FTS Capture file Fixes

Also covers breakout capture files

30 November 2004

TABLE OF CONTENTS

Section **Page**

1 How to save data with FTS . . . 3
2 Fix1.l . . . 5
2.1 Description . . . 5
2.2 Calling Sequence . . . 5
2.3 Input Data Format . . . 5
2.4 Internal Processing . . . 7
2.5 Output . . . 7
3 Fix2.l . . . 9
3.1 Description . . . 9
3.2 Calling Sequence . . . 9
3.3 Input Data Format . . . 9
3.4 Internal Processing . . . 10
3.5 Output . . . 10
4 UnBlank.l . . . 11
4.1 Description . . . 11
4.2 Calling Sequence . . . 11
4.3 Input Data Format . . . 11
4.4 Internal Processing . . . 11
4.5 Output . . . 12
5 GcProc.l . . . 13
5.1 Description . . . 13
5.2 Calling Sequence . . . 13
5.3 Input Data Format . . . 13
5.4 Internal Processing . . . 13
5.5 Output . . . 13
APPENDIX A

[1] `$Header: d:/txb-p/Captures/RCS/DocFix.tex,v 1.16 2004-11-30 09:41:04-08 Hamilton Exp Hamilton $`

A Saving capture files with BREAKOUT A-1

APPENDIX B

B DB B-1
B.1 Description B-1
B.2 Calling Sequence B-1
B.3 Input Data Format B-1
B.4 Internal Processing B-2
B.5 Output B-3

APPENDIX C

C Fix0.l C-1
C.1 Description C-1
C.2 Calling Sequence C-1
C.3 Input Data Format C-1
C.4 Internal Processing C-1
C.5 Output C-1

APPENDIX D

D FixB.l D-1
D.1 Description D-1
D.2 Calling Sequence D-1
D.3 Input Data Format D-1
D.4 Internal Processing D-1
D.5 Output D-2

APPENDIX E

1 How to save data with FTS

Utilizing the new asynchronous data capture program from FrontLine Test Systems' named "Serialtest Async" requires that it be set up as follows to capture data from a Panasonic WV-CU161 keyboard in RS-485 four wire mode:

1. `Configuration: Monitor Both 19200,N,8,1 19200,N,8,1 DTE=COM4, DCE=COM5`
2. `Capture file:E:\Capture\test`, or where ever you want to put it.
3. `Bit Order: LSB first (normal)`
4. `Hardware Settings: Use FTS Cables`
5. `System Settings: Wrap Buffer is selected`
6. `System Settings: Capture Buffer Size (in K): 2052`
7. `System Settings: File Size (in K): 4102`
8. `Program Start Up Options: Prompt for a filename, then immediately start capturing.`
9. `Advanced System Options: <all at defaults>`

After getting every thing turned on, click on the correct Icon, fool around a bit and then select "`Live: Start Capture to Disk...`".

To output the captured data:

1. `Export Events...`
2. `Apply Template: Sample Events`
3. `Displayed Fields: Side Timestamp Hexadecimal`
4. `Text Output`
5. `Separate Records With CR/LF`
6. `Output Header`
7. `Output Field Name Record`
8. `Align Field Names With Data`
9. `Timestamp Format: Native`
10. `Character Set: ASCII`
11. `Signals Characters: 1/0`
12. `Errors Characters: X/<sp>`

13. `Field Delimiter: Space`
14. `Filter Out: Neither Side`
15. `Export: Entire Buffer`
16. `As: Serialtest Async Events.txt` or other name of your choice

Note

1. In all of the following sample outputs/inputs the actual file has been edited. This has been done to change the format of the `RCS` keywords by adding a blank ("␣") after the leading dollar sign ("`$`").
2. All comments, lines begining with a per-cent sign "%", have been deleted in the examples to save paper.
3. Source code listings are suppressed in this documentation version.

2 Fix1.l

2.1 Description

This routine is used to reformat ASCII records written by FrontLine Test Systems' serial data line capture software named "Serialtest Async".

