DIGI INTERNATIONAL
9350 Excelsior Blvd, Suite 700
Hopkins, MN 55343, USA
+1 (952) 912-3444 | +1 (877) 912-3444
www.digi.com

Dynamic C 10 Release Notes

Dynamic C 10

Version 10.72E (August 2020)

INTRODUCTION

These are the release notes for Dynamic C 10.

Dynamic C 10 is the IDE (Integrated Development Environment) for Digi International's core modules and single-board computers based on Rabbit 4000, 5000 and 6000 processors.

The IDE includes a graphical editor and debugger, a command-line compiler, and a graphical tool (Rabbit Field Utility) for installing compiled binary firmware to a serially-connected device.

SUPPORTED PRODUCTS

- RCM4000, RCM4010, RCM4050
- RCM4100, RCM4110, RCM4120
- RCM4200, RCM4210
- RCM4300, RCM4310, RCM4320
- RCM4400W
- RCM4510W
- RCM5400W, RCM5450W
- RCM5600W, RCM5650W
- RCM5700, RCM5710, RCM5750, RCM5760
- RCM6600W, RCM6650W
- RCM6700, RCM6710, RCM6720, RCM6730, RCM6750, RCM6760
- BL4S100, BL4S110, BL4S150, BL4S160
- BL4S200, BL4S210, BL5S220, BL4S230

KNOWN ISSUES

1. Debugger issues

    - When debugging a program using either F7 or F8 into or over a call to `kbhit()` will cause the debug cursor to disappear. Pressing F7 or F8 two more times will make the debug cursor reappear following the `kbhit()` call.

    - The breakpoint highlight for the closing brace of a function does not display correctly if the brace is not in the first column of the editor window.

    - The debugger currently only supports 128 breakpoints. Setting more than 128 breakpoints will result in the loss of target communications.

- The debugger does not support assembly language single stepping over function calls that do not return to the instruction following the call such as "_switch". Any attempt to do so will result in loss of target communication. Stepping *into* the function in assembly is supported, as is stepping over such functions in C.
- If an `ipset` or `ipres` instruction is followed immediately by an `rst 0x28`, the debug kernel will not step over the `rst 0x28` correctly, but will end up in an unexpected location. When either of these instructions are followed by instructions other than `rst 0x28`, the debug kernel behaves correctly.
- The Register Contents debug window (F11) doesn't report the correct LXPC value if it's >= 0xFF. It reports 0x0FF as "LXPC: FFF" and leaves the high nibble off of values >= 0x100 (so 0x123 appears as "LXPC: 023").

2. Software modules.

   RabbitWeb gives a cryptic error message for buffer overflow when using the `print_select()` statement. The workaround is to make `HTTP_MAXBUFFER` large enough to hold all of the `OPTION` statements generated by the `print_select()` statement.

3. Complex casts

   The compiler will not accept multidimensional array types or derivatives thereof as cast expressions, even if the cast expression is a typedef name. Use a union to work around this problem.

4. Advanced 16-bit memory mode CPU bug

   The Rabbit 4000 CPU's advanced 16-bit memory mode has a defect which affects ioe instructions (auxiliary I/O, external I/O) and self-timed chip select. The Dynamic C BIOS has been updated to work around this ioe defect on affected boards, all of the RCM40xx family. If absolute top performance is required and the User is certain their application is unaffected by the ioe bug, the work around can be disabled by adding the `__ALLOW_16BIT_AUXIO_DEFECT` macro into Dynamic C's Project Options Defines Box. See the Rabbit 4000 Users Manual Appendix B (errata section) or TN255 for complete details.

5. In separate I/D mode, an array declared as

   ```
   far char bar[] = {0,1,2,3,...};
   ```

   will not compile correctly when the size of the array exceeds the size of the root constant space and will give an error that the '}' character is missing. The work-around is to specify the size of the array as follows:

   ```
   far char bar[32000] = {0,1,2,3,...};
   ```

6. Highlighting a block of text while in Stop mode

   In some cases, depending on the specific text, highlighting a block of text can result in a flyover watch which returns a copy of that text in the hint window.

7. Using `#pragma SIZEOF32` can result in a `sizeof` problem for a large struct.

   Using `#pragma SIZEOF32` allows the argument of `sizeof` to be up to 4GB but will return an incorrect value without warning or error if the argument involves a struct of size 32K or larger. When using `#pragma SIZEOF32`, always restore to the default with `#pragma SIZEOF16` following the use of sizeof.

8. Initialized global and static variables and `#GLOBAL_INIT` expressions run with a limited stack (256 bytes). Deeply nested functions (including `printf()`) will cause a loss of target communications.

