# Dynamic C Open Bugs List

NOTE: Feature requests and documents defects are tracked by the same tracking system, therefore many tracking numbers do not appear on the fixed or open bugs lists.

| Reference Number | Description | Work-Around | Version(s) Affected |
|---|---|---|---|
| 6 | Run-time math exceptions in watch expressions cause the target program to crash when debugging. | Don't evaluate floating point watch expressions with bad domain arguments. | 6.04-current |
| 21 | After using the print preview option with Dynamic C in full-screen mode, the taskbar will no longer automatically pop up (if set to "auto hide") until Dynamic C is exited. | Avoid using print preview if you prefer to keep your task bar as "autohide." | 6.04-current<br>fixed in a pending release |
| 160 | Constant folding does not work properly for expressions that cast int* to an integral type and then add. This defect therefore causes initializers to constant data to be evaluated incorrectly and misaligned<br>int z =10;;<br>(long)&z+1; // compiler should evaluate to constant value bug instead generates call to L_add<br>(char)&z+1; // compiler should evaluate to constant value bug instead generates call to L_add | (long)((char*)&z + 1); // semantically equivalent expression that folds correctly<br>(char)((char*)&z + 1); // semantically equivalent expression that folds correctly | 6.04-current |
| 237 | Print Preview artifact<br>1. Open a sample program.<br>2. Open print preview.<br>3. Click on the printer icon.<br>4. Open properties and adjust any of the parameters.<br>5. Accept the new parameters and close the properties window.<br>6. A print preview artifact remains. | Only make adjustments to the print properties OUTSIDE of the print preview screen (i.e. File->Print->Properties). | 6.04-current<br>fixed in a pending release |
| 399 | Execution cursor not updated in library source file when single stepping through pure assembly function. | Watch the execution cursor in the disassembly window. | 6.04 - current<br>fixed in a pending release |
| 437 | Bad code generated for a pattern of logical expression involving long*<br>long L1,L2,*pL1,*pL2;<br>// generates bad code and leaves stack imbalanced<br>( (L1 + L2) < *pL2 ); | This generates good code and the extra addition is optimized out.<br>( (L1 + L2) < *pL2+0 ); | 6.04-current |

# Dynamic C Open Bugs List

NOTE: Feature requests and documents defects are tracked by the same tracking system, therefore many tracking numbers do not appear on the fixed or open bugs lists.

| **Reference Number** | **Description** | **Work-Around** | **Version(s) Affected** |
|---|---|---|---|
| 455 | Bad stack handling with some integer expressions. For example:<br>void AFunc(void){<br>  int Var1;   auto int Var2;<br>  auto char serialno[254];//254, 255 will fail<br>  Var1 = 1;   Var2 = 15;<br>  if ( Var2 > Var1 + 1 ){// SP gets changed | Use a larger or smaller buffer, or make the buffer static. | 6.04 - current |
| 476 | using operator /= with field from structure ptr causes insufficient PUSH's when used with sizeof expression | Use sizeof(type) instead of sizeof(expression) | 7.02 - current |
| 524 | SMTP.LIB does not accept all valid variations and state codes from the server. | None | 6.53-current |
| 526 | Bug which causes incorrect conversion from unsigned long conversion to floating point:<br>void main (void){<br>        unsigned long ulTmp;    float fTmp;<br>        ulTmp = 3771964801;<br>        fTmp = (float)ulTmp;<br>        printf ( "%f", fTmp );     // prints -3771964670.000000} | None | 6.04-current |
| 532 | Character shift and test bug fails. Example:<br>The test in the while loops below continues to be true when they should actually be false.<br>main(){<br>        unsigned char i;<br>        i = 0x80;<br>        do { }     while( i <<= 1 );<br>        i = 0x80;<br>        do{ }     while( i *= 2 );<br>}} | Perform shift separately.<br>main()<br>{  unsigned char i;<br>    i = 0x80;<br>    do { i <<= 1}<br>    while( );<br>} | 6.04-current |
| 533 | Long and char comparison cause internal error. The following program generates an internal error/warning. It runs okay.<br>unsigned char UnChar(){return 1;}<br>unsigned long UnLong(){return 2;}<br>void main(){<br>  if ( UnChar() != UnLong() )<br>       printf("\nNot equal!\r"); } | None | 6.04-current |

# Dynamic C Open Bugs List

NOTE: Feature requests and documents defects are tracked by the same tracking system, therefore many tracking numbers do not appear on the fixed or open bugs lists.