2.2 Calling Sequence

```
fix1 infile outfile
```

2.3 Input Data Format

The capture data must be saved with the following fields in the following order:

1. **Date/Time:** "8/26/2004␣12:06:35.060␣PM"
2. **Data:** "02"

Typical first few lines of the capture file must be as follows:

```
 1  FTS capture file: E:\Capture\newtest.cfa (8/26/2004 12:34:27 PM)
 2  Event 1 (8/26/2004 12:06:06.434 PM) through
 3  Event 30,910 (8/26/2004 12:32:44.595 PM)
 4
 5  Timestamp                 Hx
 6  8/26/2004 12:06:06.434 PM
 7  8/26/2004 12:06:35.060 PM 02
 8  8/26/2004 12:06:35.060 PM 41
 9  8/26/2004 12:06:35.060 PM 44
10  8/26/2004 12:06:35.060 PM 30
11  8/26/2004 12:06:35.060 PM 31
12  8/26/2004 12:06:35.060 PM 3b
13  8/26/2004 12:06:35.060 PM 52
14  8/26/2004 12:06:35.066 PM 4f
15  8/26/2004 12:06:35.066 PM 4e
16  8/26/2004 12:06:35.066 PM 3a
17  8/26/2004 12:06:35.066 PM 30
18  8/26/2004 12:06:35.066 PM 03
19  8/26/2004 12:06:35.106 PM 02
20  8/26/2004 12:06:35.106 PM 41
21  8/26/2004 12:06:35.106 PM 44
22  8/26/2004 12:06:35.106 PM 30
23  8/26/2004 12:06:35.112 PM 31
24  8/26/2004 12:06:35.112 PM 3b
25  8/26/2004 12:06:35.112 PM 47
26  8/26/2004 12:06:35.112 PM 43
27  8/26/2004 12:06:35.112 PM 37
28  8/26/2004 12:06:35.112 PM 3a
29  8/26/2004 12:06:35.112 PM 32
```

```
30  8/26/2004 12:06:35.112 PM 30
31  8/26/2004 12:06:35.112 PM 32
32  8/26/2004 12:06:35.112 PM 30
33  8/26/2004 12:06:35.112 PM 30
34  8/26/2004 12:06:35.112 PM 30
35  8/26/2004 12:06:35.119 PM 30
36  8/26/2004 12:06:35.119 PM 03
37  8/26/2004 12:06:36.141 PM 02
38  8/26/2004 12:06:36.141 PM 41
39  8/26/2004 12:06:36.148 PM 44
40  8/26/2004 12:06:36.148 PM 30
41  8/26/2004 12:06:36.148 PM 31
42  8/26/2004 12:06:36.148 PM 3b
43  8/26/2004 12:06:36.148 PM 47
44  8/26/2004 12:06:36.148 PM 43
45  8/26/2004 12:06:36.148 PM 37
46  8/26/2004 12:06:36.148 PM 3a
47  8/26/2004 12:06:36.148 PM 32
48  8/26/2004 12:06:36.148 PM 30
49  8/26/2004 12:06:36.148 PM 32
50  8/26/2004 12:06:36.154 PM 30
51  8/26/2004 12:06:36.154 PM 30
52  8/26/2004 12:06:36.154 PM 32
53  8/26/2004 12:06:36.154 PM 30
54  8/26/2004 12:06:36.154 PM 03
55  8/26/2004 12:06:37.177 PM 02
56  8/26/2004 12:06:37.177 PM 41
57  8/26/2004 12:06:37.177 PM 44
58  8/26/2004 12:06:37.177 PM 30
59  8/26/2004 12:06:37.177 PM 31
60  8/26/2004 12:06:37.183 PM 3b
```

