TOSHIBA

TMP89FS60FG サンプルソフトウェア

Rev 1.0
2017 年 10 月

このサンプルソフトウェアは、あなたが東芝製マイクロコントローラを使用して製品を開発する際の、プログラム開発を習得する手助けとなるものです。

本ドキュメントと、サンプルプログラムは、下記のＷｅｂからダウンロードすることが出来ます。
https://toshiba.semicon-storage.com/jp/design-support/document/application-note.html
又は、
http://toshiba.semicon-storage.com/jp/top.html/ → マイクロコンピュータ → アプリケーションノート

目次

1. 概要

このサンプルソフトウェアは TMP89FS60FG 用です。

このサンプルプログラムは、主な MCU 内蔵機能をひとつずつ実行するようにできています。このサンプルプログラム内の一部を取り出して再利用することで、希望する機能を動作させることができます。

2. 使用する機能

機能	チャンネル	使用／未使用
SLOW/STOP		使用
ウォッチドッグタイマ		使用
電圧検出		使用
タイムベースタイマ		タイミング作成
8 ビットタイマ	TC00	8 ビット PWM 出力
	TC01	8 ビット PPG 出力
	TC02	未使用
	TC03	未使用
16 ビットタイマ	TCA0	16 ビット PPG 出力
	TCA1	未使用
時計用タイマ		0.5 秒信号出力
UART	UART0	未使用
	UART1	未使用
	UART2	UART データ出力
SIO	SIO0	未使用
	SIO1	SIO データ出力
シリアルバス	SBI	I2C データ出力
10 ビット A/D コンバータ	AIN8	A/D データ入力
	AIN0-AIN7	未使用
	AIN9-AIN15	未使用
外部割り込み	INT0	SLOW モード起動入力
	INT5	STOP モード起動入力
	INT1/2/3/4	未使用

3. 端子用途

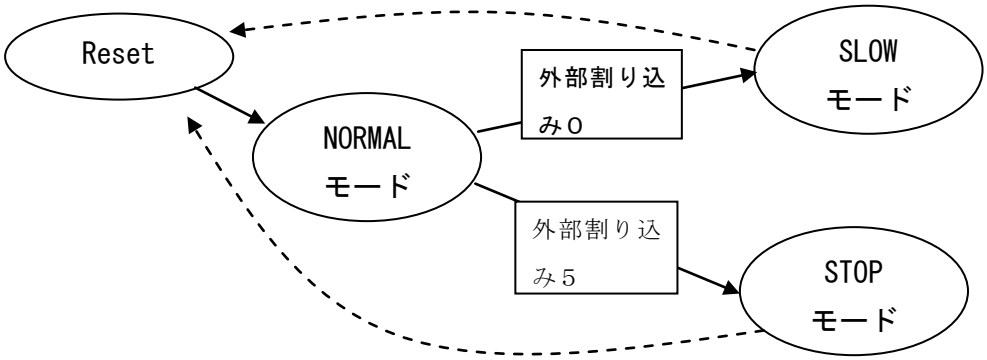
No	端子名称	用途
1	VSS	GND
2	P00(XIN)	高周波発振子接続
3	P01(XOUT)	高周波発振子接続
4	MODE	テスト(ローレベルに固定)
5	VDD	+5V
6	P02(XTIN)	低周波発振子接続
7	P03(XTOUT)	低周波発振子接続
8	P10(^RESET)	リセット信号入力
9	P11(^INT5/^STOP)	**外部割込み 5**
10	P12(^INT0)	**外部割込み 0**
11	P13(INT1)	未使用
12	P20(TXD0/SO0/OCDCK)	未使用
13	P21(RXD0/SI0/OCKI0)	未使用
14	P22(SCLK0)	未使用
15	P23(SDA0/SO0)	**I2C バスデータ出力**
16	P24(SCL0/SI0)	**I2C バスクロック出力**
17	P25(SCLK0)	未使用
18	P26	未使用
19	P27	未使用
20	AVSS	GND
21	AVDD	+5V
22	VAREF	+5V
23	P40(AIN0/KWI0)	未使用
24	P41(AIN1/KWI1)	未使用
25	P42(AIN2/KWI2)	未使用
26	P43(AIN3/KWI3)	未使用
27	P44(AIN4/KWI4)	未使用
28	P45(AIN5/KWI5)	未使用
29	P46(AIN6/KWI6)	未使用
30	P47(AIN7/KWI7)	未使用
31	P50(AIN8)	**アナログ信号入力**
32	P51(AIN9)	未使用
33	P52(AIN10)	未使用
34	P53(AIN11)	未使用
35	P54(AIN12)	未使用
36	P55(AIN13)	未使用

No	端子名称	用途
37	P56(AIN14)	未使用
38	P57(AIN15)	未使用
39	P70(TC00/^PPG00/^PWM00)	**8 ビット PWM 出力**
40	P71(TC01/^PPG01/^PWM01)	**8 ビット PPG 出力**
41	P72(TCA0/^PPGA0)	**16 ビット PPG 出力**
42	P73(TCA1/^PPGA1)	未使用
43	P74(^DV0)	未使用
44	P75(INT2)	未使用
45	P76(INT3)	未使用
46	P77(INT4)	未使用
47	P80(TC02/^PPG02/^PWM02)	未使用
48	P81(TC03/^PPG03/^PWM03)	未使用
49	P82	未使用
50	P83	未使用
51	P84	**0.5 秒信号出力**
52	P90(TXD1/SO1)	**SIO データ出力**
53	P91(RXD1/SI1)	未使用
54	P92(SCLK1)	**SIO クロック出力**
55	P93(TXD2)	**UART データ出力**
56	P94(RXD2)	Unused
57	PB0	**A/D 変換値出力**
58	PB1	**A/D 変換値出力**
59	PB2	**A/D 変換値出力**
60	PB3	**A/D 変換値出力**
61	PB4	**A/D 変換値出力**
62	PB5	**A/D 変換値出力**
63	PB6	**A/D 変換値出力**
64	PB7	**A/D 変換値出力**

4. 機能

4-1. 動作モード選択

NORMAL モードでは高周波クロックおよび低周波クロックは発振状態とします。NORMAL モード中に INT5/INT0 入力により動作モード(NORMAL/SLOW/STOP)を変更します。移行した動作モードはリセットがかかるまでその状態を保持します。



