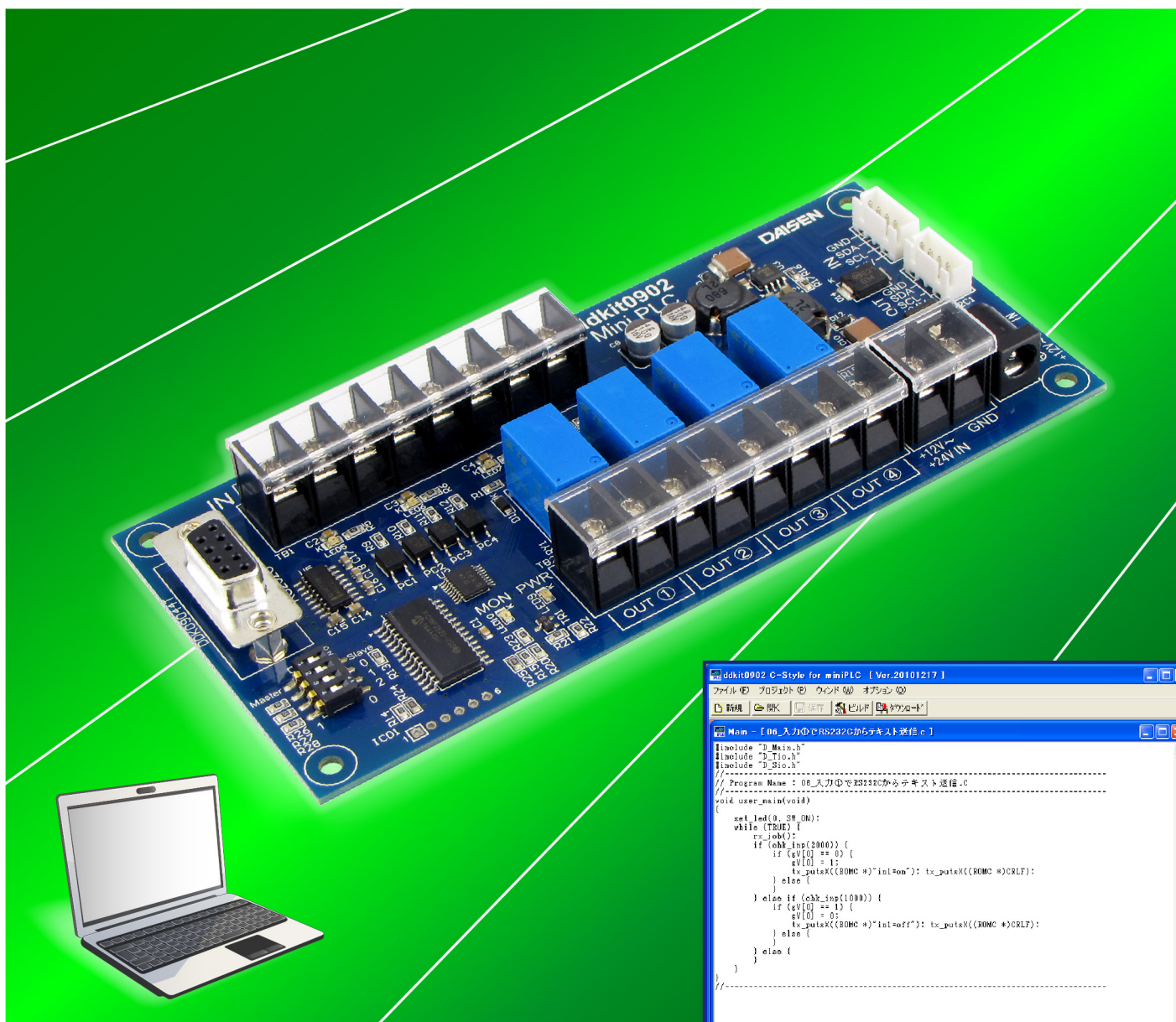


# Mini PLC

## C-Style C-Code編



```
ddkit0902 C-Style for miniPLC [ Ver.20101217 ]
ファイル  フォント  ウィンドウ  オプション
[ 新規 ] [ 開く ] [ 保存 ] [ 印刷 ] [ 実行 ] [ ヘルプ ] [ 終了 ]
Main - [ 06_入力でのRS232Cからテキスト送信.c ]
#include "D_Main.h"
#include "D_Tio.h"
#include "D_Sio.h"
//-----
// Program Name : 06_入力でのRS232Cからテキスト送信.c
//-----
void user_main(void)
{
    set_led(0, SW_ON);
    while (TRUE) {
        rx_job();
        if (chk_inp(2000)) {
            if (gV[0] == 0) {
                eV[0] = 1;
                tx_putX((ROMC *)"in-on"); tx_putX((ROMC *)ORLF);
            } else {
            } else if (chk_inp(1000)) {
                if (gV[0] == 1) {
                    eV[0] = 0;
                    tx_putX((ROMC *)"in-off"); tx_putX((ROMC *)ORLF);
                } else {
                } else {
                } else {
            }
        }
    }
}
//-----
保存  C:\Daisen\C-Style for miniPLC\Dat\Sample\06_入力でのRS232Cからテキスト送信.c
Main
```

