. Since none of the attribute outputs are active, the character Generator will not be disabled, and an indeterminate character will be generated.

Character Attribute Codes 1101, 1110, and 1111 are illegal. Blinking is active when B = 1. Highlight is active when H = 1.

Figure 3-4. Character Attributes
Figure 3-2. 8275 Row Display

Figure 3-3. CRT System Block Diagram
vated. If the output of the light pen is presented to the 8275 LPEN input, the row and character position coordinates are stored in two 8275 internal registers. These registers can be read on command by the microprocessor.

**SPECIAL CODES** – Four special codes may be used to help reduce memory, software, or DMA overhead. These codes are placed in character positions in display memory.

1. **End of Row Code** —
   Activates VSP. VSP remains active until the end of the line is reached. While VSP is active, the screen is blanked.

2. **End of Row-Stop DMA Code** —
   Causes the DMA Control Logic to stop DMA for the rest of the row when it is written into the row buffer.

It affects the display in the same way as the End of Row Code.

3. **End of Screen Code** —
   Activates VSP. VSP remains active until the end of the frame is reached.

4. **End of Screen-Stop DMA Code** —
   Causes the DMA Control Logic to stop DMA for the rest of the frame when it is written into the row buffer. It affects the display in the same way as the End of Screen Code.

**PROGRAMMABLE DMA BURST CONTROL** — The 8275 can be programmed to request single byte DMA transfers or DMA burst transfers of 2, 4, or 8 characters per burst. The interval between bursts is also programmable. This allows the user to tailor his DMA overhead to fit his system needs.

---

**Figure 3.1. 8275 Block Diagram/Pin Configuration**
the 8275 to place a cursor on the screen and to control attribute functions. Attributes will be considered in the next section.

The HLG T (Highlight) output allows an attribute function to increase the CRT beam intensity to a level greater than normal. The fifth timing signal, RVV (Reverse Video) will, when enabled, cause the system video output to be inverted.

3. Special Functions –

**VISUAL ATTRIBUTES** – Visual attributes are special codes which, when retrieved from display memory by the 8275, affect the visual characteristics of a character position or field of characters. Two types of visual attributes exist, character attributes and field attributes.

**Character Attribute Codes:** Character attribute codes are codes that can be used to generate graphics symbols without the use of a character generator. This is accomplished by selectively activating the Line Attribute outputs (LA0–LA1), the Video Suppression output (VSP), and the Light Enable output. The dot timing logic uses these signals to generate the proper symbols. Character attributes can be programmed to blink or be highlighted individually. Blinking is accomplished with the Video Suppression output (VSP). Blink frequency is equal to the screen refresh frequency divided by 32. Highlighting is accomplished by activating the Highlight output (HGLT).

Character attributes were designed to produce the graphic symbols shown in Figure 3-4.

**Field Attribute Codes:** The field attributes are control codes which affect the visual characteristics for a field of characters, starting at the character following the field attribute code up to, and including, the character which precedes the next field attribute code, or up to the end of the frame.

There are six field attributes:

1. **Blink** – Characters following the code are caused to blink by activating the Video Suppression output (VSP). The blink frequency is equal to the screen refresh frequency divided by 32.

2. **Highlight** – Characters following the code are caused to be highlighted by activating the Highlight output (HGLT).

3. **Reverse Video** – Characters following the code are caused to appear in reverse video format by activating the Reverse Video output (RVV).

4. **Underline** – Characters following the code are caused to be underlined by activating the Light Enable output (LTEN).

5. **General Purpose** – There are two additional 8275 outputs which act as general purpose, independently programmable field attributes. These attributes may be used to select colors or perform other desired control functions.

The 8275 can be programmed to provide visible or invisible field attribute characters as shown in Figure 3-5. If the 8275 is programmed in the visible field attribute mode, all field attributes will occupy a position on the screen. They will appear as blanks caused by activation of the Video Suppression output (VSP). The chosen visual attributes are activated after this blanked character. If the 8275 is programmed in the invisible field attribute mode, the 8275 row buffer FIFOs are activated. The FIFOs effectively lengthen the row buffers by 16 characters, making room for up to 16 field attribute characters per display row. The FIFOs are 16 characters by 7 bits in size. When a field attribute is placed in the row buffer during DMA, the buffer input controller recognizes it and places the next character in the proper FIFO. When a field attribute is placed in the buffer output controller during display, it causes the controller to immediately put a character from the FIFO on the Character Code outputs (CC0–6).

