Bibliography

1. International Business Machines Corporation, IBM 3270 Information Display System - 3274 Control Unit Description and Programmer's Guide, GA23-0061-0, November 1980.

2. ____. IBM 3270 Information Display System - Component Description, GA27-2749-10, February 1980.

3. ____. General Information - Binary Synchronous Communications, GA27-3004-2, October 1970.

4. ____. IBM 3270 Information Display System Data Stream - Programmer's Reference, GA23-0059-3, November 1986.

5. ____. IBM 3178 Display Station Description, GA18-2127-1, August 1983.

6. Digital Communications Associates, Inc., IRMA, 1983.

APPENDIX

Appendix A

Scan Code and Buffer Code for IBM 3270 Coaxial Transmission.

Scan Code is a position of Keyboard pressed, which is transmited from IBM 3278 Display Station to IBM 3274 Control Unit.

Buffer Code is a display character code used in transmission between IBM 3278 Display Station and IBM 3274 Control Unit.

	Hexadecimal			
Character	Scan Code	Buffer Code	EBCDIC	ASCII
	10	10	40	20
	1B	1B	4A	-
.	32	32	4B	2E
<	09	09	4C	3C
(	29	0D	4D	28
+	11	35	4E	2B
	21	16	4F	-
&	27	30	50	26
!	1B	19	5A	21
$	24	1A	5B	24
*	28	BF	5C	2A
)	20	0C	5D	29
;	7E	BE	5E	3B
	26	36	5F	-
-	30	31	60	2D
/	14	14	61	2F
	15	17	6A	7C
,	33	33	6B	2C
%	25	2E	6C	25
_	30	2F	6D	5F
>	09	08	6E	3E
?	14	18	6F	3F
'	3D	3D	79	60
:	7E	34	7A	3A
#	23	2C	7B	23

Character	Hexadecimal Scan Code	Buffer Code	EBCDIC	ASCII
@	22	2D	7C	40
'	12	12	7D	27
=	11	11	7E	3D
"	12	13	7F	22
~	3D	3B	A1	7E
{	0F	0F	C0	7B
}	0F	0E	D0	7D
\	15	15	E0	5C
a	60	80	81	61
b	61	81	82	62
c	62	82	83	63
d	63	83	84	64
e	64	84	85	65
f	65	85	86	66
g	66	86	87	67
h	67	87	88	68
i	68	88	89	69
j	69	89	91	6A
k	6A	8A	92	6B
l	6B	8B	93	6C
m	6C	8C	94	6D
n	6D	8D	95	6E
o	6E	8E	96	6F
p	6F	8F	97	70
q	70	90	98	71
r	71	91	99	72
s	72	92	A2	73
t	73	93	A3	74
u	74	94	A4	75
v	75	95	A5	76
w	76	96	A6	77
x	77	97	A7	78
y	78	98	A8	79
z	79	99	A9	7A
A	60	A0	C1	41
B	61	A1	C2	42
C	62	A2	C3	43
D	63	A3	C4	44
E	64	A4	C5	45
F	65	A5	C6	46
G	66	A6	C7	47
H	67	A7	C8	48
I	68	A8	C9	49
J	69	A9	D1	4A
K	6A	AA	D2	4B
L	6B	AB	D3	4C
M	6C	AC	D4	4D
N	6D	AD	D5	4E
O	6E	AE	D6	4F
P	6F	AF	D7	50
Q	70	B0	D8	51

Character	Hexadecimal Scan Code	Buffer Code	EBCDIC	ASCII
R	71	B1	D9	52
S	72	B2	E2	53
T	73	B3	E3	54
U	74	B4	E4	55
V	75	B5	E5	56
W	76	B6	E6	57
X	77	B7	E7	58
Y	78	B8	E8	59
Z	79	B9	E9	5A
0	20	20	F0	30
1	21	21	F1	31
2	22	22	F2	32
3	23	23	F3	33
4	24	24	F4	34
5	25	25	F5	35
6	26	26	F6	36
7	27	27	F7	37
8	28	28	F8	38
9	29	29	F9	39
Dup( )	5F	9F	-	-
Fm( )	5E	9E	-	-
Ins( )	0C	-	-	-
Del( )	0D	-	-	-
Enter	18	-	-	-
Reset	34	-	-	-
Bs( )	31	-	-	-
CurLt( )	16	-	-	-
CurRt( )	1A	-	-	-
CurUp( )	0E	-	-	-
CurDn( )	13	-	-	-
NewLn( )	08	-	-	-
BkTab( )	36	-	-	-
Tab( )	35	-	-	-
Attn	50	-	-	-
CurSel	51	-	-	-
CurBlink	54	-	-	-
Eof	55	-	-	-
Ident	56	-	-	-
Test	57	-	-	-
PF13	40	-	-	-
PF14	41	-	-	-
PF15	42	-	-	-
PF16	43	-	-	-
PF17	44	-	-	-
PF18	45	-	-	-
PF19	46	-	-	-
PF20	47	-	-	-
PF21	48	-	-	-
PF22	49	-	-	-
PF23	4A	-	-	-
PF24	4B	-	-	-

When pressing the following key, Scan Code is sent when the key is pressed and released.

Keyboard	Press Scan Code (Hexadecimal)	Release Scan Code (Hexadecimal)
Alt	4F	CF
Shift	4D	CD
Caplock	4C	CC

Note : The Scan code is obtained from Typewriter Keyboard with 24 PF keys.

Appendix B

Buffer Code for IBM 3270 Coaxial Transmission with IBM Thai language EBCDIC table B.

Character	Hexadecimal Buffer Code	EBCDIC	ASCII(TISO)
ก	54	80	A1
ข	80	81	A2
ช	81	82	A3
ค	82	83	A4
ฅ	83	84	A5
ฆ	84	85	A6
ง	85	86	A7
จ	86	87	A8
ฉ	87	88	A9
ช	88	89	AA
ซ	55	8A	AB
ฌ	56	8B	AC
ญ	57	8C	AD
ฎ	58	8D	AE
ฏ	59	8E	AF
ฐ	5A	8F	B0
ฑ	5B	90	B1
ฒ	89	91	B2
ณ	8A	92	B3
ด	8B	93	B4
ต	8C	94	B5
ถ	8D	95	B6
ท	8E	96	B7
ธ	8F	97	B8
น	90	98	B9
บ	91	99	BA
ป	5C	9A	BB
ผ	5D	9B	BC
ฝ	5E	9C	BD
พ	5F	9D	BE
ฟ	60	9E	BF
ภ	61	9F	C0
ม	62	A0	C1
ย	3B	A1	C2
ร	92	A2	C3
ฤ	93	A3	C4
ล	94	A4	C5
ฦ	95	A5	C6
ว	96	A6	C7

Character	Hexadecimal Buffer Code	EBCDIC	ASCII(TISO)
ศ	97	A7	C8
ษ	98	A8	C9
ส	99	A9	CA
ห	63	AA	CB
ฬ	64	AB	CC
อ	65	AC	CD
ฮ	66	AD	CE
ะ	67	AE	D0
ั	68	AF	D1
า	69	B0	D2
ำ	6A	B1	D3
ิ	6B	B2	D4
ี	6C	B3	D5
ึ	6D	B4	D6
ื	6E	B5	D7
ุ	6F	B6	D8
ู	70	B7	D9
เ	71	B8	E0
แ	72	B9	E1
โ	73	BA	E2
ใ	74	BB	E3
ไ	75	BC	E4
ๅ	76	BD	E5
็	77	BE	E8
่	78	BF	E9
้	79	CA	EA
๊	7A	CB	EB
๋	7B	CC	E7
์	7C	CD	EC
ๆ	-	CE	E6
	-	CF	CF

Note : This result is obtained from IBM 3274-41D Control Unit.

Appendix C

Buffer address Code.

Row	Column	High Byte (Hexadecimal)	Low Byte (Hexadecimal)
1	1	00	50
2	1	00	A0
3	1	00	F0
4	1	01	40
5	1	01	90
6	1	01	E0
7	1	02	30
8	1	02	80
9	1	02	D0
10	1	03	20
11	1	03	70
12	1	03	C0
13	1	04	10
14	1	04	60
15	1	04	B0
16	1	05	00
17	1	05	50
18	1	05	A0
19	1	05	F0
20	1	06	40
21	1	06	90
22	1	06	E0
23	1	07	30
24	1	07	80
25	1	00	00

Appendix D

Interface Command Code.

Command	Command Code	
	B5-B9	Hexadecimal
Poll	00001	01
Reset	00010	02
Read Data	00011	03
Load Address Counter(High Byte)	00100	04
Read Address Counter(High Byte)	00101	05
Clear	00110	06
Read Terminal ID	01001	09
Load Control Register	01010	0A
Read Multiple	01011	0B
Write Data	01100	0C
Read Status	01101	0D
Insert	01110	0E
Search Forward	10000	10
Poll Acknowledge	10001	11
Search Backward	10010	12
Load Address Counter(Low Byte)	10100	14
Read Address Counter(Low Byte)	10101	15
Load Mask	10110	16
Load Secondary Control Register	11010	1A

Appendix E

Data Captured Without Data bit 10 and bit 11

The following data is captured using 3270CIB without the data bit 10, and bit 11. The data is captured during the power on of IBM 3178 Display Station.

```
0D91:02F0  01 01 01 01 01 01 01 01-01 01 01 01 01 01 01 01
0D91:0300  01 01 01 01 01 02 11 00-01 00 04 00 00 00 14 00
0D91:0310  00 00 03 00 03 00 03 00-03 00 03 00 03 00 03 00
0D91:0320  03 00 03 00 03 00 03 00-03 00 03 00 03 00 03 00
0D91:0330  03 00 04 00 00 00 14 00-30 00 03 00 03 00 03 00
0D91:0340  03 00 03 00 03 00 03 00-03 00 03 00 03 00 03 00
0D91:0350  03 00 03 00 03 00 03 00-03 00 01 00 09 A4 09 A4
0D91:0360  01 00 01 00 09 A4 09 A4-0D 20 01 00 04 00 00 00
0D91:0370  14 00 00 00 07 00 04 00-00 00 14 00 00 00 07 00
0D91:0380  04 00 00 00 14 00 00 00-07 00 04 00 00 00 14 00
0D91:0390  00 00 07 00 04 00 00 00-14 00 00 00 07 00 04 00
0D91:03A0  00 00 14 00 00 00 07 00-04 00 00 00 14 00 00 00
0D91:03B0  07 00 25 00 01 00 11 00-01 00 01 00 25 00 01 00
0D91:03C0  11 00 01 00 01 00 01 00-25 00 01 00 11 00 01 00
0D91:03D0  01 00 25 00 01 00 11 00-01 00 01 00 25 00 01 00
0D91:03E0  11 00 01 00 01 00 01 00-25 00 01 00 11 00 01 00
0D91:03F0  01 00 25 00 01 00 11 00-01 00 01 00 35 00 01 00
0D91:0400  11 00 01 00 01 00 35 00-01 00 11 00 01 00 01 00
0D91:0410  01 00 35 00 01 00 11 00-01 00 01 00 35 00 01 00
0D91:0420  11 00 01 00 01 00 35 00-01 00 11 00 01 00 01 00
0D91:0430  01 00 35 00 01 00 11 00-01 00 01 00 35 00 01 00
0D91:0440  11 00 01 00 01 00 45 00-01 00 11 00 01 00 01 00
0D91:0450  45 00 01 00 11 00 01 00-01 00 01 00 45 00 01 00
0D91:0460  11 00 01 00 01 00 45 00-01 00 11 00 01 00 01 00
0D91:0470  45 00 01 00 11 00 01 00-01 00 01 00 45 00 01 00
0D91:0480  11 00 01 00 01 00 45 00-01 00 11 00 01 00 01 00
0D91:0490  55 00 01 00 11 00 01 00-01 00 55 00 01 00 11 00
0D91:04A0  01 00 01 00 01 00 55 00-01 00 11 00 01 00 01 00
0D91:04B0  55 00 01 00 11 00 01 00-01 00 55 00 01 00 11 00
0D91:04C0  01 00 01 00 01 00 55 00-01 00 11 00 01 00 01 00
0D91:04D0  55 00 01 00 11 00 01 00-01 00 65 00 01 00 11 00
0D91:04E0  01 00 01 00 65 00 01 00-11 00 01 00 01 00 01 00
0D91:04F0  65 00 01 00 11 00 01 00-01 00 65 00 01 00 11 00
0D91:0500  01 00 01 00 01 00 65 00-01 00 11 00 01 00 01 00
0D91:0510  65 00 01 00 11 00 01 00-01 00 65 00 01 00 11 00
0D91:0520  01 00 01 00 75 00 01 00-11 00 01 00 01 00 75 00
0D91:0530  01 00 11 00 01 00 01 00-01 00 75 00 01 00 11 00
0D91:0540  01 00 01 00 75 00 01 00-11 00 01 00 01 00 01 00
```

```
0D91:0550  75 00 01 00 11 00 01 00-01 00 75 00 01 00 11 00
0D91:0560  01 00 01 00 75 00 01 00-11 00 01 00 01 00 85 00
0D91:0570  01 00 11 00 01 00 01 00-85 00 01 00 11 00 01 00
0D91:0580  01 00 01 00 85 00 01 00-11 00 01 00 01 00 85 00
0D91:0590  01 00 11 00 01 00 01 00-01 00 85 00 01 00 11 00
0D91:05A0  01 00 01 00 85 00 01 00-11 00 01 00 01 00 85 00
0D91:05B0  01 00 11 00 01 00 01 00-95 00 01 00 11 00 01 00
0D91:05C0  01 00 95 00 01 00 11 00-01 00 01 00 01 00 95 00
0D91:05D0  01 00 11 00 01 00 01 00-95 00 01 00 11 00 01 00
0D91:05E0  01 00 01 00 95 00 01 00-11 00 01 00 01 00 95 00
0D91:05F0  01 00 11 00 01 00 01 00-95 00 01 00 11 00 01 00
0D91:0600  01 00 A5 00 01 00 11 00-01 00 01 00 A5 00 01 00
0D91:0610  11 00 01 00 01 00 01 00-A5 00 01 00 11 00 01 00
0D91:0620  01 00 A5 00 01 00 11 00-01 00 01 00 01 00 A5 00
0D91:0630  01 00 11 00 01 00 01 00-A5 00 01 00 11 00 01 00
0D91:0640  01 00 A5 00 01 00 11 00-01 00 01 00 B5 00 01 00
0D91:0650  11 00 01 00 01 00 B5 00-01 00 11 00 01 00 01 00
0D91:0660  01 00 B5 00 01 00 11 00-01 00 01 00 B5 00 01 00
0D91:0670  11 00 01 00 01 00 01 00-B5 00 01 00 11 00 01 00
0D91:0680  01 00 B5 00 01 00 11 00-01 00 01 00 B5 00 01 00
0D91:0690  11 00 01 00 01 00 C5 00-01 00 11 00 01 00 01 00
0D91:06A0  C5 00 01 00 11 00 01 00-01 00 01 00 C5 00 01 00
0D91:06B0  11 00 01 00 01 00 C5 00-01 00 11 00 01 00 01 00
0D91:06C0  01 00 C5 00 01 00 11 00-01 00 01 00 C5 00 01 00
0D91:06D0  11 00 01 00 01 00 01 00-C5 00 01 00 11 00 01 00
0D91:06E0  D5 00 01 00 11 00 01 00-01 00 01 00 D5 00 01 00
0D91:06F0  11 00 01 00 01 00 D5 00-01 00 11 00 01 00 01 00
0D91:0700  D5 00 01 00 11 00 01 00-01 00 01 00 D5 00 01 00
0D91:0710  11 00 01 00 01 00 D5 00-01 00 11 00 01 00 01 00
0D91:0720  01 00 D5 00 01 00 11 00-01 00 E5 00 01 00 11 00
0D91:0730  01 00 01 00 01 00 E5 00-01 00 11 00 01 00 01 00
0D91:0740  E5 00 01 00 11 00 01 00-01 00 E5 00 01 00 11 00
0D91:0750  01 00 01 00 01 00 E5 00-01 00 11 00 01 00 01 00
0D91:0760  E5 00 01 00 11 00 01 00-01 00 01 00 E5 00 01 00
0D91:0770  11 00 01 00 01 00 04 00-00 00 14 00 00 00 0C 00
0D91:0780  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0D91:0790  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0D91:07A0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0D91:07B0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0D91:07C0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0D91:07D0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0D91:07E0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0D91:07F0  01 00 04 00 00 00 14 00-29 00 0C 00 00 00 00 00
0D91:0800  04 00 00 00 14 00 50 00-41 00 04 00 00 00 14 00
0D91:0810  50 00 16 00 00 00 06 00-FF 00 01 00 01 00 01 00
0D91:0820  01 00 01 00 01 00 01 00-01 01 11 00 01 00 05 00
0D91:0830  15 00 01 00 04 00 00 00-14 00 00 00 03 00 04 00
0D91:0840  00 00 14 00 50 00 01 00-04 00 07 00 14 00 D0 00
0D91:0850  0C 00 C1 00 04 00 00 00-14 00 50 00 01 00 04 00
0D91:0860  00 00 14 00 23 00 0C 00-00 00 00 00 00 00 00 00
0D91:0870  04 00 00 00 14 00 50 00-41 00 04 00 00 00 14 00
0D91:0880  29 00 0C 00 00 00 00 00-04 00 00 00 14 00 50 00
0D91:0890  41 00 01 00 0A 04 00 04-00 00 00 14 00 08 00 0C
0D91:08A0  00 F6 00 F4 F5 00 00 00-00 00 00 04 00 00 00 14
0D91:08B0  00 50 00 41 00 0A 10 00-0A 04 00 04 00 00 00 14
```

```
0D91:08C0  00 01 00 0C 00 00 00 00-00 00 00 04 00 00 00 14
0D91:08D0  00 50 00 41 00 0A 10 00-0A 04 00 04 00 00 00 14
0D91:08E0  00 00 00 0C 00 FC 00 04-00 00 00 14 00 50 00 41
0D91:08F0  00 0A 10 00 01 00 0A 04-00 04 00 00 00 14 00 01
0D91:0900  00 0C 00 00 00 00 00 00-00 04 00 00 00 14 00 50
0D91:0910  00 C1 00 0A 10 00 0A 04-00 04 00 00 00 14 00 08
0D91:0920  00 0C 00 00 00 00 00 00-00 00 00 00 00 00 04 00
0D91:0930  00 00 14 00 50 00 C1 00-0A 10 00 01 00 0A 04 00
0D91:0940  04 00 00 00 14 00 08 00-0C 00 00 00 00 00 00 00
0D91:0950  00 00 00 00 00 04 00 00-00 14 00 50 00 C1 00 0A
0D91:0960  10 00 0A 04 00 04 00 00-00 14 00 3B 00 0C 00 D7
0D91:0970  F1 D7 20 27 00 04 00 00-00 14 00 50 00 C1 00 0A
0D91:0980  10 00 0A 04 00 0D 20 04-00 00 00 14 00 50 00 01
0D91:0990  00 0A 10 00 01 00 01 00-01 00 01 00 01 00 01 00
0D91:09A0  01 00 01 00 01 00 01 00-01 00 01 00 01 00 01 00
```