これらのモードでの各動作は次の通りとします。

機能	NORMAL モード	SLOW モード	STOP モード
ウォッチドッグタイマ	**動作**	**動作**	停止
8 ビット PWM 出力	**動作**	停止	停止
8 ビット PPG 出力	**動作**	停止	停止
16 ビット PPG 出力	**動作**	停止	停止
UART データ出力	**動作**	停止	停止
SIO データ出力	**動作**	停止	停止
I2C データ出力	**動作**	停止	停止
A/D データ入力/出力	**動作**	停止	停止
0.5 秒信号出力	**動作**	動作	停止

4-2. ウォッチドッグタイマ

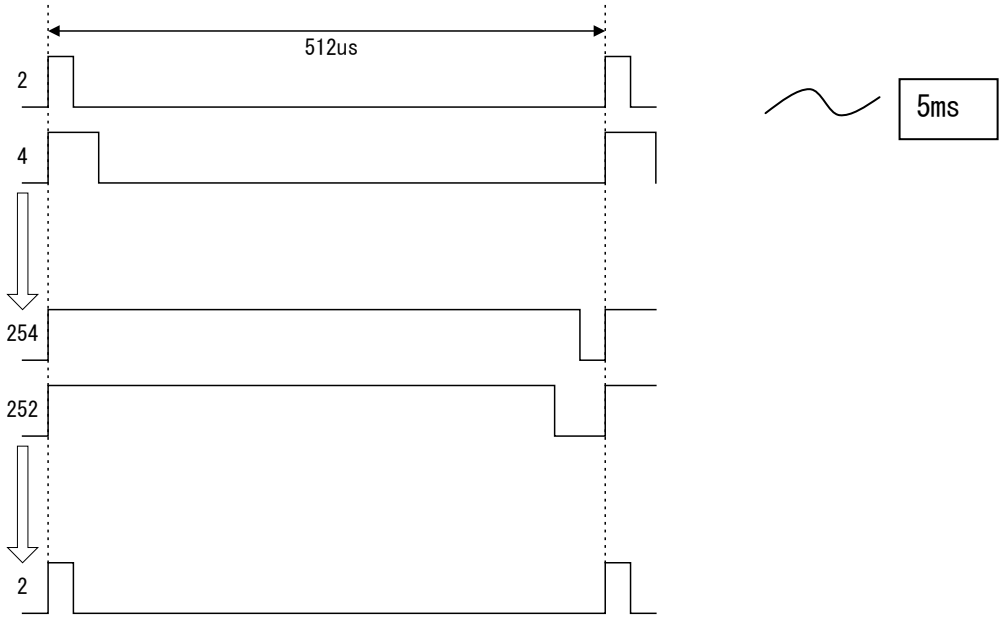
ウォッチドッグタイマは NORMAL モード中および SLOW モード中動作します（暴走検出でリセット発生するように設定)。NORMAL モードでは WDTT=11 に設定し、SLOW モードでは WDTT=00 に設定し、それらの検出時間以内にタイマをクリアします

4-3. 低電圧検出

電圧検出回路は供給電源の低下を検出して、低電圧検出リセットを発生します。

4-4. 8 ビット PWM 出力

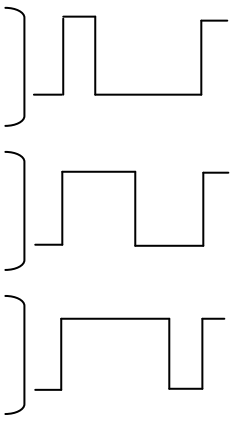
周期 512us で"H"幅 4us(設定値=0x02)から 508us(設定値=0xFE)の信号を、5ms 毎に変化させて PWM00 端子から出力します。これを繰り返します。



4-5. 8 ビット PPG 出力

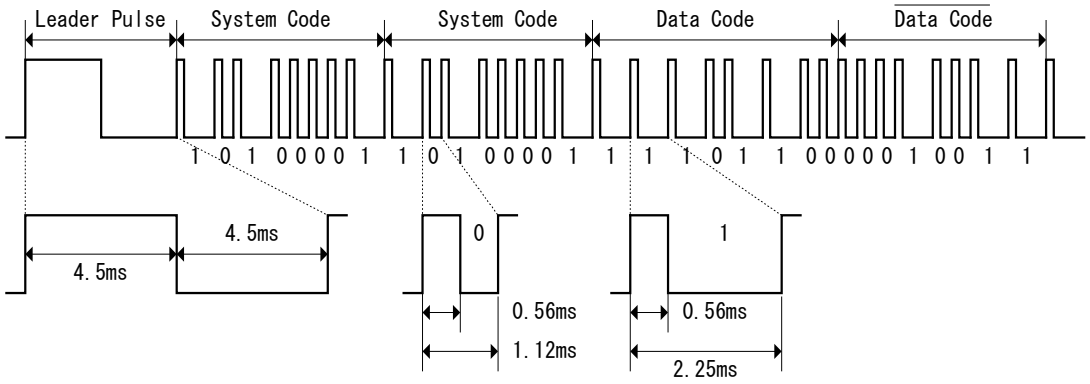
デューティと周波数を変化させた波形を 1 波形につき 1s 毎に変化させて PPG01 端子から出力します。これを繰り返します。他励式ブザーを接続するとブザー音となります。

ステップ	周波数	デューティ	出力時間
1	2441Hz	25%	1s
2	1221Hz	25%	1s
3	610Hz	25%	1s
4	2441Hz	50%	1s
5	1221Hz	50%	1s
6	610Hz	50%	1s
7	2441Hz	75%	1s
8	1221Hz	75%	1s
9	610Hz	75%	1s



4-6. 16 ビット PPG 出力

リモコン信号（赤外線リモコン送信用 IC：TC9243AFG)波形を出力します。200ms 毎に同一波形を出力します。出力データは 0x85,0x85,0x37,0xC8 とします（下位ビットから出力します)。



4-7. UART 出力

UART2 から 8 ビット UART モードで任意文字列(ASCII コード)を出力します。

項目	値
ボーレート	9600bps
パリティ	無し
ストップビット	1 ビット
出力文字列(A)	“TOSHIBA.”
文字列間隔	(A)を 1 回出力したら、100ms の間隔をあけて 10 回繰り返す。

4-8. SIO 出力

SIO1 からクロック同期 SIO 信号を出力します。20ms 周期で 2 バイト出力します。データは 0x0000 から 0xFFFF まで 1 ずつインクリメントした値とします。

