LogiCoA™電源ソリューション

スイッチング電源制御マイコン用オペレーティングシステム "RMOS"

概要

マイコンを用いてスイッチング電源装置の制御を行うためには、マイコンに搭載された機能（デジタル PWM、A/D コンバータ、通信インターフェース、データフラッシュメモリ等）を熟知し、詳細な管理を行う必要が有ります。また、スイッチング電源装置を多機能、且つ、高度に制御するためには大規模なプログラム作成が必要になります。さらには、1 個のマイコンで 2 種類の電源回路（例えば、PFC と DCDC コンバータ）を同時に制御する場合には、それぞれの電源回路の独立性と連携性を考慮する必要が有り、プログラム作成の難易度が飛躍的に高まります。

当社では、これらの課題を解決するために、スイッチング電源制御マイコン用オペレーティングシステム RMOS(アールモス；**R**eal time **M**icro **O**perating **S**ystem)を提供させて頂きます。RMOS は、ML62Q203x/ML62Q204x(ML62Q20xx グループ)上で動作します(2024 年 4 月 1 日現在)。RMOS の特長を以下に記します。

①マルチタスク・リアルタイム制御に対応したオペレーティングシステム
②制御プログラムのモジュール化によるプログラムの単純化と可読性（デバッグ性）の向上
③スイッチング電源制御用の基本機能（マイコンペリフェラルの初期設定、AD 値取得、状態モニタ、通信機能、ログ機能 (※1) 等）を実装
④電源回路の状態モニタと「状態変数」、「状態フラグ」の出力（バックグラウンド処理で実行）
⑤1 個のマイコンで 2 種類のスイッチング電源回路を同時に制御する機能を実装
⑥「状態遷移制御」を用いたスイッチング電源制御プログラミングに対応（RMOS に状態遷移管理機能を実装）
⑦スイッチング電源待機時にマイコンの消費電力を低減するローパワー動作モードを実装

（※1） 現在、ログ機能は未実装

RMOS では、マイコンの性能を 100%引き出すために、アセンブリ言語（CPU が直接実行することができる機械語を人間が容易に理解できる英単語に置き換えた言語）を用いて電源制御プログラムの記述を行います。『アセンブリ言語によるプログラミングは難しいのでは？』と思われるかもしれませんが、RMOS を使用することで、アセンブリ言語でも簡単にプログラムを作成することができます。

RMOS を使用した電源制御プログラムは、多くの場合で、RMOS が出力する状態変数(例：出力電圧 AD 値の 2 回移動平均値)や状態フラグ(例：RC 端子の起動/停止許可状態を示すフラグ)に対して処理を記述することでプログラミングを行うことができますので、他の言語と比較してもプログラミングの難易度は十分に低いものになっています。また、各種電源回路を実用レベルで制御することができるリファレンスプログラムを提供させて頂きますので、リファレンスプログラムの小変更等で電源制御プログラムを開発することが可能です。

本書では、RMOS の機能、動作、および、使用方法等について解説を行います。

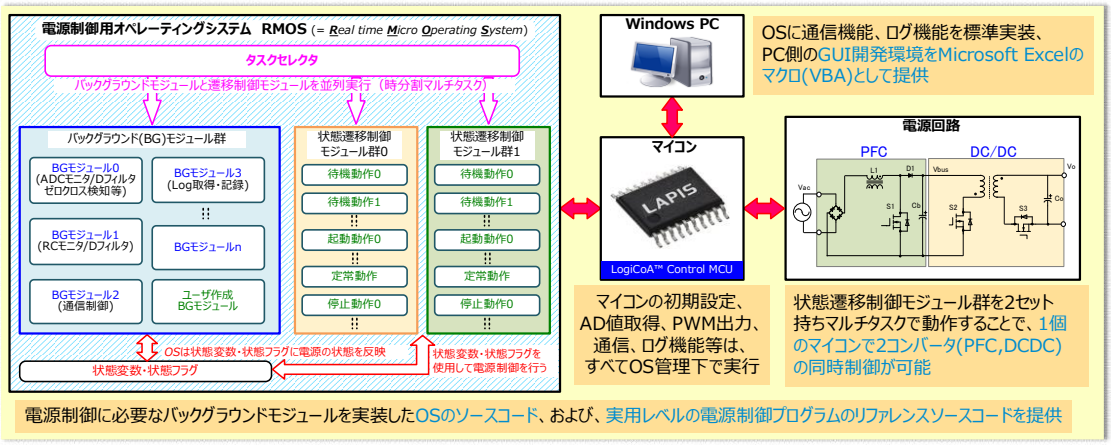

Figure. スイッチング電源制御マイコン用オペレーティングシステム "RMOS"の構成・機能

目次

1. RMOS によるスイッチング電源の制御(状態遷移制御)

RMOS では、「状態遷移制御」を用いてスイッチング電源の制御を行う機能を実装しています。以下では、スイッチング電源の状態遷移制御について説明を行います。

スイッチング電源装置の動作状態は、下記の 4 つの動作状態に分類することができます。

①待機動作 ・・・スイッチング電源装置が電圧を出力していない状態(入力電圧が起動開始電圧以下、リモート ON/OFF 制御による停止)
②起動動作 ・・・スイッチング電源装置の出力電圧がゼロから定常電圧に上昇して行く状態
③定常動作 ・・・スイッチング電源装置の出力電圧が定常電圧を出力している状態
④停止動作 ・・・スイッチング電源装置の出力電圧を停止させる状態

スイッチング電源装置の動作において上記の動作状態は独立していますので、制御プログラムも各動作状態に対して独立させて記述することができます。他の動作状態を考慮することなくプログラムを記述することができますので、プログラムを単純化して記述することができます。また、制御プログラムを記述する際には、上記の動作状態を更に細分化し、電源の動作状態に対してプログラムをモジュール化して記述します(状態遷移制御モジュール)。そして、「電源の状態に合わせて、実行するプログラムモジュールを変更(遷移)する」制御を行います(状態遷移制御)。

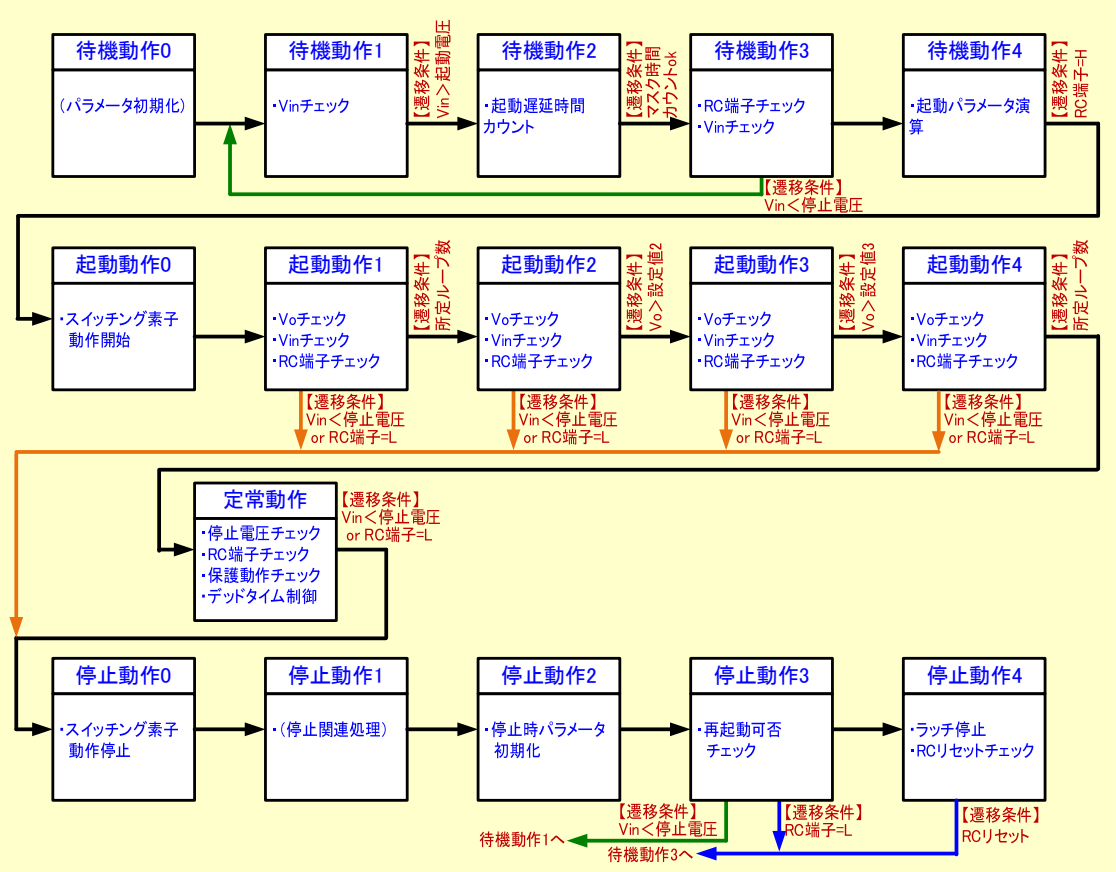

Figure 1-1. 状態遷移図(例)

RMOS では、予め、状態遷移制御モジュール記述エリアが準備されています(待機動作×5、起動動作×5、定常動作×2、停止動作×5)。プログラマは、RMOS の状態遷移制御モジュール記述エリアにプログラムを記述することで、電源制御プログラムを作成することができます。RMOS は、指定された状態遷移制御モジュールの何れかを 50μs 毎に実行します。また、電源の状態に応じて RMOS に指示を出すことで、実行する状態遷移制御モジュールを変更(遷移)することができます。また、RMOS では、2 種類のスイッチング電源回路を同時に制御するために、2 セットの状態遷移制御モジュール群が準備されています。

RMOS では、状態遷移制御モジュールの実行を遷移させる形式で電源制御を行いますので、プログラムの構成は、状態遷移図で表します。状態遷移図では、各状態遷移モジュールの動作、状態遷移の条件、状態遷移先を記述します。プログラマは、プログラムを開発するにあたっては、まずは、状態遷移図を作成し、各状態における電源の動作を検討することで、制御のヌケやモレを防ぐことができます。

2. RMOS のマルチタスク･リアルタイム制御

RMOS は、複数のタスク（プログラム）を並列処理（マルチタスク）する機能を備えたオペレーティングシステムです。RMOS のマルチタスク制御では、リアルタイム性(決められた時間間隔でプログラムを実行)が確保されるプリエンプティブマルチタスク（時分割方式）を採用しています。リアルタイム性を持つことで、制御プログラムの動作時間を正確に設計することができます(例：入力電圧が起動開始電圧に達してから電源が起動するまでの時間)。また、通常は有ってはならないことですが、何れかのタスクが暴走したとしても、他のタスクは正常に動作することができます。以下、RMOS のマルチタスク･リアルタイム制御の概要を示します。

①RMOS 内には、マルチタスクを制御するプログラムである「タスクセレクタ」と、マルチタスク制御の対象となるプログラムモジュールである「バックグラウンドモジュール群」と「状態遷移制御モジュール群」が準備されています。

②プログラマは、「バックグラウンドモジュール」と「状態遷移制御モジュール」内に設けられた「プログラム記述エリア」にプログラムを記述します。

③「タスクセレクタ」は、所定の周期(例：25μs、50μs、etc)に何れかのプログラムモジュールを実行します。また、所定の時間(例：7.5μs、9.5μs、etc)が経過すると、別のプログラムモジュールを実行します。

④プログラムモジュールの実行周期、割当時間は、予め RMOS で決められています。

⑤「バックグラウンドモジュール群」は、全てのモジュールが並列で実行されます。

⑥「状態遷移制御モジュール群」は、電源の状態に応じて、何れか 1 つのモジュールが実行されますが、定常動作のみ、2 つのモジュールが並列で実行されます。

⑦実行される状態遷移制御モジュールは、プログラム中で RMOS に指示を出します。

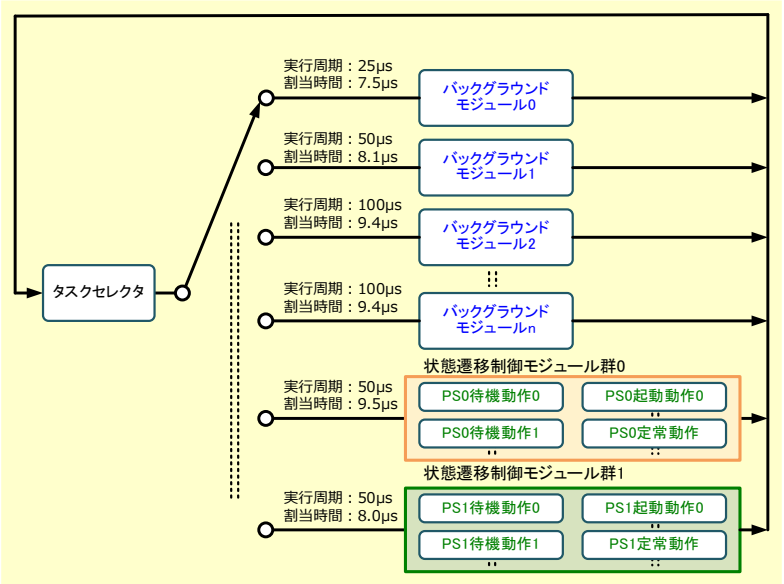

Figure 2-1. RMOS のマルチタスク･リアルタイム制御

Table 2-1 に RMOS で準備されているバックグラウンドモジュール群を示します。また、Table 2-2、2-3 に状態遷移モジュール群を示します。

Table 2-1. バックグラウンドモジュール群

No.	モジュール名	実行周期	割当時間
1	BG25u_Task	25μs	7.5μs
2	BG50u_Task	50μs	8.13μs
3	BG100u_Task	100μs	9.38μs
4	BG500u_Task	500μs	9.38μs
5	BG1m_Task0	1ms	9.38μs
6	BG1m_Task1	↑	↑
7	BG10m_Task0	10ms	↑
8	BG10m_Task1	↑	↑
9	BG25m_Task0	25ms	↑
10	BG25m_Task1	↑	↑
11	BG100m_Task0	100ms	↑
12	BG100m_Task1	↑	↑
13	BG100m_Task2	↑	↑
14	BG100m_Task3	↑	↑
15	BG100m_Task4	↑	↑
16	BG500m_Task0	500ms	↑
17	BG500m_Task1	↑	↑
18	BG500m_Task2	↑	↑
19	BG500m_Task3	↑	↑
20	BG500m_Task4	↑	↑
21	BG1000m_Task0	1000ms	↑
22	BG1000m_Task1	↑	↑
23	BG1000m_Task2	↑	↑
24	BG1000m_Task3	↑	↑
25	BG1000m_Task4	↑	↑