9. Cast ternary operator(s) if both values do not have the same type and type qualifiers - example: `condition ? &aFarChar : (far char *)NULL`. In the example, if condition is true and `NULL` is not cast, a near address is erroneously returned.

10. As of 10.66, symbols are distinguished using the first 63 characters of the name. Older versions used 31 characters, the minimum per the ANSI specification. If a long symbol name is used as a library module label (i.e., in a BeginHeader statement), the library's .MD1 file will contain the short version of the name, and the module will not be found. For example:

    ```
    /*** BeginHeader A12345678911234567892123456789312 ***/
    extern int A12345678911234567892123456789312;
    /*** EndHeader ***/
    int A12345678911234567892123456789312;
    ```

will generate the error `line  XXX : ERROR MY_LIB.LIB : Undefined (but used) global label A12345678911234567892123456789312`. The workaround is to change the timestamp on the library file (e.g., modify it and save it) and recompile. The .MD1 file will be regenerated with the correct (longer) module label.

11. More than one `return` statement in a cofunction will result in unpredictable behavior.

12. For `auto int i`, the expression `(far*)&i` should be `(far int*)&i`, but the first form will produce confusing error messages:

    ```
    Invalid expression.
    Missing character ';'.
    Missing character ')'.
    Not a pointer, cannot dereference.
    ```

13. A `c return` statement in an assembly block can produce an error like `Undefined (but used) global label .DCLAB__ZW00000190` in large programs. The assembly block can be ended followed by the `return` statement in C followed by the remaining assembly in a new assembly block.

14. The active status of hardware breakpoints is not retained when a program restarts. Each breakpoint must be modified in some way such as toggling a checkbox so that the “Update” button becomes active.

15. Setting a hardware breakpoint on some internal I/O addresses can lead to a target communication error. Since setting a breakpoint mask to 0xffffff will include all internal I/O address, the address and mask should be set to include only the intended range of addresses.

16. Programs larger than 512K will not compile to RAM successfully on an RCM5450W core module. Compiling such programs to flash works fine.

17. Number of dimensions in array initializer are not checked against the array declaration.

    Dynamic C will compile an initializer with too few dimensions without complaining:

    ```
    // two few dimensions in initializer
    const static int a[][3][2] = {{11, 12}, {21, 22}};
    ```

    The results are undefined. If there are too many dimensions in the initializer, Dynamic C will indicate an error `} is missing/expected`.

18. For functions with both a near and far syntax, function help is available only using the unadorned name. For instance, placing the cursor over `_f_memset` will not find the function help for `memset`. In order to find function help for a function prepended with either `_n_` or `_f_`, such as `_f_memset`, separate the main part of the function name before pressing `Ctrl+h`.

19. A function that returns a struct by value does not work correctly within a cofunction. A work around is to have the function return a static struct by reference and then dereference the function result on assignment in the cofunction, e.g.:

    ```
    struct point { int x; int y; };

    struct point *foop(void) {
       static struct point tmp;

       tmp.x = 3;
       tmp.y = 4;
       return &tmp;
    }

    cofunc void co_func(void) {
       struct point tmp;

       tmp = *foop();   // dereference the result at time of assignment
       printf("cofunction by reference, x=%d, y=%d\n", tmp.x, tmp.y);
    }

    void main(void) {
       struct point tmp;

       costate {
          wfd co_func();
       }
    }
    ```

20. An index variable used in calling an indexed cofunction cannot be auto. The index variable must have static (local) or global storage, e.g.:

```
#define INDICES 5

cofunc void my_icofunc[INDICES](void)
{
   ;
}

void main(void)
{
   static int i;

   while (1) {
      for (i = 0; i < INDICES; ++i) {
         costate {
            wfd my_icofunc[i]();
         }
      }
   }
}
```

21. Auto pointers to CoData structures cannot be used in named costatements. The CoData structure pointer must have static (local) or global storage, e.g.:

```
#define NUMTASKS 5

void main(void)
{
   auto int i;
   static CoData task[NUMTASKS];
   static CoData *ptr2codata;

   for (i = 0; i < NUMTASKS; ++i)
      CoBegin(&task[i]);

   while (1) {
      for (i = 0; i < NUMTASKS; ++i) {
         ptr2codata = &task[i];
         costate ptr2codata always_on {
            ;
         }
      }
   }
}
```

22. Dynamic C currently does not allow function pointers to return structures by value. The following code snippet will generate an error:

```
typedef struct { int a; } my_struct_t;
my_struct_t (*func_ptr)();
main() {
   my_struct_t s;
   s = func_ptr();
}
```

23. Assembly labels within `#asm` blocks inside C functions are treated as if they are local C-labels, rather than existing within their own scopes. In the following code, the expected behavior would be an “unknown label” error on the `jp` to `foo` since there is no global label `foo` defined. However, the linker associates the local label with the `jp` instruction and the code compiles.

