# Dynamic C 7.20 Release Highlights

See releasenotes.txt in the Dynamic C root directory for more details.

## New Features

### Project Files

Project files facilitate working on multiple Dynamic C projects. Project-level information such as compiler options, libraries used, files open, and project-scope #define macros can now be saved in project files. Use the project command on the file menu.

### Support for enums

Dynamic C now supports the standard intrinsic enum type.

### Precompilation

Functions can be listed in the file precompile.lib to save recompiling them every time.

### µC/OS-II upgrade

The Rabbit/Dynamic C port of Jean Labrosse's popular real-time operating system for embedded systems, µC/OS-II, has been upgraded to version 2.51. New features include mutex semaphores and event flags. µC/OS-II for Rabbit is available only in Dynamic Premier, and may be used license and royalty free in end products.

### TCP/IP improvements

Many improvements to the robustness and performance of the TCP/IP stack have been made. (Most of these were introduced in version 7.06P2) See releasenotes.txt in the Dynamic C root directory for details.

### Command line RFU version

A GUI-less Rabbit Field Utility for loading programs without Dynamic C is now available.
Clear all breakpoints command
A command on the Run menu to remove all breakpoints at once has been added.

### New system macros

**__LINE__, __FILE__, __DATE__, __TIME__** macros added.

### Library encryption

A utility program that is not released with Dynamic C, but is available from Z-World Tech Support will encrypt libraries in a format that Dynamic C can decrypt. This is primarily of interest to people who want to make arrangements with Z-World to resell Dynamic C as a programming tool for a programable end product.

## Other Functional Improvements

### Faster debugging

Single-stepping through code and printf are now noticeably faster.

### Large-sector flash support

Dynamic C now Supports a larger variety of flash types, including several very-large, non-uniform sector size types. This change required some fundamental changes in the way the debugger interacts with the target. Single-stepping and breakpoint setting used to cause flash writes to occur. This no longer happens.

### Faster more flexible cloning

Cloning is a feature that allows one Rabbit target to copy itself to another using a special cloning board available from Z-World. (The simple circuit is easily designed into board test fixtures also.) Previously, a “fast” cloning was option was available but had some restrictions. Now, all cloning is fast (up to 921.6 kbaud), and options having been added for creating “sterile” clones, copying the User and System ID blocks, running after copying, copying a second flash, and more. See the cloning configuration options near the top of rabbitbios.c or the Rabbit 2000 Designer's Handbook.

### Constant string optimization

If an identical literal string is defined in code multiple times, only one instance of it will be generated used.

## Other Important Functional Changes

### Assembly level debugging

Previously, #asm blocks embedded in C functions took on the “debug/nodebug” characteristic of the C function it was located in. The #asm directive to begin an assembly block now can take a “debug” or “nodebug” keyword, and the default is nodebug, regardless of the how the C function is declared. This means that a breakpoint can not be set inside an #asm block unless "#asm debug" is explicitly used. Due to the changes to support large sector flash, using debug on the assembly block now causes rst 28Hs to be placed in between instructions so that breakpoints may be set. This may have the side effect of making short jump instructions out of range (an assembler error message will be generated). This can be fixed by changing jr to jp, and will not break pre-existing code since the debug keyword was not usable with #asm before. Nodebug Assembly blocks may still be single-stepped through in the disassembly window

### Targetless compilation

Compiling with "Used Attached Target" no longer compiles the bios,
and requires that the target have a running bios/program. The Options menu now has a Define Target Configuration Command. The Compile to *.bin file command now use the target configuration defined in the new option to compile, instead of popping the Target Configuration Dialog up for every compilation.

**RabbitLink Changes**

The RabbitLink has had a major overhaul in Dynamic C 7.20 to bring it in sync with the Ethernet Loading utility of the DeviceMate, to allow for future compatibility, and to allow pure-Ethernet loading, without ever needing a serial port, through the use of DHCP. Because of this overhaul, it will be necessary to upgrade the firmware on the RabbitLink hardware for it to be able to communicate with Dynamic C 7.20 or later. The New RabbitLink firmware has the flexibility to allow for future changes in communication method between the Dynamic C and the target and avoid future incompatibilities.