4-9. I2C 出力

10ms 周期で、I2C で 1 バイト出力します。クロック周波数は n=5 設定値とし、ACK 受信できないため、アクノリッジクロックは出力しません。出力データは 0xA3 とします。

4-10. 10 ビット A/D 入力

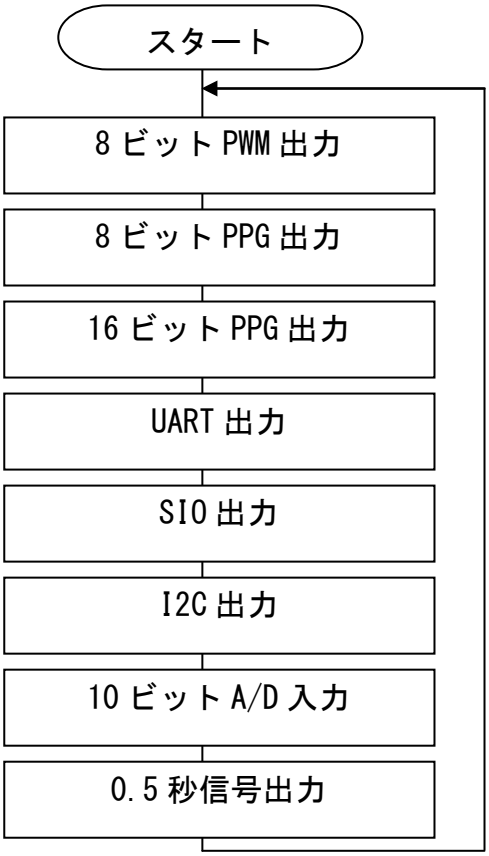
AIN8 から 2ms 毎にデータを取り込み、上位 8 ビットを PB ポートに出力します。これを 10 回繰り返します。

4-11. 0.5 秒信号出力

0.5s 周期で反転する信号を P84 端子から 10 回出力します。NORMAL モードと SLOW モードで動作します。

4-12. 機能シーケンス

各機能は、下図のように一機能毎に連続的に実行され、これを繰り返します。



5. ソフトウェア内容

5-1. メインプログラム : [main.c]

```
void main(void)
{
    WDCTR = 0x27;                           /* Start WDT */
    gMode = 0;
    test = 0x00;
    fSlowNormalChangeCheck = FALSE;         /* The Flag for Slow and Normal Mode change.
                                               FALSE --> TRUE represent Normal changes to Slow
                                               TURE --> FALSE represent Slow changes to Normal */

    fStopModeCheck = FALSE;                 /* The Flag for Normal/Stop Mode change */

    fSlowModeCheck = FALSE;                 /* The Flag for current Mode,
                                               FALSE --> Normal; TRUE --> Slow */

    for (;;)
    {

        WDCDR = 0x4E;                       /* Clear WDT counter */
        if ((fSlowNormalChangeCheck == TRUE) && (fSlowModeCheck == FALSE))
        {
            fSlowModeCheck = TRUE ;
            gMode = 7;
            _asm(" CLR (_WDCTR).5");        /* Stop WDT counter */
            _asm(" SET (_SYSCR2).4 ");
            _asm(" NOP ");
            _asm(" NOP ");
            _asm(" NOP ");
            _asm(" CLR (_SYSCR2).6 ");      /* Change to Slow Mode */
            WDCTR = 0x01;                   /* Select 2^11/fs,reset output */
            WDCDR = 0xB1;                   /* CLear the 8-bit up counter */
            _asm(" SET (_WDCTR).5");        /* Start WDT counter */
        }
        if (fStopModeCheck == TRUE)
        {
            _asm(" SET (_SYSCR1).7 ");      /* Change to Stop Mode */
        }

        switch (gMode)
        {
            case 0:
                sample_tc00_pwm();
                gMode++;
                break;

            case 1:
                sample_tc01_ppg();
                gMode++;
                break;

            case 2:
                sample_ta0_ppg();
                gMode++;
                break;

            case 3:
                sample_uart();
                gMode++;
                break;

            case 4:
                sample_sio();
                gMode++;
                break;
```

```
64
65                  case 5:
66                      sample_sbi();
67                      gMode++;
68                      break;
69
70                  case 6:
71                      sample_adc();
72                      gMode++;
73                      break;
74
75                  case 7:
76                      sample_rtc();
77                      if (fSlowModeCheck == FALSE)
78                      {
79                          gMode ++;
80                      }
81                      break;
82
83                  default:
84                      gMode = 0x00;
85                      break;
86              }
87          }
88      }
```

メインプログラムは SLOW/STOP の起動処理と gMode による各内蔵機能の振り分けで構成されています。
各内蔵機能は一処理が終わると gMode がインクリメントされることで、次の機能に移行します。
単独の機能を動作させたい場合には、17 行目に下記のような処理を追加し gMode を固定値に設定します。

gmode=n;　ここで n は 1 から 7 の値です

5-2. 8 ビット PWM 出力 : [timer8_pwm.c]

5-2-1. 制御処理

```
1   void sample_tc00_pwm(void)
2   {
3       fTC00Check = FALSE;
4       fTC00Cycle = FALSE;
5       gTC00Cnt = 5;
6
7       TC00Init();
8       TC00Start();
9       while (fTC00Check == FALSE)
10      {
11          WDCDR = 0x4E;       /* Clear WDT counter */
12      }
13      TC00Stop();
14  }
```

[7 行目 : TC00 初期化] ➡5-2-2

[8 行目 : TC00 開始] ➡ 5-2-3

[13 行目 : TC00 停止] ➡ 5-2-4

5-2-2. 初期化処理

```
1   void TC00Init(void)
2   {
3       _asm(" SET (_P7FC).0 ");
4       _asm(" SET (_P7CR).0 ");    /* Set P70 as the 8-bit PWM output port */
5       POFFCR0 = 0x10;             /* Set TC001EN = 1 */
6       _DI();
7       EIRH = EIRH | 0x10;         /* Enable INTTC00 */
8       _EI();
9       T00MOD = 0xA2;              /* Set 8-bit PWM mode and fcgck/2^4 */
10      T00PWM = 0x02;              /* 4us x 2/(2^4/fcgck) */
11  }
```

[5 行目 : TC00 クロック許可]

低消費電力レジスタ 0 制御

POFFCR0 (0x0F74)	7	6	5	4	3	2	1	0
Bit Symbol	-	-	TC023EN	TC001EN	-	-	TCA1EN	TCA0EN
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
リセット後	0	0	0	0	0	0	0	0

