



HI-TECH PICC Standard Version 9.60

Release Notes

Copyright (C) 2007 HI-TECH Software.
All Rights Reserved. Printed in Australia.
Produced on: March 19, 2007

HI-TECH Software Pty. Ltd.
ACN 002 724 549
45 Colebard Street West
Acacia Ridge QLD 4110
Australia

email: hitech@htsoft.com
web: <http://www.htsoft.com>
ftp: <ftp://www.htsoft.com>

**THIS FILE CONTAINS IMPORTANT INFORMATION RELATING TO THIS
COMPILER. PLEASE READ IT BEFORE RUNNING THIS SOFTWARE.**

Chapter 1

Introduction

1.1 Description

This is a minor update from the previous version and adds some new features and corrects reported issues.

1.2 Notes

Please note that the version 9.xx command line driver has a different format for options compared with previous versions. Please refer to the user manual for details on using the command line options. During the transitional period, the new command line driver will accept the old-style options, but this should not be relied upon for future versions.

1.3 Previous Versions

The previously released PICC compiler version was 9.50, released September 2005. Subsequent patch level releases were 9.50PL1 (October 2005) and 9.50PL2 (February 2006).

A Beta release of this compiler was released in December 2006. The Beta release was versioned 9.55.

Chapter 2

New Features

2.1 General

New Processors (9.60) Support for these processors has now been added: 16F610 and 16HV610.

MPLAB plug-in (9.60) This version of the compiler will install a new revision of the compiler's plug-in for Microchip's MPLAB IDE. This plug-in is suitable for use with all version 9.xx compilers. Once installed, a new selectable language toolsuite will appear. Set the toolsuite to *HI-TECH Universal ToolSuite*. This toolsuite can be used to compile projects using version 9.xx PICC, PICC-Lite, PICC-18 and dsPICC compilers.

Off-line activation (9.60) It is now possible to perform an off-line activation for Mac OSX and Linux versions of the compiler.

New Processors (9.55) Support for the latest processors has been added (12F609, 12HV609, 12F615, 12HV615, 16F616, 16HV616, 16F631, 16F667, 16883, 16F884, 16F886, 16F887).

2.2 Driver

-CHECKSUM option (9.60) This new option can be used to generate a checksum over the contents of an address range and store the result at a nominated address location. The result can be stored in program memory or EEPROM. Sample code has also been provided to demonstrate runtime verification of this checksum.

-RUNTIME=+RAMTEST (9.60) Adding this option to the driver during the link stage will build in a simple checkerboard RAM test to verify the integrity of each cell of general purpose RAM in the targetted device. The tests performed are contained within the library routine `__ram_cell_test`. This module can easily be replaced with a user-defined implementation if the default cell test is insufficient for requirements.

- _HTC_EDITION_ (9.60)** The driver now defines a new preprocessor symbol to identify whether the compiler in use is a PRO, Standard or Lite edition. The value of `_HTC_EDITION_` in each case is 2, 1 and 0 respectively. Source that tests the value of this symbol can compare against these values or against symbols `__PRO__`, `__STD__` or `__LITE__` definitions in `htc.h`.
- RUNTIME=+DOWNLOAD (9.55)** Adding this option to the driver during the link stage will generate a program that has particular enhancements that make the program more suitable to be downloaded to a device via a pre-installed bootloader.
- Input HEX parameters (9.55)** Intel HEX files listed on the driver's command line get merged with the compiled output via Hexmate. Now the driver permits Hexmate's additional specifications to the the input file such as address range restriction and address shifting. As the additional specifications are passed directly to Hexmate they should conform to Hexmate's expected syntax. For example the input `0-ff+1000, generic.hex` will restrict the data being read from `generic.hex` to that contained between addresses 0 and FFh and merge this data at an address which is shifted up by 1000h.
- SETOPTION enhancement (9.55)** From version 9.55, `--SETOPTION` option allows an option file to be passed to any of the intermediate applications employed in the project compilation process. Now, an intermediate application can be dropped from the build process by passing *off* as the file parameter to the application. Note that only Clist and Hexmate are considered non-critical to the build process. Turning off any other application will cause the build to fail.
- FILL=hexcode,data (9.55)** Hexmate's `-FILL=hexcode,data` enhancement is available from the driver's `-FILL` option. For more details on the `,data` enhancement read Hexmate's new features below.
- Memory symbols (9.55)** New preprocessor symbols are defined by the driver to define the amount of program memory and EEPROM memory. These symbols are named `_ROMSIZE` and `_EEPROMSIZE` respectively. These symbols are also consistent with the PICC-18 compiler to allow for greater code portability. The existing definition of `EEPROM_SIZE` will be retained for backward compatibility but is not as portable. Where possible, use the newer `_EEPROMSIZE` symbol instead.

