Example walk-through of a decompilation with problems 11/01/2022

This is an example of trying to decompile Digital Precisions Desk Top Publisher program.

Load and run the **Discharge_bas** program

```
Enter the filename of the Executable file - dos3_example_publish_t
ask

Setting dump file to use as     - dos3_example_publish_task_dmp
Setting codes file to create as - dos3_example_publish_task_codes

File path to Discharge library files - dos8_

OK?
```

```
Qemulator file header found in dump file
Compensating

Analizing Files...

Job name  : PUBLISHER
Copyright :  1987 The Turbo Team.

Version                   5.10
Dump start                $003457FE
Dump A6 value             $0034D7F6
Sub routines start around $00345A20
Subroutine end marker     JMP   $00(a6,d0.w)
Line number key code      $FFFF
First basic line start    $00349126
Program end               $00345BF6
Keyword table starts at   $00345BFA
Using identity file        dos8_TCLibrary510_5_id

Searching for imbedded SuperBASIC extensions


No SuperBASIC extensions found

There are 2 potential problems in the codes file

Processing complete. _lib and codes file created

Do you want to start the main decompiler? (y/n)
```

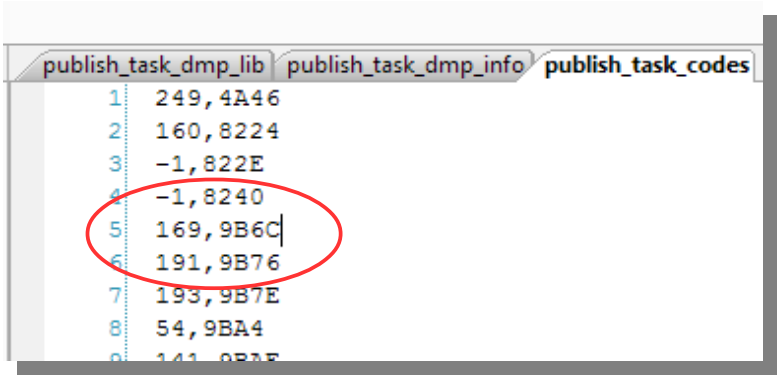
So far, so good. It reports there are a couple of potential problems in the codes file.
We now press 'y' to start the main decompiler.

```
Qemulator header found in executable file
compensating
Title     1987 The Turbo Team.
Version  5.10
Job Name PUBLISHER
```

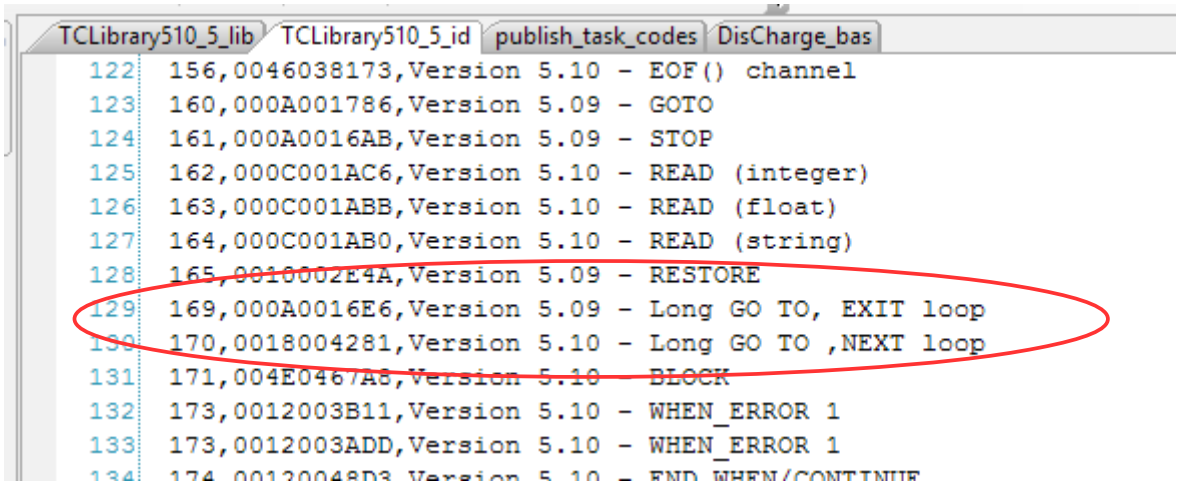
```
-1 index found for 822E, ignoring entry
-1 index found for 8240, ignoring entry
Warning, Code Array index 169 already in use, Overwriting
Warning, Code Array index 81 already in use, Overwriting
Warning, Code Array index 82 already in use, Overwriting
Warning, Code Array index 148 already in use, Overwriting

Unexpected code $9B6C found while
analyzing array variables and searching
for the start of the BASIC program.

Program removed from memory. RUN needed.
```

Something's gone wrong right from the start. There is a problem with code $9B6C while analysing arrays.

