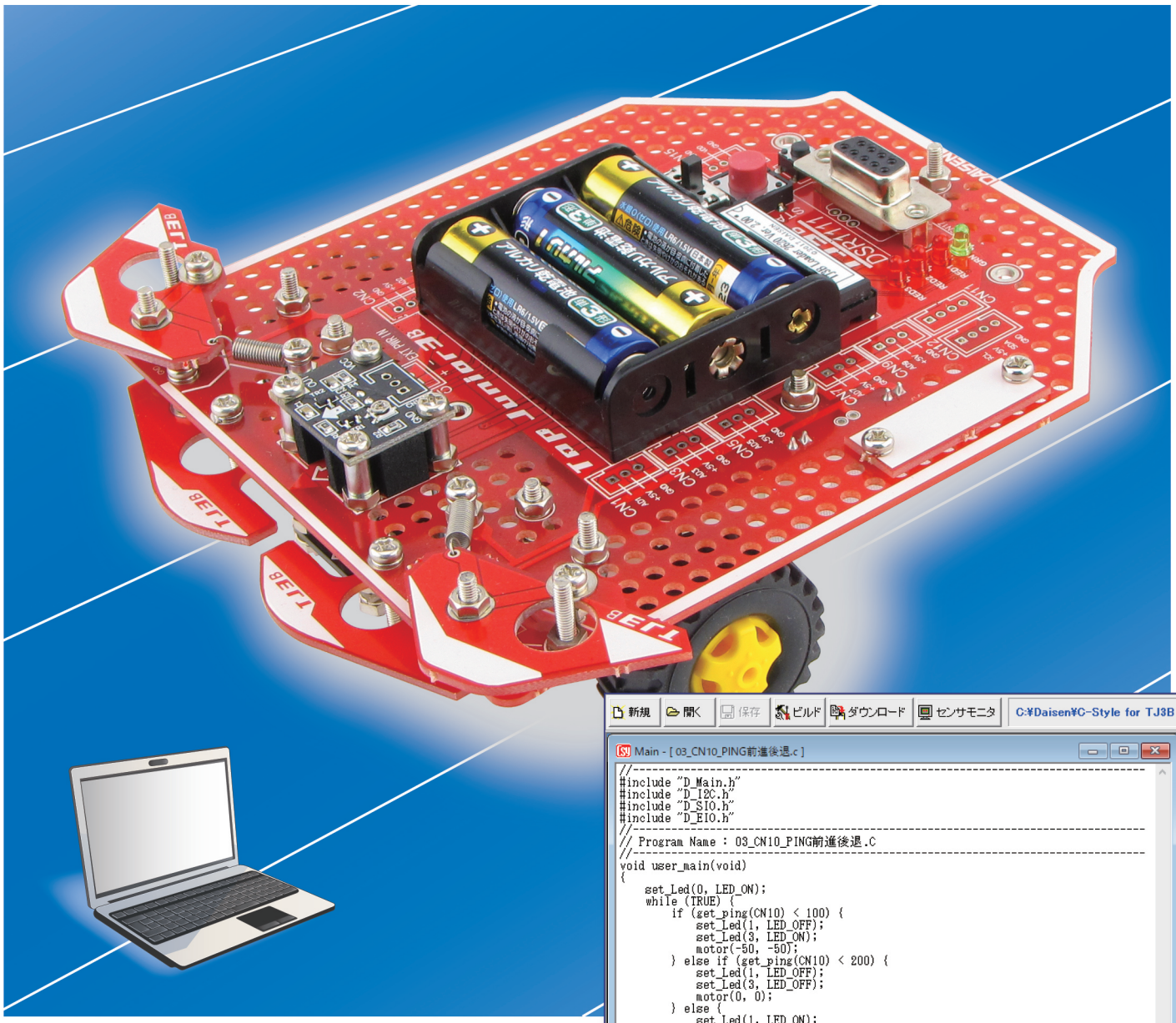
自立型ロボット製作キット
Autonomous Robot D.I.Y. Kit

