RL78/G14

7 セグメント LED 点灯制御 (Arduino API)

要旨

本アプリケーションノートでは、RL78/G14 Fast Prototyping Board (FPB) を用いて Arduino 言語のようなプログラム記述で 7 セグメント LED をダイナミック点灯制御する方法を説明します

対象デバイス

RL78/G14

本アプリケーションノートを他のマイコンへ適用する場合、そのマイコンの仕様にあわせて変更し、十分評価してください。

目次

1. 仕様

本アプリケーションノートでは、FPB を用いて Arduino 言語のようなプログラム記述で FPB に外部接続した 3 個の 7 セグメント LED (カソード・コモン) と 8 個の LED をダイナミック点灯制御します。

100 ミリ秒 (0.1 秒) 単位で表示する 1 分タイマを 7 セグメント LED で実現します。1 分経過毎に 8 個の LED を順に点灯させて 9 分までのタイマを実現します。また、FPB に搭載されているスイッチ (SW_USR) または外部接続したスイッチ (ex_SW) によって、タイマのカウント動作を制御します。

表 1.1 に本プログラムで使用する周辺機能と用途を示します。

表 1.1 使用する周辺機能と用途

周辺機能	用途
デジタル入力	スイッチ (SW_USR) または外部スイッチ (ex_SW) の状態読み込み
デジタル出力	LED (7 セグメント LED 3 個と単体 LED 8 個) の制御
タイマ・アレイ・ユニット	時間経過の計測

1.1 プログラム実行環境

本アプリケーションノートでは、RL78 ファミリ固有の開発環境上で、Arduino 言語のようなプログラムを実行させています。プログラム実行環境の概念図を図 1.1 に示します。

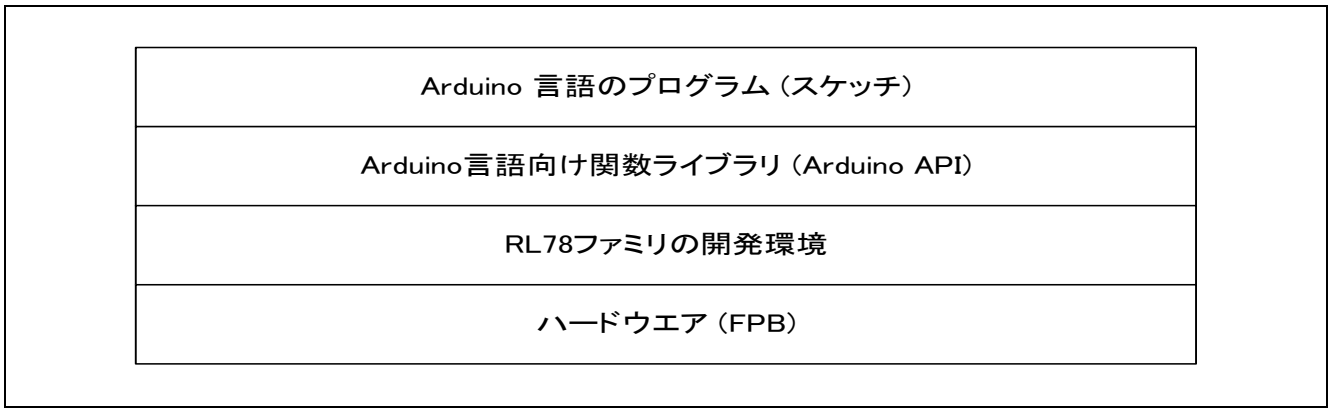

図 1.1 プログラム実行環境

本アプリケーションノートで使用可能なライブラリ関数を表 1.2 に示します。

表 1.2 ライブラリの機能

項目	ライブラリ関数	機能
デジタル入出力	pinMode(pin, mode)	pin で指定した端子の動作モード (入力モード / 出力モード / 内蔵プルアップ抵抗を有効にした入力モード) 指定
	digitalWrite (pin, value)	pin で指定した端子を value で指定した状態 (ハイレベル / ロウレベル) にする。
	digitalRead(pin)	pin で指定した端子状態を読み出す。
時間管理	millis()	プログラムの実行を開始した時から現在までの時間をミリ秒単位で返します。
	micros()	プログラムの実行を開始した時から現在までの時間をマイクロ秒単位で返します。
	delay (ms)	ミリ秒単位でプログラムを指定した時間だけ止めます。
	delayMicroseconds (us)	マイクロ秒単位でプログラムを指定した時間だけ止めます。

1.2 プログラム (スケッチ) の構成

プロジェクトが格納されているフォルダ (workspace) より下の階層には、統合開発環境ごとにサブフォルダが準備されています。それぞれの統合開発環境のフォルダには RL78 ファミリの開発環境関で使用するファイルが格納されています。

サブフォルダ sketch には、Arduino 言語のプログラム (スケッチ) である AR_SKETCH.c が格納されています。スケッチを参照もしくは変更する場合は、sketch の中の”AR_SKETCH.c”ファイルを使用します。

1.3 プロジェクトの起動準備

使用する統合開発環境ごとにプロジェクトの起動準備が異なります。詳細は、つぎのアプリケーションノートを参照してください。

RL78 ファミリ Arduino API 導入ガイド (R01AN5413)

1.4 プログラム (スケッチ) の定義

プログラム (スケッチ) の定義内容を図 1.2 に示します。

```
/*-------------------------------------------------------------------
     Definition area. Write pin definition here.
-------------------------------------------------------------------*/
1)
int segPinA = 0;                // assign D0 pin to segPinA for 7SEG_LED.
int segPinB = 1;                // assign D1 pin to segPinB for 7SEG_LED.
int segPinC = 2;                // assign D2 pin to segPinC for 7SEG_LED.
int segPinD = 3;                // assign D3 pin to segPinD for 7SEG_LED.
int segPinE = 4;                // assign D4 pin to segPinE for 7SEG_LED.
int segPinF = 5;                // assign D5 pin to segPinF for 7SEG_LED.
int segPinG = 6;                // assign D6 pin to segPinG for 7SEG_LED.
int segPinDP = 7;               // assign D7 pin to segPinDP for 7SEG_LED.

int comPin0 = 8;                // assign D8 pin to comPin0 for 7SEG_LED.
int comPin1 = 9;                // assign D9 pin to comPin1 for 7SEG_LED.
int comPin2 = 10;               // assign D10 pin to comPin2 for 7SEG_LED.
int comPin3 = 11;               // assign D11 pin to comPin3 for DOT_LEDs.

int ex_swPin = 13;              // assign D13 pin to ex_swPin for external SW.
int swPin = 18;                 // assign D18 pin to swPin for SW_USER.

2)
const int SEG_TABLE[]=
{                               // segment data table for 7seg_LED
    0x3F,                       // data "0"
    0x06,                       // data "1"
    0x5B,                       // data "2"
    0x4F,                       // data "3"
    0x66,                       // data "4"
    0x6D,                       // data "5"
    0x7D,                       // data "6"
    0x27,                       // data "7"
    0x7F,                       // data "8"
    0x6F,                       // data "9"
    0x00                        // data " "
};

3)
int old_time = 0x0000;          // previous time(milli sec.)

char mini_data = 0;             // minute data 0:1:3:7:F:1F:3F:7F:FF
char sec_data = 0;              // second count data(0 to 59)
char precount1 = 0;             // prescaler for 100milli seconds.(25)
char precount2 = 0;             // prescaler(10) for 1second
char digi_sel = 0;              // digit select

char prescale4sw = 0;           // prescaler for SW check( up to 5:20milli sec.)
int sw_data = 0xFFFF;           // SW data image(history of 20 milli sec. interval)
int time_mode = 0;              // o:stop/1:run
```