If we open the **_codes** file in a text editor.

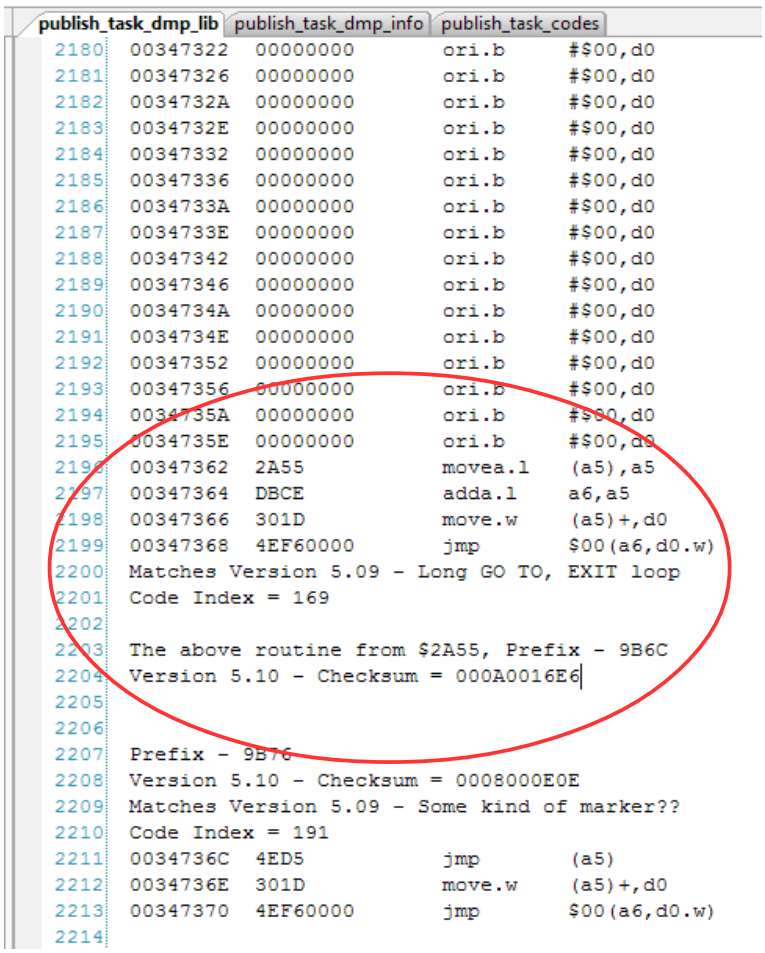


Note that 9B6C has been assigned a code of 169. And if we look at the Code array keys document, or the TCLibrary510_5_id file.



We see that code 169 is used for EXIT loop.

If we now look at the created disassembly **_dmp_lib** file. And search for 9B6C we find.



```
publish_task_dmp_lib | publish_task_dmp_info | publish_task_codes
2180  00347322  00000000        ori.b     #$00,d0
2181  00347326  00000000        ori.b     #$00,d0
2182  0034732A  00000000        ori.b     #$00,d0
2183  0034732E  00000000        ori.b     #$00,d0
2184  00347332  00000000        ori.b     #$00,d0
2185  00347336  00000000        ori.b     #$00,d0
2186  0034733A  00000000        ori.b     #$00,d0
2187  0034733E  00000000        ori.b     #$00,d0
2188  00347342  00000000        ori.b     #$00,d0
2189  00347346  00000000        ori.b     #$00,d0
2190  0034734A  00000000        ori.b     #$00,d0
2191  0034734E  00000000        ori.b     #$00,d0
2192  00347352  00000000        ori.b     #$00,d0
2193  00347356  00000000        ori.b     #$00,d0
2194  0034735A  00000000        ori.b     #$00,d0
2195  0034735E  00000000        ori.b     #$00,d0
2196  00347362  2A55            movea.l   (a5),a5
2197  00347364  DBCE            adda.l    a6,a5
2198  00347366  301D            move.w    (a5)+,d0
2199  00347368  4EF60000        jmp       $00(a6,d0.w)
2200  Matches Version 5.09 - Long GO TO, EXIT loop
2201  Code Index = 169
2202
2203  The above routine from $2A55, Prefix - 9B6C
2204  Version 5.10 - Checksum = 000A0016E6
2205
2206
2207  Prefix - 9B76
2208  Version 5.10 - Checksum = 0008000E0E
2209  Matches Version 5.09 - Some kind of marker??
2210  Code Index = 191
2211  0034736C  4ED5            jmp       (a5)
2212  0034736E  301D            move.w    (a5)+,d0
2213  00347370  4EF60000        jmp       $00(a6,d0.w)
2214
```

Notice that this is the first identified code routine 9B6C. You can tell because the routine has the header after the routine. And all those zeros above it, does not look like machine code.