| Reference Number | Description | Work-Around | Version(s) Affected |
|---|---|---|---|
| 537 | // this causes stack imbalance<br>long longfunc(void) { return 1;}<br>void main(void){ char ch;<br>( longfunc() < (ch+1) ); } | None | 6.04-current |
| 550 | Disabling FIFO for a PC COM port can cause target communications errors and DC lockups. | None | 7.10DSE-current |
| 553 | Unclosed comment blocks in BIOS can cause crash in subsequent compile.Example: | None | 7.05-current |
| 568 | printf() is limited 127 characters per call | Use buffers of at most 127 chars. | 6.53-current |
| 615 | Library's local #asm label conflicts with application's global symbol. Example:<br>The following short sample also demonstrates the bug (early versions of DC must have the "const" keywords removed):<br>char * const tail = "9876543210";<br>main(){<br>char *tailptr; int base; base = 10;<br>strtol(tail, &tailptr, base);<br>} | Avoid short common words for assembly labels. DC libraries have had short assembly labels removed. The example is no longer a problem in 7.25. A more robust is pending. An error message is now generated when conflict happens. | 6.04- current |
| 616 | Assembler should not compile ld (hl+d), a<br>The following code should generate an error, but instead generates bad code.<br>main() {<br>asm ld (hl+99), a ;} | None | 7.06 -current |
| 620 | Breakpoint in a pure assembly function can cause another window to pop up.<br>1. Open samples\tcpip\http\static.c<br>2. Add the following lines to the top:<br>#define nodebug<br>#define jr jp // Not necessary for pre-7.10<br>3. Open lib\tcipip\pktdrv.lib<br>4. Compile the program<br>5. Attempt to set a breakpoint somewhere after the _pktentry label<br>6. The REALTEK.LIB window will pop up, and the breakpoint will not be set. | None | 6.54 - Current |

# Dynamic C Open Bugs List

NOTE: Feature requests and documents defects are tracked by the same tracking system, therefore many tracking numbers do not appear on the fixed or open bugs lists.

| **Reference Number** | **Description** | **Work-Around** | **Version(s) Affected** |
|---|---|---|---|
| 624 | Bug with character evaluation in logical OR operator.. Example:<br>The value of the character gets returned instead of being normalized to zero in the following test.<br>char1 = 0;<br>char2 = 9;<br>return (char1 == 18 \|\| char2); //evaluates 9 | char1 = 0;<br>char2 = 9;<br><br>return (char1 == 18 \|\| char2 != 0); | 6.18-current |
| 630 | PPP - driver escapes bytes even when peer sets asyncmap to 0. The PPP driver will always escape byte values from 0-32 even if the peer requests to turn this behavior off. It is not harmful behavior, but will make for unnecessarily longer PPP packets. | None | 7.01-current |
| 644 | setting User Idle function in ZCONSOLE causes jump to $0 | None | 7.06P2-current |
| 647 | The address of operator does not work correctly for auto arrays. Example:<br>main() {<br>auto int ia[10]; int (*pa)[10]; int *p;<br>// should compile with warning<br>p = &ia; } | use index p = &ia[0]; | 7.05-current |
| 650 | "Soft" breakpoints do not work properly when set in an ISR. "Hard" breakpoints seem to be OK. | | 7.20-current |
| 652 | The compiler generates a "function call too long" if more than about 294 characters are used in a function call | If a literal string is making the call too long, define a pointer to it instead. | 6.04-current |
| 660 | Dynamic C does not generate an unclosed block error on the following example:<br>#ifndef NEXTTEST<br>main() {} | None | 6.54-current |
| 662 | If RabbitBIOS.C is deleted from \BIOS Dynamic C crashes. | None | 7.10DSE-current |

# Dynamic C Open Bugs List

NOTE: Feature requests and documents defects are tracked by the same tracking system, therefore many tracking numbers do not appear on the fixed or open bugs lists.

| **Reference Number** | **Description** | **Work-Around** | **Version(s) Affected** |
|---|---|---|---|
| 673 | When dumping a large amount of memory to a text file, the addresses<br>shown can be off by 128 bytes. As an example, dump 131072 bytes<br>of data starting with physical address 0x0000, then examine the file.<br>Starting at address 0x0000 looks ok, but up around 0x6000 (start<br>of XMEM) the addresses are off by 128 bytes. | None | 7.21 - current |
| 680 | Casting issue can cause infinite loop.<br>Example:<br>pkb = (uint)((TachPPS * 60L) / (*(temp[0].pValue)));// runs forever<br>where pkb is unsigned, TachPPS is the same. | None | 7.21-current |
| 687 | A very specific arrangement of arithmetic causes a run time crash. Example:<br>#define LONG_CONST 50L<br>main()<br>{<br>unsigned long ulong, result;<br>char divchar;<br>unsigned long *resultptr;<br>ulong = 100L;<br>divchar = 3;<br>resultptr = &result;<br>*resultptr = ulong/(divchar+1);//works<br>*resultptr = LONG_CONST*ulong/div-char; //works<br>*resultptr = LONG_CONST*ulong/(div-char+1); /error<br>printf("print something\n");} | None | 7.21-current |

