



Contacts: Nancy B. Green
The William Baldwin Group
nbgreen@william-baldwin.com
+1 650 856 6192

Peter Riachi
HI-TECH Software
priachi@htsoft.com
+61 7 3722 7777

FOR RELEASE APRIL 2, 2007

Call- And Pointer-Graph- based Compilation Technology Yields Denser, More Portable Code

OMNISCIENT CODE GENERATION™ Yields up to 50% Denser Code for PIC 18 MCUs

Booth 937, Embedded Systems Conference, San Jose, CA, April 2, 2007. . . HI-TECH Software today announced a new compilation technology that generates object code, based on call and pointer reference graphs derived from *all* the modules in an embedded program. Called Omniscient Code Generation (OCG), the new technology overcomes many of the pitfalls of conventional compilers that frequently miss inconsistent variable declarations, and redundant code, and use calling conventions, because they compile each module independently. OCG optimizes the size of each pointer variable, based on its usage, eliminates the need for many non-standard C qualifiers and compiler options, produces more optimal interrupt context switching code, and customizes the functionality of library functions.

OCG allows more portable source code that is nearly 50% more dense than code from competing compilers. Denser code directly results in better processor performance because the same task can be executed with fewer instructions and fewer cycles.

True Global Optimization. The phrase *global optimization* is used frequently to describe compilers that optimize within one code module, which is not really global. In contrast, OCG technology looks at *all* code modules and collects comprehensive data on register, stack, pointer,

object and variable declarations. It uses this information to: ensure consistent variable and object declarations between modules; optimize stack and register allocation; and delete unused variables and functions. This process requires no extra input from the programmer, no non-standard extensions, and is entirely transparent. Recursively or re-entrantly called functions are identified and dynamic stack space or local variable storage can be used to ensure that re-entrant calls do not overwrite existing data.

Efficient memory usage without compromising code portability. The C language assumes a single linear address space. However, many embedded processors have complex, non-linear memory architectures with different word widths, and are basically incompatible with C code. This situation makes it difficult to map the C code onto some processors' memory maps. Programmers usually resolve this problem by explicitly declaring the address spaces that a pointer will access. While this solution results in efficient pointer usage, it is by definition, architecture-specific and hampers code portability. It can also lead to subtle bugs being introduced into the program.

OCG technology has intelligence about the complete set of used variable and pointers across all program modules, it also knows exactly how big the stack must be and where it will be located before the code is generated. The compiler defines a set of address spaces for each pointer variable that is optimally efficient for the particular processor architecture, without any specific direction from the programmer.

Determining Optimizing the memory space for each pointer this is one of the significant features of OCG. An algorithm uses each and every instance of a variable having its address taken, plus each and every instance of an assignment of a pointer value to a pointer (either directly, via function return, function parameter passing, or indirectly, via another pointer) and builds a data reference graph, known as a Pointer Reference Graph. The graph is completed and then identifies all objects that can possibly be referenced by each pointer. This information is used

to determine which memory space each pointer will be required to access. Any conflicting declarations of the same object from different modules can be detected and an informative error message issued to the user. Any variables never referenced can be deleted.

Customized library functions. OCG is a truly global view of the program. Therefore, complex library functions can be implemented in a way that is specific to each particular program. A good example of this is the C library functions `sprintf()` and `printf()`. These workhorse routines for formatting text strings or output are enormously useful, but can occupy a large code footprint (5 kB or more), if implemented in their entirety. For example, if the `printf()` function, which can take up to 5,000 bytes of code when its number formatting functions are included, is being used to output simple strings, it can be reduced to 20 or 30 bytes of code.

Re-entrant code without a stack. Omniscient Code Generation builds separate call graphs for both main-line and interrupt code. Any functions that appear in more than one call graph can be replicated with their own local variable area, eliminating the requirement for a separate stack. This is useful when working with small embedded microcontrollers that cannot implement stacks for local variables.

Bottom-up code compilation optimized memory use. Machine code is generated beginning from the bottom of the call graph, with those functions that do not call any other functions. Code generation then proceeds up the call graph, so that for each function, the code generator knows exactly which functions are called by the current function, and therefore also knows exactly what registers and other resources are available at each point. Calling conventions can be tailored to the register usage and argument type of a function, instead of following a set pattern.

Generates customized runtime startup code. Conventional compilers provide canned startup code which is frequently much larger than absolutely necessary because it may be initializing

variables that aren't even used. A compiler with Omniscient Code Generation allows it to generate startup code in the minimal amount of bytes required for the application.

Implemented in new HI-TECH Software compilers. HI-TECH Software has implemented Omniscient Code Generation in compilers for Microchip's PIC18 microcontroller family and Cypress' Mixed Signal Controllers. During 2007 and 2008, the company will launch additional OCG compilers for all other Microchip MCUs and DSPs, ARM7-based MCUs from Atmel, NXP, OKI and Samsung, and 8051 microcontrollers from over 50 manufacturers.

About HI-TECH Software

HI-TECH Software is a world class developer of development tools for embedded systems, offering compilers, RTOS and an Eclipse based IDE (HI-TIDE™) for 8-, 16-, and 32-bit microcontroller and DSP chip architectures. Its products support PICmicro MCUs and DSPs, PSoC™ Mixed Signal Processors, ARM, 8051, TI MSP430, HOLTEK, ARClite, XA, and Z80 architectures to name a few. Its customers include tens of thousands of embedded system developers including General Motors, Whirlpool, Qualcomm, and John Deere. HI-TECH Software is located in Brisbane, Australia, with an office in Gilroy, California, and an extensive network of distributors around the Globe.