An alternate file format is:

```
 1  DTE 00:00.000 02
 2  DTE 00:00.000 41
 3  DTE 00:00.000 44
 4  DTE 00:00.000 30
 5  DTE 00:00.000 31
 6  DTE 00:00.000 3b
 7  DTE 00:00.000 52
 8  DTE 00:00.000 4f
 9  DTE 00:00.000 4e
10  DTE 00:00.000 3a
11  DTE 00:00.000 30
12  DTE 00:00.000 03
13  DCE 00:00.001 06
14  DCE 00:00.001 02
15  DCE 00:00.001 41
16  DCE 00:00.001 44
17  DCE 00:00.001 30
18  DCE 00:00.001 31
```

```
19  DCE 00:00.001 3b
20  DCE 00:00.001 52
21  DCE 00:00.001 4f
22  DCE 00:00.001 4e
23  DCE 00:00.001 03
24  DTE 00:00.002 02
25  DTE 00:00.002 41
26  DTE 00:00.002 44
27  DTE 00:00.002 30
28  DTE 00:00.002 31
29  DTE 00:00.002 3b
30  DTE 00:00.002 47
31  DTE 00:00.002 43
32  DTE 00:00.002 37
33  DTE 00:00.002 3a
34  DTE 00:00.002 32
35  DTE 00:00.002 30
36  DTE 00:00.002 32
37  DTE 00:00.002 30
38  DTE 00:00.002 30
39  DTE 00:00.002 30
40  DTE 00:00.002 30
41  DTE 00:00.002 03
42  DCE 00:00.003 06
```

2.4 Internal Processing

This routine will make comment lines, by prepending a percent sign ("`%`") to each input line, out of the first three lines, stick a blank line out and delete line five.

Once the first few lines have been processed, all of the following lines will have their date and the hours part of the time fields deleted.

Month, day and hour fields may be one or two digits long. The year field must be four digits long.

With the alternate file format the first three characters may be "`DCE␣`" or "`DTE␣`".

2.5 Output

This routine identifies its self on the screen and in the output file. The output file has the program's name written as a comment line.

Typical results of this step of processing are:

```
 1  06:06.434
 2  06:35.060 02
 3  06:35.060 41
 4  06:35.060 44
 5  06:35.060 30
 6  06:35.060 31
 7  06:35.060 3b
```

```
 8  06:35.060 52
 9  06:35.066 4f
10  06:35.066 4e
11  06:35.066 3a
12  06:35.066 30
13  06:35.066 03
14  06:35.106 02
15  06:35.106 41
16  06:35.106 44
17  06:35.106 30
18  06:35.112 31
19  06:35.112 3b
20  06:35.112 47
21  06:35.112 43
22  06:35.112 37
23  06:35.112 3a
24  06:35.112 32
25  06:35.112 30
26  06:35.112 32
27  06:35.112 30
28  06:35.112 30
29  06:35.112 30
30  06:35.119 30
31  06:35.119 03
32  06:36.141 02
33  06:36.141 41
34  06:36.148 44
35  06:36.148 30
36  06:36.148 31
37  06:36.148 3b
38  06:36.148 47
39  06:36.148 43
40  06:36.148 37
41  06:36.148 3a
42  06:36.148 32
43  06:36.148 30
44  06:36.148 32
45  06:36.154 30
46  06:36.154 30
47  06:36.154 32
48  06:36.154 30
49  06:36.154 03
50  06:37.177 02
51  06:37.177 41
52  06:37.177 44
53  06:37.177 30
54  06:37.177 31
55  06:37.183 3b
```

3 Fix2.l

3.1 Description

This routine is used to condense the "one byte per line" format that FIX1.L generates into something more useful.