図 1.2 プログラムの定義内容

1) 7 セグメント LED のセグメントの駆動で使用するデジタル出力端子を定義します。

セグメント A を制御する segPinA 端子に 0 を指定して D0 に割り当てます。

同様に、セグメント B、C、D、E、F、G、DP も定義します。

7 セグメント LED のコモン (LED 表示の桁選択用) の駆動で使用するデジタル出力端子を定義します。

10 秒の桁を制御する comPin0 端子に 8 を指定して D8 に割り当てます。

1 秒の桁を制御する comPin1 端子に 9 を指定して D9 に割り当てます。

0.1 秒の桁を制御する comPin2 端子に 10 を指定して D10 に割り当てます。

さらに、8 個の単体 LED を制御する comPin3 端子に 11 を指定して D11 を割り当てます。

スイッチの状態の読み込みで使用するデジタル入力端子を定義します。

外部スイッチを制御する ex_swPin 端子に 13 を指定して D13 を割り当てます。

ボード上のスイッチ（SW_USR）を制御する swPin 端子に 18 を指定して D18 に割り当てます。

2) 7 セグメント LED で表示する数字 (0～9) と表示なし (LED 消灯) を指定する配列 SEG_TABLE で制御データを定義します。

3) プログラム (スケッチ) で使用するいくつかの変数を定義します。

old_time :	経過時間 (ミリ秒単位) を確認するための変数
sec_data :	秒をカウントする変数
mini_data :	分をカウントする変数
precount1 :	4 ミリ秒の表示桁切り替えタイミングから 100 ミリ秒を得るためのカウンタ変数
precount2 :	100 ミリ秒をカウントして 1 秒を得る変数
digi_sel :	ダイナミック表示を制御するための変数
prescale4sw :	スイッチ状態を確認するタイミングをカウントする変数
sw_data :	スイッチ状態の変化を保持する変数
time_mode :	時計のカウント / 停止を制御する変数

1.5 初期設定処理

プログラム (スケッチ) の初期設定部分を図 1.3 に示します。

ここでは、setup 関数として、各端子の入出力を指定します。

```
/*-----------------------------------------------------------------------↓
     Arduino setup area. Write setup program here.↓
-----------------------------------------------------------------------*/↓
↓
void setup(void){↓
  // put your setup code here, to run once:↓
    pinMode(segPinA, OUTPUT);          // set D0pin to output mode↓
    pinMode(segPinB, OUTPUT);          // set D1pin to output mode↓
    pinMode(segPinC, OUTPUT);          // set D2pin to output mode↓
    pinMode(segPinD, OUTPUT);          // set D3pin to output mode↓
    pinMode(segPinE, OUTPUT);          // set D4pin to output mode↓
    pinMode(segPinF, OUTPUT);          // set D5pin to output mode↓
    pinMode(segPinG, OUTPUT);          // set D6pin to output mode↓
    pinMode(segPinDP, OUTPUT);         // set D7pin to output mode↓
    pinMode(comPin0, OUTPUT);          // set D8pin to output mode↓
    pinMode(comPin1, OUTPUT);          // set D9pin to output mode↓
    pinMode(comPin2, OUTPUT);          // set D10pin to output mode↓
    pinMode(comPin3, OUTPUT);          // set D11pin to output mode↓
    pinMode(ex_swPin, INPUT_PULLUP);   // set D13pin to input mode↓
    pinMode(swPin, INPUT);             // set D18pin to input mode↓
}↓
```

図 1.3 初期設定処理部分

1.6 メイン処理部

繰り返し実行されるメイン処理の先頭部分を図 1.4 に示します。「プロジェクトの起動準備」が正しく設定されていると、デバッガの起動時の画面は図 1.4 となります。

```
void loop(void){
  // put your main code here, to run repeatedly:

    int time_work;                     // present time buffer
    char com_sel;                      // common select signal
    char seg_data;                     // segment data
    char sw_work;                      // work for switch check

/*-------------------------------------
    wait for 4milli seconds interval.
-------------------------------------*/

    time_work = ( int )( millis() & 0x0FFC ); // read milli sec data

    if ( old_time != time_work )       // check timing of change LED
    {
```

図 1.4 メイン処理の先頭部分

2. 動作確認条件

本アプリケーションノートのサンプルコードは、下記の条件で動作を確認しています。

表 2.1 動作確認条件

項目	内容
使用マイコン	RL78/G14 (R5F104MLAFB)
動作周波数	● 高速オンチップ・オシレータ・クロック（f_{IH}）：32MHz ● CPU／周辺ハードウェア・クロック： 32MHz
動作電圧	3.3V（2.75V～5.5V で動作可能） LVD 動作：リセット・モード LVD 検出電圧（V_{LVD}） 立ち上がり時 TYP. 2.81V (2.76V～2.87V) 立ち下がり時 TYP. 2.75V (2.70V～2.81V)
統合開発環境	ルネサス エレクトロニクス CS+ for CC V8.03.00 ルネサス エレクトロニクス e² studio V7.7.0 IAR Systems IAR Embedded Workbench for RL78
C コンパイラ	ルネサス エレクトロニクス CC-RL V1.09.00 IAR Systems RL78 用 IAR C/C++ コンパイラ v4.20.1

3. 関連アプリケーションノート

本アプリケーションノートに関連するアプリケーションノートを以下に示します。

併せて参照してください。

RL78 ファミリ Arduino API 導入ガイド (R01AN5413)

RL78/G14 オンボード LED 点滅制御（Arduino API）（R01AN5384）

4. ハードウェア説明

4.1 ハードウェア構成例

本アプリケーションノートで使用するハードウェア（FPB）を図 4.1 に示します。

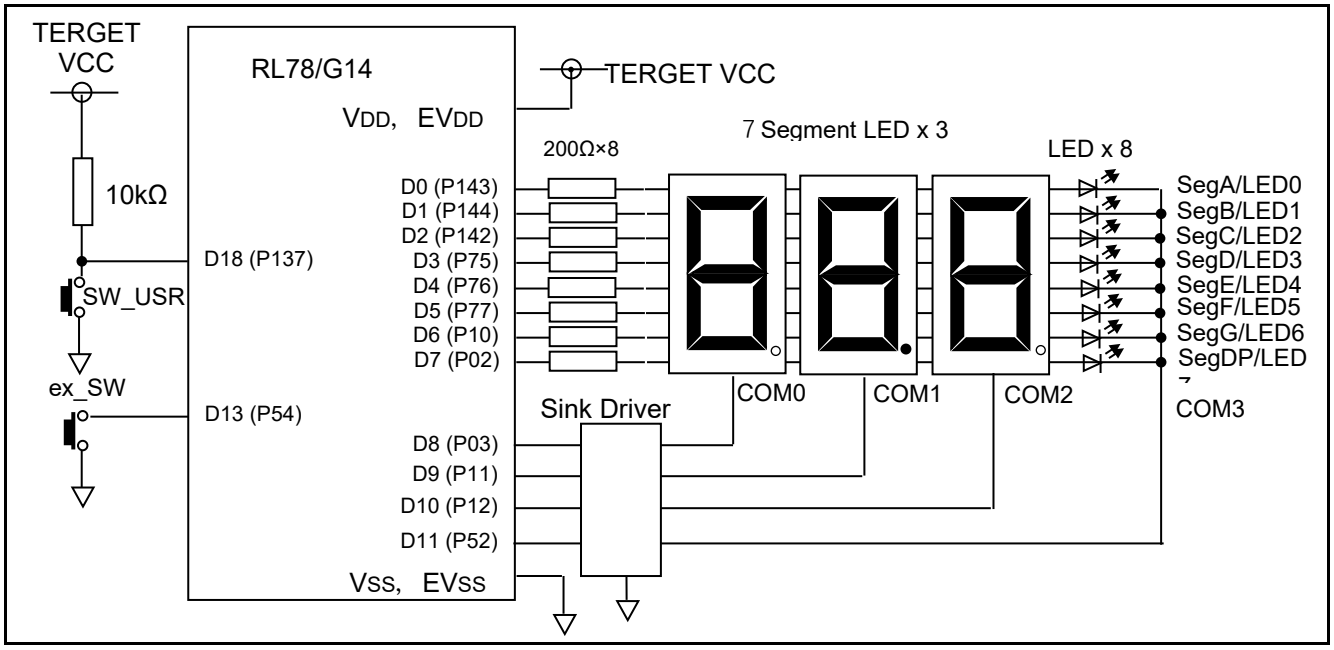

図 4.1 ハードウェア構成例

注意 この回路イメージは接続の概要を示す為に簡略化しています。

電源電圧は USB から 3.3V を供給しています。

4.2 使用端子一覧

使用端子と機能を表 4.1 に示します。

表 4.1 使用端子と機能

端子	ポート名	入出力	機能
D16	P43	出力	LED0 を制御
D17	P44	出力	LED1 を制御
D18	P137	入力	スイッチ (SW_USR) 入力