    Note that this behavior only occurs between `#asm` blocks within C functions. It does not exist between stand-alone `#asm` blocks or between blocks in separate C functions.

```
void func(void) {
   #asm
      jp foo  ; This jumps to the label in the next block
   #endasm
   // Some C code
   #asm
      foo:    ; The jp above lands here
   #endasm
}
```

24. The `strtod` function’s underflow detection fails when converting strings of the following forms and lengths:

    ```
    ".0000000000000000000000000000000000000001"
    ".00000000000000000000000000000000000000001"
    .  .  .
    ".00000000000000000000000000000000000000000000001"
    ".1e-38
    ".01e-38"
    .  .  .
    ".00000000000000000000000000000000000000001e-38"
    ```

25. Dynamic C’s assembler does not support embedded C return statements (e.g. `c return;` or `c return X;`) within an asm block that is within a C function. All such occurrences have been removed from standard libraries (i.e. DMA.LIB, FFT.LIB) and no occurrences exist in standard samples code. Custom assembly code with embedded C return statements as described above should be rewritten such that any explicit or implicit return from the C function is done in standard C code.

26. Complex macros inside a one-line for-loop may fail to compile, with an internal error (`Internal error: input stream non-existant. Compiler     confused. Look for previous syntax error.`) that requires Dynamic C to be restarted.

    Work-around: replace

    ```
    for (i = 0; i < max; i++){ complex_macro; }
    ```

    with

    ```
    for (i = 0; i < max; i++){
       complex_macro;
    }
    ```

27. When compiling an initialized array of unspecified dimension(s) in a nested scope, Dynamic C does not generate a jump over the initializer data, which results in errant execution of the initializer data. One work around is to avoid declaration of such arrays in nested scope. Another work around is to ensure all initialized arrays in nested scope have fully specified dimensions.

    Work-around example: replace

    ```
    while (1) {
       char *foo[] = { "123", "abc" };
       printf("%s   %s\n", foo[0], foo[1]);
    }
    ```

    with

    ```
    while (1) {
       char *foo[2] = { "123", "abc" };
       printf("%s   %s\n", foo[0], foo[1]);
    }
    ```

28. The compiler incorrectly throws an error (`Duplicate 'const' keywords not     allowed in this context.`) if a function contains multiple, non-pointer `const` parameters.

    Work-around example: replace

    ```
    void foo(const int bar, const int baz) {}
    ```

    with

    ```
    typedef const int const_int;
    void foo(const_int bar, const_int baz) {}
    ```

29. On platforms with an ASIX Ethernet controller (e.g., RCM42xx, RCM43xx), the network stack can get stuck in an infinite loop after executing the sequence `ifdown()/pd_powerdown()/pd_powerup()/ifup()`. Issue present since at least Dynamic C 10.60, and might only happen when calling `tcp_tick()` while interface is powered down, or calling `ifup()` too soon after `pd_powerup()`. If this affects you, contact Digi and reference issue DC-300.

30. The compiler incorrectly reports errors and warnings as occurring on line 1 if they are beyond line 32,767 of a file. Limit file sizes by creating .LIB modules loaded with `#use`, or .H header files loaded with `#include`.

31. Compiler will report `Cannot load object, report to Rabbit.` when trying to compile `(0 && j == 0)` in the code below:

    ```
    int foo(int i, int j) { return (i == 0 || (0 && j == 0)); }
    ```

This can occur when using constant macros to short-circuit conditionals at compile time. The workaround is to adapt the conditional using the macro preprocessor. In the example above:

```
int foo(int i, int j) {
    return (i == 0
        #if FOO
            || (j == 0)
        #endif
           );
}
```

32. The "Start" button may be hidden on the main Rabbit Field Utility screen when using certain Windows Display Scaling values (seen at 125% but not 199%).

33. The command-line compiler fails on project files that contain multiple .C files. As a workaround, compile a single .C file that references either libraries (with the `#use` directive) or .H files (with the `#include` directive) for additional code.

UPDATE CONSIDERATIONS

Reminder: When opening a project in a new version of Dynamic C, update the "Include Path" tab in the Project Options to use the new Dynamic C directory. (Or change it to `.\include` to always use the current compiler's include directory.)