TC023EN	タイマカウンタ 02 03 制御	0 1	Disable Enable
TC001EN	タイマカウンタ 00, 01 制御	0 1	Disable Enable
TCA1EN	タイマカウンタ A1 制御	0 1	Disable Enable
TCA0EN	タイマカウンタ A0 制御	0 1	Disable Enable

[9 行目 ： 8 ビット PWM モード/クロック設定]

タイマカウンタ 00 モードレジスタ

T00MOD (0x002A)

Bit Symbol	7 TFF0	6 DBE0	5	4 TCK0	3	2 EIN0	1	0 TCM0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
リセット後	1	1	0	0	0	0	0	0

TFF0	タイマ F/F0 の制御	0 1	クリア セット
DBE0	ダブルバッファ制御	0 1	ダブルバッファ無効 ダブルバッファ有効
TCK0	動作クロック選択		NORMAL 1/2, IDLE1/2 モード: SYSCR1<DV9CK> = "0" / SYSCR1<DV9CK> = "1" / SLOW1/2, SLEEP1 モード
		000	fcgck/2^{11} / fs/2^4 / fs/2^4
		001	fcgck/2^{10} / fs/2^3 / fs/2^3
		010	fcgck/2^8 / fcgck/2^8 / -
		011	fcgck/2^6 / fcgck/2^6 / -
		100	fcgck/2^4 / fcgck/2^4 / -
		101	fcgck/2^2 / fcgck/2^2 / -
		110	fcgck/2 / fcgck/2 / -
		111	fcgck / fcgck / fs/2^2
EIN0	外部ソースクロック使用選択	0 1	ソースクロックは内部クロック ソースクロックは外部クロック (TC00 端子の立ち下がりエッジ)
TCM0	動作モード選択	00 01 10 11	8 ビットタイマ / イベントカウンタモード 8 ビットタイマ / イベントカウンタモード 8 ビットパルス幅変調出力 (PWM) モード 8 ビットプログラマブルパルスジェネレート (PPG) モード

[10 行目 ： タイマ値設定]

5-2-3. タイマ開始処理

```
1  void TC00Start(void)
2  {
3      _asm(" SET (_T001CR).0 ");
4  }
```

[3 行目 ： タイマスタート]

タイマカウンタ 01 制御レジスタ

T001CR (0x002C)

Bit Symbol	7 -	6 -	5 -	4 -	3 OUTAND	2 TCAS	1 T01RUN	0 T00RUN
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W
リセット後	0	0	0	0	0	0	0	0

OUTAND	タイマ 00,01 の出力制御	0 1	タイマ 00 の出力を $\overline{PWM0}$, $\overline{PPG0}$、タイマ 01 の出力を $\overline{PWM1}$, $\overline{PPG1}$ 端子から出力する。 タイマ 00,01 の出力の論理積パルスを $\overline{PWM1}$, $\overline{PPG1}$ 端子から出力する。
TCAS	タイマ 00,01 のカスケード接続制御	0 1	タイマ 00,01 を独立して使用する (8 ビットモード) タイマ 00,01 をカスケード接続する (16 ビットモード)
T01RUN	タイマ 01 の制御 タイマ 00/01 の制御 (16 ビットモード)	0 1	ストップ&カウンタクリア スタート
T00RUN	タイマ 00 の制御	0 1	ストップ&カウンタクリア スタート

5-2-4. タイマ停止処理

```
1  void TC00Stop(void)
2  {
3      T001CR = 0x00;
4  }
```

[3 行目 ： タイマ停止]

5-2-5. 割り込み処理

```
1  void _interrupt IntTC00(void)
2  {
3      static UINT8_t i = 0;
4
5      i++;
6      if (i >= gTC00Cnt)
7      {
8          i = 0;
9          if (fTC00Cycle == FALSE)
10         {
11             T00PWM = T00PWM + 2;
12             if (T00PWM >= 254)
13             {
14                 fTC00Cycle = ~fTC00Cycle;
15             }
16         }
17         else
18         {
19             T00PWM = T00PWM - 2;
20             if (T00PWM <= 0)
21             {
22                 fTC00Cycle = ~fTC00Cycle;
23                 fTC00Check = TRUE;
24             }
25         }
26     }
27 }
```

[11,19 行目 ： 次回タイマ値セット]

5-3. 8 ビット PPG 出力 : [timer8_ppg.c]

5-3-1. 制御処理

```
 1  void sample_tc01_ppg(void)
 2  {
 3      UINT8_t i = 0;
 4
 5      fTC01Check = FALSE;
 6      fTC01Cycle = FALSE;
 7      sRatioMode = 0;
 8      sPPGCnt = 2441;
 9
10      TC01Init();
11      TC01Start();
12      while (fTC01Check == FALSE)
13      {
14          WDCDR = 0x4E;                       /* Clear WDT counter */
15          if (fTC01Cycle == TRUE)
16          {
17              fTC01Cycle = FALSE;
18              sRatioMode++;
19              if (sRatioMode >= 9)
20              {
21                  sRatioMode = 0;
22                  i++;
23                  if (i >= sRunTimeCnt)
24                  {
25                      fTC01Check = TRUE;
26                  }
27              }
28              switch (sRatioMode)
29              {
30                  case 0:
31                      T01REG = 0x1A;
32                      T01PWM = 0x06;
33                      sPPGCnt = 2441;
34                      break;                  /* 2441HZ,25% duty */
35
36                  case 1:
37                      T01REG = 0x33;
38                      T01PWM = 0x0D;
39                      sPPGCnt = 1221;
40                      break;                  /* 1221HZ,25% duty */
41
42                  case 2:
43                      T01REG = 0x66;
44                      T01PWM = 0x1A;
45                      sPPGCnt = 610;          /* 610HZ,25% duty */
46                      break;
47
48                  case 3:
49                      T01REG = 0x1A;
50                      T01PWM = 0x0C;
51                      sPPGCnt = 2441;
52                      break;                  /* 2441HZ,50% duty */
53
54                  case 4:
55                      T01REG = 0x33;
56                      T01PWM = 0x1A;
57                      sPPGCnt = 1221;
58                      break;                  /* 1221HZ,50% duty */
59
60                  case 5:
61                      T01REG = 0x66;
62                      T01PWM = 0x34;
63                      sPPGCnt = 610;          /* 610HZ,50% duty */
```

```
64                          break;
65
66                      case 6:
67                          T01REG = 0x1A;
68                          T01PWM = 0x12;
69                          sPPGCnt = 2441;
70                          break;                  /* 2441HZ,75% duty */
71
72                      case 7:
73                          T01REG = 0x33;
74                          T01PWM = 0x27;
75                          sPPGCnt = 1221;
76                          break;                  /* 1221HZ,75% duty */
77
78                      case 8:
79                          T01REG = 0x66;
80                          T01PWM = 0x4E;
81                          sPPGCnt = 610;          /* 610HZ,75% duty */
82                          break;
83
84                      default:
85                          break;
86                  }
87              }
88          }
89          TC01Stop();
90      }
```