## 目 次

### C-Code 操作ガイド (本書)

1 . C-Code ボタンの説明 -----	2
1-1. 時間待ちを C - C o d e ボタンで記述する例 -----	3
1-2. 変数の範囲条件を C - C o d e で記述する例 -----	4
1-3. C - C o d e ボタン使用時の注意 -----	5
2 . C-Code 編集の説明 -----	6
2-1. 編集スタイルを C - C o d e に変える -----	6
2-2. C - S t y l e でビルドした C - C o d e ファイルを開く ---	7
2-3. m i n i P L C ファームウェア仕様 -----	9

## 1 . C-Code ボタンの説明

C-Code ボタンは、C-Style プログラム中に簡単なC言語を直接記述できます。

ビルド画面の時にC-Style プログラムボタンからC言語に変換表示さるコードのことです。C-Style に慣れてくると、もう少し高度な記述をしてみたいと思ったことはありませんか？ そんな時にこの「C-Code」ボタンを使って、直接C言語を記述すれば実現できます。



オプションメニューの「C-Code ボタンの表示」を選択すると、画面右側のプログラムボタンリストに **Ccode** ボタンが表示されます。

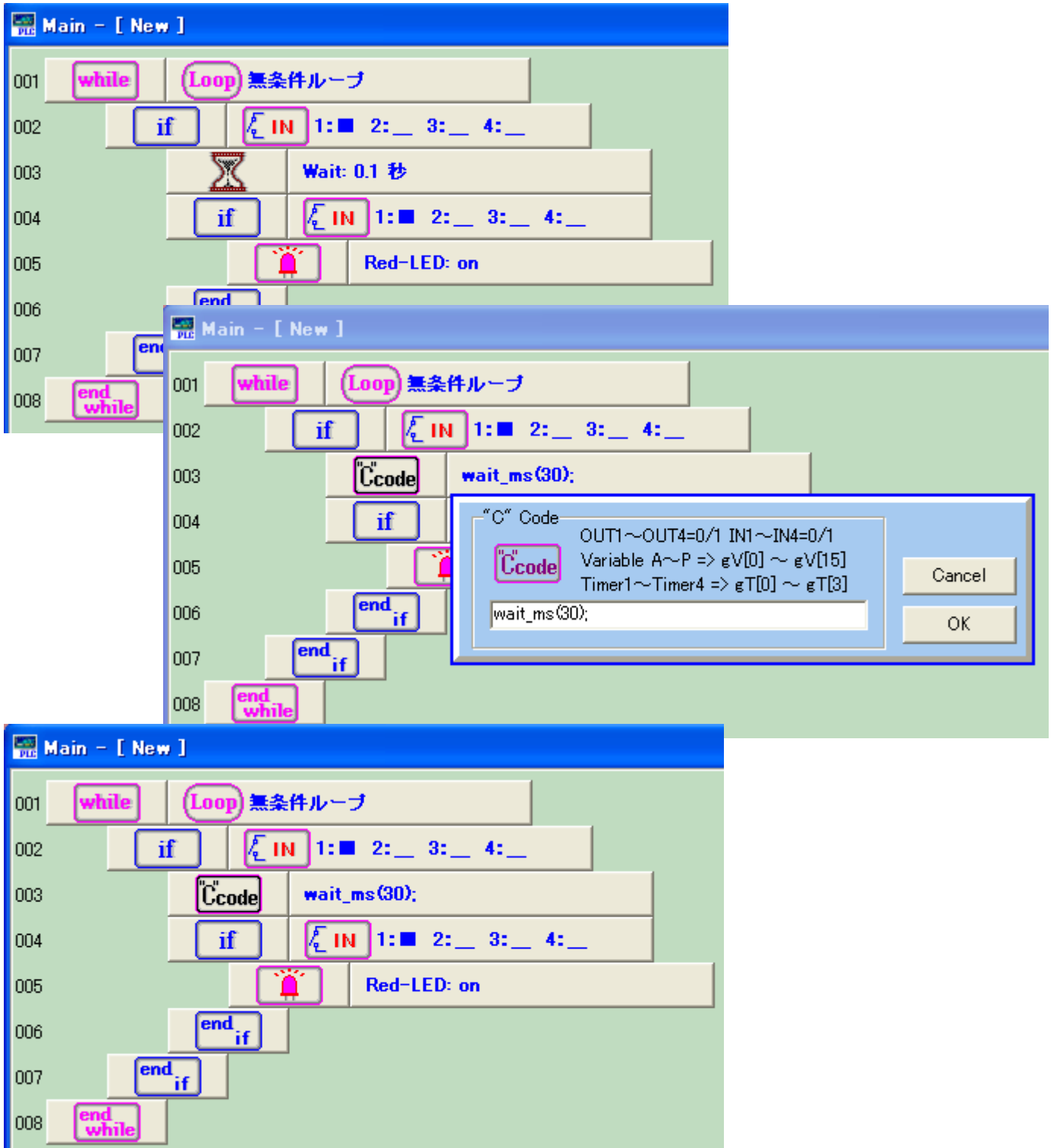


条件分岐ボタンや条件付き繰り返しボタンにも「C-Code」で条件を直接記述することが出来ます。




### 1-1. 時間待ちを C-Code ボタンで記述する例

 ボタンでは 0.1 秒以下の設定は不可能でしたが、 ボタンを使うと 1 ミリ秒単位での記述可能となります。



The image shows three sequential screenshots of a ladder logic editor window titled "Main - [ New ]".

- Top Screenshot:** Shows a ladder logic program with a "while" loop starting at line 001. Inside the loop, there is an "if" statement at line 002. At line 003, there is a "Wait: 0.1 秒" block. At line 004, there is another "if" statement. At line 005, there is a "Red-LED: on" block. The program ends with "end\_while" at line 008.
- Middle Screenshot:** A dialog box titled "C Code" is open over the "Wait: 0.1 秒" block. The dialog contains the following text:  
"C" Code  
 `wait_ms(30);`  
OUT1~OUT4=0/1 IN1~IN4=0/1  
Variable A~P => gV[0] ~ gV[15]  
Timer1~Timer4 => gT[0] ~ gT[3]  
The input field contains `wait_ms(30);`. There are "Cancel" and "OK" buttons.