3.2 Calling Sequence

```
fix2 infile outfile
```

3.3 Input Data Format

This routine is designed to take the output from FIX1.L. This output is as follows:

```
 1  06:06.434
 2  06:35.060 02
 3  06:35.060 41
 4  06:35.060 44
 5  06:35.060 30
 6  06:35.060 31
 7  06:35.060 3b
 8  06:35.060 52
 9  06:35.066 4f
10  06:35.066 4e
11  06:35.066 3a
12  06:35.066 30
13  06:35.066 03
14  06:35.106 02
15  06:35.106 41
16  06:35.106 44
17  06:35.106 30
18  06:35.112 31
19  06:35.112 3b
20  06:35.112 47
21  06:35.112 43
22  06:35.112 37
23  06:35.112 3a
24  06:35.112 32
25  06:35.112 30
26  06:35.112 32
27  06:35.112 30
28  06:35.112 30
29  06:35.112 30
30  06:35.119 30
31  06:35.119 03
32  06:36.141 02
33  06:36.141 41
34  06:36.148 44
35  06:36.148 30
36  06:36.148 31
```

```
37  06:36.148 3b
38  06:36.148 47
39  06:36.148 43
40  06:36.148 37
41  06:36.148 3a
42  06:36.148 32
43  06:36.148 30
44  06:36.148 32
45  06:36.154 30
46  06:36.154 30
47  06:36.154 32
48  06:36.154 30
49  06:36.154 03
50  06:37.177 02
51  06:37.177 41
52  06:37.177 44
53  06:37.177 30
54  06:37.177 31
55  06:37.183 3b
```

3.4 Internal Processing

Each line is scanned, the routine looks for an ASCII `STX` (`0x02`) character. When the `STX` is found the current time tag is saved "`STX`" is output. All following time characters are ignored until the next data byte is found. If the next data byte is not an ASCII `ETX` (`0x03`) byte, then the data byte is output as an ASCII character with a leading blank as a delimiter, i.e. `0x33` is output as a "␣3". When an `ETX` is found "`ETX`" is output and a new line is started.

3.5 Output

This routine identifies its self on the screen and in the output file. The output file has the program's name written as a comment line.

Typical results of this step of processing are:

```
 1  06:35.060  STX A D 0 1 ; R O N : 0 ETX
 2  06:35.106  STX A D 0 1 ; G C 7 : 2 0 2 0 0 0 0 ETX
 3  06:36.141  STX A D 0 1 ; G C 7 : 2 0 2 0 0 2 0 ETX
 4  06:37.177  STX A D 0 1 ;
```

4 UnBlank.l

4.1 Description

This routine is a filter that is designed to pass only the characters that are actually wanted in the output. It passes the following character/character strings only, all others are disposed of. (Except for the contents of comment lines.):

“ACK␣”
“␣ETX”
“NAK␣”
“STX␣”
“:␣”
“;␣”
“␣”

[passedthrough]

Processing occurs in the following order so that double blanks get translated into single blanks and all other blanks are deleted. [changed]

1. All double blanks (“␣␣”) are turned into single blanks (“␣”) so as to bypass the general blank dumping and remain as single blanks.
2. Single blanks (“␣”) are dumped, except for those in comments.

4.2 Calling Sequence

```
UnBlank infile outfile
```

4.3 Input Data Format

This routine is designed to take the output from FIX2.L. This output is as follows:

```
1  06:35.060  STX A D 0 1 ; R O N : 0 ETX
2  06:35.106  STX A D 0 1 ; G C 7 : 2 0 2 0 0 0 0 ETX
3  06:36.141  STX A D 0 1 ; G C 7 : 2 0 2 0 0 2 0 ETX
4  06:37.177  STX A D 0 1 ;
```

4.4 Internal Processing

The input file is scanned for items of interest. They are transformed (Section 4.1, page 11) or passed through (Section 4.1, page 11). Other items are dumped.

4.5 Output

This routine identifies its self on the screen and in the output file. The output file has the program's name written as a comment line.

Typical results of this step of processing are:

```
1  06:35.060  STX AD01; RON: 0 ETX
2  06:35.106  STX AD01; GC7: 2020000 ETX
3  06:36.141  STX AD01; GC7: 2020020 ETX
4  06:37.177  STX AD01;
```

5 GcProc.l

5.1 Description

This routine is a filter that translates Panasonic protocol commands into short english statements. It is assumed that the file has been processed by many of the other programs in this series. The following "batch" file is a typical example of its normal calling sequence:

```
fix1 <infile | fix2 | unblank | GcProc >outfile
```

5.2 Calling Sequence

```
GcProc infile outfile
```

5.3 Input Data Format

This routine is designed to take the output from UnBlank.l. This output is as follows:

```
1  06:35.060  STX AD01; RON: 0 ETX
2  06:35.106  STX AD01; GC7: 2020000 ETX
3  06:36.141  STX AD01; GC7: 2020020 ETX
4  06:37.177  STX AD01;
```

5.4 Internal Processing

Internally this routine matches up different translation strings with input strings and outputs them.