[10 行目 ： タイマ初期化] ➡ 5-3-2

[11 行目 ： タイマ開始] ➡ 5-3-3

[31, 32 行目 ： タイマ値セット (2441HZ, 25%デューティ)]

[37, 38 行目 ： タイマ値セット (1221HZ, 25%デューティ)]

[43, 44 行目 ： タイマ値セット (610HZ, 25%デューティ)]

[49, 50 行目 ： タイマ値セット (2441HZ, 50%デューティ)]

[55, 56 行目 ： タイマ値セット (1221HZ, 50%デューティ)]

[61, 62 行目 ： タイマ値セット (610HZ, 50%デューティ)]

[67, 68 行目 ： タイマ値セット (2441HZ, 75%デューティ)]

[73, 74 行目 ： タイマ値セット (1221HZ, 75%デューティ)]

[79, 80 行目 ： タイマ値セット (610HZ, 75%デューティ)]

[89 行目 ： タイマ停止] ➡ 5-3-4

5-3-2. 初期化処理

```
1   void TC01Init(void)
2   {
3       _asm(" SET (_P7FC).1 ");            /* Set P71 as the 8-bit PPG output port */
4       _asm(" SET (_P7CR).1 ");
5       POFFCR0 = 0x10;
6       _DI();
7       EIRH = EIRH | 0x20;
8       _EI();
9       T01MOD = 0x9B;                      /* Set 8-bit PPG mode and fcgck/2^6 */
10      T01REG = 0x1A;                       /* (1/2441)/(2^6/fcgck) */
11      T01PWM = 0x06;
12  }
```

[5 行目 ： TC01 クロック許可]

低消費電力レジスタ 0 制御

POFFCR0 (0x0F74)

Bit Symbol	7	6	5	4	3	2	1	0
	-	-	TC023EN	TC001EN	-	-	TCA1EN	TCA0EN
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
リセット後	0	0	0	0	0	0	0	0

TC023EN	タイマカウンタ 02 03 制御	0 1	Disable Enable
TC001EN	タイマカウンタ 00, 01 制御	0 1	Disable Enable
TCA1EN	タイマカウンタ A1 制御	0 1	Disable Enable
TCA0EN	タイマカウンタ A0 制御	0 1	Disable Enable

[9 行目 ： 8 ビット PPG モード／クロックセット]

タイマカウンタ 01 モードレジスタ

T01MOD (0x002B)

Bit Symbol	7	6	5	4	3	2	1	0
	TFF1	DBE1	TCK1			EIN1	TCM1	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
リセット後	1	1	0	0	0	0	0	0

TFF1	タイマ F/F1 の制御	0 1	クリア セット		
DBE1	ダブルバッファ制御	0 1	ダブルバッファ無効 ダブルバッファ有効		
TCK1	動作クロック選択		NORMAL 1/2, IDLE1/2 モード		SLOW1/2, SLEEP1 モード
			SYSCR1<DV9CK> = "0"	SYSCR1<DV9CK> = "1"	
		000	fcgck/2^{11}	fs/2^4	fs/2^4
		001	fcgck/2^{10}	fs/2^3	fs/2^3
		010	fcgck/2^8	fcgck/2^8	-
		011	fcgck/2^6	fcgck/2^6	-
		100	fcgck/2^4	fcgck/2^4	-
		101	fcgck/2^2	fcgck/2^2	-
		110	fcgck/2	fcgck/2	-
		111	fcgck	fcgck	fs/2^2
EIN1	外部ソースクロック使用選択	0 1	ソースクロックは内部クロック ソースクロックは外部クロック (TC01 端子の立ち下がりエッジ)		
TCM1	動作モード選択		T001CR<TCAS>="0" (8 ビットモード)	T001CR<TCAS>="1" (16 ビットモード)	
		00	8 ビットタイマ / イベントカウンタモード	16 ビットタイマ / イベントカウンタモード	
		01	8 ビットタイマ / イベントカウンタモード	16 ビットタイマ / イベントカウンタモード	
		10	8 ビットパルス幅変調出力 (PWM) モード	12 ビットパルス幅変調出力 (PWM) モード	
		11	8 ビットプログラマブルパルスジェネレート (PPG) モード	16 ビットプログラマブルパルスジェネレート (PPG) モード	

[10, 11 行目 ： タイマ初期値セット]

5-3-3. タイマ開始処理

```
1  void TC01Start(void)
2  {
3      _asm(" SET (_T001CR).1 ");
4  }
```

[3 行目 ： タイマ開始]

タイマカウンタ 01 制御レジスタ

T001CR (0x002C)	7	6	5	4	3	2	1	0
Bit Symbol	-	-	-	-	OUTAND	TCAS	T01RUN	T00RUN
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W
リセット後	0	0	0	0	0	0	0	0

OUTAND	タイマ 00,01 の出力制御	0 1	タイマ 00 の出力を $\overline{PWM0}$, $\overline{PPG0}$、タイマ 01 の出力を $\overline{PWM1}$, $\overline{PPG1}$ 端子から出力する。 タイマ 00,01 の出力の論理積パルスを $\overline{PWM1}$, $\overline{PPG1}$ 端子から出力する。
TCAS	タイマ 00,01 のカスケード接続制御	0 1	タイマ 00,01 を独立して使用する (8 ビットモード) タイマ 00,01 をカスケード接続する (16 ビットモード)
T01RUN	タイマ 01 の制御 タイマ 00/01 の制御 (16 ビットモード)	0 1	ストップ&カウンタクリア スタート
T00RUN	タイマ 00 の制御	0 1	ストップ&カウンタクリア スタート

5-3-4. タイマ停止処理

```
1  void TC01Stop(void)
2  {
3      T001CR = 0x00;
4  }
```

[3 行目 : TC01 停止]