5. ソフトウェア説明

5.1 動作概要

本アプリケーションノートでは、初期設定 (端子の設定) が完了して、メイン処理 (loop) が起動すると、7 セグメント LED の表示は、「00.0」となります。

ex_SW または SW_USR スイッチを押下すると、7 セグメント LED の表示がカウントアップします。再度 ex_SW または SW_USR スイッチを押下すると、カウントが停止します。以下、ex_SW または SW_USR スイッチを押下するごとに、カウントの再開と停止を繰り返します。

カウントアップし、7 セグメント LED の表示が再び「00.0」になると、LED が 1 個点灯します。7 セグメント LED の表示が「00.0」になるごとに LED が 1 個ずつ追加で点灯していきます。8 個の LED が全て点灯して 7 セグメント LED の表示が「00.0」になると、8 個の LED は全て消灯します。

以降、上記動作を繰り返します。

下記①～③に詳細を記載します。

① setup 関数で使用する端子の設定を行います。

- 7 セグメント LED のセグメント駆動用端子 (segPinA～segPinDP) をデジタル出力に設定します。
- 7 セグメント LED のコモン駆動用端子 (comPin0～comPin3) をデジタル出力に設定します。
- 外部スイッチ (ex_SW) の状態確認用端子 (ex_swPin) を内蔵プルアップ抵抗有効にした入力モードに設定します。
- オンボード・スイッチ (SW_USR) の状態確認用端子 (swPin) をデジタル入力に設定します。

② loop 関数でメイン処理を行います。

- 起動時からの経過時間 (ミリ秒単位) のビット 11～ビット 2 の 10 ビット分を取得します。
- 取得データが古いデータ (old_time) から変化しているか確認します。
- 変化していなければ、処理を終了して loop 関数の先頭に戻ります。
- 変化 (4 ミリ秒経過) していたら、old_time を取得データに変更します。
- ビット 3、ビット 2 の値に応じて処理を切り替えます。

  (ビット 3, ビット 2) = (0, 0) の場合、comPin3 端子を Low (選択解除) に設定します。次に、10 秒の桁のセグメント・データを求め、set_SEG 関数で segPin に設定します。表示桁の指定 (comPin0 端子) を変数 com_sel に設定します。

  (ビット 3, ビット 2) = (0, 1) の場合、comPin0 端子を Low (選択解除) に設定します。次に、1 秒の桁のセグメント・データを求め、set_SEG 関数で segPin に設定します。このとき、DP も点灯に設定します。表示桁の指定 (comPin1 端子) を変数 com_sel に設定します。

  (ビット 3, ビット 2) = (1, 0) の場合、comPin1 端子を Low (選択解除) に設定します。次に、100 ミリ秒の桁のセグメント・データを求め、set_SEG 関数で segPin に設定します。表示桁の指定（comPin2 端子）を変数 com_sel に設定します。

  (ビット 3, ビット 2) = (1, 1) の場合、comPin2 端子を Low (選択解除) に設定します。次に、分の桁データを set_SEG 関数で segPin に設定します。表示桁の指定（comPin3 端子）を変数 com_sel に設定します。
- 変数 com_sel で指定された表示桁を選択して表示を行います。

- カウント許可 (変数 time_mode が 1) のとき、時間のカウントを行います。

  変数 precount1 (4 ミリ単位) をカウントアップして 25 (100 ミリ秒経過) になったら、precount1 をクリアして変数 precount2 (100 ミリ秒単位) をカウントアップします。

  変数 precount2 (100 ミリ秒単位) をカウントアップして 10 (1 秒経過) になったら、変数 precount2 をクリアして変数 sec_data (1 秒単位) をカウントアップします。

  変数 sec_data (1 秒単位) をカウントアップして 60 (1 分経過) になったら、変数 sec_data をクリアして、変数 mini_data (1 分単位) を更新します。mini_data の値が 0xFF の場合、0x00 に設定します。mini_data の値が 0xFF でない場合、1 ビット左シフトして LSB (最下位ビット) をセット (1) します。mini_data の値は、0x00、0x01、0x03、0x07、0x0F、0x1F、0x3F、0x7F、0xFF と変化します。

- 20 ミリ秒毎に、スイッチの状況を確認します。

  変数 prescale4sw をカウントアップして 5 (20 ミリ秒) になったら、変数 prescale4sw をクリアします。変数 sw_data を 1 ビット左シフトしてビット 0 に 2 つのスイッチ状態の論理和 (スイッチは負論理なので、論理積) をとった結果を設定します。どちらかのスイッチが押された場合は、変数 time_mode を変更します。

③ poll_sw 関数で、SW_USR スイッチの状態を 10m ミリ秒ごとに確認し、押された回数をカウントします。

- 引数のセグメントのデータを作業用変数 work_data に読み込みます。
- 変数 work_data の LSB (最下位ビット) を segPinA 端子に出力し、変数 work_data を 1 ビット右シフトします。
- 変数 work_data の LSB (最下位ビット) を segPinB 端子に出力し、変数 work_data を 1 ビット右シフトします。
- 変数 work_data の LSB (最下位ビット) を segPinC 端子に出力し、変数 work_data を 1 ビット右シフトします。
- 変数 work_data の LSB (最下位ビット) を segPinD 端子に出力し、変数 work_data を 1 ビット右シフトします。
- 変数 work_data の LSB (最下位ビット) を segPinE 端子に出力し、変数 work_data を 1 ビット右シフトします。
- 変数 work_data の LSB (最下位ビット) を segPinF 端子に出力し、変数 work_data を 1 ビット右シフトします。
- 変数 work_data の LSB (最下位ビット) を segPinG 端子に出力し、変数 work_data を 1 ビット右シフトします。
- 変数 work_data の LSB (最下位ビット) を segPinDP 端子に出力します。

5.2 定数一覧

表 5.1 にサンプルコードで使用する定数を示します。

表 5.1 サンプルコードで使用する定数

定数名	設定値	内容
segPinA	**0**	segment A を制御する端子の番号
segPinB	**1**	segment B を制御する端子の番号
segPinC	**2**	segment C を制御する端子の番号
segPinD	**3**	segment D を制御する端子の番号
segPinE	**4**	segment E を制御する端子の番号
segPinF	**5**	segment F を制御する端子の番号
segPinG	**6**	segment G を制御する端子の番号
segPinDP	**7**	segment DP を制御する端子の番号
comPin0	**8**	10 秒の桁を選択する端子の番号
comPin1	**9**	1 秒の桁を選択する端子の番号
comPin2	**10**	100 ミリ秒の桁を選択する端子の番号
comPin3	**11**	単体 LED を選択する端子の番号
ex_swPin	**13**	外部スイッチ (ex_SW) を読む端子の番号
swPin	**18**	SW_USR を読む端子の番号
SEG_TABLE[]	**0x3F**	7 セグメント LED に 0 を表示するデータ
	0x06	7 セグメント LED に 1 を表示するデータ
	0x5B	7 セグメント LED に 2 を表示するデータ
	0x4F	7 セグメント LED に 3 を表示するデータ
	0x66	7 セグメント LED に 4 を表示するデータ
	0x6D	7 セグメント LED に 5 を表示するデータ
	0x7D	7 セグメント LED に 6 を表示するデータ
	0x27	7 セグメント LED に 7 を表示するデータ
	0x7F	7 セグメント LED に 8 を表示するデータ
	0x6F	7 セグメント LED に 9 を表示するデータ
	0x00	7 セグメント LED を消灯するデータ
HIGH	**0x01**	ハイレベル
LOW	**0x00**	ロウレベル