- Bottom Screenshot:** The "Wait: 0.1 秒" block has been replaced by a "Ccode" block containing `wait_ms(30);`. The rest of the ladder logic program remains the same.

C-Style で使用する時間待ちの C 言語関数は、`wait_ms(unsigned int)` で、1 ~ 65535 ミリ秒の範囲で指定できます。

## 1-2. 変数の範囲条件を C-Code で記述する例

C-Style で変数の範囲を判定する場合 2 行の記述が必要でしたが、C-Code 条件判定すると 1 行で記述することが出来ます。

The image consists of three screenshots from a PLC ladder logic editor, illustrating the process of converting a C-style range condition to C-code.

**Top Screenshot:** Shows a ladder logic network with a 'while' loop (無条件ループ) starting at step 001. Inside the loop, an 'if' condition (A > 0) is used at step 002. Within this 'if' block, another 'if' condition (A < 10) is used at step 003. This second 'if' block leads to an 'OUT' instruction at step 004, which is connected to four outputs: 1: (with a blue square symbol), 2: \_\_, 3: \_\_, and 4: \_\_. The 'if' blocks are closed with 'end\_if' instructions at steps 005 and 006, and the 'while' loop is closed with 'end\_while' at step 007.

**Middle Screenshot:** Shows the same ladder logic network, but the 'if' condition at step 002 has been updated to '0 < eV[0] & eV[0] < 10'. A dialog box titled '"C" Code' is open over the 'if' block. The dialog contains the following text:  
"C" Code  
OUT1~OUT4=0/1 IN1~IN4=0/1  
Variable A~P => eV[0] ~ eV[15]  
Timer1~Timer4 => eT[0] ~ eT[3]  
The input field contains the C-code: `0 < eV[0] && eV[0] < 10`. The dialog has 'Cancel' and 'OK' buttons.

