22th July 2018 Martin Head

SuperCharge & TurboCharge Decompiler Technical Notes

Checksums in Library files

The checksum is comprised of 40 bits. Split into two parts.
A 16 bit, byte count of the routine. And a 24 bit checksum of the bytes of the routine. And is shown as a 10 character hexadecimal number

example -
00100033C7

This routine contains 16 bytes ($0010), and the 16 bytes make a checksum of ($00033C7)

Procedures and Functions

PROCedure SaveMe
Saves a copy of the program

FuNction GetString$(ptr)
Returns a string of a standard QDOS string (word length then bytes of string) stored in memory at ptr

FuNction GetInt(ptr)
Returns a signed integer stored in memory at ptr

FuNction GetWord(address)
Returns an unsigned integer stored in memory at address

FuNction FLT(string$)
Converts a 12 character (six byte) hex string into a floating point number

FuNction FLT$(n)
Convert a Floating point number into a 12 character (six byte) hex string
by PJW & Steven Pool V0.02 03/12/17

FuNction NameListLen(start)
Returns the number of the highest name list entry in the keywords section at the end of the compiled program
start is the address in memory of the keyword table

PROCedure EmptyPipe (channel)
Makes sure the pipe is empty. If it is not empty, it is reported to channel, the pipe is emptied displaying the contents of the pipe.

FuNction GetTOS$
Remove and return the item at the rear end of the pipe.

PROCedure FlagLine (num$)
If the line number, num$ does not already exist in the Proc/Fun list, it is added.

FuNction CheckProcFun(num$)
Checks to see if the line number, num$ is in the Proc/Fun list. If it is, then create a DEFine PROCedure/FuNction line.
Also checks for unresolved SELect's and IF's.
Returns

0 If the line number is not in the Proc/Fun list.
1 If its a multiline DEFine PROCedure/FuNction.
2 If its a single line DEFine PROCedure/FuNction.

PROCedure AssignArray(varNum,noElements)
Adds an array entry into the arrayElements array

FuNction GetElements(varNum)
Returns the number of elements for an array. varNum is the SuperCharge reference for the array.

PROCedure ListProcs
List the Procedure/Function line numbers to the log file

PROCedure ReadCodeArray (file$)
Read in the code array from file$. The Format is index number,hex word index value
E.g. 23,90FA - codeArray(23)=$90FA

PROCedure DoDataLines
Create DATA statements. DATAarray(0) holds the number of RESTORE commands.
Other indexes are addresses of data starts.

PROCedure CheckELSE (ptr)
Check to see if an ELSE is needed. ptr is the current program position.

PROCedure StripIFStack
Strips the top item off the IFStack and ELSEStack string

FuNction FindLine(ptr)
Scan from the current program point (ptr) looking for the start of the next program line.

PROCedure LoadFile (file$)
Load the file (file$) and set base addresses initialBase, base, and filelength.

PROCedure CheckValues
Check values in file, and set various pointers.

PROCedure GetVersion
Get program version information and set marker variables.

PROCedure FindArrayElements
Scan variable initialization area looking for arrays, and filling in arrayElements.

PROCedure StartNewLine (linenumber) (tc)
Start a new listing line starting with the supplied line number.

PROCedure InitNewLine (tc)
Initialize line decompose variables.

PROCedure InitMain (tc)
Set up variables for decompiling.

PROCedure StatementSeperator (tc)
If line numbers are suppressed, start a new line. Otherwise output a colon (:) separator.

PROCedure CheckEndSelects (tc)
Checks for any outstanding END SELects to deal with

Global variables

(sc)	SuperDisCharge	If one of these two are shown, it means that this variable is
(tc)	TurboDisCharge	specific to one of the decompilers.

filelength	Length of executable file.
initalBase	Loaded base address, may be different to ‘base’ if Qemulator header found.
base	Start of the executable.
finish	End of the executable.
progOffset	Offset from start of executable to start of BASIC program area.
prog	Running pointer for the BASIC program.
basicProgStart	Start of BASIC program area.
varInitOffset	Offset from start of executable to start of variable init area.
varInitStart	Start of variable init area.
keywords	Start of keyword table.
progEnd	End of BASIC program area.
TCver$	Library file version from executable.
TCtitle$	TurboCharge copyright message.
jobName$	Executables job name.