5.3 変数一覧

表 5.2 にグローバル変数を示します。

表 5.2 グローバル変数

型	変数名	内容	使用関数
int	old_time	前回の起動からの経過時間 (ミリ秒単位)	loop()
char	mini_data	分の経過を示すデータ: 0x00/0x01/0x03/0x07/0x0F/0x1F/0x3F/0x7F/0xFF	loop()
char	sec_data	秒データ	loop()
char	precount1	100 ミリ秒のタイミング生成用 (4 ミリ秒毎に更新)	loop()
char	Precount2	1 秒のタイミング生成用 (100 ミリ秒毎に更新)	loop()
char	digi_sel	表示する桁 (端子番号: 8～11) を指定する。	loop()
char	prescale4sw	20 ミリ秒のタイミング生成用	loop()
Int	sw_data	スイッチの状態変化を格納する。(初期値は 0xFF)	loop()
Int	time_mode	時間カウントを制御するフラグ （0:カウントしない／1:カウントする）	loop()

5.4 関数一覧

表 5.3 に関数一覧を示します。

表 5.3 関数一覧

関数名	概要
loop	メイン処理 (スケッチ)
setup	初期化関数 (スケッチ)
set_SEG	7 セグメント LED への表示データ設定関数 (スケッチ)
pinMode	端子の動作モード (入力モード / 出力モード / 内蔵プルアップ抵抗を有効にした入力モード) 指定
digitalWrite	端子へのデータ出力
digitalRead	端子状態を読み出す。
micros	プログラムの実行を開始した時から現在までの時間をマイクロ秒単位で返します。
millis	プログラムの実行を開始した時から現在までの時間をミリ秒単位で返します。
delay	ミリ秒単位でプログラムを指定した時間だけ止めます。
delayMicroseconds	マイクロ秒単位でプログラムを指定した時間だけ止めます。

5.5 関数仕様

サンプルコードの関数仕様を示します。

[関数名] loop	
概　要	メイン関数
ヘッダ	AR_LIB_PORT.h、AR_LIB_TIME.h、r_cg_macrodriver.h、AR_SKETCH.h、r_cg_userdefine.h
宣　言	void loop(void)
説　明	スタートすると、起動からの時間を確認して 4 ミリ秒ごとに表示する 7 セグメント LED の桁データの出力を切り替える。更に、カウント許可の場合、100 ミリ秒、1 秒、1 分のカウントを行う。 20 ミリ秒ごとにスイッチの状態を確認し、押下されている場合はカウント許可フラグを変更する。
引　数	なし
リターン値	なし

[関数名] setup	
概　要	初期化関数
ヘッダ	AR_LIB_PORT.h、r_cg_macrodriver.h、r_cg_userdefine.h
宣　言	void setup(void);
説　明	プログラム (スケッチ) で使用する端子の設定を行う。
引　数	なし
リターン値	なし

[関数名] set_SEG	
概　要	7 セグメント LED への表示データ設定関数
ヘッダ	AR_LIB_PORT.h
宣　言	void set_SEG(char data);
説　明	引数で渡されたデータを 7 セグメント LED の点灯データとして対応するセグメントの制御端子に設定する。
引　数	Data : 点灯させるセグメントのデータ
リターン値	なし

[関数名] pinMode	
概　要	端子機能設定関数
ヘッダ	AR_LIB_PORT.h、r_cg_macrodriver.h、r_cg_userdefine.h
宣　言	void pinMode(uint8_t pin,uint8_t mode)
説　明	第 1 引数で示された端子を第 2 引数で示されたモードに設定する。
引　数	uint8_t pin : 指定する端子の番号 uint8_t mode : OUTPUT/INPUT/INPUT_PULLUP で端子のモードを指定
リターン値	なし

[関数名] digitalWrite	
概　要	端子へのデジタル・データ出力関数
ヘッダ	AR_LIB_PORT.h、r_cg_macrodriver.h、r_cg_userdefine.h
宣　言	void digitalWrite(uint8_t pin, uint8_t value);
説　明	第 1 引数で示す端子に、第 2 引数で示すデータを出力する
引　数	uint8_t pin : 出力する端子の番号 uint8_t value : 出力するデータ (HIGH/LOW)
リターン値	なし

[関数名] digitalRead	
概　要	端子からのデジタル・データの読み出し関数
ヘッダ	AR_LIB_PORT.h、r_cg_macrodriver.h、r_cg_userdefine.h
宣　言	uint8_t digitalRead(uint8_t pin);
説　明	引数で指定された端子の状態を読み出す。
引　数	uint8_t pin : 読み出す端子の番号
リターン値	uint8_t : 読み出したデータ (HIGH/LOW)

[関数名] millis	
概　要	ミリ秒単位での経過時間を得る関数
ヘッダ	AR_LIB_TIME.h、r_cg_macrodriver.h、r_cg_userdefine.h
宣　言	uint32_t millis (void);
説　明	起動してからのミリ秒単位の経過時間を戻す。
引　数	なし
リターン値	uint32_t : ミリ秒単位の経過時間

5.6 フローチャート

5.6.1 初期設定関数

図 5.1 と図 5.2 に初期設定のフローを示します。

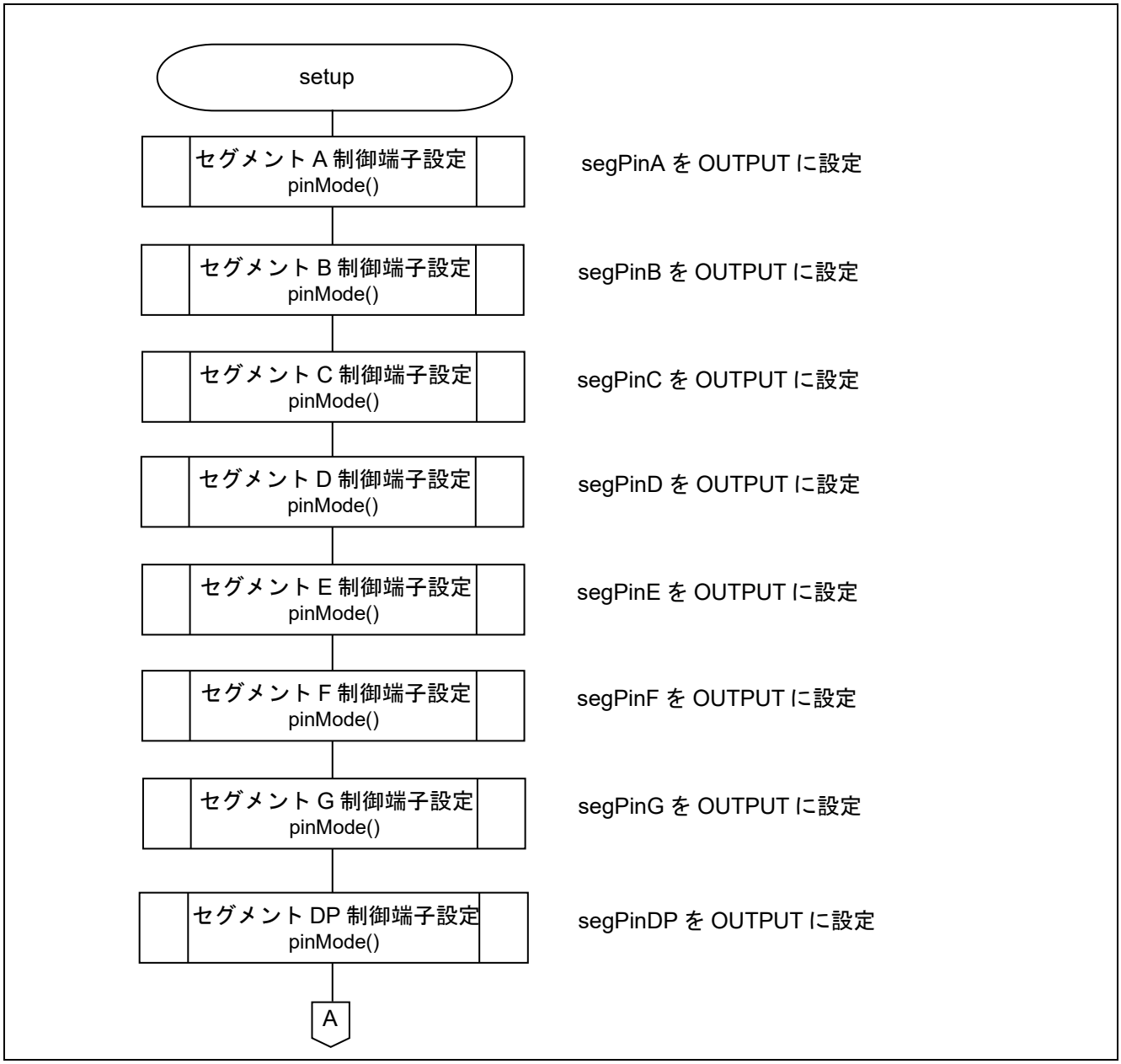

図 5.1 初期設定関数 (1/2)