5.5 Output

This routine identifies its self on the screen and in the output file. The output file has the program's name written as a comment line.

Typical results of this step of processing are:

```
1  06:35.060 RON 0 <newcc> Suppress: None
2  06:35.106 2020000 <rsro><rfco>  Unknown command
3  06:36.141 2020020 <rsro><rfco>  Unknown command
```

APPENDIX A

A Saving capture files with BREAKOUT

The BREAKOUT program may be used to save communications line data captures. To do this:

1. Start BREAKOUT running.
2. Verify that the communication parameters are correct:
   - 2.1. Inside COMMUNICATIONS (Alt-C):
     - 2.1.1. MONITOR is selected
     - 2.1.2. NORMAL is selected
     - 2.1.3. SINGLE PORT SETTINGS is selected
     - 2.1.4. Inside PORT SETTINGS...
       - 2.1.4.1. BAUD RATE... is correct.
       - 2.1.4.2. DATA BITS... is correct. Default is eight or "8".
       - 2.1.4.3. PARITY... is correct. Default is None or "N".
       - 2.1.4.4. STOP BITS... is correct. Default is one or "1".
   - 2.2. Inside VIEW (Alt-V):
     - 2.2.1. ONLINE is selected.
     - 2.2.2. SINGLE is selected. Other choices are acceptable.
     - 2.2.3. DISPLAY is selected.
     - 2.2.4. UNDERLINE CURSOR is selected. Don't really what this does.
     - 2.2.5. Inside DISPLAY CHARACTER SET... Select something. These don't effect the capture data, but they do make the display meaningful.
   - 2.3. Inside OPTIONS (Alt-O):
     - 2.3.1. Select CAPTURE
     - 2.3.2. Do not select TIMESTAMP, it causes BREAKOUT to crash after awhile and DB does not process its data correctly.
3. Collect a bunch of data. As this is happening the data will be written to the screen and saved in memory.
4. When you want to make a disk copy of your captured data:
   - 4.1. Inside VIEW (Alt-V):
     - 4.1.1. Select Editor

[2] `$Header: d:/txb-p/Captures/RCS/BreakSav.inc,v 1.1 2004-11-04 12:10:27-08 Hamilton Exp Hamilton $`

4.1.1.1. Select FILE (Alt-F):

4.1.1.2. While in FILE select SAVE AS

4.1.1.3. Then take the time tag that shows up in ENTER DESCRIPTOR.

4.1.1.4. Now select SAVE WITHOUT CHANGE.

4.1.1.5. Enter in a file name to use in saving the data. I prefer to use the data and an incrementing last letter. BREAKOUT will append a `.CAP` to your name.
At this point the data will be written to the hard disk. If there is a lot of data this may take a minute or so.

5. There are other options which I do not need to use and thus don't know how to use them.

6. Be sure to copy your data from the hard disk to a floppy for analyzation.

APPENDIX B

B DB

B.1 Description

DB, Dump Breakout, is used to convert saved BREAKOUT files into usable ASCII files.

B.2 Calling Sequence

There are three runtime options to DB:

1. 2: This modifies the operation of DB by putting out a pair of line feed/carriage returns at each change of direction of data instead of the more usual signal pair.
2. A: Prints out an ASCII representation of each input character like this:

```
 1   O-02(.) O-41(A) O-44(D) O-30(0) O-31(1) O-3B(;) O-52(R) O-4F(O) O-4E(N) O-3A(:) O-30(0)
O-03(.)
 2   I-06(.) I-02(.) I-41(A) I-44(D) I-30(0) I-31(1) I-3B(;) I-52(R) I-4F(O) I-4E(N) I-03(.)
 3   O-02(.) O-41(A) O-44(D) O-30(0) O-31(1) O-3B(;) O-47(G) O-43(C) O-37(7) O-3A(:) O-32(2)
O-30(0) O-32(2) O-30(0) O-30(0) O-30(0) O-30(0) O-03(.)
 4   I-06(.)
 5   O-06(.)
```