Top Junior 3B

ティ・ジェイ・スリー ビー
TJ3B

C-Style C-Code編



Rev.190619

DAISEN
株式会社ダイセン電子工業

目　次

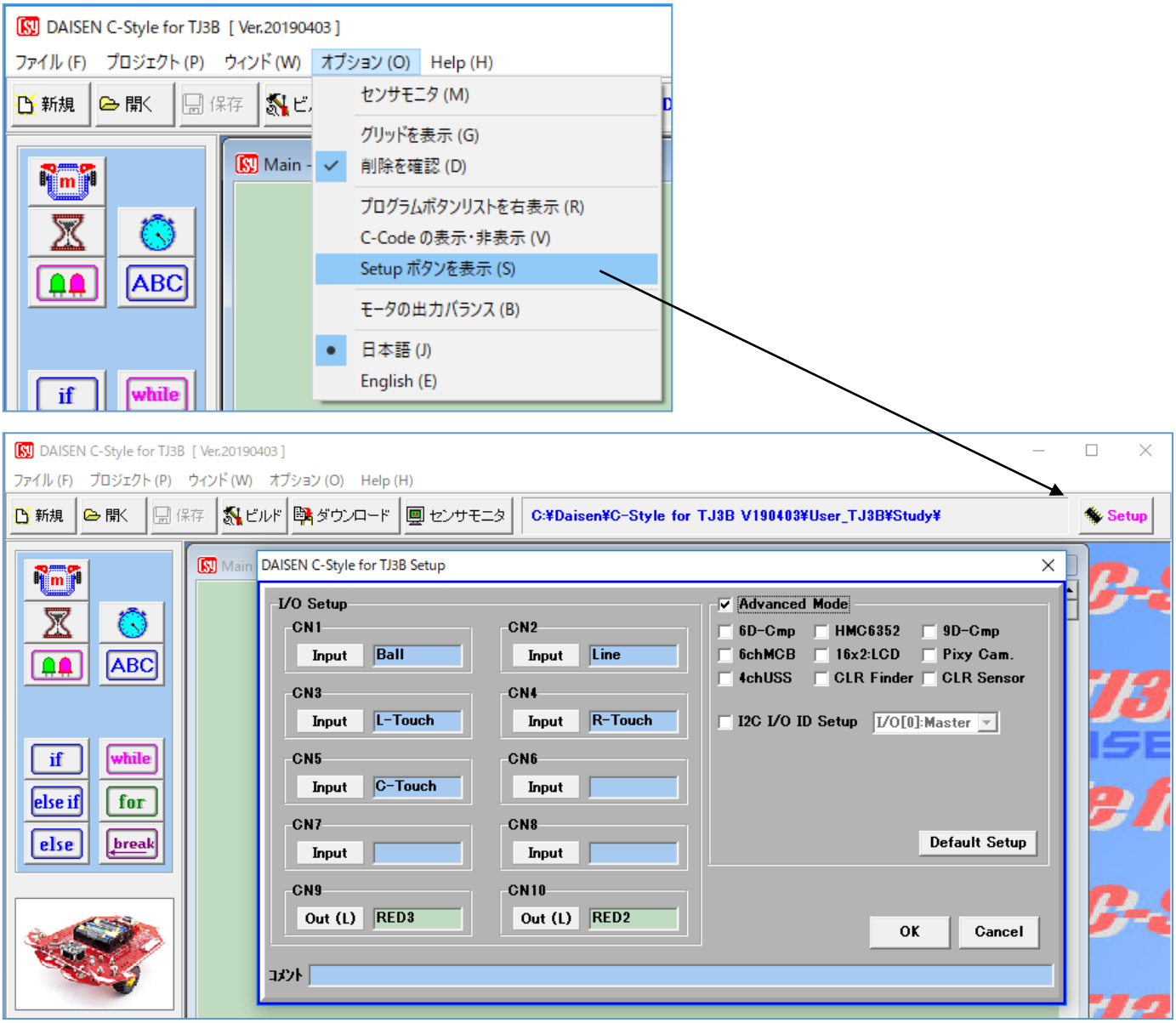
C-Code 編（本書）

１．C-Style 編集モードで C-Code ボタンを使う

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。

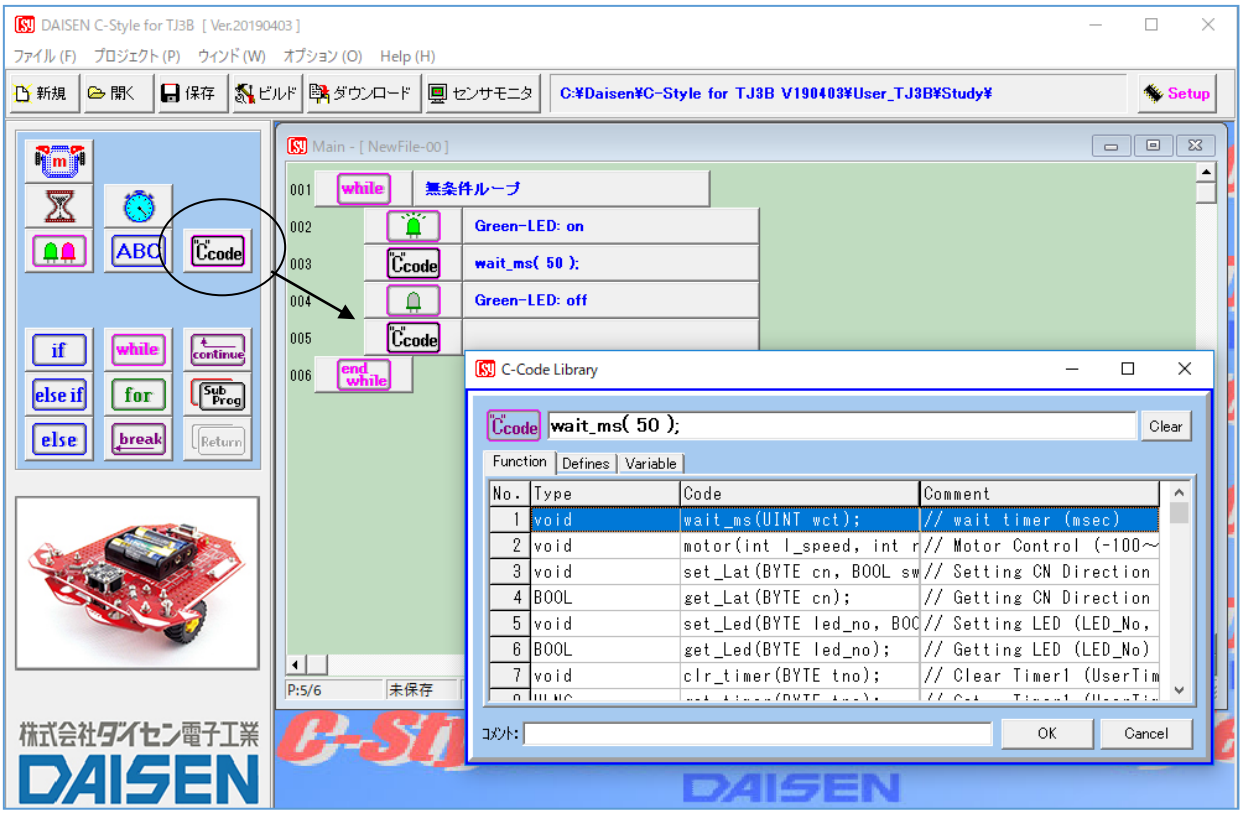
オプションメニューの「Setup ボタンの表示」を選択すると、画面右側に 

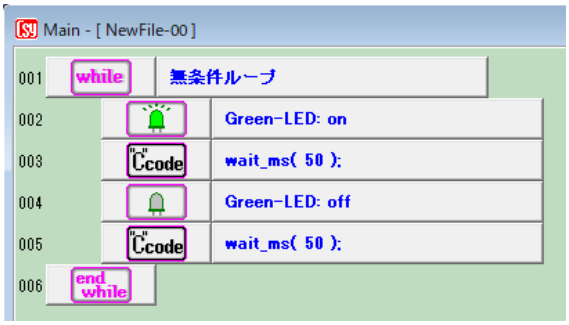
ボタンが表示されます。



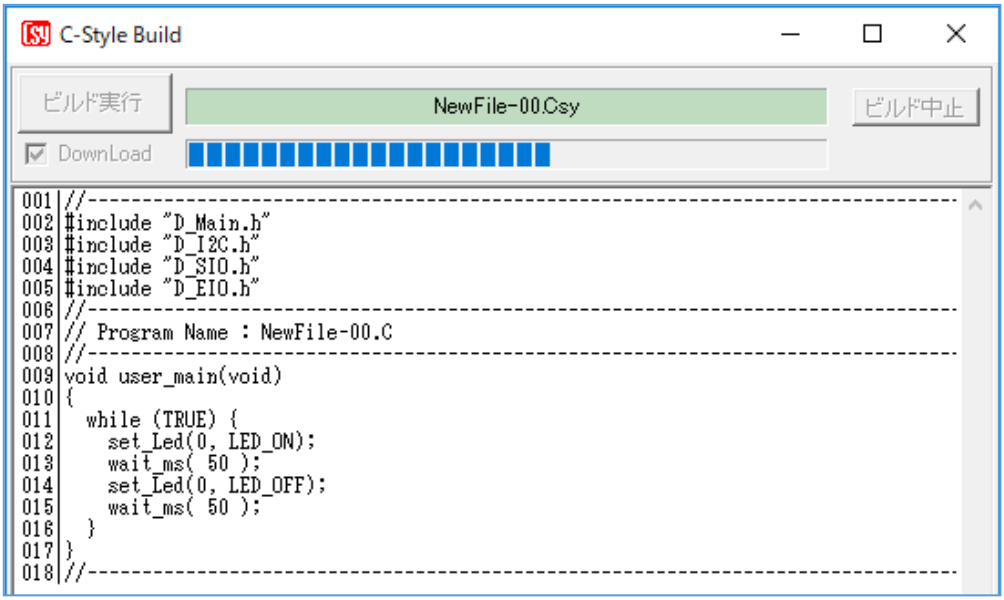
Setup ボタンを表示すると I/O Setup（入出力設定）ダイアログが表示されます。
“□Advanced Mode”にチェックを付けて「OK」ボタンでダイアログを閉じると、拡張されたプログラムボタンリストが表示されます。

例えば、C-Style ボタンでは、時間待ちの最小時間は 0.1 秒でしたが、「C-Code」ボタンを使って、直接Ｃ言語コードを記述することで、１ミリ秒単位のプログラムが実現できます。



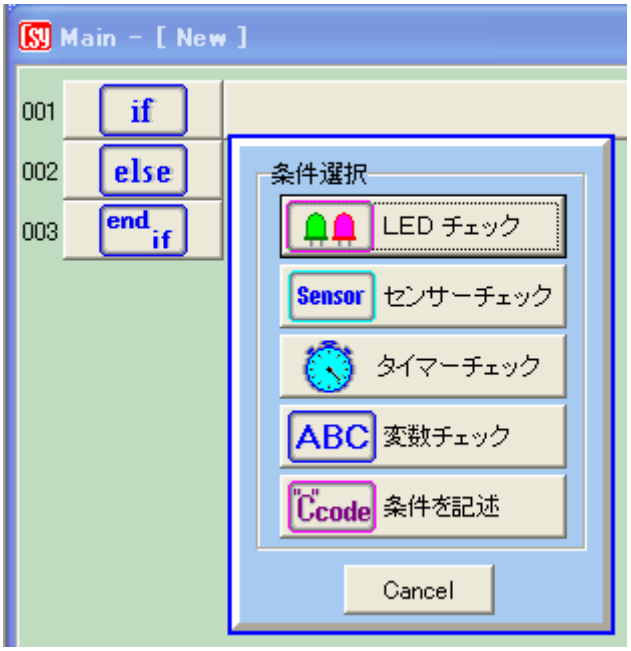


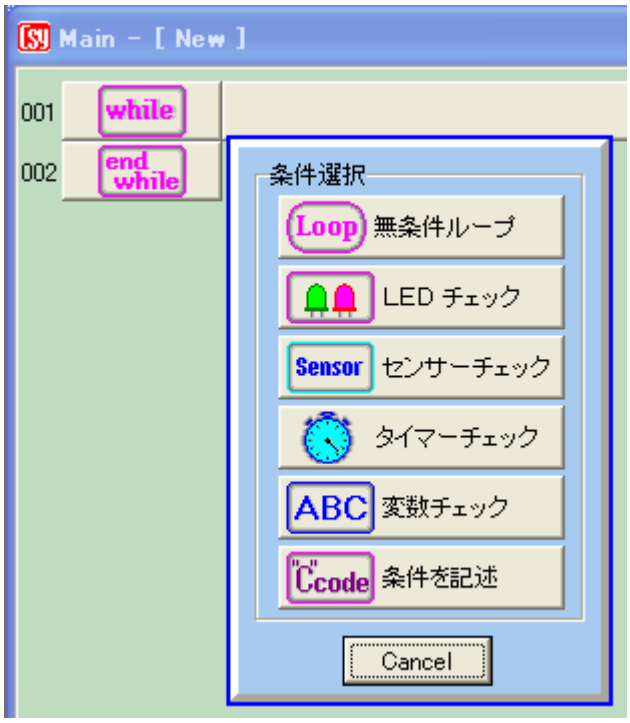
緑色 LED の 50 ミリ秒の高速点滅が出来ます。



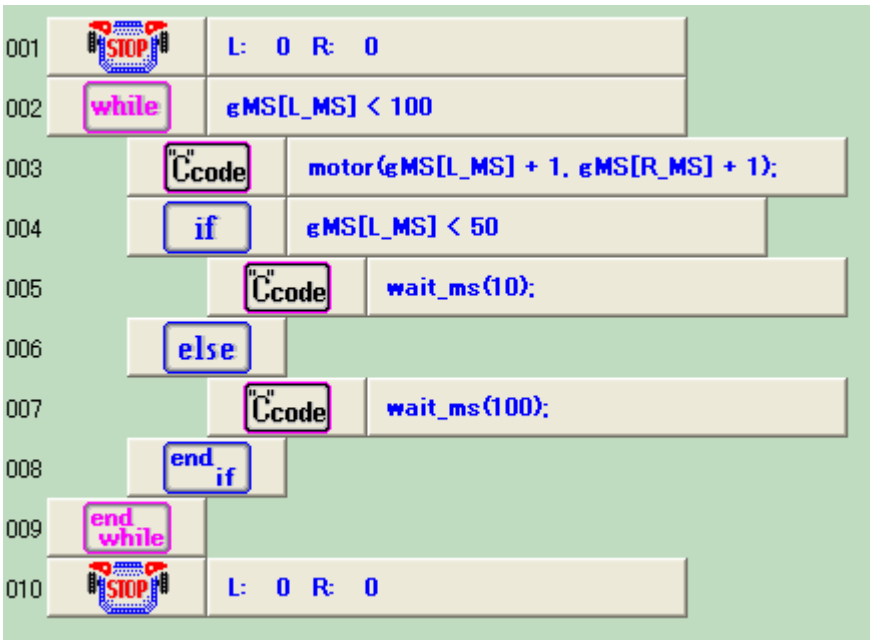
ビルドすると C-Code ボタンで記述されたままのコードが出力されていることがわかりますね！