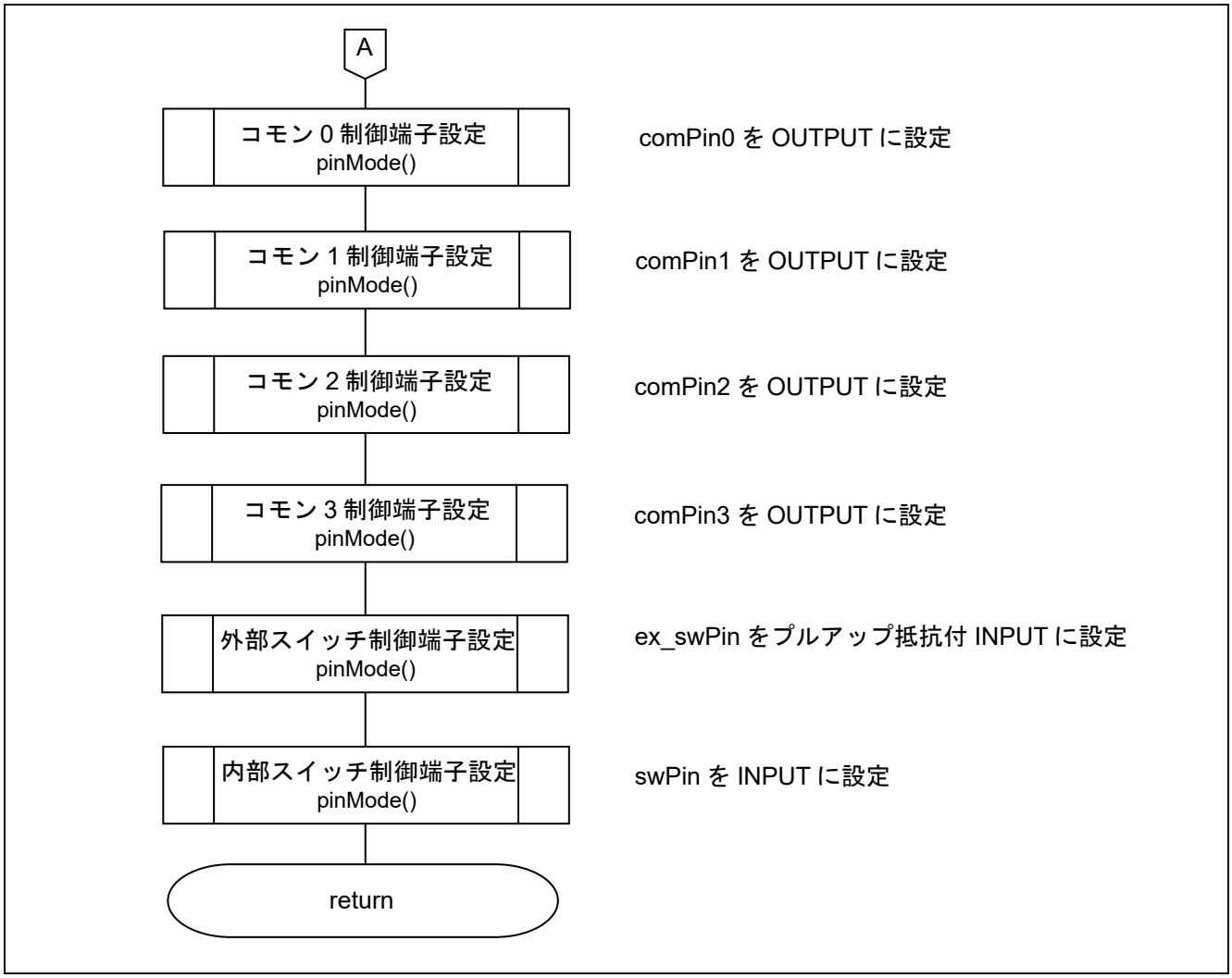

図 5.2 初期設定関数 (2/2)

5.6.2 メイン処理関数

図 5.3～図 5.6 にメイン処理関数のフローチャートを示します。

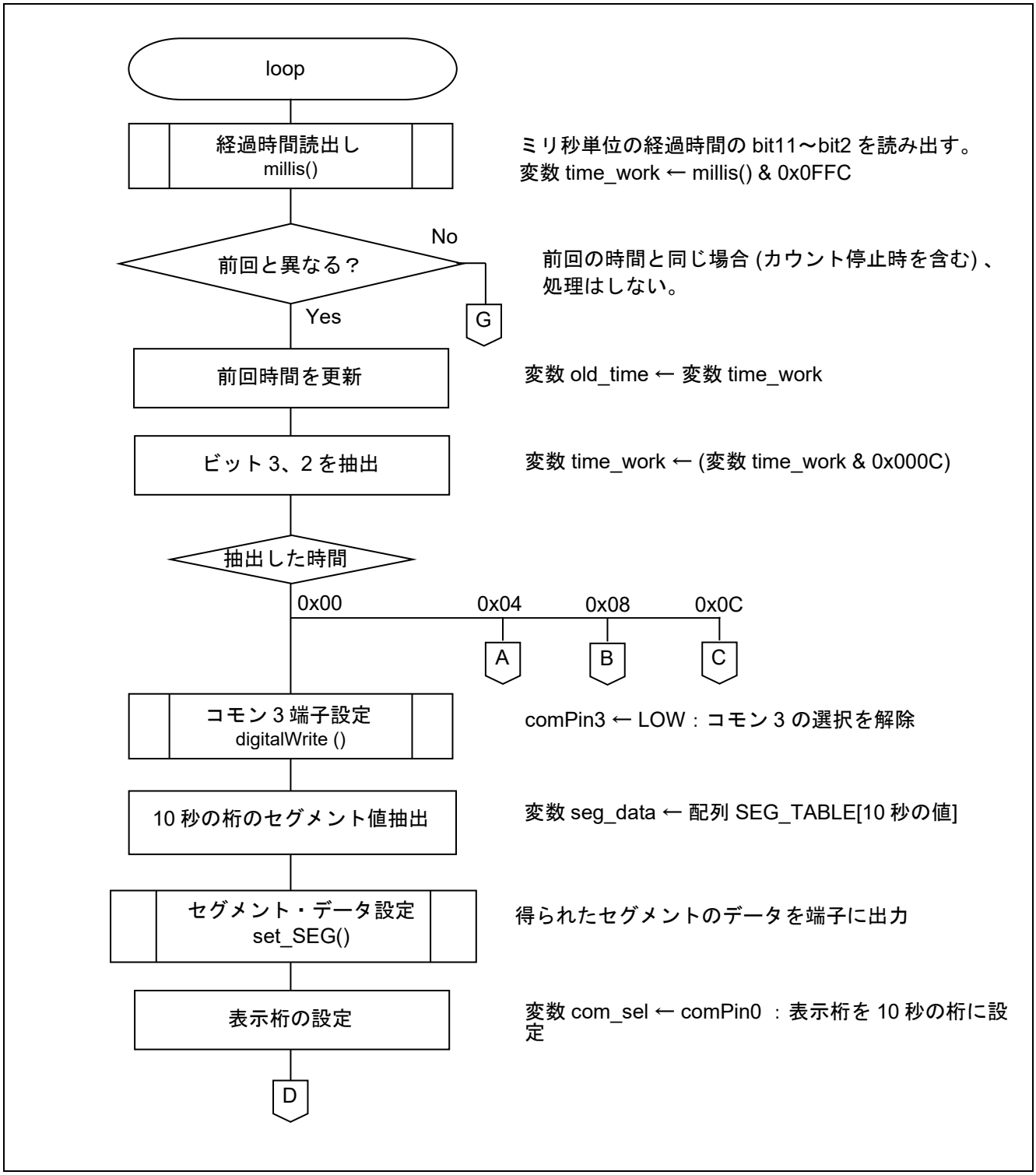

図 5.3 メイン関数 (1/4)