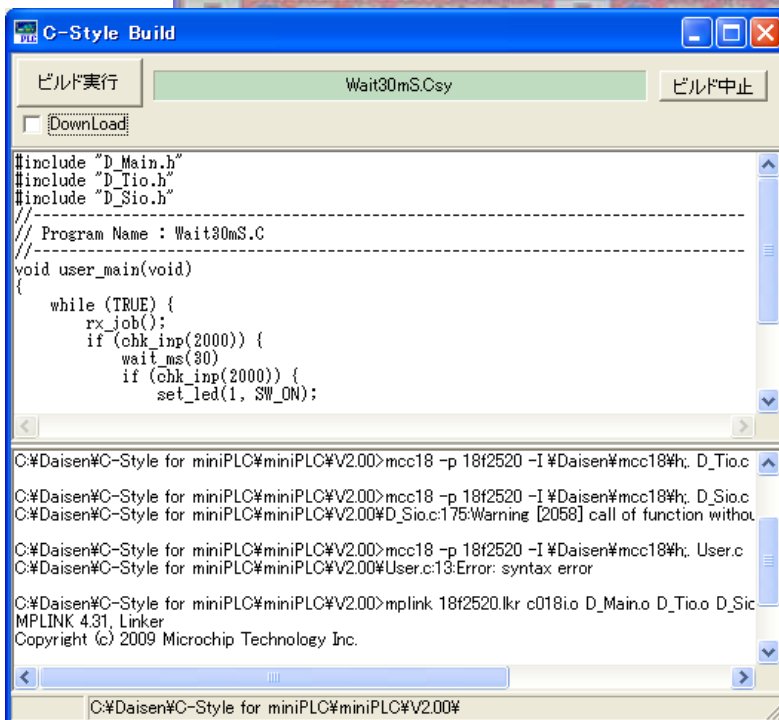
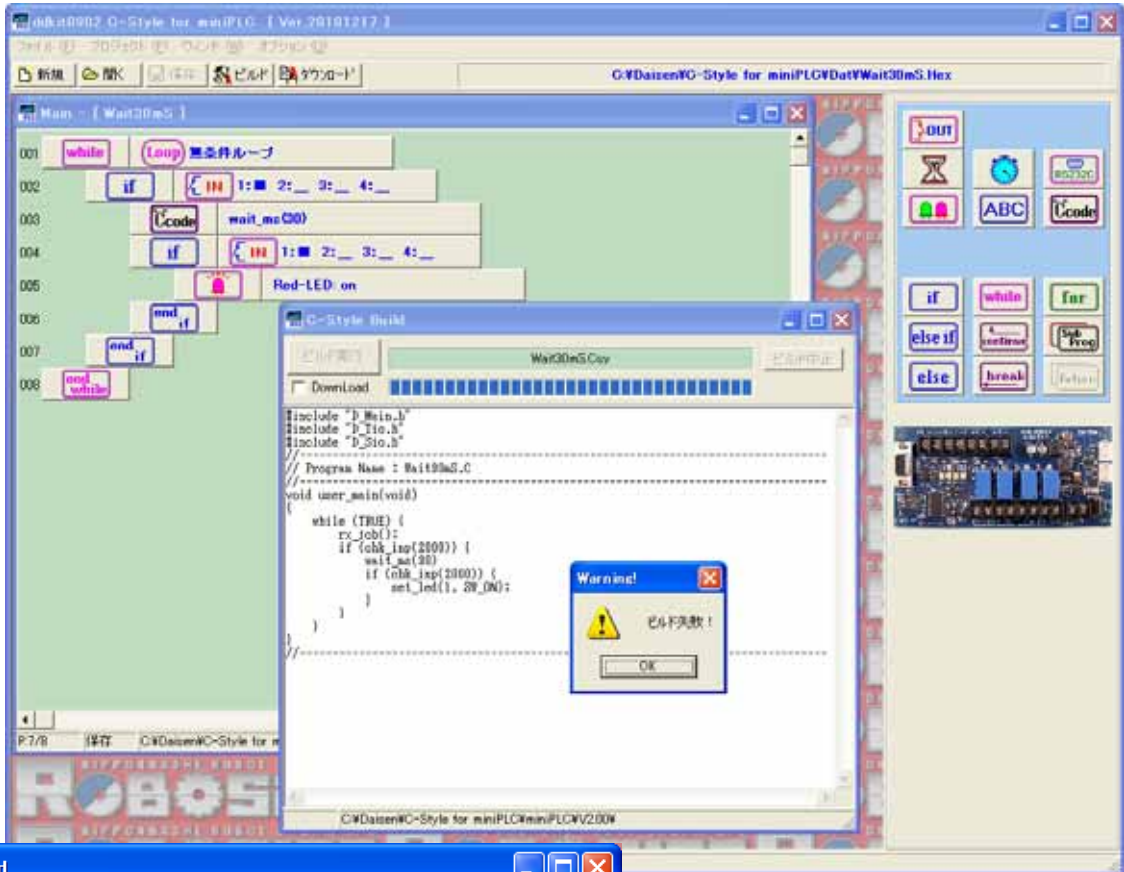
**Bottom Screenshot:** Shows the final ladder logic network after the conversion. The 'if' condition at step 002 is now '0 < eV[0] & eV[0] < 10'. The 'end\_if' instruction at step 004 and the 'end\_while' instruction at step 005 remain the same. The 'OUT' instruction at step 003 is also present.