# Dynamic C Open Bugs List

NOTE: Feature requests and documents defects are tracked by the same tracking system, therefore many tracking numbers do not appear on the fixed or open bugs lists.

| Reference Number | Description | Work-Around | Version(s) Affected |
|---|---|---|---|
| 689 | When displaying addresses in 00:e000-00:ffff in the disassembly window, the information is incorrect due to bad logic in the target communications code.<br>Workaround:<br>For addresses between 00:e000-00:efff use ff:f000-ff:ffff<br>For addresses between 00:f000-00:ffff use 01:e000-01:efff<br>When debugging assembly in this region, insert filler code to move it to a different xpc value. | See Description | 7.20-current |
| 690 | Curly brace char literal in assembly causes compile time error.<br>Example the following program should compile:<br>main() {<br>;<br>#asm<br>ld a, '{'<br>#endasm<br>}<br>Instead it generates the following error:<br>line 4 : ERROR : Unmatched '{' | Use ascii value instead<br><br>main() {<br>;<br>#asm<br>ld a, 0x7B<br>#endasm<br>} | 7.20 - current |
| 693 | int boundary case breaks division operation.<br>Example:<br>int f(int a,int b){printf("%x",(a/b));return (a/b);}<br>main(){<br>int a,b;<br>a = -32768;<br>b = -1;<br>if(f(a,b)!=32768){<br>printf("Sample Failed");<br>}<br>else{<br>printf("Sample Passed");<br>}<br>}<br>Note that the "Demotion of value" warning is appropriate when running the sample, the return of -32768 IS NOT. | None | 6.04-current |

# Dynamic C Open Bugs List

NOTE: Feature requests and documents defects are tracked by the same tracking system, therefore many tracking numbers do not appear on the fixed or open bugs lists.

| Reference Number | Description | Work-Around | Version(s) Affected |
|---|---|---|---|
| 694 | For double variables -0.0 < 0.0, Boundary condition exposes error. Example:<br>foo (double d)<br>{<br>  d = -d;<br>  if (d < 0.0)<br>    printf("Test Failed");<br>  else<br>   printf("Test Passed");<br>}<br>main ()<br>{<br>  foo (0.0);<br>} | None | 6.04-current |
| 699 | Bad error message for invalid symbol name.<br>The first error should say<br>line    4 : ERROR Untitled     : Illegal symbol name ’13’.<br>#define CR 13<br>main()<br>{  char CR;}<br>line    4 : ERROR Untitled     : Illegal symbol name ”.<br>line    4 : ERROR Untitled     : Missing character ’;’.<br>line    4 : ERROR Untitled     : Bad declaration: ’,’ ’;’ or ’=’ expected. | None | 7.25 - current |
| 720 | Dynamic C’s default behavior is correct for constant union variables, if the initializer is enclosed in curly braces: the initalizer is casted (with no conversion) to the type of the _first_ element in the declaration list of the union type. However, mixing float and integer types in a union can cause strange behavior with initializers (this is due to the differing internal representation of float and integer types, but the behavior is correct) . Dynamic C allows non-brace-enclosed initializers for constant union variables, which is incorrect. | None | 6.04-current |
| 721 | The functions HDLCopenE and HDLCopenF do not return correct values. The variable baud ratio is never calculated. | None | 7.25-current |

# Dynamic C Open Bugs List

NOTE: Feature requests and documents defects are tracked by the same tracking system, therefore many tracking numbers do not appear on the fixed or open bugs lists.

| Reference Number | Description | Work-Around | Version(s) Affected |
|---|---|---|---|
| 733 | No Error messages received when using an ellipsis in an indexed cofunction. | Do not use ellipsis's within an indexed cofunction for they are not supported | 6.04-current |
| 735 | Certain macros containing ' are parsed incorrectly, produce unnecessary error report. Example:<br>The first of the following similar macro sets is not parsed correctly:<br>// error messages follow:<br>#define to_decimal(x) (x-'0')<br>#define htod(x) (x-'A'+10)<br>// no error messages follow (notice the added spaces):<br>#define to_decimal2(x) (x - '0')<br>#define htod2(x) (x- 'A' + 10)<br>main(){} | See Description. | 7.25-current |
| 753 | Trying to compile to files with Ramonly-bios.C does not work and results with error warnings. | None | 7.02P-current |
| 758 | Commenting out a middle line in a multiline macro definition can cause confusing erroneous error messages. | None | 6.04-current |
| 760 | The error below is not reported on the correct line and can open and report the error in another file.<br>#define SID(sid, s) xstring sid {s};<br>SID(SIDDsipDispatch1, "DispReadThr: dropping 199 msg timeout reading<br>value\n); // <-- missing "<br>SID(SIDDsipDispatch2, "DispReadThr: dropping 198 msg timeout reading<br>bodyLen\n");<br>void main()<br>{} | None | 7.25-current |