Appendix F

Data Captured With Data bit 10 and bit 11

The following data is captured using 3270CIB with the data bit 10, and bit 11. The data is captured during the power on of IBM 3178 Display Station.

```
0D91:04D0  01 03 01 03 01 03 01 03-01 03 01 03 01 03 01 03
0D91:04E0  01 03 01 03 01 03 01 03-01 03 01 03 01 03 01 03
0D91:04F0  01 03 01 03 01 02 02 03-11 00 00 03 01 00 00 03
0D91:0500  04 00 00 02 00 00 00 03-14 00 00 02 00 00 00 03
0D91:0510  03 02 00 03 03 02 00 03-03 02 00 03 03 02 00 03
0D91:0520  03 02 00 03 03 02 00 03-03 02 00 03 03 02 00 03
0D91:0530  03 02 00 03 03 02 00 03-03 02 00 03 03 02 00 03
0D91:0540  03 02 00 03 03 02 00 03-03 02 00 03 03 02 00 03
0D91:0550  04 00 00 02 00 00 00 03-14 00 00 02 30 00 00 03
0D91:0560  03 02 00 03 03 02 00 03-03 02 00 03 03 02 00 03
0D91:0570  03 02 00 03 03 02 00 03-03 02 00 03 03 02 00 03
0D91:0580  03 02 00 03 03 02 00 03-03 02 00 03 03 02 00 03
0D91:0590  03 02 00 03 03 02 00 03-03 02 00 03 03 02 00 03
0D91:05A0  01 00 00 03 09 00 A4 03-09 00 A4 03 01 00 00 03
0D91:05B0  01 00 00 03 09 00 A4 03-09 00 A4 03 0D 00 20 03
0D91:05C0  01 00 00 03 04 00 00 02-00 00 00 03 14 00 00 02
0D91:05D0  00 00 00 03 07 00 00 03-04 00 00 02 00 00 00 03
0D91:05E0  14 00 00 02 00 00 00 03-07 00 00 03 04 00 00 02
0D91:05F0  00 00 00 03 14 00 00 02-00 00 00 03 07 00 00 03
0D91:0600  04 00 00 02 00 00 00 03-14 00 00 02 00 00 00 03
0D91:0610  07 00 00 03 04 00 00 02-00 00 00 03 14 00 00 02
0D91:0620  00 00 00 03 07 00 00 03-04 00 00 02 00 00 00 03
0D91:0630  14 00 00 02 00 00 00 03-07 00 00 03 04 00 00 02
0D91:0640  00 00 00 03 14 00 00 02-00 00 00 03 07 00 00 01
0D91:0650  25 00 00 03 01 01 00 03-11 00 00 03 01 00 00 03
0D91:0660  01 00 00 01 25 00 00 03-01 01 00 03 11 00 00 03
0D91:0670  01 00 00 03 01 00 00 03-01 00 00 01 25 00 00 03
0D91:0680  01 01 00 03 11 00 00 03-01 00 00 03 01 00 00 01
0D91:0690  25 00 00 03 01 01 00 03-11 00 00 03 01 00 00 03
0D91:06A0  01 00 00 01 25 00 00 03-01 01 00 03 11 00 00 03
0D91:06B0  01 00 00 03 01 00 00 03-01 00 00 01 25 00 00 03
0D91:06C0  01 01 00 03 11 00 00 03-01 00 00 03 01 00 00 01
0D91:06D0  25 00 00 03 01 01 00 03-11 00 00 03 01 00 00 03
0D91:06E0  01 00 00 03 35 00 00 03-01 01 00 03 11 00 00 03
0D91:06F0  01 00 00 03 01 00 00 03-35 00 00 03 01 01 00 03
0D91:0700  11 00 00 03 01 00 00 03-01 00 00 03 01 00 00 03
0D91:0710  35 00 00 03 01 01 00 03-11 00 00 03 01 00 00 03
0D91:0720  01 00 00 03 35 00 00 03-01 01 00 03 11 00 00 03
```

```
0D91:0730  01 00 00 03 01 00 00 03-35 00 00 03 01 01 00 03
0D91:0740  11 00 00 03 01 00 00 03-01 00 00 03 01 00 00 03
0D91:0750  35 00 00 03 01 01 00 03-11 00 00 03 01 00 00 03
0D91:0760  01 00 00 03 35 00 00 03-01 01 00 03 11 00 00 03
0D91:0770  01 00 00 03 01 00 00 01-45 00 00 03 01 01 00 03
0D91:0780  01 03 01 00 00 01 45 00-00 03 01 01 00 03 11 00
0D91:0790  00 03 01 00 00 03 01 00-00 03 01 00 00 01 45 00
0D91:07A0  00 03 01 01 00 03 11 00-00 03 01 00 00 03 01 00
0D91:07B0  00 01 45 00 00 03 01 01-00 03 11 00 00 03 01 00
0D91:07C0  00 03 01 00 00 01 45 00-00 03 01 01 00 03 11 00
0D91:07D0  00 03 01 00 00 03 01 00-00 03 01 00 00 01 45 00
0D91:07E0  00 03 01 01 00 03 11 00-00 03 01 00 00 03 01 00
0D91:07F0  00 01 45 00 00 03 01 01-00 03 11 00 00 03 01 00
0D91:0800  00 03 01 00 00 03 55 00-00 03 01 01 00 03 11 00
0D91:0810  00 03 01 00 00 03 01 00-00 03 55 00 00 03 01 01
0D91:0820  00 03 11 00 00 03 01 00-00 03 01 00 00 03 01 00
0D91:0830  00 03 55 00 00 03 01 01-00 03 11 00 00 03 01 00
0D91:0840  00 03 01 00 00 03 55 00-00 03 01 01 00 03 11 00
0D91:0850  00 03 01 00 00 03 01 00-00 03 55 00 00 03 01 01
0D91:0860  00 03 11 00 00 03 01 00-00 03 01 00 00 03 01 00
0D91:0870  00 03 55 00 00 03 01 01-00 03 11 00 00 03 01 00
0D91:0880  00 03 01 00 00 03 55 00-00 03 01 01 00 03 11 00
0D91:0890  00 03 01 00 00 03 01 00-00 03 65 00 00 03 01 01
0D91:08A0  00 03 11 00 00 03 01 00-00 03 01 00 00 03 65 00
0D91:08B0  00 03 01 01 00 03 11 00-00 03 01 00 00 03 01 00
0D91:08C0  00 03 01 00 00 03 65 00-00 03 01 01 00 03 11 00
0D91:08D0  00 03 01 00 00 03 01 00-00 03 65 00 00 03 01 01
0D91:08E0  00 03 11 00 00 03 01 00-00 03 01 00 00 03 65 00
0D91:08F0  00 03 01 01 00 03 11 00-00 03 01 00 00 03 01 00
0D91:0900  00 03 01 00 00 03 65 00-00 03 01 01 00 03 11 00
0D91:0910  00 03 01 00 00 03 01 00-00 03 65 00 00 03 01 01
0D91:0920  00 03 11 00 00 03 01 00-00 03 01 00 00 01 75 00
0D91:0930  00 03 01 01 00 03 11 00-00 03 01 00 00 03 01 00
0D91:0940  00 01 75 00 00 03 01 01-00 03 11 00 00 03 01 00
0D91:0950  00 03 01 00 00 03 01 00-00 01 75 00 00 03 01 01
0D91:0960  00 03 11 00 00 03 01 00-00 03 01 00 00 01 75 00
0D91:0970  00 03 01 01 00 03 11 00-00 03 01 00 00 03 01 00
0D91:0980  00 01 75 00 00 03 01 01-00 03 11 00 00 03 01 00
0D91:0990  00 03 01 00 00 03 01 00-00 01 75 00 00 03 01 01
0D91:09A0  00 03 11 00 00 03 01 00-00 03 01 00 00 01 75 00
0D91:09B0  00 03 01 01 00 03 11 00-00 03 01 00 00 03 01 00
0D91:09C0  00 01 85 00 00 03 01 01-00 03 11 00 00 03 01 00
0D91:09D0  00 03 01 00 00 01 85 00-00 03 01 01 00 03 11 00
0D91:09E0  00 03 01 00 00 03 01 00-00 03 01 00 00 01 85 00
0D91:09F0  00 03 01 01 00 03 11 00-00 03 01 00 00 03 01 00
0D91:0A00  00 01 85 00 00 03 01 01-00 03 11 00 00 03 01 00
0D91:0A10  00 03 01 00 00 01 85 00-00 03 01 01 00 03 11 00
0D91:0A20  00 03 01 00 00 03 01 00-00 03 01 00 00 01 85 00
0D91:0A30  00 03 01 01 00 03 11 00-00 03 01 00 00 03 01 00
0D91:0A40  00 01 85 00 00 03 01 01-00 03 11 00 00 03 01 00
0D91:0A50  00 03 01 00 00 03 95 00-00 03 01 01 00 03 11 00
0D91:0A60  00 03 01 00 00 03 01 00-00 03 95 00 00 03 01 01
0D91:0A70  00 03 11 00 00 03 01 00-00 03 01 00 00 03 01 00
0D91:0A80  00 03 95 00 00 03 01 01-00 03 11 00 00 03 01 00
0D91:0A90  00 03 01 00 00 03 95 00-00 03 01 01 00 03 11 00
```

```
0D91:0AA0  00 03 01 00 00 03 01 00-00 03 95 00 00 03 01 01
0D91:0AB0  00 03 11 00 00 03 01 00-00 03 01 00 00 03 01 00
0D91:0AC0  00 03 95 00 00 03 01 01-00 03 11 00 00 03 01 00
0D91:0AD0  00 03 01 00 00 03 95 00-00 03 01 01 00 03 11 00
0D91:0AE0  00 03 01 00 00 03 01 00-00 03 A5 00 00 03 01 01
0D91:0AF0  00 03 11 00 00 03 01 00-00 03 01 00 00 03 A5 00
0D91:0B00  00 03 01 01 00 03 11 00-00 03 01 00 00 03 01 00
0D91:0B10  00 03 01 00 00 03 A5 00-00 03 01 01 00 03 11 00
0D91:0B20  00 03 01 00 00 03 01 00-00 03 A5 00 00 03 01 01
0D91:0B30  00 03 11 00 00 03 01 00-00 03 01 00 00 03 A5 00
0D91:0B40  00 03 01 01 00 03 11 00-00 03 01 00 00 03 01 00
0D91:0B50  00 03 01 00 00 03 A5 00-00 03 01 01 00 03 11 00
0D91:0B60  00 03 01 00 00 03 01 00-00 03 A5 00 00 03 01 01
0D91:0B70  00 03 11 00 00 03 01 00-00 03 01 00 00 01 B5 00
0D91:0B80  00 03 01 01 00 03 11 00-00 03 01 00 00 03 01 00
0D91:0B90  00 01 B5 00 00 03 01 01-00 03 11 00 00 03 01 00
0D91:0BA0  00 03 01 00 00 03 01 00-00 01 B5 00 00 03 01 01
0D91:0BB0  00 03 11 00 00 03 01 00-00 03 01 00 00 01 B5 00
0D91:0BC0  00 03 01 01 00 03 11 00-00 03 01 00 00 03 01 00
0D91:0BD0  00 03 01 00 00 01 B5 00-00 03 01 01 00 03 11 00
0D91:0BE0  00 03 01 00 00 03 01 00-00 01 B5 00 00 03 01 01
0D91:0BF0  00 03 11 00 00 03 01 00-00 03 01 00 00 01 B5 00
0D91:0C00  00 03 01 01 00 03 11 00-00 03 01 00 00 03 01 00
0D91:0C10  00 03 C5 00 00 03 01 01-00 03 11 00 00 03 01 00
0D91:0C20  00 03 01 00 00 03 C5 00-00 03 01 01 00 03 11 00
0D91:0C30  00 03 01 00 00 03 01 00-00 03 01 00 00 03 C5 00
0D91:0C40  00 03 01 01 00 03 11 00-00 03 01 00 00 03 01 00
0D91:0C50  00 03 C5 00 00 03 01 01-00 03 11 00 00 03 01 00
0D91:0C60  00 03 01 00 00 03 01 00-00 03 C5 00 00 03 01 01
0D91:0C70  00 03 11 00 00 03 01 00-00 03 01 00 00 03 C5 00
0D91:0C80  00 03 01 01 00 03 11 00-00 03 01 00 00 03 01 00
0D91:0C90  00 03 C5 00 00 03 01 01-00 03 11 00 00 03 01 00
0D91:0CA0  00 03 01 00 00 01 D5 00-00 03 01 01 00 03 11 00
0D91:0CB0  00 03 01 00 00 03 01 00-00 01 D5 00 00 03 01 01
0D91:0CC0  00 03 11 00 00 03 01 00-00 03 01 00 00 03 01 00
0D91:0CD0  00 01 D5 00 00 03 01 01-00 03 11 00 00 03 01 00
0D91:0CE0  00 03 01 00 00 01 D5 00-00 03 01 01 00 03 11 00
0D91:0CF0  00 03 01 00 00 03 01 00-00 03 01 00 00 01 D5 00
0D91:0D00  00 03 01 01 00 03 11 00-00 03 01 00 00 03 01 00
0D91:0D10  00 01 D5 00 00 03 01 01-00 03 11 00 00 03 01 00
0D91:0D20  00 03 01 00 00 01 D5 00-00 03 01 01 00 03 11 00
0D91:0D30  00 03 01 00 00 03 01 00-00 01 E5 00 00 03 01 01
0D91:0D40  00 03 11 00 00 03 01 00-00 03 01 00 00 01 E5 00
0D91:0D50  00 03 01 01 00 03 11 00-00 03 01 00 00 03 01 00
0D91:0D60  00 03 01 00 00 01 E5 00-00 03 01 01 00 03 11 00
0D91:0D70  00 03 01 00 00 03 01 00-00 01 E5 00 00 03 01 01
0D91:0D80  00 03 11 00 00 03 01 00-00 03 01 00 00 03 01 00
0D91:0D90  00 01 E5 00 00 03 01 01-00 03 11 00 00 03 01 00
0D91:0DA0  00 03 01 00 00 01 E5 00-00 03 01 01 00 03 11 00
0D91:0DB0  00 03 01 00 00 03 01 00-00 01 E5 00 00 03 01 01
0D91:0DC0  00 03 11 00 00 03 01 00-00 03 01 00 00 03 04 00
0D91:0DD0  00 02 00 00 00 03 14 00-00 02 00 00 00 03 0C 00
0D91:0DE0  00 02 00 02 00 00 00 02-00 02 00 02 00 00 00 02
0D91:0DF0  00 02 00 02 00 00 00 02-00 02 00 02 00 00 00 02
0D91:0E00  00 02 00 02 00 00 00 02-00 02 00 02 00 00 00 02
```

```
0D91:0E10  00 02 00 02 00 00 00 02-00 02 00 00 00 03 01 00
0D91:0E20  00 03 01 00 00 03 04 00-00 02 00 00 00 03 14 00
0D91:0E30  00 00 29 00 00 03 0C 00-00 02 00 02 00 02 00 00
0D91:0E40  00 03 04 00 00 02 00 00-00 03 14 00 00 02 50 00
0D91:0E50  00 03 41 00 00 03 04 00-00 02 00 00 00 03 14 00
0D91:0E60  00 02 50 00 00 03 16 00-00 02 00 00 00 03 06 00
0D91:0E70  00 02 FF 00 00 03 01 00-00 03 01 00 00 03 01 00
0D91:0E80  00 03 01 00 00 03 01 00-00 03 01 00 00 03 01 00
0D91:0E90  01 03 11 00 00 03 01 00-00 03 05 02 00 03 15 02
0D91:0EA0  00 03 01 00 00 03 04 00-00 02 00 00 00 03 14 00
0D91:0EB0  00 02 00 00 00 03 03 02-00 03 04 00 00 02 00 00
0D91:0EC0  00 03 14 00 00 02 50 00-00 03 01 00 00 03 04 00
0D91:0ED0  00 00 07 00 00 03 14 00-00 00 D0 00 00 03 0C 00
0D91:0EE0  00 00 C1 00 00 03 04 00-00 02 00 00 00 03 14 00
0D91:0EF0  00 02 50 00 00 03 01 00-00 03 04 00 00 02 00 00
0D91:0F00  00 03 14 00 00 00 23 00-00 03 0C 00 00 02 00 02
0D91:0F10  00 02 00 00 00 03 04 00-00 02 00 00 00 03 14 00
0D91:0F20  00 02 50 00 00 03 41 00-00 03 04 00 00 02 00 00
0D91:0F30  00 03 14 00 00 00 29 00-00 03 0C 00 00 02 00 02
0D91:0F40  00 02 00 00 00 03 04 00-00 02 00 00 00 03 14 00
0D91:0F50  00 02 50 00 00 03 41 00-00 03 01 00 00 01 0A 00
0D91:0F60  04 00 00 03 04 00 00 02-00 00 00 03 14 00 00 00
0D91:0F70  08 00 00 03 0C 00 00 02-F6 02 00 00 00 03 04 00
0D91:0F80  00 02 00 00 00 03 14 00-00 02 50 00 00 03 41 00
0D91:0F90  00 01 0A 00 10 00 00 01-0A 00 04 00 00 03 04 00
0D91:0FA0  00 02 00 00 00 03 14 00-00 00 01 00 00 03 0C 00
0D91:0FB0  00 02 00 02 00 00 00 02-00 00 00 03 04 00 00 02
0D91:0FC0  00 00 00 03 14 00 00 02-50 00 00 03 41 00 00 01
0D91:0FD0  0A 00 10 00 00 01 0A 00-04 00 00 03 04 00 00 02
0D91:0FE0  00 00 00 03 14 00 00 02-00 00 00 03 0C 00 00 02
0D91:0FF0  FC 00 00 03 04 00 00 02-00 00 00 03 14 00 00 02
0D91:1000  50 00 00 03 41 00 00 01-0A 00 10 00 00 03 01 00
0D91:1010  00 01 0A 00 04 00 00 03-04 00 00 02 00 00 00 03
0D91:1020  14 00 00 00 01 00 00 03-0C 00 00 02 00 02 00 02
0D91:1030  00 00 00 03 04 00 00 02-00 00 00 03 14 00 00 02
0D91:1040  50 00 00 01 C1 00 00 01-0A 00 10 00 00 01 0A 00
0D91:1050  04 00 00 03 04 00 00 02-00 00 00 03 14 00 00 00
0D91:1060  08 00 00 03 0C 00 00 02-00 02 00 02 00 02 00 02
0D91:1070  00 02 00 02 00 00 00 03-04 00 00 02 00 00 00 03
0D91:1080  14 00 00 02 50 00 00 01-C1 00 00 01 0A 00 10 00
0D91:1090  00 03 01 00 00 01 0A 00-04 00 00 03 04 00 00 02
0D91:10A0  00 00 00 03 14 00 00 00-08 00 00 03 0C 00 00 02
0D91:10B0  00 02 00 02 00 00 00 02-00 00 00 03 04 00 00 02
0D91:10C0  00 00 00 03 14 00 00 02-50 00 00 01 C1 00 00 01
0D91:10D0  0A 00 10 00 00 01 0A 00-04 00 00 03 04 00 00 02
0D91:10E0  00 00 00 03 14 00 00 00-3B 00 00 03 0C 00 00 02
0D91:10F0  D7 00 F1 02 D7 00 20 02-27 00 00 03 04 00 00 02
0D91:1100  00 00 00 03 14 00 00 02-50 00 00 01 C1 00 00 01
0D91:1110  0A 00 10 00 00 01 0A 00-04 00 00 03 0D 00 20 03
0D91:1120  04 00 00 02 00 00 00 03-14 00 00 02 50 00 00 03
0D91:1130  01 00 00 01 0A 00 10 00-00 03 01 00 00 03 01 00
0D91:1140  00 03 01 00 00 03 01 00-00 03 01 00 00 03 01 00
0D91:1150  00 03 01 00 00 03 01 00-00 03 01 00 00 03 01 00
```

Appendix G

Data Captured during emulating 3270CIB as IBM 3278

The following data is captured using 3270CIB with the data bit 10, and bit 11. The data is captured during the testing of emulating 3270CIB as IBM 3278.

```
0CBE:0780  01 01 01 01 01 01 01 01-01 01 01 01 01 01 01 01
0CBE:0790  01 01 01 01 01 01 01 01-01 01 01 01 01 01 01 01
0CBE:07A0  02 01 02 11 00 01 00 04-00 00 00 14 00 00 00 03
0CBE:07B0  00 03 00 03 00 03 00 03-00 03 00 03 00 03 00 03
0CBE:07C0  00 03 00 03 00 03 00 03-00 03 00 03 00 03 00 04
0CBE:07D0  00 00 00 14 00 30 00 03-00 03 00 03 00 00 03 00
0CBE:07E0  03 00 03 00 03 00 03 00-03 00 03 00 03 00 03 00
0CBE:07F0  03 00 03 00 03 00 01 00-09 A4 09 A4 01 00 01 00
0CBE:0800  09 A4 09 A4 0D 20 01 00-04 00 00 00 14 00 00 00
0CBE:0810  07 00 04 00 00 00 14 00-00 00 07 00 04 00 00 00
0CBE:0820  14 00 00 00 07 00 04 00-00 00 14 00 00 00 07 00
0CBE:0830  04 00 00 00 14 00 00 00-07 00 04 00 00 00 14 00
0CBE:0840  00 00 07 00 04 00 00 00-14 00 00 00 07 00 25 00
0CBE:0850  01 00 11 00 01 00 01 00-25 00 01 00 11 00 01 00
0CBE:0860  01 00 01 00 25 00 01 00-11 00 01 00 01 00 25 00
0CBE:0870  01 00 11 00 01 00 01 00-25 00 01 00 11 00 01 00
0CBE:0880  01 00 01 00 25 00 01 00-11 00 01 00 01 00 25 00
0CBE:0890  01 00 11 00 01 00 01 00-35 00 01 00 11 00 01 00
0CBE:08A0  01 00 35 00 01 00 11 00-01 00 01 00 01 00 35 00
0CBE:08B0  01 00 11 00 01 00 01 00-35 00 01 00 11 00 01 00
0CBE:08C0  01 00 35 00 01 00 11 00-01 00 01 00 01 00 35 00
0CBE:08D0  01 00 11 00 01 00 01 00-35 00 01 00 11 00 01 00
0CBE:08E0  01 00 45 00 01 00 11 00-01 00 01 00 45 00 01 00
0CBE:08F0  11 00 01 00 01 00 01 00-45 00 01 00 11 00 01 00
0CBE:0900  01 00 45 00 01 00 11 00-01 00 01 00 45 00 01 00
0CBE:0910  11 00 01 00 01 00 01 00-45 00 01 00 11 00 01 00
0CBE:0920  01 00 45 00 01 00 11 00-01 00 01 00 55 00 01 00
0CBE:0930  11 00 01 00 01 00 55 00-01 00 11 00 01 00 01 00
0CBE:0940  01 00 55 00 01 00 11 00-01 00 01 00 55 00 01 00
0CBE:0950  11 00 01 00 01 00 55 00-01 00 11 00 01 00 01 00
0CBE:0960  01 00 55 00 01 00 11 00-01 00 01 00 55 00 01 00
0CBE:0970  11 00 01 00 01 00 65 00-01 00 11 00 01 00 01 00
0CBE:0980  65 00 01 00 11 00 01 00-01 00 01 00 65 00 01 00
0CBE:0990  11 00 01 00 01 00 65 00-01 00 11 00 01 00 01 00
0CBE:09A0  65 00 01 00 11 00 01 00-01 00 01 00 65 00 01 00
0CBE:09B0  11 00 01 00 01 00 65 00-01 00 11 00 01 00 01 00
0CBE:09C0  75 00 01 00 11 00 01 00-01 00 75 00 01 00 11 00
0CBE:09D0  01 00 01 00 01 00 75 00-01 00 11 00 01 00 01 00
```

```
0CBE:09E0  75 00 01 00 11 00 01 00-01 00 75 00 01 00 11 00
0CBE:09F0  01 00 01 00 01 00 75 00-01 00 11 00 01 00 01 00
0CBE:0A00  75 00 01 00 11 00 01 00-01 00 85 00 01 00 11 00
0CBE:0A10  01 00 01 00 85 00 01 00-11 00 01 00 01 00 01 00
0CBE:0A20  85 00 01 00 11 00 01 00-01 00 85 00 01 00 11 00
0CBE:0A30  01 00 01 00 85 00 01 00-11 00 01 00 01 00 01 00
0CBE:0A40  85 00 01 00 11 00 01 00-01 00 85 00 01 00 11 00
0CBE:0A50  01 00 01 00 95 00 01 00-11 00 01 00 01 00 95 00
0CBE:0A60  01 00 11 00 01 00 01 00-01 00 95 00 01 00 11 00
0CBE:0A70  01 00 01 00 95 00 01 00-11 00 01 00 01 00 95 00
0CBE:0A80  01 00 11 00 01 00 01 00-01 00 95 00 01 00 11 00
0CBE:0A90  01 00 01 00 95 00 01 00-11 00 01 00 01 00 A5 00
0CBE:0AA0  01 00 11 00 01 00 01 00-A5 00 01 00 11 00 01 00
0CBE:0AB0  01 00 01 00 A5 00 01 00-11 00 01 00 01 00 A5 00
0CBE:0AC0  01 00 11 00 01 00 01 00-A5 00 01 00 11 00 01 00
0CBE:0AD0  01 00 01 00 A5 00 01 00-11 00 01 00 01 00 A5 00
0CBE:0AE0  01 00 11 00 01 00 01 00-B5 00 01 00 11 00 01 00
0CBE:0AF0  01 00 B5 00 01 00 11 00-01 00 01 00 B5 00 01 00
0CBE:0B00  11 00 01 00 01 00 01 00-B5 00 01 00 11 00 01 00
0CBE:0B10  01 00 B5 00 01 00 11 00-01 00 01 00 01 00 B5 00
0CBE:0B20  01 00 11 00 01 00 01 00-B5 00 01 00 11 00 01 00
0CBE:0B30  01 00 C5 00 01 00 11 00-01 00 01 00 C5 00 01 00
0CBE:0B40  11 00 01 00 01 00 C5 00-01 00 11 00 01 00 01 00
0CBE:0B50  01 00 C5 00 01 00 11 00-01 00 01 00 C5 00 01 00
0CBE:0B60  11 00 01 00 01 00 01 00-C5 00 01 00 11 00 01 00
0CBE:0B70  01 00 C5 00 01 00 11 00-01 00 01 00 D5 00 01 00
0CBE:0B80  11 00 01 00 01 00 D5 00-01 00 11 00 01 00 01 00
0CBE:0B90  D5 00 01 00 11 00 01 00-01 00 01 00 D5 00 01 00
0CBE:0BA0  11 00 01 00 01 00 D5 00-01 00 11 00 01 00 01 00
0CBE:0BB0  01 00 D5 00 01 00 11 00-01 00 01 00 D5 00 01 00
0CBE:0BC0  11 00 01 00 01 00 E5 00-01 00 11 00 01 00 01 00
0CBE:0BD0  E5 00 01 00 11 00 01 00-01 00 E5 00 01 00 11 00
0CBE:0BE0  00 01 00 01 00 E5 00 01-00 11 00 01 00 01 00 E5
0CBE:0BF0  00 01 00 11 00 01 00 01-00 01 00 E5 00 01 00 11
0CBE:0C00  00 01 00 01 00 E5 00 01-00 11 00 01 00 01 00 04
0CBE:0C10  00 00 00 14 00 00 00 0C-00 00 00 00 00 00 00 00
0CBE:0C20  00 00 00 00 00 00 00 00-04 00 00 00 14 00 00 00
0CBE:0C30  0C 00 00 00 00 00 00 00-00 04 00 00 00 14 00 00
0CBE:0C40  00 0C 00 00 00 00 00 00-00 00 04 00 00 00 14 00
0CBE:0C50  00 00 0C 00 00 00 00 00-00 00 00 00 00 00 00 00
0CBE:0C60  00 00 00 00 00 00 00 00-00 00 00 00 00 04 00 00
0CBE:0C70  00 14 00 00 00 0C 00 00-00 00 00 00 00 00 00 04
0CBE:0C80  00 00 00 14 00 00 00 0C-00 00 00 00 00 00 00 00
0CBE:0C90  04 00 00 00 14 00 00 00-0C 00 00 00 00 00 00 00
0CBE:0CA0  00 01 00 01 00 01 00 01-00 01 00 01 00 01 00 01
0CBE:0CB0  00 01 00 01 00 01 00 01-00 01 00 01 00 01 00 01
0CBE:0CC0  00 01 00 01 00 01 00 01-00 01 00 01 00 01 00 01
0CBE:0CD0  00 01 00 01 00 01 00 01-00 01 00 01 00 01 00 01
0CBE:0CE0  00 01 00 01 00 01 00 01-00 01 00 01 00 01 00 01
0CBE:0CF0  00 01 00 01 00 01 00 01-00 01 00 01 00 01 00 01
```

Appendix H

Attribute Assignments

Attribute character bit assignments are summarized as follows:

X	X	U/P	A/N	D/SPD		Reserved	MDT
0	1	2	3	4	5	6	7

EBCDIC Bit	Field Description
0, 1	- Value determined by contents of bits 2–7. See Figure 2-6 for hexadecimal values.
2	- 0 = Unprotected 1 = Protected
3	- 0 = Alphameric 1 = Numeric (causes automatic upshift of data entry keyboard) Note: *Bits 2 and 3 equal to 11 causes an automatic skip. See text.*
4, 5	- 00 = Display/not selector-light-pen detectable. 01 = Display/selector-light-pen detectable. 10 = Intensified display/selector-light-pen-detectable. 11 = Nondisplay, nonprint, nondetectable.
6	- Reserved. Must always be 0.
7	- Modified Data Tag (MDT); identifies modified fields during Read Modified command operations. 0 = Field has not been modified. 1 = Field has been modified by the operator. Can also be set by program in data stream.

Bits 2–7	Graphic	EBCDIC	ASCII
00 0000	SP	40	20
00 0001	A	C1	41
00 0010	B	C2	42
00 0011	C	C3	43
00 0100	D	C4	44
00 0101	E	C5	45
00 0110	F	C6	46
00 0111	G	C7	47
00 1000	H	C8	48
00 1001	I	C9	49
00 1010	¢	4A	—
	[	—	5B
00 1011	.	4B	2E
00 1100	<	4C	3C
00 1101	(	4D	28
00 1110	+	4E	2B
00 1111	!	4F	—
	!	—	21
01 0000	&	50	26
01 0001	J	D1	4A
01 0010	K	D2	4B
01 0011	L	D3	4C
01 0100	M	D4	4D
01 0101	N	D5	4E
01 0110	O	D6	4F
01 0111	P	D7	50
01 1000	Q	D8	51
01 1001	R	D9	52
01 1010	]	5A	—
	]	—	5D
01 1011	$	5B	24
01 1100	*	5C	2A
01 1101	)	5D	29
01 1110	;	5E	3B
01 1111	¬	5F	
	^	—	5E

Bits 2–7	Graphic	EBCDIC	ASCII
10 0000	-	60	2D
10 0001	/	61	2F
10 0010	S	E2	53
10 0011	T	E3	54
10 0100	U	E4	55
10 0101	V	E5	56
10 0110	W	E6	57
10 0111	X	E7	58
10 1000	Y	E8	59
10 1001	Z	E9	5A
10 1010	¦ (EBCDIC)	6A	7C
10 1011	,	6B	2C
10 1100	%	6C	25
10 1101	—	6D	5F
10 1110	>	6E	3E
10 1111	?	6F	3F
11 0000	0	F0	30
11 0001	1	F1	31
11 0010	2	F2	32
11 0011	3	F3	33
11 0100	4	F4	34
11 0101	5	F5	35
11 0110	6	F6	36
11 0111	7	F7	37
11 1000	8	F8	38
11 1001	9	F9	39
11 1010	:	7A	3A
11 1011	#	7B	23
11 1100	@	7C	40
11 1101	'	7D	27
11 1110	=	7E	3D
11 1111	"	7F	22

Note: *The characters above are used as attribute, AID, write control (WCC), copy control (CCC), CU and device address, and buffer address. They are also used as status and sense, except by the 3274 and 3276 when operating in BSC. When any of these characters is transmitted to the program, the CU assigns the appropriate EBCDIC code. If transmission is in ASCII, the CU translates the EBCDIC code to ASCII code prior to transmission.*

To use this table to determine the hex code transmitted for an address or control character, first determine the values of bits 2–7. Select this bit configuration from the "Bits 2–7" column. The hex code that will be transmitted (either in EBCDIC or in ASCII) is to the right of the bit configuration.

Use this table also to determine equivalent EBCDIC and ASCII hex codes and their associated graphic characters. See Figure 2-4, Note 5, for ASCII A and B graphic character difference for ASCII codes 21 and 5E (hex).

