Tak: Racing Code Through Time

Setup and User Guide

```
DEFFN TAK(X%,Y%,Z%)
  IF X% <= Y% THEN = Z%
  = TAK( TAK(X%-1,Y%,Z%), TAK(Y%-1,Z%,X%), TAK(Z%-1,X%,Y%) )
```

acheton1984, Issue 1, 24th March 2025

Table of Contents

Final page: 35

1 Introduction

This Setup and User Guide accompanies the main "Tak: Racing Code Through Time" analysis article and explains how to set up and run the Tak benchmarks on an Acorn BBC Microcomputer or emulator.

To avoid distributing additional or partial copies of language software, which is readily available elsewhere, none is included. Instead, this guide tells you where to find the necessary language ROMs, discs, and manuals, and what to merge with the project's Tak DFS disc images to get each language or test working.

You may choose an alternative approach—such as multiple floppies, ADFS, or Econet—but that will be your own adventure! Though hopefully this guide will provide enough information to point you in the right direction for your system.

Prior experience with BBC Microcomputer systems is assumed, including familiarity with system '*' commands, disc management, and ROM loading.

All project materials, including analysis, results spreadsheet, disc images, and this setup guide, are available in the project repository:
https://github.com/acheton1984/ReTestingTheTak

For questions, issues, or discussion, please open an issue on GitHub:
https://github.com/acheton1984/ReTestingTheTak/issues

1.1 Languages and Tests Covered

All disc images are available from the project repository and include the following languages:

- Assembler & BASIC (TakBasicAsm.ssd)
- BCPL (TakBCPL.ssd)
- Beebug C (TakBeebugC.ssd)
- COMAL (TakCOMAL.ssd)
- FORTH (TakFORTH.ssd)
- ISO-Pascal (TakIsoPascal.ssd)
- LISP (TakLISP.ssd)
- micro-PROLOG (TakmProlog.ssd)
- S-Pascal (TakSPascal.ssd)

Additional tests:

- BCPL ADFS v DFS performance (BCPL_FSTest.ssd)

Links to original manuals, if needed, are included in Appendix B.

1.2 Recommended System Requirements

Recommended system requirements (either real or emulated):

- BBC B or Master 128.
- External 6502 “cheese wedge” second processor or internal “Turbo” co-processor.
- An 80-track single-sided drive (or equivalent) and DFS.
- Two free Sideways RAM/ROM slots.

Even with a second processor, run times can be up to 15 minutes (28+ for Logo, currently being researched). Therefore, a top tip is to press Ctrl-G after starting the Tak function, so the machine will wake you with a beep when it finishes.

1.3 BBZ: Languages at Your Command (Line…)

Late addition: just before this guide went to press, I discovered BBZ. It is not an emulator per se, but a BBC-like 6502 environment that intercepts just enough MOS calls for many BBC Micro languages to run successfully in a text console on Mac, Linux or Windows.

Although it was not used during the Tak project, and its run times will depend more on the host CPU rather than a BBC's 6502 (real or emulated), it nevertheless seems to offer an intriguing way to develop, test, and run language code quickly and easily.

Source and compiled BBZ code is available from https://github.com/ivanizag/bbz

BBZ already includes most of the necessary language ROMs, but adding the two Beebug C ROMs allows nearly all tested languages to run:

```
./bbz -rom0 BASIC.ROM -rom1 ROMs/Forth_103.rom -rom2 ROMs/LISP501.ROM -rom3
ROMs/COMAL.rom -rom4 ROMs/MPROLOG310.rom -rom5 ROMs/Pascal-1.10-Compiler.rom -rom6
ROMs/Pascal-1.10-Interpreter.rom -rom7 ROMS/BCPL-7.0.rom -rom8 ROMS/BeebugC-COMP -rom9
ROMS/BeebugC-LANG
```

Once running, a limited subset of ‘*’ commands is available. You can use *. and *DIR as normal, though *DIR .. is needed to go up a level instead of *DIR ^.

Not everything runs, and compiling hasn't been tested, but here's a summary of what works:

No:	BASIC/TAKscv, S-Pascal.
Yes:	BASIC/TAKasm, TAK, TAKfp, TAKstr, BCPL, Beebug C, COMAL, ISO-Pascal, LISP, micro-PROLOG.
Yes, but:	FORTH - paste F.TAK’s contents into the console, as *EXEC is unavailable.

2 Preparing Your Discs

As well as the Tak disc images from the discs directory of the project repository, you'll also need to download the original or patched language ROMs and discs.

The minimum needed to run the recreated Tak programs (without compiling or modifying them) is:

- BCPL, COMAL, FORTH, LISP, and micro-PROLOG each need a single ROM.
  - BCPL and LISP also have bug-fixed alternative ROMs available.
- Beebug C and ISO-Pascal require two ROMs.
  - Disc-based ISO-Pascal can be used instead.
- BASIC uses disc-based "Hi" languages.
- S-Pascal has no ROM; it needs its program disc.

To recompile source code for BCPL or Beebug C, files from their accompanying support disc are also needed.

Download links for ROMs and support discs are included in each language section and summarized in Appendix B. Each ROM should be downloaded using the suggested file name and added to the Tak disc, except for S-Pascal, where the Tak files should be added to the language disc instead.

There are a variety of host tools and utilities for manipulating Acorn DFS and ADFS disc images, though I tend to use Disc Image Manager.

2.1 Using ROMs

How you load a ROM image depends on your setup. If using an emulator, you might save the ROM file to your host computer (typically with a .rom extension) and load it directly through the emulator's interface.

Or load it from disc on a real or emulated machine. The following command works on a BBC Master, though your machine may require different ROM slot values (W and X in this example) or a different command altogether:

```
*SRLOAD R.[rom1] 8000 W Q
*SRLOAD R.[rom2] 8000 X Q // if using a second ROM
```

Once loaded, press Ctrl-Break to initialise the ROM.

2.2 Using Disc-Based Languages

Some languages, even if "Hi" ROMs, are stored on disc and loaded directly into second processor memory. These include HiBASIC (stored in the 'H' directory) and DPascal. They can simply be run using *RUN or */:

```
*/H.HiBasic  // on  OS 1.20
*/H.HiBas4   // on MOS 3.20
*DPascal
```

Generally, the "Hi" type languages require load and execution addresses to be set to &0000B800 before running them. The easiest way to do this is using Disc Image Manager, as there are no suitable BBC-based command-line tools.

2.3 File Names

Files and directories broadly follow the naming convention below and are grouped into three sections based on their purpose.