条件判断での C-Code を編集する場合は、行末のセミコロン「 ; 」は必要ありません。

### 1-3.C-Code ボタン使用時の注意

通常の C-Style ボタンだけで作成されたプログラムは、ビルド成功が当たり前でしたが、C-Code ボタンで直接 C 言語を記述するとタイプミスしたり、C 言語のルール違反でエラーが発生し、ビルド失敗も起こります。

画面の例では、"wait\_ms(30)"の最後に ; (セミコロン) が抜けているだけでエラーが起こります。



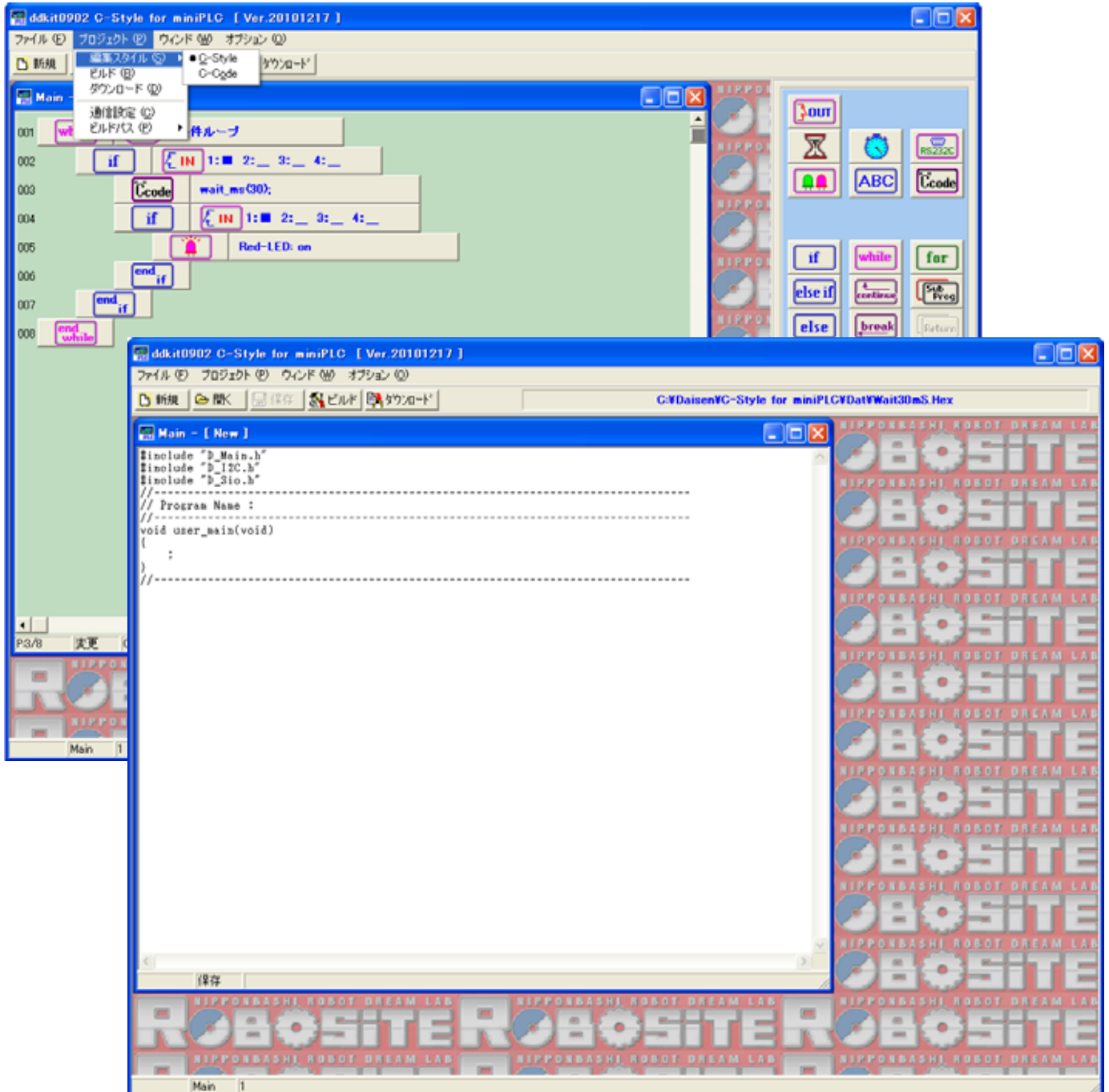
ビルド失敗のダイアログが表示され「OK」ボタンをクリックするとビルド画面は閉じないで、エラー表示をします。

この場合は、「ビルド中止」ボタンをクリックして一旦ビルド画面を閉じてから、問題の箇所を修正します。再度ビルド実行し、成功するまで繰り返します。

## 2 . C-Code 編集の説明

### 2-1. 編集スタイルを C-Code に変える

C-Code ボタンが表示されている場合にプロジェクトメニューに編集スタイルの変更メニューが追加され、C-Style 編集または C-Code 編集を選択することが出来ます。