Graphic characters for the United States I/O interface codes are shown. Graphic characters might differ for particular World Trade I/O interface codes. Refer to IBM 3270 Information Display System: Character Set Reference, *GA27-2837, for possible graphic differences when these codes are used.*

Appendix I

EBCDIC Code

Hex 1 / Bits 4567		00				01				10				11				← Bits 0, 1
		00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11	← 2, 3
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	← Hex 0
0000	0	NUL				SP	&	-						{	}	\	0	
0001	1		SBA					/		a	j	~		A	J		1	
0010	2		EUA							b	k	s		B	K	S	2	
0011	3		IC							c	l	t		C	L	T	3	
0100	4									d	m	u		D	M	U	4	
0101	5	PT	NL							e	n	v		E	N	V	5	
0110	6									f	o	w		F	O	W	6	
0111	7									g	p	x		G	P	X	7	
1000	8									h	q	y		H	Q	Y	8	
1001	9		EM						·	i	r	z		I	R	Z	9	
1010	A					¢	!	¦	:									
1011	B					.	$	,	#									
1100	C	FF	DUP		RA	<	*	%	@									
1101	D	CR	SF			(	)	_	'									
1110	E		FM			+	;	>	=									
1111	F				SUB	|	¬	?	"									

Legend:

CR	Carriage Return Code
DUP	Duplicate Character Code
EM	End-of-Message Printer Control Character
EUA	Erase Unprotected to Address Order Code
FF	Form Feed Printer Control Character
FM	Field Mark Character
IC	Insert Cursor Order Code
NL	New Line Code
NUL	Null Character Code
PT	Program Tab Order Code
RA	Repeat to Address Order Code
SBA	Set Buffer Address Order Code
SF	Start Field Order Code
SP	Space Character
SUB	Substitute Character (Error Override)

Appendix J

Thai EBCDIC Code B

CODE TRANSLATION TABLE (Contd) (B3)

Dec.	Hex	Instruction and Format	Graphics and Controls BCDIC	EBCDIC(1)	ASCII	Thai Character	Card Code EBCDIC	Binary
128	80	SSM -S				ก	12-0-1-8	1000 0000
129	81			a	a	ข	12-0-1	1000 0001
130	82	LPSW -S		b	b	ข	12-0-2	1000 0010
131	83	Diagnose		c	c	ค	12-0-3	1000 0011
132	84	WRD SI		d	d	ค	12-0-4	1000 0100
133	85	RDD SI		e	e	ฆ	12-0-5	1000 0101
134	86	BXH SI		f	f	ง	12-0-6	1000 0110
135	87	BXLE SI		g	g	จ	12-0-7	1000 0111
136	88	SRL		h	h	ฉ	12-0-8	1000 1000
137	89	SLL		i	i	ช	12-0-9	1000 1001
138	8A	SRA				ช	12-0-2-8	1000 1010
139	8B	SLA RS			{	ซ	12-0-3-8	1000 1011
140	8C	SRDL			≤	ฌ	12-0-4-8	1000 1100
141	8D	SLDL			(	ญ	12-0-5-8	1000 1101
142	8E	SRDA			+	ฎ	12-0-6-8	1000 1110
143	8F	SLDA			!	ฏ	12-0-7-8	1000 1111
144	90	STM			&	ฐ	12-11-1-8	1001 0000
145	91	TM SI		j	j	ฑ	12-11-1	1001 0001
146	92	MVI SI		k	k	ฒ	12-11-2	1001 0010
147	93	TS -S		l	l	ณ	12-11-3	1001 0011
148	94	NI		m	m	ด	12-11-4	1001 0100
149	95	CLI SI		n	n	ต	12-11-5	1001 0101
150	96	OI SI		o	o	ถ	12-11-6	1001 0110
151	97	XI SI		p	p	ท	12-11-7	1001 0111
152	98	LM -RS		q	q	ธ	12-11-8	1001 1000
153	99			r	r	น	12-11-9	1001 1001
154	9A					บ	12-11-2-8	1001 1010
155	9B				}	ป	12-11-3-8	1001 1011
156	9C	SIO, SIOF S			¤	ผ	12-11-4-8	1001 1100
157	9D	TIO, CLRIO S			!	ฝ	12-11-5-8	1001 1101
158	9E	HIO, HDV S			±	พ	12-11-6-8	1001 1110
159	9F	TCH S			¤	ฟ	12-11-7-8	1001 1111
160	A0				~	ภ	11-0-1-8	1010 0000
161	A1				•	ม	11-0-1	1010 0001
162	A2			s	s	ย	11-0-2	1010 0010
163	A3			t	t	ร	11-0-3	1010 0011
164	A4			u	u	ฤ	11-0-4	1010 0100
165	A5			v	v	ล	11-0-5	1010 0101
166	A6			w	w	ว	11-0-6	1010 0110
167	A7			x	x	ศ	11-0-7	1010 0111
168	A8			y	y	ษ	11-0-8	1010 1000
169	A9			z	z	ส	11-0-9	1010 1001
170	AA					ห	11-0-2-8	1010 1010
171	AB				¬	ฬ	11-0-3-8	1010 1011
172	AC	STNSM SI			¬	อ	11-0-4-8	1010 1100
173	AD	STOSM SI			[	ฮ	11-0-5-8	1010 1101
174	AE	SIGP -RS			≥	ฯ	11-0-6-8	1010 1110
175	AF	MC -SI			¤	ะ	11-0-7-8	1010 1111
176	B0				0	ั	12-11-0-1-8	1011 0000
177	B1	LRA -RX			1	า	12-11-0-1	1011 0001
178	B2	See below			2	ำ	12-11-0-2	1011 0010
179	B3				3	ิ	12-11-0-3	1011 0011
180	B4				4	ี	12-11-0-4	1011 0100
181	B5				5	ึ	12-11-0-5	1011 0101
182	B6	STCTL RS			6	ื	12-11-0-6	1011 0110
183	B7	LCTL RS			7	ุ	12-11-0-7	1011 0111
184	B8				8	ู	12-11-0-8	1011 1000
185	B9				9	ฺ	12-11-0-9	1011 1001
186	BA	CS RS				เ	12-11-0-2-8	1011 1010
187	BB	CDS RS			!	แ	12-11-0-3-8	1011 1011
188	BC				^	โ	12-11-0-4-8	1011 1100
189	BD	CLM RS			]	ใ	12-11-0-5-8	1011 1101
190	BE	STCM RS			+	ไ	12-11-0-6-8	1011 1110
191	BF	ICM RS			-	ๆ	12-11-0-7-8	1011 1111

CODE TRANSLATION TABLE (Contd)

(B4)

Dec.	Hex	Instruction (SS)	Graphics and Controls BCDIC	EBCDIC(1)	ASCII	Thai Character	Card Code EBCDIC	Binary
192	C0		?	{			12-0	1100 0000
193	C1		A	A	A		12-1	1100 0001
194	C2		B	B	B		12-2	1100 0010
195	C3		C	C	C		12-3	1100 0011
196	C4		D	D	D		12-4	1100 0100
197	C5		E	E	E		12-5	1100 0101
198	C6		F	F	F		12-6	1100 0110
199	C7		G	G	G		12-7	1100 0111
200	C8		H	H	H		12-8	1100 1000
201	C9		I	I	I		12-9	1100 1001
202	CA					๊	12-0-2-8-9	1100 1010
203	CB					๋	12-0-3-8-9	1100 1011
204	CC			‚		์	12-0-4-8-9	1100 1100
205	CD					๎	12-0-5-8-9	1100 1101
206	CE			¥		๏	12-0-6-8-9	1100 1110
207	CF					๐	12-0-7-8-9	1100 1111
208	D0		!	}			11-0	1101 0000
209	D1	MVN	J	J	J		11-1	1101 0001
210	D2	MVC	K	K	K		11-2	1101 0010
211	D3	MVZ	L	L	L		11-3	1101 0011
212	D4	NC	M	M	M		11-4	1101 0100
213	D5	CLC	N	N	N		11-5	1101 0101
214	D6	OC	O	O	O		11-6	1101 0110
215	D7	XC	P	P	P		11-7	1101 0111
216	D8		Q	Q	Q		11-8	1101 1000
217	D9		R	R	R		11-9	1101 1001
218	DA					๚	12-11-2-8-9	1101 1010
219	DB					๛	12-11-3-8-9	1101 1011
220	DC	TR					12-11-4-8-9	1101 1100
221	DD	TRT					12-11-5-8-9	1101 1101
222	DE	ED					12-11-6-8-9	1101 1110
223	DF	EDMK					12-11-7-8-9	1101 1111
224	E0		ŧ	\			0-2-8	1110 0000
225	E1						11-0-1-9	1110 0001
226	E2		S	S	S		0-2	1110 0010
227	E3		T	T	T		0-3	1110 0011
228	E4		U	U	U		0-4	1110 0100
229	E5		V	V	V		0-5	1110 0101
230	E6		W	W	W		0-6	1110 0110
231	E7		X	X	X		0-7	1110 0111
232	E8		Y	Y	Y		0-8	1110 1000
233	E9		Z	Z	Z		0-9	1110 1001
234	EA						11-0-2-8-9	1110 1010
235	EB						11-0-3-8-9	1110 1011
236	EC			‹			11-0-4-8-9	1110 1100
237	ED						11-0-5-8-9	1110 1101
238	EE						11-0-6-8-9	1110 1110
239	EF						11-0-7-8-9	1110 1111
240	F0	SRP	0	0	0		0	1111 0000
241	F1	MVO	1	1	1		1	1111 0001
242	F2	PACK	2	2	2		2	1111 0010
243	F3	UNPK	3	3	3		3	1111 0011
244	F4		4	4	4		4	1111 0100
245	F5		5	5	5		5	1111 0101
246	F6		6	6	6		6	1111 0110
247	F7		7	7	7		7	1111 0111
248	F8	ZAP	8	8	8		8	1111 1000
249	F9	CP	9	9	9		9	1111 1001
250	FA	AP		!			12-11-0-2-8-9	1111 1010
251	FB	SP					12-11-0-3-8-9	1111 1011
252	FC	MP					12-11-0-4-8-9	1111 1100
253	FD	DP					12-11-0-5-8-9	1111 1101
254	FE						12-11-0-6-8-9	1111 1110
255	FF			EO			12-11-0-7-8-9	1111 1111

Appendix K

Data Sheets

82C570 CHIPSLink™
SINGLE CHIP "3270" PROTOCOL CONTROLLER

- Implements IBM 3270 Communication Protocol
- Provides IBM 3278/79 and IRMA™ emulation adapter cards interface
- Supports both CUT and DFT modes
- Type A coaxial transmitter and receiver
- 8X Digital Phase Locked Loop (DPLL)
- On chip 4.7 MIPS microcontroller
- Dual port RAM control function
- 2.4K bytes internal microcode
- Provisions for external microcode
- Low power CMOS technology
- 18.8696 MHz crystal oscillator
- 84 pins PLCC package

The 82C570 is a highly integrated IBM 3270 coaxial type A protocol controller chip. It serves as an I/O processor to emulate most of the IBM terminals and printers. It works with IBM 3276/3274/3174 control units either locally or remotely attached.

The 82C570 internal microcode supports most display terminals and printers in both CUT and DFT modes except 3279-S3G due to the large memory requirement. The programmed symbol and APA graphics for 3179/G and 3270 PC/G can only be supported in DFT mode. There are provisions for adding an additional 8K × 8 SRAM and using external microcode.

The 82C570 provides both IRMA and IBM emulation adapter compatible interface. All the IRMA and IBM emulation, file transfer and application software can run on the adapter using 82C570.

The 82C570 along with an external 120ns 8K × 8 SRAM , line driver, line receiver and pulse transformer, a complete 3270 protocol emulation adapter can be built easily.

The 82C570 is fabricated using advanced CMOS technology and is packaged in an 84 pin PLCC.

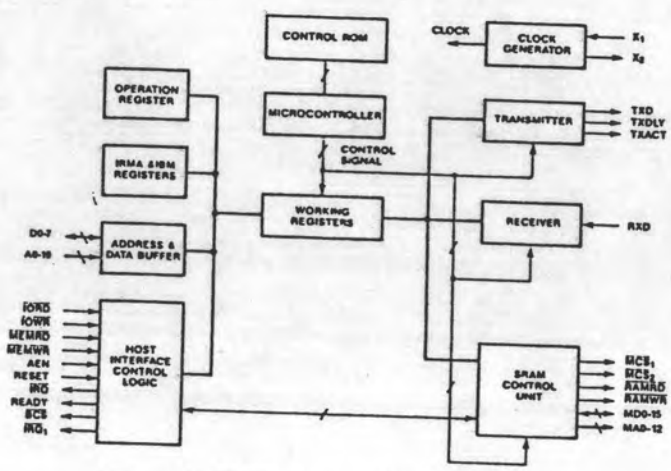
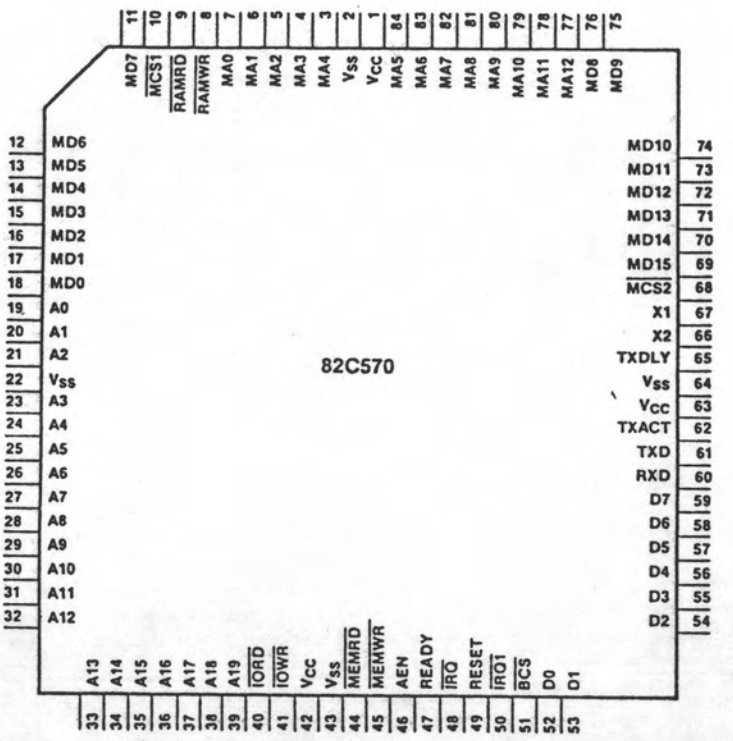


Figure 1. 82C570 Functional Block Diagram


82C570
1 VCC
2 VSS
3 MA4
4 MA3
5 MA2
6 MA1
7 MA0
8 RAMWR
9 RAMRD
10 MCS1
11 MD7
12 MD6
13 MD5
14 MD4
15 MD3
16 MD2
17 MD1
18 MD0
19 A0
20 A1
21 A2
22 VSS
23 A3
24 A4
25 A5
26 A6
27 A7
28 A8
29 A9
30 A10
31 A11
32 A12
33 A13
34 A14
35 A15
36 A16
37 A17
38 A18
39 A19
40 IORD
41 IOWR
42 VCC
43 VSS
44 MEMRD
45 MEMWR
46 AEN
47 READY
48 IRQ
49 RESET
50 IRQ1
51 BCS
52 D0
53 D1
54 D2
55 D3
56 D4
57 D5
58 D6
59 D7
60 RXD
61 TXD
62 TXACT
63 VCC
64 VSS
65 TXDLY
66 X2
67 X1
68 MCS2
69 MD15
70 MD14
71 MD13
72 MD12
73 MD11
74 MD10
75 MD9
76 MD8
77 MA12
78 MA11
79 MA10
80 MA9
81 MA8
82 MA7
83 MA6
84 MA5

82C570 Pin Description

Pin No.	Pin Type	Symbol	Pin Description
PC Bus Interface			
19-21 23-39	I I	A0-2 A3-19	System Address bit 0 to 19. These bits are used to address the memory and I/O devices. A0 is the least significant bit (LSB) and A19 is the most significant bit (MSB).
40	I	$\overline{\text{IORD}}$	Active low I/O read strobe. It is used by the system CPU to read the 82C570 internal registers.
41	I	$\overline{\text{IOWR}}$	Active low I/O write strobe. It is used by the system CPU to load the information into the internal registers.
44	I	$\overline{\text{MEMRD}}$	Active low memory read strobe. When this signal is active, the display buffer is read.
45	I	$\overline{\text{MEMWR}}$	Active low memory write strobe. When active, the display buffer is written or the external microcode is being down-loaded.
46	I	AEN	Active high address enable for DMA transfers. When this line is active, DMA controller has the control of the bus. When it is low, $\overline{\text{IORD}}$ $\overline{\text{IOWR}}$ $\overline{\text{MEMRD}}$ $\overline{\text{MEMWR}}$ are enabled for 82C570.
47	T	READY	READY is a active high output signal to indicate to the host system that a data transfer will be completed. During I/O transfer, this signal will always be tri-stated. For memory read or write operation, READY will be deasserted for a period of 220 ns - 460 ns to inform the host to insert the wait state. READY has the tri-state buffer and requires external pull up resistor. It can drive the system bus directly.
48	T	$\overline{\text{IRQ}}$	Interrupt request signal. When $\overline{\text{IRQ}}$ is active, it will go low for 100 ns - 250 ns then return to tri-state. It is designed for the edge triggered interrupt system.
49	I	RESET	Active high hardware reset signal. When reset is active, it initializes the chip and the program counter of the micro-controller is reset to address 0.
50	T	$\overline{\text{IRQ1}}$	Level interrupt request signal. When a interrupt is initiated, $\overline{\text{IRQ1}}$ will be held active low untill it is acknowledged by the system interrupt service routine. It stays tri-stated when it is inactive.
51	O	$\overline{\text{BCS}}$	Active low external 74LS245 buffer enable signal. It goes active when the internal registers or the display buffer RAM are accessed. (Either read or write.)
52-59	B	D0-7	System data bit 0 to 7. These bits are used to transfer data to and from the CPU data bus. They are 3 state bidirectional lines.

82C570 Pin Description (Continued)

Pin No.	Pin Type	Symbol	Pin Description
Serial interface			
60	I	RXD	Receive input data. It is the serial biphase Manchester II encoded bit stream from the controller unit. RXD is connected to the TTL level output of the external differential receive amplifier.
61	O	TXD	Transmit data output. It is the biphase Manchester II encoded data output from the transmitter. An external line driver is required for cable interface.
62	O	TXACT	Active high transmit active signal. It goes high while the transmitter is transmitting.
65	O	TXDLY	Delayed transmit output data. It has a delay of 1/4 bit time from TXD. TXDLY is designed for an easy implementation of the cable waveform precompensation.
Buffer RAM interface			
7-3 84-77	O O	MA0-4 MA5-12	Memory address bit 0 to 12. These bits are used to address the RAM buffer. MA0 is the least significant bit and MA12 is the most significant bit.
18-11 76-69	B B	MD0-7 MD8-15	Memory data bit 0 to bit 15. They are 3 state lines used to transfer the data between RAM, 82C570, control units and system CPU. MD0 is the least significant bit and MD15 is the most significant bit.
8	O	$\overline{\text{RAMWR}}$	Active low memory write strobe signal. It is active when system CPU, control unit or 82C570 write the buffer RAM.
9	O	$\overline{\text{RAMRD}}$	Active low memory read strobe signal.
10	O	$\overline{\text{MCS1}}$	Active low memory selection 1. It is used to enable the first RAM.
68	O	$\overline{\text{MCS2}}$	Active low memory selection 2 used for the enabling of the second RAM.
Miscellaneous			
67,66	I/O	X1, X2	Crystal oscillator input. A fundamental frequency parallel resonant crystal should be connected to this pair. Alternatively, an external clock source may be connected to X1 input by floating X2. The clock frequency should be 18.8696 MHz with an accuracy of ± 0.01%.
1,42,63	I	VCC	5 Volt power supply.
2,22,43,64	I	VSS	Power supply ground.

Note: I = Input, O = Output, T = 3-state output, B = Bidirectional

MICROCONTROLLER 8X305

FEATURES

- Fetch, Decode, and Execute a 16-bit instruction in a minimum of 200 nanoseconds (one machine cycle)
- Bit-oriented instruction set (addressable single-or-multiple bit subfields)
- Separate buses for Instruction, Instruction Address and Three-State I/O
- Thirteen 8-bit general-purpose working registers
- Source/destination architecture
- Bipolar low-power Schottky technology/TTL inputs and outputs
- On-chip oscillator and timing generation
- Single +5V supply
- 0.9-in. 50-pin DIP

PRODUCT DESCRIPTION

The Signetics 8X305 MicroController (Figure 1) is a high-speed bipolar microprocessor implemented with low-power Schottky technology. In a single chip, the 8X305 combines speed, flexibility, and a bit-oriented instruction set. These features and other basic characteristics of the chip combine to provide cost-effective solutions for a broad range of applications. The 8X305 is particularly useful in systems that require high-speed bit manipulations — sophisticated controllers, data communications, very fast interface control, and other applications of a similar nature.

The 8X305 can fetch, decode, and execute a 16-bit instruction word in a minimum of 200 nanoseconds. Within one instruction cycle, the 8-bit data-processing path can be programmed to rotate, mask, shift, and/or merge single or multiple bit subfields and, in addition, perform an ALU operation; in the same instruction, an external data field can be input, processed, and output to a specified destination — likewise, single or multiple bit data fields can be internally moved from a given source to a given destination. To summarize, fixed or variable-length data fields can be fetched, processed, operated on by the ALU, and moved to a different location — all in a timeframe of 200 nanoseconds. To interface with I/O and program memory, the 8X305 uses a 13-bit instruction address bus, a 16-bit instruction bus, an 8-bit bidirectional multiplexed I/O data/address bus and a 5-bit I/O control bus.

A wide selection of I/O devices, interface chips, and special-purpose parts are available for systems use. In most applications, the more powerful 8X305 is functionally interchangeable with its predecessor — the 8X300.

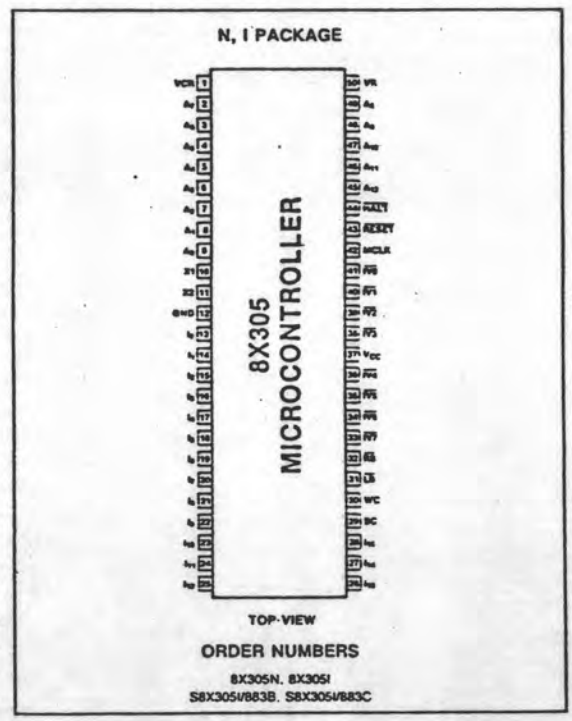
ASSOCIATED DOCUMENTATION

Other documents directly relating to *design* and *applications use* of the 8X305 MicroController are:

- Product Capabilities Manual
- 8X305 Users Manual

These documents and other current literature (Data Sheets, Product Bulletins, Applications Notes, etc.) are available at all Signetics Sales and Service Offices — see rear cover of this data sheet for the office in your locality.

PIN CONFIGURATION



ORDER NUMBERS

8X305N, 8X305I
S8X305I/883B, S8X305I/883C

FUNCTIONAL OPERATION

Typical System Configuration

Although the system hookup shown in Figure 3 is of the simplest form, it provides a fundamental look at the 8X305 MicroController and peripheral relationships. As indicated, the 8X305 can directly address up to 8K words of program storage — either ROM or PROM. The user interface ($\overline{IV0}$ through $\overline{IV7}$) is capable of uniquely addressing 256 Input/Output locations and, with additional bank bits ($\overline{LB}$, $\overline{RB}$), this number is expanded to 512 — each bank comprising 256 addressable locations. The addressable locations of each bank can be used in a variety of ways; a simple method of implementation is shown in Figure 3. When $\overline{LB}$ is active low, the left bank is enabled and any one of 256 locations within the RAM memory can be accessed for input/output operations. A similar set of "enable/access" conditions are applicable to the right bank when $\overline{RB}$ is active low.