5-3-5. 割り込み処理

```
1  void _interrupt IntTC01(void)
2  {
3      static UINT16_t i = 0;
4
5      i++;
6      if (i >= sPPGCnt)
7      {
8          i = 0;
9          fTC01Cycle = TRUE;
10     }
11 }
```

5-4. 16 ビット PPG 出力 : [timer16_ppg.c]

5-4-1. 制御処理

```
 1  void sample_ta0_ppg(void)
 2  {
 3      UINT8_t i = 0;
 4
 5      f200msCheck = FALSE;
 6      fTA0Check = FALSE;
 7      gCheckCnt = 200;
 8
 9      TCA0Init();
10      TCA0Start();
11      TBTInit();
12      TBTStart();
13      while (fTA0Check == FALSE)
14      {
15          WDCDR = 0x4E;                   /* Clear WDT counter */
16          if (f200msCheck == TRUE)
17          {
18              f200msCheck = FALSE;
19              i++;
20              if (i < sRunTimeCnt)
21              {
22                  TCA0Start();
23              }
24              else
25              {
26                  fTA0Check = TRUE;
27              }
28          }
29      }
30      TCA0Stop();
31      TBTStop();
32  }
```

[9 行目 : TCA0 初期化] ➡ 5-4-2

[10 行目 : TCA0 開始] ➡ 5-4-3

[30 行目 : TCA0 停止] ➡ 5-4-4

5-4-2. 初期処理

```
 1  void TCA0Init(void)
 2  {
 3      _asm(" SET (_P7CR).2 ");          /* P72 as the 16-bit PPG output port */
 4      _asm(" SET (_P7FC).2 ");
 5      POFFCR0 = 0x01;                    /* Set TCA0EN = 1 */
 6      _DI();
 7      EIRH = EIRH | 0x40;
 8      _EI();
 9      TA0MOD = 0x13;                     /* fcgck/2^2 */
10      TA0CR = 0xC0;
11  }
```

[5 行目 : TCA0 クロック許可]

低消費電力レジスタ 0 制御

POFFCR0		7	6	5	4	3	2	1	0
(0x0F74)	Bit Symbol	-	-	TC023EN	TC001EN	-	-	TCA1EN	TCA0EN
	Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
	リセット後	0	0	0	0	0	0	0	0

TC023EN	タイマカウンタ 02 03 制御	0 1	Disable Enable
TC001EN	タイマカウンタ 00, 01 制御	0 1	Disable Enable
TCA1EN	タイマカウンタ A1 制御	0 1	Disable Enable
TCA0EN	タイマカウンタ A0 制御	0 1	Disable Enable

［ 9 行目 ： 16 ビット PPG モード／クロック設定 ］

タイマカウンタ A0 モードレジスタ

TA0MOD		7	6	5	4	3	2	1	0
(0x0031)	Bit Symbol	TA0DBE	TA0TED	TA0MCAP TA0METT	TA0CK		TA0M		
	Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
	リセット後	1	0	0	0	0	0	0	0

TA0DBE	ダブルバッファ制御	0 1	ダブルバッファ無効 ダブルバッファ有効		
TA0TED	外部トリガ入力の選択	0 1	立ち上がりエッジ / H レベル 立下りエッジ / L レベル		
TA0MCAP	パルス幅測定モード制御	0 1	両エッジキャプチャ 片エッジキャプチャ		
TA0METT	外部トリガタイマモード制御	0 1	トリガスタート トリガスタート＆ストップ		
TA0CK	タイマカウンタ 1 のソースクロックの選択		NORMAL 1/2, IDLE1/2 モード		SLOW1/2, SLEEP1 モード
			SYSCR1<DV9CK> ="0"	SYSCR1<DV9CK> ="1"	
		00	$fc_{gck}/2^{10}$	$fs/2^3$	$fs/2^3$
		01	$fc_{gck}/2^5$	$fc_{gck}/2^6$	-
		10	$fc_{gck}/2^2$	$fc_{gck}/2^2$	-
		11	$fc_{gck}/2$	$fc_{gck}/2$	-
TA0M	タイマカウンタ 1 の動作モードの選択	000	タイマモード		
		001	タイマモード		
		010	イベントカウンタモード		
		011	PPG 出力モード (ソフトウェアスタート)		
		100	外部トリガタイマモード		
		101	ウィンドウモード		
		110	パルス幅測定モード		
		111	Reserved		

[10 行目 ： タイマ条件設定]

タイマカウンタ A0 制御レジスタ

TA0CR (0x0032)

Bit Symbol	7 TA0OVE	6 TA0TFF	5	4 TA0NC	3 -	2 -	1 TA0CAP TA0MPPG	0 TA0S
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W
リセット後	0	1	0	0	0	0	0	0

TA0OVE	オーバーフロー割り込み制御	0	カウンタオーバーフロー時、INTTA0 割り込み要求を発生させない	
		1	カウンタオーバーフロー時、INTTA0 割り込み要求を発生させる	
TA0TFF	タイマ F/F 制御	0	クリア	
		1	セット	
TA0NC	ノイズキャンセラサンプリングの間隔設定		NORMAL 1/2, IDLE1/2 モード	SLOW1/2, SLEEP1 モード
		00	ノイズキャンセラなし	ノイズキャンセラなし
		01	fcgck/2	-
		10	fcgck/2^2	-
		11	fcgck/2^8	fs/2
TA0ACAP	自動キャプチャ機能	0	自動キャプチャディセーブル	
		1	自動キャプチャイネーブル	
TA0MPPG	PPG 出力制御	0	連続	
		1	単発	
TA0S	タイマカウンタ A のスタート制御	0	ストップ＆カウンタクリア	
		1	スタート	

5-4-3. タイマ開始処理

```
1  void TCA0Start(void)
2  {
3      TA0DRAL = 0x28;
4      TA0DRAH = 0x23;
5      TA0DRBL = 0x94;
6      TA0DRBH = 0x11;
7      _asm(" SET (_TA0CR).0 ");
8  }
```

[3-6 行目 ： タイマ初期値セット]

[7 行目 ： タイマ開始]

5-4-4. タイマ停止処理

```
1  void TCA0Stop(void)
2  {
3      TA0CR = 0xC0;
4  }
```

[3 行目 ： タイマ停止]