DATAendMarker		Word that marks the end of any DATA in the initialization area.
lineStart		Word that marks ‘move.w (a5)+,d6’, the start of a BASIC program line.
progStartAdj	(tc)	An adjustment for the start of the BASIC program in compiler versions that have a RETRY_HERE inserted before the start of the BASIC program
A6		Address in loaded executable that represents the A6 register.
keywordOffset		Address that is a base for accessing Keywords. Not a pointer to the keyword

		list itself.
lineNoExist		1 if compiled program has embedded line numbers 0 if compiled program does not have line numbers.
freeform	(tc)	1 if the FREEFORM option was used during compiling. 0 if the STRUCTURED option was used during compiling. The decompiler will assume FREEFORM until it encounters a STRUCTURED style Procedure of Function definition.
lineStartPrefix		Prefix code for BASIC program line start, or -1 if no embedded line numbers.
lineCount		Number of program lines decompiled.
lineNo		Line number of line currently de compiling.
pseudoLine	(tc)	Address of a reference point when line numbers are suppressed
ch		Channel to send decompiled output to.
er (sc) errCh	(tc)	Channel to send Errors & log to.
proc$		Line numbers of Procedures and Functions, separated by *****
pcount		Number of items in the pipe.
InIF		Number of levels of nested IF's.
IFStack$		List of 8 character HEX destination addresses of END IF's or ELSE's.
ELSEStack$		
FORselect$	(tc)	Used to hold the selection for FOR statement e.g. "1 TO 5,8,10 TO 12"
FORstep	(tc)	Used to hold the STEP value
FORvar$	(tc)	Used to hold the FOR variable reference
newLine		0 = Not at the start of a new line.
InPrint		0 = Not in a PRINT/INPUT.
InMultiSel	(sc)	0 = Not in a multi SELect ON e.g. y=1,2,3,4.
EndSelAdd		0 = Not in a SELect, or addrerss of END SELect.
SELselect$	(tc)	Used to hold the selection for SELect statement e.g. "1 TO 5,8,10 TO 12"
doPreLoad		1 = Do any preLoad$.
InProcFun		0 = Not in a Procedure or Function
InSelOn		0 = Not in a SELect, or address of next SELect test, ON REMAINDER, or end of SELect.
SelVar$		Last used SELect variable.
Indef		0 = Not in any DEFine's. Once in a DEFinition, it points to just past the end of the definition block. Unless the STRUCTED option was used during compiling, Then Indef is set to -1.
channelOpenCheck		0 = No pending channel operations
key		Holds the current operation prefix code.

Arrays

nameList()	Array of offsets for keywords at end of code.
arrayElements(20,1)	Holds the number of elements in arrays, 21 entries of 2
DATAarray(30)	DATA statement RESTORE points
codeArray(249)	The code array, 250 entries

Turbo Toolkit commands handled in the Keywords section

ALLOCATION, DEALLOCATE, SEARCH_MEMORY, PEEK_F, BASIC_F, FREE_MEMORY, BASIC_B%, BASIC_W%, BASIC_L, BASIC_POINTER, BASIC_NAME$, BASIC_TYPE%, BASIC _INDEX%, TYPE_IN, SET_PRIORITY, COMMAND_LINE, END_CMD, CONNECT, CHANNEL_ID, SET _CHANNEL, EXECUTE, EXECUTE_W, EXECUTE _A, DEFAULT _DEVICE, LIST_TASKS, REMOVE_TASK, SUSPEND_TASK, RELEASE_TASK, DATASPACE, CURSOR_ON, CURSOR_OFF, EDIT$, EDITF, EDIT%, SET_FONT, POSITION, SET_POSITION, INTEGER$, FLOAT$, STRING$, STRING%, STRINGF, GET%, GETF, GET$, INPUT$, DEVICE_STATUS, DEVICE_SPACE,

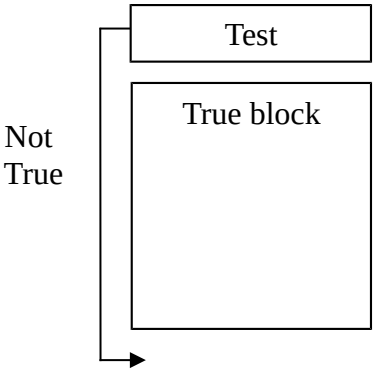
Turbo toolkit commands handled by the compiler internally

PEEK$, POKE$, COMPILED, OPTION_CMD$, DATA_AREA (makes an empty line), RETRY_HERE

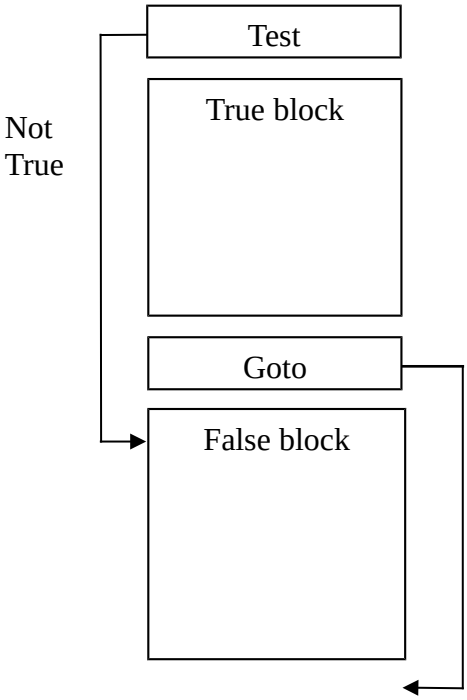
Format of program storage in the compiled program

Where there are differences between SuperCharge and TurboCharge, and the different versions. It will be maked, as so - (TC 5.10) meaning the format for Turbo with a code generator of Version 5.10 is different.