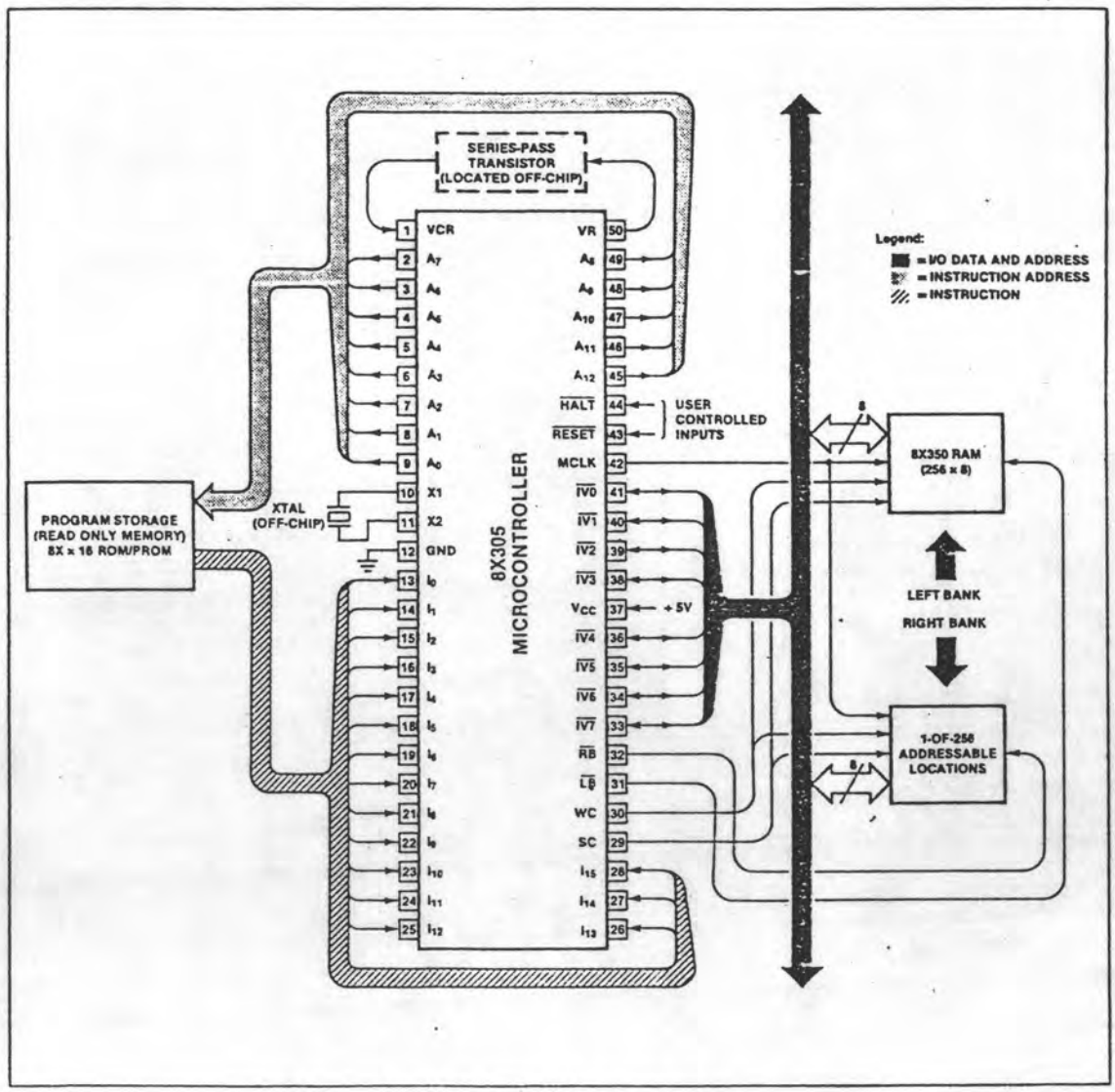
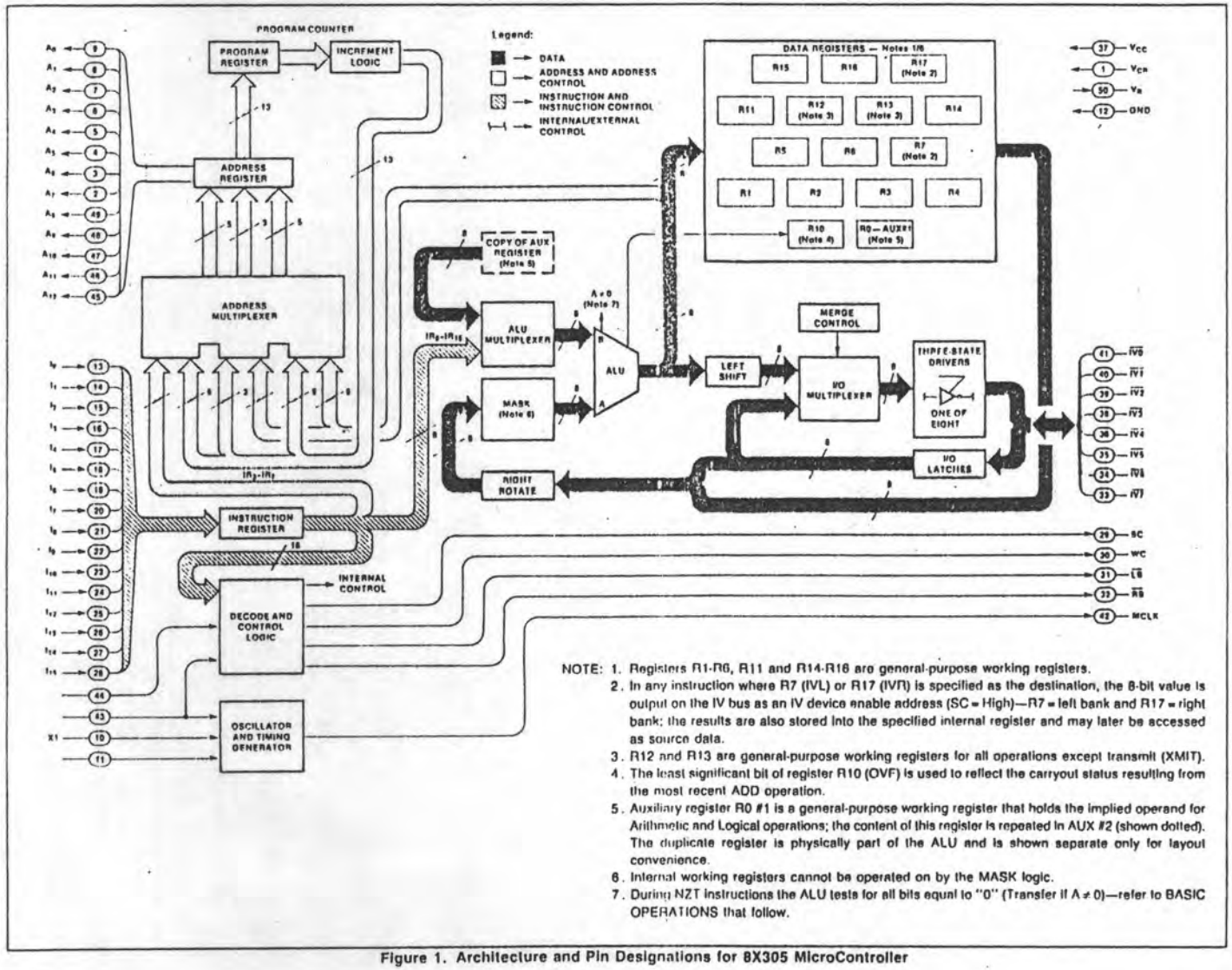


Figure 3. Typical 8X305 System Hookup



NOTE: 1. Registers R1-R6, R11 and R14-R16 are general-purpose working registers.
2. In any instruction where R7 (IVL) or R17 (IVH) is specified as the destination, the 8-bit value is output on the IV bus as an IV device enable address (SC = High)—R7 = left bank and R17 = right bank; the results are also stored into the specified internal register and may later be accessed as source data.
3. R12 and R13 are general-purpose working registers for all operations except transmit (XMIT).
4. The least significant bit of register R10 (OVF) is used to reflect the carryout status resulting from the most recent ADD operation.
5. Auxiliary register R0 #1 is a general-purpose working register that holds the implied operand for Arithmetic and Logical operations; the content of this register is repeated in AUX #2 (shown dotted). The duplicate register is physically part of the ALU and is shown separate only for layout convenience.
6. Internal working registers cannot be operated on by the MASK logic.
7. During NZT instructions the ALU tests for all bits equal to "0" (Transfer if A ≠ 0)—refer to BASIC OPERATIONS that follow.

Figure 1. Architecture and Pin Designations for 8X305 MicroController

BASIC OPERATIONS OF 8X305

Refer to a later discussion of "Instruction Fields" for a detailed examination of all operand fields and subdivisions thereof—"S" (S_0, S_1), "D" (D_0, D_1), "R", "L", "J", and "A".

MOVE OPERATIONS

REGISTER-TO-REGISTER

SOURCE	DATA PROCESSING (PRE-ALU)	ALU	DATA PROCESSING (POST-ALU)	DESTINATION
R0–R17 as specified by "S" field of instruction.	Right rotate as specified by "R" field of instruction.	No operation.	No operation.	R0–R7, R11–R17 as specified by "D" field of instruction.

REGISTER-TO-$\overline{IV}$ BUS

SOURCE	DATA PROCESSING (PRE-ALU)	ALU	DATA PROCESSING (POST-ALU)	DESTINATION
R0–R17 as specified by "S" field of instruction.	No operation.	No operation.	Shift and merge as specified by "D_0" and "L" fields of instruction.	Variable length field of $\overline{IV}$ bus—Left Bank ($\overline{LB}$) or Right Bank ($\overline{RB}$) as specified by "D_0" and "D_1" fields of instruction.

$\overline{IV}$ BUS-TO-REGISTER

SOURCE	DATA PROCESSING (PRE-ALU)	ALU	DATA PROCESSING (POST-ALU)	DESTINATION
Variable length field of $\overline{IV}$ bus—Left Bank ($\overline{LB}$) or Right Bank ($\overline{RB}$) as specified by "S_1" and "S_0" fields of instruction.	Right rotate and mask as specified by "S_0" and "L" fields of instruction.	No operation.	No operation.	R0–R7, R11–R17 as specified by "D" field of instruction.

$\overline{IV}$ BUS-TO-$\overline{IV}$ BUS

SOURCE	DATA PROCESSING (PRE-ALU)	ALU	DATA PROCESSING (POST-ALU)	DESTINATION
Variable length field of $\overline{IV}$ bus—Left Bank ($\overline{LB}$) or Right Bank ($\overline{RB}$) as specified by "S_1" and "S_0" fields of instruction.	Right rotate and mask as specified by "S_0" and "L" fields of instruction.	No operation.	Shift and merge as specified by "D_0" and "L" fields of instruction.	Variable length field of $\overline{IV}$ bus—Left Bank ($\overline{LB}$) or Right Bank ($\overline{RB}$) as specified by "D_0" and "D_1" fields of instruction.

ADD OPERATIONS

REGISTER-to-REGISTER → SOURCE → PRE-ALU
REGISTER-TO-$\overline{IV}$ BUS → SOURCE → PRE-ALU
$\overline{IV}$ BUS-to-REGISTER → SOURCE → PRE-ALU
$\overline{IV}$ BUS-to-$\overline{IV}$ BUS → SOURCE → PRE-ALU

Same as MOVE operations, except source data is ADDed to contents of AUXiliary Register R0 via the ALU; if appropriate, Overflow Register R10 (OVF) is also set.

POST-ALU → DEST

AND OPERATIONS

REGISTER-to-REGISTER → SOURCE → PRE-ALU
REGISTER-TO-$\overline{IV}$ BUS → SOURCE → PRE-ALU
$\overline{IV}$ BUS-to-REGISTER → SOURCE → PRE-ALU
$\overline{IV}$ BUS-to-$\overline{IV}$ BUS → SOURCE → PRE-ALU

Same as MOVE operations, except source data is ANDed with contents of AUXiliary Register R0 via the ALU.

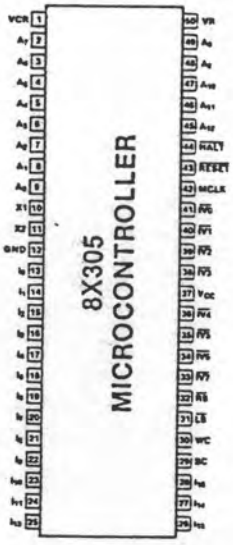
POST-ALU → DEST

EXCLUSIVE OR (XOR) OPERATIONS

REGISTER-to-REGISTER → SOURCE → PRE-ALU
REGISTER-TO-$\overline{IV}$ BUS → SOURCE → PRE-ALU
$\overline{IV}$ BUS-to-REGISTER → SOURCE → PRE-ALU
$\overline{IV}$ BUS-to-$\overline{IV}$ BUS → SOURCE → PRE-ALU

Same as MOVE operations, except source data is Exclusively ORed with contents of AUXiliary Register R0 via the ALU.

POST-ALU → DEST



PIN NO.	IDENTIFIER	FUNCTION
1	VCR	Regulated voltage input from series-pass transistor (2N5320 or equivalent).
2-9, 45-49	$A_0 - A_{12}$	**Program Address Lines:** These active-high outputs permit direct addressing of up to 8192 words of program storage; A_{12} is least significant bit.
10, 11	X1, X2	Timing generator connections for a capacitor, a series resonant crystal, or an external clock source with complementary outputs.
12	GND	Ground.
13-28	$I_0 - I_{15}$	**Instruction Lines:** These active-high input lines receive 16-bit instructions from program storage; I_{15} is least significant bit.
29	SC	**Select Command:** When high (binary 1), an address is being output on pins $\overline{IV0}$ through $\overline{IV7}$.
30	WC	**Write Command:** When high (binary 1), data is being output on pins $\overline{IV0}$ through $\overline{IV7}$.
31	$\overline{LB}$	**Left Bank Control:** When low (binary 0), devices connected to the Left Bank are accessed. (Note. *Typically, the $\overline{LB}$ signal is tied to the $\overline{ME}$ input pin of I/O peripherals).*
32	RB	**Right Bank Control:** When low (binary 0), devices connected to the Right Bank are accessed (Note. *Typically, the $\overline{RB}$ signal is tied to the $\overline{ME}$ input pin of I/O peripherals).*
33-36, 38-41	$\overline{IV0}$-$\overline{IV7}$	**Interface Vector** (Input/Output Bus) — these bidirectional active-low three-state lines communicate data and/or addresses to I/O devices and memory locations. A low voltage level equals a binary "1"; $\overline{IV7}$ is Least Significant Bit.
37	V_{CC}	+5V power supply.
42	MCLK	**Master Clock:** This active-high output signal is used for clocking I/O devices and/or synchronization of external logic.
43	$\overline{RESET}$	When $\overline{RESET}$ input is low, (binary 0), the 8X305 is initialized — sets Program Counter/Address Register to zero and inhibits MCLK. For the period of time $\overline{RESET}$ is low, the Left Bank/Right Bank ($\overline{LB}$/$\overline{RB}$) signals are forced high asynchronously.
44	$\overline{HALT}$	When $\overline{HALT}$ input is low (binary 0), internal operation of the 8X305 stops at the start of next instruction; MCLK is not inhibited nor is any internal register affected; however, both the Left Bank/Right Bank ($\overline{LB}$/$\overline{RB}$) signals are synchronously driven high during the first quarter of the instruction cycle time and remain high during the time HALT is low.
50	VR	Internally-generated reference output voltage for external series-pass regulator transistor.

Figure 2. Designations and Descriptions for Pins of 8X305 MicroController.

National Semiconductor

Data Communications Support

DP8340 Serial Bi-Phase Transmitter/Encoder

General Description

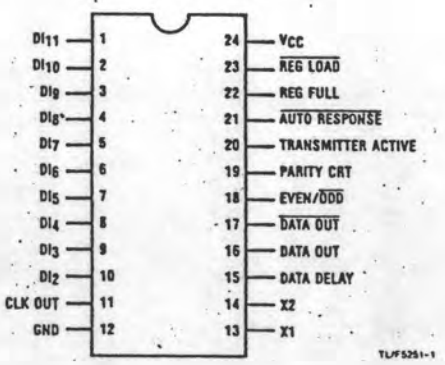
The DP8340 generates a complete encoding of parallel data for high speed serial transmission which conforms to the protocol as defined by the IBM 3270 information display system standard. The DP8340 converts parallel input data into a serial data stream. Although the IBM standard covers bi-phase serial data transmission over a coax line, the DP8340 also adapts to general high speed serial data transmission over other than coax lines, at frequencies either higher or lower than the IBM standard.

The DP8340 and its complementary chip, the DP8341 (receiver/decoder) have been designed to provide maximum flexibility in system designs. The separation of the transmitter/receiver functions provides convenient addition of more receivers at one end of a bi-phase line without the need of unused transmitters. This is specifically advantageous in control units where typical bi-phase data is multiplexed over many bi-phase lines and the number of receivers generally exceeds the number of transmitters.

Features

- Ten bits per data byte transmission
- Single-byte or multi-byte transmission
- Internal parity generation (even or odd)
- Internal crystal controlled oscillator used for the generation of all required chip timing frequencies
- Clock output directly drives receiver (DP8341) clock input
- Input data holding register
- Automatic clear status response feature
- Line drivers at data outputs provide easy interface to bi-phase coax line or general transmission lines
- <2ns driver output skew
- Bipolar technology provides TTL input/output compatibility
- Data outputs power up/down glitch free
- Internal power up clear and reset
- Single +5V power supply

Connection Diagram



Block Diagram

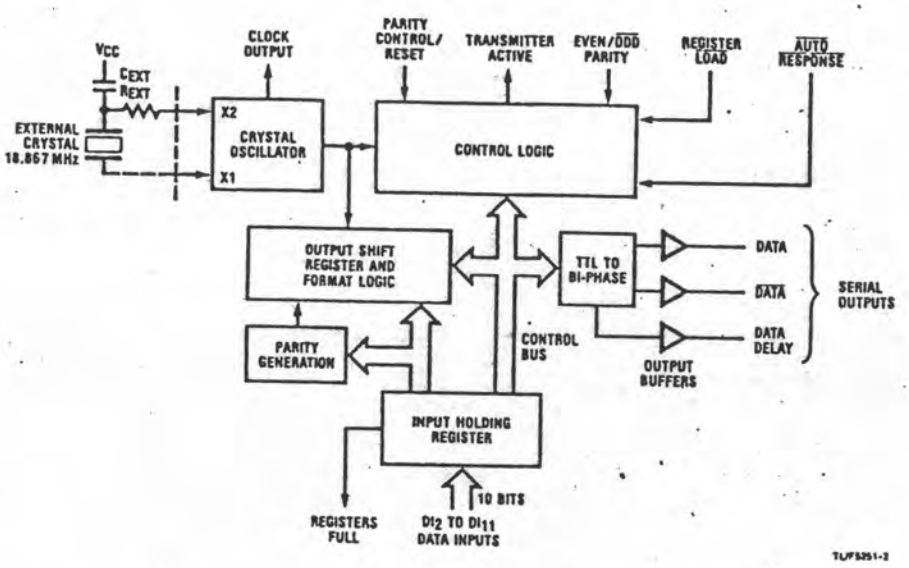


FIGURE 2. DP8340 Serial Bi-Phase Transmitter/Encoder Block Diagram

Block Diagram Functional Description

Figure 2 is a block diagram of the DP8340 Bi-Phase Transmitter/Encoder. The transmitter/encoder contains a crystal oscillator whose input is a crystal with a frequency eight (8) times the data rate. A Clock Output is provided to drive the DP8341 receiver/decoder Clock Input and other system components at the oscillator frequency. Additionally, the oscillator drives the control logic and output shift register/format logic blocks.

Data is parallel loaded from the system data bus to the transmitter/encoder's input holding register. This data is in turn loaded by the transmitter/encoder to its output shift register if this register was empty at the time of the load. During this load, message formatting and parity are generated. The formatted message is then shifted out at the bit rate frequency to the TTL to Bi-Phase block which generates the proper data bit formatting. The three data outputs, DATA, $\overline{\text{DATA}}$, and DATA DELAY provide for flexible interface to the coax line with a minimum of external components.

The Control Logic block interfaces to all blocks to insure proper chip operation and sequencing. It controls the type of parity generation through the Even/Odd Parity input. An additional feature provided by the transmitter/encoder is generation of odd parity and placement in bit 10 position while still maintaining even or odd parity in the bit 12 position. This is the format of data word byt and other commands in the 3270 Standard. The Par Control input is the pin which controls when t operation is in effect.

Another feature of the transmitter/encoder is the intern TT/AR (Transmission Turnaround/Auto Response) cap ability. After each Write type message from the contr unit in the 3270 Standard, the receiving unit mus respond with clean status (bits 2 through 11). With th transmitter/ encoder this function is accomplishe simply by forcing the Auto-Response input to the Log "0" state.

Operation of the transmitter/encoder is automati After the first data byte is loaded, the Transmitter Acti output is set and the transmitter/encoder immediatel formats the input data and serially shifts it out its data outputs. If the message is a multi-byte message, the internal format logic will modify the message data format for multibyte as long as the next byte is loaded to the input holding register before the last data bit o the previous data byte is transferred out of the interna output shift register. After all data is shifted out of the transmitter/encoder the Transmitter Active output wi return to the inactive state.

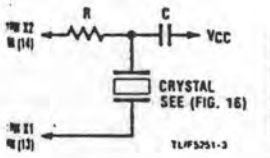
ıtailed Pin/Functional Description

ystal Inputs X1 and X2

e oscillator is controlled by an external, parallel reso-t crystal connected between the X1 and X2 pins. Nor-ly, a fundamental mode crystal is used to determine operating frequency of the oscillator; however, over-e mode crystals may be used.

ystal Specifications (Parallel Resonant)

pe	AT-cut crystal
erance	0.005% at 25°C
ability	0.01% from 0°C to +70°C
onance	Fundamental (Parallel)
imum Series Resistance	Dependent on Frequency (For 18.867 MHz, 50Ω)
d Capacitance	15pF



FREQ	R	C
<20 MHz	500Ω ± 10%	30pF
>20 MHz	120Ω ± 10%	15pF

FIGURE 3. Connection Diagram

the DP8340 transmitter is clocked by a system clock ystal oscillator not used), pin 13 (X1 input) should be cked directly using a Schottky series (74S) circuit. n 14 (X2 input) may be left open. The clocking frequency ust be set at eight times the data bit rate. Maximum put frequency is 28 MHz. For the IBM 3270 interface, is frequency is 18.867 MHz. At this frequency, the rial bit rate will be 2.358 Mbits/sec.

lock Output

e Clock Output is a buffered output derived directly m the crystal oscillator block and clocks at the oscilla-r frequency. It is designed to directly drive the DP8341 ceiver/decoder Clock Input as well as other system mponents.

gisters Full

s output is used as a flag by the external operating stem. A logic "1" (active state) on this output indicates at both the internal output shift register and the input lding register contain active data. No additional data ould be loaded until this output returns to the logic "0" te (inactive state).

nsmitter Active

s output will be in the logic "1" state while the trans-tter/encoder is about to transmit or in the process of nsmitting data. Otherwise, it will assume the logic "0" te indicating no data presently in either the input lding or output shift registers.

gister Load

e Register Load input is used to load data from the ta inputs to the input holding register, The loading function is edge sensitive, the data present during the logic "0" state of this input is loaded, and the input data must be valid before the logic "0" to logic "1" transition. It is after this transition that the transmitter/encoder begins formatting of data for serial transmission.

Auto Response (TT/AR)

This input provides for automatic clear data transmission (all bits in logic "0") without the need of loading all zero's. When a logic "0" is forced on this input the transmitter/encoder immediately responds with transmission of "clean status". This function is necessary after the completion of each write type command and in other functions in the 3270 specification. In the logic "1" state the transmitter/encoder transmits data entered on the Data Inputs.

Even/Odd Parity

This input sets the internal logic of the DP8340 transmitter/encoder to generate either even or odd parity for the data byte in the bit 12 position. When this pin is in the logic "0" state odd parity is generated. In the logic "1" state even parity is generated. This feature is useful when the control unit is performing a loop back check and at the same time the controller wishes to verify proper data transmission with its receiver/decoder.

Parity Control/Reset

Depending on the type of message transmitted, it is at times necessary in the IBM 3270 specification to generate an additional parity bit in the bit 10 position. The bit generated is odd parity on the previous eight (8) bits of data. When the Parity Control input is in the logic "1" state the data entered at the Data Bit 10 position is placed in the transmitted word. With the Parity Control input in the logic "0" state the Data Bit 10 input is ignored and odd parity on the previous data bits is placed in the normal bit 10 position while overall word parity (bit 12) is even or odd (controlled by Even/Odd Parity Input). This eliminates the need for external logic to generate the parity on the data bits.

Truth Table

Parity Control Input	Transmitted Data Bit 10
Logic "1"	Data entered on Data Input 10
Logic "0"	Odd Parity on 8-bit data byte

When this input is driven to a voltage that exceeds the power supply level (7V to 13V) the transmitter/encoder is reset.

Serial Outputs — DATA, DATA, and DATA DELAY

These three output pins provide for convenient application of data to the Bi-Phase Coax line (see Figure 15 for application). The Data outputs are a direct bit representation of the Bi-Phase data while the DATA DELAY output provides the necessary increment to clearly define the four (4) DC levels of the pulse. The DATA and DATA outputs add flexibility to the DP8340 transmitter/encoder for use in high speed differential line driving applications.