Running:	Bare minimum needed to re-run the tests and obtain timings.
Support:	Original article code, sizing comparison versions, and debugging tools.
Compiling:	Source code, make files, and language-specific components needed to re-build Tak from source.

Running

'R' directory:	Language ROMs to be loaded with *SRLOAD.
'H' directory:	"Hi" version of the language to be *RUN.
$.TAK:	The main Tak program.
$.TAKxx:	Variants of the Tak program (e.g., fp for floating point, scv for stored computed values, str for string, etc.).

Support Materials

!BOOT:	Sets MODE 3, enables paged mode, and runs *TYPE !ReadMe.
!ReadMe:	Points to this document and gives key running instructions.
'A' directory:	Copies of the code from the original article in plain text; use *TYPE to view the files.
'D' directory:	Code used for debugging.
'S' directory:	Used solely for Sizing purposes, along with files ending with "0" or "-".

Compiling

Language dirs:	The initial letter of a language is used for the source code where possible (e.g., B for BASIC and BCPL, P for Pascal, L for LISP).
Any other files:	Typically required by the language itself.

3 Assembler and BBC BASIC

3.1 Language Sources and Disc Merging

Download the files using the names given and add them to the Tak disc, TakBasicAsm.ssd.

TakBasicAsm.ssd: Language-specific Tak project disc from the GitHub repository
H.HiBasic: HiBASIC III is from the "HiBasic 3" link on mdfs or the ROM Library
H.HiBas4: HiBASIC 4 is on the BBC Master 65C102 Co-Processor Support Disc, under ESSENTIALS on stairwaytohell, or the ROM Library

Both "Hi" languages should have their load and execution addresses set to &0000B800.

Your merged disc will now contain:

Running
'H' directory: Contains "Hi" versions of BASIC (H.HiBasic, H.HiBas4).
TAKAsm: The Assembler version of the Tak function.
TAK: The original integer BASIC program.
TAKfp: A floating-point version of Tak.
TAKscv: The "stores computed values" version from Article 2.
TAKstr: The string-based version from Article 2. The result is the number of "X"s.

Support Materials
!ReadMe: Points to this document and gives key running instructions.
'A' directory: Plain text code from the article (A.TAKAsm, A.TAK, A.TAKscv, A.TAKstr).
'S' directory: Versions for size comparisons (S.TAK-, S.TAKfp-, S.TAKscv-, S.TAKstr-).

3.2 Running Tak in BBC BASIC or Assembler

To run Tak in BASIC or Assembler, ensure the second processor is active and then run the appropriate HiBasic:

```
for  OS 1.20 : */H.HiBasic
for MOS 3.20 : */H.HiBas4
for MOS 3.50 : not needed as BASIC 4r32 auto-relocates
```

Then, CHAIN the program you wish to run, eg:

```
CHAIN "TAK"
CHAIN "TAKfp"
CHAIN "TAKscv"
CHAIN "TAKstr"

CHAIN "TAKAsm"
```

3.3 Important Notes

- You must run HiBasic before running the stored-value version; otherwise, you will encounter a “No room” error.
- The run-time of TAKscv does not include pre-computing the stored values.

3.4 Program Sizes

The size of the Tak function is the difference between the base program’s size with the Tak function and without it (the “-” suffix variant).

Program	With Tak (&bytes)	Tak- (&bytes)	Size (bytes)
TAK	00A0	0053	77
TAKfp	0092	0053	63
TAKscv	015E	00B7	167
TAKstr	00CE	0072	92
TAKasm	n/a	n/a	231

As an alternative to sizing with files for BASIC: LOAD the program, delete non-function lines (e.g., DELETE 10,50 for TAK), then PRINT TOP-&802 to give the function size. This method matches file size results for all four BASIC programs.

4 BCPL (Basic Combined Programming Language)

4.1 Language Sources and Disc Merging

Download the files using the names given and add them to the Tak disc, TakBCPL.ssd.

Note:

- If using ADFS, use the patched version of the BCPL ROM instead of the original.
- The compiler utility is named "BCPL", so avoid using that name for any file in '$'.

TakBCPL.ssd:	Language-specific Tak project disc from the GitHub repository
R.BCPL:	BCPL.rom from BCPL_for_the_BBC_Microcomputer.zip at "BCPL for the BBC Microcomputer [Remastered PDF]" or the ROM Library
R.BCPLpat:	Faster, ADFS-aware patched BCPL ROM from pl7_patched.zip at "Sideways ROM software development" or the ROM Library

Your merged disc will now contain:

Running

'R' directory:	Contains BCPL ROMs (R.BCPL, R.BCPLpat).
TAK:	The compiled, runnable BCPL program.
TAKsa:	The stand-alone version that (should) be runnable outside of BCPL.

Support Materials

!ReadMe:	Points to this document and gives key running instructions.
A.TAK:	Plain text code from the articles for reference (may not run).
'S' directory:	Non-runnable variants for size comparisons (S.TAK0, S.TAK0-).

Compiling

NB: Additional files are required if compiling BCPL programs. See 4.5 Compiling for details.

'B' directory:	The BCPL source code files (B.TAK, B.TAK0, B.TAK0-, B.TAKsa).
E.mkTAK:	The "EX" script to make the various B.TAK* source files.
E.mkTAKsa:	The "EX" script to make the B.TAKsa source file.

4.2 Running Tak in BCPL

To run Tak in BCPL, ensure the R.BCPL ROM is loaded and the second processor is active, then type:

```
*BCPL
TAK
```

NB: OS '*' commands work as they do in BASIC.

4.3 Important Notes

BCPL doesn't seem to work with shadow screen modes.

4.4 Program Sizes

The size of the Tak function is the difference between the modified program's compiled CINTCODE size with the Tak function ("0" suffix variant) and without it ("0-" suffix variant).

Program	With Tak (&bytes)	Tak- (&bytes)	Size (bytes)
TAK	00D4	009A	58

4.5 Compiling

BCPL runs on ADFS but doesn't recognize it or its capabilities, so it falls back on slower OSBGET / OSBPUT calls. For best performance on ADFS, use the patched ROM (R.BCPLpat).

Download BCPL_for_the_BBC_Microcomputer.zip from "BCPL for the BBC Microcomputer [Remastered PDF]". Unzip the file and extract the BCPL disc images. Then add a minimal set of files from the BCPL discs to the Tak disc.