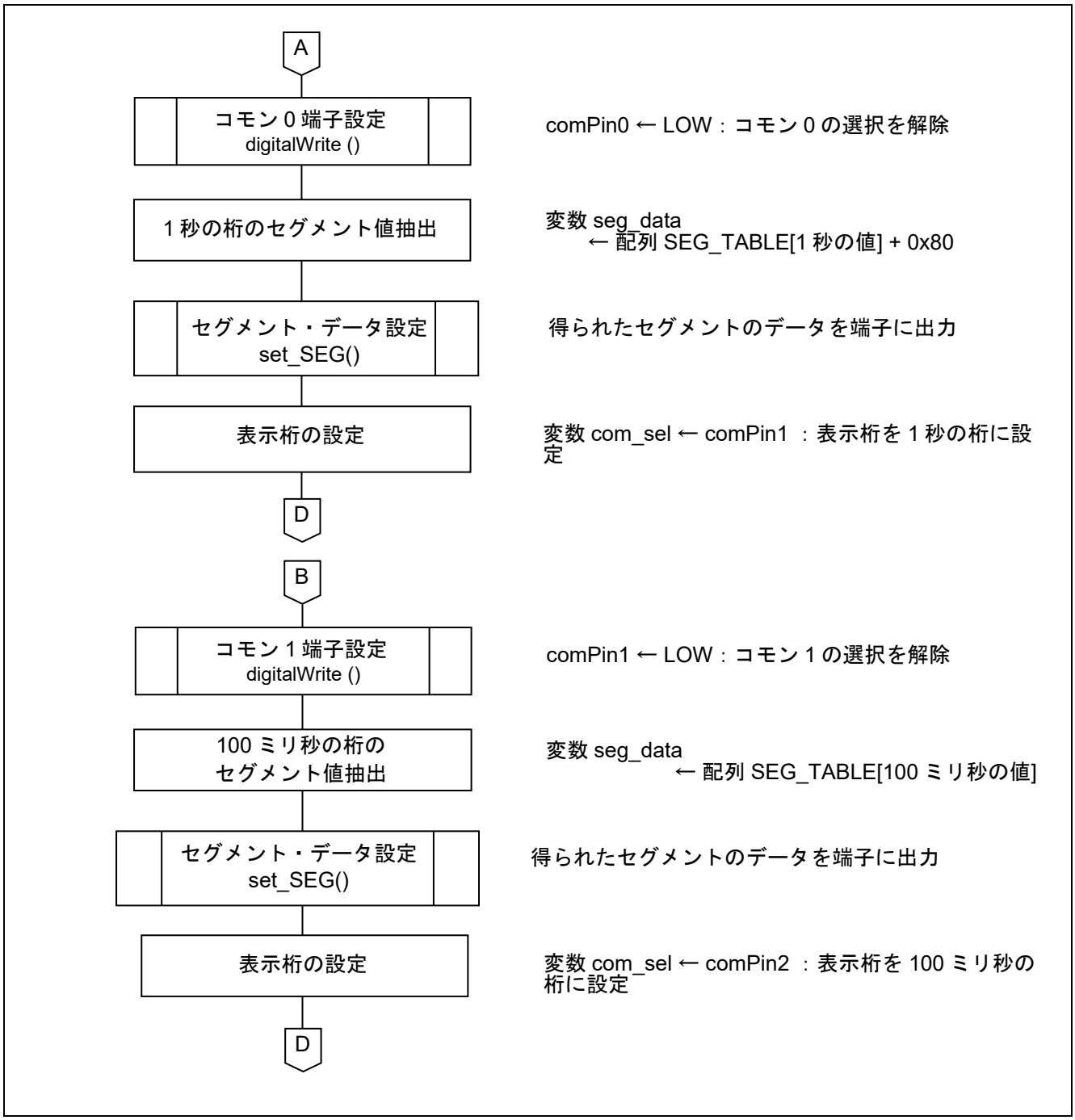
A
コモン 0 端子設定
digitalWrite ()
comPin0 ← LOW：コモン 0 の選択を解除
1 秒の桁のセグメント値抽出
変数 seg_data
← 配列 SEG_TABLE[1 秒の値] + 0x80
セグメント・データ設定
set_SEG()
得られたセグメントのデータを端子に出力
表示桁の設定
変数 com_sel ← comPin1 ：表示桁を 1 秒の桁に設定
D
B
コモン 1 端子設定
digitalWrite ()
comPin1 ← LOW：コモン 1 の選択を解除
100 ミリ秒の桁の
セグメント値抽出
変数 seg_data
← 配列 SEG_TABLE[100 ミリ秒の値]
セグメント・データ設定
set_SEG()
得られたセグメントのデータを端子に出力
表示桁の設定
変数 com_sel ← comPin2 ：表示桁を 100 ミリ秒の桁に設定
D

図 5.4　メイン関数 (2/4)

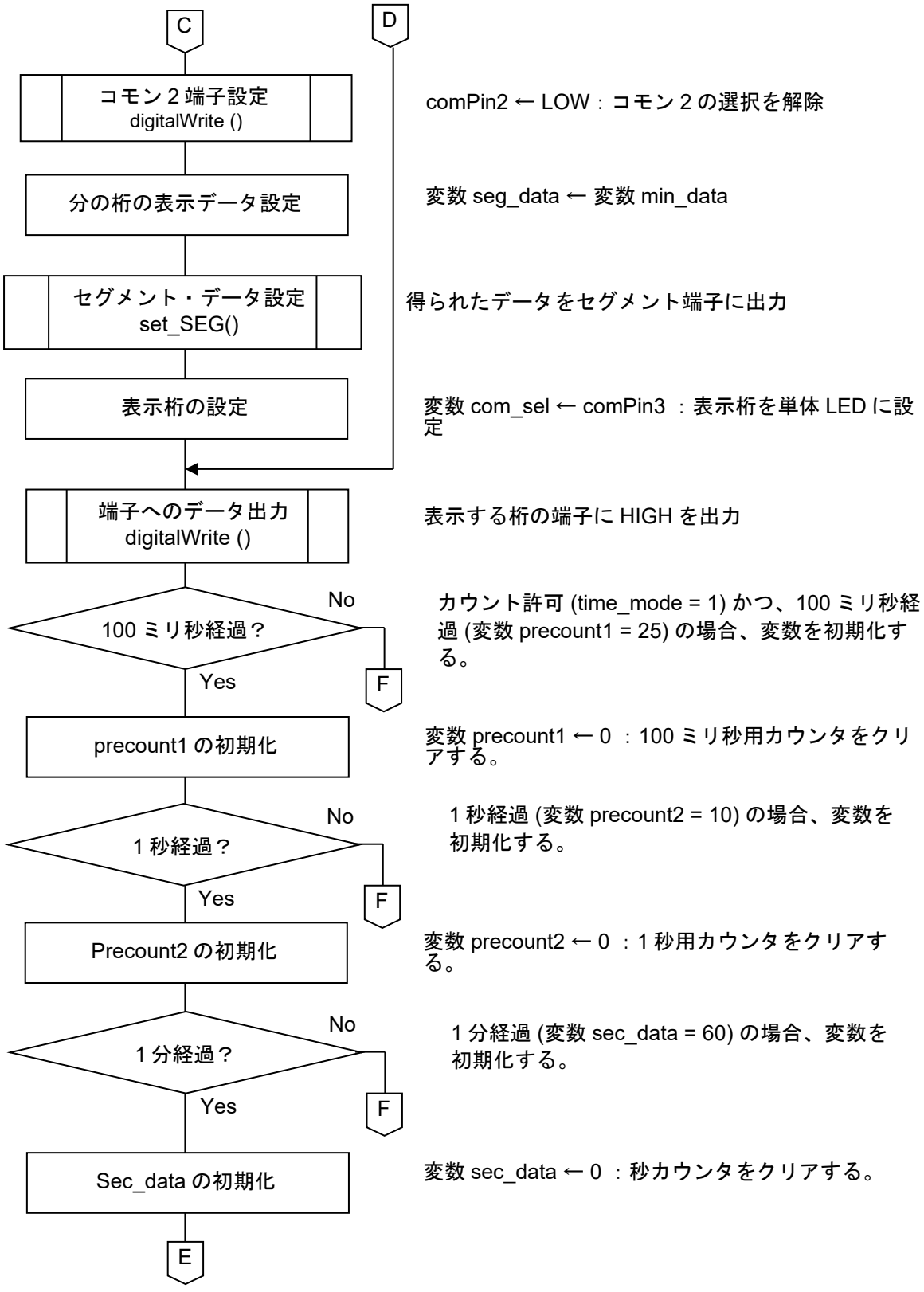
C
D
コモン 2 端子設定
digitalWrite ()
comPin2 ← LOW：コモン 2 の選択を解除
分の桁の表示データ設定
変数 seg_data ← 変数 min_data
セグメント・データ設定
set_SEG()
得られたデータをセグメント端子に出力
表示桁の設定
変数 com_sel ← comPin3 ：表示桁を単体 LED に設定
端子へのデータ出力
digitalWrite ()
表示する桁の端子に HIGH を出力
100 ミリ秒経過？
No
Yes
F
カウント許可 (time_mode = 1) かつ、100 ミリ秒経過 (変数 precount1 = 25) の場合、変数を初期化する。
precount1 の初期化
変数 precount1 ← 0 ：100 ミリ秒用カウンタをクリアする。
1 秒経過？
No
Yes
F
1 秒経過 (変数 precount2 = 10) の場合、変数を初期化する。
Precount2 の初期化
変数 precount2 ← 0 ：1 秒用カウンタをクリアする。
1 分経過？
No
Yes
F
1 分経過 (変数 sec_data = 60) の場合、変数を初期化する。
Sec_data の初期化
変数 sec_data ← 0 ：秒カウンタをクリアする。
E