Table 2-2. 状態遷移モジュール群 0

No.	モジュール名	動作状態	割当時間
1	PS0_Standby_0	待機状態	9.50μs
2	PS0_Standby_1		↑
3	PS0_Standby_2		↑
4	PS0_Standby_3		↑
5	PS0_Standby_4		↑
6	PS0_Startup_0	起動状態	9.50μs
7	PS0_Startup_1		↑
8	PS0_Startup_2		↑
9	PS0_Startup_3		↑
10	PS0_Startup_4		↑
11	PS0_Normal_50u	定常状態	9.50μs
12	PS0_Normal_500u		9.38μs
13	PS0_Stop_0	停止状態	9.50μs
14	PS0_Stop_1		↑
15	PS0_Stop_2		↑
16	PS0_Stop_3		↑
17	PS0_Stop_4		↑

※1)制御周期：50μs
※2)電源の状態に対応する何れか 1 つのモジュールが実行される。定常動作のみ PS0_Normal_50u モジュールと PS0_Normal_500u モジュールが並列で実行される。

Table 2-3. 状態遷移モジュール群 1

No.	モジュール名	動作状態	割当時間
1	PS1_Standby_0	待機状態	8.00μs
2	PS1_Standby_1		↑
3	PS1_Standby_2		↑
4	PS0_Standby_3		↑
5	PS1_Standby_4		↑
6	PS1_Startup_0	起動状態	8.00μs
7	PS1_Startup_1		↑
8	PS1_Startup_2		↑
9	PS1_Startup_3		↑
10	PS1_Startup_4		↑
11	PS1_Normal_50u	定常状態	8.00μs
12	PS1_Normal_500u		9.38μs
13	PS1_Stop_0	停止状態	8.00μs
14	PS1_Stop_1		↑
15	PS1_Stop_2		↑
16	PS1_Stop_3		↑
17	PS1_Stop_4		↑

※1)制御周期：50μs
※2)電源の状態に対応する何れか 1 つのモジュールが実行される。定常動作のみ PS1_Normal_50u モジュールと PS1_Normal_500u モジュールが並列で実行される。

3. RMOS の使用方法

3-1. ソフトウエア・機材

RMOS を使用したスイッチング電源制御用プログラムの開発には、下記を使用します。

①統合開発環境 LEXIDE-Ω
②RMOS プロジェクトファイル(LEXIDE-Ωに読み込んで使用するファイル)
③Windows PC(Windows10 64bit 版 or Windows11 64bit 版)
④オンチップエミュレータ EASE1000 V2
⑤シリアル通信モジュール：FT234x FTDI 社製
⑥Microsoft Excel 64bit 版(通信機能の確認に使用、マクロ機能の使用許可が必要)

「統合開発環境 LEXIDE-Ω」は、オープンソースの統合開発環境である「Eclipse」をベースとして、開発されたソフトウエアです。PC にインストールして使用します。インストーラは、当社の web サイトからダウンロードできます。

「RMOS プロジェクトファイル」は、zip 圧縮形式で提供しています。Windows PC 内の HDD（SSD）ドライブ内の任意のフォルダに解凍して使用します。

「シリアル通信モジュール」は、降圧 DCDC コンバータ(以下、バックコンバータ) EVK LogiCoA001-EVK-001（以下、バックコンバータ EVK）に実装されています。このモジュールは PC と ML62Q203x/ML62Q204x（以下、ML62Q20xx グループ）とのシリアル通信を行うために使用します。

「オンチップエミュレータ EASE1000 V2」(以下、オンチップエミュレータ)は PC の USB 端子、および、マイコン ML62Q20xx グループのデバッグ用端子に接続して使用します。オンチップエミュレータと LEXIDE-Ωを使用することで、デバッグ作業（マイコンへのプログラムの書き込み、実行、停止、ステップ実行、内部メモリの読み出し等）を行うことができます。

オンチップエミュレータはマイコンデバッグ時の動作用電源として 3.3V のみをマイコンに供給することが可能な仕様となっています。そのため、ML62Q20xx グループ(電源電圧範囲：4.5-5.5V)のデバッグ作業を行うには、別途 5V 電源を準備する必要があります。そこでシリアル通信モジュールを用いることで PC から電圧 5V を供給します。ただし使用方法によってオンチップエミュレータの接続やバックコンバータ EVK のジャンパセット位置が異なります。

3-2. RMOS プロジェクトの読込・LEXIDE-Ωの使用方法

以下では、ML62Q2035 を搭載したバックコンバータ EVK を用いて、LEXIDE-Ωへの RMOS プロジェクトファイルの読み込み、バックコンバータ EVK 上のマイコン ML62Q20xx グループへのプログラムの書き込み、プログラムの実行/停止、および、RMOS へのプログラミング方法について説明を行います。

LEXIDE-Ωの PC へのインストール方法、使用方法の詳細等は、LEXIDE-Ωリリースノート[1]、LEXIDE-Ωユーザーズマニュアル[2]を参照してください。また、バックコンバータ EVK の使用方法(入力電源、負荷の接続方法等)、制御プログラムの解説等に関しても、バックコンバータ EVK ユーザーズガイド[3]を参照してください。

(1)接続方法

a.マイコン書き込みおよびデバッグ時の接続方法

バックコンバータ EVK、オンチップエミュレータの 14pin コネクタとフラットケーブルを Figure 3-1 のように接続します。バックコンバータ EVK のジャンパは、JP_REG を左側（マイコンのプログラム書き込み時に動作用電源を FT234x から給電）にセットし、JP_LDO はオープンにしてください（FT234x から LDO の出力端子への逆電流の防止）。

オンチップエミュレータと PC 間には①USBMiniB ケーブルを、シリアル通信モジュールと PC 間には②USBMicroB ケーブルを使用し、①②の順に PC に接続してください。

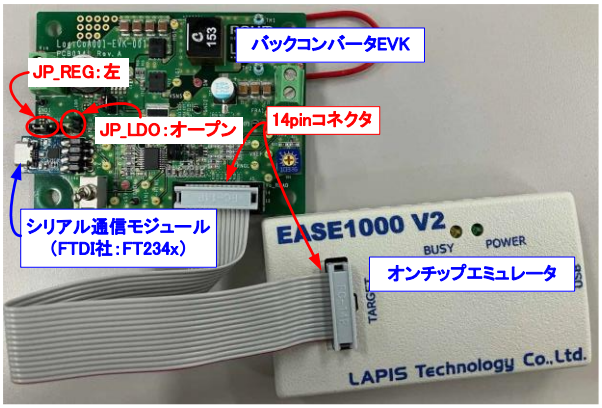

Figure 3-1. マイコン書き込みおよびデバッグ時の接続

b.バックコンバータ EVK 通常動作時の接続方法

バックコンバータ EVK の通常動作時は、Figure 3-2 のように接続します。通常動作中、シリアル通信モジュールは PC とマイコン間の通信を行うために使用します。この時マイコンの電源 5V は入力電圧端子に印加される電圧からレギュレータ IC を通じて供給されます。バックコンバータ EVK のジャンパは JP_REG を右側にセットし、JP_LDO はショートさせてください。このジャンパのセット時に、オンチップエミュレータは接続しないでください。オンチップエミュレータの 3.3V 出力端子に外部から電圧が印加され破壊の恐れがあります。

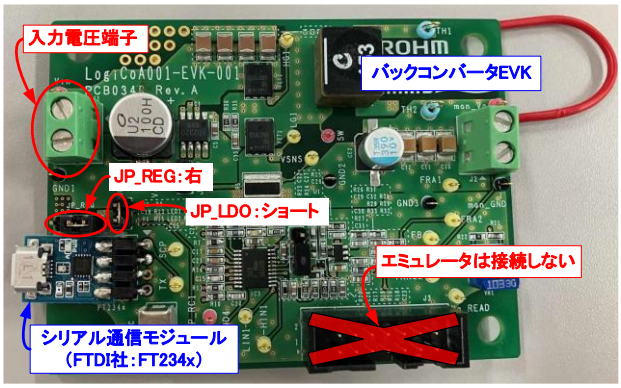

Figure 3-2. バックコンバータ EVK 通常動作時の接続

c.バックコンバータ EVK に入力電圧を印加した際のデバッグ時の接続方法

バックコンバータ EVK に入力電圧を印加した際に、LEXIDE-Ωでデバッグを行う時は、Figure 3-3 のように接続します。バックコンバータ EVK のジャンパは JP_REG をオープンに、JP_LDO をショートさせてください。JP_REG のジャンパを左右どちらかの位置にセットするとオンチップエミュレータの 3.3V 出力端子に外部から電圧が印加され破壊の恐れがあります。

デバッグ時の PC-マイコン間のシリアル通信は、Figure 3-3 の接続状態であっても、LEXIDE-Ωによりマイコンのプログラムが実行されていれば問題なく通信を行うことができます。

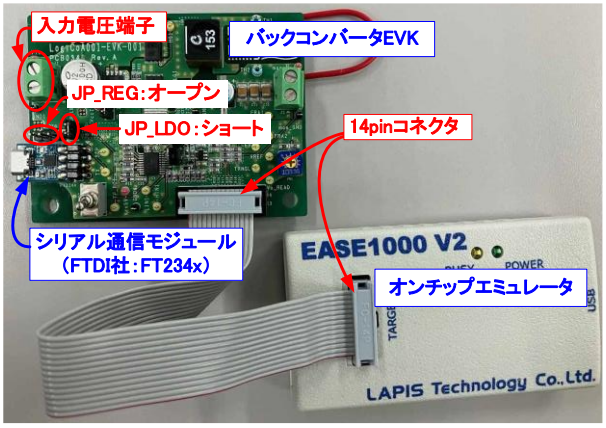

Figure 3-3. バックコンバータ EVK に入力電圧を印加した際のデバッグ時の接続

(2)バックコンバータ EVK 用 RMOS プロジェクトファイル(フォルダ)のコピー

バックコンバータ EVK 用の RMOS プロジェクトファイルとして、「RMOS100-ML62Q2035_Buck.zip」を提供しております。

「RMOS100-ML62Q2035_Buck.zip」を Windows のエクスプローラで開くと（RMOS100-ML62Q2035_Buck.zip ファイルをダブルクリックする）、Figure 3-4(a)のように zip ファイルの内容(RMOS100-ML62Q2035_Buck フォルダ)が表示されます。

LEXIDE-Ωをインストールした PC は、C ドライブ直下に「LAPIS」フォルダが生成されています（「LAPIS」フォルダ内には「LEXIDE」フォルダが生成されています）。「LAPIS」フォルダに zip ファイルの内の「RMOS100-ML62Q2035_Buck フォルダ」をコピーします。