If updating an existing installation of 10.72D or earlier via the Dynamic C 10 GitHub repository, you should manually apply the changes in `ucos2.patch`, found in the Dynamic C installation directory.

If using I2C with Dynamic C 10.72B on Rabbit 6000 products, you must update your calls to `I2CRead()` and `I2CWrite()` to pass the slave address in bits 1-7 instead of bits 0-6 of the first parameter. This release restores compatibility with software-based I2C from Dynamic C 10.72A and earlier. In addition, the APIs for `I2C_ReadFromSlave()` and `I2C_WriteToSlave()` now include an index parameter so `I2C_ReadFromSlave()` can send a START, SLAVE_WRITE, INDEX, START, SLAVE_READ sequence for efficient reads. If necessary, use the new `CC_REV` compiler macro to conditionally compile code so it can work with new and old APIs.

If updating iDigi-enabled applications to Dynamic C 10.72B or later, you *should* update all IDIGI/idigi references to CLOUD/cloud and replace `#use "iDigi.lib"` with `#use "Device_Cloud.lib"`. All of the iDigi libraries, APIs and samples were renamed to match the (then) current "Device Cloud" name for that service. `iDigi.lib` remains with macros mapping old names to new ones.

UPDATE BEST PRACTICES

Digi recommends testing the new release in a controlled environment with your application before you deploy firmware to production devices.

TECHNICAL SUPPORT

Get the help you need via our Technical Support team and online resources. Digi offers multiple support levels and professional services to meet your needs. All Digi customers have access to product documentation. firmware, drivers, knowledge base and peer-to-peer support forums.

Visit us at https://www.digi.com/support to find out more.

CHANGE LOG

10.72E (August 31, 2020)

This is a recommended release.

Note that the bugfix for DC-391 below is included in the Dynamic C 10.72E installer, but included as a patch in the GitHub repository due to licensing issues with the uC/OS-II code. View `ucos2.patch` for instructions on how to apply it.

NEW FEATURES

- Add sample with default calibration constants for BL4S2xx SBCs.

- Add `Samples/RCM4200/Serial_Flash/SFLASH_ERASE.C` to completely erase serial flash.

BUG FIXES

- DC-319: Only cache ARP info for unicast UDP frames (ignore broadcast/multicast) in anticipation of responding.
- DC-391: Usage of `OS_TASK_OPT_STK_CLR` with `OSTaskCreateExt()` would erase an extra byte before the stack, and leave out the last byte of the stack (see `ucos2.patch`).
- DC-428: Increase TLS Record Limit from 16KB+53 to 16KB+256 (maximum specified in TLS 1.3).
- GITHUB-16: Fix `X509.LIB` bugs parsing BasicConstraints (2.5.29.19).
- GITHUB-17: Increase `SSL_MAX_HANDSHAKE_SIZE` to allow 4096-bit key in client key exchange.
- GITHUB-19: Fix `I2CRead()` length calculation in `I2C_DEVICES.LIB`.
- GITHUB-20: Correctly reference far pointer in `httpc_set_extra_headers()`.
- GITHUB-23: Fix field in `Samples/RemoteProgramUpdate/pages/upload.zhtml`.
- GITHUB-25: Don't allow expiration of existing ARP entry in `router_add()`. Ensures that we don't flush entries for routers added via `ifconfig()`'s `IFS_ROUTER_SET`.
- Create Function Help (`CTRL-H`) for `MP_SIZE` macro (SSL/TLS related).
- Document command-line compiler errors with project files containing multiple C source files as "Known Issue #33".
- Fix reporting of write errors in `Samples/RemoteProgramUpdate/upload_firmware.c`.

10.72D (April 13, 2018)

This is a required release for Wi-Fi products, and recommended for all others.

The primary focus of this release is to address the Key Reinstallation Attacks (KRACK) against WPA2. Dynamic C includes two versions of wpa_supplicant – v0.2.8 used when WPA_USE_EAP isn't defined and v0.5.11 for when it is. This release updates to the last stable release of each series and patches them to block the KRACKs.

The release also addresses routing problems present on RCM66xxW modules that enable both Ethernet and Wi-Fi interfaces.