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





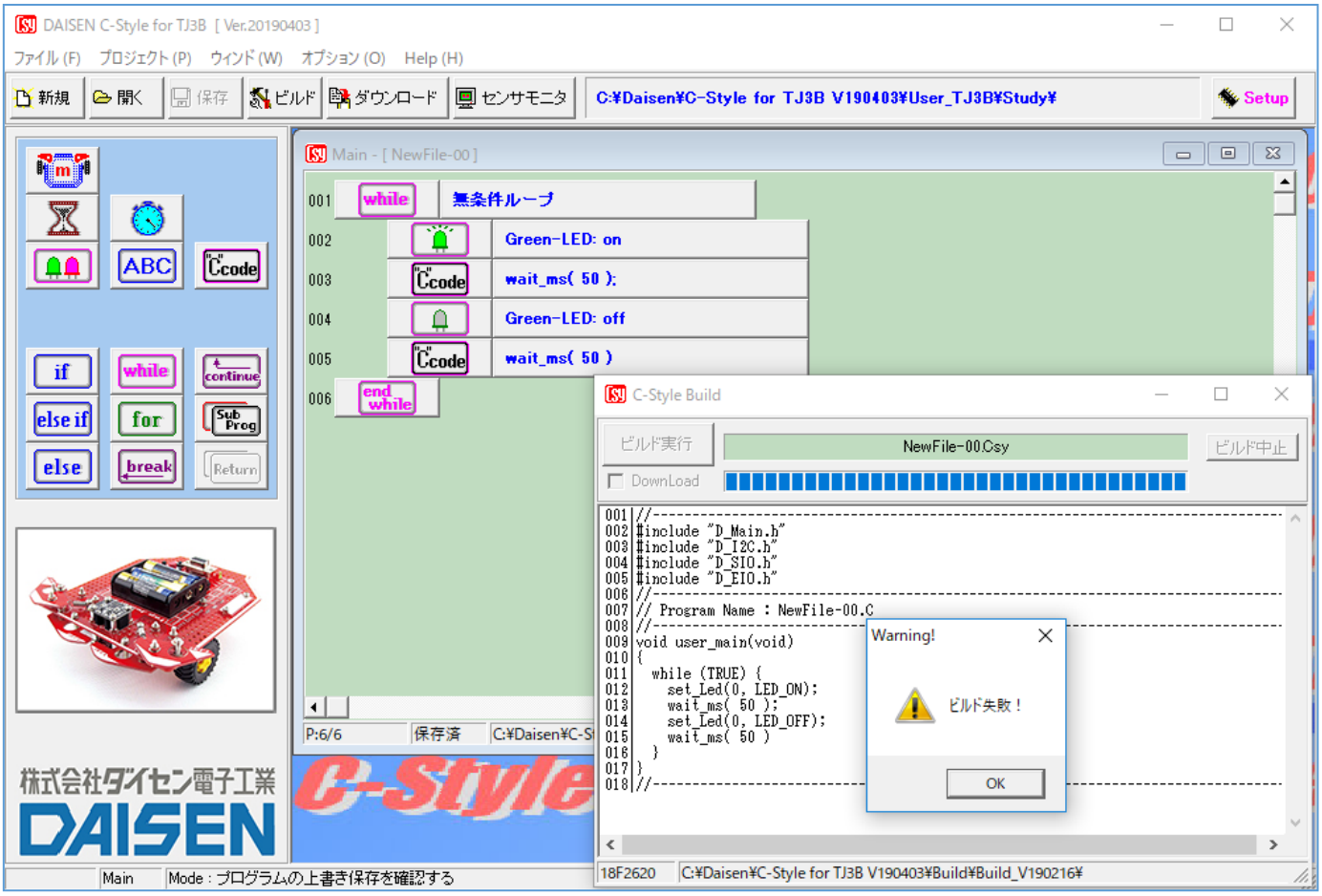
while, if 文で C-Code を使った例



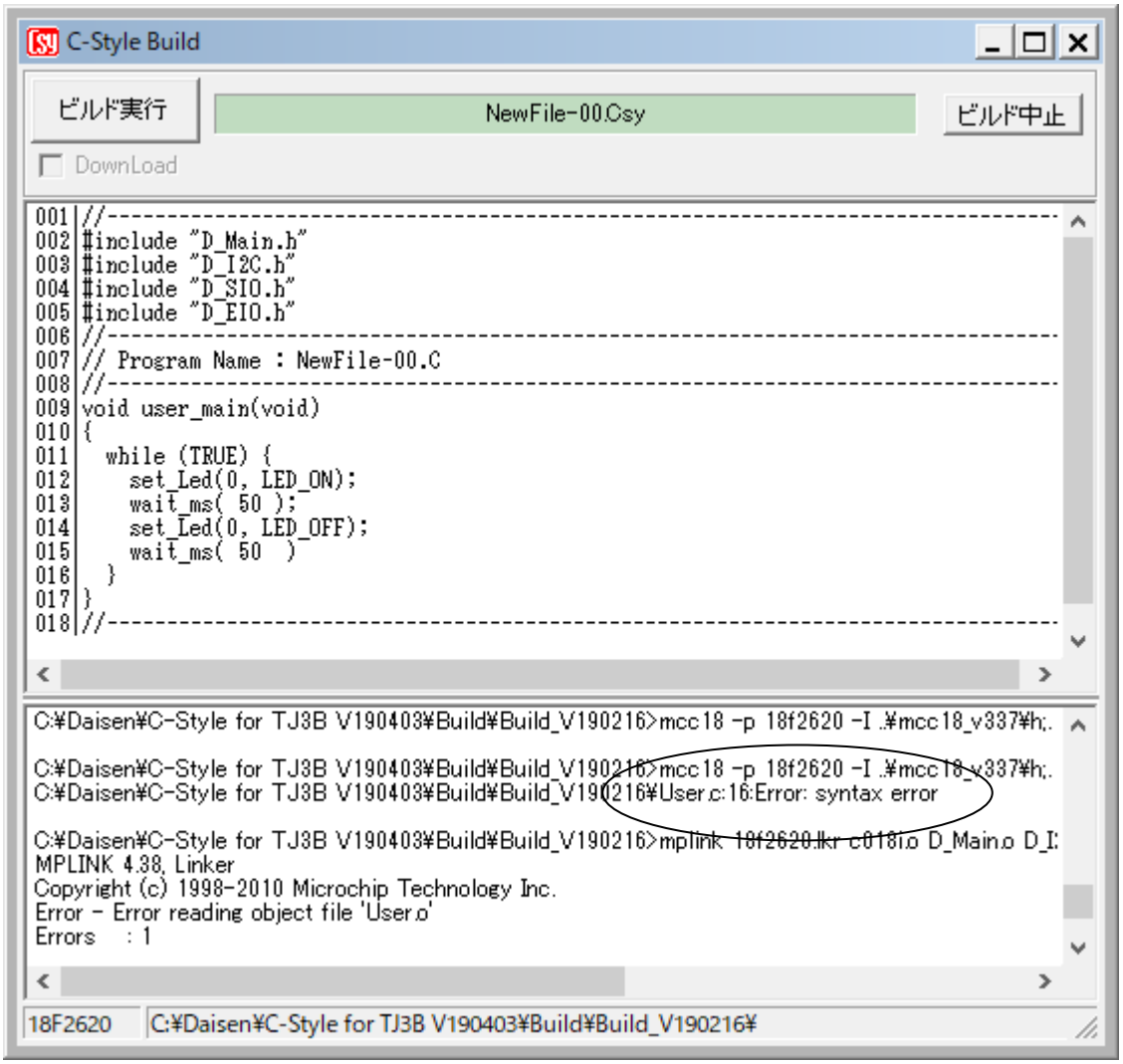
モータ制御関数が保持する現在の速度 gMS[L_MS], gMS[R_MS]に+1して 100 になるまで加速するプログラム例です。

■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や、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(50)”の最後に‘;’（セミコロン）が抜けているだけでエラー発生です。