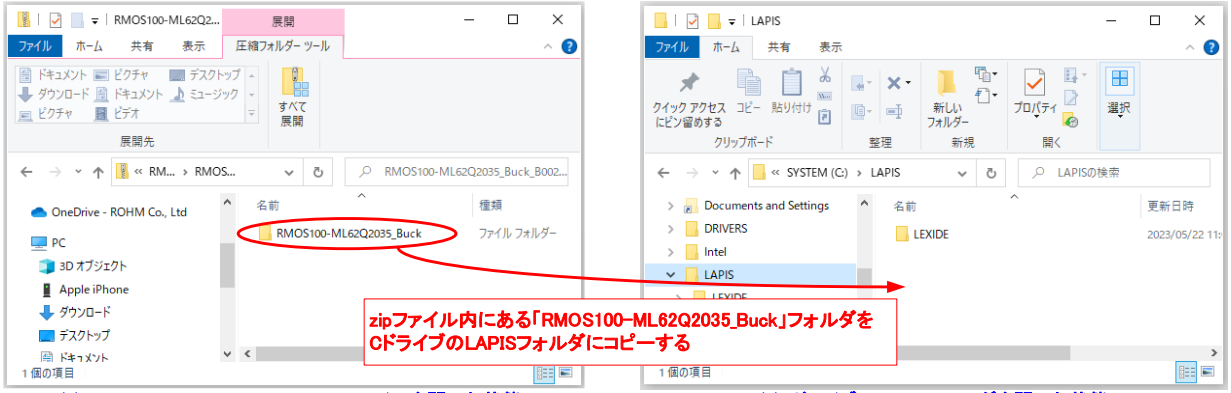

Figure 3-4. RMOS100-ML62Q2035_Buck フォルダのコピー方法

(※)RMOS プロジェクトファイル(フォルダ)は、任意のフォルダ(LEXIDE フォルダ以外)にコピーしても使用できます。

(3)LEXIDE-Ωの起動

下記の①~⑨を行うことで、LEXIDE-Ωを起動します。

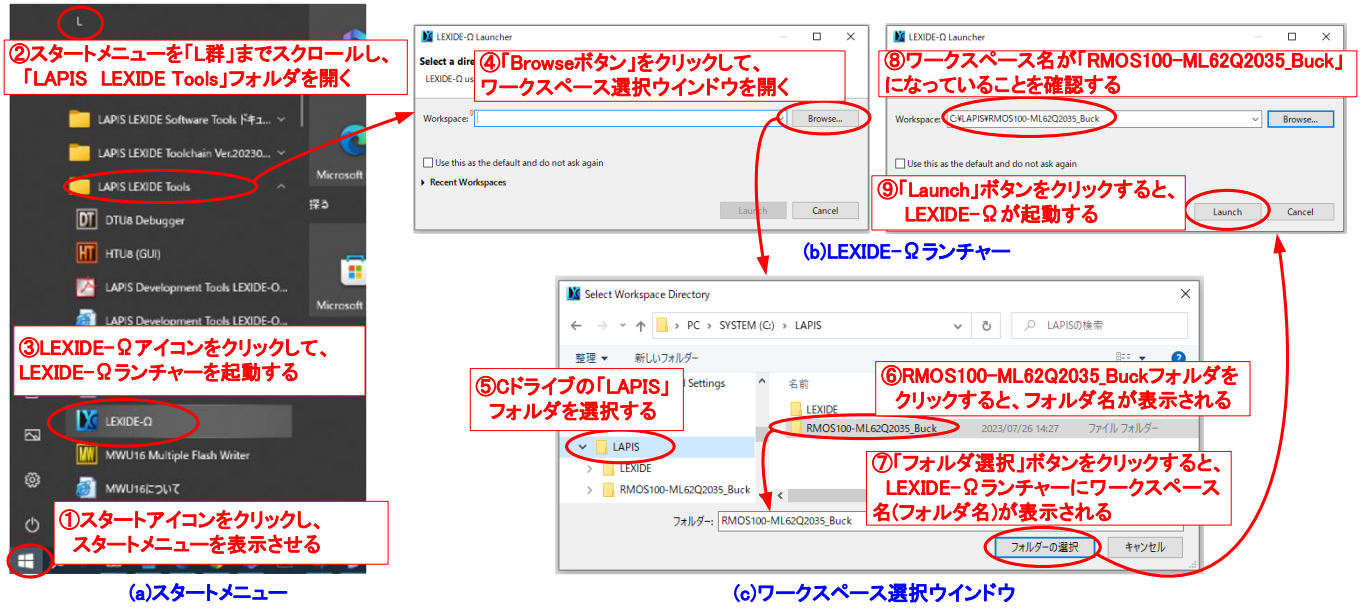

Figure 3-5. LEXIDE-Ωの起動

①Windows の「スタートアイコン」をクリックしてスタートメニューを表示させます(Figure 3-5a)。
②スタートメニューを「L 群」までスクロールし、「LAPIS LEXIDE Tools」フォルダを開きます。
③「LEXIDE-Ω」アイコンをクリックすることで、LEXIDE-Ωランチャーが起動します(Figure 3-5b)。
④LEXIDE-Ωランチャーの「Browse ボタン」をクリックして、ワークスペース選択ウインドウを開きます(Figure 3-5c)。
⑤ワークスペース選択ウインドウで、C ドライブの「LAPIS」フォルダを選択します。
⑥ワークスペース選択ウインドウで、「RMOS100-ML62Q2035_Buck」フォルダをクリックします。
⑦ワークスペース選択ウインドウで、「「フォルダ選択」ボタンをクリックすると、LEXIDE-Ωランチャーにワークスペース名(フォルダ名)が表示されます。
⑧LEXIDE-Ωランチャーのワークスペース名が「RMOS100-ML62Q2035_Buck」になっていることを確認します。
⑨LEXIDE-Ωランチャーの「Launch」ボタンをクリックすると、LEXIDE-Ωが起動します。

また、②の「L 群」にある「LAPIS LEXIDE Software Tools ドキュメント」フォルダには、LEXIDE-Ωユーザーズマニュアル[2]、アセンブリ言語のインストラクションマニュアル[4]もあります。LEXIDE-Ωの使用方法の詳細やアセンブリ言語の記述方法につきましては、こちらを参照してください。

(4)LEXIDE-Ωの基本操作

LEXIDE-Ωが起動すると、Figure 3-6 の画面が表示されます。RMOS のプログラミングおよびデバッグは、右上の「Debug」アイコンを選択した状態で行います。

画面の左下のプロジェクトエクスプローラには、RMOS のプログラムモジュール（バックグラウンドモジュール、状態遷移モジュール等）が表示されます。ここで、編集するプログラムモジュールを選択することができます。画面の中央下ではプログラムソースコードの編集を行うことができます。画面の右下は、マイコンプログラムの実行を一時停止した場合に、マイコンの内部情報が表示されます。

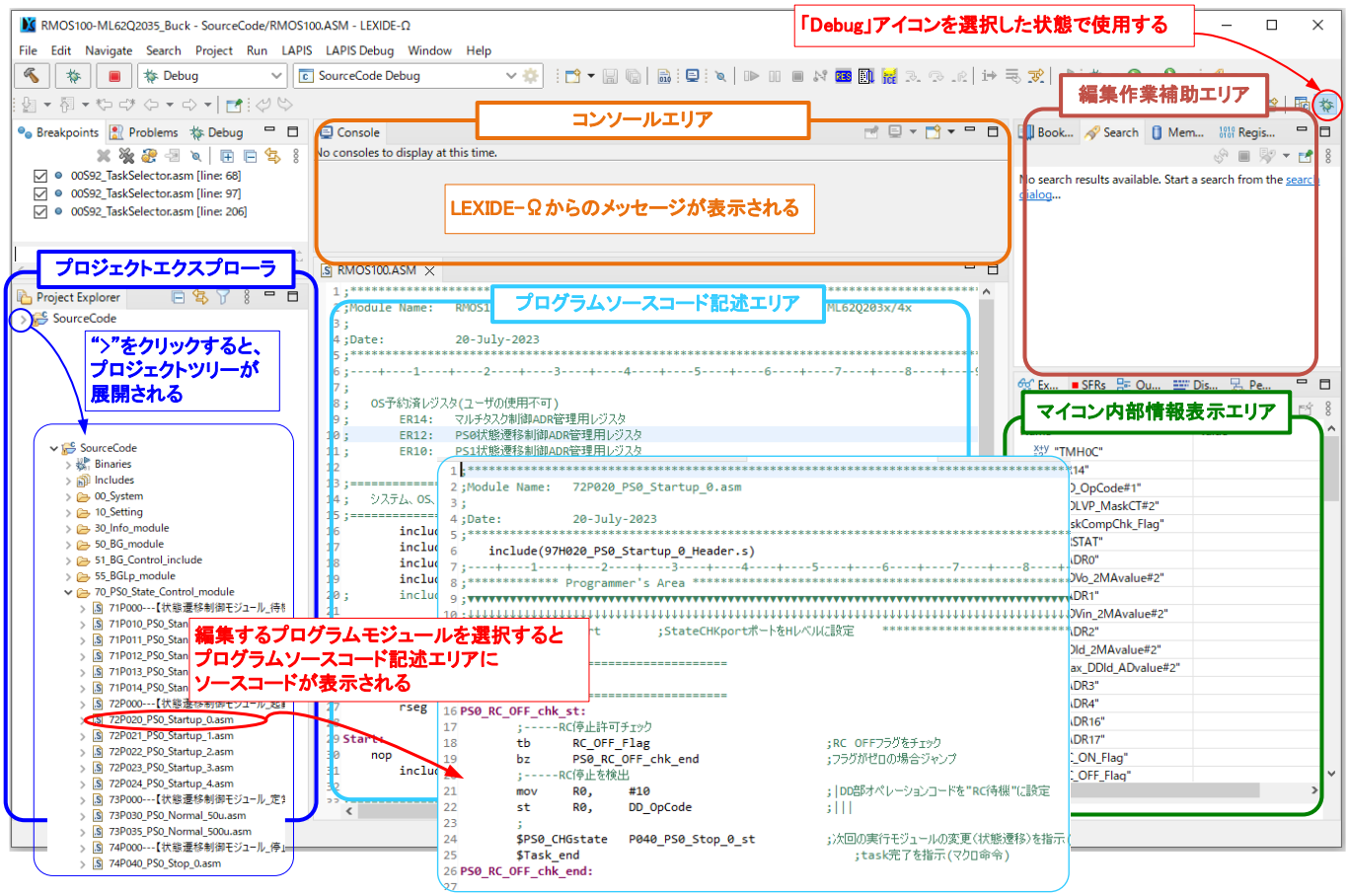

Figure 3-6. LEXIDE-ΩでバックコンバータEVK用のRMOSプロジェクトファイルを起動した時の画面

(5)プログラムのビルド

「ビルド」は、LEXIDE-Ω上で作成したプログラムをバックコンバータ EVK 上のマイコンへ書き込むことが可能なデータに変換するための作業です。ビルドを行うには、Figure 3-7 に示した下記①～③の操作を行います。

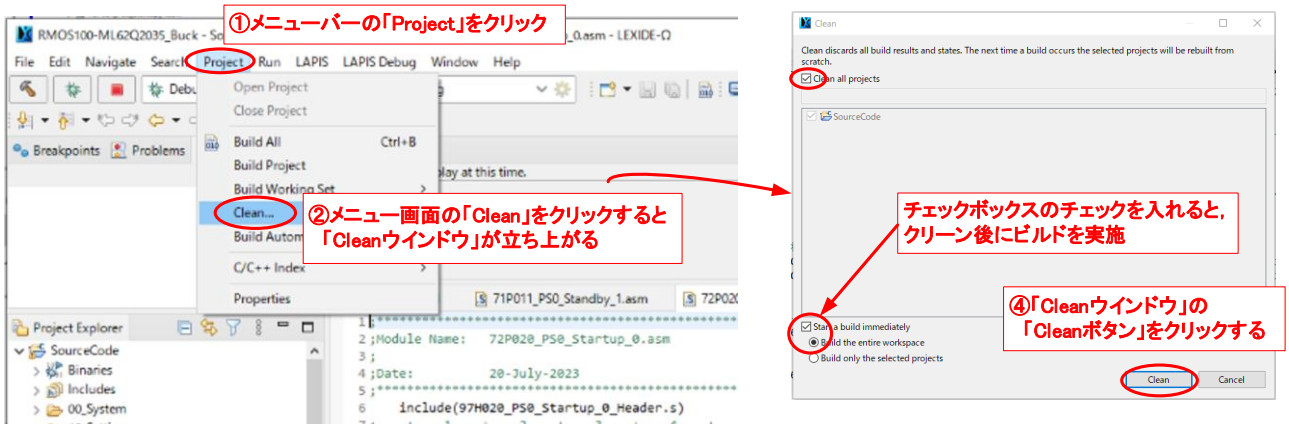

Figure 3-7. ビルド操作

①LEXIDE-Ωのメニューバーの「Project」をクリックするとメニュー画面が表示されます。
②メニュー画面の「Clean」をクリックすると、「Clean ウインドウ」が立ち上がります。
③「Clean ウインドウ」の「Clean ボタン」をクリックするとビルド作業が開始されます。「Start a build immediately」チェックボックスにチェックがあることでクリーン後にビルドが実施されます。

ビルドが開始されると、コンソールエリアにビルド作業の経過を示すメッセージが表示されます。

コンソールエリアに表示されるメッセージのスクロールが停止して、「Build Finished. 0 errors, 0 warnings」のメッセージが表示されましたら、ビルドが正常に完了しています。

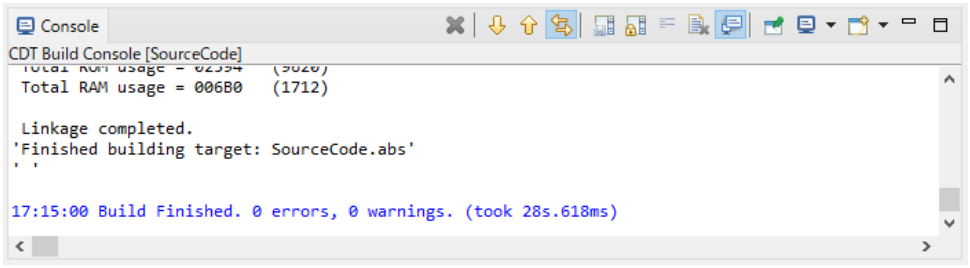

Figure 3-8. ビルドが正常に完了した時のコンソールエリアの表示

(6)マイコンへのプログラムの書き込み・実行・一時停止

バックコンバータ EVK 上のマイコンへのプログラムの書き込み、実行、一時停止を行うには、下記①~⑤の操作を行います。

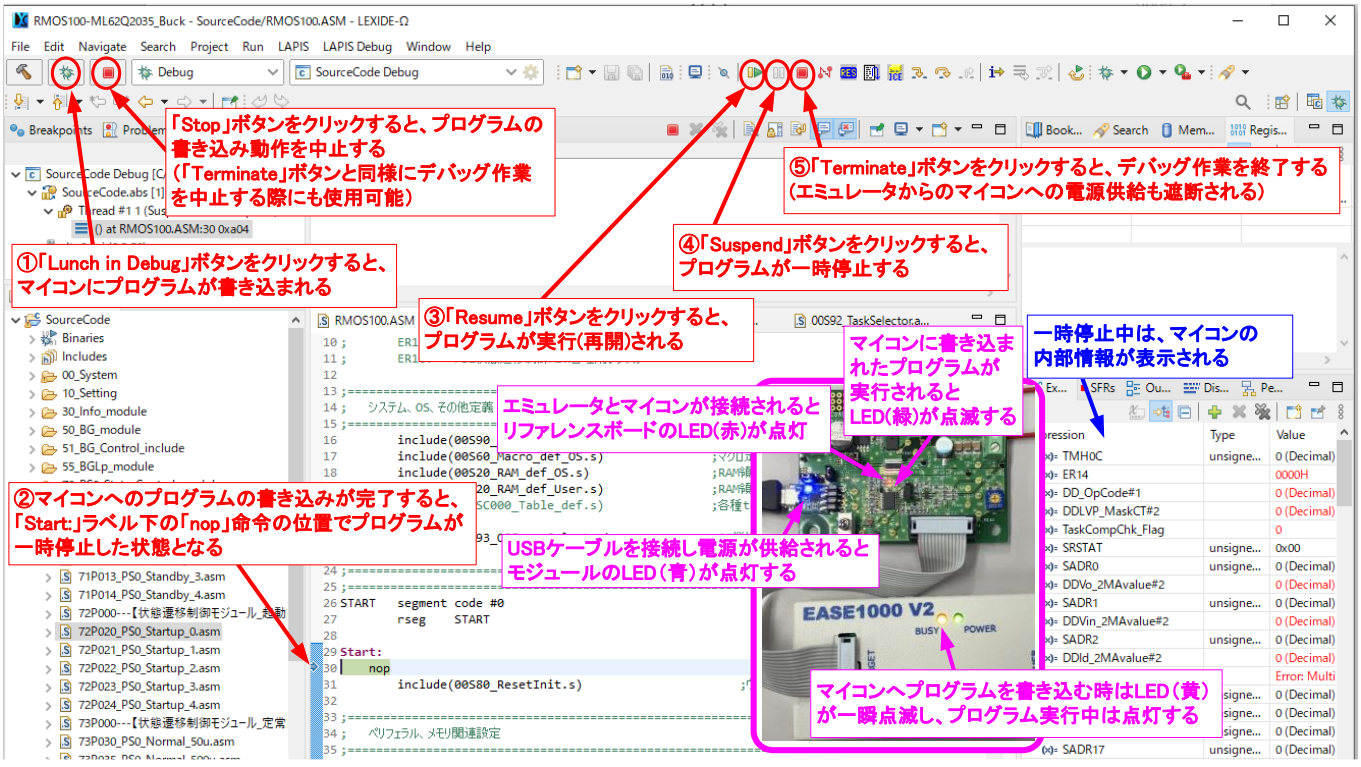

Figure 3-9. プログラムの書き込み・実行・一時停止

①「Launch in Debug」ボタンをクリックすると、マイコンへのプログラムの書き込みが開始されます。この時、シリアル通信モジュールからマイコンに電源が供給されていますので、オンチップエミュレータの LED(黄)が一瞬点滅しバックコンバータ EVK 上の LED(赤)が点灯します。シリアル通信モジュールは USB ケーブルを PC に接続すると LED(青)が常時点灯します。
②マイコンへのプログラムの書き込みが完了すると、「Start:」ラベル下の「nop」命令の位置でプログラムが一時停止した状態となります。
③「Resume」ボタンをクリックすると、プログラムが実行(再開)されます。プログラムが実行されると、オンチップエミュレータの LED(黄)が点灯しバックコンバータ EVK 上の LED(緑)が点滅します。LED(緑)の点滅は、電源の状態によって変化するようにプログラムされています。
④「Suspend」ボタンをクリックすると、プログラムが一時停止します。オンチップエミュレータの LED(黄)の消灯、バックコンバータ EVK の LED(緑)の点滅も一時停止します。
⑤「Terminate」ボタンをクリックすると、デバッグ作業を終了します(オンチップエミュレータとマイコンとの通信が遮断されるのでバックコンバータ EVK 上の LED(赤)が消灯しますが、シリアル通信モジュールの USB ケーブルを抜かなければ LED(青)は点灯しています)。