SECURITY FIXES

- DC-349: Fix WPA2 KRACK vulnerability.
  - CWE-323 - Reusing a Nonce, Key Pair in Encryption.
  - CVE-2017-13077 - reinstallation of the pairwise key in the Four-way handshake
  - CVE-2017-13078 - reinstallation of the group key in the Four-way handshake
  - CVE-2017-13079 - reinstallation of the integrity group key in the Four-way handshake
  - CVE-2017-13080 - reinstallation of the group key in the Group Key handshake
  - CVE-2017-13081 - reinstallation of the integrity group key in the Group Key handshake
  - CVE-2017-13082 - accepting a retransmitted Fast BSS Transition Reassociation Request and reinstalling the pairwise key while processing it
  - CVE-2017-13084 - reinstallation of the STK key in the PeerKey handshake
  - CVE-2017-13086 - reinstallation of the Tunneled Direct-Link Setup (TDLS) PeerKey (TPK) key in the TDLS handshake
  - CVE-2017-13087 - reinstallation of the group key (GTK) when processing a Wireless Network Management (WNM) Sleep Mode Response frame
  - CVE-2017-13088 - reinstallation of the integrity group key (IGTK) when processing a Wireless Network Management (WNM) Sleep Mode Response frame

BUG FIXES

- DC-255: Updated change from 10.72B release related to IP address reuse during PPP handshaking. Users can re-enable the disabled feature by defining `IPCP_ENABLE_IPADDR_REUSE`.
- DC-337: Store correct value for `freq_divider`. Was previously one less than the correct value which would affect baud rate calculations.
- DC-344: Fix error when setting default Wi-Fi router when also using Ethernet.
- DC-345: Fix behavior of `IFG_WIFI_MODE` option to `ifconfig()`.
- DC-346: Fix default route when using both Ethernet and Wi-Fi.
- DC-357: Document compiler error (aborts compilation of valid code) as "Known Issue #31".
- Document Rabbit Field Utility UI issue when using Display Scaling as "Known Issue #32".

OTHER FUNCTIONAL CHANGES

An I2C-related change from 2017-08-30 was erroneously tagged as included in the 10.72C release. The shipping installer for 10.72C did not include the commit and it first appeared in this release.

10.72C (August 16, 2017)

This is a recommended release.

NEW FEATURES

- Adds support for RCM4320 variant – an RCM4310 with 1MB of SRAM for firmware (instead of 512KB) and a 4MB serial flash for storage (instead of 1MB).
- Updated `sdflash.lib` to support SDHC cards from 4GB to 32GB. Note that Dynamic C still does not support FAT32. When using SDHC cards you must format them on the Rabbit with up to four FAT16 partitions, but you can still read and write them on Mac/Windows/Linux. Email Jim.Pause@digi.com if interested in FAT32 support in Dynamic C.
- Updated `FAT16.lib` to work on SDHC cards (larger than 2GB).
- Defined a `CC_REV` macro as 'C' for use in compiler version checks.
- Added RCM6xxx support for Panel-Mount Keypad/Display Unit (Digi part number 20-101-0502).

ENHANCEMENTS

- Updated TLS to send Server Name Indication (SNI) in ClientHello. Includes updates to HTTP, SMTP and POP3 client libraries.
- Updated X.509 certificate handling to process `subjectAltName` extension, allowing for improved hostname matching in TLS client connections.

BUG FIXES

- Addresses compiler errors with `I2C_HW.LIB` for Rabbit 6000 boards.
- HTTP client now sends correct Content-Type on non-blocking PUT/POST.
- Moved duplicated `MODBUS_CRC()` from `modbus_{master,slave}.lib` to `CRC16.LIB`.
- Updated FTP server samples to work with default FileZilla FTP client settings.
- DC-295: `Get_Stack_Size()` returns correct size instead of smallest stack size.
- DC-304: After DHCP fallback, further DHCP discovery would fail.
- DC-310: Revert commit that incorrectly processed slave address in `I2CRead()` and `I2CWrite()` calls on Rabbit 6000 hardware with DC 10.72B.
- DC-317: Correctly set interrupt level when `RS232_INT_LEVEL != 1`.
- DC-322: Don't hard-code keysize in `AES{en,de}cryptStream_CBC()` and `_CBC_XMEM()` functions. Now works correctly for AES-192/256.
- DC-332: Fix Rabbit 6000 link recovery, broken in pre-release 10.72C.
- DC-334: Ignore non-fatal "warning" TLS alerts instead of aborting connection.
- GITHUB-13: Fix return value of `_n_strcpy()`.
- GITHUB-14: Fix bug in `httpc_skip_headers()` for non-blocking connections.
- RCM-143: Only call `dhcp_tick()` if interface is up.
- RCM-143: When Ethernet powered down, link status should always be "down" (changed for Rabbit 6000 and ASIX-based hardware).