5-4-5. 割り込み処理

```
 1  void _interrupt IntTCA0(void)
 2  {
 3      static UINT8_t i = 0;
 4
 5      if (i < 33)
 6      {
 7          if (((OUTPUTDATA >> i) & 0x01) == 0)
 8          {
 9              TA0DRAL = 0x60;
10              TA0DRAH = 0x04;
11              TA0DRBL = 0x30;
12              TA0DRBH = 0x02;
13              i++;
14          }
15          else
16          {
17              i++;
18              TA0DRAL = 0xCA;
19              TA0DRAH = 0x08;
20              TA0DRBL = 0x30;
21              TA0DRBH = 0x02;
22          }
23      }
24      else
25      {
26          i = 0;
27          TCA0Stop();
28      }
29  }
```

[9-12,18-21 行目 ： 次回タイマ値セット]

5-5. UART 出力 : [uart.c]

5-5-1. 制御処理

```
 1  void sample_uart(void)
 2  {
 3      UINT8_t i = 0;
 4
 5      fUARTTXCheck = FALSE;
 6      f100msCheck = FALSE;
 7      gCheckCnt = 100;
 8
 9      UARTTXInit();
10      TBTInit();
11      UARTTXStart();
12      TBTStart();
13      UARTTXTrans();
14      while (fUARTTXCheck == FALSE)
15      {
16          WDCDR = 0x4E;                    /* Clear WDT counter */
17          if (f100msCheck == TRUE)
18          {
19              f100msCheck = FALSE;
20              i++;
21              if (i < sRunTimeCnt)
22              {
23                  UARTTXStart();
24                  UARTTXTrans();
25              }
26              else
27              {
28                  fUARTTXCheck = TRUE;
29              }
30          }
31      }
32      UARTTXStop();
33      TBTStop();
34  }
```

[9 行目 : UART 初期化] ➡5-5-2

[11 行目 : UART 開始] ➡ 5-5-3

[13 行目 : 先頭出力文字セット] ➡ 5-5-4

[32 行目 : UART 停止] ➡ 5-5-5

5-5-2. 初期化処理

```
 1  void UARTTXInit(void)
 2  {
 3      POFFCR1 = 0x04;                  /* Set UART2EN = 1 */
 4      _asm(" SET (_P9CR).3 ");
 5      _asm(" SET (_P9FC).3 ");         /* Set P93 as the UART2 output */
 6
 7      P9DR = 0x08;
 8      P9OUTCR = 0x08;
 9      P9PU = 0x08;
10
11      _DI();
12      EIRD = EIRD | 0x08;              /* Enable INTTXD2 */
13      _EI();
14      UART2CR1 = 0x00;                 /* 1 stop bit,No parity */
15      UART2CR2 = 0x00;
16      UART2DR = 0x19;                  /* 9600bps */
17  }
```

[3 行目 : UART クロック許可]

低消費電力レジスタ 1 制御

POFFCR1 (0x0F75)

Bit Symbol	7	6	5	4	3	2	1	0
	-	-	-	SBI0EN	-	UART2EN	UART1EN	UART0EN
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
リセット後	0	0	0	0	0	0	0	0

SBI0EN	I2C0 制御	0 1	Disable Enable
UART2EN	UART2 制御	0 1	Disable Enable
UART1EN	UART1 制御	0 1	Disable Enable
UART0EN	UART0 制御	0 1	Disable Enable

［ 14 行目 ： ストップビット／パリティセット ］

UART0 制御レジスタ 1

*UART2CR1 は UART0CR1 と同じ構成です。

UART0CR1 (0x001A)

Bit Symbol	7	6	5	4	3	2	1	0
	TXE	RXE	STOPBT	EVEN	PE	IRDASEL	BRG	-
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R
リセット後	0	0	0	0	0	0	0	0

TXE	送信動作	0: 1:	ディセーブル イネーブル
RXE	受信動作	0: 1:	ディセーブル イネーブル
STOPBT	送信ストップビット長	0: 1:	1 ビット 2 ビット
EVEN	パリティ選択	0: 1:	奇数パリティ 偶数パリティ
PE	パリティ付加	0: 1:	パリティなし パリティ付加
IRDASEL	TXD 端子の出力選択	0: 1:	UART 出力 IrDA 出力
BRG	転送ベースクロックの選択		SYSCR2<SYSCK>="0" 時 / SYSCR2<SYSCK>="1" 時
		0:	fcgck / fs
		1:	TCA0 出力

［ 15 行目 ： ノイズ除去設定 ］

UART0 制御レジスタ 2

*UART2CR1 は UART0CR1 と同じ構成です。

UART0CR2 (0x001B)

Bit Symbol	7	6	5	4	3	2	1	0
	-	-	RTSEL			RXDNC		STOPBR
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W
リセット後	0	0	0	0	0	0	0	0

RTSEL	RT クロック数の選択		転送フレームの奇数ビット	転送フレームの偶数ビット
		000:	16 クロック	16 クロック
		001:	16 クロック	17 クロック
		010:	15 クロック	15 クロック
		011:	15 クロック	16 クロック
		100:	17 クロック	17 クロック
		101:	Reserved	
		11*:	Reserved	
RXDNC	RXD 入力のノイズ除去時間の選択（確実にノイズ除去されるパルスの時間）	00: 01: 10: 11:	ノイズ除去なし 1 × (UART0DR+1) / (転送ベースクロックの周波数) [s] 2 × (UART0DR+1) / (転送ベースクロックの周波数) [s] 4 × (UART0DR+1) / (転送ベースクロックの周波数) [s]	
STOPBR	受信ストップビット長	0: 1:	1 ビット 2 ビット	

［ 16 行目 ： ボーレート設定 ］

表 16-6 転送ボーレートに対する UART0DR、UART0CR2<RTSEL> の設定値 (fcgck = 8 ~ 1MHz, UART0CR2 <RXDNC> = 0y00)