Functional Timing Waveforms — Message Format

Single Byte Transmission

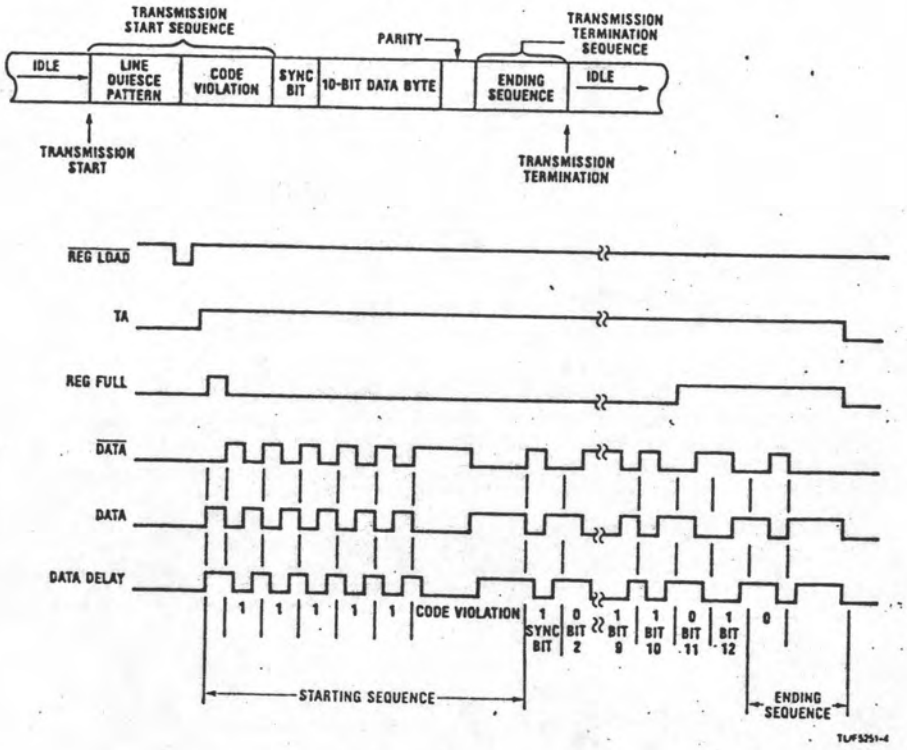


FIGURE 4. Overall Timing Waveforms for Single Byte

Multi-Byte Transmission

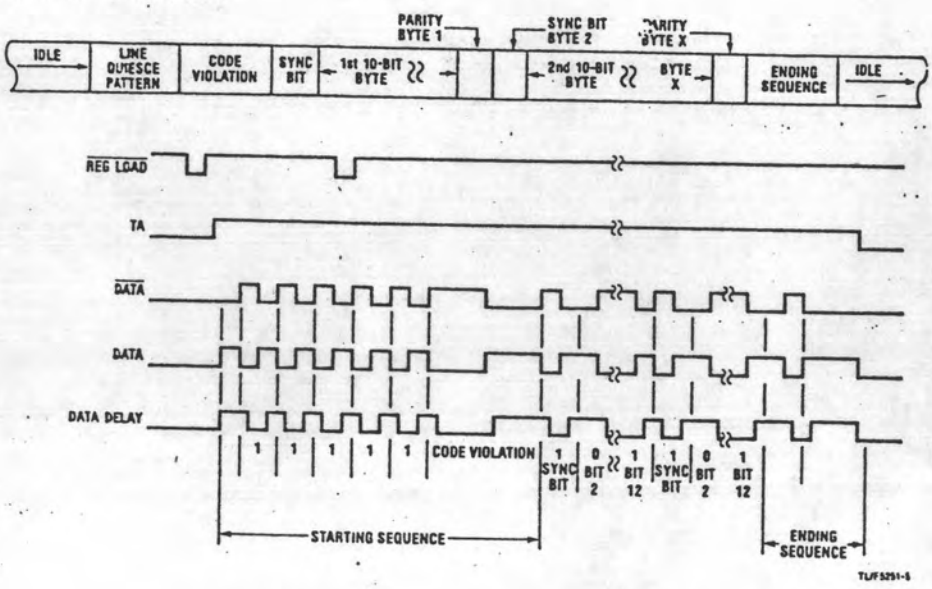


FIGURE 5. Overall Timing Waveforms for Multi-Byte

National Semiconductor

Data Communications Support

DP8341 Serial Bi-Phase Receiver/Decoder

General Description

The DP8341 provides complete decoding of data for high speed serial data communications. In specific, the DP8341 recognizes serial data that conforms to the IBM 3270 Information Display System Standard and converts it into ten (10) bits of parallel data. Although this standard covers Bi-Phase serial data transmission over a coax line, this device easily adapts to generalized high speed serial data transmission on other than coax lines at frequencies either higher or lower than the IBM 3270 standard.

The DP8341 receiver and its complementary chip, the DP8340 transmitter, are designed to provide maximum flexibility in system designs. The separation of transmitter and receiver functions allows addition of more receivers at one end of the Bi-Phase line without the necessity of adding unused transmitters. This is advantageous specifically in control units where typically Bi-Phase data is multiplexed over many Bi-Phase lines and the number of receivers generally outnumber the number of transmitters. The separation of transmitter and receiver function provides an additional advantage in flexibility of data bus organization. The data bus outputs of the receiver are TRI-STATE®, thus enabling the bus configuration to be organized as either a common transmit/receive (bi-directional) bus or as separate transmit and receive busses for higher speed.

Features

- DP8341 receives ten (10) bit data bytes and conforms to the IBM 3270 Interface Display System Standard
- Separate receiver and transmitter provide maximum system design flexibility
- Even parity detection
- High sensitivity input on receiver easily interfaces to coax line
- Standard TTL data input on receiver provides generalized transmission line interface and also provides hysteresis
- Data holding register
- Multi-byte or single byte transfers
- TRI-STATE receiver data outputs provide flexibility for common or separated transmit/receive data bus operation.
- Data transmission error detection on receiver provides for both error detection and error type definition
- Bi-polar technology provides TTL input/output compatibility with excellent drive characteristics
- Single +5V power supply operation

TRI-STATE® is a registered trademark of National Semiconductor Corp.

Connection Diagram

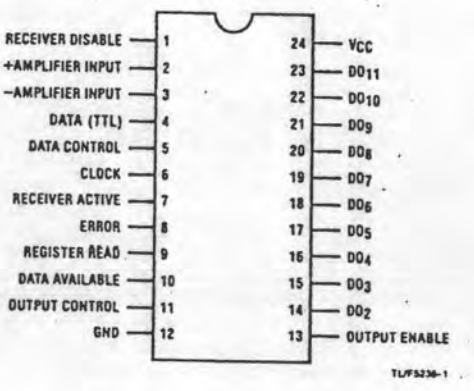


FIGURE 1. Pin-Out Diagram

Order Number DP8341J or DP8341N
See NS Package J24A or N24A

Diagram

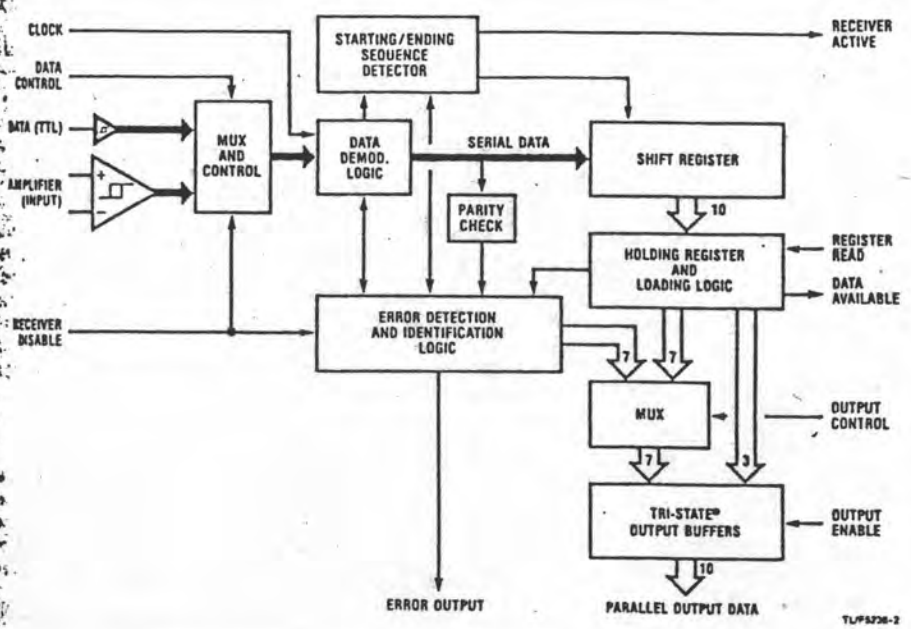


FIGURE 2. DP8341 Serial Bi-Phase Receiver/Decoder Block Diagram

Diagram Functional Description

2 is a block diagram of the DP8341. This chip is ...ally a serial in/parallel out shift register. However, ...rial input data must conform to a very specific for... (see *Figures 3-5*). The message will not be recognized ... the format of the starting sequence is correct. ...ions from the format in the data, sync bit, parity ...ding sequence will cause an error to be detected, ...ating the message.

...ters the receiver through the differential input am... for the TTL Data input. The differential amplifier is a ...sitivity input which may be used by connecting it ... to a transformer coupled coax line, or other trans... medium. The TTL Data input provides 400 mV of ...sis and recognizes TTL logic levels. The data then ... the demodulation block.

...a demodulation block samples the data at eight ... the data rate and provides signals for detecting ...rting sequence, ending sequence, and errors. ...ion of the starting sequence sets the Receiver ... output high and enables the input shift register.

... bits of data are shifted into the shift register, ...iver will verify that even parity is maintained on ... bits and the sync bit. After one complete data ... received, the contents of the input shift register ... loaded to the holding register, assuming the ... register is empty, and the Data Available output ... If the holding register is full, this load will be ... until that register has been read. If another data ... received when the shift register and the holding register are full a Data Overflow Error will be detected, terminating the message. Data is read from the holding register through the TRI-STATE Output Buffers. The Output Enable input is the TRI-STATE control for these outputs and the Register Read input signals the receiver that the read has been completed.

When the receiver detects an ending sequence the Receiver Active output will be reset to a logic "0" indicating the message has been terminated. A message will also terminate when an error is detected. The Receiver Active output used in conjunction with the Error output allows quick response to the transmitting unit when an error free message has been received.

The Error Detection and Identification block insures that valid data reaches the outputs of the receiver. Detection of an error sets the Error output to a logic "1" and resets the Receiver Active output to a logic "0" terminating the message. The error type may be read from the data bus outputs by setting the Output Control input to logic "0" and enabling the TRI-STATE outputs. The data bit outputs have assigned error definitions (see error code definition table). The Error output will return to a logic "0" when the next starting sequence is received, or when the error is read (Output Control to logic "0" and a Register Read performed).

The Receiver Disable input is used to disable both the amplifier and TTL Data receiver inputs. It will typically be connected directly to the Transmitter Active output of the DP8340 transmitter circuit (see *Figure 12*).

Detailed Functional Pin Description

Receiver Disable

This input is used to disable the receiver's data inputs. The Receiver Disable input will typically be connected to the Transmitter Active output of the DP8340. However, at the system controller it is necessary for both the transmitter and receiver to be active at the same time in the loop-back check condition. This variation can be accomplished with the addition of minimal external logic.

Truth Table

Receiver Disable	Data Inputs
Logic "0"	Active
Logic "1"	Disabled

Amplifier Inputs

The receiver has a differential input amplifier which may be directly connected to the transformer coupled coax line. The amplifier may also be connected to a differential type TTL line. The amplifier has 20 mV of hysteresis.

Data Input

This input can be used either as an alternate data input or as a power-up check input. If the system designer prefers to use his own amplifier, instead of the one provided on the receiver, then this TTL input may be used. Using this pin as an alternate data input allows self-test of the peripheral system without disturbing the transmission line.

Data Control

This input is the control pin that selects which of the inputs are used for data entry to the receiver.

Truth Table

Data Control	Data Input To
Logic "0"	Data Input
Logic "1"	Amplifier Inputs

Note: This input is also used for testing. When the inut voltage is raised to 7.5 V the chip resets.

Clock Input

This input is the internal clock of the receiver. It must be set at eight (8) times the line data bit rate. For the IBM 3270 Standard, this frequency is 18.87 MHz or a data bit rate of 2.358 MHz. The crystal-controlled oscillator provided in the DP8340 transmitter also operates [illegible] frequency. The Clock Output of the transmitter [illegible] signed to directly drive the receiver's Clock [illegible] addition, the receiver is designed to operate [illegible] a data bit rate of 3.5 MHz.

Receiver Active

The purpose of this output is to inform the exter[illegible] tem when the DP8341 is in the process of rece[illegible] message. This output will transition to a logic "1" after the receipt of a valid starting sequence transition to logic "0" when a valid ending sequence received or an error is detected. This output com[illegible] with the Error output will inform the operating sy[illegible] the end of an error free data transmission.

Error

The Error output transitions to a logic "1" when an [illegible] is detected. Detection of an error causes the Rec[illegible] Active and the Data Available outputs to transition logic "0". The Error output returns to a logic "0" the error register has been read or when the next sta[illegible] sequence is detected.

Register Read

The Register Read input when driven to the logic state signals the receiver that data in the holding [illegible] ter is being read by the external operating system, data present in the holding register will continue t[illegible] main valid until the Register Read input returns t[illegible] logic "1" condition. At this time, if an additional by[illegible] present in the input shift register it will be transfer[illegible] the holding register, otherwise the data will remain in the holding register. The Data Available output g[illegible] in the logic "0" state for a short interval while a byte is transferred to the holding register after a [illegible] ter read.

Data Available

This output indicates the existence of a data byte in the output holding register. It may also indicate presence of a data byte in both the holding register the input shift register. This output will transition to logic "1" state as soon as data is available and rem[illegible] the logic "0" state after each data byte has been r[illegible] However, even after the last data byte has been read the Data Available output has assumed the logic state, the last data byte read from the holding reg[illegible] will remain until new data has been received.

...Control

...Control Input determines the type of information appearing at the data outputs. In the logic "1" state data will appear, in the logic "0" state error codes

Truth Table

...put Control	Data Outputs
Logic "0"	Error Codes
Logic "1"	Data

...Enable

...Enable Input controls the state of the TRI-...outputs.

Truth Table

...Output Enable	TRI-STATE® Data Outputs
Logic "0"	Disabled
Logic "1"	Active

...Outputs

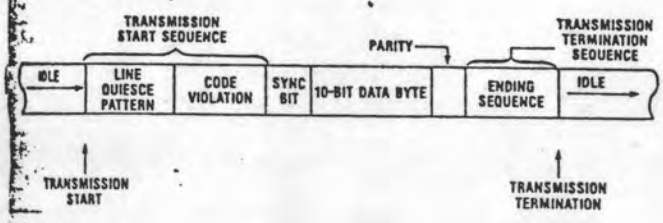
...has a ten (10) bit TRI-STATE data bus. Seven ...multiplexed with error bits. The error bits are defined in the table below. The Output Control input is the multiplexer control for the Data/Error bits.

Error Code Definition

Data Bit	Error Type
DO2	Data Overflow (Byte not removed from holding register when it and the input shift register are both full and new data is received)
DO3	Parity Error (Odd parity detected)
DO4	Transmit Check conditions (existence of errors on any or all of the following data bits: DO3, DO5, and DO6
DO5	An invalid ending sequence
DO6	Loss of mid-bit transition detected at other than normal ending sequence time
DO7	New starting sequence detected before data byte in holding register has been read
DO8	Receiver disabled during receiver active mode

...age Format

Single Byte Transmission



Multi-Byte Transmission

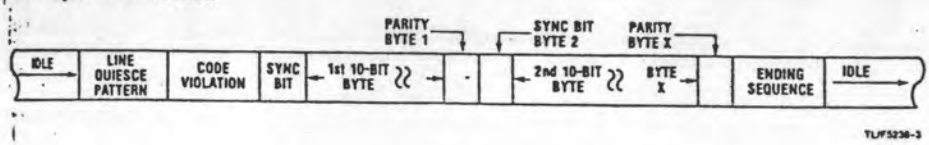


FIGURE 3. IBM 3270 Message Format

Message Format

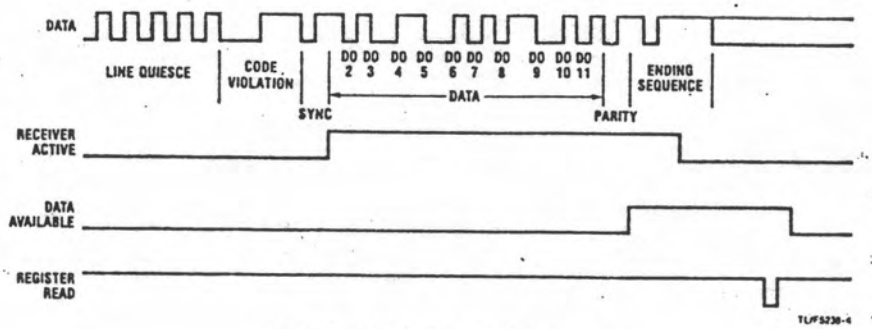


FIGURE 4A. Single Byte Message

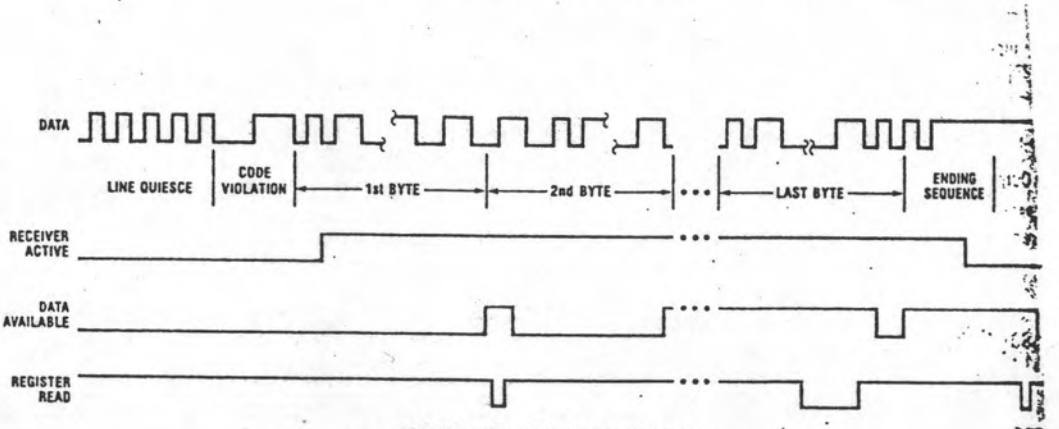


FIGURE 4B. Multi-Byte Message

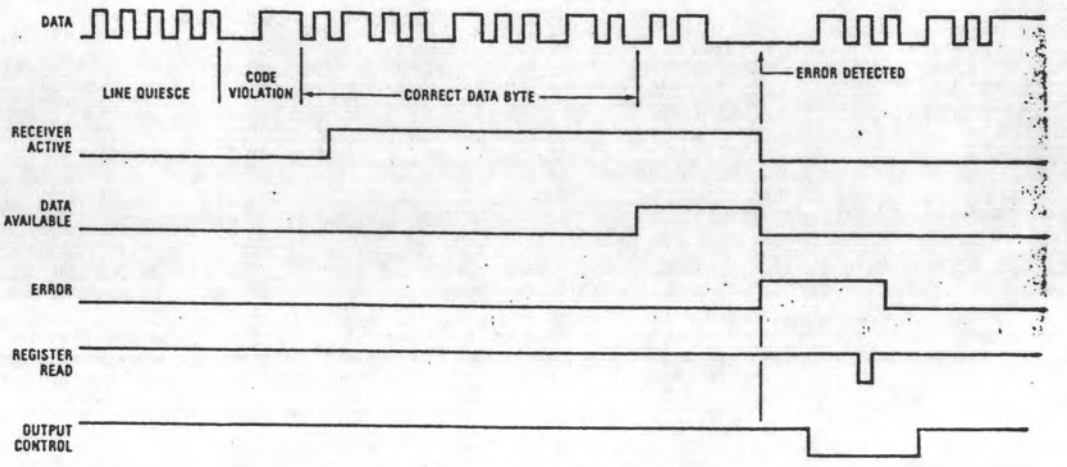
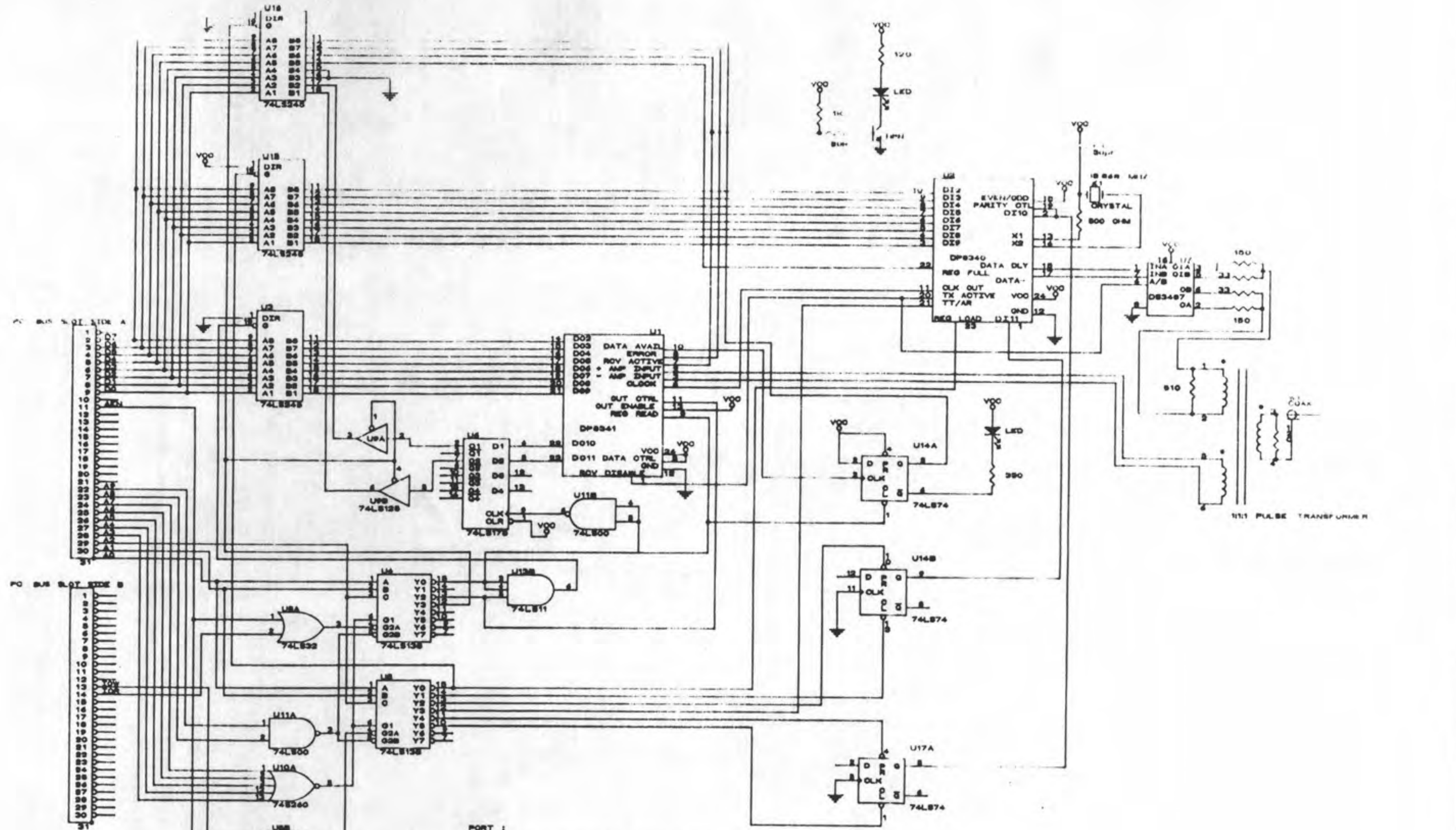


FIGURE 5. Message with Error

Appendix L

Circuit Diagram



PORT :

IN 300H, READ 8 BIT DATA AND CLEAR DATA AVAILABLE FLIP-FLOP
IN 301H, NOT USED
IN 302H, READ ERROR STATUS
IN 303H, READ RECEIVER STATUS

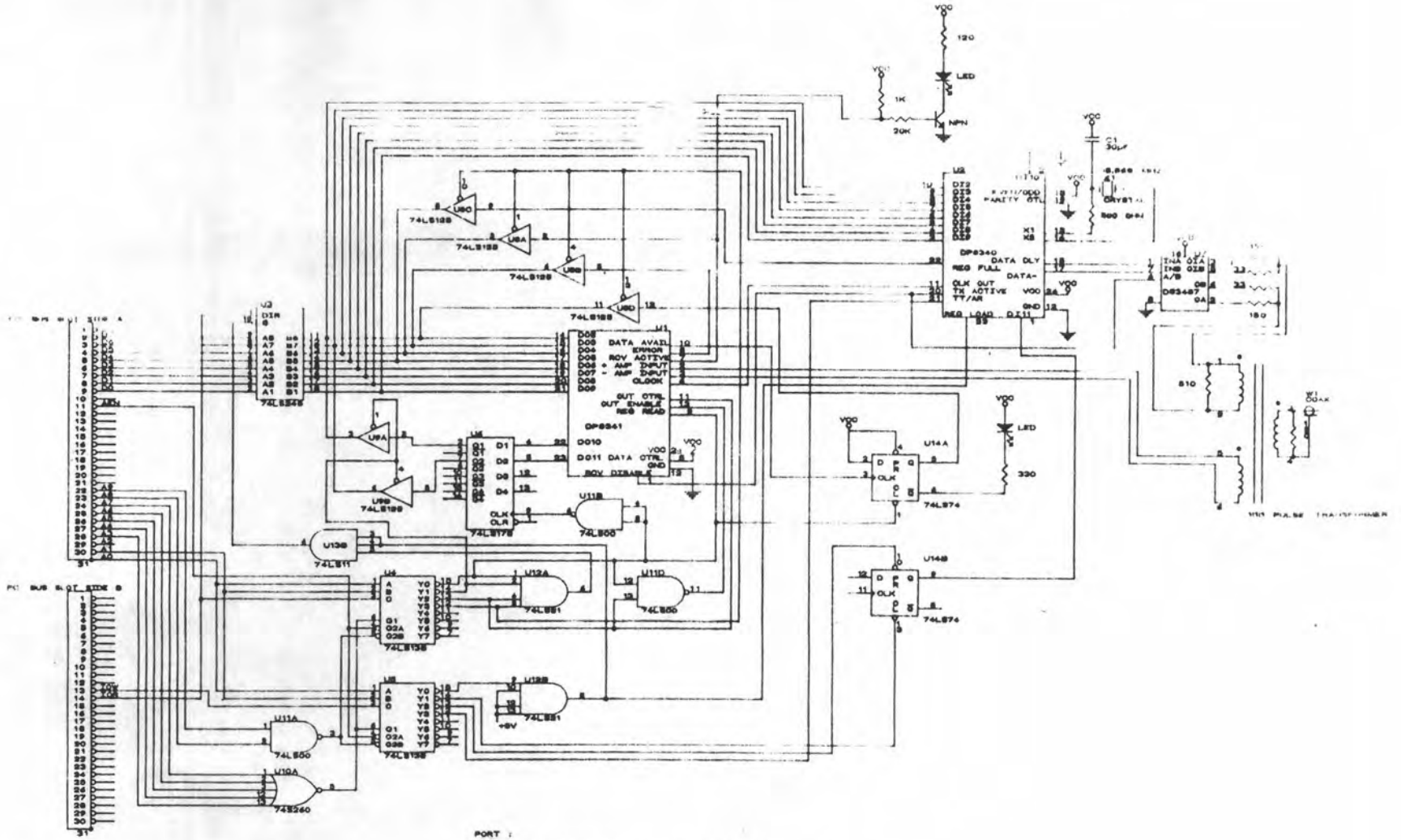
D7 = DATA AVAILABLE	D3 = NOT USED
D6 = ERROR	D2 = NOT USED
D5 = RECEIVER ACTIVE	D1 = DATA FROM b10
D4 = REGISTER FULL	D0 = DATA FROM b11