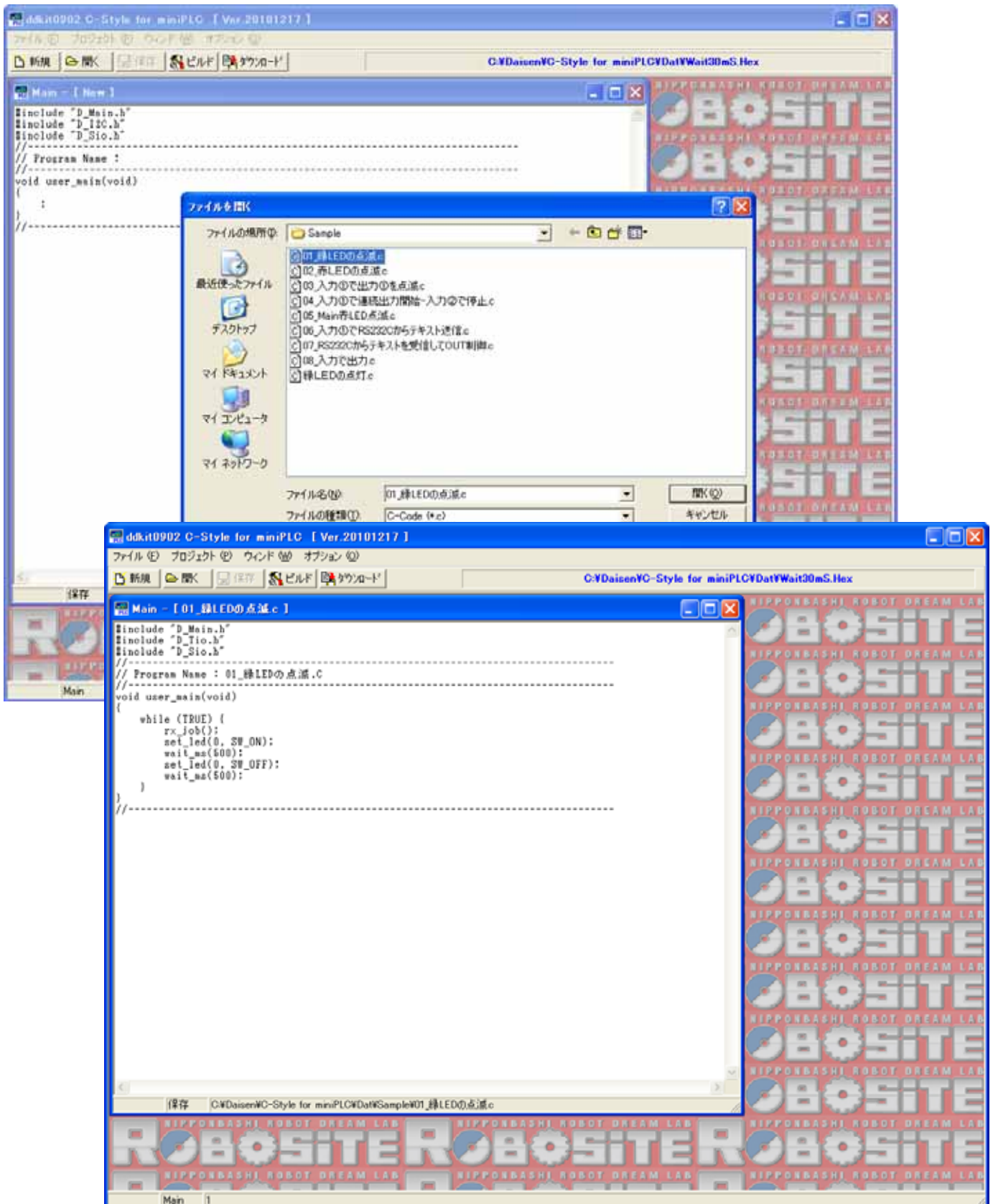
C-Code 編集メニューを選択すると C-Style ボタンの表示が無くなり、全て C 言語の編集となります。この画面で直接 C 言語のコードを記述するか、または、別のテキスト編集ソフトで編集した C 言語ソースファイルを開くことも出来ます。