OTHER FUNCTIONAL CHANGES

- Disabled incomplete I2C slave mode for Rabbit 6000 hardware.
- Please note that while all Rabbit MAC addresses have previously used an OUI (manufacturer code) of 00:90:C2, future products may ship with an alternate OUI. Please be proactive and replace any code that requires that byte sequence.
- Added "Known Issue" about compiler errors/warnings in files that exceed 32,767 lines.

10.72B (October 31, 2016)

This is a recommended release.

This release builds upon the TLS 1.2 update in Dynamic C 10.72A, by adding support for SHA384 and SHA512 hashes in TLS communications and X.509 certificates. It also brings back support for TLS 1.0 in outbound client connections to support legacy servers that have not implemented TLS 1.2 yet. Details in function help for `TCPCONFIG`, `tls_set_flags()`, `httpc_set_tls()`, `pop3_set_tls()`, and `smtp_set_tls()`.

There's an important update to `ASIX.LIB` which should finally resolve all outstanding link recovery issues.

NEW FEATURES

- Adds support for SHA384/SHA512 hashes:
  - New library and updated sample.
  - TLS update to support for SHA384 and SHA512 hashes.
  - X.509 update to support sha224, sha384 and sha512 signatures.
- Adds support for falling back to TLS 1.0 on outbound (client) TLS connections.
- Adds source code and tools for building binaries in BIOS directory (cold loaders and pilot BIOSes).
- Adds sample for run-from-RAM boards to download a copy of the boot flash via HTTP. Useful for firmware recovery and analysis of flash contents.
- Improves boot time on 6000-based hardware (now < 1 second) when `_FAST_BIOS_LOAD` macro defined in Project Options [DC-269].
- Replaces the outdated SSL Certificate Creation GUI (in `Utilities/SSL_Utilities`) with command-line OpenSSL version 1.0.2j. Added instructions on using OpenSSL to create certificates, which should allow for more control over key size and choice of hashing algorithm.

BUG FIXES

- Corrected compilation errors when `X509_VERBOSE` or `WPA_VERBOSE` defined.
- Fix bug in `http_finderrbuf()` affecting variable names longer than `SSPEC_MAXSPEC` characters.
- DC-216: Update I2C for Rabbit 6000 hardware.
- DC-255: Disabled IP address reuse during negotiation of PPP connections with other active interfaces (e.g. Ethernet or Wi-Fi).
- DC-260: Improve ASIX Ethernet link recovery by allowing more time to establish link after turning PHY back on (8s vs. 2s).
- DC-264: IIS servers require `signature_algorithms` extension in TLS Client Hello.
- DC-264: Increase TCP receive buffer size to account for additional TLS 1.2 overhead.
- DC-267: FTP server incorrectly stripped leading slash from absolute pathnames in STOR, MKD and RMD commands.
- DC-275: Disable attempted lease renewal for permanent DHCP leases.
- DC-280: Fix for serial connections with 2 stop bits. First byte was always sent with 1 stop bit.
- DC-281: Correct invalid JSON generated by `web_iter_next()` (RabbitWeb).
- DC-283: Repeatedly setting a gateway then deleting all routers no longer fills APR cache.
- DC-285: Fix possible overflow when calculating PMKSA reauth timeout for WPA/WPA2 Enterprise Wi-Fi connections.
- DC-287: Fix for FTP client uploading files using a datahandler instead of a fixed-length buffer (broken by DC-129 in Dynamic C 10.72A).
- DC-290: Ensure that `SSL_set_private_key()` always returns error upon failure to parse key.
- GITHUB-7/DC-265: Handle Ethernet netport interrupt on 6000-based hardware so transmit errors don't hold buffers forever.
- RCM-77: Improve Ethernet link recovery on 5000/6000-based hardware.
- RCM-121: Fix buffer overflow caused by large packets on WPA/TKIP (but not WPA2/CCMP) networks.

OTHER FUNCTIONAL CHANGES

- Update samples using `SSL_Cert_t` to store certs in far memory and not on the stack.
- Optimization from petermcs to reduce code size and speed execution of `sha256_process()` in `SHA2.LIB`.
- DC-268: Changed some function variables in FTP server from "static" to "auto".
- DC-284: Updated `firmware_info_t` structure to include compiler revision.

10.72A (March 17, 2016)

This is a recommended release.

The big change for this release is that Dynamic C is now an Open Source project on GitHub. It includes source code to the previously-encrypted Wi-Fi and SSL libraries, and the license was changed to MPL 2.0 (Libraries) and ISC (Samples).

This release was made as 10.72A since it does not include changes to the compiler and only contains modified libraries, samples and documentation. It uses shortcuts (`.lnk` files) to launch the compiler with a command-line parameter of `-c A`, forcing the compiler to identify itself as 10.72A and use registry entries for version 10.72A.