The first code routine is usually a **GO TO** code of 160, But as the publisher program is very large, we would expect it to be a long **GO TO**, code 240.

Note that it's been identified as a code 169, which is used in EXIT loop's. note the Checksum value.

If we look at the **TCLibrary510_5_id** file again.

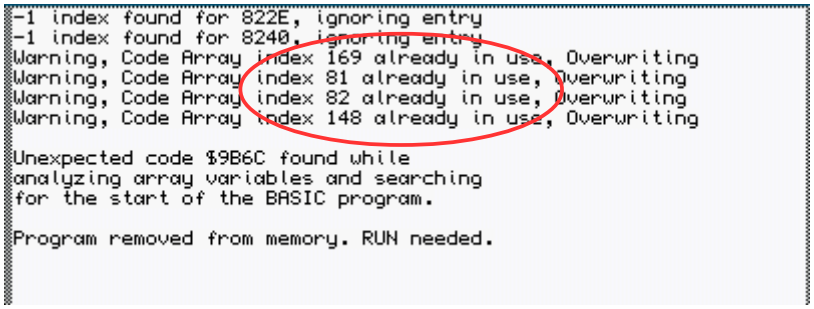
```
127  164,000C001AB0,Version 5.10 - READ (string)
128  165,0010002E4A,Version 5.09 - RESTORE
129  169,000A0016E6,Version 5.09 - Long GO TO, EXIT loop
130  170,0018004281,Version 5.10 - Long GO TO ,NEXT loop
131  171,004E0467A8,Version 5.10 - BLOCK
```

```
155  226,003E0179F1,Version 5.10 - MOVE_MEMORY4
156  230,00260091ED,Version 5.10 - PEEK$
157  240,000A0016E6,Version 5.10 - Move program pointer to the long wo
158  231,0014004639,Version 5.10 - OPTION_CMD$
159  232,0018004C1E,Version 5.10 - LEN(string variable) - different to
```

Notice that the checksum for code 169, and 240 are the same. This is an example of there being two identical code routines, with different uses

So we can change the 169,9B6C in the **_codes** file to 240,9B6C

We are probably fixing another problem at the same time.



code 169 was defined twice, this is probably the real **EXIT loop**.

Code 81 is assigned as both A580, and B4AA. And code 82 is assigned as both A58A, and B4B4

If you look at the Code array document, you will notice the code 81, and 82. Do the same job as codes 290 and 291. So we can amend B4AA to 290, and B4B4 to 291, in the **_codes** file.

This leaves code 148. Looking in the **_codes** file, we find code 148 defined twice as 9C12 and B554.

If we look at code 148 in the Code array keys document, we find it defined in **SELect On** as '**= (ON)(integer) same as index 1**'. And index 1 is defined as '**= Equal (integer)**'.

So one of these two 148's need to be changed to 1.

You could search the **_dmp_lib** file for the B554 and 9C12 code and hand decompile around around them to try to establish which is which (see the Hand Decompiling section in the main DisCharge manual). Or, you can do a short cut, as I have here.

If you look at the Decompiler Technical Notes document. In the flow diagrams for **SElect**, you will notice that after a code 148, there will be either a code 140, a code 144, or a code 145.