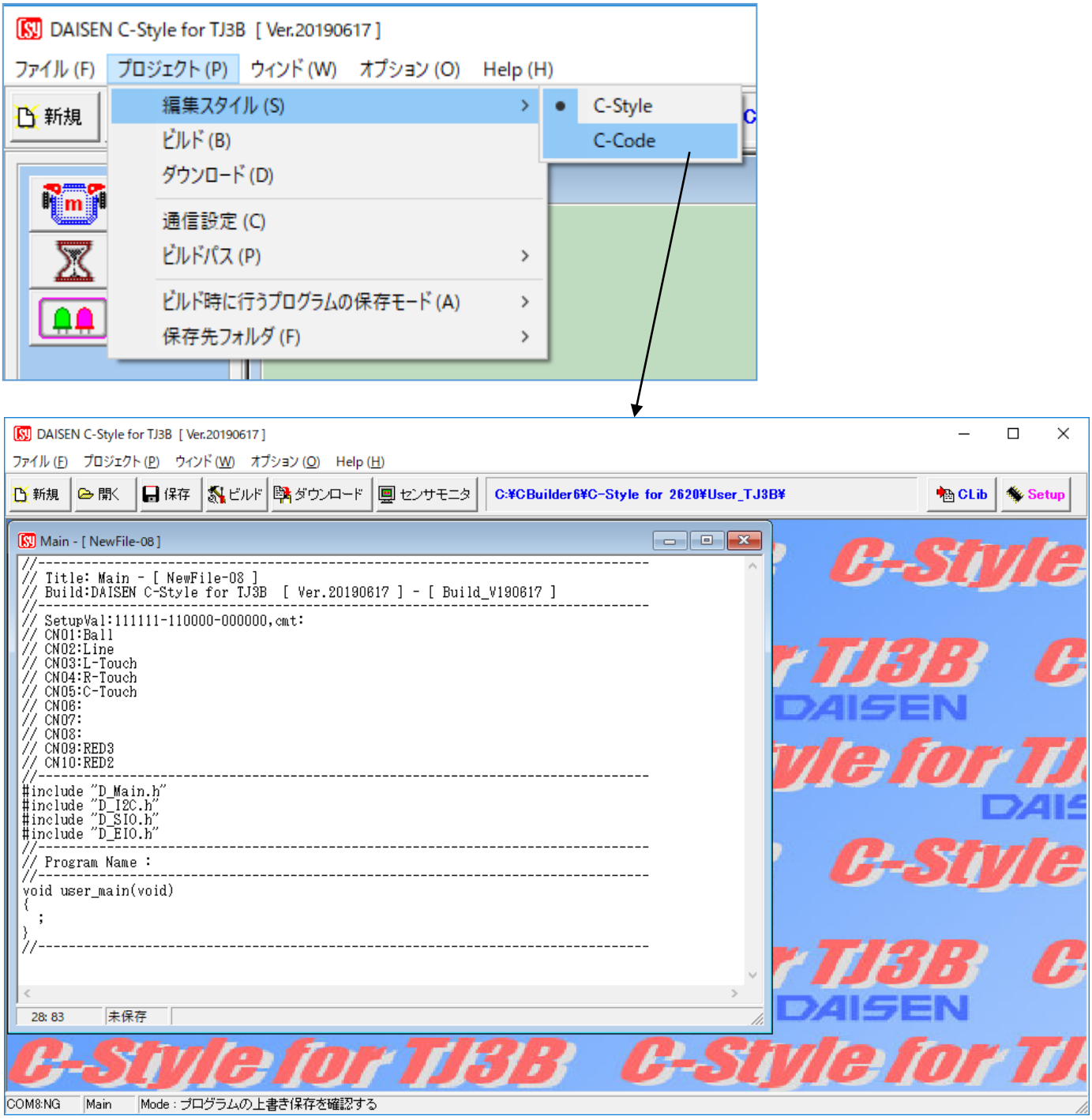
ビルド失敗のダイアログが表示され「OK」ボタンをクリックするとビルド画面は閉じないで、エラー表示をします。