図 5.5 メイン関数 (3/4)

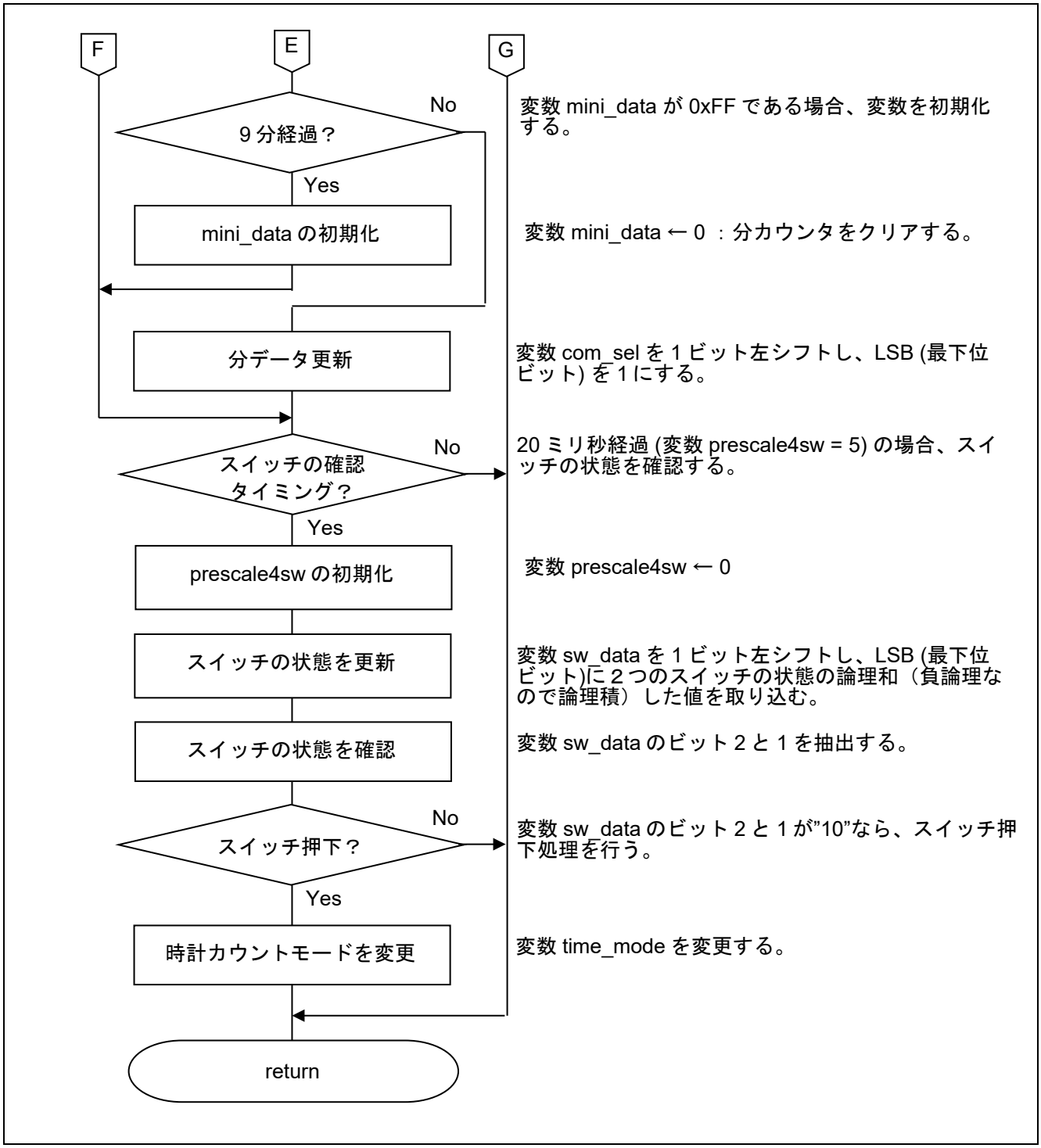
F
E
G
9 分経過？
No
Yes
mini_data の初期化
変数 mini_data が 0xFF である場合、変数を初期化する。
変数 mini_data ← 0 ：分カウンタをクリアする。
分データ更新
変数 com_sel を 1 ビット左シフトし、LSB (最下位ビット) を 1 にする。
スイッチの確認タイミング？
No
Yes
20 ミリ秒経過 (変数 prescale4sw = 5) の場合、スイッチの状態を確認する。
prescale4sw の初期化
変数 prescale4sw ← 0
スイッチの状態を更新
変数 sw_data を 1 ビット左シフトし、LSB (最下位ビット)に 2 つのスイッチの状態の論理和（負論理なので論理積）した値を取り込む。
スイッチの状態を確認
変数 sw_data のビット 2 と 1 を抽出する。
スイッチ押下？
No
Yes
変数 sw_data のビット 2 と 1 が”10”なら、スイッチ押下処理を行う。
時計カウントモードを変更
変数 time_mode を変更する。
return