OUT 300H, OUTPUT DATA TO DP8340   OUT 304H, SET b10 AND PARITY CTRL
OUT 301H, CLEAR BIT b11   OUT 305H, CLEAR b10 AND PARITY CTRL
OUT 302H, SET BIT b11
OUT 303H, AUTO RESPONSE (TT/AR)

IBM3270 Coaxial Type A Interface Board
Size C | Document Number BY : SUMCHAI ASAVAKUL | Rev 1
Date: April 17, 1989 | Sheet 1 of 1

Appendix M

Early Design's Circuit Diagram



PORT :

IN 300H, READ 8 BIT DATA AND CLEAR DATA AVAILABLE FLIP-FLOP
IN 301H, READ b10 AND b11 DATA
IN 302H, READ ERROR STATUS
IN 303H, READ RECEIVER STATUS
D7 = DATA AVAILABLE
D6 = ERROR
D5 = RECEIVER ACTIVE
D4 = REGISTER FULL

OUT 300H, OUTPUT DATA TO DP8340
OUT 301H, CLEAR BIT b11
OUT 302H, SET BIT b11
OUT 303H, AUTO RESPONSE (TT/AR)

IBM1270 Coaxial Type A Interface Board

Size: 0 | Document Number: | BY : SOMJHAI ASAVAKUL

Date: March 17, 1988 | Sheet 1 of 1

Appendix N

Listing of Program using 3270CIB to Capture Data With Data Bit 10 and Bit 11

```
CODE    SEGMENT PARA    PUBLIC  'CODE'
        ASSUME  CS:CODE, DS:CODE        ;ALL SEGMENT SET TO CODE
        ORG     100H

BEGIN:  JMP     SET_UP  ;JUMP OVER THE RESIDENT ROUTINE

DATA1:  DB      50000 DUP(20H)

ROUTINE PROC    NEAR    ;START OF RESIDENT PROGRAM

        STI             ;ENABLE MASKABLE INTERRUPT


        PUSH    DS
        PUSH    CS
        POP     DS
        MOV     CX,10000
        MOV     BX,OFFSET DATA1     ;  MIN      MAX  CPU CYCLES.
LOOP:   MOV     DX,303H             ;   2       19
NOT:    IN      AL,DX               ;   5        5
        AND     AL,80H              ;   2        7
        JZ      NOT                 ;   7        7
        MOV     DX,300H             ;   2       19
        IN      AL,DX               ;   5        5
        MOV     AH,AL               ;   2       19
        MOV     DX,303H             ;   2       19
        IN      AL,DX               ;   5        5
        AND     AL,03H              ;   2        7
        MOV     [BX],AX             ;   2       19
        INC     BX                  ;   2        7
        INC     BX                  ;   2        7
        LOOP    LOOP                ;   8        8
        POP     DS              ;TOTAL  48      153
        iret

ROUTINE ENDP

FINISH  LABEL   BYTE
;-------------------------------------------------------------------------
;MAIN PROGRAM, THIS PART WILL NOT BE RESIDENT
;-------------------------------------------------------------------------
```

```
SET_UP:                 ;MAIN PROGRAM TO INSTALL INTERRUPT ROUTINE



        MOV     AL,73H  ;CHECK IF INTERRUPT ALREADY INSTALL
        MOV     AH,35H
        INT     21H
        CMP     BX,OFFSET ROUTINE
        JNE     RES
        JMP     QUIT

RES:    MOV     DX,OFFSET ROUTINE               ;CHANGE INT VEC TO OUR ROUTINE
        MOV     AL,73H  ;INTERUPT VECTOR NUMBER
        MOV     AH,25H  ;FUNCTION TO SET VECTOR
        INT     21H     ;SET VECTOR

        MOV     DX,OFFSET FINISH + 1    ;TERMINATE AND STAY RESIDENT
        INT     27H

QUIT:   INT     20H     ;TERMINATE BUT NOT STAY RESIDENT


CODE    ENDS
        END     BEGIN
```

Appendix O

Listing of Program using 3270CIB to Capture Data Without Data Bit 10 and Bit 11

```
CODE    SEGMENT PARA    PUBLIC  'CODE'
        ASSUME  CS:CODE, DS:CODE        ;ALL SEGMENT SET TO CODE
        ORG     100H

BEGIN:  JMP     SET_UP  ;JUMP OVER THE RESIDENT ROUTINE

DATA1:  DB      50000 DUP(20H)

ROUTINE PROC    NEAR    ;START OF RESIDENT PROGRAM

        STI             ;ENABLE MASKABLE INTERRUPT


        PUSH    DS
        PUSH    CS
        POP     DS
        MOV     CX,10000
        MOV     BX,OFFSET DATA1     ;  MIN      MAX  CPU CYCLES.
LOOP:   MOV     DX,303H             ;   2       19
NOT:    IN      AL,DX               ;   5        5
        AND     AL,80H              ;   2        7
        JZ      NOT                 ;   7        7
        MOV     DX,300H             ;   2       19
        IN      AL,DX               ;   5        5
        MOV     [BX],AL             ;   2       19
        INC     BX                  ;   2        7
        LOOP    LOOP                ;   8        8
        POP     DS              ;TOTAL  35       96
        iret

ROUTINE ENDP

FINISH  LABEL   BYTE
;-------------------------------------------------------------------------
;MAIN PROGRAM, THIS PART WILL NOT BE RESIDENT
;-------------------------------------------------------------------------


SET_UP:                 ;MAIN PROGRAM TO INSTALL INTERRUPT ROUTINE

        MOV     AL,73H  ;CHECK IF INTERRUPT ALREADY INSTALL
        MOV     AH,35H
        INT     21H
```

```
        CMP     BX,OFFSET ROUTINE
        JNE     RES
        JMP     QUIT

RES:    MOV     DX,OFFSET ROUTINE               ;CHANGE INT VEC TO OUR ROUTINE
        MOV     AL,73H  ;INTERUPT VECTOR NUMBER
        MOV     AH,25H  ;FUNCTION TO SET VECTOR
        INT     21H     ;SET VECTOR

        MOV     DX,OFFSET FINISH + 1    ;TERMINATE AND STAY RESIDENT
        INT     27H

QUIT:   INT     20H     ;TERMINATE BUT NOT STAY RESIDENT


CODE    ENDS
        END     BEGIN
```