4. RMOS プロジェクトの構成と記述

RMOS プロジェクトには多数のファイル(プログラムモジュール)が格納されています。ファイルには、拡張子「.asm」と拡張子「.s」の 2 種類を使用しています。「.asm」ファイルは、LEXIDE-Ωがビルド対象とするファイルであり、それぞれのファイルが独立して取り扱われます。「.s」ファイルは、インクルードファイル(プログラムの可読性や移動性を高めるためにソースコードをまとめたファイル)として使用しており、「.asm」ファイル内に配置されるファイルです。

RMOS プロジェクトでは、ファイルをいくつかののフォルダに分類して格納しています。Table 4-1 にフォルダ一覧を示します(RMOS100.ASM はファイル)。

Table 4-1. プログラムモジュール格納フォルダー一覧

No.	フォルダ名(ファイル名)	内容
1	RMOS100.ASM	マイコンリセット時に実行されるプログラム(マルチタスク制御開始前)
2	00_System	マイコン選択、RAM 領域定義、RAM 初期値定義、OS 機能設定、OS 制御(マルチタスク制御等)プログラム
3	10_Setting	電源パラメータ初期設定、電源動作モード初期設定、マイコン内蔵機能(タイマ、A/D コンバータ等)の設定
4	30_Info_module	通信コマンド定義(作成)、通信コマンド実行プログラム
5	50_BG_module	バックグラウンドモジュール群
6	51_BG_Control_include	バックグラウンドモジュール群で使用するインクルードファイル
7	55_BGLp_module	ローパワー動作モード時に実行されるバックグラウンドモジュール群
8	70_PS0_State_Control_module	状態遷移制御モジュール群 0(例：DCDC コンバータ部の制御)
9	71_PS0_Control_include	状態遷移制御モジュール群 0 で使用するインクルードファイル
10	80_PS1_State_Control_module	状態遷移制御モジュール群 1(例：PFC 部の制御を想定)
11	81_PS1_Control_include	状態遷移制御モジュール群 1 で使用するインクルードファイル
12	90_Header	各プログラムモジュールの先頭に配置される OS 制御用プログラム

以下では「RMOS100-ML62Q2035_Buck」プロジェクトファイルを例として、プログラムモジュールおよびプログラムの記述に関して説明を行います。

4-1. 「RMOS100.ASM」ファイル(初期実行ファイル)

マイコンリセット時において、マルチタスク制御開始前に実行されるファイルです。マイコンにプログラムを書き込んだ直後は、本ファイルの「Start:」ラベルの下の「NOP」命令でプログラムが一時停止します。

本プログラムモジュールでは、マイコンの初期設定、マルチタスク動作の開始設定が行われます。通常は、このファイルを編集する必要はありません。

4-2. 「00_System」フォルダ(マイコンおよび RMOS システム設定)

本フォルダには、Table 4-2 のファイルが格納されています。以下では、編集の際に注意が必要なファイルについて説明を行います。

Table 4-2. 「00_System」フォルダ内のファイル

No.	ファイル名	内容
1	00S00_MCUselect.s	使用するマイコンの型番を選択する
2	00S60_Macro_def_OS.s	アセンブラ命令を拡張するためのマクロ命令を格納する
3	※00S80_ResetInit.s	マイコンリセット時の初期化処理
4	※00S92_TaskSelector.asm	バックグラウンドモジュール、状態遷移制御モジュールのマルチタスク・リアルタイム動作を制御
5	00S93_Func_Select.s	プログラムデバッグ時の OS の機能設定
6	※00S95_Module_Common.s	全ての「.asm」ファイルに対する定義を行う
7	※00S96_HBG_module.s	OS が使用するバックグラウンドモジュール
8	※00S97_LpEnterExit.asm	ローパワー動作モードへの切替/復帰を制御
9	※00S98_Idling.asm	各プログラムモジュールにおいて実行完了時の待機用
10	※00S99_VectorTBL_OpByte_def.s	マイコンの初期化定義

※は通常は編集する必要はありません

(1)「00S00_MCUselect.s」ファイル（マイコン型番選択）

使用するマイコンの型番を変更する時は、Figure 4-1 に従って、編集作業を行います。

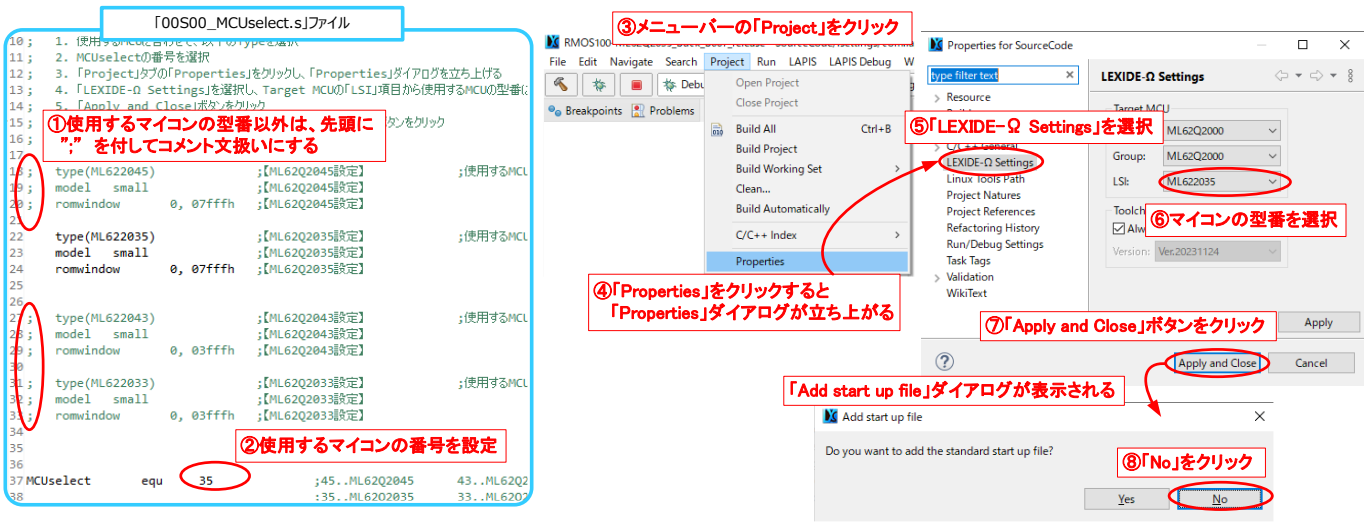

Figure 4-1. マイコンの型番変更

注意：⑧の手順で誤って「Yes」をクリックすると、「SourceCode」フォルダ内に「ML6220xx.ASM」ファイルが生成されます。この状態でビルドを実行するとエラーメッセージが表示されます。そのため、「ML6220xx.ASM」ファイルは必ず削除した後にビルドを実行してください。ファイルが削除できていれば、Figure 4-1 の手順を再度行う必要はありません。

(2)「00S60_Macro_def_OS.s」ファイル（マクロ命令の定義）

「00S60_Macro_def_OS.s」ファイルは、RMOS 内のプロジェクト全体で使用するマクロ命令の定義を行います。マクロ命令は、RMOS に対して指示を行う命令と ML62Q20xx グループのニーモニックの拡張を行うために使用しています。RMOS では、マクロ命令は先頭に"$"を付すこととしています。Table 4-3 に RMOS のマクロ命令一覧を示します。

Table 4-3. RMOS のマクロ命令

No.	マクロ命令	分類	内容
1	$Task_end	RMOS への指示	タスクが完了したことを RMOS に指示
2	$SlowTask_end		タスクが完了したことを RMOS に指示(実行周期 10ms 以上のモジュール)
3	$LpTask_end		ローパワー動作モードにおいてタスクが完了したことを RMOS に指示
4	$PS0_CHGstate ラベル名		状態遷移制御モジュール群 0 において、次回周期で実行するモジュールを RMOS に指示
5	$PS1_CHGstate ラベル名		状態遷移制御モジュール群 1 において、次回周期で実行するモジュールを RMOS に指示
6	$add16__ERn 数値(シンボル名)	ニーモニック拡張	ERn(n=0,2,4,6,8)レジスタに 16bit 長の数値を加算
7	$cmp16__ERn_ERm		ERn(n=0,2,4,6,8)レジスタと ERm(m=0,2,4,6,8)レジスタの比較
8	$mov16__ERn 数値(シンボル名)		ERn(n=0,2,4,6,8)レジスタに 16bit 長の数値を代入
9	$sll16__ ERn 数値(シンボル名)		ERn(n=0,2,4,6,8)レジスタを数値で指定された bit 数分だけ左にシフト
10	$srl16__ ERn 数値(シンボル名)		ERn(n=0,2,4,6,8)レジスタを数値で指定された bit 数分だけ右にシフト
11	$sub16__ERn_ERm		ERn(n=0,2,4,6,8)レジスタと ERm(m=0,2,4,6,8)レジスタの減算

4-3.「10_Setting」フォルダ(マイコンハードウエア設定)

本フォルダには、Table 4-4 のファイルが格納されています。以下では、編集の際に注意が必要なファイルについて説明を行います。

Table 4-4.「10_Setting」フォルダ内のファイル

No.	ファイル名	内容
1	10S01_Parameter_Init.s	電源の動作パラメータ初期値を設定
2	10S05_PS_Mode_Set.s	電源動作において動的に変更可能にプログラムされた項目の初期値を設定
3	10S20_RAM_def.s	RAM 上に変数領域を確保するためのラベルを定義する
4	10S21_RAM_Init.s	RAM 上に確保した変数に初期値を代入する
5	10S25_Extrn_def.s	「10S20_RAM_def.s」で定義にしたラベルを他のプログラムモジュールで使用可能にするための設定
6	10S50_GPIO_Set.s	デジタル入出力ポートの割付け・動作設定
7	10S51_OperationalTimer_Set.s	オペレーショナルタイマ(電源制御用タイマ)の設定
8	※10S52_16bitTimer_Set.s	16bit タイマの設定(マルチタスクの制御に使用)、設定変更不可
9	10S53_CMP_Set.s	コンパレータの割付け・動作設定
10	10S54_EXTRG_Set.s	外部トリガ入力の割付け・動作設定(バックコンバータでは未使用)
11	10S60_ADconv_TEMPsens_PGA_Set.s	A/D コンバータ、PGA の割付け・動作設定、温度センサ設定
12	10S61_DAconverter_Set.s	D/A コンバータの動作設定
13	※10S70_UART_Set.s	UART の割付け・動作設定

※は通常は編集する必要はありません

(1) バージョン管理

RMOS のバージョンは「10S01_Parameter_Init.s」ファイル内にシンボル名で記述されています。

「RMOSVer」(OS バージョン)、「OSBuildNo」(OS ビルドナンバ)はバージョンの数値を表しています。弊社からリリースする際にこのバージョンが更新されます。そのため、この値を編集する必要はありません(Figure 4-2 は 2024/4/1 時点のバージョンになります)。

「PSFMNo」は RMOS で制御する電源トポロジを表す数値が記述されています(バックコンバータは"001")。また、「PSFMVer」、「PSFMBuildNo」は制御する電源トポロジのファームウェアのバージョンを表しています。これは設計した電源で状態遷移モジュール、バックグラウンドモジュール及びペリフェラルの設定などの記述を変更した際にバージョンの更新を行います。こちらのシンボル値は自由に書き換えが可能です。

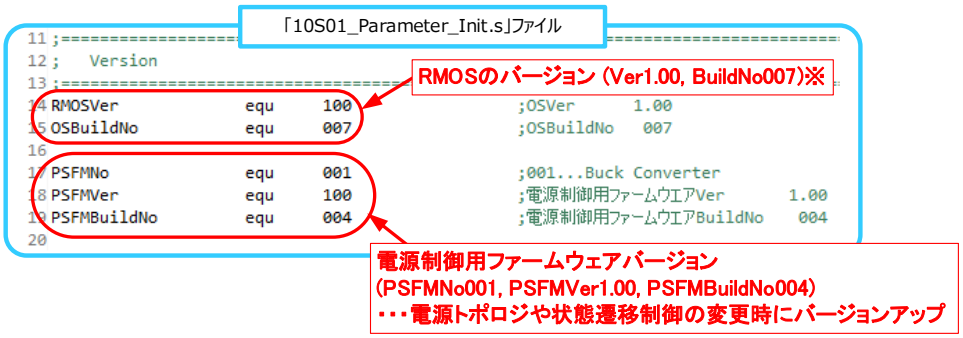

Figure 4-2. バージョンの記述

※は通常は編集する必要はありません

(2) 変数確保 (RAM 領域の定義)

RAM 領域に変数を確保するためには、以下の 4 つのファイルを編集します。

①10S20_RAM_def.s ②10S25_Extrn_def.s
③10S01_Parameter_Init.s ④10S21_RAM_Init.s

「10S20_RAM_def.s」ファイルは、RAM 上に変数領域を確保(ラベルを定義)するために使用します。定義したラベルは、本ファイル内で public 宣言を行います。