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、成功するまで繰返します。

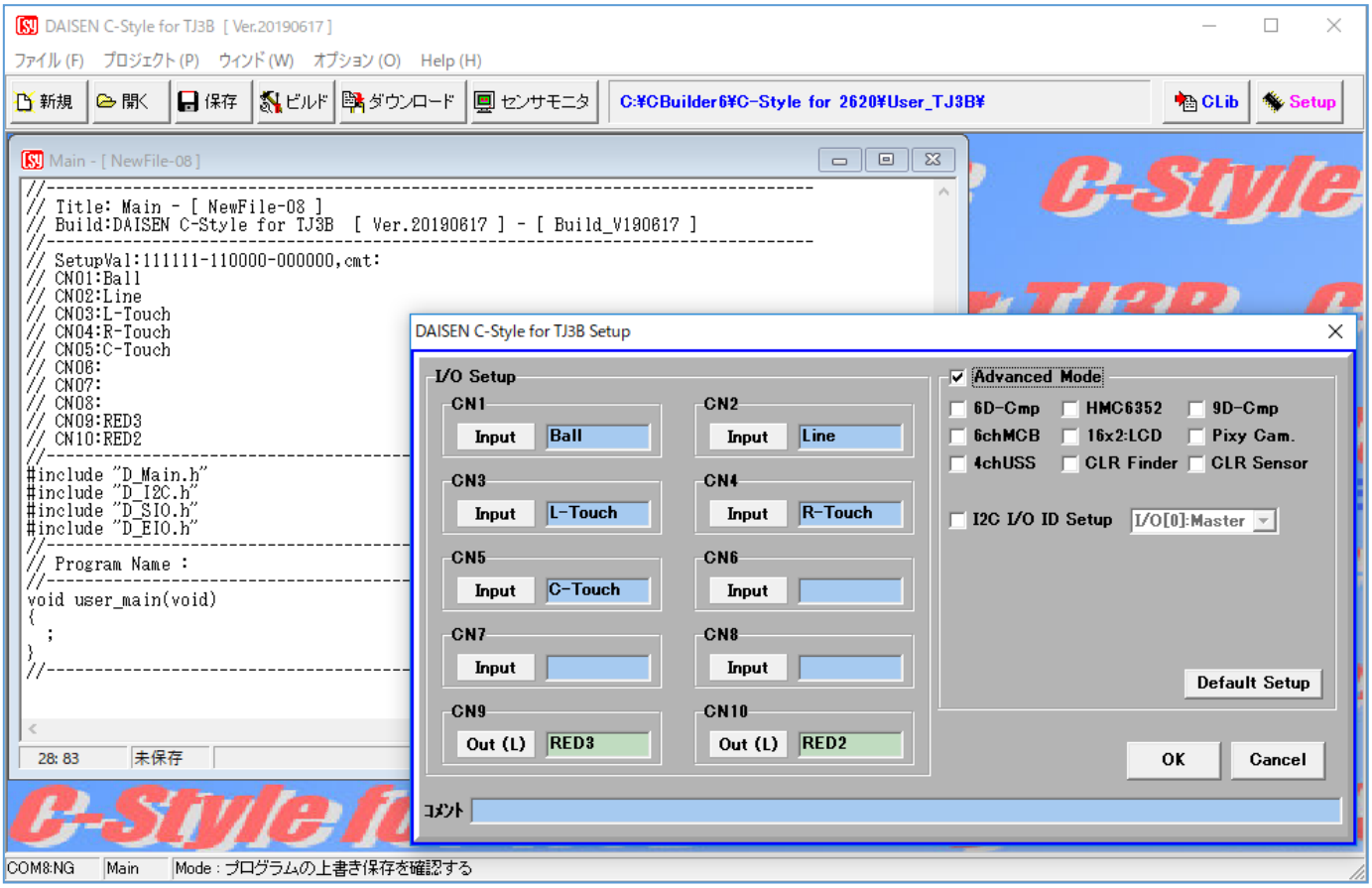
２．C-Code 編集モード

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。



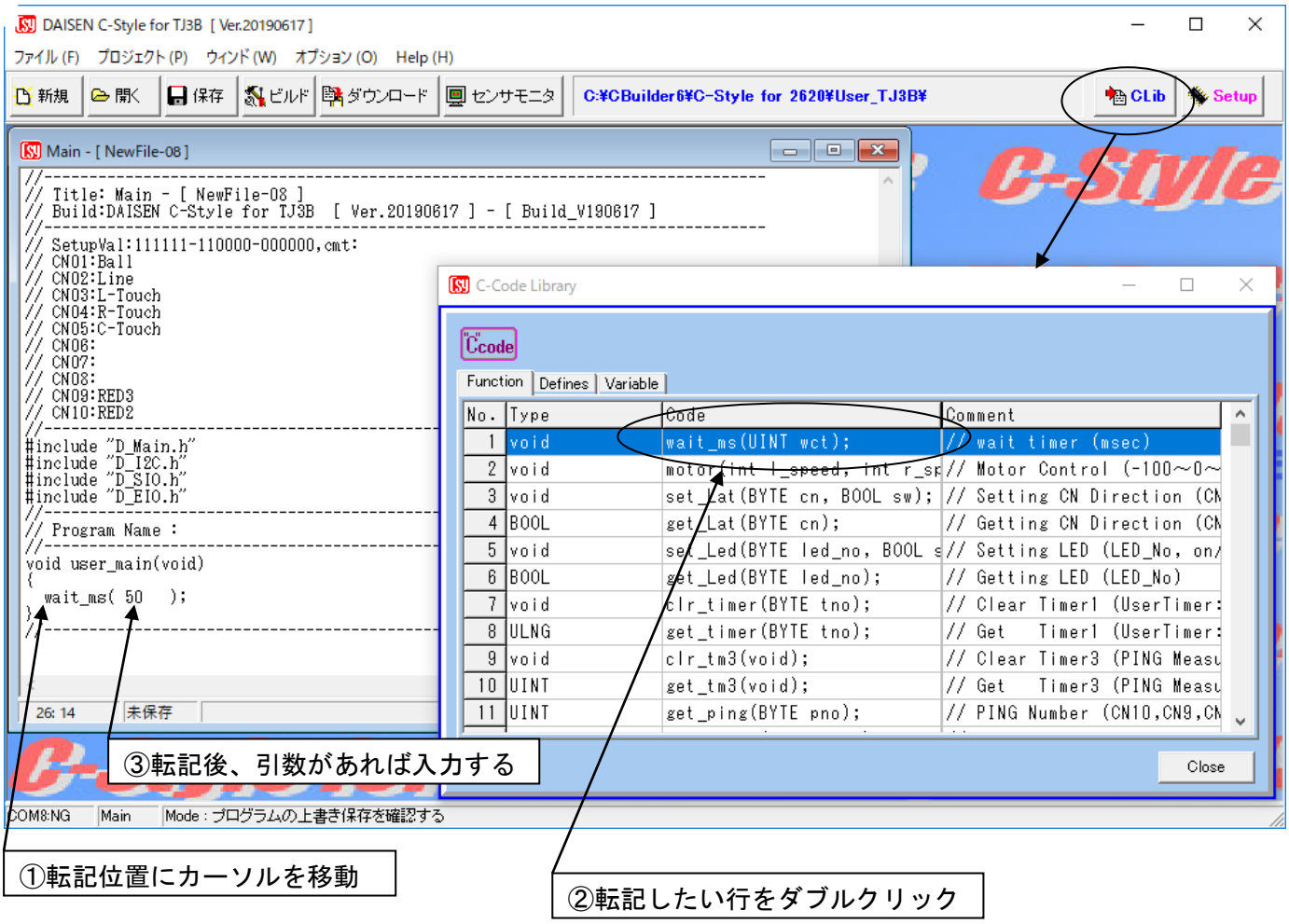
C-Code 編集を選択すると C-Style のプログラムボタンリストの表示が無くなり、全てＣ言語での編集となります。この画面で直接Ｃ言語のコードを記述するか、または、別のテキスト編集ソフトで編集したＣ言語ソースファイルを開くことも出来ます。
ビルド及びダウンロードは C-Style 同様に行うことが出来ます。

C-Code 編集モードでの入出力設定は Setup ボタンで行います。



C-Style の Ver. 190403 以降から Setup 情報は C-Code ソースファイルの先頭にコメントとして保存する機能を追加しましたので、次回ファイルを開いた時はその情報を自動的に読み込みます。
S-Style 編集モードでビルドした場合に生成される C-Code ソースファイルも同様に処理されます。

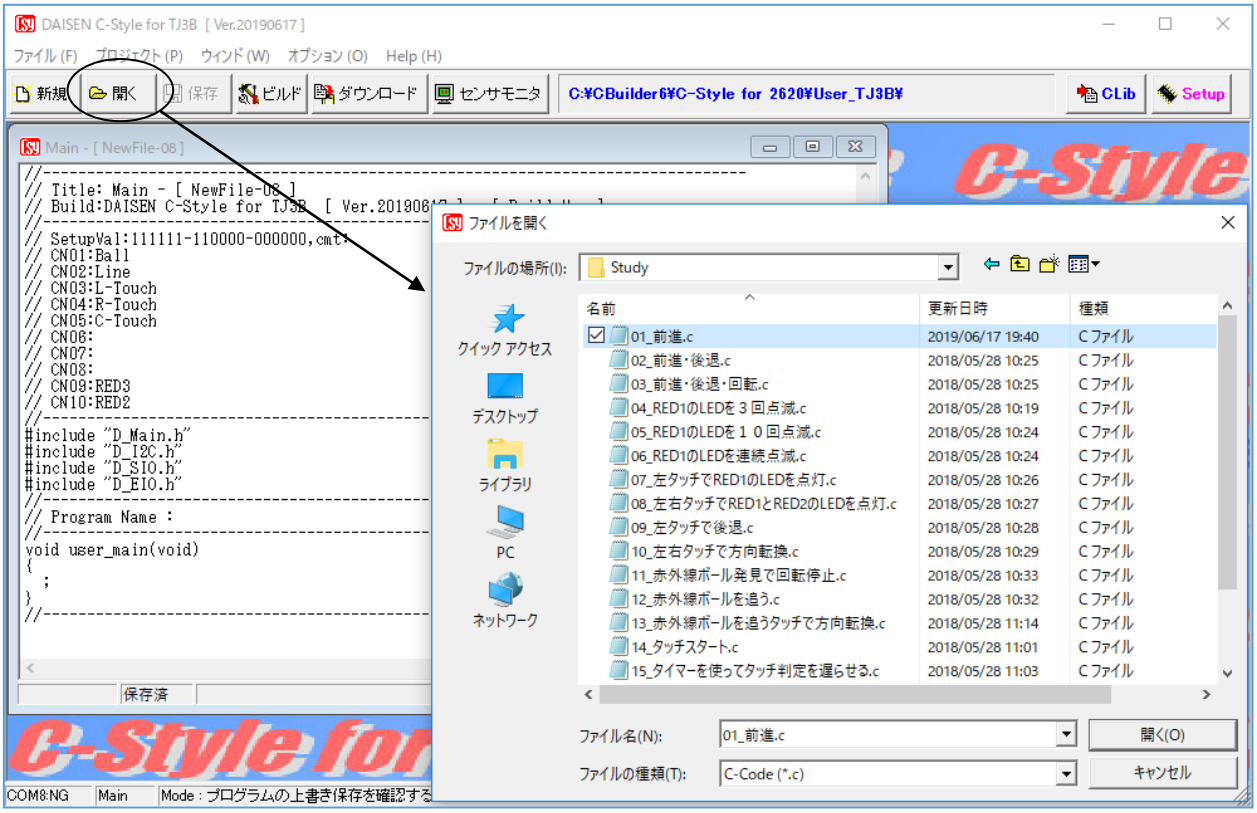
C-Code　Library の表示

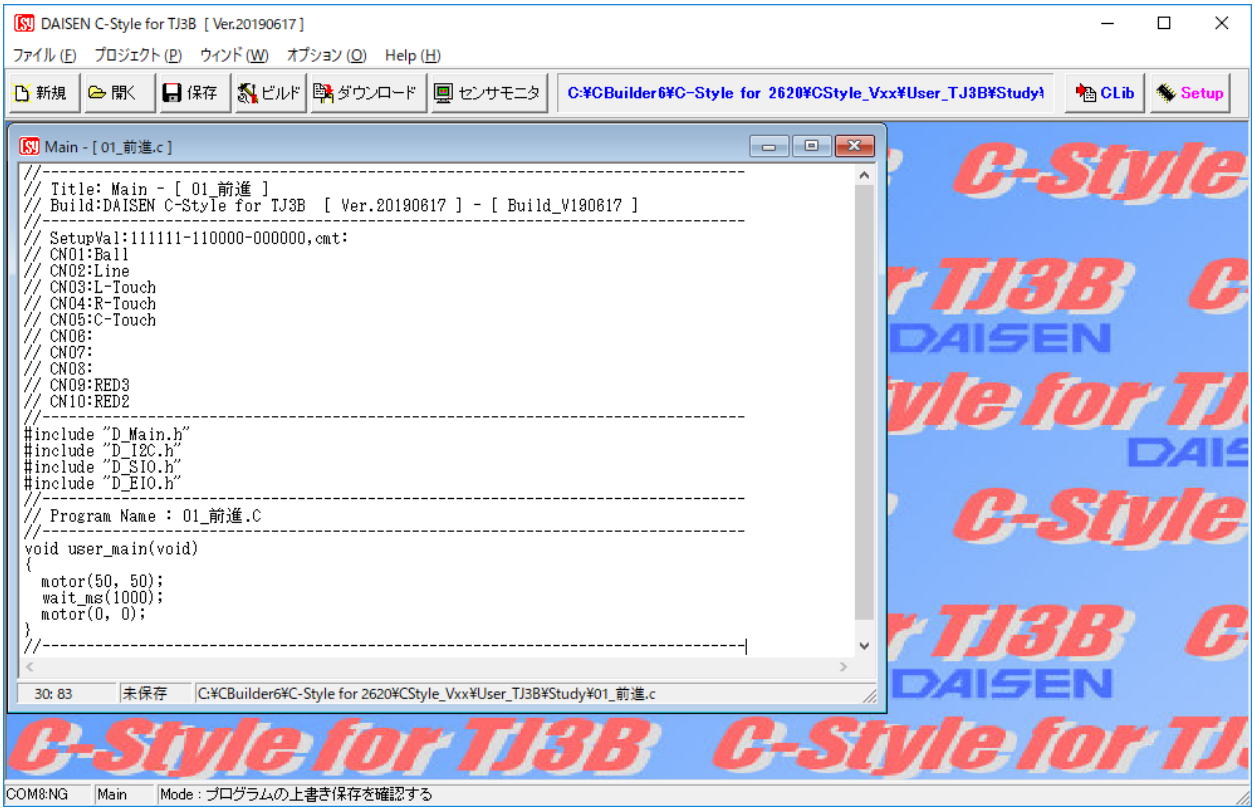


TJ3B で使用できる関数、定数定義文字、外部変数のライブラリを常に表示しながら C-Code の編集をサポートできます。該当する関数などにカーソルを移動してダブルクリックしますと、C-Code 編集領域のカーソル位置に転記されます。

