Source Code for Version 8 of Icon for the Atari ST

Ralph E. Griswold

April 2, 1990

Introduction

Version 8 of Icon compiles using the Lattice 68000 C compiler, Version 6.04. There are hooks for the Mark Williams C compiler, but not all the configuration information is in place yet.

The enclosed distribution diskette contains source code for Version 8 of the Icon programming language [1,2], configured to compile on the Atari ST. It also contains documentation, some tools to assist in the implementation — ASH, a command-line processor and arc.prg, a program for extracting files from ARC format.

If you do not have arc.prg installed already, copy it from the distribution diskette to the folder in which you plan to work.

If you do not have ASH installed or are not familiar with its use, copy ash.prg and ash.ini from the distribution diskette to the folder in which you plan to work. Read ash.doc and ash.hlp on the distribution diskette for more information. Start up ASH and proceed.

Printed documentation comes with diskettes provided by the Icon Project at the University of Arizona. If you got your Icon diskette from another source without printed copies of the documentation, dearchive the documents. Assuming the distribution diskette is in drive a:, do

```
arc x a:docs.arc
```

Then print the files that are produced.

In order to unload and compile Icon, you need plenty of disk space, and the resources to create at least seven new folders.

Unloading Icon Source Code

Create a folder in which you want Icon source code to reside (which we call icon here).

Make four folders in icon for the source code:

```
mkdir h
mkdir icont
mkdir iconx
mkdir common
```

To unload the source code, place the Icon distribution diskette in a disk drive (assumed to be a: here). Then

```
cd h
arc x a:h.arc
cd ..\icont
arc x a:icont.arc
cd ..\iconx
arc x a:iconx.arc
cd ..\common
arc x a:common.arc
cd ..
```

Compiling Icon

Your PATH variable should be set so that the Lattice compiler can be found. For example, if the compiler is in d:\lattice, the command

```
set PATH="d:\lattice:$PATH"
```

can be used. Better yet, include this PATH in your ash.ini file, so that subshells can find the compiler.

Scripts are provided to carry out the compilation. First compile the programs in common, which are used in building the two executable parts of Icon:

```
cd icont
makecomn
cd ..
```

Then build icont, which translates and links Icon programs:

```
cd icont
makeicnt
mv icont.prg ..
cd ..
```

Next, build iconx, the Icon executor:

```
cd iconx
makeicnx
mv iconx.prg ..
cd ..
```

If all goes well, you should have a running version of Icon in your icon folder. You may wish to move the two .prg files to a convenient place on your search path.

Testing Icon

To test Icon, create a folder tests in your icon folder and unload the test programs from the distribution diskette:

```
mkdir tests
cd tests
arc x a:tests.arc
```

As a simple first test, try

```
icont hello
iconx hello
```

You should get output that looks like this:

```
Icon Version 8. March 25, 1990.
Atari ST (GEMDOS) Lattice C 3.04.02
Atari ST
co-expressions
environment variables
error trace back
expandable regions
external functions
keyboard functions
math functions
string invocation
system function
```

To perform more extensive testing, create two new folders in tests to hold "standard" and "local" test output:

```
mkdir stand
mkdir local
```

Standard output is on the distribution diskette. To unload it,

```
cd stand
arc x a:stand.arc
cd ..
```

Complete testing is time consuming. You may want to do only part of it, or do it in pieces. The script dtest.ash runs one test program and puts the output in local with a .out extension. For example,

```
dtest hello
```

runs hello.icn and puts the output in local\hello.out.

If you want to do all the testing at once, use

```
alltest
```

(and prepare to wait a while).

To be sure Icon is running correctly, compare the output in stand to that in local. There will be some differences, since some of the tests depend on the time, date, and environment. These differences should be obvious. If you're ambitious, you may wish to write an Icon program to compare output files in the two directories and list and differences. For a start, write a program that compares output files and simply reports if there are any differences.

Trouble Shooting

The most likely problem you may encounter is inadequate memory for iconx. It reserves (via MNEED) 200,000 bytes of RAM for its allocated storage. If you do not have that much available, iconx terminates with a "setblock failure".

If this happens, you may need to remove resources that are using memory, such as RAM disks. Alternatively, you can reduce MNEED (which is in iconx\rlocal.c). If you reduce MNEED, you'll probably also have to reduce the sizes of Icon's allocated data regions. This can be done using environment variables as described in [3]. For example, Icon's string and block regions normally are 65,000 bytes. You can try small sizes to get started, such as

```
set STRSIZE=10000
set HEAPSIZE=10000
```

It's also possible that most programs will have enough memory, but some won't. For example, very large programs (like augment.icn in tests) may exceed the amount of memory available. This shows up as a run-time error during startup of iconx. Setting environment variables may provide a work-around. Similarly, programs that create an unusually large number of co-expressions (like coexpr.icn in tests) may run out of memory during execution. Again, taking space from one region and giving it to another may provide a work-around.

If you find that Icon's default region sizes will not work for routine programs, you may wish to put appropriate alternative settings in ash.ini so that you don't have to set them manually every time you start up.

Technical Notes

The -x option to icont is disabled because it does not work properly. The system(s) function also does not work.

Acknowledgements

The original modifications to adapt the Icon source code to the Atari ST were done by Rick Fonorow and Jerry Nowlin. Charles Richmond did most of the subsequent work.

References

1. R. E. Griswold and M. T. Griswold, *The Icon Programming Language*, Prentice-Hall, Inc., Englewood Cliffs, NJ, second edition, in press.
2. R. E. Griswold, *Version 8 of Icon*, The Univ. of Arizona Tech. Rep. 90-1, 1990.
3. R. E. Griswold, *Version 8 of Icon for the Atari ST*, The Univ. of Arizona Icon Project Document IPD136, 1990.