図 5.6　メイン関数（4/4）

5.6.3 7 セグメント LED への表示データ設定関数

図 5.7 と図 5.8 に 7 セグメント LED への表示データ設定関数のフローチャートを示します。

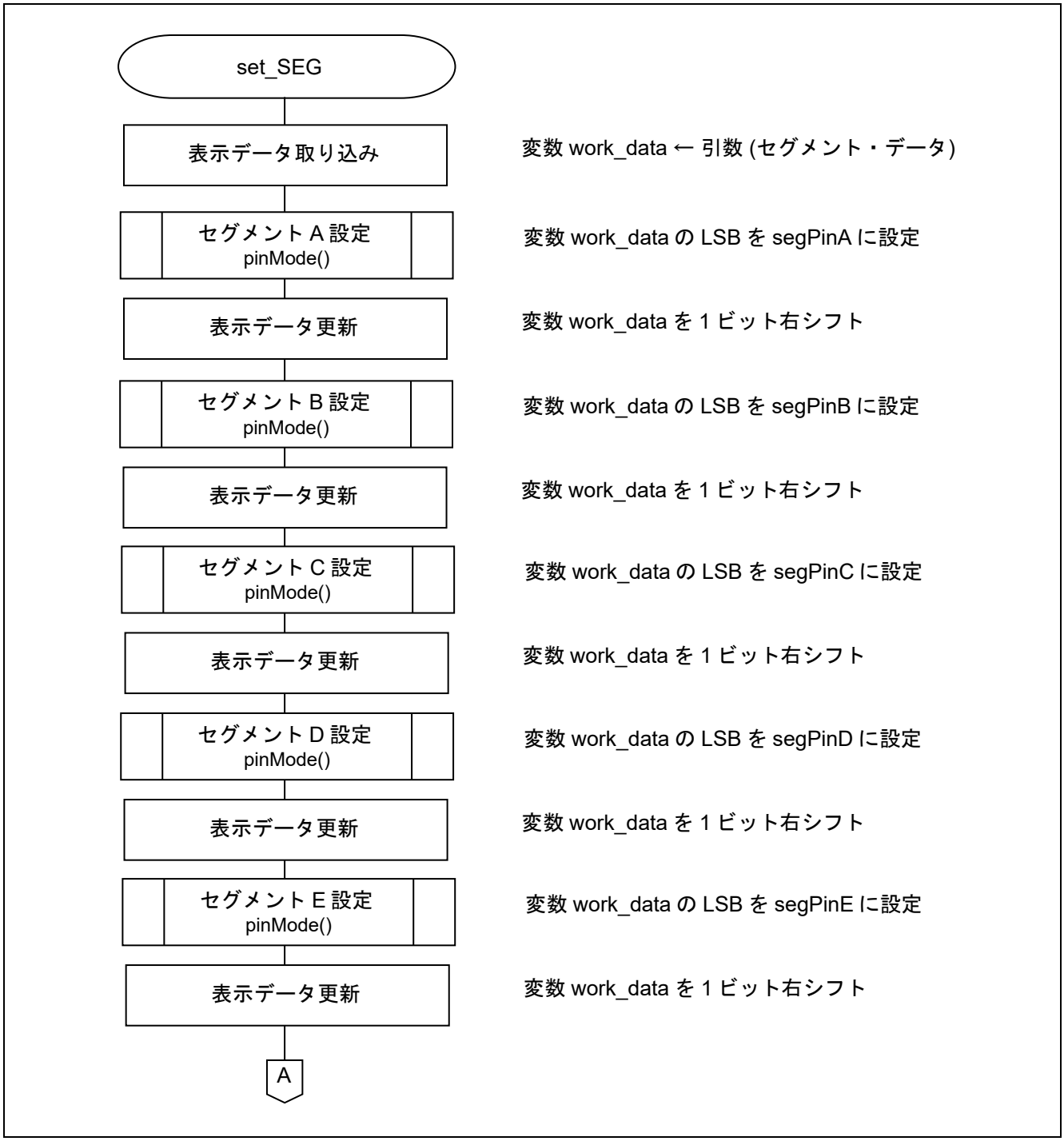

図 5.7 セグメント LED への表示データ設定関数 (1/2)

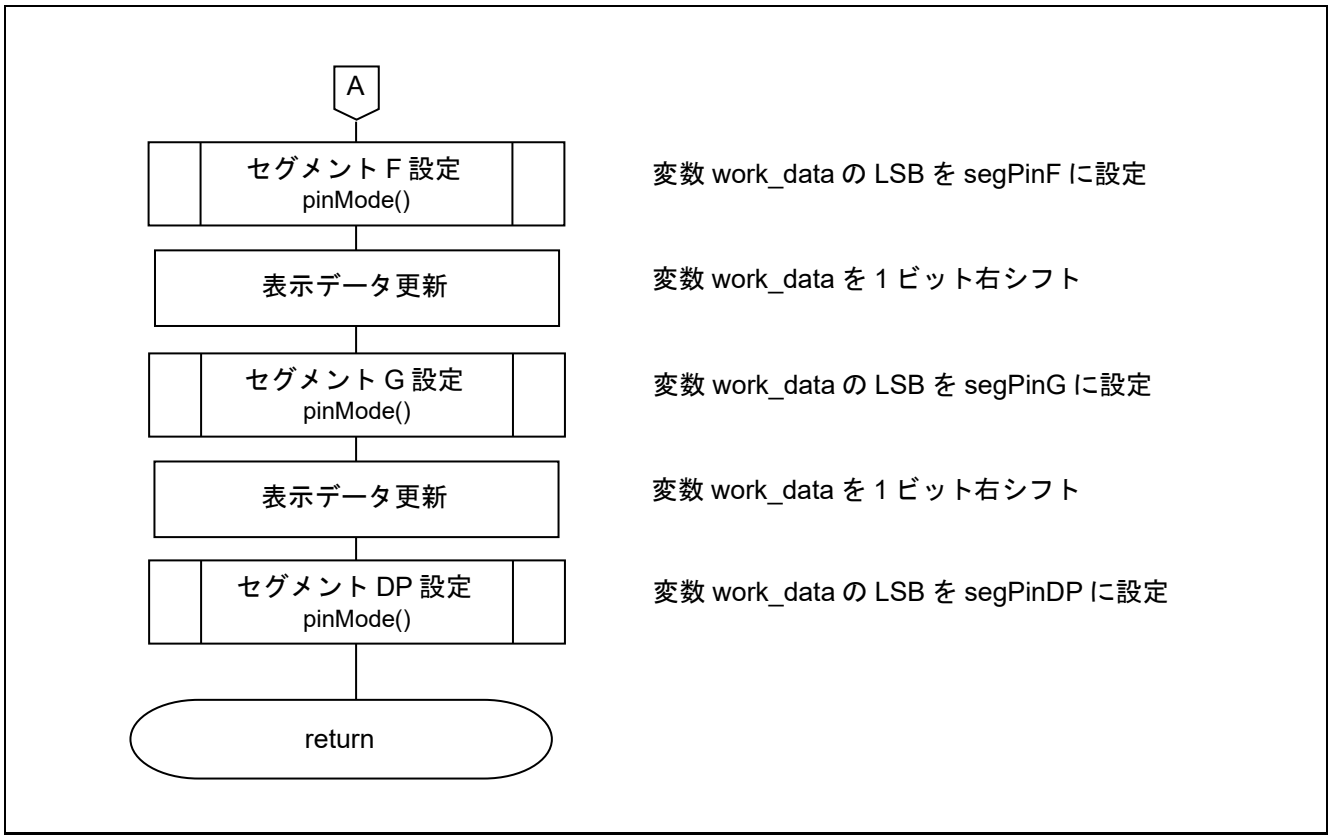

図 5.8 セグメント LED への表示データ設定関数 (2/2)

6. サンプルコード

サンプルコードは、ルネサス エレクトロニクスホームページから入手してください。

7. 参考ドキュメント

エラー! プロパティ名に誤りがあります。 ユーザーズマニュアル ハードウェア編 (R01UH0186)
RL78 ファミリ ユーザーズマニュアル ソフトウェア編 (R01US0015)
RL78/G14 Fast Prototyping Board ユーザーズマニュアル (R20UT4573)
（最新版をルネサス エレクトロニクスホームページから入手してください。）

テクニカルアップデート／テクニカルニュース
（最新の情報をルネサス エレクトロニクスホームページから入手してください。）

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ
http://japan.renesas.com

お問合せ先
http://japan.renesas.com/contact/

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2020.7.10	—	初版発行

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 静電気対策
   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。
2. 電源投入時の処置
   電源投入時は、製品の状態は不定です。電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. 電源オフ時における入力信号
   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。
4. 未使用端子の処理
   未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。
5. クロックについて
   リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
6. 入力端子の印加波形
   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。
7. リザーブアドレス（予約領域）のアクセス禁止
   リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
8. 製品間の相違について
   型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品、本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。
   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等
   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等
   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
6. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
10. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
11. 本資料の全部または一部を当社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禁じます。
12. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

(Rev.4.0-1 2017.11)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。