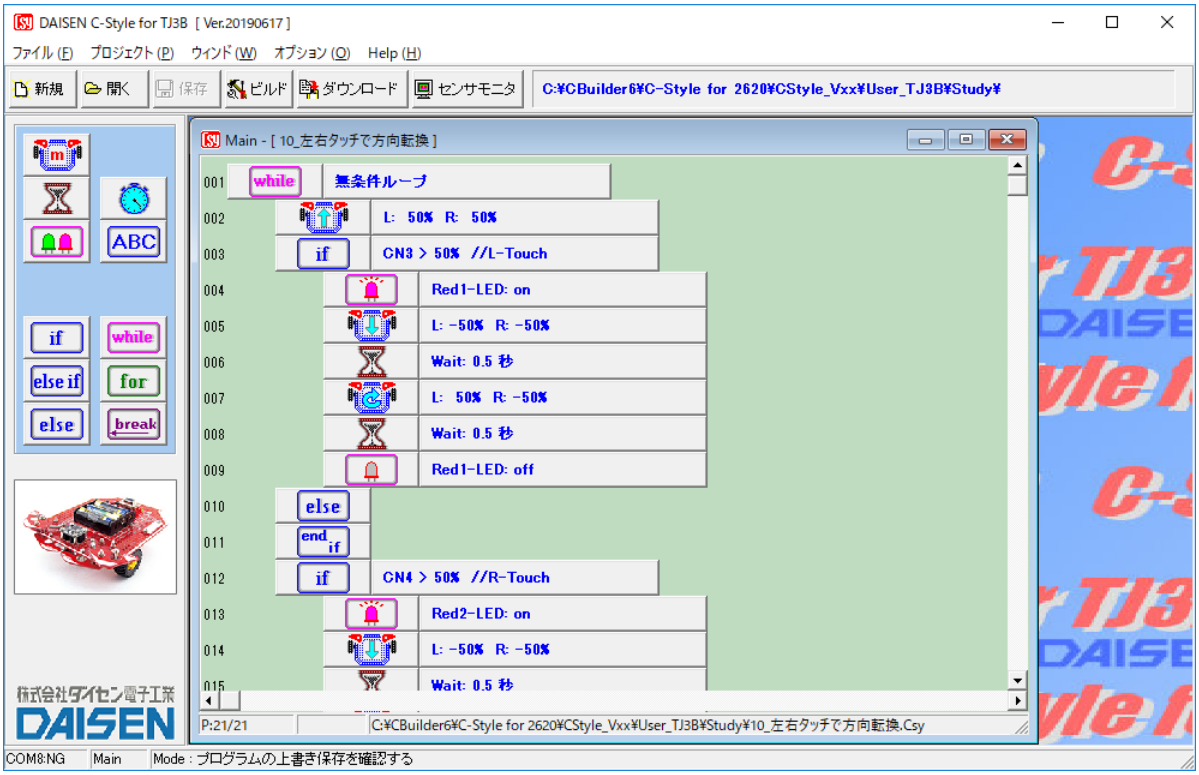
C-Style でビルドしたファイルは常にＣ言語ソースファイルとして残されていますので、一からＣ言語を記述することなく、C-Code 編集が行えます。

C-Style の１秒前進プログラムの C-Code ファイル（Ｃ言語ソースファイル）を開く

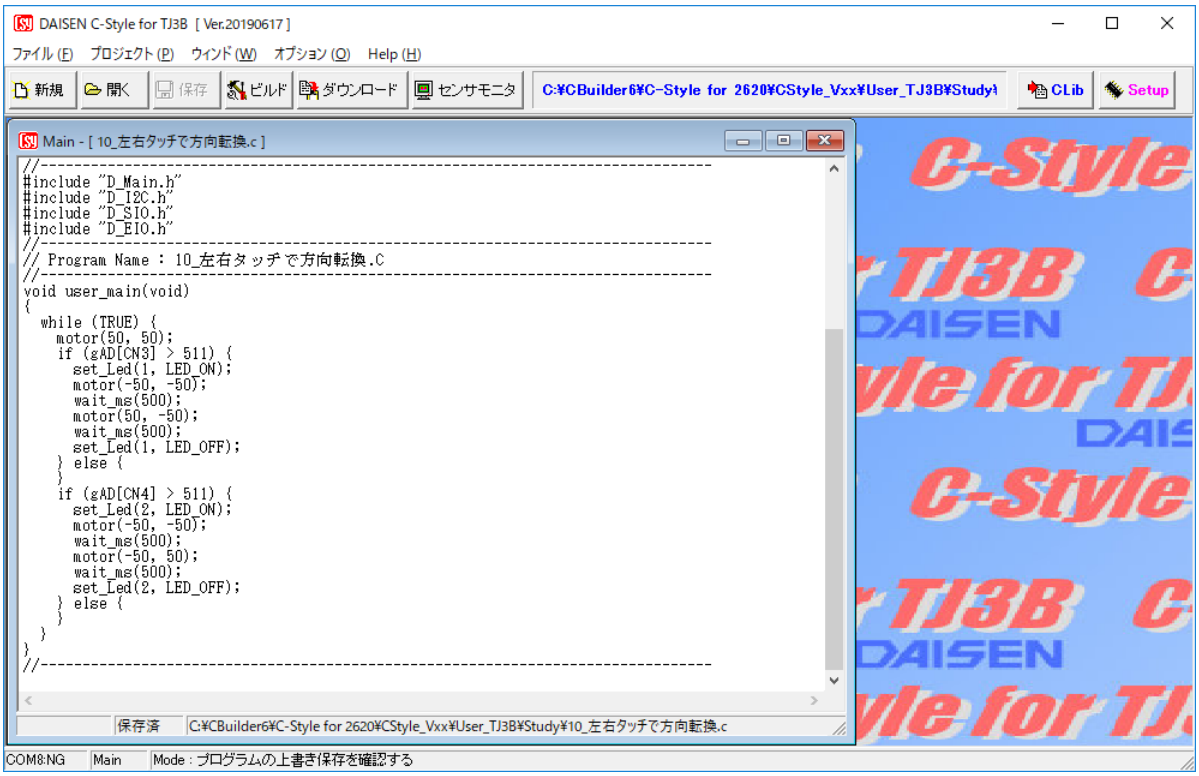




付属の C-Style プログラムをビルドする



C-Code 編集に切換えてからＣ言語ソースコードを開く



３．C-Code で PING(超音波距離センサ)を使う

関数名は　UINT get_ping(BYTE pno)
I/O Setup 画面の CN10～CN7 までを PING 設定に変更します。
計測には 20mSec プログラムは占有されます。(CN10～CN7 の 4 個の場合 20mSec×4=80mSec)

pno パラメータは、計測したＣＮ７からＣＮ１０の内、戻り値としてほしいＣＮ番号を指定
定数：CN7～CN10 (CN1～CN10 は D_Main.h で 0～9 の値が定数定義されています)

例：PING 設定が CN10～CN7 までの場合

```
  if (get_ping(CN10) < 100) motor(0,0);      // CN10 の値が 10cm 以下で停止
  if (get_ping(CN9 ) < 100) motor(50,0);     //  CN9 の値が 10cm 以下で右旋回
  if (get_ping(CN8 ) < 100) motor(0,50);     //  CN8 の値が 10cm 以下で左旋回
  if (get_ping(CN7 ) < 100) motor(-50,-50); //  CN7 の値が 10cm 以下で後退
   :
   :
```

入出力設定で PING 設定された CN の gAD[n]の値は get_ping(n)を呼出した時に計測され
A/D の値として格納されています。
PING 設定が CN10 のみの場合は、gAD[CN10]だけが超音波距離センサの計測値となります。

通常の A/D の値は 0～1023 ですが、PING の場合は、0～3000 の値が格納されます。単位：mm
但し 30mm 以下は計測不能または未接続、3000mm 以上は 3000mm の測定値となります。