「10S25_Extrn_def.s」ファイルで extrn 宣言を行うことで、RMOS 内の他のプログラムモジュール(「.asm」モジュール)で使用できます。

「10S01_Parameter_Init.s」ファイルは、RAM 上に確保した変数の初期値を定義するために使用します。初期値は、まず、シンボル名を定義し、シンボルに対して数値を与えます。

「10S21_RAM_Init.s」ファイルは、「10S20_RAM_def.s」ファイルで確保した変数に対して、「10S01_Parameter_Init.s」で定義したシンボルの数値を代入します。

Figure 4-3. RAM 上への変数定義と public/extrn 宣言・初期値の代入

(3)「10S50_GPIO_Set.s」ファイル(デジタル入出力として定義する I/O ポートの設定)

GPIO ポートを入力端子、もしくは、出力端子として使用する場合の設定を行います。また、GPIO ポートへのシンボル割付けを行います。GPIO ポートへ割付けを行ったシンボルは、本ファイル内で public 宣言を行い、「10S25_Extrn_def.s」ファイルで extrn 宣言を行うことで、RMOS 内の他のプログラムモジュール(「.asm」モジュール)で使用できるようになります。

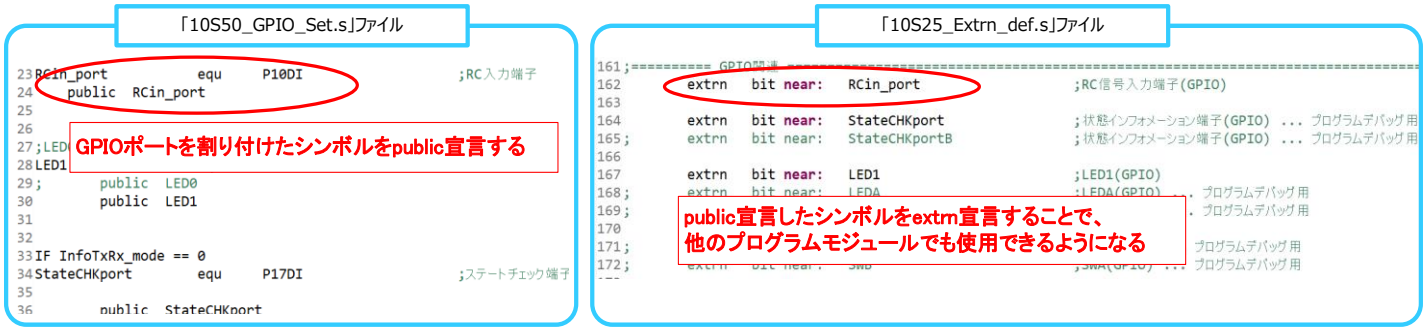

Figure 4-4. GPIO ポートの設定とシンボルの割付け、public/extrn 宣言

(4)「10S60_ADconv_TEMPsens_PGA_Set.s」ファイル(A/D コンバータ、温度センサ、PGA の設定)

A/D コンバータ、PGA、温度センサの設定を行います(通常、温度センサの設定は変更する必要はありません)。また、各ポートへのシンボル割付けを行います。RMOS では、バックグラウンドで A/D コンバータの変換動作、および、自動取得を行っています。通常動作時は最大 4ch、ローパワー動作モード時は最大 2ch の自動取得を行います。本ファイルで自動取得対象とする AD チャンネルの設定を行います。

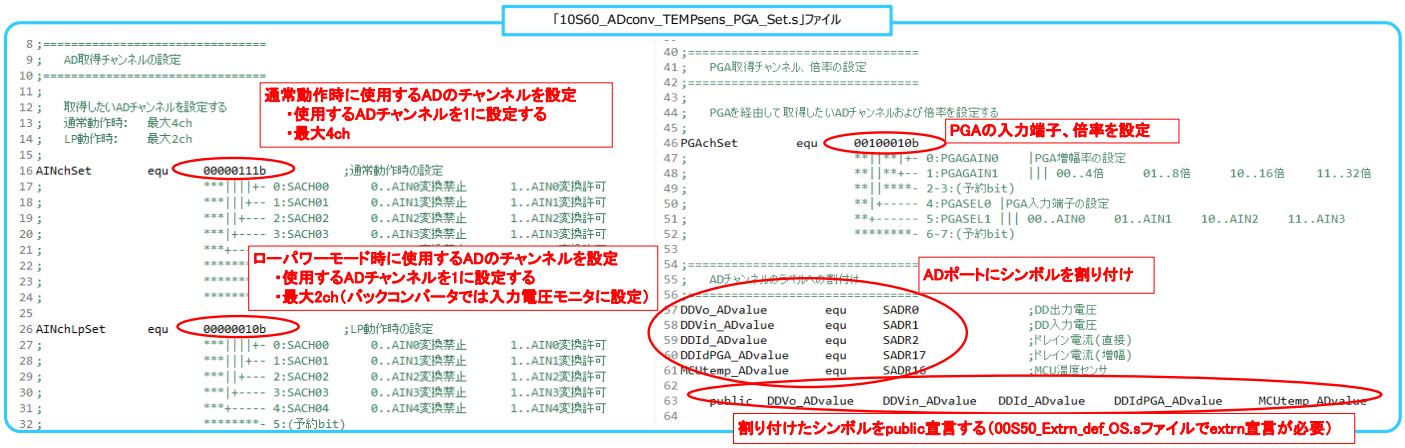

Figure 4-5. A/D コンバータ、PGA、温度センサ設定

4-4. 「30_Info_module」フォルダ(通信制御用モジュール)

通信の制御を行っているプログラムモジュールが格納されています。通信コマンドの追加/変更を行う際に、本ファイルを編集します。通信の使用方法の詳細に関しては、通信機能および GUI 作成の解説アプリケーションノート[5]を参照してください。

Table 4-5. 「30_Info_module」フォルダ内のファイル

No.	ファイル名	内容
1	30I01_InfoCMD_Exec.asm	通信コマンドの実行プログラムを記述
2	30I11_InfoCMD_Table_def.s	通信コマンドの割付け

4-5. 「50_BG_module」フォルダ(バックグラウンドモジュール群)

バックグラウンドモジュール群が格納されているフォルダです。状態遷移制御と独立で動作する必要があるプログラムをバックグラウンドモジュールに記述します。バックグラウンドモジュールには、予めプログラム記述エリアが設けられており、ここにプログラムを記述することで、既定の実行周期でプログラムが実行されます。状態変数、および、状態フラグを出力するプログラム等も本モジュール内に記述します。

Table 4-6. 「50_BG_module」フォルダ内のファイル

No.	ファイル名	実行周期	No.	ファイル名	実行周期
1	50B002_BG25u_Task.asm	25µs	14	50Bb03_BG100m_Task3.asm	100ms
2	50B005_BG50u_Task.asm	50µs	15	50Bb04_BG100m_Task4.asm	↑
3	50B010_BG100u_Task.asm	100µs	16	50Bc00_BG500m_Task0.asm	100ms
4	50B050_BG500u_Task.asm	500µs	17	50Bc01_BG500m_Task1.asm	↑
5	50B100_BG1m_Task0.asm	1ms	18	50Bc02_BG500m_Task2.asm	↑
6	50B101_BG1m_Task1.asm	↑	19	50Bc03_BG500m_Task3.asm	↑
7	50Ba00_BG10m_Task0.asm	10ms	20	50Bc04_BG500m_Task4.asm	↑
8	50Ba01_BG10m_Task1.asm	↑	21	50Bd00_BG1000m_Task0.asm	1000ms
9	50Ba20_BG25m_Task0.asm	25ms	22	50Bd01_BG1000m_Task1.asm	↑
10	50Ba21_BG25m_Task1.asm	↑	23	50Bd02_BG1000m_Task2.asm	↑
11	50Bb00_BG100m_Task0.asm	100ms	24	50Bd03_BG1000m_Task3.asm	↑
12	50Bb01_BG100m_Task1.asm	↑	25	50Bd04_BG1000m_Task4.asm	↑
13	50Bb02_BG100m_Task2.asm	↑			

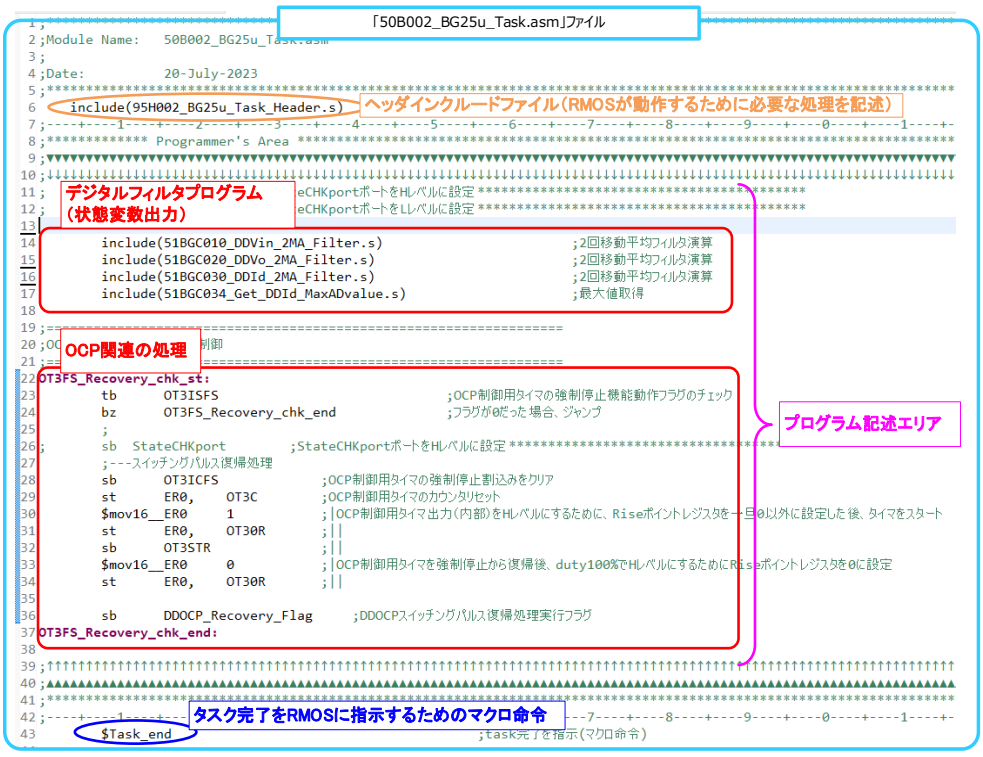

Figure 4-6. バックグラウンドモジュールの記述

4-6.「51_BG_Control_include」フォルダ(バックグラウンドモジュール用インクルードファイル)

バックグラウンドモジュール群で使用するインクルードファイル(拡張子「.s」のファイル)を格納するフォルダです。現在は、バックコンバータの制御に使用するための、状態変数、および、状態フラグを出力させるためにインクルードファイルを格納しています。

Table 4-7.「51_BG_Control_include」フォルダ内のファイル

No.	ファイル名	内容
1	51BGC010_DDVin_2MA_Filter.s	入力電圧 AD 値の 2 回移動平均値(状態変数)を出力
2	51BGC020_DDVo_2MA_Filter.s	出力電圧 AD 値の 2 回移動平均値(状態変数)を出力
3	51BGC030_DDId_2MA_Filter.s	ドレイン電流 AD 値の 2 回移動平均値(状態変数)を出力
4	51BGC034_Get_DDId_MaxADvalue.s	ドレイン電流 AD 値の最大値(状態変数)を出力
5	51BGC035_DDIdPGA_2MA_Filter.s	ドレイン電流 PGA 値の 2 回移動平均値(状態変数)を出力
6	51BGC036_DDIdPGA_8MA_Filter.s	ドレイン電流 PGA 値の 8 回移動平均値(状態変数)を出力
7	51BGC210_DDVin_RiseFall_Chk.s	入力電圧を監視し、起動許可(状態フラグ)、停止許可(状態フラグ)を出力
8	51BGC900_RC_Chk.s	RC 端子を監視し、起動許可(状態フラグ)、停止許可(状態フラグ)を出力

4-7.「55_BGLp_module」フォルダ(ローパワー動作モード時用バックグラウンドモジュール)

ローパワー動作モード時に動作するバックグラウンドモジュール群が格納されているフォルダです。ローパワー動作モード時は、状態遷移制御モジュール群の動作が停止し、Table 4-6 の No.7 以降のバックグラウンドモジュール(実行周期 10ms 以上)と、Table 4-8 のローパワー動作モード時用のバックグラウンドモジュールが動作します。

Table 4-8.「55_BGLp_module」フォルダ内のファイル

No.	ファイル名	実行周期	No.	ファイル名	実行周期
1	55B005_BGLp50u_Task.asm	50μs	2	55B010_BGLp100u_Task.asm	100μs

4-8.「70_PS0_State_Control_module」フォルダ(状態遷移制御モジュール群 0)

状態遷移制御モジュール群 0 が格納されているフォルダです。状態遷移制御モジュールにも予めプログラム記述エリアが設けられています。状態遷移制御では、Table 4-9 のプログラムモジュールの何れか 1 つ(定常動作のみ、2 つ同時)が実行されます。実行モジュールを遷移させる際には、RMOS に状態遷移を指示するマクロ命令を使用します。

電源制御プログラムの記述性・可読性を高めるために各動作状態を複数に分けています。以下では、バックコンバータにおける各状態遷移モジュールの制御内容を示します。

Table 4-9.「70_PS0_State_Control_module」フォルダ内のファイル

No.	ファイル名	動作状態	バックコンバータにおける制御例
1	71P010_PS0_Standby_0.asm	待機状態	処理無し(待機動作は 4 状態で記述できるため本モジュールは空とした)
2	71P011_PS0_Standby_1.asm		入力電圧起動許可チェック
3	71P012_PS0_Standby_2.asm		起動遅延時間カウント
4	71P013_PS0_Standby_3.asm		RC 起動許可チェック・ソフトスタート演算 1
5	71P014_PS0_Standby_4.asm		ソフトスタート演算 2
6	72P020_PS0_Startup_0.asm	起動状態	ソフトスタート動作 0（クランプ回路電圧を三角波下限まで上昇）
7	72P021_PS0_Startup_1.asm		ソフトスタート動作 1（出力電圧を設定値の 50%まで上昇）
8	72P022_PS0_Startup_2.asm		ソフトスタート動作 2（出力電圧を設定値の 75%まで上昇）
9	72P023_PS0_Startup_3.asm		ソフトスタート動作 3（出力電圧を設定値の 93%まで上昇）
10	72P024_PS0_Startup_4.asm		ソフトスタート動作 4（クランプ回路とフィードバック制御回路の引渡）
11	73P030_PS0_Normal_50u.asm	定常状態	各種停止チェック、OVP チェック、OCP 時スイッチングパルス制限
12	73P035_PS0_Normal_500u.asm		OCP 復帰時スイッチングパルス制限解除、LVP チェック
13	74P040_PS0_Stop_0.asm	停止状態	スイッチング素子類停止、ペリフェラル停止
14	74P041_PS0_Stop_1.asm		処理無し(停止動作は 4 状態で記述できるため本モジュールは空とした)
15	74P042_PS0_Stop_2.asm		各種設定値初期化(次回起動用)
16	74P043_PS0_Stop_3.asm		正常停止チェック
17	74P044_PS0_Stop_4.asm		ラッチ停止処理