in the **_codes** file you will find there is no codes 140, or 145. But there is a code 144,9D58. If you search the **_dmp_lib** file for B554, and 9C12. You will notice that there are several 9D58 following B554's. But no 9D58 following a 9C12

publish_task_codes | publish_task_dmp_lib

```
33688  0035B7D6  0002ABCC        ori.b     #$CC,d2                  ....
33689  0035B7DA  9E8E            sub.l     a6,d7                    ..
33690  0035B7DC  00054D4F        ori.b     #$4F,d5                  ..MO
33691  0035B7E0  4445            neg.w     d5                       DE
33692  0035B7E2  2000            move.l    d0,d0                     .
33693  0035B7E4  ABF4            ILLEGAL   INSTRUCTION              ..
33694  0035B7E6  9C4C            sub.w     a4,d6                    .L
33695  0035B7E8  93F89BE8        suba.l    $FFFF9BE8,a1             ....
33696  0035B7EC  0001B554        ori.b     #$54,d1                  ...T
33697  0035B7F0  9D58            sub.w     d6,(a0)+                 .X
33698  0035B7F2  0000E006        ori.b     #$06,d0                  ....
33699  0035B7F6  9B6C0000        sub.w     d5,$0000(a4)             .l..
33700  0035B7FA  E01E            ror.b     #$08,d6                  ..
33701  0035B7FC  9BE80002        suba.l    $0002(a0),a5             ....
```

So it's reasonable to assume that B554 is a code 148, so 9C12 must be a code 1.

You could just try the two codes one way around, and do a decompile. And if you get problems, try the other way around.

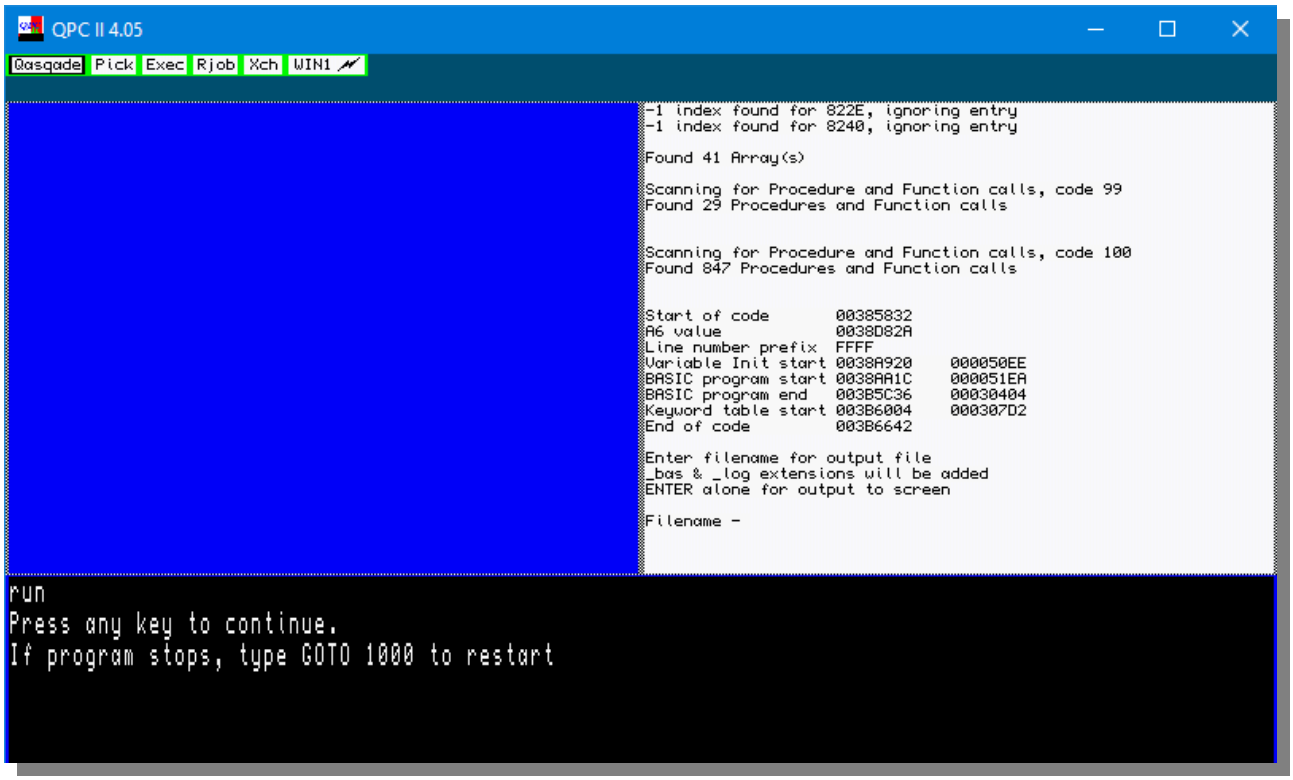
That just leaves one more issue to look at in the **_codes** file. 822E and 8240.