４．C-Code で 4chUSS(超音波距離センサ PING アダプタボード)を使う

オプションパーツの 4chUSS は超音波センサ(PING)を最大４個接続して I2C 通信にて情報を得ることができます。ロボット本体に接続した場合と違って、1mS 程度の時間で情報を得ることが出来ます。
I/O Setup 画面の 4chUSS のチェックボックスにチェックマークを付けます。
関数名は UINT get_Uss(uss_no)
uss_no:0～3　戻り値:0：未接続、30～3000：有効(単位 mm)
外部変数　UINT gUss[4] は get_Uss()を一回呼出すだけで 4ch 分のデータが格納されます。
gUss[0]:1ch, gUss[1]:2ch, gUss[2]:3ch, gUss[3]:4ch に対応しています。
但し、超音波による距離測定の性質上データ更新は 20mS 毎となりますので、20mS 以内のデータ要求は、以前のデータを返すこととなります。

５．C-Code で電子コンパス(HMC6352)を使う

I/O Setup 画面の HMC6352 のチェックボックスにチェックマークを付けます。

関数名は　UINT get_hmc(void)
戻り値は地磁気の方位(0～359)の値が返されます。(0 度が北の方向です)

例１：地磁気の方位が北：0 度付近(±5 度)になったらモータ停止

```
UINT d;
d = get_hmc();
if (355 < d || d < 5) motor(0,0);
```

例２：地磁気の方位が北：180 度付近(±5 度)になったらモータ停止

```
UINT d;
d = get_hmc();
if (175 < d && d < 185) motor(0,0);
```

他社製の HMC6352 を使用する場合は、連続読出しを設定する　BOOL set_hmc(void)　を実行する必要があります（一度実行すればそのセンサは以後連続読出しの設定を記憶します）
※ダイセン製(DSR1302)の場合は、出荷時に実行されています。

６．C-Code で多機能電子コンパス(9D-Cmp/6D-Cmp)を使う

I/O Setup 画面の 9D-Cmp または 6D-Cmp のチェックボックスにチェックマークを付けます。

9D-Cmp：DSR1603　関数名は　UINT get_bno(BYTE dno)
6D-Cmp：DSR1401　関数名は　UINT get_dir(BYTE dno)
dno パラメータは、戻り値としてほしいデータ番号 0～2 を指定します。
　0：(Dir) 地磁気の方位(0～359)で　 0 度が北
　1：(Pitch) 前後の角度(0～359)で 180 度が水平　(6D-Cmp は 0～179 で 90 度が水平)
　2：(Roll)　左右の角度(0～179)で　90 度が水平

例：地磁気の方位が南：180 度付近(±5 度)になったら緑色 LED を点灯

```
UINT d;
while (LOOP){
  d = get_bno(0);        // 6D-Cmp の場合は    d = get_dir(0);
  if (175 < d && d < 185){
    LED_GREEN = LED_ON;
  } else {
    LED_GREEN = LED_OFF;
}
```

※C-Code ならではの便利な使い方
get_bno()または get_dir()を一回コールすると外部変数 UINT gDeg[3] に全ての情報が格納されます。　gDeg[0]が地磁気の方向,gDeg[1]が前後の水平角度,gDeg[2]が左右の水平角度です。

```
while (LOOP){
  get_bno(0);                              // 一回の呼出で Dir,Pitch,Roll を得る
  if (175 < gDeg[0] && gDeg[0] < 185){     // 南方向：180 度付近（±5 度）で
    LED_GREEN = LED_ON;                    // 緑色 LED を点灯
  } else {
    LED_GREEN = LED_OFF;                   // 緑色 LED を消灯
  }
  if (85 < gDeg[2] && gDeg[2] < 95){       // 左右が水平：90 度付近（±5 度）で
    LED_RED2 = LED_ON;                     // 赤色 LED2 を点灯
  } else {
    LED_RED2 = LED_OFF;                    // 赤色 LED2 を消灯
  }
}
```

７．C-Code でカラーイメージセンサ(Pixy Cam.)を使う

I/O Setup 画面の Pixy Cam.のチェックボックスにチェックマークを付けます。

関数名は以下の 6 個が使用出来ます。

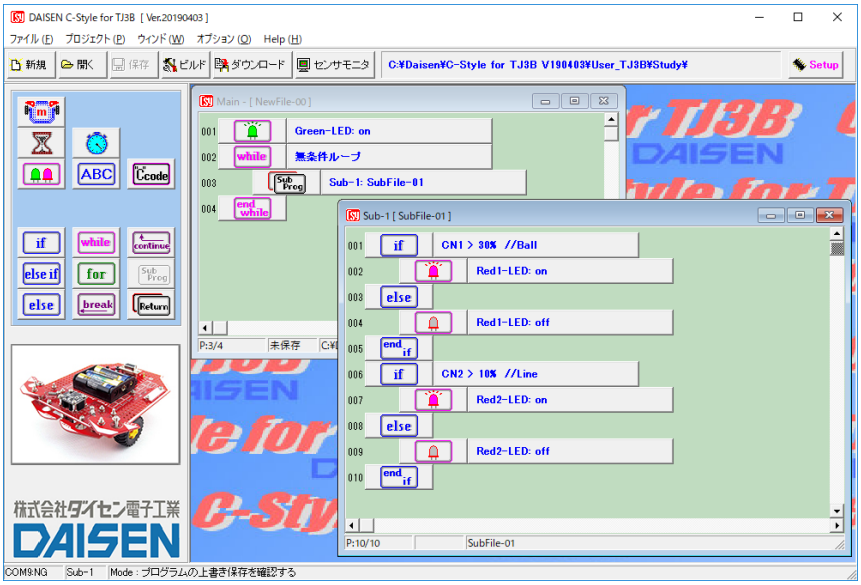
UINT get_pixydat_x(BYTE sig_no)
　　指定された sig_no：1～7 の中心水平座標値を返す（ 0:無、1～320:有効 ）
UINT get_pixydat_y(BYTE sig_no)
　　指定された sig_no：1～7 の中心垂直座標値を返す（ 0:無、1～200:有効 ）
UINT get_pixydat_w(BYTE sig_no)
　　指定された sig_no：1～7 の水平サイズを返す（ 0:無、1～320:有効 ）
UINT get_pixydat_h(BYTE sig_no)
　　指定された sig_no：1～7 の垂直サイズを返す（ 0:無、1～200:有効 ）
UINT get_pixydat_s(BYTE sig_no)
　　指定された sig_no：1～7 の面積を返す（ 0:無、100～64000:有効 ）
　　get_pixydat_w(sig_no)と get_pixydat_h(sig_no)を掛合せた値を返します。
BOOL chk_pixydat_p(BYTE sig_no, UINT pos)
　　指定された sig_no:1～7 が pos:9 分割された画面位置の有無を返す(true:有,false:無)
　　UINT pos 値の与え方
　　(MSB) 15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0 (LSB)　Position No.
　　POS No　－　－　－　－　－　9 8 7 － 6 5 4 － 3 2 1　　[1] [2] [3]
　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　[4] [5] [6]
　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　[7] [8] [9]
　　例：sig_no：1 が画面左側 [1],[4],[7]の何れかに現れたかを判定する場合

```
if (chk_pixydat_p(1, 0x0111)) {
   // 発見した処理
} else {
   // 未発見の処理
}
```

画面座標は左上側が x:0,y:0, で右下側が x:320,y:200 となります。
PixyCam のデータは 20mS 間隔で更新されるので、各関数を 20mS 以内にコールした場合は、更新前のデータを返します。

BOOL i2c_get_pixy(void) と　BYTE get_pixydat_index(BYTE sig_no) は上記 6 個の関数内でコールされる内部関数なので、特にコールする必要はありません。