The installer is also smaller, due to the removal of an outdated XCTU installer, the "XBee GPIO GUI" utility/source (designed for sample programs removed in the 10.70 release), and a Microsoft .NET installer (`dotnetfx.exe`) required by that utility.

The latest version of XCTU is on Digi's website.

NEW FEATURES

- Includes library and utility programs to write a System ID Block to a device's flash (`Utilities/Write_ID`).
- Incorporates NAND flash library patch (40002851_A, 2012-12-01) for Micron/Numonyx/ST NAND256W3A (32MB) devices used in new hardware.
- Incorporates Serial Flash update (40002882_A) for revision E of 45DB641 chips used in new hardware.
- Add `httpc_set_extra_headers()` API to `HTTPC.LIB` for user code to add headers to outbound requests.
- Upgrades TLS (Transport Layer Security, aka SSL/HTTPS) implementation from 1.0 to 1.2.
  - Is compatible with modern web browsers.
  - Removes support for insecure SSLv2, SSLv3 and TLS 1.0 protocols.
  - Removes support for insecure RC4 cipher and MD5 digest.
  - Automatically enables required AES128_CBC cipher.
  - Defaults to 2048-bit RSA keys (previously 1024-bit).
  - Adds optional AES256_CBC cipher, SHA256 digests, and support for SHA256 signatures in X.509 (TLS/SSL) certificates.

BUG FIXES

- Fix conversion of RabbitWeb multi-select 32-bit enum to text. (contributed by titobrasolin).
- Initialize `struct tm` in `_atodt()` before use to avoid random hour/minute/second values affecting result (contributed by titobrasolin).
- Fix return value of `asix_ioctl()` for unsupported features. Was previously returning uninitialized value from stack instead of 0.
- DC-6: Fix rebalancing of HTTPS Rx and Tx transport buffers, enabling HTTPS upload of large files.
- DC-10: Fix PPP baud rate calculations, corrects problem with 9600 baud.
- DC-19: ASIX PHY link-fail recovery now works after extended cable disconnection (20 minutes or more).
- DC-27: Improved `SPI.LIB`'s `SPIWrRd()` function's Rabbit 6000-specific work around.
- DC-31: `HTTP.LIB`'s `zhtml_handler()` function now correctly handles `HttpState` (unsigned) `extlen` values greater than 32767.
- DC-55: Fixed multiple stacks (as when using e.g. uC/OS-II multi-tasking) misplacement bug affecting Dynamic C versions 10.62 through 10.72, inclusive.
- DC-78: Fix memory leaks in `idigi_put()` and `idigi_upload()`.
- DC-117: `PPP.LIB`'s `LCPsendEchoReply()` now echoes the request's non-zero length data.
- DC-129: Fix `FTP_CLIENT` uploading files smaller than TCP socket buffer size.
- DC-130: SNMP: Correctly store zero-length octet strings and prevent xmem window wraparound (in all data formats).
- DC-215: Fix keepalive handling in `tcp.lib`.
- DC-200: Update some incorrect macro names (contributed by Richard Pletcher).
- DC-200: Update start of write buffer when adjusting TCP buffer split between read/write. Possible fix for broken HTTPS Upload (contributed by Richard Pletcher).
- DC-207: Correctly render SSI/RabbitWeb (shtml/zhtml) tags spanning multiple 256-byte blocks in files on FAT filesystem.
- DC-217: Add full HDLC support for Rabbits 4000-6000, including sample program.
- DC-220: Added test for non-zero length in `tcp_write()`, for retransmission.
- DC-224: Fixed corruption of register `ix` around `_pb_free` call in `SERLINK.LIB`.
- DC-235: Corrects an error where the HTTP server would fail to find files on FAT filesystem due to stale data in `SSpecFileUnion` structure allocated from a pool of memory.
- DC-241: Function `sdspi_process_command()` in `SDFLASH.LIB` wasn't releasing the SPI semaphore (contributed by Mark Leichty).
- DC-248: Update `HTTPS_CLIENT.c` to follow URL redirects and correctly print `far` hostname from `httpc_Socket`.
- GITHUB-1: Correct invalid `strtol()` and `strtoul()` behavior.
- GITHUB-2: Fix `%g` formatting error for floats smaller than 0.1.
- RCM-7: Don't reset TCP keepalive timer on reused connections.
- RCM-109: Fix issue preventing module from joining WPA mixed-mode (TKIP/CCMP) networks.

10.72 (December 2012)

This is a recommended release.