The chosen attributes are also activated.

**LIGHT PEN DETECTION** – A light pen consists fundamentally of a switch and light sensor. When the light pen is pressed against the CRT screen, the switch enables the light sensor. When the raster sweep coincides with the light sensor position on the display, the light pen output is acti-
3. COMPONENT DESCRIPTION

3.1 8275

The block diagram and pin configuration for the 8275 Programmable CRT Controller are presented in Figure 3-1. The 8275 provides the following general capabilities:

1. **CRT Display Refreshing** – The 8275, having been programmed to a specific screen format, generates a series of DMA request signals, resulting in the transfer of a row of characters from display memory, via the 8257 DMA Controller, to the 8275's row buffers. The 8275 presents the character codes to an external character generator ROM. The 8275 character code outputs CCO–CC6 are used for this purpose. External dot timing logic is then utilized to transfer the parallel output data from the character generator ROM, serially, to the video input of the CRT. The character rows are displayed on the CRT one line at a time. Line count outputs LC0–LC3 are applied to the character generator ROM to perform the line selection function. The display process is graphically illustrated in Figure 3-2. The entire process is repeated for each display row. At the beginning of the last display row, the 8275 issues an interrupt via the INT output line. The 8275 interrupt output will normally be connected to the interrupt input of the system central processor. The interrupt causes the CPU to execute an interrupt service subroutine. The service subroutine typically re-initializes DMA controller parameters for the next display refresh cycle, polls the system keyboard controller, and/or executes other appropriate functions. A block diagram of a CRT system implemented with the 8275 CRT Controller is provided in Figure 3-3. Proper CRT refreshing requires that certain 8275 parameters be programmed prior to the beginning of display operation. The 8275 has two types of programming registers, the Command Registers (CREG) and the Parameter Registers (PREG). It also has a Status Register (SREG). The Command Registers may only be written to and the Status Registers may only be read. The 8275 expects to receive a command followed by a sequence of from 0 to 4 parameters, depending on the command. The 8275 instruction set consists of 8 commands:

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>NO. OF PARAMETER BYTES</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESET</td>
<td>4</td>
<td>Display format parameters required</td>
</tr>
<tr>
<td>START DISPLAY</td>
<td>0</td>
<td>DMA operation parameters included in command</td>
</tr>
<tr>
<td>STOP DISPLAY</td>
<td>0</td>
<td>--</td>
</tr>
<tr>
<td>READ LIGHT PEN</td>
<td>2</td>
<td>--</td>
</tr>
<tr>
<td>LOAD CURSOR</td>
<td>2</td>
<td>Cursor X,Y position parameters required</td>
</tr>
<tr>
<td>ENABLE INTERRUPT</td>
<td>0</td>
<td>--</td>
</tr>
<tr>
<td>DISABLE INTERRUPT</td>
<td>0</td>
<td>--</td>
</tr>
<tr>
<td>PRESET COUNTERS</td>
<td>0</td>
<td>Clears all internal counters</td>
</tr>
</tbody>
</table>

In order to establish the format of the display, the 8275 provides a number of user programmable display format parameters. Display formats having from 1 to 80 characters per row, 1 to 64 rows per screen, and from 1 to 16 horizontal lines per row are available.

In addition to transferring characters from memory to the CRT screen, the 8275 features cursor position control. The cursor position may be programmed, via X and Y cursor position registers, to any character position on the display. The user may select from 4 cursor formats. Blinking or non-blinking underline and reverse video block cursors are available.