ビルド及びダウンロードは C-Style 同様に行うことが出来ます。

## 2-2.C-Style でビルドした C-Code ファイルを開く

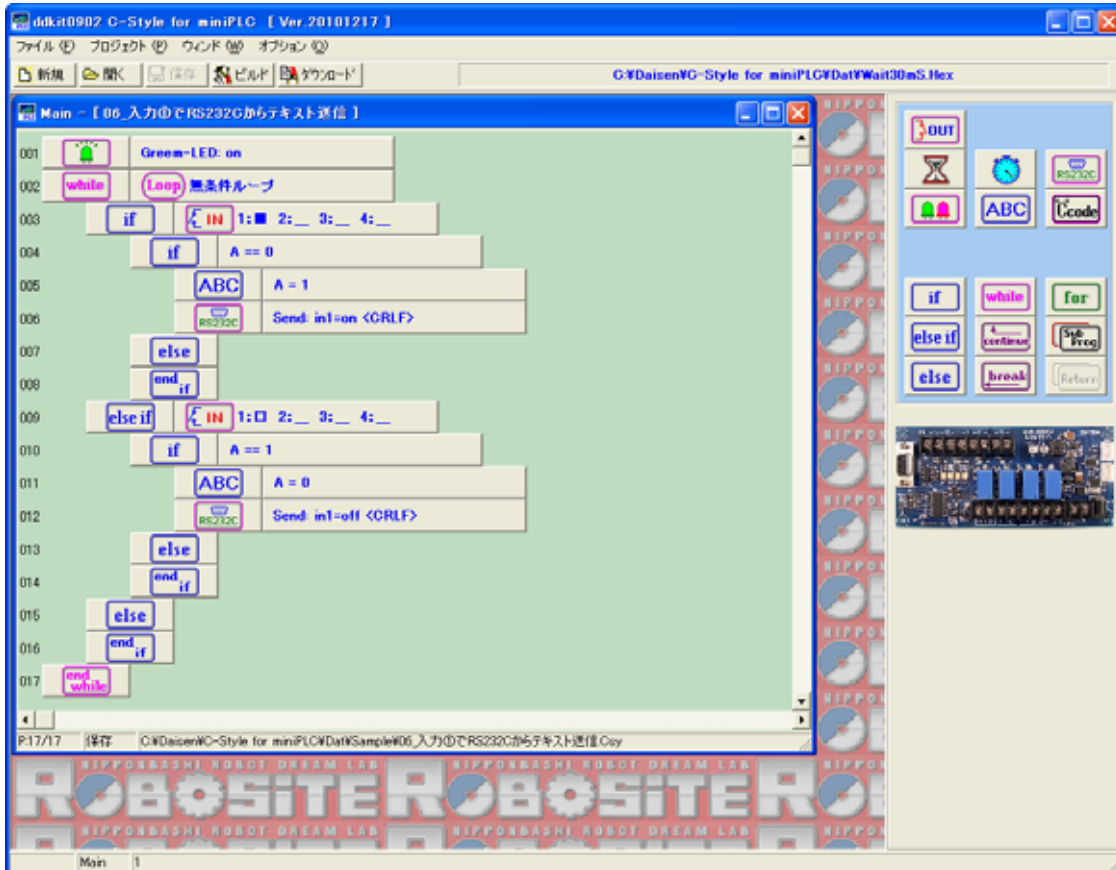
C-Style でビルドしたファイルは常にC言語ソースファイルとして残されていますので、一からC言語を記述することなく、C - C o d e 編集が行えます。