Appendix P

Listing of Program using 3270CIB to Interface PC to IBM 3278 and emulate PC as IBM 3274

```
TRUE              EQU     1         ;TRUE STATUS
FALSE             EQU     0         ;FLASE STATUS
TIME_OUT          EQU     20        ;TIME OUT FOR POLLING OF RECEIVER
ONE_ROW           EQU     50H       ;ONE ROW OF 80 COLUMNS
FIRST_ROW_OFFSET  EQU     50H       ;OFFSET FOR CALCULATE THE OFFSET
LAST_ROW_OFFSET   EQU     80H       ;LAST ROW OFFSET FOR CALCULATE THE OFFSET
UPPER_LEFT        EQU     0050H     ;UPPER LEFT MOST POSITION
LOWER_RIGHT       EQU     07CFH     ;LOWER RIGHT MOST POSITION
LOWER_LEFT        EQU     0780H     ;LOWER LEFT MOST POSITION
MASK_TX           EQU     10H       ;MASK FOR TX FULL STATUS
MASK_RX           EQU     80H       ;MASK FOR RX DATA AVAILABLE
INHIBIT_DSP_MASK  EQU     08H       ;MASK FOR SET 3278 DISPLAY INHIBIT
INHIBIT_CUR_MASK  EQU     04H       ;MASK FOR SET 3278 CURSOR INHIBIT
NORMAL_DSP_MASK   EQU     00H       ;MASK FOR SET 3278 DISPLAY TO NORMAL
MASK_8BIT         EQU     0FFH      ;SET MASK REG FOR COMPARE ALL 8 BIT
TT_AR             EQU     00H       ;TRANSMIT TURNAROUND/AUTO RESPONSE CHARACTER
POLL_CHAR         EQU     01H       ;POLLING CHARACTER
POLL_ACK          EQU     11H       ;POLL ACKNOWLEDGE CHARACTER
SHF_PRESS         EQU     4DH       ;SCAN CODE FOR SHIFT KEY PRESSED
SHF_RELEASE       EQU     0CDH      ;SCAN CODE FOR SHIFT KEY RELEASED
CAP_PRESS         EQU     4CH       ;SCAN CODE FOR CAPLOCK KEY PRESS
CAP_RELEASE       EQU     0CCH      ;SCAN CODE FOR CAPLOCK KEY RELEASED
LEFT_ARROW        EQU     16H       ;SCAN CODE FOR LEFT ARROW
RIGHT_ARROW       EQU     1AH       ;SCAN CODE FOR RIGHT ARROW
UP_ARROW          EQU     0EH       ;SCAN CODE FOR UP ARROW
DOWN_ARROW        EQU     13H       ;SCAN CODE FOR DOWN ARROW
WRITE_DATA        EQU     0CH       ;FUNCTION WRITE DATA
LOAD_ADDR_LO      EQU     14H       ;FUNCTION LOAD LOW BYTE BUFFER ADDRESS COUNTER
LOAD_ADDR_HI      EQU     04H       ;FUNCTION LOAD HIGH BYTE BUFFER ADDRESS COUNTER
READ_ADDR_LO      EQU     15H       ;FUNCTION READ LOW BYTE BUFFER ADDRESS COUNTER
READ_ADDR_HI      EQU     05H       ;FUNCTION READ HIGH BYTE BUFFER ADDRESS COUNTER
RESET             EQU     02H       ;FUNCTION RESET
READ_DATA         EQU     03H       ;FUNCTION READ DATA
CLEAR             EQU     06H       ;FUNCTION CLEAR
READ_TERM_ID      EQU     09H       ;FUNCTION READ TERMINAL ID
LOAD_CTRL_REG     EQU     0AH       ;FUNCTION LOAD CONTROL REGISTER
READ_STATUS       EQU     0DH       ;FUNCTION READ STATUS
INSERT            EQU     0EH       ;FUNCTION INSERT
FORWARD_SEARCH    EQU     10H       ;FUNCTION SEARCH FORWARD
BACKWARD_SEARCH   EQU     12H       ;FUNCTION SEARCH BACKWARD
LOAD_MASK         EQU     16H       ;FUNCTION LOAD MASK
SET_b11           EQU     302H      ;HARDWARE PORT FOR SET BIT b11
CLEAR_b11         EQU     301H      ;HARDWARE PORT FOR CLEAR BIT b11
```

```
DATA_PORT       EQU     300H    ;HARDWARE PORT FOR DATA I/O
STATUS_PORT     EQU     303H    ;HARDWARE PORT FOR STATUS
BUFFER_CH_1     EQU     21H     ;BUFFER CODE FOR '1'
BUFFER_CH_A     EQU     0A0H    ;BUFFER CODE FOR 'A'

;
;THIS MODULE CONTAINS SUBROUTINES WHICH CONTROL THE INTERFACE TO DOS

        NAME    PCIO
        PUBLIC  PC_DSP,CHK_KEY,GET_KEY

CODE    SEGMENT PARA    PUBLIC  'CODE'
        ASSUME  CS:CODE, DS:CODE

;
;SUBROUTINE CHK_KEY:
;THIS SUBROUTINE CALL DOS INT16 FUNTION 01, TO CHECK WHETHER A KEY WAS PRESS.
;ZERO FLAG IS SET IF A KEY WAS PRESS.
;
CHK_KEY:        MOV     AH,01
                INT     16H

                RET

;
;SUBROUTINE GET_KEY:
;THIS SUBROUTINE CALL DOS INT16 FUNTION 0, TO READ THE KEYBOARD.
;KEYBOARD CHARACTER IS PUT IN REG AL.
;
GET_KEY:        MOV     AH,0
                INT     16H

                RET

;
;SUBROUTINE PC_DSP:
;THIS SUBROUTINE CALL DOS INT21 FUNTION 2, TO DISPLAY CHARACTER ON PC.
;CHARACTER IN REG AL. WILL BE DISPLYED.
;
PC_DSP: MOV     AH,02
        MOV     DL,AL
        INT     21H

        RET

CODE    ENDS
        END
;
;END PCIO.ASM

;
;THIS MODULE CONTAINS SUBROUTINES WHICH HANDLE COMMAND INTERFACE
;OF THE IBM3278

        PUBLIC  DSP_CHAR,DISPLAY
```

```
        PUBLIC  RD_HI_ADDR,RD_LO_ADDR
        PUBLIC  SET_HI_ADDR,SET_LO_ADDR
        PUBLIC  OUT_COMMAND,OUT_DATA,CHECK_RX,POLL
        EXTRN   HIBYTE:BYTE,LOBYTE:BYTE,REPLY:BYTE,TABLE:BYTE

INCLUDE EQU.INC

CODE    SEGMENT PARA    PUBLIC  'CODE'
        ASSUME  CS:CODE, DS:CODE
;
;SUBROUTINE DSP_CHAR:
;THIS SUBROUTINE WRITE DATA IN REG AL. IT USES THE WRITE DATA COMMAND.
;
DSP_CHAR:
        PUSH    AX
        MOV     AH,WRITE_DATA           ;SET COMMAND
        CALL    OUT_COMMAND
        PUSH    AX
        CALL    CHECK_RX                ;DISREGARD RESPONSE FROM 3278
        POP     AX
        MOV     AH,AL
        CALL    OUT_DATA
        CALL    CHECK_RX                ;DISREGARD RESPONSE FROM 3278
        POP     AX

        RET

;
;SUBROUTINE DISPLAY:
;THIS SUBROUTINE WRITE STRINGS OF DATA POINT TO BY REGISTER BX, WHILE
;REGISTER CX CONTAINS THE NUMBER OF BYTES TO BE OUTPUTED. IT USES THE
;WRITE DATA COMMAND. THIS SUBROUTINE ALSO CONVERT ASCII TO BUFFER CODE.
;
DISPLAY:
        PUSH    AX
        MOV     AH,WRITE_DATA           ;SET COMMAND
        CALL    OUT_COMMAND
        CALL    CHECK_RX                ;DISREGARD RESPONSE FROM 3278

LOOP:   MOV     AL,BYTE PTR [BX]        ;DISPLAY DATA UNTIL FINISHES

        PUSH    BX                      ;CONVERT ASCII TO BUFFER CODE
        MOV     BX,OFFSET TABLE
        XLAT
        MOV     AH,AL
        POP     BX

        CALL    OUT_DATA
        CALL    CHECK_RX                ;DISREGARD RESPONSE FROM 3278
        INC     BX                      ;GET NEXT BYTE
        LOOP    LOOP

        POP     AX

        RET
```

```
;
;SUBROUTINE RD_HI_ADDR:
;THIS SUBROUTINE USES THE READ BUFFER ADDRESS COUNTER HIGH BYTE COMMAND,
;THE HI_BYTE READ WAS PLACE IN VARIABLE 'HIBYTE'.
;
RD_HI_ADDR:
        PUSH    AX

        MOV     AH,READ_ADDR_HI         ;SET COMMAND
        CALL    OUT_COMMAND

        CALL    CHECK_RX

        MOV     HIBYTE,AL               ;PLACE DATA IN 'HIBYTE'

        POP     AX

        RET

;
;SUBROUTINE RD_LO_ADDR:
;THIS SUBROUTINE USES THE READ BUFFER ADDRESS COUNTER LOW BYTE COMMAND,
;THE LO_BYTE READ WAS PLACE IN VARIABLE 'LOBYTE'.
;
RD_LO_ADDR:
        PUSH    AX

        MOV     AH,READ_ADDR_LO         ;SET COMMAND
        CALL    OUT_COMMAND

        CALL    CHECK_RX

        MOV     LOBYTE,AL               ;PLACE DATA IN 'LOBYTE'

        POP     AX

        RET

;
;SUBROUTINE SET_HI_ADDR:
;THIS SUBROUTINE USES THE LOAD BUFFER ADDRESS COUNTER HIGH BYTE COMMAND,
;THE HIGH BYTE ADDRESS TO BE SET WAS TAKEN FROM VARIABLE 'HIBYTE'.
;
SET_HI_ADDR:
        PUSH    AX

        MOV     AH,LOAD_ADDR_HI         ;SET COMMAND
        CALL    OUT_COMMAND

        CALL    CHECK_RX                ;DISREGARD RESPONSE FROM 3278

        MOV     AH,HIBYTE               ;SET POSITION
        CALL    OUT_DATA

        CALL    CHECK_RX                ;DISREGARD RESPONSE FROM 3278
```

```
        POP     AX

        RET

;
;SUBROUTINE SET_LO_ADDR:
;THIS SUBROUTINE USES THE LOAD BUFFER ADDRESS COUNTER LOW BYTE COMMAND,
;THE LOW BYTE ADDRESS TO BE SET WAS TAKEN FROM VARIABLE 'LOBYTE'.
;
SET_LO_ADDR:
        PUSH    AX

        MOV     AH,LOAD_ADDR_LO         ;SET COMMAND
        CALL    OUT_COMMAND

        CALL    CHECK_RX                ;DISREGARD RESPONSE FROM 3278

        MOV     AH,LOBYTE               ;SET POSITION
        CALL    OUT_DATA

        CALL    CHECK_RX                ;DISREGARD RESPONSE FROM 3278

        POP     AX

        RET

;
;SUBROUTINE OUT_COMMAND:
;THIS SUBROUTINE OUTPUT CONTENTS IN REGISTER AH AS COMMAND WORD
;
OUT_COMMAND:
        PUSH    AX

        CALL    CHECK_TX        ;WAIT UNTIL TRANSMITTER IS EMPTY
        MOV     AL,AH

        MOV     DX,SET_b11      ;SET AS COMMAND WORD
        OUT     DX,AL

        MOV     DX,DATA_PORT    ;OUTPUT COMMAND
        OUT     DX,AL

        POP     AX

        RET

;
;SUBROUTINE OUT_DATA:
;THIS SUBROUTINE OUTPUT CONTENTS IN REGISTER AH AS DATA WORD
;
OUT_DATA:
        PUSH    AX

        CALL    CHECK_TX        ;WAIT UNTIL TRANSMITTER IS EMPTY
        MOV     AL,AH
```

```
        MOV     DX,CLEAR_b11    ;SET AS DATA WORD
        OUT     DX,AL

        MOV     DX,DATA_PORT    ;OUTPUT DATA
        OUT     DX,AL

        POP     AX

        RET

;
;SUBROUTINE CHECK_TX:
;THIS SUBROUTINE LOOP UNTIL TRANSMITTER IS EMPTY
;
CHECK_TX:
        PUSH    AX

        MOV     DX,STATUS_PORT  ;WAIT UNTIL TRANSMITTER IS EMPTY
FULL:   IN      AL,DX
        AND     AL,MASK_TX
        JNZ     FULL

        POP     AX

        RET

;
;SUBROUTINE CHECK_RX:
;THIS SUBROUTINE LOOP UNTIL RECEIVER HAS DATA, DATA IS READ INTO REG AL.
;REG AH = 0 IF NO RESPONSE WAS RECEIVED.
;
CHECK_RX:
        MOV     AH,TIME_OUT
        MOV     DX,STATUS_PORT  ;WAIT UNTIL RECEIVER HAS DATA
WAIT:   CMP     AH,0
        JZ      SKIP
        DEC     AH
        IN      AL,DX
        AND     AL,MASK_RX
        JZ      WAIT
SKIP:   MOV     DX,DATA_PORT
        IN      AL,DX

        MOV     REPLY,AH


        RET

;
;SUBROUTINE POLL:
;THIS SUBROUTINE OUTPUT POLL COMMAND TO 3278 UNTIL RESPONSE WAS RECEIVE,
;THE RECEIVE DATA WAS PUT IN REG AL.
;
POLL:
        MOV     AH,POLL_CHAR            ;POLL 3278 (POLL COMMAND = 01H)
```

```
        CALL    OUT_COMMAND
        CALL    CHECK_RX                ;WAIT UNTIL RESPONSE IS POR (02H)
        CMP     BYTE PTR REPLY,0
        JZ      POLL

        RET

CODE    ENDS
        END
;
;END 3278IO.ASM

;
;THIS MODULE CONTAINS SUBROUTINES WHICH CONTROL THE CURSOR MOVEMENTS
;OF THE IBM3278

        NAME    CURSOR
        PUBLIC  CUR_LT,CUR_RT,CUR_UP,CUR_DN
        EXTRN   RD_HI_ADDR:NEAR,RD_LO_ADDR:NEAR
        EXTRN   SET_HI_ADDR:NEAR,SET_LO_ADDR:NEAR
        EXTRN   HIBYTE:BYTE,LOBYTE:BYTE

INCLUDE EQU.INC

CODE    SEGMENT PARA    PUBLIC  'CODE'
        ASSUME  CS:CODE, DS:CODE

;
;SUBROUTINE CUR_LT:
;THIS SUBROUTINE READ THE CURSOR POSITION FROM 3278 AND MOVE IT LEFT ONE
;POSITION, IF CUSOR AT UPPER LEFT THEN IT IS MOVED TO THE LOWER RIGHT.
;
CUR_LT:
        PUSH    AX

        CALL    RD_HI_ADDR              ;GET CURRENT CURSOR POSITION
        CALL    RD_LO_ADDR

        MOV     AH,HIBYTE               ;DECREMENT POSITION
        MOV     AL,LOBYTE
        DEC     AX

        CMP     AX,UPPER_LEFT           ;SET TO LOWER RIGHT IF AT UPPER LEFT
        JAE     SET1
        MOV     AX,LOWER_RIGHT

SET1:   MOV     HIBYTE,AH
        MOV     LOBYTE,AL
        CALL    SET_HI_ADDR
        CALL    SET_LO_ADDR
        POP     AX

        RET

;
```

```
;SUBROUTINE CUR_RT:
;THIS SUBROUTINE READ THE CURSOR POSITION FROM 3278 AND MOVE IT RIGHT ONE
;POSITION, IF CUSOR AT LOWER RIGHT THEN IT IS MOVED TO THE UPPER LEFT.
;
CUR_RT:
        PUSH    AX

        CALL    RD_HI_ADDR              ;GET CURRENT CURSOR POSITION
        CALL    RD_LO_ADDR

        MOV     AH,HIBYTE               ;INCREMENT POSITION
        MOV     AL,LOBYTE
        INC     AX

        CMP     AX,LOWER_RIGHT          ;SET TO UPPER LEFT IF AT LOWER RIGHT
        JBE     SET2
        MOV     AX,UPPER_LEFT

SET2:   MOV     HIBYTE,AH
        MOV     LOBYTE,AL
        CALL    SET_HI_ADDR
        CALL    SET_LO_ADDR
        POP     AX

        RET


;
;SUBROUTINE CUR_UP:
;THIS SUBROUTINE READ THE CURSOR POSITION FROM 3278 AND MOVE IT UP ONE ROW,
;IF CUSOR AT FIRST ROW THEN IT IS MOVED TO THE LAST SCREEN ROW.
;
CUR_UP:
        PUSH    AX

        CALL    RD_HI_ADDR              ;GET CURRENT CURSOR POSITION
        CALL    RD_LO_ADDR

        MOV     AH,HIBYTE               ;DECREMENT BY ONE ROW
        MOV     AL,LOBYTE
        SUB     AX,ONE_ROW

        CMP     AX,UPPER_LEFT           ;SET TO LAST SCREEN ROW AT FIRST ROW
        JAE     SET3
        MOV     AX,LOWER_LEFT
        MOV     DX,0
        MOV     DL,LOBYTE
        SUB     DL,FIRST_ROW_OFFSET
        ADD     AX,DX

SET3:   MOV     HIBYTE,AH
        MOV     LOBYTE,AL
        CALL    SET_HI_ADDR
        CALL    SET_LO_ADDR
        POP     AX
```

```
        RET

;
;SUBROUTINE CUR_DN:
;THIS SUBROUTINE READ THE CURSOR POSITION FROM 3278 AND MOVE IT DOWN ONE ROW,
;IF CUSOR AT LAST ROW THEN IT IS MOVED TO THE FIRST SCREEN ROW.
;
CUR_DN:
        PUSH    AX

        CALL    RD_HI_ADDR              ;GET CURRENT CURSOR POSITION
        CALL    RD_LO_ADDR

        MOV     AH,HIBYTE               ;INCREMENT BY ONE ROW
        MOV     AL,LOBYTE
        ADD     AX,ONE_ROW

        CMP     AX,LOWER_RIGHT          ;SET TO LAST SCREEN ROW AT FIRST ROW
        JBE     SET4
        MOV     AX,UPPER_LEFT
        MOV     DX,0
        MOV     DL,LOBYTE
        SUB     DL,LAST_ROW_OFFSET
        ADD     AX,DX

SET4:   MOV     HIBYTE,AH
        MOV     LOBYTE,AL
        CALL    SET_HI_ADDR
        CALL    SET_LO_ADDR
        POP     AX

        RET

CODE    ENDS
        END
;
;END CURSOR.ASM

;PROGRAM MAIN.ASM
;THIS PROGRAM WAS MADE TO SIMULATE THE COMMUNICATION BETWEEN IBM 3278
;AND MICROCOMPUTER THROUGH A 3270 COAXIAL TYPE A INTERFACE PROTOCOL.
;
        NAME    MAIN
        PUBLIC  HIBYTE,LOBYTE,REPLY,SHIFT_FLAG,FUNC_KEY,TABLE
        EXTRN   CUR_LT:NEAR,CUR_RT:NEAR,CUR_UP:NEAR,CUR_DN:NEAR
        EXTRN   DSP_CHAR:NEAR,DISPLAY:NEAR,CHECK_RX:NEAR
        EXTRN   SET_HI_ADDR:NEAR,SET_LO_ADDR:NEAR
        EXTRN   OUT_COMMAND:NEAR,OUT_DATA:NEAR,POLL:NEAR
        EXTRN   PC_DSP:NEAR,CHK_KEY:NEAR,GET_KEY:NEAR

INCLUDE EQU.INC

CODE    SEGMENT PARA    PUBLIC  'CODE'
        ASSUME  CS:CODE, DS:CODE        ;ALL SEGMENT SET TO CODE
        ORG     100H
```

```
START:  JMP     BEGIN

;
;THIS TABLE IS USED FOR CONVERTING ASCII TO BUFFER CODE
;
TABLE:  DB      000H,000H,000H,000H,000H,000H,000H,000H         ;00-07
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;08-0F
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;10-17
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;18-1F
        DB      010H,019H,013H,02CH,01AH,02EH,030H,012H         ;20-27
        DB      00DH,00CH,0BFH,035H,033H,031H,032H,014H         ;28-2F
        DB      020H,021H,022H,023H,024H,025H,026H,027H         ;30-37
        DB      028H,029H,034H,0BEH,009H,011H,008H,018H         ;38-3F
        DB      02DH,0A0H,0A1H,0A2H,0A3H,0A4H,0A5H,0A6H         ;40-47
        DB      0A7H,0A8H,0A9H,0AAH,0ABH,0ACH,0ADH,0AEH         ;48-4F
        DB      0AFH,0B0H,0B1H,0B2H,0B3H,0B4H,0B5H,0B6H         ;50-57
        DB      0B7H,0B8H,0B9H,000H,015H,000H,000H,02FH         ;58-5F
        DB      03DH,080H,081H,082H,083H,084H,085H,086H         ;60-67
        DB      087H,088H,089H,08AH,08BH,08CH,08DH,08EH         ;68-6F
        DB      08FH,090H,091H,092H,093H,094H,095H,096H         ;70-77
        DB      097H,098H,099H,00FH,017H,00EH,03BH,000H         ;78-7F

;
;THIS TABLE IS USED FOR CONVERTING SCAN CODE TO BUFFER CODE (LOWER CASE)
;
TABLE1: DB      000H,000H,000H,000H,000H,000H,000H,000H         ;00-07
        DB      000H,009H,000H,000H,000H,000H,000H,00FH         ;08-0F
        DB      010H,011H,012H,000H,014H,015H,000H,000H         ;10-17
        DB      000H,000H,000H,01BH,000H,000H,000H,000H         ;18-1F
        DB      020H,021H,022H,023H,024H,025H,026H,027H         ;20-27
        DB      028H,029H,000H,000H,000H,000H,000H,000H         ;28-2F
        DB      031H,000H,032H,033H,000H,000H,000H,000H         ;30-37
        DB      000H,000H,000H,000H,000H,03DH,000H,000H         ;38-3F
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;40-47
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;48-4F
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;50-57
        DB      000H,000H,000H,000H,000H,000H,09EH,09FH         ;58-5F
        DB      080H,081H,082H,083H,084H,085H,086H,087H         ;60-67
        DB      088H,089H,08AH,08BH,08CH,08DH,08EH,08FH         ;68-6F
        DB      090H,091H,092H,093H,094H,095H,096H,097H         ;70-77
        DB      098H,099H,000H,000H,000H,000H,0BEH,000H         ;78-7F

;
;THIS TABLE IS USED FOR CONVERTING SCAN CODE TO BUFFER CODE (UPPER CASE)
;
TABLE2: DB      000H,000H,000H,000H,000H,000H,000H,000H         ;00-07
        DB      000H,008H,000H,000H,000H,000H,000H,00EH         ;08-0F
        DB      010H,035H,013H,000H,018H,017H,000H,000H         ;10-17
        DB      000H,000H,000H,019H,000H,000H,000H,000H         ;18-1F
        DB      00CH,016H,02DH,02CH,01AH,02EH,036H,030H         ;20-27
        DB      0BFH,00DH,000H,000H,000H,000H,000H,000H         ;28-2F
        DB      02FH,000H,032H,033H,000H,000H,000H,000H         ;30-37
        DB      000H,000H,000H,000H,000H,03BH,000H,000H         ;38-3F
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;40-47
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;48-4F
```

```
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;50-57
        DB      000H,000H,000H,000H,000H,000H,09EH,09FH         ;58-5F
        DB      0A0H,0A1H,0A2H,0A3H,0A4H,0A5H,0A6H,0A7H         ;60-67
        DB      0A8H,0A9H,0AAH,0ABH,0ACH,0ADH,0AEH,0AFH         ;68-6F
        DB      0B0H,0B1H,0B2H,0B3H,0B4H,0B5H,0B6H,0B7H         ;70-77
        DB      0B8H,0B9H,000H,000H,000H,000H,034H,000H         ;78-7F

;
;THIS TABLE IS USED FOR CONVERTING SCAN CODE TO ASCII CODE (LOWER CASE)
;
TABLE3: DB      000H,000H,000H,000H,000H,000H,000H,000H         ;00-07
        DB      000H,03CH,000H,000H,000H,000H,000H,07BH         ;08-0F
        DB      020H,03DH,027H,000H,02FH,05CH,000H,000H         ;10-17
        DB      000H,000H,000H,020H,000H,000H,000H,000H         ;18-1F
        DB      030H,031H,032H,033H,034H,035H,036H,037H         ;20-27
        DB      038H,039H,000H,000H,000H,000H,000H,000H         ;28-2F
        DB      02DH,000H,02EH,02CH,000H,000H,000H,000H         ;30-37
        DB      000H,000H,000H,000H,000H,060H,000H,000H         ;38-3F
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;40-47
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;48-4F
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;50-57
        DB      000H,000H,000H,000H,000H,000H,020H,020H         ;58-5F
        DB      061H,062H,063H,064H,065H,066H,067H,068H         ;60-67
        DB      069H,06AH,06BH,06CH,06DH,06EH,06FH,070H         ;68-6F
        DB      071H,072H,073H,074H,075H,076H,077H,078H         ;70-77
        DB      079H,07AH,000H,000H,000H,000H,03BH,000H         ;78-7F

;
;THIS TABLE IS USED FOR CONVERTING SCAN CODE TO ASCII CODE (UPPER CASE)
;
TABLE4: DB      000H,000H,000H,000H,000H,000H,000H,000H         ;00-07
        DB      000H,03EH,000H,000H,000H,000H,000H,07DH         ;08-0F
        DB      020H,02BH,022H,000H,03FH,07CH,000H,000H         ;10-17
        DB      000H,000H,000H,021H,000H,000H,000H,000H         ;18-1F
        DB      029H,020H,040H,023H,024H,025H,020H,026H         ;20-27
        DB      02AH,028H,000H,000H,000H,000H,000H,000H         ;28-2F
        DB      05FH,000H,02EH,02CH,000H,000H,000H,000H         ;30-37
        DB      000H,000H,000H,000H,000H,07EH,000H,000H         ;38-3F
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;40-47
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;48-4F
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;50-57
        DB      000H,000H,000H,000H,000H,000H,020H,020H         ;58-5F
        DB      041H,042H,043H,044H,045H,046H,047H,048H         ;60-67
        DB      049H,04AH,04BH,04CH,04DH,04EH,04FH,050H         ;68-6F
        DB      051H,052H,053H,054H,055H,056H,057H,058H         ;70-77
        DB      059H,05AH,000H,000H,000H,000H,03AH,000H         ;78-7F

;
;THIS TABLE IS USED FOR CONVERTING BUFFER TO ASCII CODE
;
TABLE6: DB      041H,000H,000H,000H,000H,000H,000H,000H         ;00-07
        DB      03EH,03CH,000H,000H,029H,028H,07DH,07BH         ;08-0F
        DB      020H,03DH,027H,022H,02FH,05CH,020H,07CH         ;10-17
        DB      03FH,021H,024H,020H,000H,000H,000H,000H         ;18-1F
        DB      030H,031H,032H,033H,034H,035H,036H,037H         ;20-27
```

```
        DB      038H,039H,000H,000H,023H,040H,025H,05FH         ;28-2F
        DB      026H,02DH,02EH,02CH,03AH,02BH,020H,000H         ;30-37
        DB      000H,000H,000H,07EH,000H,060H,000H,000H         ;38-3F
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;40-47
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;48-4F
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;50-57
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;58-5F
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;60-67
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;68-6F
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;70-77
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;78-7F
        DB      061H,062H,063H,064H,065H,066H,067H,068H         ;80-87
        DB      069H,06AH,06BH,06CH,06DH,06EH,06FH,070H         ;88-8F
        DB      071H,072H,073H,074H,075H,076H,077H,078H         ;90-97
        DB      079H,07AH,000H,000H,000H,000H,020H,020H         ;98-9F
        DB      041H,042H,043H,044H,045H,046H,047H,048H         ;A0-A7
        DB      049H,04AH,04BH,04CH,04DH,04EH,04FH,050H         ;A8-AF
        DB      051H,052H,053H,054H,055H,056H,057H,058H         ;B0-B7
        DB      059H,05AH,000H,000H,000H,000H,03BH,02AH         ;B8-BF
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;C0-C7
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;C8-CF
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;D0-D7
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;D8-DF
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;E0-E7
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;E8-EF
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;F0-F7
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;F8-FF
;
;MESSAGE
;
MSG0:   DB      'Chulalongkorn University '
MSG1:   DB      '3270 Coaxial Type A Protocol Testing '
MSG2:   DB      'Thesis Adviser : Assoc. Prof. Somchai Thayanyong '
MSG3:   DB      'By : Somchai Asavakul '

;
;VARIABLE DECLARATION
;
HIBYTE  DB      0       ;HIGH BYTE BUFFER ADDRESS COUNTER
LOBYTE  DB      0       ;LOW BYTE BUFFER ADDRESS COUNTER
REPLY   DB      1       ;DATA RECEIVE FROM RECEIVE, 0 = NO, OTHER = YES
SHIFT_FLAG      DB      0       ;SHIFT FLAG, 0 = NORMAL, 1 = SHIFT
FUNC_KEY        DB      0       ;0 = NORMAL KEY, 1 = FUNCTION KEY
;
;MAIN PROGRAM
;
BEGIN:
        CALL    POLL                    ;POLL 3278 TO REPONSE WITH POR
        CMP     AL,TT_AR
        JZ      BEGIN
        MOV     AH,POLL_ACK             ;POLL ACKNOWLEDGE TO POR
        CALL    OUT_COMMAND
        CALL    CHECK_RX                ;DISREGARD RESPONSE FROM 3278

        MOV     AH,LOAD_MASK            ;SET MASK REGISTER
```

```
CALL    OUT_COMMAND
CALL    CHECK_RX                ;DISREGARD RESPONSE FROM 3278
MOV     AH,MASK_8BIT            ;LOAD CONTROL MASK
CALL    OUT_DATA
CALL    CHECK_RX                ;DISREGARD RESPONSE FROM 3278


MOV     HIBYTE,00H              ;ROW 25
CALL    SET_HI_ADDR
MOV     LOBYTE,00H              ;COL 1
CALL    SET_LO_ADDR

MOV     AL,0FCH                 ;DISPLAY 4A ON 3278 STATUS LINE
CALL    DSP_CHAR
MOV     AL,0FDH
CALL    DSP_CHAR
MOV     AL,0DFH
CALL    DSP_CHAR

MOV     HIBYTE,01H              ;ROW 4
CALL    SET_HI_ADDR
MOV     LOBYTE,56H              ;COL 22
CALL    SET_LO_ADDR

MOV     AL,0F8H
CALL    DSP_CHAR

MOV     BX,OFFSET MSG0          ;DISPLAY MESSAGE 0
MOV     CX,25
CALL    DISPLAY

MOV     HIBYTE,02H              ;ROW 7
CALL    SET_HI_ADDR
MOV     LOBYTE,40H              ;COL 16
CALL    SET_LO_ADDR

MOV     AL,0F0H
CALL    DSP_CHAR

MOV     BX,OFFSET MSG1          ;DISPLAY MESSAGE 1
MOV     CX,37
CALL    DISPLAY

MOV     HIBYTE,03H              ;ROW 10
CALL    SET_HI_ADDR
MOV     LOBYTE,2AH              ;COL 10
CALL    SET_LO_ADDR


MOV     AL,0F8H
CALL    DSP_CHAR

MOV     BX,OFFSET MSG2          ;DISPLAY MESSAGE 2
MOV     CX,48
CALL    DISPLAY
```

```
        MOV     HIBYTE,04H              ;ROW 13
        CALL    SET_HI_ADDR
        MOV     LOBYTE,26H              ;COL 22
        CALL    SET_LO_ADDR


        MOV     AL,0F0H
        CALL    DSP_CHAR

        MOV     BX,OFFSET MSG3          ;DISPLAY MESSAGE 3
        MOV     CX,22
        CALL    DISPLAY

POLL1:                                  ;POLL 3278 AND MICROCOMPUTER KEYBOARD

        CALL    CHK_KEY                 ;PROCESS KEY IF KEYBOARD WAS PRESSED
        JZ      NEXT1
        JMP     KEY
NEXT1:
        CALL    POLL                    ;POLL IF 3278 HAS SCAN CODE
        CMP     AL,TT_AR
        JZ      POLL1

        PUSH    AX                      ;SAVE SCAN CODE FROM 3278

        MOV     AH,POLL_ACK                     ;POLL ACKNOWLEDGE
        CALL    OUT_COMMAND
        CALL    CHECK_RX                ;DISREGARD RESPONSE FROM 3278

        POP     AX                      ;RESTORE SCAN CODE FROM 3278

        MOV     FUNC_KEY,TRUE
        CALL    PROC_3278_KEY
        CMP     FUNC_KEY,TRUE
        JE      POLL1


        PUSH    AX                      ;SAVE 3278 SCAN CODE

        CMP     BYTE PTR SHIFT_FLAG,FALSE  ;TRANSLATE SCAN CODE TO BUFFER CODE
        JNE     SHIFT
        MOV     BX,OFFSET TABLE1
        JMP     CON
SHIFT:  MOV     BX,OFFSET TABLE2
CON:
        XLAT
        CALL    DSP_CHAR                ;DISPLAY CHARACTER ON 3278

        POP     AX                      ;RESTORE 3278 SCAN CODE

        CMP     BYTE PTR SHIFT_FLAG,FALSE  ;TRANSLATE SCAN CODE TO ASCII CODE
        JNE     SHIFT1
        MOV     BX,OFFSET TABLE3
        JMP     CONN
SHIFT1: MOV     BX,OFFSET TABLE4
```

```
CONN:
        XLAT
        CALL    PC_DSP                  ;DISPLAY CHARACTER ON PC.

        JMP     POLL1

KEY:    CALL    GET_KEY                 ;GET PC KEYBOARD AND DISPLAY IT
        CALL    PC_DSP
        CMP     AL,'Q'                  ;QUIT IF KEY CHARACTER IS 'Q'
        JNE     KEYIN
        JMP     QUIT
KEYIN:                                  ;PROCESS KEYBOARD
        CALL    PROC_KEY
        JMP     POLL1

;
;SUBROUTINE PROC_KEY :
;THIS SUBROUTINE PEOCESS THE PC KEY SET FOR TESTING 3270 PROTOCOL
;
PROC_KEY:
        CMP     AL,'R'                  ;RESET 3278
        JNE     KEY1
        MOV     AH,RESET
        CALL    OUT_COMMAND
        CALL    CHECK_RX                ;DISREGARD RESPONSE FROM 3278
        RET

KEY1:
        CMP     AL,'D'                  ;READ 3278 DATA
        JNE     KEY2
        MOV     AH,READ_DATA
        CALL    OUT_COMMAND

L1:     CALL    CHECK_RX                ;DISPLAY DATA READ
        CMP     BYTE PTR REPLY,0
        JZ      L1
        MOV     BX,OFFSET TABLE6
        XLAT
        CALL    PC_DSP                  ;DISPLAY CHARACTER ON PC.
        RET

KEY2:
        CMP     AL,'T'                  ;READ 3278 TERMINAL ID
        JNE     KEY3
        MOV     AH,READ_TERM_ID
        CALL    OUT_COMMAND
        CALL    CHECK_RX                ;DISREGARD RESPONSE FROM 3278
        CALL    DSPHEX                  ;DISPLAY CHARACTER IN HEX ON PC.
        RET

KEY3:
        CMP     AL,'S'                  ;READ STATUS
        JNE     KEY4
        MOV     AH,READ_STATUS
        CALL    OUT_COMMAND
```

```
        CALL    CHECK_RX                ;DISREGARD RESPONSE FROM 3278
        CALL    DSPHEX                  ;DISPLAY CHARACTER IN HEX ON PC.
        RET

KEY4:
        CMP     AL,'I'                  ;INSERT CHARACTER '1' TO 3278
        JNE     KEY5
        MOV     AH,INSERT
        CALL    OUT_COMMAND
        CALL    CHECK_RX                ;DISREGARD RESPONSE FROM 3278
        MOV     AH,BUFFER_CH_1
        CALL    OUT_DATA
L2:     CALL    CHECK_RX                ;WAIT UNTIL OPERATION COMPLETE
        CMP     BYTE PTR REPLY,0
        JZ      L2
        RET

KEY5:
        CMP     AL,'X'                  ;INHIBIT 3278 DISPLAY
        JNE     KEY6
        MOV     AH,LOAD_CTRL_REG        ;LOAD CONTROL REGISTER
        CALL    OUT_COMMAND
        CALL    CHECK_RX                ;DISREGARD RESPONSE FROM 3278
        MOV     AH,INHIBIT_DSP_MASK     ;LOAD CONTROL MASK
        CALL    OUT_DATA
L4:     CALL    CHECK_RX                ;WAIT UNTIL OPERATION COMPLETE
        CMP     BYTE PTR REPLY,0
        JZ      L4
        RET

KEY6:
        CMP     AL,'Z'                  ;SET NORMAL 3278 DISPLAY
        JNE     KEY7
        MOV     AH,LOAD_CTRL_REG        ;LOAD CONTROL REGISTER
        CALL    OUT_COMMAND
        CALL    CHECK_RX                ;DISREGARD RESPONSE FROM 3278
        MOV     AH,NORMAL_DSP_MASK      ;LOAD CONTROL MASK
        CALL    OUT_DATA
L5:     CALL    CHECK_RX                ;WAIT UNTIL OPERATION COMPLETE
        CMP     BYTE PTR REPLY,0
        JZ      L5
        RET

KEY7:
        CMP     AL,'C'                  ;INHIBIT CURSOR ON 3278 DISPLAY
        JNE     KEY8
        MOV     AH,LOAD_CTRL_REG        ;LOAD CONTROL REGISTER
        CALL    OUT_COMMAND
        CALL    CHECK_RX                ;DISREGARD RESPONSE FROM 3278
        MOV     AH,INHIBIT_CUR_MASK     ;LOAD CONTROL MASK
        CALL    OUT_DATA
L6:     CALL    CHECK_RX                ;WAIT UNTIL OPERATION COMPLETE
        CMP     BYTE PTR REPLY,0
        JZ      L6
        RET
```

```
KEY8:
        CMP     AL,'F'                  ;SEARCH FORWARD ON 3278 DISPLAY
        JNE     KEY9
        MOV     AH,FORWARD_SEARCH       ;SET FORWARD COMMAND
        CALL    OUT_COMMAND
        CALL    CHECK_RX                ;DISREGARD RESPONSE FROM 3278
        MOV     AH,BUFFER_CH_A          ;SET COMPARE CHARACTER
        CALL    OUT_DATA
L7:     CALL    CHECK_RX                ;WAIT UNTIL OPERATION COMPLETE
        CMP     BYTE PTR REPLY,0
        JZ      L7
        RET

KEY9:
        CMP     AL,'B'                  ;SEARCH BACKWARD ON 3278 DISPLAY
        JNE     KEYA
        MOV     AH,BACKWARD_SEARCH      ;SET BACKWARD SEARCH
        CALL    OUT_COMMAND
        CALL    CHECK_RX                ;DISREGARD RESPONSE FROM 3278
        MOV     AH,BUFFER_CH_A          ;SET COMPARE CHARACTER
        CALL    OUT_DATA
L8:     CALL    CHECK_RX                ;WAIT UNTIL OPERATION COMPLETE
        CMP     BYTE PTR REPLY,0
        JZ      L8
        RET

KEYA:
        CMP     AL,'M'                  ;CLEAR 3278 DISPLAY TO CHAR
        JNE     KEYB
        MOV     AH,CLEAR                ;SET CLEAR COMMAND
        CALL    OUT_COMMAND
        CALL    CHECK_RX                ;DISREGARD RESPONSE FROM 3278
        MOV     AH,BUFFER_CH_A          ;SET COMPARE CHARACTER
        CALL    OUT_DATA
L9:     CALL    CHECK_RX                ;WAIT UNTIL OPERATION COMPLETE
        CMP     BYTE PTR REPLY,0
        JZ      L9
        RET

KEYB:
        MOV     BX,OFFSET TABLE         ;TRANSLATE ASCII TO BUFFER CODE
        XLAT
        CALL    DSP_CHAR                ;DISPLAY CHARACTER ON 3278
        RET

;
;SUBROUTINE PROC_3278_KEY:
;THIS SUBROUTINE PEOCESS THE 3278 FUNCTION KEY, "FUNC_KEY" IS CLEAR IF KEY
;PRESS IS NOT A FUNCTION KEY
;
PROC_3278_KEY:

        CMP     AL,SHF_PRESS            ;SET SHIFT IF IT IS A SHIFT CODE
        JNE     CON1
        MOV     SHIFT_FLAG,TRUE
```

```
        RET

CON1:   CMP     AL,CAP_PRESS            ;TOGGLE SHIFT IF IT IS A CAPLOCK CODE
        JNE     CON2
        CMP     SHIFT_FLAG,FALSE
        JNE     TOGGLE
        MOV     SHIFT_FLAG,TRUE
        RET
TOGGLE: MOV     SHIFT_FLAG,FALSE
        RET

CON2:   CMP     AL,SHF_RELEASE          ;RELEASE SHIFT, SHIFT KEY WAS RELEASED
        JNE     CON3
        MOV     SHIFT_FLAG,FALSE
        RET

CON3:   CMP     AL,CAP_RELEASE          ;IGNORE WHEN CAPLOCK KEY WAS RELEASED
        JNE     CON4
        RET

CON4:   CMP     AL,LEFT_ARROW
        JNE     CON5
        CALL    CUR_LT
        RET

CON5:   CMP     AL,RIGHT_ARROW
        JNE     CON6
        CALL    CUR_RT
        RET

CON6:   CMP     AL,UP_ARROW
        JNE     CON7
        CALL    CUR_UP
        RET

CON7:   CMP     AL,DOWN_ARROW
        JNE     CON8
        CALL    CUR_DN
        RET

CON8:   MOV     FUNC_KEY,FALSE          ;NOT A FUNCTION KEY
        RET

;
;SUBROUTINE DSPHEX :
;THIS SUBROUTINE DISPLAY CONTENTS OF REG AL, ON PC IN HEX FORMAT
;
DSPHEX: PUSH    AX              ;CONVERT HI NIBBLE TO HEX CHAR
        MOV     CL,4
        RCR     AL,CL
        AND     AL,0FH
        ADD     AL,30H
        CMP     AL,39H
        JBE     CH2
        ADD     AL,07H
```

```
CH2:    CALL    PC_DSP
        POP     AX

        AND     AL,0FH          ;CONVERT LO NIBBLE TO HEX CHAR
        ADD     AL,30H
        CMP     AL,39H
        JBE     CH1
        ADD     AL,07H
CH1:    CALL    PC_DSP
        RET

;
QUIT:   INT     20H     ;TERMINATE PROGRAM


CODE    ENDS
        END     START

;END PROGRAM MAIN.ASM
```