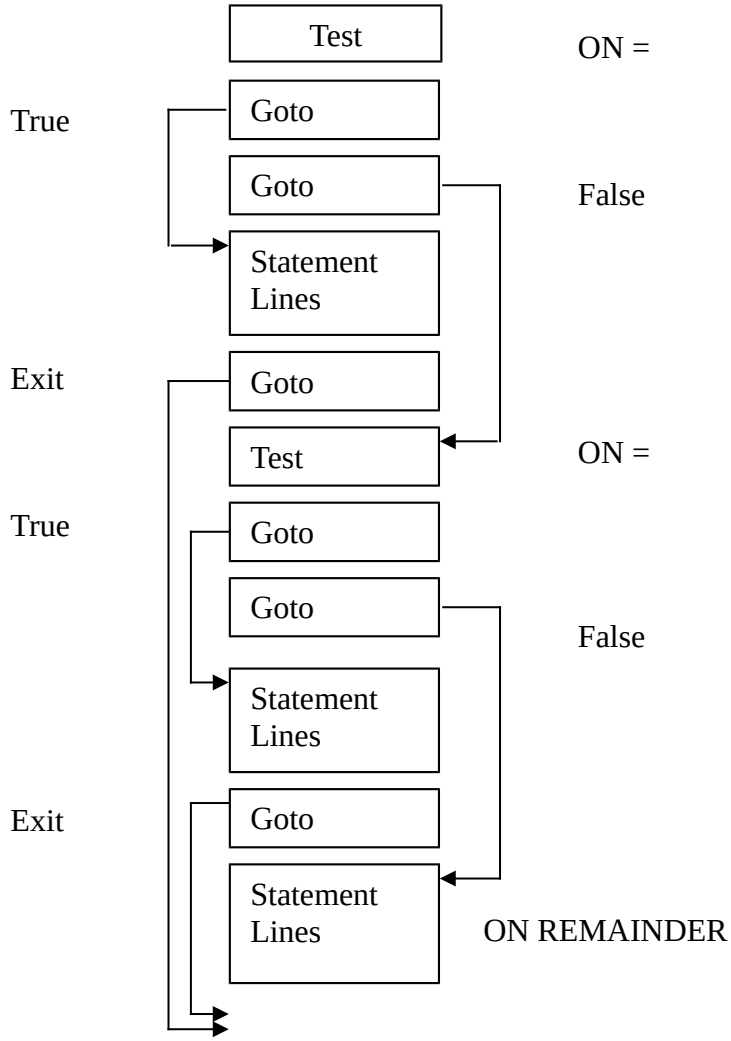
IF..THEN



IF..THEN..ELSE



SELect ON



Procedure and Function definitions

SuperCharge

Start	2 words	Go To (160),	A6 offset

Followed by the parameters, if any. Note that the parameters are in reverse order

Float	4 words	Code 102,	variable reference,
		Code 63,	variable reference
String	6 words	Code 85	
		Code 103,	variable reference
		Code ??	
		Code 64	variable reference

Turbo is as above except for strings

String	2 Words	Code 103

If the Freeform option was used in compilation, then there is no GOTO (code 160) and offset at the start. Except for the first definition, which has a STOP (code 161).

Calling Procedures and Functions

SuperCharge
After the parameters (if any) have been placed on the stack.

	2 Words	Code 100,	A6 offset
Turbo	3 Words	Code100,	Long A6 offset

In-bedded DATA statements

The contents of DATA lines are not in-bedded in the codec BASIC program where they belong, But inserted in the initialization area before the main part of the BASIC program.

The end of the DATA being marked with a $8300 for SuperCharge, and $8001 for TurboCharge.

Decoding the DATA is as follows -

SuperCharge

TurboCharge

If the first word is less then $7FFF, Then it's a string in the normal QDOS fashion of a length words followed by the characters of the string

If the first word is $8000 and above, Then its a number.
- Do a 2's complement (invert all bits and add 1) to the word
- If the result is less than, or equal to $FFF, Then it's a float. In which case, subtract $1BB8 from this result. And this is the first two bytes of the float. And the next two words are the rest of the float.
- If the result is greater than $FFF, Then it's an integer.

If the first word is exactly $7FFF, Then it's part of a T_CONFIG. In which case, read the next word, which is a normal integer.