2. **CRT Timing** – The 8275 provides two timing outputs, HRTC and VRTC, which are utilized in synchronizing CRT horizontal and vertical oscillators to the 8275 refresh cycle. In addition, whenever HRTC or VRTC are active, a third timing output, VSP (Video Suppress) is true, providing a blanking signal to the dot timing logic. The dot timing logic will normally inhibit the video output to the CRT during the time when video suppress signal is true. An additional timing output, LTEN (Light Enable) is used to provide the ability to force the video output high regardless of the state of VSP. This feature is utilized by
2.4 CRT TERMINAL IMPLEMENTATION

A typical microprocessor-based CRT terminal is presented in block diagram form in Figure 2-5. The terminal consists of the CRT monitor, monitor electronics, memory for storing the information to be displayed, a serial communication device, keyboard, keyboard interface device, CRT controller, central processor and associated program memory, and a DMA device. The primary function of the CRT controller is to refresh the display. It does this by controlling the periodic transfer of information from display memory to the CRT screen. The central processor unit (CPU) coordinates the transfer of information to and from the terminal peripheral devices and external devices. When information from an external device is received by the terminal, the central processor performs character recognition and handling functions, display memory management functions, and cursor control functions. The CPU also interrogates the keyboard interface device. If a key depression is detected by the keyboard interface device, the CPU responds by transmitting the ASCII character representing the key to the terminal serial output line via the serial communication device. A direct memory access (DMA) device is required in the system to effect the necessary memory to screen data transfer rate.

The CRT terminal control functions under consideration may be implemented with LSI devices at a considerable cost savings over earlier terminal designs using MSI and SSI components. This cost savings is complemented by an increase in the number of features which can be incorporated in terminal designs. The additional features stem from the programmable nature of the devices. In addition, utilizing a microprocessor as the terminal controller allows considerable intelligence to be built into the terminal for decision making, computational, and control functions. The design example presented in Section 4 of the application note illustrates the use of the 8275 Programmable CRT Controller and 8279 Keyboard Controller in a typical terminal design. In the following section, the 8275 and 8279 are considered in depth.

![CRT Terminal Block Diagram](Figure 2.5)
2.3 CRT TERMINAL DESCRIPTION

A CRT terminal consists basically of a CRT monitor, monitor control electronics, memory for storing display information, logic to control information transfer to and from external devices and between internal devices, and a keyboard. The fundamental operations performed by a CRT terminal consist of the display of information contained in internal memory on the CRT screen, communication with manual data entry devices such as keyboards or light pens, and communication with external intelligent devices such as computers or data communication terminals. Typical CRT terminal communication functions are illustrated in Figure 2-4.
1. INTRODUCTION

The purpose of this application note is to provide the reader with the conceptual and factual tools needed to apply the 8275 Programmable CRT Controller and 8279 Programmable Keyboard/Display Interface in CRT system design. The 8275 Controller is designed to interface CRT raster scan displays with Intel® Microcomputer Products. Its primary functions include refreshing the CRT display by buffering information from display memory and generating horizontal and vertical timing signals used for CRT synchronization. The programmable features of the 8275 allow it to be interfaced to almost any raster scan display with a minimum of external hardware. In addition, visual attribute features allow the implementation of specialized graphic display functions and display enhancement operations. The 8279 Keyboard Interface provides key scanning, debounce, and buffering features required for interfacing CRT terminal keyboards to the system processor. Two key or N-key rollover is provided. The use of these devices in a microcomputer based CRT terminal yields substantial savings in component count, printed circuit board area, and power consumption.

The application note is divided into five sections:

1. Introduction
2. CRT System Design Concepts
3. Component Description
4. CRT System Design Example
5. Appendix

Readers desiring an overview of CRT system design should consider reading the first three sections of the application note. Individuals requiring an in-depth knowledge of CRT system design should read the first three sections, then proceed to the design example. The design example consists of a description of the design of a complete CRT terminal. Both hardware and software aspects of the design are included. It will be assumed in Section 4 that the reader is familiar with the 8275, 8279, and 8257 data sheets, and the operation of the 8080A microprocessor.