The 9 files from BCPL.ssd are:
BCPL* (BCPL, BCPLARG, BCPLCCG, BCPLSYN, BCPLTRN),
EX, LIB* (LIB, LIBHDR), NEEDCIN

The 4 files from BCPL_Stand_Alone_Generator.ssd are:
FIX* (FIXCIN, FIXINI), PACKCIN, SYSLIB1

4.5.1 Recompiling TAK

TAK, TAK0, and TAK0- files are supplied pre-compiled, but BCPL's EX automation script can be used to recompile any of them if needed. You will need to unlock the target TAK file in $ before compiling.

```
*BCPL
*ACCESS <file>
EX E.mkTAK [TAK | TAK0 | TAK0-]
```

The "EX" script E.mkTAK contains this code:

```
.KEY TFILE/A
DELETE /F.$.<TFILE>
BCPL /F.B.<TFILE> $TEMP1
NEEDCIN $TEMP1 LIB /F.$.<TFILE>
DELETE $TEMP1
```

NB: the /F. filename prefix tells BCPL to write to the current filing system rather than STORE, which acts as a ram disc.

4.5.2 Standalone Version

If using ADFS, the original FIXCIN utility sets the wrong execution address. Instead, download and use the patched version (FIXCIN2), which writes the correct address.

FIXCIN2: Available from fixcin2.zip at "BCPL Standalone Generator on ADFS?"

There are three key files for the stand-alone version:

B.TAKsa: The source code, with extra library to support standalone execution.
E.mkTAKsa: BCPL's "EX" automation script for the compile and link process.
TAKsa: The pre-compiled standalone executable, runnable with *RUN TAKsa.

The "EX" script E.mkTAKsa can be used to recompile TAKsa if needed, though it will need modifying to use FIXCIN2 if on ADFS. You will need to unlock $.TAKsa before compiling.

```
*BCPL
*ACCESS $.TAKsa
EX E.mkTAKsa
```

The "EX" script E.mkTAKsa contains this code.

```
.KEY TFILE
.DEF TFILE=TAKsa

DELETE /F.$.<TFILE>
BCPL /F.B.<TFILE> $TEMP1
NEEDCIN $TEMP1 SYSLIB1 $TEMP2
DELETE $TEMP1
PACKCIN $TEMP2 $TEMP3
DELETE $TEMP2
FIXCIN $TEMP3 $.<TFILE> GV=1900
DELETE $TEMP3
```

If using ADFS, change the penultimate line to call FIXCIN2 instead of FIXCIN to ensure the correct execution address is written. It will then look like this:

```
FIXCIN2 $TEMP3 $.<TFILE> GV=1900
```

NB: I've never actually been able to get the resulting standalone code to run to completion.

5 BCPL: ADFS v DFS Performance

This is a filing system (FS) test, comparing BCPL compile times on DFS vs ADFS. It is not a language performance benchmark and can be run (albeit slowly) without a second processor.

Although built on a Master, it should run on a BBC B with a working *TIME command.

5.1 Assemble the FS Test Discs

Download the files using the names given below, and add them to the DFS-format FS test disc, BCPL_FSTest.ssd. An ADFS-format disc will be created from it later.

NB: To simplify file access on ADFS, directories have been avoided, and files are stored in a flat structure using a single-letter prefix (eg rBCPL for the ROM).

BCPL_FSTest.ssd: DFS-format FS test disc from the GitHub repository
rBCPL: BCPL.rom from BCPL_for_the_BBC_Microcomputer.zip at "BCPL for the BBC Microcomputer [Remastered PDF]" or the ROM Library
rBCPLpt: Faster, ADFS-aware patched BCPL ROM from pl7_patched.zip at "Sideways ROM software development" or the ROM Library

From BCPL_for_the_BBC_Microcomputer.zip also extract the BCPL disc images. Then add a minimal set of files from the BCPL discs to the FS test disc.

The 9 files from BCPL.ssd are:
BCPL* (BCPL, BCPLARG, BCPLCCG, BCPLSYN, BCPLTRN),
EX, LIB* (LIB, LIBHDR), NEEDCIN

The 4 files from BCPL_Stand_Alone_Generator.ssd are:
FIX* (FIXCIN, FIXINI), PACKCIN, SYSLIB1

Your merged disc will now contain:

rBCPL: The original BCPL ROM image.
rBCPLpt: The patched BCPL ROM image for improved ADFS performance.
bTAKsa: The source code, with extra library to support standalone execution.
mkTAKsa: The "EX" script to time and compile bTAKsa.
!ReadMe: Points to this document and gives key running instructions.
Other files: Remaining 13 files in $ are needed by BCPL or the SAG.

Once the DFS disc is assembled and saved, make an ADFS copy:

1. Use DIM to create a blank ADFS "L" disc image.
2. Drag the DFS image into the ADFS image to import all the files.
3. Save the new ADFS disc as BCPL_FSTest.adl.

5.2 Running The FS Tests

There are four test combinations available, combining the two ROM versions (original and patched) with the two filing systems (DFS and ADFS).

Ensure the second processor is running, mount either the DFS or ADFS floppy disc image, then use *SRLOAD to load either (but not both!) the original or patched BCPL ROM:

```
*SRLOAD rBCPL   8000 W Q // either original
*SRLOAD rBCPLpt 8000 W Q // or patched
```

Then start BCPL and use “EX” to run the scripted compile process:

```
*BCPL
EX mkTAKsa
```

The “EX” make script uses the /F. filename prefix to ensure all files are written to disc rather than the faster memory-based STORE. This would not normally be done but helps exaggerate the impact of disc performance differences.

When the script finishes, you will need to manually calculate the elapsed time by finding the difference between the two *TIME results.

An example DFS session output is given below (note, this was tested in 2025, not 1925 as the log suggests!)

```
!EX ETAKSA
!
!*TIME
Wed,12 Feb 1925.13:17:47
!DELETE /F.TAKsa
!BCPL /F.bTAKsa /F.xTEMP1 REPORT /N
!NEEDCIN /F.xTEMP1 SYSLIB1 /F.xTEMP2
1817 word file /F.xTEMP2 created
0 NEED(S) unsatisfied
!DELETE /F.xTEMP1
!PACKCIN /F.xTEMP2 /F.xTEMP3 REPORT /N
!DELETE /F.xTEMP2
!FIXCIN /F.xTEMP3 TAKsa GV=1900 REPORT /N
!DELETE /F.xTEMP3
!*TIME
Wed,12 Feb 1925.13:20:15
!
EX File Terminated
```

The run time was 13:20:15 - 13:17:47, which is 148 seconds.

Note: As mentioned in the previous section, ADFS writes the incorrect execution address for standalone files. Use the patched FIXCIN2 utility to correct this.

