Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

JANUARY 2005

News

HI-TECH SOFTWARE

NEW! Three Award-Winning Compilers in One

HI-TECH Software's PICC Enterprise Edition is the only compiler package available with comprehensive support for all 10/12/14/16/17/18 series PICs and also the new dsPIC processors. PICC Enterprise Edition is a low-cost package that unites our successful PICC, PICC-18 and dsPICC compilers.

With the choice of over 300 different PIC processors, deciding which one to use is a challenge. Even when you do decide, changing project requirements during development often call for a larger or more sophisticated chip. A change in processor can also mean buying a new compiler… until now!

With HI-TECH Software's PICC Enterprise Edition, there is finally a single product that supports **ALL** of Microchip's PICmicro processors. Not only does this simplify which compiler to choose, it also eases migration between processors by having a familiar tool-set.

PICC Enterprise Edition is particularly advantageous to existing PICC, PICC-18 and dsPICC customers as the package is a cost-effective way of keeping all of your compilers up-to-date through a single license option. Best of all, upgrades to the full PICC Enterprise Edition are at a special price!

With the strong relationship between HI-TECH Software and Microchip, you can be assured that the PICC Enterprise Edition will be the first to support Microchip's devices as they become available.

For further information or to download a demo, go to **www.htsoft.com/products/eepiccompiler.php** ◘

Coming Up...

HI-TIDE

Beta 1 of the HI-TIDE v3.xx series was released on November 2, 2004. This is available for PICC compiler owners with extended support and those who own the PICC Enterprise Edition. As the first in a series of releases to provide a quality HI-TECH IDE (HI-TIDE) for our compilers based on Eclipse, Beta 1 is targeted at the PICC compiler and allows project management, incremental builds, colour coded editing and CVS support.

Beta 2 is set to be released in the New Year with a number of user interface improvements and initial implementation of the C Wiz code wizard for PICC. C Wiz is a development tool that provides initialization code for on board peripherals.

HOLTEK

A release candidate of HI-TECH C for Holtek MCU is currently undergoing verification ahead of the final release of this new compiler in early 2005. With support for the full Holtek MCU family, this compiler features the latest in code generation technology that streamlines programming, yet delivers outstanding code density. The compiler driver and debug output integrates with Holtek's development tools making development a breeze.

For more information on these upcoming compilers, contact one of our sales offices or email us on **sales@htsoft.com** ◘

Recognized Microchip Third-Party Tool Provider

HI-TECH Software proudly supports the Microchip brand with high-quality C compilers.

Inside this Issue

Design Tip: Give Your Project the Boot!	**2**
Embedded Systems Conference	**2**
Tech Tip: Beware the Buffer Overflow	**3**
Upgrade Now: Latest Compiler Versions	**4**
HI-TECH C Powering On...	**4**

Design Tip: Give Your Project the Boot!

With flash memory becoming more readily available in microcontrollers, the possibility of software updates in applications can now be more easily realized. This article is a brief outline on how to develop a bootloader for a Microchip PIC processor (16F87x). The same procedure, however, could be applied to many other types of processors (8051, ARM, MSP430, etc.).

Before any coding starts, you should first think about how things are going to be positioned in program memory. Since the bootloader needs control after reset, it will need to use the reset vector. Since it would be nice to make it easy to build bootloader compatible programs, it is best if the bootloader is positioned entirely at the start of memory, or, all at the end of memory. On mid-range PICs, the regular interrupt vector is located at address 0x4. If the bootloader was going to be positioned entirely at the start of memory, it would then have to redirect the interrupt vector to the downloaded program. Since this would add complexity to the code and latency to the vector, the bootloader will be positioned at the end of memory. The reset vector will still be needed, so this will mean shifting the downloaded program's reset vector elsewhere.