2.3 Libraries

- New routines `__ram_cell_test` / `ram_test_failed` (9.60)** This routine has been added to the compiler library. These routines are not intended to be called directly from user code but will be called from the generated runtime startup code to include a RAM integrity test. The cell test will be called repeatedly for every address in general purpose RAM. If the cell test detects a failure on the RAM cell, it can pass the failure code to the failure handling routine. Being library routines, a user can replace either of these routines with more sophisticated implementations to suit their needs.

2.4 Header files

`_HTC_EDITION_` identifiers (9.60) The header file `htc.h` now provides definitions for the possible values that the preprocessor symbol `_HTC_EDITION_` could contain. The possible values of this symbol are `__PRO__`, `__STD__` and `__LITE__`.

`__IDLOC7()` macro (9.55) A new macro defined in `pic.h` permits programming of up to seven bits per ID location. This macro should **ONLY** be used on those devices which support seven bit ID location programming. Other devices should continue to use `__IDLOC()`.

2.5 Hexmate

`-ADDRESSING` (9.60) This new option will allow the address fields in all input options to be entered in an addressing format other than byte addresses. This is particularly advantageous for the PICC compiler as the program memory is word addressed (use `-ADDRESSING=2`).

`-FILL=hexcode,data` (9.55) The `-FILL` option now accepts additional flags to further customize how a fill code will be applied to unused program memory. The first (and only) flag implemented is `,data`. Adding this to the `-FILL` command will restrict the program filling to act only on areas of program space that already contain data. This facility will cause the records of the generated output file to all be of the same length and to all align on addresses which are a multiple of that length. No new data records will be generated for regions of program memory that are void. The default length is 16 bytes and can be changed with the `-FORMAT` option. Intel hex files that have been conditioned in this way are optimized for bootloaders.

Data record length (9.55) Now the maximum length of an INHX data record has been increased from 32 to 128 data bytes. The default maximum length of a data record still remains at 16 bytes.

Checksum algorithms (9.55) Previously checksums would only be calculated using an 8 bit summation algorithm. This option has now been enhanced so that one of eight simple algorithms can be selected to perform the computation.

Checksum endianism (9.55) You can now control whether multi-byte checksum result be stored in little-endian or big-endian format. When specifying the width of the checksum, use a negative width to specify little endian format. If no width is specified, the default result will be two bytes wide stored in little endian format.

No output option (9.55) It is now permissible to use the `-Ofilename` option without specifying a filename. Leaving *filename* blank will suggest that no hex file will be generated. This is useful if hexmate is being used purely for its diagnostic abilities. For example verifying hex file integrity or creating a logfile summary. This is different from not specifying the `-O` option at all, which will send the generated output to *stdout*.

2.6 Sample code

Checksum verification (9.60) A new sample is available that demonstrates how to calculate a checksum over code space and verify this result against the executable's built-in result which was calculated at compile time using the driver's `--CHECKSUM` option.

Bootloader version 3.11 (9.55) An updated bootloader is now distributed. The new bootloader works across a greater range of baud rates, is more reliable, can be compiled into an even smaller code footprint and is simpler to use than the previous bootloader. Combined with other features in this tool-suite release, download applications are no longer required to implement a dummy interrupt routine.

Chapter 3

Changes

3.1 Driver

Startup clearing code (9.60) The generated startup code for clearing bit types and uninitialized variables contains more comments to better describe the memory clearing process.

Address ranges in `-FILL` and `input.hex` (9.60) Previously any address range fields defined in a `--FILL` option or when selecting a specific range in an input hexfile had to be entered as byte addresses (which mean multiplying each program address by two). Now address fields are required to be entered as word addresses which corresponds to the processor's natural form of addressing.