6 Beebug C

6.1 Language Sources and Disc Merging

Download the files using the names given and add them to the Tak disc, TakBeebugC.ssd.

TakBeebugC.ssd: Language-specific Tak project disc from the GitHub repository

BeebugC-1.5.zip: From the thread "Beebug C"
R.C-COMP: beebugC-com.rom from BeebugC-1.5.zip or the ROM Library
R.C-LANG: beebugC-lang.rom from BeebugC-1.5.zip or the ROM Library

BC15LibReco.ssd:The reconstructed v1.5 Library disc, from the thread "Beebug C".
Add a minimal set of files from BC15LibReco.ssd to the Tak disc.
CSP, rtlib, h.stdlib, h.stdio

Your merged disc will now contain:

Running
'R' directory: Contains Beebug C ROMs (R.C-COMP, R.C-LANG)
E.TAK: The compiled, executable C program.
CSP: The C Second Processor support utility.

Support Materials
!ReadMe: Points to this document and gives key running instructions.
'E' directory: Executable files (E.TAK, E.TAK0, E.TAK0-).
A.Z80SmC: Z80 Small-C code from the article for reference.
*0, *0-: Non-runnable variants for size comparisons.

Compiling
'C' directory: TAK C source files (C.TAK, C.TAK0, C.TAK0-).
'O' directory: Compiled TAK object files (O.TAK, O.TAK0, O.TAK0-).
'H' directory: Header files (h.stdlin, h.stdio).
rtlib: The run-time library.

6.2 Running Tak in Beebug C

To run Tak in Beebug C, ensure both the R.C-LANG and R.C-COMP ROMs are loaded and the second processor is active, then type:

```
*FX142,76 // only if running MOS 3.50

*CSP
*C
RUN TAK
```

NB: OS '*' commands work as they do in BASIC.

6.3 Important Notes

- Adapted from the Z80 Small-C code that was included in the second article.
- Code to print the run time has been added.
- To optimize code size and performance, the following options were used:
  - COMPILE /OPTIMISE /NODEBUG
  - LINK /NODEBUG
- MOS 3.50: *CSP doesn't work unless BASIC's auto-relocation is disabled. Find BASIC's ROM slot, add 64 (usually 12 + 64 = 76), then type *FX142,76 before running *CSP.

6.4 Program Sizes

The size of the Tak function is the difference between the modified program's compiled CINTCODE size with the Tak function ("0" suffix variant) and without it ("0-" suffix variant).

Program	With Tak (&bytes)	Tak- (&bytes)	Size (bytes)
TAK	01AB	0122	137

6.5 Compiling

Warning! In MOS 3.20, the default DFS 2.24 can write file sizes incorrectly. As a result, COMPILE creates a 16KB file (&4000 bytes), and LINK then fails. To resolve this, load DFS 2.29 from the Welcome disc and unplug DFS 2.24 before compiling.

R.DFS2,29: $.LIBRARY.DFS2,29 from "Master Welcome Disc ADFS.adl" at huininga or the ROM Library

- If using MOS 3.20, ensure DFS 2.29 is the active filing system.
- If using MOS 3.50 with a second processor, first issue the workaround (*FX142,76).
- If not using a second processor, just start at *C, without *CSP.

Ensure E.TAK and O.TAK are not locked; the compile process is then:

```
*ACCESS *.TAK
*FX142,76     // if using a second processor and MOS 3.50
*CSP          // if using a second processor
*C
COMPILE /OPTIMISE /NODEBUG TAK
LINK /NODEBUG TAK
```

7 COMAL (Common Algorithmic Language)

7.1 Language Sources and Disc Merging

Download the files using the names given, and add them to the Tak disc, TakCOMAL.ssd.

TakCOMAL.ssd:	Language-specific Tak project disc from the GitHub repository
R.COMAL:	COMAL.rom from "COMAL on the BBC Microcomputer [Remastered PDF]" or the ROM Library

Your merged disc will now contain:

Running

'R' directory:	Contains the COMAL ROM (R.COMAL).
TAK:	the integer program.
TAKfp:	Floating-point version of TAK

Support Materials

!ReadMe:	Points to this document and gives key running instructions.
D.VarSpd:	Tests timing of FOR loops and FUNC calls with different variable types.
D.TAKffp:	Test variant of TAKfp, with 3-character variable names instead of 2.
'S' directory:	Versions for size comparisons (S.TAK-, S.TAKfp-).

7.2 Running Tak in COMAL

To run Tak in COMAL, ensure the R.COMAL ROM is loaded and the second processor is active, then RUN the appropriate program:

```
*COMAL

RUN "TAK"
RUN "TAKfp"
```

NB: OS '*' commands work as they do in BASIC.

7.3 Important Notes

- Help adapting BASIC was given in "Comal from Basic, Recursive Variable Scope?".
- A modern "Hi" version is available from "Making a co-pro non-Hi Language, Hi" or GitHub, but it was not used in these tests.

7.4 Program Sizes

The size of the Tak function is the difference between the base program's size with the Tak function and without it (the "-" suffix variant).

Program	With Tak (&bytes)	Tak- (&bytes)	Size (bytes)
TAK	00FE	005A	164
TAKfp	00E9	005A	143

As an alternative to sizing with files, LOAD the program, delete non-function lines (e.g., DEL 10,50 for TAK), then PRINT SIZE-2 to give the function size. This method matches file size results for both programs.

7.5 Debug Versions

Unusually, TAKfp (floating-point) ran faster than TAK (integer). To investigate, two test programs were created and discussed in "COMAL FP Faster than Integer?". The results below are from MAME emulating a Master 128 "Turbo" with MOS 3.20.

D.TAKffp tests whether variable name length affects run time. It's based on TAKfp, with variable names extended from 2 to 3 characters. Run times, in seconds:

- TAK (integer) 622
- TAKfp 615
- TAKffp 630

D.VarSpd reports run times for different operations over 6000 iterations across 4 test groups:

- Simple FOR loops using Resident Integer, Integer, and Float variables.
- Calling a FUNC which returns the called value with Integer, and Float parameters and corresponding CLOSED versions (similar to LOCAL in BASIC).
- Calling a FUNC with either a long name / Integer or short name / Float.
- Recursive PROC with Integer, and Float parameters.

Example output from VarSpd (timings in centiseconds):

```
]RUN "D.VarSpd"
ResInt  554
Integer 673
Float   753
//
Int FN:        2334
Float FN:      2345
Int CLOSED:    2405
Float CLOSED: 2416
//
Long Int FN:    3333
Short Float FN: 2275
//
Int Recursive:   3972
Float Recursive: 3994
```

Of particular interest is the difference in run times between the tests with long and short FUNC names: 'Long Int FN' and 'Short Float FN'.