3. R: Reverses the direction indicators. I.e. the "O-" indicators for "output" become "I-" indicators to show input data. When this is done the above example becomes:

```
 1   I-02(.) I-41(A) I-44(D) I-30(0) I-31(1) I-3B(;) I-52(R) I-4F(O) I-4E(N) I-3A(:) I-30(0)
I-03(.)
 2   O-06(.) O-02(.) O-41(A) O-44(D) O-30(0) O-31(1) O-3B(;) O-52(R) O-4F(O) O-4E(N) O-03(.)
 3   I-02(.) I-41(A) I-44(D) I-30(0) I-31(1) I-3B(;) I-47(G) I-43(C) I-37(7) I-3A(:) I-32(2)
I-30(0) I-32(2) I-30(0) I-30(0) I-30(0) I-30(0) I-03(.)
 4   O-06(.)
 5   I-06(.)
```

All options may be entered in upper or lower case values.

```
db infile outfile
```

B.3 Input Data Format

The file format of a breakout file is detailed in the BREAKOUT documentation. It is in binary and a partial dump is included here.

```
1  000000 0F A5 66 20 53 61 76 65 64 20 46 72 69 20 4E 6F
2  000010 76 20 31 39 20 31 32 3A 35 36 3A 30 33 20 32 30
```

```
 3  000020 30 34 00 20 00 C0 00 02 00 C0 00 41 00 C0 00 44
 4  000030 00 C0 00 30 00 C0 00 31 00 C0 00 3B 00 C0 00 52
 5  000040 00 C0 00 4F 00 C0 00 4E 00 C0 00 3A 00 C0 00 30
 6  000050 00 C0 00 03 00 C1 00 06 00 C1 00 02 00 C1 00 41
 7  000060 00 C1 00 44 00 C1 00 30 00 C1 00 31 00 C1 00 3B
 8  000070 00 C1 00 52 00 C1 00 4F 00 C1 00 4E 00 C1 00 03
 9  000080 00 C0 00 02 00 C0 00 41 00 C0 00 44 00 C0 00 30
10  000090 00 C0 00 31 00 C0 00 3B 00 C0 00 47 00 C0 00 43
11  0000A0 00 C0 00 37 00 C0 00 3A 00 C0 00 32 00 C0 00 30
12  0000B0 00 C0 00 32 00 C0 00 30 00 C0 00 30 00 C0 00 30
13  0000C0 00 C0 00 30 00 C0 00 03 00 C1 00 06 00 C0 00 06
```

B.4 Internal Processing

When called DB reads the command line for any runtime options, it then reads the BREAKOUT capture's file header and generates its own header: (Note that the RCS control line has been intentionally modified.)

```
%       db.c,  163: $ Id: db.c,v 1.17 2002-07-12 08:13:19-07 Hamilton Exp Hamilton $
%
%       db.c,  166: Dumping "breakout\19nov04a.cap" a 88888 byte file.
%
%       db.c,  176: The .CAP file's header is:
%       db.c,  178: "Saved Fri Nov 19 12:56:03 2004
%
%       db.c,  187: Format of this output is as follows:
%       db.c,  188: There are two directions of data flow.
%       db.c,  189: Data direction is indicated with either
%       db.c,  190: a "1" or "0" followed by a "-".
%       db.c,  191: It is intended that "1" will
%       db.c,  192: be used to indicate data that is coming in,
%       db.c,  193: and that "0" is used for data
%       db.c,  194: going out. If this is not true, then use
%       db.c,  195: the /R option to reverse the logic
%       db.c,  196: of these indicators
%       db.c,  197: Following the "-" is the value of the
%       db.c,  198: data byte in hexidecimal.
%       db.c,  199: Whenever a direction change is detected
%       db.c,  200: a new line is output.
%       db.c,  202: Data source directions are reversed.
%       db.c,  205: Each direction-data item is preceeded
%       db.c,  206: by a single blank.
%
```

It then reads the capture file and translates it into ASCII. Translation consists of marking input data with "`I-`" and output data with "Ø-" indicators. In reality the directions are totally arbitrary and may be changed by using the "`R`" runtime flag. When a change from input to output, or visa versa, occurs a carriage return/line feed is output.