2. CRT SYSTEM DESIGN CONCEPTS

2.1 CRT OPERATION

In order to fully understand the CRT terminal design process, it is necessary to consider the fundamentals of CRT operation. A typical CRT Monitor is shown in Figure 2-1. The CRT consists of an evacuated glass structure having a phosphorescent coating on the inner surface of the rectangular frontal region (screen). A filament contained in the narrow cylindrical region (neck) of the CRT heats the cathode, causing the cathode to give off electrons by thermionic emission. Heating is accomplished by applying a low voltage source across the filament leads. A high voltage source applied between the cathode and the screen electrode (anode) accelerates the electrons toward the screen. The electron beam, upon striking the phosphorescent inner surface of the screen, produces light. To control the point at which the beam strikes the screen, two primary deflection techniques are utilized. The first technique, electromagnetic deflection, involves applying a current through a deflection coil placed around the neck of the CRT. The resulting magnetic field forces the electron beam to be deflected in proportion to the magnitude of the applied current. Electrostatic deflection involves placing deflection electrodes in the neck of the CRT perpendicular to the electron beam. An applied voltage changes the position of the beam accordingly.

2.2 MONITOR OPERATION

A CRT monitor consists of a CRT and the electronics required for positioning the beam in the desired manner. A block diagram of the control electronics contained within a typical CRT monitor is provided in Figure 2-2.

The horizontal oscillator is designed to move the electron beam horizontally across the CRT screen and then return the beam rapidly to its original position. As the beam is moved horizontally, the vertical oscillator causes the beam to be deflected vertically. The net result of these operations is to move the beam in a manner shown in Figure 2-3. If the intensity of the electron beam is modulated in a controlled manner as the beam sweeps across the screen, it is possible to display pictorial information on the CRT screen surface. It will be assumed that the monitor in question will be used for displaying alphanumeric characters or graphic symbols. In this case, the electron beam will be turned on to display a light region on the screen and turned off to display a dark region. Display information appearing at the video input to the CRT is applied through the video amplifier to a control grid located in the neck of the CRT. The magnitude of the video signal determines whether the electron beam will be on or off.
Contents

1. INTRODUCTION .............................................. 1

2. CRT SYSTEM DESIGN CONCEPTS. ......................... 1
   2.1 CRT OPERATION ....................................... 1
   2.2 MONITOR OPERATION .................................. 1
   2.3 CRT TERMINAL DESCRIPTION ....................... 2
   2.4 CRT TERMINAL IMPLEMENTATION ................... 3

3. COMPONENT DESCRIPTION ................................. 4
   3.1 8275 .................................................. 4
   3.2 8279 .................................................. 9

4. CRT SYSTEM DESIGN EXAMPLE ........................... 10
   4.1 SCOPE OF THE PROJECT ............................. 10
   4.2 SYSTEM SPECIFICATIONS ......................... 11
   4.3 SYSTEM HARDWARE DESIGN ......................... 11
      4.3.1 General Considerations ..................... 11
      4.3.2 Operation .................................... 12
      4.3.3 System Timing ................................ 13
      4.3.4 Dot Timing Logic ............................. 15
      4.3.5 Keyboard Interface Design ................. 17
      4.3.6 System Memory Design ..................... 17
   4.4 SYSTEM SOFTWARE DESIGN ......................... 18
      4.4.1 General Considerations ..................... 18
      4.4.2 Software Development ....................... 18
      4.4.3 Operation .................................... 18
      4.4.4 System Subroutines ......................... 22

APPENDIX 5.1 – CRT TERMINAL SCHEMATICS
   CPU Section .......................................... 31
   Memory Section ....................................... 33
   Peripherals Section ................................ 35
   Dot Timing Logic Section ......................... 37
   Serial Communications Section ................. 39

APPENDIX 5.2 – ESCAPE/CONTROL/DISPLAY CHARACTER SUMMARY ................. 40

APPENDIX 5.3 – SUBROUTINE INTER-RELATIONSHIPS ................. 41

APPENDIX 5.4 – SOFTWARE TIMING ......................... 42

APPENDIX 5.5 – VISUAL ATTRIBUTE IMPLEMENTATION CONSIDERATIONS .......... 42

APPENDIX 5.6 – SOFTWARE LISTINGS ....................... 45

Intel Corporation assumes no responsibility for the use of any circuitry embodied in an Intel product. No other circuit patent licenses are implied.
RELATED INTEL PUBLICATIONS

"MCS-80™ USER'S MANUAL"

"MEMORY DESIGN HANDBOOK"

"PERIPHERALS DATA NOTEBOOK"
CRT Terminal Design Using The Intel® 8275 and 8279

John Murray and George Alexy
Microcomputer Applications

November 1977