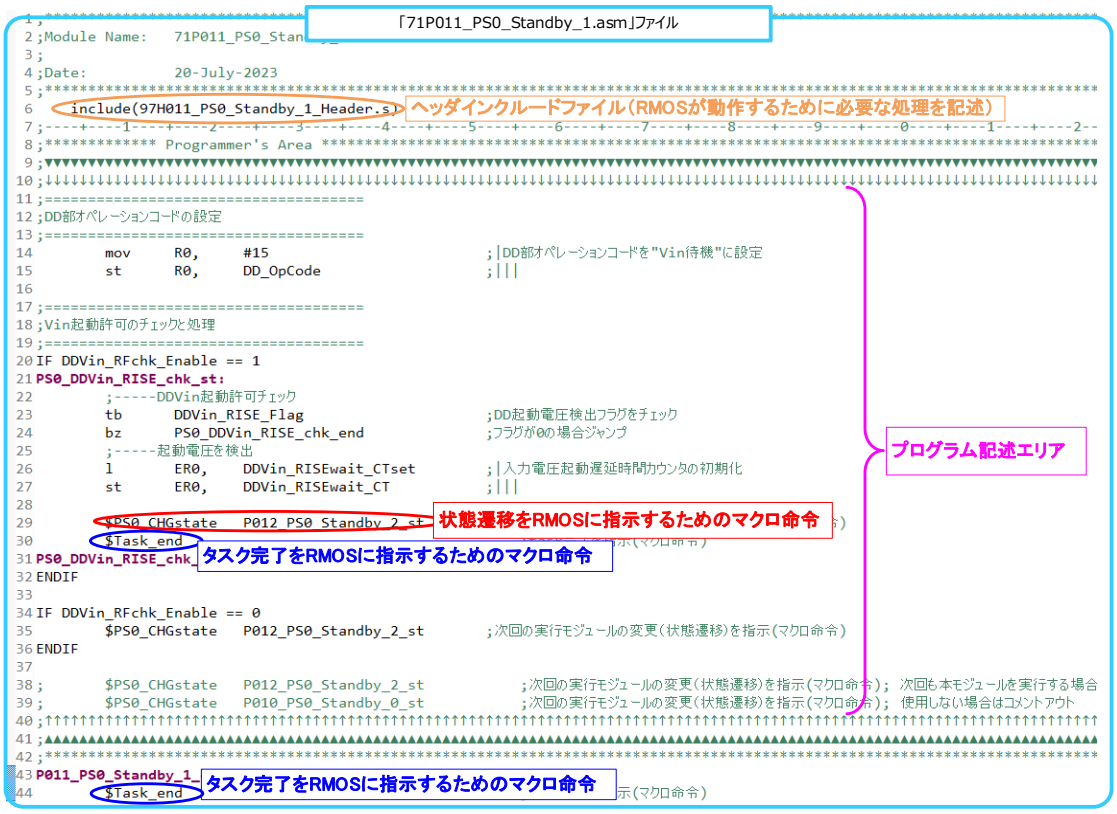

Figure 4-7. 状態遷移制御モジュールの記述

4-9. 「71_PS0_Control_include」フォルダ(状態遷移制御モジュール群 0 用インクルードファイル)

状態遷移制御モジュール群 0 で使用するインクルードファイル(拡張子「.s」のファイル)を格納するフォルダです。バックコンバータの状態遷移制御では、インクルードファイルは使用していません。

4-10. 「80_PS1_State_Control_module」フォルダ、「81_PS0_Control_include」フォルダ(状態遷移制御モジュール群 1)

状態遷移制御モジュール群 1 が格納されているフォルダ、および、状態遷移制御モジュール群 1 で使用するインクルードファイル(拡張子「.s」のファイル)を格納するフォルダです。

状態遷移制御モジュール群 0 と 1 を使用することで、2 つのコンバータを独立して制御を行うことができます。バックコンバータ EVK では、1 つのコンバータのみの制御ですので、状態遷移制御モジュール群 1 は使用していません。

4-11. 「90_Header」フォルダ(ヘッダーファイル群)

バックグラウンドモジュール、状態遷移制御モジュール等の先頭に配置するヘッダーファイル（インクルードファイル）を集めたフォルダです。ヘッダーファイルでは、RMOS が動作するために必要な処理を行っています。通常は本フォルダ内のファイルを編集する必要はありません。

5. RMOS を使用したプログラミング

5-1. RMOS で占有されるマイコンのリソース

RMOS では、下記のリソースを使用しています。プログラムを作成する際には、下記のリソースを使用しないで下さい。

Table 5-1. RMOS が使用しているマイコンのリソース

No.	リソース名	備考
1	ER10 レジスタ	状態遷移制御モジュール 0 群で実行されるプログラムモジュールの管理に使用
2	ER12 レジスタ	状態遷移制御モジュール 1 群で実行されるプログラムモジュールの管理に使用
3	ER14 レジスタ	マルチタスク動作において実行されるプログラムモジュールの管理に使用
4	16bit タイマ	マルチタスク動作の制御周期管理に使用
5	割込	マルチタスク動作に影響を与えるため、プログラマでの使用を禁止
6	ウオッチドッグタイマ	50ms 毎にリセット

5-2. RMOS でバックグラウンド処理される機能

RMOS では、下記の機能をバックグラウンドで処理しています。

(1)A/D コンバータ値取得

通常動作時は、最大 4ch までの A/D コンバータ値を自動取得しています（取得周期：25μs）。取得するチャンネルは、「10S60_ADconv_TEMPsens_PGA_Set.s」ファイルで設定します。

ローパワー動作モード時は、最大 2ch の A/D コンバータ値を自動取得しています（取得周期：50μs）。取得するチャンネルは、「10S60_ADconv_TEMPsens_PGA_Set.s」ファイルで設定します。

A/D コンバータ取得結果が記録されている SFR に状態変数を割り付けることで、2 回移動平均値を計算するデジタルフィルタ等を使用することができます。

(2)PGA 取得

通常動作時は、PGA 値を自動取得しています（取得周期：50μs）。端子の割付けは「10S60_ADconv_TEMPsens_PGA_Set.s」ファイルで行います。

ローパワー動作モード時は、PGA 値の取得を停止します。

(3)温度センサ値取得

通常動作時、ローパワー動作モード時共に、温度センサ値を自動取得しています（取得周期：50ms）。

(4)通信

ハードウエアとして UART を使用し、ソフトウエア処理を行うことで、1 台の PC に対して、最大 32 台の電源と通信を行うことが可能な通信プロトコルを実装しています。通信機能を使用するには、RMOS が準備している状態変数を用います。通信の詳細は、通信機能および GUI 作成の解説アプリケーションノート[5]を参照してください。

(5)電源動作ログ・電源設定パラメータ記録　　※現在、未実装

マイコン内のデータフラッシュメモリに、電源の動作ログ、電源設定パラメータ(起動電圧、停止電圧、起動遅延時間、過電流設定値、etc)を記録する機能を実装する予定です。

5-3. 状態変数・状態フラグ

RMOS を使用して電源制御プログラムを作成する際には、RMOS が出力する「状態変数」、および、「状態フラグ」を使用してプログラミングを行います。RMOS では、バックグラウンドモジュールに「状態変数」、および、「状態フラグ」を処理するプログラムが記述されています。「状態変数」、および、「状態フラグ」の処理プログラムは、プログラマでも作成することができます。

以下では、バックコンバータの制御プログラムにおいて、状態フラグを使用した電源の起動/停止判定プログラム例を示します。

(1) 起動/停止状態フラグ

起動/停止状態フラグは、「51BGC210_DDVin_RiseFall_Chk.s」モジュールで処理しています。本モジュールは、実行周期 50µs のバックグラウンドモジュールに配置されていますので、50µs 毎に入力電圧を監視します。本モジュールは、例えば、実行周期 100µs のバックグラウンドモジュールに配置すると、入力電圧の監視周期を 100µs に変更することができます。

本モジュールの動作を下記に示します。

①入力電圧 2 回移動平均(DDVin_2MAvalue;入力電圧 AD 値を基に出力される状態変数)が起動判定電圧(DDVin_RISEsetinit に定義された値)以上になったことを規定回数(DDVin_RISEchk_CTinit に設定された値)だけ検出すると、DDVin_RISE_Flag=1 を出力します。

②入力電圧 2 回移動平均値(DDVin_2MAvalue)が停止判定電圧(DDVin_FALLsetinit に定義された値)以下になったことを規定回数(DDVin_FALLchk_CTinit に設定された値)だけ検出すると、DDVin_FALL_Flag=1 を出力します。

「50B005_BG50u_Task.asm」ファイル

```
14
15        include(51BGC900_RC_Chk.s)                    ;RC端子チェック
16
17        include(51BGC210_DDVin_RiseFall_Chk.s)        ;DDVin起動停止チェック
18
```

Figure 5-1. 起動/停止状態フラグを出力するプログラムモジュールの実行周期

「10S01_Parameter_Init.s」ファイル

```
41 ;========== 入力電圧起動停止処理 ==========================================================================
42 DDVin_RISEsetinit          equ 12892          ;起動電圧初期値   12892 = 9Vdc
43 DDVin_FALLsetinit          equ 11459          ;停止電圧初期値   11459 = 8Vdc
44
45
46 DDVin_RISEchk_CTinit        equ 3              ;起動電圧チェック時のノイズ除去カウンタの初期値
47 DDVin_FALLchk_CTinit        equ 3              ;停止電圧チェック時のノイズ除去カウンタの初期値
```

Figure 5-2. 起動/停止電圧等設定

(2) 電源起動判定プログラム

電源の起動判定は、「DDVin_RISE_Flag」の状態を判定して行います。DDVin_RISE_Flag=1 の場合は RMOS に状態遷移の指示を出すようにプログラミングします。

「71P011_PS0_Standby_1.asm」ファイル

```
17 ;=====================================
18 ;Vin起動許可のチェックと処理
19 ;=====================================
20 PS0_DDVin_RISE_chk_st:
21         ;-----DDVin起動許可チェック
22         tb      DDVin_RISE_Flag                    ;DDVin_RISE_Flagをチェック
23         bz      PS0_DDVin_RISE_chk_end             ;フラグが0の場合ジャンプ
24         ;-----起動電圧を検出
25         l       ER0,    DDVin_RISEwait_CTset       ;|入力電圧起動遅延時間カウンタの初期化
26         st      ER0,    DDVin_RISEwait_CT          ;|||
27
28         $PS0_CHGstate   P012_PS0_Standby_2_st      ;次回の実行モジュールの変更(状態遷移)を指示(マクロ命令)
29         $Task_end                                  ;task完了を指示(マクロ命令)
30 PS0_DDVin_RISE_chk_end:
```

DDVin_RISE_Flagをチェック

DDVin_RISE_Flag=1の場合は、RMOSに状態遷移を指示

Figure 5-3. 電源起動判定プログラム

(3) 電源停止判定プログラム

電源の停止判定は、「DDVin_FALL_Flag」の状態を判定して行います。DDVin_FALL_Flag=1 の場合は RMOS に状態遷移の指示を出すようにプログラミングします。

「73P030_PS0_Normal_50u.asm」ファイル

```
60 ;=====================================
61 ;Vin停止許可のチェックと処理
62 ;=====================================
63 PS0_DDVin_FALL_chk_st:
64         ;-----DDVin停止許可チェック
65         tb      DDVin_FALL_Flag                 ;フラグをチェック
66         bz      PS0_DDVin_FALL_chk_end          ;フラグが0の場合ジャンプ
67         ;-----停止電圧を検出
68         mov     R0,     #15                     ;|DD部オペレーションコードを"Vin待機"に設定
69         st      R0,     DD_OpCode               ;|||
70         ;
71         $PS0_CHGstate   P040_PS0_Stop_0_st      ;次回の実行モジュールの変更(状態遷移)を指示(マクロ命令)
72         $Task_end
73 PS0_DDVin_FALL_chk_end:
```

DDVin_FALL_Flagをチェック

DDVin_FALL_Flag=1の場合は、RMOSに状態遷移を指示

Figure 5-4. 電源停止判定プログラム

5-4. ローパワー動作モード

通常動作モードからローパワー動作モードへの切替を行う場合には、状態変数「LpMode_Enter_Flag」を使用します。バックコンバータの制御プログラムでは、RC 端子を使用して電源を停止させた場合にローパワー動作モードに移行させています。

ローパワー動作モードでは、状態遷移制御モジュール群 0、および、状態遷移制御モジュール群 1 が停止します。また、Table 4-6 の No.1-No6 のバックグラウンドモジュールが停止し、Table 4-8 のローパワー動作モード時用バックグラウンドモジュールが動作します。

「71P013_PS0_Standby_3.asm」ファイル

```
27 ;=========================
28 ;RC起動のチェックと処理
29 ;=====================================
30 PS0_RC_chk_st:
31         ;-----RC起動許可チェック
32         tb      RC_ON_Flag                      ;RCオンフラグをチェック
33         bnz     PS0_RC_ON_detect                ;フラグが1だった場合、ジャンプ
34         ;-----RC停止を検出
35         sb      LpMode_Enter_Flag
36         b       PS0_RC_chk_end
37 PS0_RC_ON_detect:
```

LpMode_Enter_Flag=1とすることで、通常動作モードから
ローパワー動作モードへ移行する

Figure 5-5. ローパワー動作モードへの移行

ローパワー動作モードから通常動作モードへの切替を行う場合には、「LpMode_Exit_Flag」を使用します。バックコンバータの制御プログラムでは、RC 端子を使用して電源停止の解除を行った時に通常動作モードに移行させています。