MPLAB_ICD symbol (9.60) The defined preprocessor symbol `MPLAB_ICD` is being phased out. The replacement symbol `__MPLAB_ICD__` is now defined and performs exactly the same duty. The new symbol provides better standards conformance and portability across into PICC-18 and dsPICC tool-suites. For a time the old symbol will continue to be defined for maintenance of existing projects, however new projects should not refer to the old symbol.

`-MAPFILE` option (9.55) This option provided duplication of the `-M` option's functionality. It has now been removed. Any project which currently uses `--MAPFILE` should now use the equivalent option, `-M`.

EEPROM_SIZE (9.55) A new driver defined preprocessor symbol, `_EEPROMSIZE` supersedes the previous `EEPROM_SIZE` definition. Existing `EEPROM_SIZE` definitions will be retained for maintenance of older projects, however it is advised that new projects should use the `_EEPROMSIZE` symbol.

3.2 Assembler

Object file format (9.55) The object file format has changed since version 9.50 (which used version 3.7). The new object file version is 3.8. The consequence of this is an object file compiled from the version

9.60 toolsuite cannot be linked by the version 9.50 (or prior) toolsuite. However objects compiled by a version 9.50 toolsuite can be linked by the 9.60 toolsuite. If you are distributing objects or library files containing objects compiled by the version 9.60 compiler, be aware that if your customers are using a lesser version of the toolsuite they may also need to obtain an updated linker.

3.3 Hexmate

Input file specifications (9.55) Special conditions for input files such as range-restriction and address shift were requested with the command line syntax *filename,<specs>*. Now the specifications are to be listed before the filename ie. *<specs>,filename*.

Default checksum endianness (9.55) Previously calculating a checksum to store a result of unspecified width, the default result width would be two bytes and the bytes stored in big-endian format. Now the default result will be stored in little-endian format. To select the old behaviour, specify a width of two by adding *w2* to the *-CK* option.

3.4 Chip configuration file

New attributes (9.55) The following new attributes have been added to the chip configuration file to better describe various devices:

- **FLASH_READ** Defines the number of program words that are read upon an execution of a read from flash memory. Defaults to zero if unspecified.
- **FLASH_WRITE** Defines the number of words that are transferred to flash memory during execution of a single write to flash. Defaults to zero if unspecified.
- **FLASH_ERASE** Defines the number of program words that are erase upon execution of a single flash erasure operation. If unspecified, defaults to the **FLASH_WRITE** value.
- **CONFIG** Define an address range for device configuration registers if this device differs from the default for that architecture.
- **IDLOC** Define an address range for user ID locations if these locations differs from the default for this architecture.

3.5 Libraries

pic???-p.libs (9.55) These libraries which contain the routines used for writing to flash have been further classified to describe new PIC variants that employ new techniques for writing to flash. This has allowed further optimization of the flash writing routines for some devices.

3.6 Documentation

New readme layout (9.55) As the content of the readme file has increased in recent releases, the style of this document has been enhanced to include more formal sections for easier navigation. The PDF version of the readme document now includes bookmarks.

Chapter 4

Limitations

4.1 Preprocessor

Incorrect result for sizeof The result of the `sizeof` preprocessor operator when applied to a pointer will always return 1. Use of the C `sizeof` operator will return the correct value.

4.2 Parser

Missing parenthesis not detected A missing end parenthesis is not detected in variable declarations when the variable is type cast. For example,

```
static char C @ ((unsigned)&PORTA;
```

This problem does not appear to affect any output code, but may cause problems when using third-party software tools.

Qualified arrays generate error The compiler generates an error when compiling complicated qualified arrays, e.g.:

```
const char * const * const listnames[] = {menu0, menu1};
```

The error is not issued if the array is not qualified, or for arrays of more basic types.

Structure initialization Locally defined structures cannot be initialized.

4.3 Code Generator

Functions returning ROM structure For Highend devices (17Cxxx), a function cannot return a copy of a structure which resides in ROM.

Indirect function calls For Highend devices (17Cxxx), certain indirect function calls with more than one byte of argument may produce incorrect code. Indirect function calls should not be used on Baseline devices.