8 FORTH

8.1 Language Sources and Disc Merging

Download the files using the names given, and add them to the Tak disc, TakFORTH.ssd.

TakFORTH.ssd:	Language-specific Tak project disc from the GitHub repository
R. FORTH:	FORTH_103.rom from "FORTH on the BBC Microcomputer [Remastered PDF]" or the ROM Library

Your merged disc will now contain:

Running

'R' directory:	Contains the FORTH ROM (R.FORTH).
F.TAK:	EXEC script to load and calculate Tak function size, run and time it.

Support Materials

!ReadMe:	Points to this document and gives key running instructions.
A.TAK:	Plain text code from the articles for reference (may not run).

8.2 Running Tak in FORTH

To run Tak in FORTH, ensure the R.FORTH ROM is loaded and the second processor is active, then EXEC the program:

```
*FORTH
C        // COLD start

OS' EXEC F.TAK
```

As a by-product of using *EXEC to "type in" the word definitions, FORTH commands can also be executed. By running HERE (which returns the address of the first free byte in the dictionary) both before and after the Tak definition, the difference gives us the Tak function's size, which is then printed at the end of the code:

```
OK
SWAP - . ." bytes" CR
58 bytes
OK
```

Now, start Tak running:

```
TEST
```

NB: OS ‘*’ commands are introduced with OS' [space], e.g.:

```
OS' BASIC
OS' INFO R.*
```

8.3 Important Notes

FORTH saves to equal-sized nSCREEN files on-disc and lacks direct access to timing functions, needing a different approach than other languages. Struggling with the saving process, I chose to EXEC the word definitions into FORTH, which, by chance, also simplified code size measurement.

FORTH does not have direct access to the system timer, so custom words have been created to use OSWORD 1 and OSWORD 2 to read and set it.

- ZTIME pushes three zeros (total 48 bits) onto the stack, then calls
- (ZTIME), which invokes OSWORD 2 to set the system timer using the zeros on the stack and resets the stack pointer on exit.
- TIME pushes three zeros (total 48 bits) onto the stack, then calls
- (TIME), which invokes OSWORD 1 to read the current time into the stack. Since the returned time is 40 bits and FORTH works with 32-bit values, the top words are discarded, and the two remaining items are swapped to ensure “D.” displays the values correctly.

In summary, the EXEC script contains the original Tak definition (surrounded by HERE to measure its size), the new time-handling words, and a modified TEST function that calls these timing words. When you EXEC the script, it defines the words and shows the Tak function size. Running TEST will display the Tak result and time in centiseconds.

9 ISO-Pascal

9.1 Language Sources and Disc Merging

Download the files using the names given, and add them to the Tak disc, TakIsoPascal.ssd.

TakIsoPascal.ssd: Language-specific Tak project disc from the GitHub repository
isopascal.zip: Zipped v1.1 ROMS from the thread "ISO Pascal ROMs - not working?"

R.PASCAL1: ISO-Pascal1-1.1.rom from isopascal.zip or the ROM Library
R.PASCAL2: ISO-Pascal2-1.1.rom from isopascal.zip or the ROM Library

Your merged disc will now contain:

Running
'R' directory: Contains the ISO-Pascal ROMs (R.PASCAL1, R.PASCAL2).
TAK: Tak function (integer) Pascal source.
TAKfp: Floating-point version of TAK.

Support Materials
!ReadMe: Points to this document and gives key running instructions.
A.TAK: Plain text code from the articles for reference (may not run).
'S' directory: Versions for size comparisons.

Compiling
'P' directory: Pascal sources of both runnable and sizing variants.

9.2 Running Tak in ISO-Pascal

To run Tak in ISO-Pascal, ensure both the R.PASCAL1 and R.PASCAL2 ROMs are loaded and the second processor is active, then type:

```
*PASCAL
TAK (or TAKfp)
```

NB: OS '*' commands work as they do in BASIC.

9.3 Important Notes

The original article mentioned floating-point results but included no code, so the integer version was modified by declaring x, y, z as real and starting with values n.0.

9.4 Program Sizes

The size of the Tak function is the difference between the modified program's compiled BL-Code size with the Tak function ("0" suffix variant) and without it ("0-" suffix variant).

Program	With Tak (&bytes)	Tak- (&bytes)	Size (bytes)
TAK	0089	0042	71
TAKfp	0090	0046	74

9.5 Compiling

You will need to unlock the appropriate target TAK files before compiling:

```
*ACCESS $.TAK*
*ACCESS S.TAK*
```

For the ROM-based version, compile with:

```
*PASCAL
COMPILE P.TAK TAK
COMPILE P.TAKfp TAKfp
COMPILE P.TAK0 S.TAK0
COMPILE P.TAK0- S.TAK0-
COMPILE P.TAKfp0 S.TAKfp0
COMPILE P.TAKfp0- S.TAKfp0-
```

9.6 Disc-Based Version

The ROM-based language is certified as ISO Level Zero, but an alternative ISO Level One version was also available. Like a "Hi" language, it requires no ROMs, is disc-based, and needs a second processor. Although not used in these tests, it is noted here for completeness.

- The latest versions of DPascal and DComp (the disc-based language and compiler) are available at gtoal's website.
- Important: Set DPascal's load and execution addresses to &0000B800 before use.
- Use *DPASCAL and DCOMP instead of *PASCAL and COMPILE.

```
*/DPASCAL
DCOMP P.TAK TAK
DCOMP P.TAKfp TAKfp
DCOMP P.TAK0 S.TAK0
DCOMP P.TAK0- S.TAK0-
DCOMP P.TAKfp0 S.TAKfp0
DCOMP P.TAKfp0- S.TAKfp0-
```

10 LISP (List Processing)

10.1 Language Sources

Download the files using the names given and add them to the Tak disc, TakLISP.ssd.

TakLISP.ssd:	Language-specific Tak project disc from the GitHub repository
R.LISP500:	LISP-500.rom (standard version 5) from LISP_roms.zip at "LISP on the BBC Microcomputer [Remastered PDF]" or the ROM Library
R.LISP407:	Patched LISP v4 ROM at https://www.cogsci.ed.ac.uk/~richard/LISP407esc2 linked from "LISP: Slower with with Co-Pros and MOS 3.50"or the ROM Library

Your merged disc will now contain:

Running

'R' directory:	Contains LISP ROMs (R.LISP500, R.LISP407).
TAK:	The LOADable LISP program.