This processing continues until the end of data when a short summary of the work is output:

```
%
%       db.c,  224: There were 22213 data bytes converted
%       db.c,  226: There were 1412 lines output
%
```

B.5 Output

Typical results of this step of processing are:

```
1
2   O-02 O-41 O-44 O-30 O-31 O-3B O-52 O-4F O-4E O-3A O-30 O-03
3   I-06 I-02 I-41 I-44 I-30 I-31 I-3B I-52 I-4F I-4E I-03
4   O-02 O-41 O-44 O-30 O-31 O-3B O-47 O-43 O-37 O-3A O-32 O-30 O-32 O-30 O-30 O-30 O-30 O-03
5   I-06
```

APPENDIX C

C Fix0.l

C.1 Description

DB breaks sequences of commands whenever the direction of data flow changes. Sometimes additional breaks are desirable and FIX0 makes them.

C.2 Calling Sequence

```
fix0 infile outfile
```

C.3 Input Data Format

FIX0 expects the output of DB to be its input.

```
1
2   O-02 O-41 O-44 O-30 O-31 O-3B O-52 O-4F O-4E O-3A O-30 O-03
3   I-06 I-02 I-41 I-44 I-30 I-31 I-3B I-52 I-4F I-4E I-03
4   O-02 O-41 O-44 O-30 O-31 O-3B O-47 O-43 O-37 O-3A O-32 O-30 O-32 O-30 O-30 O-30 O-30 O-03
5   I-06
```

C.4 Internal Processing

Whenever any of several data patterns are discovered, they are written out with with a carriage return/line feed inserted in them. The patterns that are "broken" are: \n indicates the location of the carriage return/line feed pair.)

1. ␣I-03␣I-02 is translated to: ␣I-03\n␣I-02.
2. ␣O-03␣O-02 is translated to: ␣O-03\n␣O-02.
3. ␣O-06␣O-02 is translated to: ␣O-06\n␣O-02.
4. ␣I-06␣I-02 is translated to: ␣I-06\n␣I-02.

C.5 Output

Typical results of this step of processing are:

```
1
2   O-02 O-41 O-44 O-30 O-31 O-3B O-52 O-4F O-4E O-3A O-30 O-03
3   I-06
4   I-02 I-41 I-44 I-30 I-31 I-3B I-52 I-4F I-4E I-03
5   O-02 O-41 O-44 O-30 O-31 O-3B O-47 O-43 O-37 O-3A O-32 O-30 O-32 O-30 O-30 O-30 O-30 O-03
6   I-06
```

APPENDIX D

D FixB.l

D.1 Description

This routine is used to "translate" capture files from BREAKOUT format files into FTS format capture files. Prior to running this filter the BREAKOUT capture file must have been run through DB, which converts the raw breakout capture file into an ASCII character representation of the data. And through FIX0 to complete the clean up.

This has been done so that all processing of Panasonic protocol data may use a common set of analysis filters.

D.2 Calling Sequence

```
fixb infile outfile
```

D.3 Input Data Format

This routine is designed to take a BREAKOUT capture file that has been processed by FIX0. The output of FIX0 is as follows:

```
1
2   O-02 O-41 O-44 O-30 O-31 O-3B O-52 O-4F O-4E O-3A O-30 O-03
3   I-06
4   I-02 I-41 I-44 I-30 I-31 I-3B I-52 I-4F I-4E I-03
5   O-02 O-41 O-44 O-30 O-31 O-3B O-47 O-43 O-37 O-3A O-32 O-30 O-32 O-30 O-30 O-30 O-30 O-03
6   I-06
```

D.4 Internal Processing

Before the file is read, a fake header is written out. While processing the file, any lines that are comment lines are deleted. Then each five character ASCII representation of a BREAKOUT captured protocol byte is translated by outputting:

1. A data source ID code, "DCE␣" or "DCD␣".
2. A fake date of "11/22/3333␣".
3. A fake time tag which is the message number. The message number is incremented on each ETX
4. Followed by a blank and the actual data byte.