Functions as arguments to printf The argument list to printf or sprintf should not include function calls. For example,

```
printf("x = %f, sin(x) = %f\n", x, sin(x*3.141592/180.0));
```

This code demonstrates a workaround:

```
y = sin(x*3.141592/180.0);
printf("x = %f, sin(x) = %f\n", x, y);
```

Error on initialization of complex structures Any structure using structures within a structure or union cannot be directly initialized. For example, using the following types where a structure is a member of a union, initialization of the structure's bitfields will generate an error.

```
typedef struct {
    unsigned b0:1, b1:1, b2:1, b3:1, b4:1, b5:1, b6:1, b7:1
} byte_bits;
typedef union {
    byte_bits  bits;
    char       byte;
} byte_or_bits;
```

This will generate a compiler error:

```
byte_or_bits example = { {1,0,1,1,0,1,0,0 } };
```

Instead, use a single value:

```
byte_or_bits example = { 0b10110100 };
```

Multiple block variable declaration Functions which have variables declared within multiple blocks where code within the block requires the use of temporary memory, may get corrupted when compiled with global optimizations. A workaround is to move the inner variable declaration outside of the block.

Void pointer size Void pointers are one byte in size and may not be large enough to point to all objects.

4.4 Libraries

Minimum floating point value The floating point math libraries cannot perform operations predictably on numbers below the order of $1e-35$. The result is either correct or zero.

Chapter 5

Bug Fixes

The following are descriptions of bugs that were present in the previous version(s) of the compiler and have been fixed in this release.

5.1 Driver

Psects in COMBANK class (9.60) Psects to be linked in the COMBANK memory class may have missed out on allocated space if psects linked in the overlayed BANK0 class had already filled the COMBANK address range. This resulted in a *Can't find space...in segment COMBANK* linker error. Now the psects of the COMBANK class have first priority when being linked so that this scenario is now avoided.

Memory clearing in BANK0 (9.60) In rare circumstances the clearing of memory in bank 0 may not have occurred on some Baseline or Midrange PIC devices. This scenario was dependant on the size of various psects. This event may have been triggered if the size size of `rbit_0` psect was greater than zero bytes AND the size of the COMBANK class minus `temp` psect was greater than the size of `rbit_0` psect but less than (`rbit_0` plus `rbss_0`).

10F222/0 ID locations (9.60) The user ID locations were being linked at the wrong address for the PIC10F220 and PIC10F222.

PIC17 reset bits (9.60) The `--runtime+=resetbits` feature used to save the reset condition indicator bits during startup, saved the wrong bits for PIC17 devices.

Indirect access in PIC17 (9.60) In some cases (also dependent on potential optimizations occurring later in the build process), indirect data access would not have worked for PIC17 devices.

Language fallback (9.55) If the compiler generated a message in a language other than English, for which the languages messaging file contained no translation, the English message should have been gener-

ated. This was not happening. Instead, the result would be an error claiming, *An error, warning or message was generated, but no description is available.*

Interrupt context with 16F873/4 (9.55) Fixed a problem that affected only the 16F873 and 16F874 devices where their W register could be corrupted during an interrupt if being debugged with the MPLAB ICD2.

PIC17 linker options (9.55) A number of issues have been found in the driver generated linker options when compiling for PIC17 series devices. These issues may result in *Can't find space* messages relating to common memory; or wrongly positioned psects which may cause erroneous program behaviour.

Powerup psect (9.55) An issue has been found concerning the placement of the powerup psect. When defined, this psect may overlap with others generating an error message from the linker.

Memory summary (9.55) The percentage values displayed by the driver after successful compilation are sometimes incorrect. The values displayed in the map file are valid.

Linux activation (9.55) When using a system language setting under Linux (say through the LANG or LANGUAGE environment variables), all activations will fail.

5.2 Code Generator

Pointer to signed long (9.60) Fixed a *Can't generate code error* that resulted from code sequenced that looked like this:

```
if((*signedLongPointer += constant) == x)
```

Copying of const structures (9.60) Copying the contents of a structure in program memory to a structure in RAM would generate a *Can't generate code error* if the structure was greater than 4 bytes in size. This has now been fixed.

Pointers to const structures (9.60) Fixed a *Can't generate code error* that resulted from an increment or decrement of a pointer to a structure in program memory if the structure was greater than 255 bytes in size.

if(temp32 > 24 != x) (9.60) Testing the result of a 32 bit variable right-shifted by 24 bit positions returned the wrong result. This was only the case where the result from the right shift was used for comparative purposes. Assigning the value of this right-shift to a variable may have assigned a correct value.