Listing of Program using 3270CIB to Interface PC

to IBM 3274 and emulate PC as IBM 3278

```
;FILE NAME 3278IO1.ASM
;THIS MODULE CONTAINS SUBROUTINES WHICH HANDLE COMMAND INTERFACE
;OF THE IBM3278

        PUBLIC  DSP_CHAR,DISPLAY
        PUBLIC  RD_HI_ADDR,RD_LO_ADDR
        PUBLIC  SET_HI_ADDR,SET_LO_ADDR
        PUBLIC  OUT_COMMAND,OUT_DATA,CHECK_RX,POLL
        EXTRN   HIBYTE:BYTE,LOBYTE:BYTE,REPLY:BYTE,TABLE:BYTE

INCLUDE EQU.INC

CODE    SEGMENT PARA    PUBLIC  'CODE'
        ASSUME  CS:CODE, DS:CODE
;
;SUBROUTINE DSP_CHAR:
;THIS SUBROUTINE WRITE DATA IN REG AL. IT USES THE WRITE DATA COMMAND.
;
DSP_CHAR:
        PUSH    AX
        MOV     AH,WRITE_DATA           ;SET COMMAND
        CALL    OUT_COMMAND
        PUSH    AX
        CALL    CHECK_RX                ;DISREGARD RESPONSE FROM 3278
        POP     AX
        MOV     AH,AL
        CALL    OUT_DATA
        CALL    CHECK_RX                ;DISREGARD RESPONSE FROM 3278
        POP     AX
        RET

;
;SUBROUTINE DISPLAY:
;THIS SUBROUTINE WRITE STRINGS OF DATA POINT TO BY REGISTER BX, WHILE
;REGISTER CX CONTAINS THE NUMBER OF BYTES TO BE OUTPUTED. IT USES THE
;WRITE DATA COMMAND. THIS SUBROUTINE ALSO CONVERT ASCII TO BUFFER CODE.
;
DISPLAY:
```

```
LOOP:   MOV     AL,BYTE PTR [BX]        ;DISPLAY DATA UNTIL FINISHES

        PUSH    BX                      ;CONVERT ASCII TO BUFFER CODE
        MOV     BX,OFFSET TABLE
        XLAT
        MOV     AH,AL
        POP     BX

        CALL    OUT_DATA
        CALL    CHECK_RX                ;DISREGARD RESPONSE FROM 3278
        INC     BX                      ;GET NEXT BYTE
        LOOP    LOOP

        POP     AX

        RET
;
;SUBROUTINE RD_HI_ADDR:
;THIS SUBROUTINE USES THE READ BUFFER ADDRESS COUNTER HIGH BYTE COMMAND,
;THE HI_BYTE READ WAS PLACE IN VARIABLE 'HIBYTE'.
;
RD_HI_ADDR:
        PUSH    AX

        MOV     AH,READ_ADDR_HI         ;SET COMMAND
        CALL    OUT_COMMAND

        CALL    CHECK_RX

        MOV     HIBYTE,AL               ;PLACE DATA IN 'HIBYTE'

        POP     AX

        RET

;
;SUBROUTINE RD_LO_ADDR:
;THIS SUBROUTINE USES THE READ BUFFER ADDRESS COUNTER LOW BYTE COMMAND,
;THE LO_BYTE READ WAS PLACE IN VARIABLE 'LOBYTE'.
;
RD_LO_ADDR:
        PUSH    AX

        MOV     AH,READ_ADDR_LO         ;SET COMMAND
        CALL    OUT_COMMAND

        CALL    CHECK_RX

        MOV     LOBYTE,AL               ;PLACE DATA IN 'LOBYTE'

        POP     AX

        RET

;
```

```
;SUBROUTINE SET_HI_ADDR:
;THIS SUBROUTINE USES THE LOAD BUFFER ADDRESS COUNTER HIGH BYTE COMMAND,
;THE HIGH BYTE ADDRESS TO BE SET WAS TAKEN FROM VARIABLE 'HIBYTE'.
;
SET_HI_ADDR:
        PUSH    AX

        MOV     AH,LOAD_ADDR_HI         ;SET COMMAND
        CALL    OUT_COMMAND

        CALL    CHECK_RX                ;DISREGARD RESPONSE FROM 3278

        MOV     AH,HIBYTE               ;SET POSITION
        CALL    OUT_DATA

        CALL    CHECK_RX                ;DISREGARD RESPONSE FROM 3278

        POP     AX

        RET

;
;SUBROUTINE SET_LO_ADDR:
;THIS SUBROUTINE USES THE LOAD BUFFER ADDRESS COUNTER LOW BYTE COMMAND,
;THE LOW BYTE ADDRESS TO BE SET WAS TAKEN FROM VARIABLE 'LOBYTE'.
;
SET_LO_ADDR:
        PUSH    AX

        MOV     AH,LOAD_ADDR_LO         ;SET COMMAND
        CALL    OUT_COMMAND

        CALL    CHECK_RX                ;DISREGARD RESPONSE FROM 3278

        MOV     AH,LOBYTE               ;SET POSITION
        CALL    OUT_DATA

        CALL    CHECK_RX                ;DISREGARD RESPONSE FROM 3278

        POP     AX

        RET

;
;SUBROUTINE OUT_COMMAND:
;THIS SUBROUTINE OUTPUT CONTENTS IN REGISTER AH AS COMMAND WORD
;
OUT_COMMAND:
        PUSH    AX

        CALL    CHECK_TX        ;WAIT UNTIL TRANSMITTER IS EMPTY
        MOV     AL,AH

        MOV     DX,SET_b11      ;SET AS COMMAND WORD
        OUT     DX,AL
```

```
        MOV     DX,DATA_PORT    ;OUTPUT COMMAND
        OUT     DX,AL

        POP     AX

        RET

;
;SUBROUTINE OUT_DATA:
;THIS SUBROUTINE OUTPUT CONTENTS IN REGISTER AH AS DATA WORD
;
OUT_DATA:
        PUSH    AX

        CALL    CHECK_TX        ;WAIT UNTIL TRANSMITTER IS EMPTY
        MOV     AL,AH

        MOV     DX,CLEAR_b11    ;SET AS DATA WORD
        OUT     DX,AL

        MOV     DX,DATA_PORT    ;OUTPUT DATA
        OUT     DX,AL

        POP     AX

        RET

;
;SUBROUTINE CHECK_TX:
;THIS SUBROUTINE LOOP UNTIL TRANSMITTER IS EMPTY
;
CHECK_TX:
        PUSH    AX

        MOV     DX,STATUS_PORT  ;WAIT UNTIL TRANSMITTER IS EMPTY
FULL:   IN      AL,DX
        AND     AL,MASK_TX
        JNZ     FULL

        POP     AX

        RET

;
;SUBROUTINE CHECK_RX:
;THIS SUBROUTINE LOOP UNTIL RECEIVER HAS DATA, DATA IS READ INTO REG AL.
;REG AH = 0 IF NO RESPONSE WAS RECEIVED.
;
CHECK_RX:
        MOV     DX,STATUS_PORT  ;WAIT UNTIL RECEIVER HAS DATA
        IN      AL,DX
        AND     AL,MASK_RX
        JZ      NODAT
SKIP:   MOV     DX,DATA_PORT
        IN      AL,DX
```

```
        MOV     AH,AL
        MOV     DX,STATUS_PORT
        IN      AL,DX
        MOV     REPLY,1
        RET

NODAT:
        MOV     REPLY,0
        RET

;
;SUBROUTINE POLL:
;THIS SUBROUTINE OUTPUT POLL COMMAND TO 3278 UNTIL RESPONSE WAS RECEIVE,
;THE RECEIVE DATA WAS PUT IN REG AL.
;
POLL:
        MOV     AH,POLL_CHAR            ;POLL 3278 (POLL COMMAND = 01H)
        CALL    OUT_COMMAND
        CALL    CHECK_RX                ;WAIT UNTIL RESPONSE IS POR (02H)
        CMP     BYTE PTR REPLY,0
        JZ      POLL

        RET

CODE    ENDS
        END
;
;END 3278IO1.ASM


;PROGRAM MAIN1.ASM
;THIS PROGRAM WAS MADE TO SIMULATE THE COMMUNICATION BETWEEN IBM 3274
;AND MICROCOMPUTER THROUGH A 3270 COAXIAL TYPE A INTERFACE PROTOCOL.
;
        NAME    MAIN
        PUBLIC  HIBYTE,LOBYTE,REPLY,SHIFT_FLAG,FUNC_KEY,TABLE
        EXTRN   CUR_LT:NEAR,CUR_RT:NEAR,CUR_UP:NEAR,CUR_DN:NEAR
        EXTRN   DSP_CHAR:NEAR,DISPLAY:NEAR,CHECK_RX:NEAR
        EXTRN   SET_HI_ADDR:NEAR,SET_LO_ADDR:NEAR
        EXTRN   OUT_COMMAND:NEAR,POLL:NEAR,OUT_DATA:NEAR
        EXTRN   PC_DSP:NEAR,CHK_KEY:NEAR,GET_KEY:NEAR

INCLUDE EQU.INC

CODE    SEGMENT PARA    PUBLIC  'CODE'
        ASSUME  CS:CODE, DS:CODE        ;ALL SEGMENT SET TO CODE
        ORG     100H
START:  JMP     BEGIN1

;
;THIS TABLE IS USED FOR CONVERTING ASCII TO BUFFER CODE
;
TABLE:  DB      000H,000H,000H,000H,000H,000H,000H,000H         ;00-07
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;08-0F
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;10-17
```

```
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;18-1F
        DB      010H,019H,013H,02CH,01AH,02EH,030H,012H         ;20-27
        DB      00DH,00CH,0BFH,035H,033H,031H,032H,014H         ;28-2F
        DB      020H,021H,022H,023H,024H,025H,026H,027H         ;30-37
        DB      028H,029H,034H,0BEH,009H,011H,008H,018H         ;38-3F
        DB      02DH,0A0H,0A1H,0A2H,0A3H,0A4H,0A5H,0A6H         ;40-47
        DB      0A7H,0A8H,0A9H,0AAH,0ABH,0ACH,0ADH,0AEH         ;48-4F
        DB      0AFH,0B0H,0B1H,0B2H,0B3H,0B4H,0B5H,0B6H         ;50-57
        DB      0B7H,0B8H,0B9H,000H,015H,000H,000H,02FH         ;58-5F
        DB      03DH,080H,081H,082H,083H,084H,085H,086H         ;60-67
        DB      087H,088H,089H,08AH,08BH,08CH,08DH,08EH         ;68-6F
        DB      08FH,090H,091H,092H,093H,094H,095H,096H         ;70-77
        DB      097H,098H,099H,00FH,017H,00EH,03BH,000H         ;78-7F

;
;THIS TABLE IS USED FOR CONVERTING SCAN CODE TO BUFFER CODE (LOWER CASE)
;
TABLE1: DB      000H,000H,000H,000H,000H,000H,000H,000H         ;00-07
        DB      000H,009H,000H,000H,000H,000H,000H,00FH         ;08-0F
        DB      010H,011H,012H,000H,014H,015H,000H,000H         ;10-17
        DB      000H,000H,000H,01BH,000H,000H,000H,000H         ;18-1F
        DB      020H,021H,022H,023H,024H,025H,026H,027H         ;20-27
        DB      028H,029H,000H,000H,000H,000H,000H,000H         ;28-2F
        DB      031H,000H,032H,033H,000H,000H,000H,000H         ;30-37
        DB      000H,000H,000H,000H,000H,03DH,000H,000H         ;38-3F
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;40-47
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;48-4F
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;50-57
        DB      000H,000H,000H,000H,000H,000H,09EH,09FH         ;58-5F
        DB      080H,081H,082H,083H,084H,085H,086H,087H         ;60-67
        DB      088H,089H,08AH,08BH,08CH,08DH,08EH,08FH         ;68-6F
        DB      090H,091H,092H,093H,094H,095H,096H,097H         ;70-77
        DB      098H,099H,000H,000H,000H,000H,0BEH,000H         ;78-7F

;
;THIS TABLE IS USED FOR CONVERTING SCAN CODE TO BUFFER CODE (UPPER CASE)
;
TABLE2: DB      000H,000H,000H,000H,000H,000H,000H,000H         ;00-07
        DB      000H,008H,000H,000H,000H,000H,000H,00EH         ;08-0F
        DB      010H,035H,013H,000H,018H,017H,000H,000H         ;10-17
        DB      000H,000H,000H,019H,000H,000H,000H,000H         ;18-1F
        DB      00CH,016H,02DH,02CH,01AH,02EH,036H,030H         ;20-27
        DB      0BFH,00DH,000H,000H,000H,000H,000H,000H         ;28-2F
        DB      02FH,000H,032H,033H,000H,000H,000H,000H         ;30-37
        DB      000H,000H,000H,000H,000H,03BH,000H,000H         ;38-3F
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;40-47
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;48-4F
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;50-57
        DB      000H,000H,000H,000H,000H,000H,09EH,09FH         ;58-5F
        DB      0A0H,0A1H,0A2H,0A3H,0A4H,0A5H,0A6H,0A7H         ;60-67
        DB      0A8H,0A9H,0AAH,0ABH,0ACH,0ADH,0AEH,0AFH         ;68-6F
        DB      0B0H,0B1H,0B2H,0B3H,0B4H,0B5H,0B6H,0B7H         ;70-77
        DB      0B8H,0B9H,000H,000H,000H,000H,034H,000H         ;78-7F

;
```


```
;THIS TABLE IS USED FOR CONVERTING SCAN CODE TO ASCII CODE (LOWER CASE)
;
TABLE3: DB      000H,000H,000H,000H,000H,000H,000H,000H          ;00-07
        DB      000H,03CH,000H,000H,000H,000H,000H,07BH          ;08-0F
        DB      020H,03DH,027H,000H,02FH,05CH,000H,000H          ;10-17
        DB      000H,000H,000H,020H,000H,000H,000H,000H          ;18-1F
        DB      030H,031H,032H,033H,034H,035H,036H,037H          ;20-27
        DB      038H,039H,000H,000H,000H,000H,000H,000H          ;28-2F
        DB      02DH,000H,02EH,02CH,000H,000H,000H,000H          ;30-37
        DB      000H,000H,000H,000H,000H,060H,000H,000H          ;38-3F
        DB      000H,000H,000H,000H,000H,000H,000H,000H          ;40-47
        DB      000H,000H,000H,000H,000H,000H,000H,000H          ;48-4F
        DB      000H,000H,000H,000H,000H,000H,000H,000H          ;50-57
        DB      000H,000H,000H,000H,000H,000H,020H,020H          ;58-5F
        DB      061H,062H,063H,064H,065H,066H,067H,068H          ;60-67
        DB      069H,06AH,06BH,06CH,06DH,06EH,06FH,070H          ;68-6F
        DB      071H,072H,073H,074H,075H,076H,077H,078H          ;70-77
        DB      079H,07AH,000H,000H,000H,000H,03BH,000H          ;78-7F

;
;THIS TABLE IS USED FOR CONVERTING SCAN CODE TO ASCII CODE (UPPER CASE)
;
TABLE4: DB      000H,000H,000H,000H,000H,000H,000H,000H          ;00-07
        DB      000H,03EH,000H,000H,000H,000H,000H,07DH          ;08-0F
        DB      020H,02BH,022H,000H,03FH,07CH,000H,000H          ;10-17
        DB      000H,000H,000H,021H,000H,000H,000H,000H          ;18-1F
        DB      029H,020H,040H,023H,024H,025H,020H,026H          ;20-27
        DB      02AH,028H,000H,000H,000H,000H,000H,000H          ;28-2F
        DB      05FH,000H,02EH,02CH,000H,000H,000H,000H          ;30-37
        DB      000H,000H,000H,000H,000H,07EH,000H,000H          ;38-3F
        DB      000H,000H,000H,000H,000H,000H,000H,000H          ;40-47
        DB      000H,000H,000H,000H,000H,000H,000H,000H          ;48-4F
        DB      000H,000H,000H,000H,000H,000H,000H,000H          ;50-57
        DB      000H,000H,000H,000H,000H,000H,020H,020H          ;58-5F
        DB      041H,042H,043H,044H,045H,046H,047H,048H          ;60-67
        DB      049H,04AH,04BH,04CH,04DH,04EH,04FH,050H          ;68-6F
        DB      051H,052H,053H,054H,055H,056H,057H,058H          ;70-77
        DB      059H,05AH,000H,000H,000H,000H,03AH,000H          ;78-7F
;
;THIS TABLE IS USED FOR CONVERTING ASCII TO SCAN CODE
;
TABLE5  DB      000H,000H,000H,000H,000H,000H,000H,000H          ;00-07
        DB      000H,000H,000H,000H,000H,000H,000H,000H          ;08-0F
        DB      000H,000H,000H,000H,000H,000H,000H,000H          ;10-17
        DB      000H,000H,000H,000H,000H,000H,000H,000H          ;18-1F
        DB      010H,01BH,012H,023H,024H,025H,027H,012H          ;20-27
        DB      029H,020H,028H,011H,033H,030H,032H,014H          ;28-2F
        DB      020H,021H,022H,023H,024H,025H,026H,027H          ;30-37
        DB      028H,029H,07EH,07EH,009H,011H,009H,014H          ;38-3F
        DB      022H,060H,061H,062H,063H,064H,065H,066H          ;40-47
        DB      067H,068H,069H,06AH,06BH,06CH,06DH,06EH          ;48-4F
        DB      06FH,070H,071H,072H,073H,074H,075H,076H          ;50-57
        DB      077H,078H,079H,000H,015H,000H,000H,030H          ;58-5F
        DB      03DH,060H,061H,062H,063H,064H,065H,066H          ;60-67
        DB      067H,068H,069H,06AH,06BH,06CH,06DH,06EH          ;68-6F
```

```
        DB      06FH,070H,071H,072H,073H,074H,075H,076H         ;70-77
        DB      077H,078H,079H,00FH,015H,00FH,03DH,000H         ;78-7F

;
;THIS TABLE IS USED FOR CONVERTING BUFFER TO ASCII CODE
;
TABLE6: DB      041H,000H,000H,000H,000H,000H,000H,000H         ;00-07
        DB      03EH,03CH,000H,000H,029H,028H,07DH,07BH         ;08-0F
        DB      020H,03DH,027H,022H,02FH,05CH,020H,07CH         ;10-17
        DB      03FH,021H,024H,020H,000H,000H,000H,000H         ;18-1F
        DB      030H,031H,032H,033H,034H,035H,036H,037H         ;20-27
        DB      038H,039H,000H,000H,023H,040H,025H,05FH         ;28-2F
        DB      026H,02DH,02EH,02CH,03AH,02BH,020H,000H         ;30-37
        DB      000H,000H,000H,07EH,000H,060H,000H,000H         ;38-3F
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;40-47
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;48-4F
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;50-57
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;58-5F
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;60-67
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;68-6F
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;70-77
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;78-7F
        DB      061H,062H,063H,064H,065H,066H,067H,068H         ;80-87
        DB      069H,06AH,06BH,06CH,06DH,06EH,06FH,070H         ;88-8F
        DB      071H,072H,073H,074H,075H,076H,077H,078H         ;90-97
        DB      079H,07AH,000H,000H,000H,000H,020H,020H         ;98-9F
        DB      041H,042H,043H,044H,045H,046H,047H,048H         ;A0-A7
        DB      049H,04AH,04BH,04CH,04DH,04EH,04FH,050H         ;A8-AF
        DB      051H,052H,053H,054H,055H,056H,057H,058H         ;B0-B7
        DB      059H,05AH,000H,000H,000H,000H,03BH,02AH         ;B8-BF
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;C0-C7
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;C8-CF
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;D0-D7
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;D8-DF
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;E0-E7
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;E8-EF
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;F0-F7
        DB      000H,000H,000H,000H,000H,000H,000H,000H         ;F8-FF

;
;VARIABLE DECLARATION
;
HIBYTE  DB      0       ;HIGH BYTE BUFFER ADDRESS COUNTER
LOBYTE  DB      0       ;LOW BYTE BUFFER ADDRESS COUNTER
REPLY   DB      1       ;DATA RECEIVE FROM RECEIVE, 0 = NO, OTHER = YES
SHIFT_FLAG      DB      0       ;SHIFT FLAG, 0 = NORMAL, 1 = SHIFT
FUNC_KEY        DB      0       ;0 = NORMAL KEY, 1 = FUNCTION KEY
KEY_CODE DB     0
KEY_AVAIL DB    0
ADDR    DW      0
SCREEN  DB      1920 DUP(00H)
RETCOM  DB      0
;
;MAIN PROGRAM
;
```

```
BEGIN1:
        PUSH    CS
        POP     DS

BEGIN0: CALL    CHECK_RX        ;CHECK IF THERE IS A POLL FROM 3274
        CMP     REPLY,0
        JE      BEGIN0

BEGIN2: MOV     AH,02H          ;RESPONSE WITH POR
        MOV     DX,304H
        OUT     DX,AL
        CALL    OUT_DATA

BEGIN3: CALL    CHECK_RX;       ;REPLY WITH POR UNTIL 3274 RESPONSE
        CMP     REPLY,0         ;WITH POLL ACK
        JE      BEGIN3
        CMP     AH,11H
        JNE     BEGIN2

        MOV     DX,303H
        OUT     DX,AL

        MOV     DX,305H
        OUT     DX,AL

BEGIN:
        CALL    CHECK_RX        ;POLL RECEIVER
        CMP     REPLY,0
        JZ      POLL1           ;CHECK PC IF NO DATA

        AND     AL,01H          ;IS IT A COMMAND OR DATA WORD
        JNZ     CHKPOLL
        JMP     DATA1

CHKPOLL:
        CMP     AH,01H          ;IF IT A POLL CHARACTER AND A KEY IS
        JNZ     COM0            ;WAITING, THEN SEND IT
        CMP     KEY_AVAIL,0
        JZ      TTAR
        MOV     KEY_AVAIL,0
        MOV     AH,KEY_CODE
        MOV     DX,304H
        OUT     DX,AL
        CALL    OUT_DATA
        MOV     DX,305H
        OUT     DX,AL
        JMP     BEGIN

TTAR:
        CMP     RETCOM,0
        JZ      TTAR1
        MOV     AH,0
        CALL    OUT_COMMAND
        MOV     RETCOM,0
        JMP     BEGIN
```

```
TTAR1:
        MOV     DX,303H
        OUT     DX,AL
        JMP     BEGIN

POLL1:
        CALL    CHK_KEY             ;PROCESS KEY IF KEYBOARD WAS PRESSED
        JZ      BEGIN

KEY:    CALL    GET_KEY             ;GET PC KEYBOARD AND DISPLAY IT
        CMP     AL,'Q'              ;QUIT IF KEY CHARACTER IS 'Q'
        JNE     KEYIN
        JMP     QUIT
KEYIN:                              ;TRANSLATE ASCII TO SCAN CODE
        MOV     BX,OFFSET TABLE5
        XLAT
        MOV     KEY_CODE,AL
        MOV     KEY_AVAIL,1
        JMP     BEGIN

COM0:   CMP     AH,0CH
        JNZ     COM1
        JMP     COMX
COM1:
        CMP     AH,03H              ;READ DATA
        JNZ     COM2
        MOV     BX,OFFSET SCREEN
        ADD     BX,WORD PTR ADDR
        MOV     AH,[BX]
        CALL    OUT_DATA
        INC     WORD PTR ADDR
        JMP     BEGIN

COM2:   CMP     AH,05H              ;READ ADDR HIGH
        JNZ     COM3
        MOV     AX,WORD PTR ADDR
        CALL    OUT_DATA
        JMP     BEGIN

COM3:   CMP     AH,15H              ;READ ADDR LOW
        JNZ     COM4
        MOV     AX,WORD PTR ADDR
        MOV     AH,AL
        CALL    OUT_DATA
        JMP     BEGIN

COM4:   CMP     AH,09H              ;READ TERM ID
        JNZ     COM5
        MOV     AH,0A4H
        CALL    OUT_DATA
        JMP     BEGIN

COM5:   CMP     AH,0DH              ;READ STATUS
        JNZ     COM6
        MOV     AH,020H
```

```
        CALL    OUT_DATA
        JMP     BEGIN

COM6:   CMP     AH,04H                  ;LOAD ADDR HIGH
        JNZ     COM7
        MOV     DX,303H
        OUT     DX,AL
LOOP11: CALL    CHECK_RX
        CMP     REPLY,0
        JZ      LOOP11
        MOV     BX,WORD PTR ADDR
        MOV     AL,BL
        MOV     WORD PTR ADDR,AX
        MOV     DX,303H
        OUT     DX,AL
        JMP     BEGIN

COM7:   CMP     AH,14H                  ;LOAD ADDR LOW
        JNZ     COM8
        MOV     DX,303H
        OUT     DX,AL
LOOP12: CALL    CHECK_RX
        CMP     REPLY,0
        JZ      LOOP12
        MOV     BX,WORD PTR ADDR
        MOV     BL,AH
        MOV     WORD PTR ADDR,BX
        MOV     DX,303H
        OUT     DX,AL
        JMP     BEGIN

COM8:   CMP     AH,25H                  ;UNKNOW COMMAND
        JNZ     COM9
        MOV     RETCOM,1
        JMP     COMX

COM9:   CMP     AH,35H                  ;UNKNOW COMMAND
        JNZ     COM10
        MOV     RETCOM,1
        JMP     COMX

COM10:  CMP     AH,45H                  ;UNKNOW COMMAND
        JNZ     COM11
        MOV     RETCOM,1
        JMP     COMX

COM11:  CMP     AH,55H                  ;UNKNOW COMMAND
        JNZ     COM12
        MOV     RETCOM,1
        JMP     COMX

COM12:  CMP     AH,65H                  ;UNKNOW COMMAND
        JNZ     COM13
        MOV     RETCOM,1
        JMP     COMX
```

```
COM13:  CMP     AH,75H                  ;UNKNOW COMMAND
        JNZ     COM14
        MOV     RETCOM,1
        JMP     COMX

COM14:  CMP     AH,85H                  ;UNKNOW COMMAND
        JNZ     COM15
        MOV     RETCOM,1
        JMP     COMX

COM15:  CMP     AH,95H                  ;UNKNOW COMMAND
        JNZ     COM16
        MOV     RETCOM,1
        JMP     COMX

COM16:  CMP     AH,0A5H                 ;UNKNOW COMMAND
        JNZ     COM17
        MOV     RETCOM,1
        JMP     COMX

COM17:  CMP     AH,0B5H                 ;UNKNOW COMMAND
        JNZ     COM18
        MOV     RETCOM,1
        JMP     COMX

COM18:  CMP     AH,0C5H                 ;UNKNOW COMMAND
        JNZ     COM19
        MOV     RETCOM,1
        JMP     COMX

COM19:  CMP     AH,0D5H                 ;UNKNOW COMMAND
        JNZ     COM20
        MOV     RETCOM,1
        JMP     COMX

COM20:  CMP     AH,0E5H                 ;UNKNOW COMMAND
        JNZ     COMX
        MOV     RETCOM,1
        JMP     COMX

COMX:
        MOV     DX,303H
        OUT     DX,AL
        JMP     BEGIN

DATA1:
        CALL    CHECK_RX        ;POLL RECEIVER
        CMP     REPLY,0
        JZ      COMX            ;CHECK PC IF NO DATA

        AND     AL,01H          ;IS IT A COMMAND OR DATA WORD
        JZ      DATA1
        JMP     CHKPOLL
```

```
;
QUIT:   INT     20H     ;TERMINATE PROGRAM


CODE    ENDS
        END     START

;END PROGRAM MAIN1.ASM
```

Biography

Mr. Somchai Asavakul was born in Bangkok on 23rd July 1962, graduated in 1983 with the Bachelor Degree in Electronic Engineering from the Department of Electrical and Electronic Engineering, Faculty of Engineering, Western Australian Institute of Technology.