Support Materials

!ReadMe:	Points to this document and gives key running instructions.
A.TAK:	Plain text code from the articles for reference (may not run).
L.TAK:	Plain-text LISP source code, ready to be *EXEC'd into LISP.
L.TAK-:	A sizing version of L.TAK without the tak() function.
L.TAKfn:	Just the Tak function in isolation, also for calculating sizes.
'S' directory:	Non-runnable variant for size comparison (S.TAK-).

10.2 Running Tak in LISP

To run Tak in LISP, ensure either the R.LISP500 (original) or R.LISP407 (patched) ROMs are loaded and the second processor is active, then type:

```
*LISP
(LOAD 'TAK)
(TEST)
```

The output includes two times in centiseconds: elapsed time and garbage collection time, eg:

```
7 // The result of the calculation
(-32478 584)

Value is : (-32478 584)
```

As LISP uses 16-bit signed integers, times above 32767 (around 5½ minutes) wrap into negative values. These can be converted to a positive by adding 65536, giving a run time of 33058 cs (5 minutes, 31 seconds) for this example.

NB: OS '*' commands are like LOAD; they have brackets and a leading single quote, eg:

```
(* 'BASIC)
```

For OS commands with special characters (e.g., spaces, dots), prefix each with a '!', eg:

```
(* 'INFO! L!.TAK)
```

10.3 Program Sizes

The size of the Tak function is the difference between the base program's size with the Tak function and without it (the "-" suffix variant).

Program	With Tak (&bytes)	Tak- (&bytes)	Size (bytes)
TAK	1042	0F76	204

An alternative to sizing with files in LISP is to increase the debugging level and force a garbage collection before and after loading just the Tak function, calculating the difference between the two "Bytes free" values:

```
(MESSON 1)          // Enable memory debug message
(MODE 4)
(RECLAIM)           // Garbage collect and print free memory
(* 'EXEC! L!.TAKfn) // Load just the Tak function
(RECLAIM)           // Print free memory
```

This method produces a slightly larger calculated size: 48653 - 48440 = 213 bytes.

11 micro-PROLOG (**Pro**grammation en **log**ique)

11.1 Language Sources and Disc Merging

Download the files using the names given and add them to the Tak disc, TakmProlog.ssd.

TakmProlog.ssd: Language-specific Tak project disc from the GitHub repository
R.PROLOG: micro-PROLOG ROM from the ROM Library

Your merged disc will now contain:

Running
'R' directory: Contains micro-PROLOG ROM (R.PROLOG).
TAK: The main Tak program.
TAKacl: Tak with ADDCL clause.

Support Materials
!ReadMe: Points to this document and gives key running instructions.
'A' directory: Plain text code from the article (A.TAK, A.TAKacl).
'S' directory: Versions for size comparisons (S.TAK-, S.TAKacl-).
'T' directory: Source code text files, ready to be EXEC'd into micro-PROLOG.

If needed, the entire micro-PROLOG package is available on the Flaxcottage Educational Software Archive website.

1. Type the keyword "prolog" in the SEARCH area.
2. Press the [Package Title] button below it.
3. On the results page, click the "Micro PROLOG" link below the box cover image.
4. On the item's summary page, scroll to the bottom and click DOWNLOAD.
5. The ROM, if needed, is $.PROLOG on MicroProlog.ssd.

11.2 Running Tak in micro-PROLOG

To run Tak in micro-PROLOG, ensure the R.PROLOG ROM is loaded and the second processor is active. micro-PROLOG is case-sensitive, so type the following exactly, including file names:

```
*PROLOG
LOAD TAK
?((test))
```

Before running another program, clear all memory with KILL ALL (similar to BASIC's NEW):

```
KILL ALL
LOAD TAKacl
?((test))
```

NB: OS '*' commands require two asterisks, a space, and the command, e.g.,

```
** BASIC
```

If the command contains spaces, enclose it all in quotes, e.g.,

```
** "INFO TAK"
```

11.3 Important Notes

- micro-PROLOG is case-sensitive and can be temperamental, sometimes behaving inconsistently depending on previous actions. If you encounter issues, try:
  - Ensure the command is typed as printed, with the exact case and spacing.
  - If the issue persists, run "KILL ALL" to clear memory and then try again.
  - If the problem continues, restart the system and try the command once more.
- Remember, you must run "KILL ALL" between different programs.

11.4 Program Sizes

The size of the Tak function is the difference between the base program's size with the Tak function and without it (the "-" suffix variant).

Program	With Tak (&bytes)	Tak- (&bytes)	Size (bytes)
TAK	00F7	0049	174
TAKacl	0114	0049	203

12 S-Pascal

12.1 Language Sources and Disc Merging

S-Pascal runs on top of BASIC and even stores its Pascal programs as BASIC. Because of this, the S-Pascal disc should be used as the basis for the merged disc, rather than the Tak disc.

S-Pascal.SSD: The S-Pascal disc from S-Pascal.zip from thread "S-Pascal".
TakSPascal.ssd: Language-specific Tak project disc from the GitHub repository

Open S-Pascal.SSD, delete all files in the 'E' directory, then add all files from TakSPascal.ssd. Finally, add:

$.HiBasic: HiBASIC III is from the "HiBasic 3" link on mdfs or the ROM Library. Set its load and execution addresses set to &0000B800.

Your merged disc will now contain:

Running
HiBasic: HiBasic III, automatically loaded if present.
EXKEYS: An EXEC file that sets up helper function keys.
B.TAK: The main TAK program.

Support Materials
!ReadMe: Points to this document and gives key running instructions.
TAK: The code after compilation. Can be *RUN without S-Pascal loaded.
'B' directory: Contains S-Pascal programs saved from BASIC (B.TAK, B.TAK0, B.TAK0-).
S.PASCAL: Required by S-Pascal.
'S' directory: Non-runnable size comparison variants (S.TAK0, S.TAK0-).

12.2 Running Tak in S-Pascal

To run Tak in S-Pascal, ensure the second processor is active, then boot the disc and type:

```
[Shift-Break] // Boot the disc

*EXEC EXKEYS  // Sets up helper function keys
LOAD "B.TAK"
*COMPILE
```

Warning: f0 is used by S-Pascal – don't press it!

After the assembler part of the compilation finishes, you must press a SHIFT key to list the program and exit the compile process. You'll then be returned to the ">" prompt.

Normally, you would type *GO to run the machine code in memory. However, since S-Pascal cannot access TIME, use the function keys to run it and capture timing data:

- Press f1 to store TIME, execute *GO and run TAK.
- Immediately press f2 to display the elapsed time upon completion.