Bit assign bank selection (9.55) Corrected the following scenario:

```
static bankX bitX1, bitX2;
static bankY bitY;
bitX1 |= bitY; // if bitY==0, bankY remains selected
bitX2 = 0;    // assumed bankX was selected, assignment failed
```

Bitfield inversion (9.55) A syntax error may be generated when a single bit bitfield is XORed with 1 and written back to itself in the structure.

Can't generate code for function pointers (9.55) The following code will incorrectly cause a CGC error:

```
void (*func_ptr)(void);
void main(void){
    (*func_ptr)();
}
```

5.3 Assembler

IORLW 0 removal (9.60) In some cases this instruction is used to assert the Z status flag based on the contents of the working register. If a CLRF instruction appeared between the IORLW and the instruction that actually loaded the working register it was possible that the IORLW would be optimized away erroneously.

MOVF x,W removal (9.60) An optimization that removed a MOVF *x*,W instruction when the working register was known to already contain the same value as *x* should not be performed if *x* was INDF.

Unknown op else/elseif (9.60) The assembler did not recognise these directives and would produce an *unknown op* error if used.

Crash on endm (9.60) In rare cases, the assembler optimizer would crash if the source contained an *endm* directive. Whether the crash occurred or not was dependent on the code sequences that surrounded the directive.

IRP removal (9.55) A bug in the assembler may incorrectly remove instructions that clear the IRP (indirect data access) bit prior to accessing the INDF register. This bug will only occur for code that indirectly accesses objects in bank 2 or 3; that is compiled with the assembler optimizer enabled; and contains a code sequence that sequentially clears, sets then clears again the IRP bit, however not all such code sequences will trigger this bug.

Removal of bank selection instructions (9.55) Several instances of the assembler optimizer removing instructions that select the required RAM bank have been found and corrected.

ORG directive (9.55) The ORG directive was not being processed correctly. Although correctly displayed in the assembler list file, the output following the directive was not being offset by the specified amount.

5.4 Parser

#pragma regsused (9.60) This pragma was broken in the beta release (version 9.55) and has been corrected.

Structures in unions (9.55) Initialization of a structure that was the first member of a union would result in unnecessary errors. This initialization is now possible.

5.5 Preprocessor

Numbers adjoining definitions (9.55) When a number immediately preceded a preprocessor-defined symbol, and the two were not separated by white-space, and the preprocessor symbol started with the characters A-F, the preprocessor would fail to expand that symbol.

5.6 Linker

Linker % operator produces incorrect address (9.55) The % operator in the linker options which can allow placement of a psect at an address being a multiple of the % operand does not always work for some values of the operand.

5.7 Libraries

Fixup errors in persist.c (9.55) When a persistent psect was linked at the top of a RAM bank, fixup errors may be produced for the `__Hxxx` style symbols used in these routines.

5.8 Hexmate

-STRING crash (9.55) In some cases when -STRING was used incorrectly, Hexmate would crash rather than producing an error.

-FIND did not find (9.55) It was possible for the -FIND feature to fail to detect a particular code sequence if the opcode spanned over an address which was a multiple of 64.

Used memory log (9.55) If generating a logfile, used memory summaries may have included large ranges of unused memory. For this to occur a hex data source needed to have a segment of data aligned up to an address which was a (multiple of 64) minus 1, and the next address used was a multiple of 64 and was from the same hex data source. This did not affect the generated hex file in any way, only the used memory summary for that data source (in the log file) was affected.

32-Bit Checksums (9.55) If generating a 32 bit wide checksum, the result would always be zero.

Extra extended record (9.55) If a data record was being output and it was to be less than the maximum length of a data record and the record is followed by an extended address record specifying a +2 upper-address shift or greater, the first data record would be preceded by an erroneous extended address record.

Chapter 6

Addendum

6.1 Microchip errata

Refer to Microchip's website to there are any known errata issues relevant to the device that you are targeting.

6.2 Updates

Monitor [HI-TECH Software's web site](#) or subscribe to *Announcements* on [HI-TECH Software's on-line forums](#) for information relating to new versions or patches. Intermediate fixes to issues reported within this release will be posted to the *Known issues* forum relevant to this compiler. Note that the known issues will only be accessible to forum users once they have logged-in.