緑 LED の点滅プログラムのC言語ソースファイルを開く

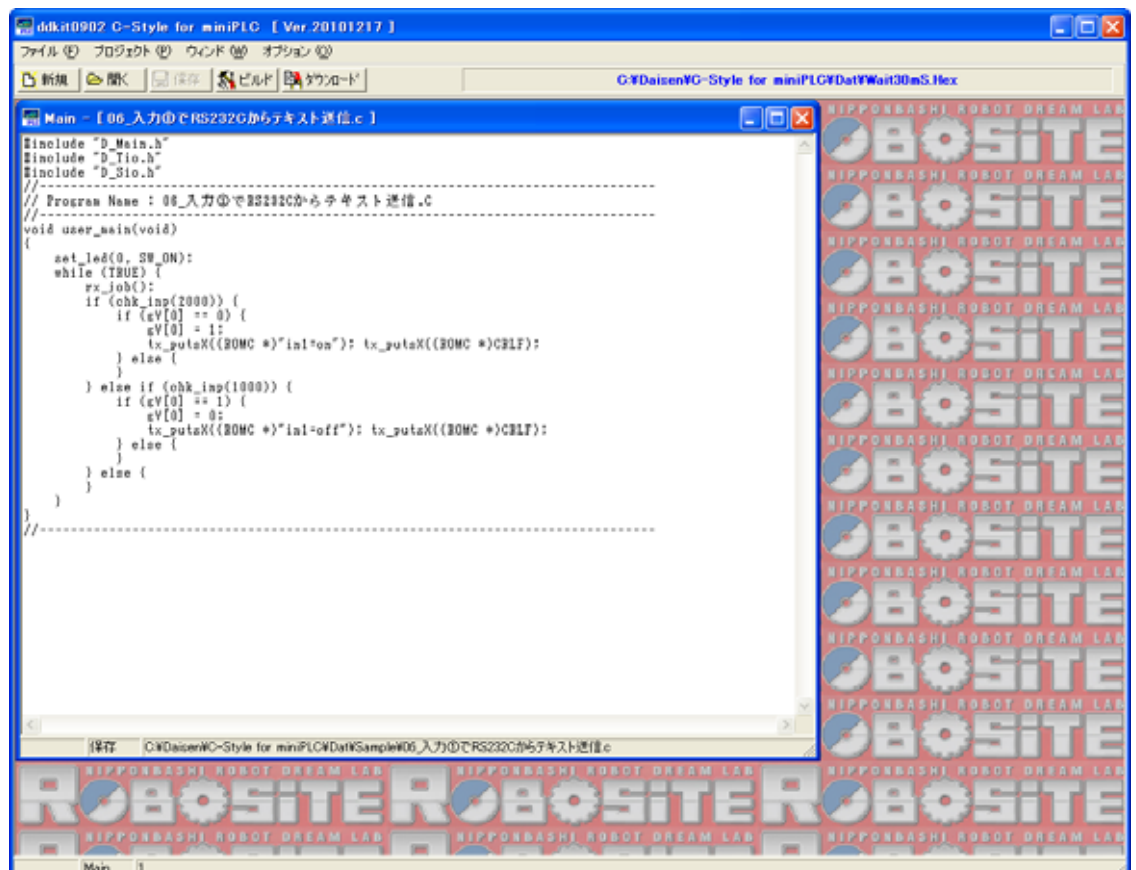




## 付属の C-Style プログラムをビルドする



## C-Code 編集に切替えてから C 言語ソースコードを開く



## 2-3.miniPLC ファームウェア仕様 (V2.00:2010.12.17)

C-Code 編集する上で必要な miniPLC のファームウェアで使用する関数と変数仕様

//状態判定関数

```
BOOL get_led(BYTE led_no); //LED 点灯状態を得る (led_no:GREEN(0)/RED(1)) ret:ON/OFF
BOOL get_inp(BYTE pin); //入力状態を得る (pin:0~3:入力1~4) ret:ON/OFF
BOOL chk_inp(UINT dat); //入力判定(dat:チェックデータ) ret:TRUE/FALSE
BOOL chk_out(UINT dat); //出力判定(dat:チェックデータ) ret:TRUE/FALSE
```

//出力制御関数

```
void set_led(BYTE led_no, BOOL sw); // LED 点灯制御 led_no:0(GREEN)/1(RED), sw:ON/OFF
void set_out(UINT dat); // 出力制御(dat:nnnn => OUT1,2,3,4) nnnn n:0(無し),1(OFF),2(ON)
```

//シリアルデータ状態返送関数 (I/O モニターで使用)

```
void chk_var(void) // 変数とタイマー値をシリアルデータで返送
void chk_io(void) // 入出力状態をシリアルデータで返送
```

//外部変数

```
ULNG gT[4]; // タイマー値
long gV[16]; // ユーザー変数 : A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P;
```

//シリアルコマンド

```
"Reset" // リセットコマンド
"M:0/1" // I/O モニターコマンド 0:禁止,1:許可
"lb:?" // 入力ポート状態問合せ (返り値 xx:00~FF)
"Ob:?" // 出力ポート状態問合せ (返り値 xx:00~FF)
"On:c" // 出力制御設定 (設定値 n:1~4(OUT1~4), c:0/1)

"Lg:?" // 緑 LED 点灯状態問合せ (返り値 0:消灯, 1:点灯)
"Lg:0/1" // 緑 LED 点灯設定 (設定値 0:消灯, 1:点灯)
"Lr:?" // 赤 LED 点灯状態問合せ (返り値 0:消灯, 1:点灯)
"Lr:0/1" // 赤 LED 点灯設定 (設定値 0:消灯, 1:点灯)

"T1:?" ~ "T4:?" // タイマー変数問合せ (返り値 n:0~long 最大値)
"Va:?" ~ "Vp:?" // 変数(A~P)問合せ (返り値 n:0~long 最大値)
"Ver" // ファームウェアバージョン問合せ "miniPLC V2.00"
```

メモ 1

**▲注意**

本製品は一般の民生・産業用として使用されることを前提に設計されています。人命や危害に直接的、間接的にかかわるシステムや医療機器など、高い安全性が必要とされる用途にはお使いにならないでください。

本製品の故障・誤動作・不具合によりシステムに発生した付随的障害および、本製品を用いたことによって生じた損害に対し、当社は一切責任を負いません。あらかじめご了承ください。

株式会社ダイセン電子工業  
**DAISEN**

〒556-0005 大阪市浪速区日本橋 4-9-24  
TEL: 06-6631-5553 / FAX: 06-6631-6886  
URL: <http://www.daisendenshi.com>  
e-mail: [ddk@daisendenshi.com](mailto:ddk@daisendenshi.com)