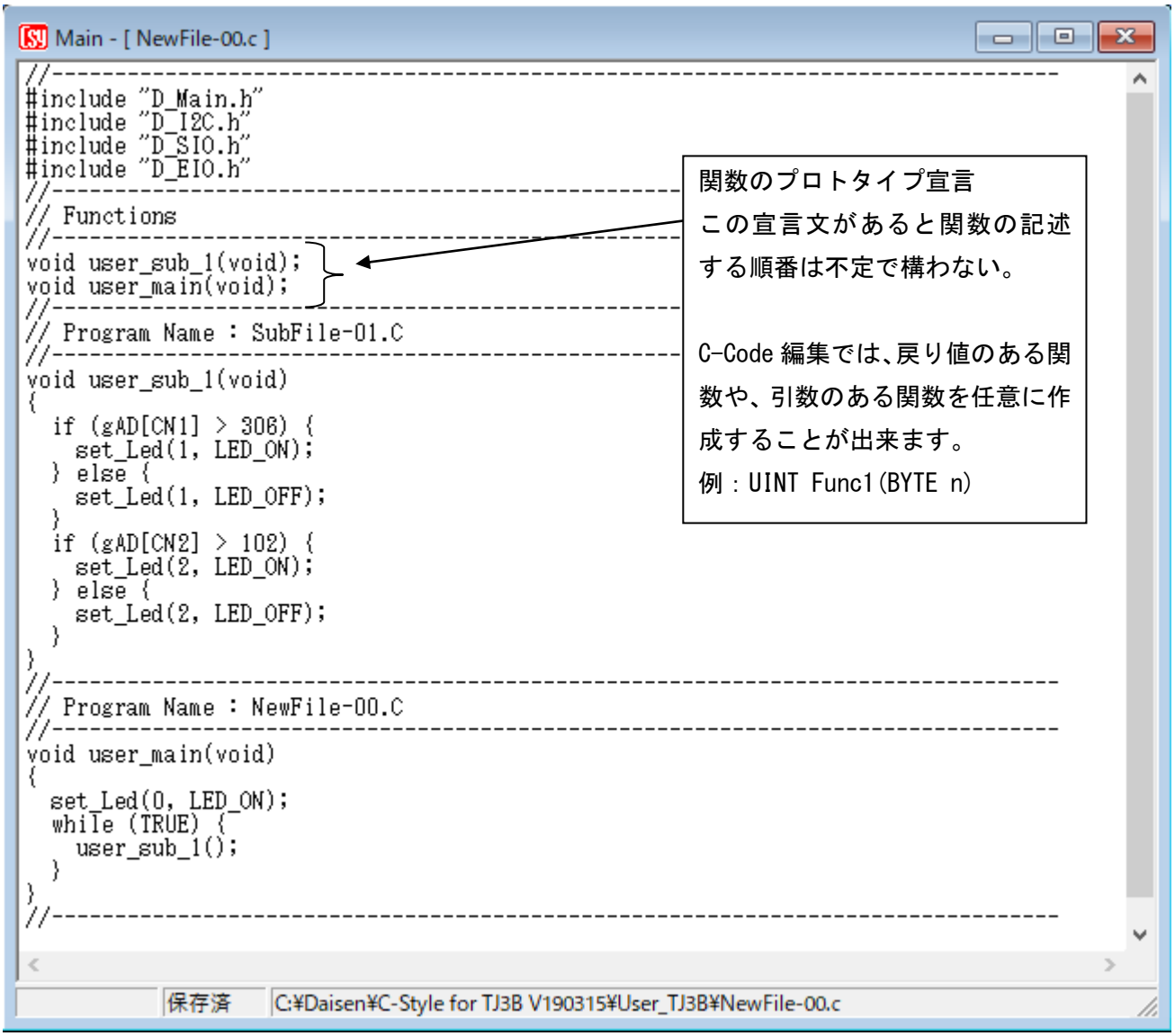
8. C-Code でサブプログラムを作成



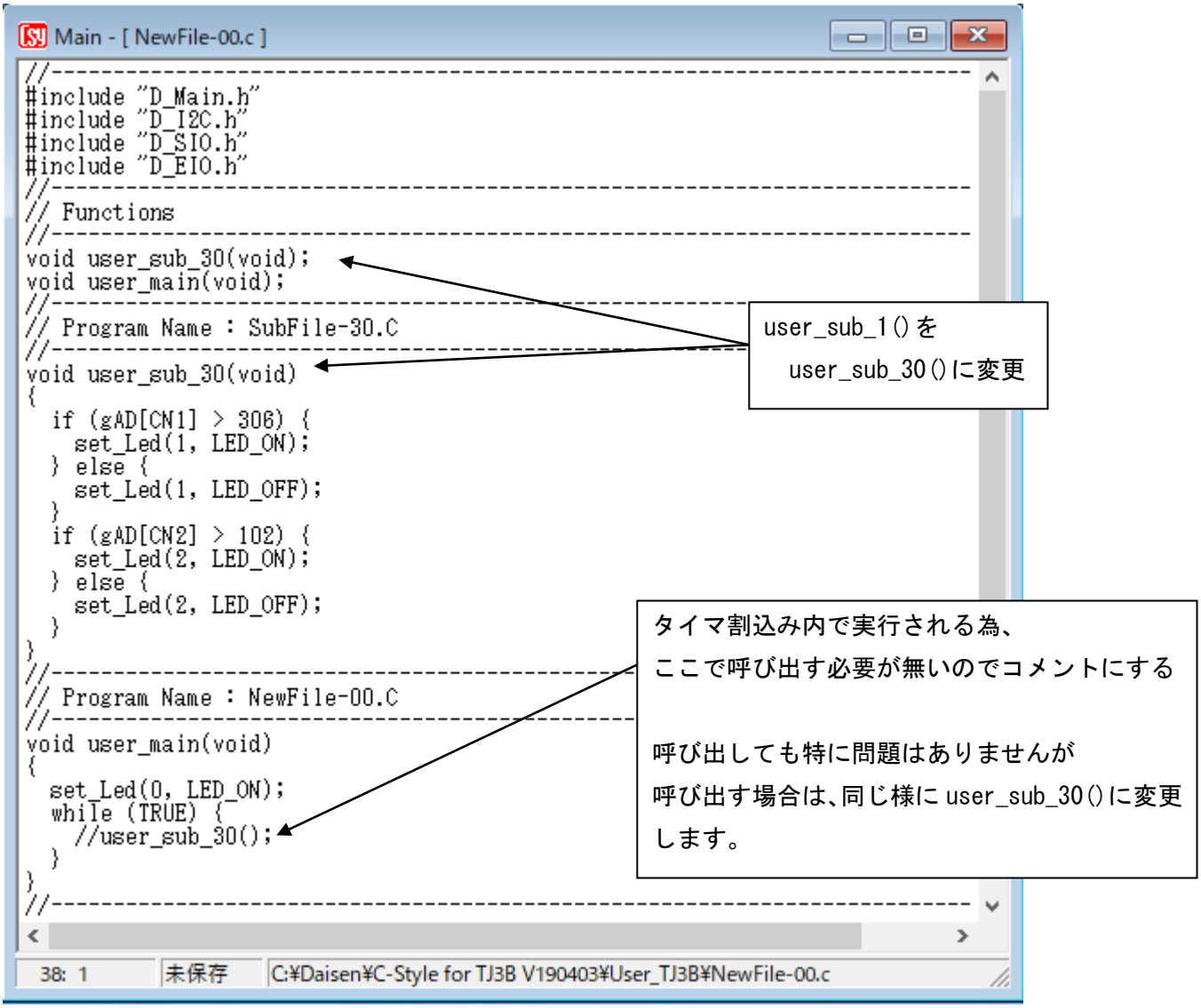
① 通常の C-Style の編集モードでサブプログラムを作成してビルドのみ実行します。

② 編集モードを C-Code に切り換えて①でビルドした時に作成される C-Code ファイルを開きます。

③ C-Code ファイルを編集後は C-Style へは変換されないので別の名前で保存します。



９．C-Code でサブプログラムをタイマ割込み内で実行させる方法



C-Style のファームウェア（Build_V19xxxx）内にある“D_Main.C”のタイマ割込み内で実行できるサブプログラムの名前は“**user_sub_30(void)**”と固定の名称で指定していますので、ユーザが C-Code 編集にてサブプログラムの関数名をこの名前で記述すると自動的にタイマ割込み内で実行されます。
上記の例では、user_sub_1(void) を user_sub_30(void)に変更しています。
また user_main()内で呼び出されていた user_sub_1()はメインでは呼び出す必要がなくなります

記述ルールとしては、センサ値の判定、LED の点灯制御、変数の演算などにして下さい。
タイマ割込みは 1mS 毎に発生しますので、処理時間の長い記述は避けて下さい。
またモータ制御関数や I2C 関連の関数を呼出すとプログラムの暴走が発生し致命的な故障になる可能性がありますので注意して下さい。

例：割込み内でセンサを監視してモータを止める方法

```
//-------------------------------------------------
// 割込み内で CN2 が 10%以上を監視
void user_sub_30(void)
{
   if (102 < gAD[CN2]){      // CN2 が 10% 以上を監視、102 = (10% × 1024) ÷ 100%
      gV[VAR_A] = 1;         // C-Style での変数 A のこと
   }
}
//-------------------------------------------------
// ユーザのメインプログラムで変数の変化を監視
void user_main(void)
{
  gV[VAR_A] = 0;                    // 最初は変数 A を 0 にすること
  while (LOOP){
    if (gV[VAR_A] == 0){            // 変数 A が 0 の時の処理
      motor(30, 30);                // 変数 A が 0 の間モータは 30%で前進
      get_ping(CN10);               // PING 情報の習得等の時間の長い処理中でも
      :                             // 割込み内で CN2 を監視しているので
      :                             // ここでの処理が終わった時に変数 A の値が 1 に
    } else if (gV[VAR_A] == 1){     // 変数 A が 1 の時の処理
      motor(-30, -30);              // モータを後退させる
      while (gAD[CN2] < 102);       // CN2 が 10%以下の間ループ(10%以上になるまで戻る)
      while (102 < gAD[CN2]);       // CN2 が 10%以上の間ループ(10%以下になるまで戻る)
      motor(0, 0);                  // モータを停止する
      gV[VAR_A] = 0;                // 変数 A を 0 に戻す
    }
  }
}
//-------------------------------------------------
```

この場合、割込み内で CN2 を常に監視しているので変数 A が 1 にされた場合、メインプログラムではどこで変数 A を 0 に戻すかがポイントとなりますので、よく考えてみて下さい。

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丁目 9-24
TEL 06-6631-5553　(FAX 06-6631-6886)
URL http://www.daisendenshi.com
Email ddk@daisendenshi.com