通常動作モードへ移行した際には、状態遷移制御モジュールは、ローパワー動作モードへ移行する直前に実行されていたモジュールが動作します。

「55B005_BGLp50u_Task.asm」ファイル

```
37 ;=====================================
38 ;RC起動許可チェック
39 ;=====================================
40 RC_chk_st:
41         ;-----RCリセット処理有効化のチェック
42         tb      LpMode_Use_RcReset_Flag         ;RCリセット処理の使用フラグをチェックし、
43         bnz     RC_chk_end                      ;フラグが1の場合は、RC起動許可を無効化するために、ジャンプ
44         ;-----RC起動許可チェック
45         tb      RC_ON_Flag                      ;RCオンフラグをチェック
46         bz      RC_chk_end                      ;フラグが0の場合ジャンプ
47         ;-----RC起動を検出
48         sb      LpMode_Exit_Flag
49 RC_chk_end:
```

LpMode_Exit_Flag=1とすることで、ローパワー動作モードから
通常動作モードへ移行する

Figure 5-6. 通常動作モードへの移行

5-5. デバッグ作業(CPU のブレーク、ステップ実行、変数の確認)

RMOS では、CPU をブレークした場合でも、タイマ、コンパレータ、および、D/A コンバータの動作が継続するように設定されています。アナログ・デジタル融合制御のスイッチング電源は、通常は、CPU を介さずにスイッチング動作を行うように回路設計を行いますので、CPU が停止しても電源回路は動作を継続します。従いまして、電源回路のスイッチング動作を継続させながら、ステップ実行を使用してプログラムの動作確認を行うことができます。

(1)ブレークポイントの設定

Figure 5-7 では、「73P030_PS0_Normal_50u.asm」にブレークポイントを設定する場合を示します。ブレークポイントは、「拡張子.asm」ファイルには設定できますが、「拡張子.s」ファイルには設定することができません(LEXIDE-Ωの仕様)。

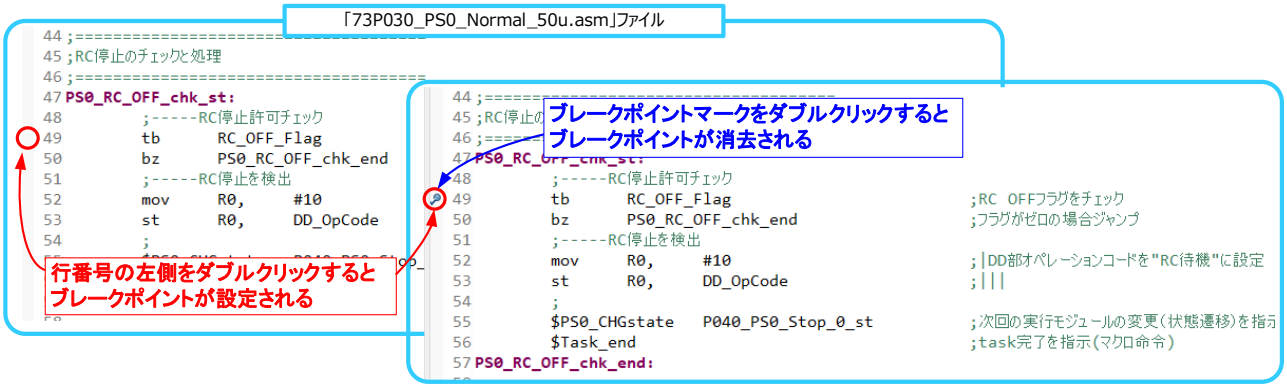

Figure 5-7. ブレークポイントの設定

ブレークポイントを設定した状態でプログラムを実行すると、プログラムがブレークポイントに到達した時にプログラムがブレークします。例えば、「73P030_PS0_Normal_50u.asm」にブレークポイントを設定すると、バックコンバータが定常動作(5V が出力)に至ったところでプログラムがブレークします。プログラムはブレークしますが、スイッチング動作は継続していますので電源の出力電圧は 5V が出力された状態を維持します。

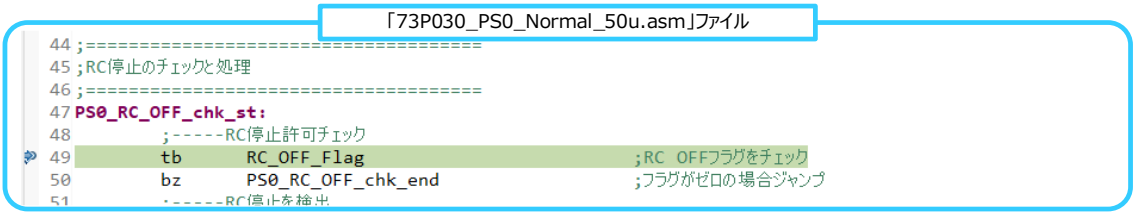

Figure 5-8. ブレーク時の画面

プログラムがブレークすると、画面右下の「マイコン内部情報表示エリア」の「Expression ウインドウ」には、変数、レジスタおよび、SFR 等の値が表示されます。「Expression ウインドウ」には、「00S20_RAM_def_OS.s」で設定した変数、および、フラグも表示させることができます。Figure 5-9 では、RMOS が出力する状態変数、状態フラグを表示させた例を示します。

「Expression ウインドウ」に表示させる変数やフラグを変更する場合には、「Expression」欄の名称を変更します。この際、「00S20_RAM_def_OS.s」で 1 バイト変数(DS 1 命令で確保した変数)は、変数名の後ろに、「表示バイト数指定」として、「#1」を付します。また、2 バイト変数(DS 2 命令で確保した変数)は、変数名の後ろに「#2」を付します。

フラグを表示させる場合には、表示バイト数指定は不要です。

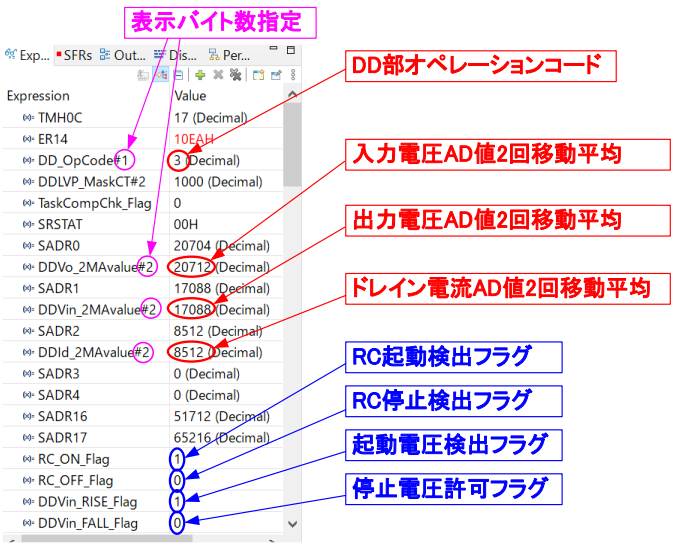

Figure 5-9. マイコン内部情報表示エリアエリアの表示

ブレークされたプログラムは、ステップ実行を行うことで、プログラムを 1 行ずつ進めることができます。また、プログラムを再開すると、次回の実行周期までプログラムが進みます。「73P030_PS0_Normal_50u.asm」にブレークポイントを設定した場合にプログラムを再開すると、マイコンとしては、50μs(実行周期)後にブレークされることになります。

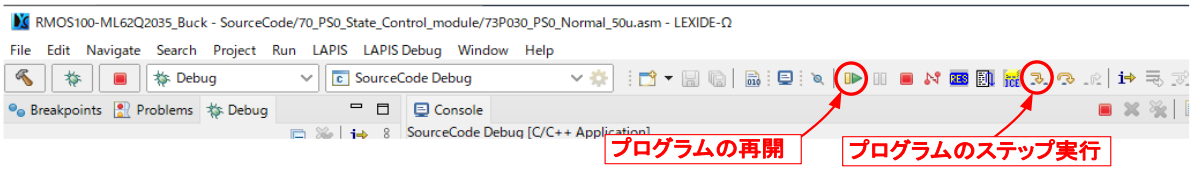

Figure 5-10. プログラムのステップ実行/再開ボタン

5-6. タスク割当許容時間（許容カウント値）の確認

RMOS のマルチタスク処理では、プログラムモジュールの割当時間内にプログラムの動作が完了するようにプログラミングする必要が有ります。もし、割当時間内にプログラムの動作が完了しなかった場合、割当時間に到達したところでプログラムの実行が停止され、次のプログラムモジュールに制御が移されます）。

タスクの割当時間の管理は 16bit タイマで行っていますので、16bit タイマのカウント値で確認することができます。16bit タイマのカウント値の確認は、Figure 5-11 の手順で行います。各プログラムモジュールにおいて、プログラムの終了箇所における 16bit タイマのカウント値が下表の許容カウント値内に収まっていることを確認します。

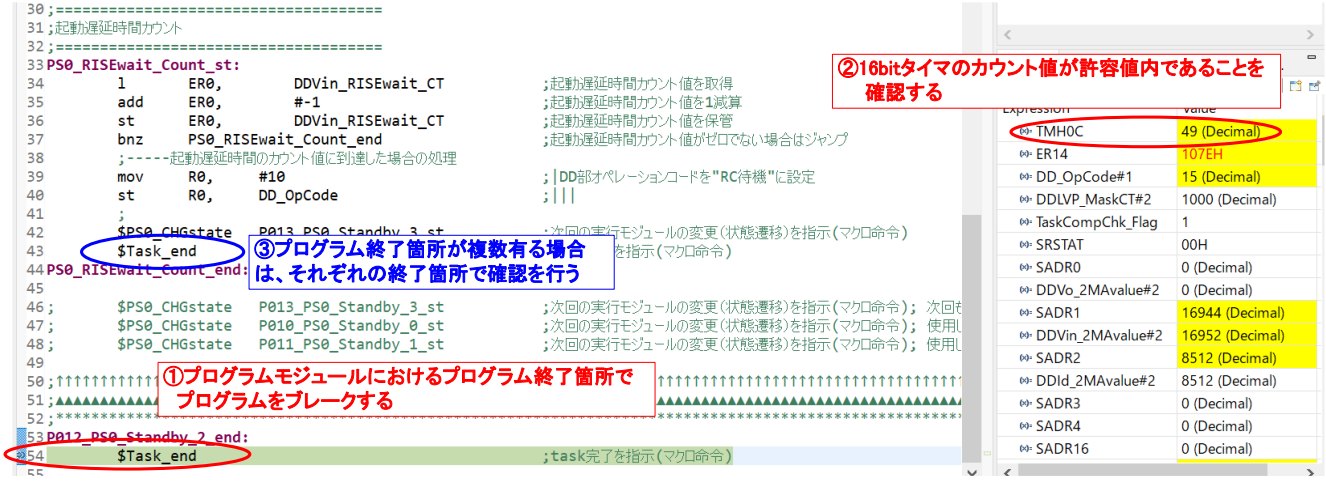

Figure 5-11. 16bit タイマカウント値によるタスク割当時間の確認

Table 5-2. バックグラウンドモジュール群のタスク割当許容カウント値(通常動作モード)

No.	ファイル名	許容カウント値	No.	ファイル名	許容カウント値
1	50B002_BG25u_Task.asm	114	14	50Bb03_BG100m_Task3.asm	140
2	50B005_BG50u_Task.asm	124	15	50Bb04_BG100m_Task4.asm	↑
3	50B010_BG100u_Task.asm	144	16	50Bc00_BG500m_Task0.asm	↑
4	50B050_BG500u_Task.asm	↑	17	50Bc01_BG500m_Task1.asm	↑
5	50B100_BG1m_Task0.asm	↑	18	50Bc02_BG500m_Task2.asm	↑
6	50B101_BG1m_Task1.asm	↑	19	50Bc03_BG500m_Task3.asm	↑
7	50Ba00_BG10m_Task0.asm	140	20	50Bc04_BG500m_Task4.asm	↑
8	50Ba01_BG10m_Task1.asm	↑	21	50Bd00_BG1000m_Task0.asm	↑
9	50Ba20_BG25m_Task0.asm	↑	22	50Bd01_BG1000m_Task1.asm	↑
10	50Ba21_BG25m_Task1.asm	↑	23	50Bd02_BG1000m_Task2.asm	↑
11	50Bb00_BG100m_Task0.asm	↑	24	50Bd03_BG1000m_Task3.asm	↑
12	50Bb01_BG100m_Task1.asm	↑	25	50Bd04_BG1000m_Task4.asm	↑
13	50Bb02_BG100m_Task2.asm	↑			

Table 5-3. 状態遷移制御モジュール群 0 のタスク割当許容カウント値(通常動作モード)

No.	ファイル名	許容カウント値	No.	ファイル名	許容カウント値
1	71P010_PS0_Standby_0.asm	146	11	73P030_PS0_Normal_50u.asm	146
2	71P011_PS0_Standby_1.asm	↑	12	73P035_PS0_Normal_500u.asm	144
3	71P012_PS0_Standby_2.asm	↑	13	74P040_PS0_Stop_0.asm	146
4	71P013_PS0_Standby_3.asm	↑	14	74P041_PS0_Stop_1.asm	↑
5	71P014_PS0_Standby_4.asm	↑	15	74P042_PS0_Stop_2.asm	↑
6	72P020_PS0_Startup_0.asm	↑	16	74P043_PS0_Stop_3.asm	↑
7	72P021_PS0_Startup_1.asm	↑	17	74P044_PS0_Stop_4.asm	↑
8	72P022_PS0_Startup_2.asm	↑			
9	72P023_PS0_Startup_3.asm	↑			
10	72P024_PS0_Startup_4.asm	↑			

Table 5-4. 状態遷移制御モジュール群 1 のタスク割当許容カウント値(通常動作モード)

No.	ファイル名	許容カウント値	No.	ファイル名	許容カウント値
1	81P110_PS1_Standby_0.asm	122	11	83P130_PS1_Normal_50u.asm	122
2	81P111_PS1_Standby_1.asm	↑	12	83P135_PS1_Normal_500u.asm	144
3	81P112_PS1_Standby_2.asm	↑	13	84P140_PS1_Stop_0.asm	122
4	81P113_PS1_Standby_3.asm	↑	14	84P141_PS1_Stop_1.asm	↑
5	81P114_PS1_Standby_4.asm	↑	15	84P142_PS1_Stop_2.asm	↑
6	82P120_PS1_Startup_0.asm	↑	16	84P143_PS1_Stop_3.asm	↑
7	82P121_PS1_Startup_1.asm	↑	17	84P144_PS1_Stop_4.asm	↑
8	82P122_PS1_Startup_2.asm	↑			
9	82P123_PS1_Startup_3.asm	↑			
10	82P124_PS1_Startup_4.asm	↑			