NB: OS '*' commands work as they do in BASIC.

12.3 Important Notes

- Be cautious when saving any files using the 8271 DFS. I encountered an issue with the disc corruption when saving files to the S-Pascal disc with the 8271 DFS on MAME, though the root cause is unknown.
- S-Pascal runs on top of BASIC, so all usual BASIC commands are available.

12.4 Program Sizes

The size of the Tak function is the difference between the modified program's compiled code size with the Tak function ("0" suffix variant) and without it ("0-" suffix variant).

Program	With Tak (&bytes)	Tak- (&bytes)	Size (bytes)
TAK	01BB	0013	424

12.5 Saving and Running the Code

Saving, running and timing the code will be eased by setting up predefined function keys using *EXEC $.EXKEYS. The function keys are then defined as follows:

f1: Captures TIME in T% and runs the compiled code with *GO.
f2: Displays elapsed time by printing TIME - T%.
f3: Captures TIME in T% and runs pre-compiled code with */$.TAK.
f4: Saves the compiled code from memory to disk.

As S-Pascal runs on top of BASIC, P% points to the byte after the last one assembled when *COMPILE finishes. The code is stored in memory starting from address &1F00.

It can be saved by pressing f4, or manually using:

```
OSCLI "SAVE $.TAK 1F00 " + STR$ ~(P%-1) + " 1F00"
```

The saved code can be run with timing by pressing f3, then immediately pressing f2. Or run it as normal using:

```
*/TAK
```

Appendix A Original Source Code

The original code from the articles, exactly as printed, is included here and arranged in alphabetical order (excepting Assembler, which is relatively long, and is moved to the end). The same code is also available on each relevant language's disc in the 'A' directory.

Disclaimer: The code in this appendix is not my original work. It is reproduced here from the published articles for educational and historical purposes. All rights remain with their respective owners. If any rights holders have concerns about its inclusion, please contact me to discuss removal or further permissions.

Click on a language name to jump to:

A.1 BASIC

Integer BASIC:

```
   10 REM TAK FUNCTION
   30 TIME=0
   35 PRINT"FNT(";A;",";B;",";C;")=";FNT(A,B,C)
   40 PRINTTIME/100;" SECS."
   50 END
   60 DEFFNT(X%,Y%,Z%)IFX%<=Y%=Z%ELSE=
FNT(FNT(X%-1,Y%,Z%),FNT(Y%-1,Z%,X%),FNT(Z%-1,X%,Y%))
```

BASIC "Stores Computed Values" (scv):

```
   10 REM TABLE TAK FUNCTION
   20 DIM K%(18,18,18)
   30 FOR L%=0 TO 18
   40 FOR M%=0 TO 18
   50 FOR N%=0 TO 18
   60 K%(L%,M%,N%)=-1
   70 NEXT: NEXT: NEXT
   80 TIME=0
   90 PRINT"FNT(18,12,6)=";FNT(18,12,6)
  100 PRINTTIME/100;" SECS."
  110 END
  120 DEFFNT(X%,Y%,Z%)
  130 IF K%(X%,Y%,Z%)<>-1 =K%(X%,Y%,Z%)
  140 IFX%<=Y% K%(X%,Y%,Z%)=Z%: =Z%
  150 K%(X%,Y%,Z%)=FNT(FNT(X%-1,Y%,Z%),FNT(Y%-1,Z%,X%),FNT(Z%-1,X%,Y%))
  160 =K%(X%,Y%,Z%)
```

BASIC Strings (str):

```
   10 REM STRING TAK FUNCTION
   30 TIME=0
   35 PRINT FNT("XXXXXXXXXXXXXXXXXX", "XXXXXXXXXXXX", "XXXXXX")
   40 PRINTTIME/100;" SECS."
   50 END
   60 DEF FNT(X$,Y$,Z$)IF LEN(X$)<=LEN(Y$) =Z$ ELSE=
FNT(FNT(MID$(X$,2),Y$,Z$),FNT(MID$(Y$,2),Z$,X$),FNT(MID$(Z$,2),X$,Y$))
```

A.2 BCPL

```
SECTION "PROGA"
GET "LIBHDR"

LET start() BE
$( LET t,int = ?,?
   t := TIME()
   WRITEF( "tak(18,12,6)= %N*N", tak(18,12,6) )
   WRITEF( "Time= %N*N", TIME()-t )
$)

AND tak(x,y,z) = y<x -> tak(tak(x-1,y,z),tak(y-1,z,x),tak(z-1,x,y)),z
```

A.3 Beebug C

Derived from the Small-C original:

```
main()
{
   puts("TAK(18,12,6) = ");
   printd(tak(18,12,6));
}

tak(x, y, z)
int x, y, z;
{
   if (y >= x) return (z);
   else
      return tak(tak(x-1, y, z), tak(y-1, z, x), tak(z-1, x, y));
}

printd(n) /* print n in decimal (recursive) */
int n;
{
   int i;
   if (n < 0) {puts("-"); n = 0 - n; }
   if ((i = n/10) != 0) printd(i);
   putch(n % 10 + '0');
```

A.4 COMAL

Derived from the BASIC equivalents.

A.5 FORTH

```
0 ( TAK FUNCTION )
1 R: TAK ( Z Y X )
2 2DUP <
3 IF DUP 2OVER DUP 2OVER 1- TAK
4    >R 1- TAK
5    >R 1- TAK
6    R> R> SWAP ROT TAK
7 ELSE 2DROP
8 THEN R;
9
10 : TEST 6 12 18 TAK . ;
```

A.6 ISO-Pascal

```
{$D-,R-,U-}
{Tak function}
program taktest (input,output);

function tak(x,y,z:integer): integer;
begin
   if y<x then tak:=tak( tak(x-1,y,z), tak(y-1,z,x), tak(z-1,x,y) )
          else tak:=z
end;

{main program}

begin
   settime(0);
   write('tak=',tak(18,12,6),' time=',time);
end.
```

A.7 LISP

```
(DEFUN TAK (X Y Z)
   (COND
      ((LESSP Y X)
         (TAK
            (TAK (SUB1 X) Y Z)
            (TAK (SUB1 Y) Z X)
            (TAK (SUB1 Z) X Y)))
      (T Z)))

(DEFUN TEST ()
   (RESET)
   (PRINT (TAK 18 12 6))
   (PRINT (LIST (TIME) (GCTIME))))
```