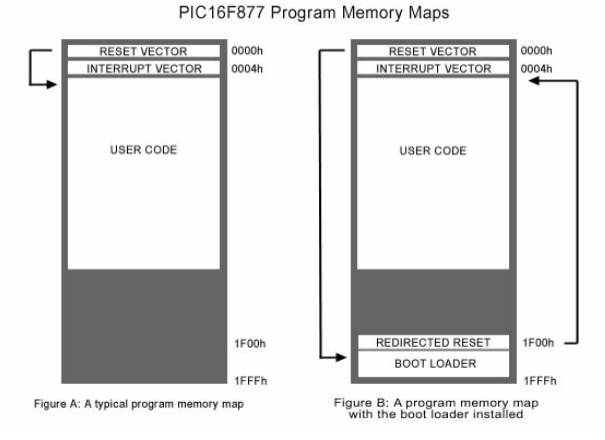

Figure A: A typical program memory map

Figure B: A program memory map with the boot loader installed

The diagram above shows the memory map of a program with and without the bootloader installed.

In Figure A, typically the reset vector contains instructions to jump into the main program. As the bootloader needs the reset vector, you must move the downloaded program's reset vector elsewhere. Figure B shows how the reset vector now points to the bootloader code and the downloaded program's reset vector will be positioned to an address just before the bootloader.

How it Works:

- On reset the bootloader will take control and prompt the user to send a hex file. It will wait here (with count down) for a configurable number of seconds.
- If no hex file is sent, the bootloader will assume that no update is required and it will jump to the redirected reset vector which in turn will run the previously downloaded program. If no program has ever been downloaded, the redirected reset vector will simply contain a jump to the beginning of the bootloader and the whole process will start again.
- If a HEX file is sent to the bootloader within the count down period, it will start interpreting the data and writing it to program memory. The bootloader will look for addresses less than 0x4 (reset vector) and will instead write this elsewhere. It will also ignore addresses conflicting with those used by the bootloader, thus protecting itself from being overwritten.

This bootloader is suitable for the following processors: 16F870, 16F871, 16F873, 16F873A, 16F874, 16F874A, 16F876, 16F876A, 16F877, 16F877A. All source code, documentation and a sample download program is included with the latest version of the compiler. There is also further information on HI-TECH Software's online forums at **www.htsoft.com/support/forums.php** ◘

Tech Tip: Beware the Buffer Overflow

A Spotter's Guide

The phrase "buffer overflow" is almost making it into the mainstream press these days, but many people who write programs which exhibit these mythical beasts are even unaware of what they are or why they are a problem.

```
#include <string.h>
main() {
        const char greet[] =
                    "hello";
        char buffer[5];
        char precious = 7;

        strcpy(buffer,
               greet);
/* assume precious==7
*/
}
```

Here's a buffer overflow. The constant array named `greet` actually contains 6 characters, which `strcpy()` tries to copy into `buffer` where there is only room for 5. What will probably happen is that the last character (the invisible '\0' that terminates the string "hello") will be copied into the space formerly known as `precious`, and the assumption that `precious` contains the 7 we put into it will be violated.

```
#include <string.h>
main() {
        const char greet[5] = "hello";
        char buffer[6];
        char precious = 7;

        strcpy(buffer, greet);
        memcpy(buffer, greet,
               sizeof buffer);
}
```

Now we've fixed the first problem by making `buffer` big enough to hold all of `greet`, the `strcpy()` is then broken because `greet` is smaller. "How?" you ask. By making `greet` into a char array that is not a string, i.e. a char array that has no terminating '\0'. The `strcpy()` now will not know where to stop, and will keep copying until it finds a zero byte, if that ever happens.

The `memcpy()` is better, because it is guaranteed to stop, but it still does (possibly unquantifiable) Bad Things, as it tries to copy from one byte past the end of `greet`.

A word of warning: do not assume `strncpy()` will save you: it is, unfortunately, not simply a bounded version of `strcpy()`, and has its own set of fangs. Most significantly, unlike all the other `str...()` functions in the Standard Library, it will not necessarily give you back a NUL-terminated string even if you give it one to work with. If its source is a long but valid string, and its destination is a shorter buffer, it will fill the buffer with characters from the string but it will NOT terminate that buffer with a '\0'.