D.5 Output

This routine identifies its self on the screen and in the output file. The output file has the program's name written as a comment line.

Typical results of this step of processing are:

```
 1  FTS capture file, generated from a breakout file.
 2  Event <none>
 3  Event <none>
 4
 5  Timestamp <none>
 6  Sd Timestamp <none>
 7
 8  DTE 11/22/3333 44:00:00.000 PM 02
 9  DTE 11/22/3333 44:00:00.000 PM 41
10  DTE 11/22/3333 44:00:00.000 PM 44
11  DTE 11/22/3333 44:00:00.000 PM 30
12  DTE 11/22/3333 44:00:00.000 PM 31
13  DTE 11/22/3333 44:00:00.000 PM 3b
14  DTE 11/22/3333 44:00:00.000 PM 52
15  DTE 11/22/3333 44:00:00.000 PM 4f
16  DTE 11/22/3333 44:00:00.000 PM 4e
17  DTE 11/22/3333 44:00:00.000 PM 3a
18  DTE 11/22/3333 44:00:00.000 PM 30
19  DTE 11/22/3333 44:00:00.000 PM 03
20  DCE 11/22/3333 44:00:00.001 PM 06
21  DCE 11/22/3333 44:00:00.001 PM 02
22  DCE 11/22/3333 44:00:00.001 PM 41
23  DCE 11/22/3333 44:00:00.001 PM 44
24  DCE 11/22/3333 44:00:00.001 PM 30
25  DCE 11/22/3333 44:00:00.001 PM 31
26  DCE 11/22/3333 44:00:00.001 PM 3b
27  DCE 11/22/3333 44:00:00.001 PM 52
28  DCE 11/22/3333 44:00:00.001 PM 4f
29  DCE 11/22/3333 44:00:00.001 PM 4e
30  DCE 11/22/3333 44:00:00.001 PM 03
31  DTE 11/22/3333 44:00:00.002 PM 02
32  DTE 11/22/3333 44:00:00.002 PM 41
33  DTE 11/22/3333 44:00:00.002 PM 44
34  DTE 11/22/3333 44:00:00.002 PM 30
35  DTE 11/22/3333 44:00:00.002 PM 31
36  DTE 11/22/3333 44:00:00.002 PM 3b
37  DTE 11/22/3333 44:00:00.002 PM 47
38  DTE 11/22/3333 44:00:00.002 PM 43
39  DTE 11/22/3333 44:00:00.002 PM 37
40  DTE 11/22/3333 44:00:00.002 PM 3a
41  DTE 11/22/3333 44:00:00.002 PM 32
42  DTE 11/22/3333 44:00:00.002 PM 30
43  DTE 11/22/3333 44:00:00.002 PM 32
44  DTE 11/22/3333 44:00:00.002 PM 30
45  DTE 11/22/3333 44:00:00.002 PM 30
46  DTE 11/22/3333 44:00:00.002 PM 30
```

```
47  DTE 11/22/3333 44:00:00.002 PM 30
48  DTE 11/22/3333 44:00:00.002 PM 03
49  DCE 11/22/3333 44:00:00.003 PM 06
```

APPENDIX E

Index

0x02, 10
0x03, 10
0x33, 10

ASCII, 5, 10, B-1, B-2, D-1

breakout, A-1, B-1, B-2, D-1

db, A-1, B-1, B-2, C-1, D-1

ETX, 10, D-1

fix0, C-1, D-1
fix1.l, 9
fix2.l, 11
FTS, D-1

Panasonic, 13, D-1

RCS, 4
RS-485, 3

STX, 10

WV-CU161, 3