NEW FEATURES

- FCC-labeled versions of RCM6600W family boards are locked to the Americas region code.
- The following new options are available in the command line Rabbit Field Utility (`Utilities\clRFU.exe`):

  ```
  -i pathname Specify application binary image file, -i BinPath
  -si         Print formatted contents of System ID block
  -ma         Print formatted MAC address
  -bi         Print Board ID
  --boot-flashtype flash_type
              Specify flash_type as either parallel or serial
  --boot-memorywidth memory_width
              Specify memory_width (in bits) as either 8 or 16
  --boot-ram-selectenable mb0cr_value
              Specify mb0cr_value in the range [0,7]
              /CS0:3 in bits 1:0, /OE0:1 (== /WE0:1) in bit 2
  ```

  Previously, the command line RFU required that its first parameter must be the path name of the binary image to load to the Rabbit target board.

  Now, when the new `-i pathname` option is not specified, it is possible to check Rabbit board information via the new `-si` and/or `-ma` and/or `-bi` options without loading a binary image to the Rabbit target board.

- The Rabbit Field Utility, after successfully loading a binary image onto a Rabbit target board, now automatically starts program execution. `RabbitBios.c` has been updated to provide the following three behavior options after the RFU has started program execution:

  - Default behavior. When neither `RFU_BIN_RUN_IMMEDIATELY` nor `RFU_BIN_WAIT_FOR_RUN_MODE` are defined, a BIN program image successfully loaded by RFU 4.72 or later will run BIOS code only and then will wait in an idle loop until a hardware reset occurs. Following a hardware reset and with the programming cable disconnected, BIOS code and user code will execute as usual.

    This mode of operation most closely emulates the behavior of RFU versions prior to 4.72. It can also be useful in test fixture code, for example, where a test fixture can control at least one of the `SMODEx` levels as well as the `RESET` level, and manual disconnection of the programming cable is inconvenient or undesirable.

  - When `RFU_BIN_RUN_IMMEDIATELY` is defined in Dynamic C's Project Options' Defines tab, a BIN program image successfully loaded by RFU 4.72 or later will run both BIOS code and user code immediately after loading, with the programming cable still connected. With the programming cable disconnected, BIOS code and user code will execute as usual.

    This mode of operation is enforced when Dynamic C's debug kernel is enabled. It can also be useful in test fixture code, for example, where a test fixture can not or does not control the `SMODEx` levels and manual disconnection of the programming cable is inconvenient or undesirable.

  - When `RFU_BIN_WAIT_FOR_RUN_MODE` is defined in Dynamic C's Project Options' Defines tab, a BIN program image successfully loaded by RFU 4.72 or later will run BIOS code and then pause in an idle loop, waiting for the programming cable to be disconnected. As soon as the programming cable is disconnected, user code will begin to execute. With the programming cable disconnected, BIOS code and user code will execute as usual.

    This mode of operation can be useful in test fixture code, for example, where a test fixture can control at least one of the `SMODEx` levels and manual disconnection of the programming cable is inconvenient or undesirable.

ENHANCEMENTS

- The `Lib\Rabbit4000\NandFlash\nflash.lib` driver library has been updated with support for Micron/Numonyx/ST NANDxxxW3A nand flash devices. Previously supported nand flash devices, used on RCM4000 and RCM4050 boards, have been discontinued and are no longer available. As of 01-Dec-2012, new production RCM4000 and RCM4050 boards will have a Micron/Numonyx/ST NAND256W3A (32MB) nand flash device installed. Custom applications which use nand flash, whether based on RCM4000, RCM4050 boards or a custom nand flash board design, should be recompiled using Dynamic C 10.72 in order to

support the new, increased selection of small-block nand flash devices.
- The EXCEPTION(x) macro is no longer used in any standard Dynamic C code and is now deprecated. Please refer to the associated deprecation comments in ERRORS.LIB for more information.
- iDigi access via a PPP serial port can now be hosted on an alternate parallel port by defining one of `IDIGI_PPP_USE_PORTD` or `IDIGI_PPP_USE_PORTE` before `#use idigi.lib`.

BUG FIXES

- Defect #41708. Prevented possible program hang and possible false-positive file transfer success result in `ftp_client.lib`.
- Defect #41912. When `HTTP_CUSTOM_HEADERS` is defined and used, an incorrect buffer offset calculation results in corrupted HTTP header buffer content.
- Defect #42073. The `digOutConfig_H()` function in `BLxS2xx.LIB` now returns the expected 0 result on success.
- Defect #43173. Dynamic C's run time error reporting now consistently reports the return address on the stack when `exception()` is called.

Release Notes Part Number: 93000751