Table 5-5. ローパワー動作モード時用バックグラウンドモジュール群のタスク割当許容カウント値

No.	ファイル名	許容カウント値	No.	ファイル名	許容カウント値
1	55B005_BGLp50u_Task.asm	315	2	55B010_BGLp100u_Task.asm	315

5-7. RMOS によるタスク未完了の検出

RMOS には、プログラムモジュールの割当時間内にプログラムの動作が完了しなかったこと(以下、タスク未完了)をチェックするための状態フラグ「TaskCompChk_Flag」を持っています。マルチタスク動作において、「タスクセレクタ」に制御が移された時に TaskCompChk_Flag=1 の状態になっていた場合には、タスク未完了であることを示します。NOP 命令にブレークポイントを設けることで、タスク未完了を検出することができます。

・通常動作モード： 「00S92_TaskSelector.asm」の 96 行目にブレークポイントを設ける
・ローパワー動作モード： 「00S92_TaskSelector.asm」の 67 行目にブレークポイントを設ける

Figure 5-12 は、通常動作モードにおいてタスク未完了のチェックを行うために「00S92_TaskSelector.asm」の 96 行目にブレークポイントを設けた場合を示します。タスク未完了が検出されると、「00S92_TaskSelector.asm」の 96 行目でプログラムが停止します。

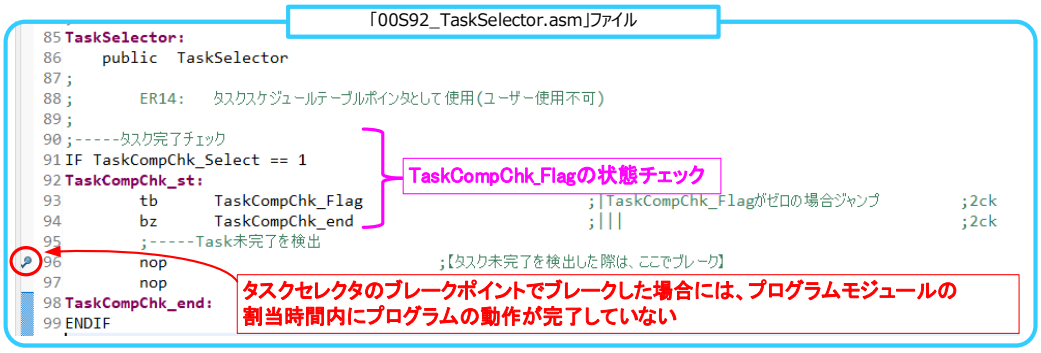

Figure 5-12. タスク未完了の検出

6. 状態変数・状態フラグ一覧

現在の RMOS がバックグラウンドで処理を行っている状態変数、状態フラグの一覧を下記に示します。なお、下記の一覧のパラメータが設定された RMOS のバージョンは「RMOSVer=1.00、OSBuildNo=007、PSFMNo=001、PSFMVer=1.00、PSFMBuildNo=004」（4-3 章(1)に記載のバージョンと同様）になります。

Table 6-1. RMOS の状態変数・状態フラグ一覧（10S20_RAM_def.s ファイルで定義）

(1)起動停止

No.	変数名・フラグ名	Byte数	機能	初期設定用シンボル名	初期値	備考
1	DDVin_RISEset	2	上記の起動開始電圧設定値	DDVin_RISEsetinit	12880	モニタ周期: 50μs
2	DDVin_RISE_Flag	Flag	入力電圧によるDD部の起動許可の検出	–	–	
3	DDVin_RISEchk_CT	1	上記のノイズ除去カウンタ	DDVin_RISEchk_CTinit	3	
4	DDVin_FALLset	2	上記の停止開始電圧設定値	DDVin_FALLsetinit	11459	
5	DDVin_FALL_Flag	Flag	入力電圧によるDD部の停止許可の検出	–	–	
6	DDVin_FALLchk_CT	1	上記のノイズ除去カウンタ	DDVin_FALLchk_CTinit	3	

(2)リモートON/OFF

No	フラグ名・変数名	Byte数	機能	初期設定用シンボル名	初期値	備考
1	RC_ON_Flag	Flag	RC端子による起動許可の検出	–	–	モニタ周期: 50μs
2	RC_ONchk_CT	1	上記のノイズ除去カウンタ	RC_ONchk_CTinit	3	
3	RC_OFF_Flag	Flag	RC端子による停止許可の検出	–	–	
4	RC_OFFchk_CT	1	上記のノイズ除去カウンタ	RC_OFFchk_CTinit	25	
5	RClogic_Inv_Flag	Flag	RC端子入力ロジック反転フラグ	–	–	

(3)デジタルフィルタ

No	変数名	Byte数	機能	参照変数名	Byte数	備考
1	DDVin_2MAvalue	2	入力電圧AD値の2回移動平均	DDVin_ADvalue	2	計算周期：25μs
2	DDVo_2MAvalue	2	出力電圧AD値の2回移動平均	DDVo_ADvalue	2	計算周期：25μs
3	DDIdPGA_2MAvalue	2	ドレイン電流PGA値の2回移動平均	DDIdPGA_ADvalue	2	計算周期：50μs
4	DDIdPGA_8MAvalue	2	ドレイン電流PGA値の8回移動平均	DDIdPGA_ADvalue	2	計算周期：50μs
5	DDId_2MAvalue	2	ドレイン電流AD値の2回移動平均	DDId_ADvalue	2	計算周期：25μs
6	DDId_MaxADvalue	2	ドレイン電流AD値の最大値	DDId_ADvalue	2	取得周期：25μs

(4)通信

No	変数名	Byte数	機能	初期設定用シンボル名	初期値	備考
1	PS_ADR	1	通信用アドレス設定	PS_ADR_init	31	
2	RXD_CmdGr	1	通信により受信したコマンドグループ	–	–	
3	RXD_CmdNo	1	通信により受信したコマンドナンバー	–	–	
4	RXD_Data16	2	通信により受信した16bitデータ	–	–	

(5)ローパワー動作モードの制御

No	変数名	Byte数	機能	備考
1	LpMode_Enter_Flag	Flag	ローパワー動作モードに移行する	電源の異常停止した場合にラッチ停止処理を行う場合には、ローパワー動作モードへ以降させる前に、LpMode_Use_RcReset_Flag=1とすることで、RCリセット処理を有効化できる
2	LpMode_Exit_Flag	Flag	通常動作モードに移行する	
3	LpMode_Use_RcReset_Flag	Flag	RCリセット処理の有効化	

(6)システム関連

No	変数名	Byte数	機能	備考
1	DD_OpCode	1	DCDC部の動作コードを記録	
2	DD_OpCode_FailRec	1	DCDC部の異常停止時の動作コードを記録	
3	TaskCompChk_Flag	Flag	タスク完了チェックフラグ	

(7)LED点滅

No	シンボル名	Byte数	機能	備考
1	LED1FP_VinStby	–	起動電圧以下待機時の点滅パターン	
2	LED1FP_RcStby	–	RC待機時の点滅パターン	
3	LED1FP_NomOP	–	定常動作時の点滅パターン	
4	LED1FP_FAIL	–	異常停止時の点滅パターン	

また、参考にバックコンバータの制御に使用している状態変数、状態フラグの一覧を下記に示します。

Table 6-2. バックコンバータ制御用の状態変数・状態フラグ一覧

(1)スイッチング素子・同期整流素子設定

No	変数名	Byte数	機能	初期設定用シンボル名	初期値	備考
1	Fsw_CTset	2	スイッチング周波数設定（OTM設定値）	Fsw_CTinit	399	
2	dmax_CTset	2	スイッチング素子の最大duty設定（OTM設定値）	dmax_CTinit	319	
3	DTimeHoffLon_CTset	2	スイッチング素子オフと同期整流素子オンのデッドタイム設定（OTM設定値）	DTimeHoffLon_CTinit	9	
4	DTimeLoffHon_CTset	2	同期整流素子オフとスイッチング素子オンのデッドタイム設定（OTM設定値）	DTimeLoffHon_CTinit	379	

(2)起動遅延

No.	変数名	Byte数	機能	初期設定用シンボル名	初期値	備考
1	DDVin_RISEwait_CTset	2	起動遅延時間設定	DDVin_RISEwait_CTinit	20000	カウント周期：50μs
2	DDVin_RISEwait_CT	2	起動遅延時間用カウンタ	–	–	

(3)ソフトスタート

No	変数名	Byte数	機能	初期設定用シンボル名	初期値	備考
1	SSRampRate1set	1	ソフトスタート1ランプレート設定値	SSRampRate1init	50	
2	SSRampRate2set	1	ソフトスタート2ランプレート設定値	SSRampRate2init	25	
3	SSRampRate3set	1	ソフトスタート3ランプレート設定値	SSRampRate3init	12	
4	SSTimeupCTset	2	起動時間オーバー検出時間	SSTimeupCTinit	1000	
5	SS0loopCTset	1	ソフトスタート開始前ループカウント値	SS0loopCTinit	3	
6	SS4loopCTset	1	定常動作引渡ループカウント値	SS4loopCTinit	150	

(4)出力電圧設定

No	変数名・シンボル名	Byte数	機能	初期設定用シンボル名	初期値	備考
1	DDVo_DACset	1	DCDC部出力電圧設定用DAC値	DDVo_DACinit	101	
2	DDVo_DAC_MaxLmt	–	DCDC部出力電圧設定用DAC制限値	（本変数はシンボル）	181	

(5)保護動作

No	変数名	Byte数	機能	初期設定用シンボル名	初期値	備考
1	DDOCP_Ioset	2	DCDC部過電流保護設定値	DDOCP_Ioinit	6000	
2	DDLVP_VoADset	2	DCDC部不足電圧保護設定値	DDLVP_VoADinit	12528	
3	DDLVP_MaskCTset	2	上記のマスク時間(ノイズ除去)設定	DDLVP_MaskCTinit	1000	
4	DDOVP_VoADset	2	DCDC部過電圧保護設定値	DDOVP_VoADinit	25056	
5	DDOVP_MaskCTset	1	上記のマスク時間(ノイズ除去)設定	DDOVP_MaskCTinit	5	
6	DDIVP_VinADset	2	DCDC部入力過電圧保護設定値	DDIVP_VinADinit	54432	
7	DDIVP_MaskCTset	1	上記のマスク時間(ノイズ除去)設定	DDIVP_MaskCTinit	5	

7. 参考ドキュメント

[1] ReleaseNote_LEXIDE_V1_1_1_j、LAPIS Development Tools LEXIDE-Ω V1.1.1 リリースノート

[2] FJXTLEXIDE_OMEGA_UM-02、LEXIDE-Ω ユーザーズマニュアル

[3] 66UG089J、Rev.001、同期整流 降圧 DCDC コンバータ 評価ボード LogiCoA001-EVK-001

[4] FJUL-U16-100-INST-03、nX-U16/100 コア インストラクションマニュアル

[5] 66AN148J、Rev.001、RMOS 搭載通信機能および GUI 作成環境解説書

改訂履歴

Date	Revision Number	Description
2024. 4. 15	001	新規作成

ご 注 意

1) 本資料に記載されている内容は、ロームグループ(以下「ローム」という)製品のご紹介を目的としています。ローム製品のご使用にあたりましては、別途最新のデータシートもしくは仕様書を必ずご確認ください。

2) ローム製品は、一般的な電子機器(AV機器、OA機器、通信機器、家電製品、アミューズメント機器等)もしくはデータシートに明示した用途への使用を意図して設計・製造されています。したがいまして、極めて高度な信頼性が要求され、その故障や誤動作が人の生命、身体への危険もしくは損害、またはその他の重大な損害の発生に関わるような機器または装置(医療機器、輸送機器、交通機器、航空宇宙機器、原子力制御装置、燃料制御、カーアクセサリーを含む車載機器、各種安全装置等)(以下「特定用途」という)にローム製品のご使用を検討される際は事前にローム営業窓口までご相談くださいますようお願いいたします。ロームの文書による事前の承諾を得ることなく、特 定用途にローム製品を使用したことによりお客様または第三者に生じた損害等に関し、ロームは一切その責任を負いません。

3) 半導体を含む電子部品は、一定の確率で誤動作や故障が生じる場合があります。万が一、誤動作や故障が生じた場合であっても、人の生命、身体、財産への危険または損害が生じないように、お客様の責任においてフェールセーフ設計など安全対策をお願いいたします。

4) 本資料に記載された応用回路例やその定数などの情報は、ローム製品の標準的な動作や使い方を説明するためのもので、実際に使用する機器での動作を明示的にも黙示的にも保証するものではありません。したがいまして、お客様の機器の設計において、回路やその定数及びこれらに関連する情報を使用する場合には、外部諸条件を考慮し、お客様の判断と責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関 し、ロームは一切その責任を負いません。

5) ローム製品及び本資料に記載の技術を輸出または国外へ提供する際には、「外国為替及び外国貿易法」、「米国輸出管理規則」など適用される輸出関連法令を遵守し、それらの定めにしたがって必要な手続きを行ってください。

6) 本資料に記載された応用回路例などの技術情報及び諸データは、あくまでも一例を示すものであり、これらに関する第三者の知的財産権及びその他の権利について権利侵害がないことを保証するものではありません。また、ロームは、本資料に記載された情報について、ロームもしくは第三者が所有または管理している知的財産権その他の権利の実施、使用または利用を、明示的にも黙示的にも、お客様に許諾するものではありません。

7) 本資料の全部または一部をロームの文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。

8) 本資料に記載の内容は、本資料発行時点のものであり、予告なく変更することがあります。ローム製品のご購入及びご使用に際しては、事前にローム営業窓口で最新の情報をご確認ください。

9) ロームは本資料に記載されている情報に誤りがないことを保証するものではありません。万が一、本資料に記載された情報の誤りによりお客様または第三者に損害が生じた場合においても、ロームは一切その責任を負いません。

ローム製品のご検討ありがとうございます。
より詳しい資料やカタログなどご用意しておりますので、お問い合わせください。

ROHM Customer Support System

https://www.rohm.co.jp/contactus