The line with `gets()` might not be a buffer overflow, if the user (or wherever `gets()` is getting its input from) enters at most 4 characters before a newline; `gets()` cannot tell that it is not allowed to write past `buffer[4]`. The line with `fgets()` can never overflow its buffer. Note, though, that `fgets()` does not eat the '\n' like `gets()` does.

```
#include <stdio.h>
main() {
        char buffer[5];
        gets(buffer);
        fgets(buffer, sizeof
              buffer, stdin);
}
```

There are many more possible ways to overflow a `buffer`, writing data in all sorts of places it was never meant to go. Sometimes the effect might not be noticeable, but you can never be sure. And while the humble string or character array is the most common target, buffer overflows are not type-bigots - they will strike anything they can touch.

To Slay the Monster

When hunting them, the first thing to look for is functions which write into arrays without any way of knowing the amount of space available.

If you get to define the function, you can pass a length parameter in or ensure that every array you ever pass to it is at least as big as the hard-coded maximum amount it will write.

If you are using a function someone else (say, the compiler vendor) has provided then avoiding functions like `gets()`, which take some amount of data over which you have no control and stuff it into arrays they can never know the size of, is a good start.

Make sure that functions like the `str...()` family which expect NUL-terminated strings actually get them - store a '\0' in the last element of each array involved just before you call the function, if necessary.

Good hunting ◘

Upgrade Now: Latest Compiler Versions

Compiler	Version
68000 C Cross Compiler	V7.21
68HC05 C Cross Compiler	V7.80
68HC11 C Cross Compiler	V7.80
8051 C Cross Compiler	V9.01
ARClite C Cross Compiler	V7.85PL1
dsPICC Compiler	V9.50
H8/300(H) C Cross Compiler	V7.80
HI-TECH C for ARM	V9.12

Compiler	Version
HI-TECH C for msp430	V9.01
HI-TECH C for XA	V7.73PL1
PICC Compiler	V8.05PL2
PICC Enterprise Edition	V2005
PICC-18 Compiler	V8.35PL2
Salvo for 8051/msp430/PICs	V3.2.3
Z80/Z180 C Cross Compiler	V7.80PL2

For more information, go to **www.htsoft.com/updates** ◘

HI-TECH C Powering On...

Foxboro's Remote Terminal Unit Development Team

invensys® FOXBORO®

With a renowned reputation in the control industry, Foxboro SCADA Australia sees its primary focus as developing world-class Supervisory Control and Data Acquisitions (SCADA) systems. For over 15 years, Foxboro has been using HI-TECH C to design and develop software for Remote Terminal Units (RTUs), which are installed into numerous large power substations - including those of Powerlink Queensland in Australia and Transpower in New Zealand.

Working under strict validation and verification regimes, HI-TECH C was selected as the best software to address these requirements. "With high quality, low-cost software and readily available access to expert technical support - who else would we have chosen for the job," states Colin Weaver, Senior Software Engineer of Foxboro. "All the hardware designers and embedded software engineers in the company have had some form of contact with HI-TECH Software's tools and are extremely pleased and comfortable with using HI-TECH C compilers. We are particularly impressed with and heavily use the remote debugger feature."

This enduring and successful relationship between HI-TECH Software and Foxboro is set to continue for many more years to come. As Foxboro looks into developing new generations of hardware, they are adamant with maintaining their HI-TECH C compiler-based designs.

For more on what our customers say, go to **www.htsoft.com/news/testimonials.php** ◘

Check the Tech!

HI-TECH Software's online forum is where over 3,000 members from around the world come to discuss and get support with all issues related to our products. Go to **www.htsoft.com/forum** ◘

HI-TECH Software LLC 6600 Silacci Way Gilroy, CA 95020 USA
Ph: 800 735 5715 **Fax:** 866 898 8329

HI-TECH Software PO Box 103, Alderley, QLD 4051, Australia
Ph: +61 7 3552 7777 **Fax:** +61 7 3552 7778

Web: http://www.htsoft.com
Email: sales@htsoft.com