# Dynamic C Open Bugs List

NOTE: Feature requests and documents defects are tracked by the same tracking system, therefore many tracking numbers do not appear on the fixed or open bugs lists.

| **Reference Number** | **Description** | **Work-Around** | **Version(s) Affected** |
|---|---|---|---|
| 772 | Using "Inspect > Dump at Address > Save Entire Flash to File" results in a binary file that is highly repetitive and incomplete.<br>The first 128 bytes of flash content is repeated through the binary file's 1st 4K, then the flash's next (presumably) 128 bytes is repeated through the following 4K, and so on up until the size of the flash. | Use Dump to File (specifying Hex Address 0, # Bytes (Dec.) as the decimal size of the entire flash, and the File name) to generate a text dump file. Then (if necessary) write a converter program to translate the text dump file to a binary file. | 7.20 - current |
| 774 | For useix functions ix may corrupt stack.<br>Example:<br>`foobar(int left){}`<br>`nodebug`<br>`root useix void foo(int left) {`<br>`    long l;`<br>`    l = 0;`<br>`    if((left&0x07) == 0){`<br>`       foobar(left);`<br>`      return; // ld sp, ix generated is wrong`<br>`    }`<br>`    #asm`<br>`        ld ix, 0x9000`<br>`    #endasm}` | Don't use useix since its new rabbit instructions using sp are better anyway. | 6.04-current |
| 775 | The compiler allows arrays larger than 64K bytes for 4 byte types. | None | 6.50-current |
| 776 | RS232 - serXclose should restore output. Closing a serial port should return it's corresponding TX pin back to a normal output. | None | 7.30 |

# Dynamic C Open Bugs List

NOTE: Feature requests and documents defects are tracked by the same tracking system, therefore many tracking numbers do not appear on the fixed or open bugs lists.

| Reference Number | Description | Work-Around | Version(s) Affected |
|---|---|---|---|
| 778 | Long type with built-in function bit and #useix doesn't work<br>The following program generates bad code for the bit test.<br>#useix<br>main()<br>{   int i,oldi;<br>   long l;<br>   i = 0;<br>   oldi = i;<br>   set (&i, bit(l,1));<br>   if(i==oldi){printf("error\n");}} | Use assembly code or don't use useix. | 6.04-current |
| 782 | Watches broken when using function pointers in xmem with an ximport. Example:<br>Steps to reproduce:<br>1. Compile the program below.<br>2. Add watch for 'test' function<br>3. Recompile program.  Note that test does not reappear as it should<br>4. Add test watch again and DC will lose target communication and may crash.<br>Ximport, xmem, function pointer, and the order of the code all appear to matter.<br>#memmap xmem<br>#ximport "samples/tcpip/http/pages/rabbit1.gif"    rabbit1_gif<br>int test();<br>typedef int (*fcnptr)();<br>const fcnptr erg = test;<br>void main()<br>{}<br>int test(){}; | Two possible work-arounds:<br>1. Change order of code.  For example, put test function before function pointer and watches will work again.<br>2. Exit DC. Remove .dbg file associated with .C file.  Reopen DC and recompile. | 7.20-current |

# Dynamic C Open Bugs List

NOTE: Feature requests and documents defects are tracked by the same tracking system, therefore many tracking numbers do not appear on the fixed or open bugs lists.

| Reference Number | Description | Work-Around | Version(s) Affected |
|---|---|---|---|
| 783 | The following code should not compile because the types conflict.<br>extern int array[512];<br>extern int array[0];<br>main() {<br>} | None | 6.04-current |
| 785 | Some (odd) opcodes don't display properly in asm window when db'd in (may cause crash). There are two sets of opcodes for "LD HL, (mn)". The standard (shorter) one that is generated by the compiler is "0x2A mn" and displays correctly. The longer one is "0xED 0x6B mn" and does not display correctly.<br>As an example,<br>#asm<br>db 0xED, 0x6B, 0x24, 0x00<br>#endasm<br>should display<br>ED6B2400 LD HL, (0024)<br>but instead displays<br>ED LD HL, (0024)<br>6B LD L, E<br>24 INC H<br>00 NOP | None | 7.05P2 - current |