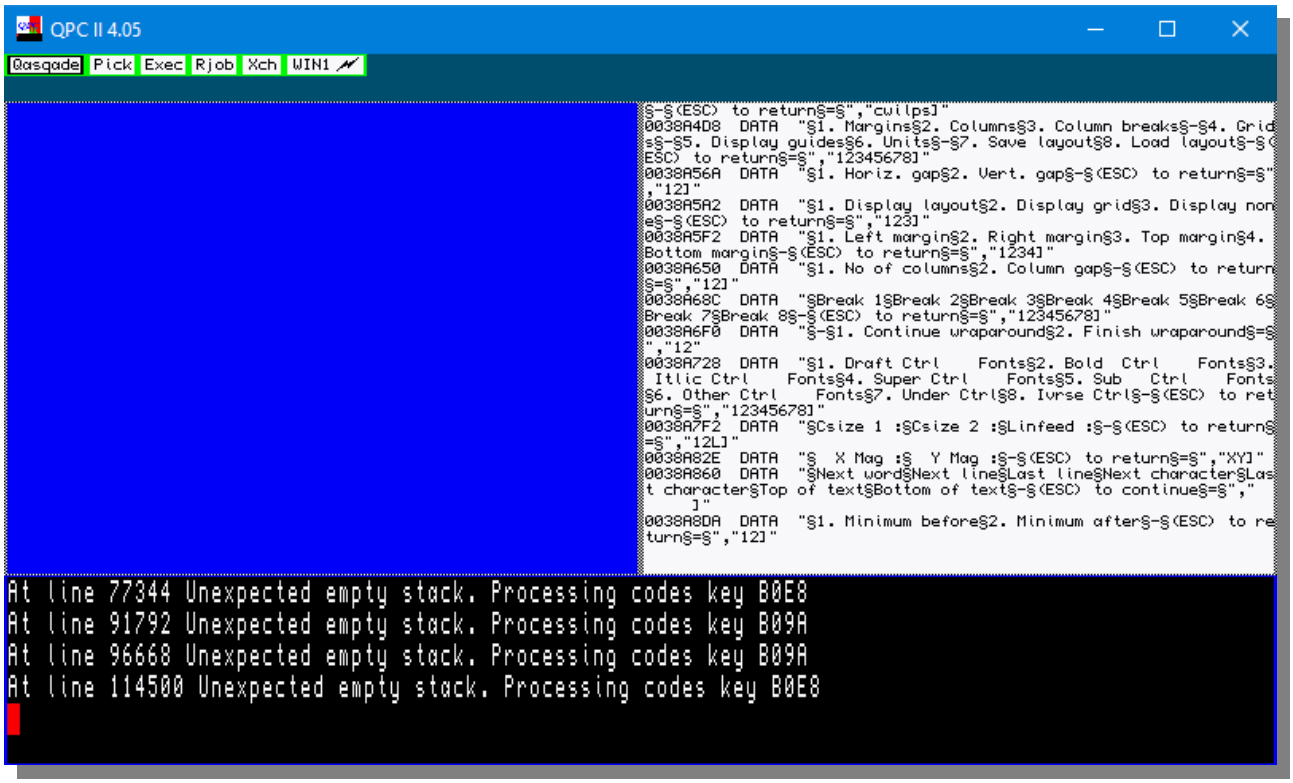
```
-1 index found for 822E, ignoring entry
-1 index found for 8240, ignoring entry
Warning, Code Array index 169 already in use, Overwriting
Warning, Code Array index 81 already in use, Overwriting
Warning, Code Array index 82 already in use, Overwriting
Warning, Code Array index 148 already in use, Overwriting

Unexpected code $9B6C found while
analyzing array variables and searching
for the start of the BASIC program.

Program removed from memory. RUN needed.
```

We already decided earlier that 9B6C is the start of the code routines. So anything lower than that is probably a miss-identification. That can just be ignored. If you search for 822E in the **_bmp_lib** file, you will notice that 8240 does not look like a valid machine code routine.

So **RUN** the program again, or if you have reset at all. Load and run the **TurboDisCharge2_bas** program. (TurboDisCharge2_bas because the Line number key code was $FFFF meaning there are no included line numbers).

```
Enter the filename of the Executable file - dos3_example_publish_t
ask

Using codes file  - dos3_example_publish_task_codes

OK?

Qemulator header found in executable file
compensating
Title     1987 The Turbo Team.
Version  5.10
Job Name PUBLISHER
```





And the decompile now completes. The Unexpected empty stack messages are not errors, but warnings that the decompiler expected something to be on the stack, but it was not there. These are most likely to be caused by a Defined Function being miss-recognised as a Procedure. Where a Function would leave a return value on the stack.