A.8 micro-PROLOG

Standard micro-PROLOG:

```
((tak X Y Z x)
  (LESS Y X)
  (SUM x1 1 X)
  (SUM y1 1 Y)
  (SUM z1 1 Z)
  /
  (tak x1 Y Z x2)
  (tak y1 Z X y2)
  (tak z1 X Y z2)
  (tak x2 y2 z2 x))

((tak X Y Z Z))

((test)
  (TIME 0)
  (tak 18 12 6 y)
  (TIME z)
  (PP y)
  (PP z))
```

micro-PROLOG ADDCL (acl)

```
((tak X Y Z x)
  (LESS Y X)
  (SUM x1 1 X)
  (SUM y1 1 Y)
  (SUM z1 1 Z)
  /
  (tak x1 Y Z x2)
  (tak y1 Z X y2)
  (tak z1 X Y z2)
  (tak x2 y2 z2 x)
  (ADDCL ((tak X Y Z x)) 0))

((tak X Y Z Z))

((call X Y Z x)
  (TIME 0)
  (X Y Z x y)
  (TIME z)
  (PP y)
  (PP z))
```

A.9 S-Pascal

Derived from ISO-Pascal.

A.10 Assembler

```
 10 REM MACHINE-CODE VERSION OF TAK FUNCTION
 20 DIM code% 1000, stack% 1000
 30 x%=0:y%=1:z%=2: REM parameter stack positions
 40 GOTO230
 50 REM Define Macros
 60 DEF FNhi(i%)=stack%+1+2*i%
 70 DEF FNlo(i%)=stack%+2*i%
 80 DEF FNdec(i%) REM decrements word i% on stack
 90 [     LDA FNlo(i%),X
100       SEC
110       SBC #1
120       STA FNlo(i%),X
130       LDA FNhi(i%),X
140       SBC #0
150       STA FNhi(i%),X
160 ]: =pass
170 DEF FNmov(i%,j%) REM copies word i% to j% on stack
180 [     LDA FNlo(i%),X
190       STA FNlo(j%),X
200       LDA FNhi(i%),X
210       STA FNhi(j%),X
220 ]: =pass
230 REM Main Program
240 FOR pass=1 TO 3 STEP 2
250 P%=code%
260 [     OPT pass
270 .tak
280 \ on entry X-reg points to parameters
290 \ on exit X-reg points to result
300       SEC
310       LDA FNlo(y%),X
320       SBC FNlo(x%),X
330       LDA FNhi(y%),X
340       SBC FNhi(x%),X
350 \ Carry clear if x%>y%
360       BCC else
370       OPT FNmov(z%,0)
380       RTS
390 .else INX: INX: INX: INX: INX: INX
400       OPT FNmov(-3,x%)
410       OPT FNmov(-2,y%)
420       OPT FNmov(-1,z%)
430       OPT FNdec(x%)
440       JSR tak
450       INX: INX
460       OPT FNmov(-3,x%)
470       OPT FNmov(-2,y%)
480       OPT FNmov(-4,z%)
490       OPT FNdec(x%)
500       JSR tak
510       INX: INX
520       OPT FNmov(-3,x%)
530       OPT FNmov(-5,y%)
540       OPT FNmov(-4,z%)
550       OPT FNdec(x%)
560       JSR tak
570       DEX: DEX: DEX: DEX
580       JSR tak
590       OPT FNmov(0,-3)
600       DEX: DEX: DEX: DEX: DEX: DEX
610       RTS
620 ]: NEXT
630 REM Call routine
640 !stack%=18: stack%!2 = 12: stack%!4 = 6
650 X%=0: TIME=0: CALL tak
660 PRINT"TAK(18,12,6)="; !stack% AND &FFFF ; " Time="; TIME
```

Appendix B Languages, patches and documentation

Links to all the source materials, patched software and manuals.

Correct at the time of writing, March 2025.

Project Repository

- The Tak project files, disc images, and documentation
  https://github.com/acheton1984/ReTestingTheTak

The BBC Micro ROM Library

- Which “aspires to list, identify and supply all ROMs available for the BBC Micro”
  https://tobylobster.github.io/rom_library/

Disc Image Manager

- The indispensable tool for managing disc images on Mac, Linux or Windows
  https://www.geraldholdsworth.co.uk/index.php?link=DiscImageReader

BCPL

- “Remastered” software and documentation
  https://www.stardot.org.uk/forums/viewtopic.php?p=298394
- Patched ROM with ADFS support
  https://stardot.org.uk/forums/viewtopic.php?p=355455#p355455
- Patched FIXCIN with ADFS support
  https://www.stardot.org.uk/forums/viewtopic.php?p=433622#p433622

Beebug C

- ROMS
  https://stardot.org.uk/forums/viewtopic.php?p=136335#p136335
- Disc images
  https://www.stardot.org.uk/forums/viewtopic.php?p=432462#p432462
- Manuals
  https://stardot.org.uk/forums/viewtopic.php?p=341803#p341803

COMAL

- “Remastered” software and documentation
  https://www.stardot.org.uk/forums/viewtopic.php?t=20910
- “Hi” COMAL
  https://stardot.org.uk/forums/viewtopic.php?p=371150#p371150

FORTH

- “Remastered” software and documentation
  https://www.stardot.org.uk/forums/viewtopic.php?t=20292

ISO-Pascal

- ROMS
  https://www.stardot.org.uk/forums/viewtopic.php?p=350313#p350313
- Updated Dpascal and Dcomp utilities
  https://gtoal.com/acorn/arm/TutuPascal/BBCPascal
- Manuals
  https://www.stardot.org.uk/forums/viewtopic.php?p=370334#p370334
- List of ISO-Pascal links
  https://www.stardot.org.uk/forums/viewtopic.php?t=27469

LISP

- “Remastered” software and documentation
  https://www.stardot.org.uk/forums/viewtopic.php?t=17811
- Patched (faster) ROM
  https://www.stardot.org.uk/forums/viewtopic.php?p=439985#p439985
- Reassembly of LISP
  https://github.com/stardot/AcornsoftLISP

micro-PROLOG

- Enter keyword “prolog”, press [Program Title], click “Micro PROLOG”
  https://www.flaxcottage.com/Educational/Default.asp
- “Hi” micro-PROLOG, “LPA__HI-micro_PROLOG__3.1 “ at
  https://acorn.huininga.nl/pub/unsorted/roms/

S-Pascal

- Disc image
  https://stardot.org.uk/forums/viewtopic.php?p=279871#p279871
- Manual (search down the page for “SLD14”, download the PDF)
  http://8bs.com/othrdnld/manuals/applications.shtml