基準ボーレート [baud]	レジスタ	動作周波数									
		8MHz	7.3728 MHz	6.144 MHz	6MHz	5MHz	4.9152 MHz	4.19MHz	4MHz	2MHz	1MHz
128000	UART0DR	0x03	-	0x02	0x02	-	-	0x01	0x01	0x00	-
	RTSEL	0y011	-	0y000	0y011	-	-	0y001	0y011	0y011	-
	誤差	(+0.81%)	-	(0%)	(+0.81%)	-	-	(-0.80%)	(+0.81%)	(+0.81%)	-
115200	UART0DR	-	0x03	-	0x02	-	-	-	0x01	0x00	-
	RTSEL	-	0y000	-	0y100	-	-	-	0y100	0y100	-
	誤差	-	(0%)	-	(+2.12%)	-	-	-	(+2.12%)	(+2.12%)	-
76800	UART0DR	0x06	0x05	0x04	0x04	0x03	0x03	-	0x02	-	-
	RTSEL	0y010	0y000	0y000	0y011	0y000	0y000	-	0y100	-	-
	誤差	(-0.79%)	(0%)	(0%)	(+0.81%)	(+1.73%)	(0%)	-	(+2.12%)	-	-
62500	UART0DR	0x07	0x06	0x05	0x05	0x04	0x04	0x03	0x03	0x01	0x00
	RTSEL	0y000	0y100	0y001	0y000	0y000	0y000	0y100	0y000	0y000	0y000
	誤差	(0%)	(-0.87%)	(-0.70%)	(0%)	(0%)	(-1.73%)	(-1.41%)	(0%)	(0%)	(0%)
57600	UART0DR	0x08	0x07	-	0x06	-	0x04	-	0x03	0x01	0x00
	RTSEL	0y011	0y000	-	0y010	-	0y100	-	0y100	0y100	0y100
	誤差	(-0.44%)	(0%)		(-0.79%)		(+0.39%)		(+2.12%)	(+2.12%)	(+2.12%)
38400	UART0DR	0x0C	0x0B	0x09	0x09	0x07	0x07	0x06	0x06	0x02	-
	RTSEL	0y000	0y000	0y000	0y011	0y000	0y000	0y011	0y010	0y100	-
	誤差	(+0.16%)	(0%)	(0%)	(+0.81%)	(+1.73%)	(0%)	(+0.57%)	(-0.79%)	(+2.12%)	-
19200	UART0DR	0x19	0x17	0x13	0x12	0x0F	0x0F	0x0C	0x0C	0x06	0x02
	RTSEL	0y000	0y000	0y000	0y001	0y000	0y000	0y000	0y000	0y010	0y100
	誤差	(+0.16%)	(0%)	(0%)	(-0.32%)	(+1.73%)	(0%)	(+1.74%)	(+0.16%)	(-0.79%)	(+2.12%)
9600	UART0DR	0x33	0x2F	0x27	0x26	0x20	0x1F	0x19	0x19	0x0C	0x06
	RTSEL	0y000	0y000	0y000	0y000	0y000	0y000	0y100	0y000	0y000	0y010
	誤差	(+0.16%)	(0%)	(0%)	(+0.16%)	(-1.38%)	(0%)	(-1.25%)	(+0.16%)	(+0.16%)	(-0.79%)
4800	UART0DR	0x67	0x5F	0x4F	0x4D	0x40	0x3F	0x35	0x33	0x19	0x0C
	RTSEL	0y000	0y000	0y000	0y000	0y000	0y000	0y000	0y000	0y000	0y000
	誤差	(+0.16%)	(0%)	(0%)	(+0.16%)	(+0.16%)	(0%)	(+1.03%)	(+0.16%)	(+0.16%)	(+0.16%)
2400	UART0DR	0xCF	0xBF	0x9F	0x9B	0x81	0x7F	0x6B	0x67	0x33	0x19
	RTSEL	0y000	0y000	0y000	0y000	0y000	0y000	0y000	0y000	0y000	0y000
	誤差	(+0.16%)	(0%)	(0%)	(+0.16%)	(+0.16%)	(0%)	(+1.03%)	(+0.16%)	(+0.16%)	(+0.16%)
1200	UART0DR	-	-	-	-	0xFF	0xFF	0xD9	0xCF	0x67	0x33
	RTSEL	-	-	-	-	0y000	0y000	0y000	0y000	0y000	0y000
	誤差	-	-	-	-	(+1.73%)	(+0%)	(+0.11%)	(+0.16%)	(+0.16%)	(+0.16%)

5-5-3. UART 開始処理

```
1  void UARTTXStart(void)
2  {
3      _asm(" SET (_UART2CR1).7 ");
4  }
```

[3 行目 : UART 転送開始]

5-5-4. 先頭出力文字セット

```
1  void UARTTXTrans(void)
2  {
3      TD2BUF = OutputData[0];
4  }
```

[3 行目 : 送信バッファへの先頭文字セット]

5-5-5. UART 停止処理

```
1  void UARTTXStop(void)
2  {
3      UART2CR1 = 0x00;
4  }
```

[3 行目 : UART 停止]

5-5-6. 割り込み処理

```
1  void _interrupt IntTXD2(void)
2  {
3      static UINT8_t i = 0;
4
5      i++;
6      if (i <= 7)
7      {
8          TD2BUF = OutputData[i];
9      }
10     else
11     {
12         UARTTXStop();
13         i = 0;
14     }
15 }
```

[8 行目 : 次回転送データセット]

[12 行目 : UART 停止]

5-6. SIO 出力 : [sio.c]

5-6-1. 制御処理

```
 1  void sample_sio(void)
 2  {
 3      fSIOTXCheck = FALSE;
 4      f20msCheck = FALSE;
 5      gCheckCnt = 20;
 6      cSIOTXData.word = 0x00;
 7
 8      TBTInit();
 9      TBTStart();
10      SIOTXInit();
11      SIOTXTrans();
12      SIOTXStart();
13      while (fSIOTXCheck == FALSE)
14      {
15          WDCDR = 0x4E;                       /* Clear WDT counter */
16          if (f20msCheck == TRUE)
17          {
18              f20msCheck = FALSE;
19              cSIOTXData.word++;
20              if (cSIOTXData.word == 0x00)
21              {
22                  fSIOTXCheck = TRUE;
23              }
24              else
25              {
26                  SIOTXTrans();
27                  SIOTXStart();
28              }
29          }
30      }
31      SIOTXStop();
32      TBTStop();
33  }
```

[10 行目 : SIO 初期化] ➡ 5-6-2

[11 行目 : 先頭転送データセット] ➡ 5-6-3

[12 行目 : SIO 開始] ➡ 5-6-4

[31 行目 : SIO 停止] ➡ 5-6-5

5-6-2. 初期化処理

```
 1  void SIOTXInit(void)
 2  {
 3      POFFCR2 = 0x02;             /* Enable SIO1 */
 4      _asm(" SET (_P9CR).0 ");    /* Set P90 as the SIO data output port */
 5      _asm(" SET (_P9FC).0 ");
 6      _asm(" SET (_P9CR).2 ");    /* Set P92 as the SCLK1 output pin */
 7      _asm(" SET (_P9FC).2 ");
 8      P9DR = 0x05;                /* P90 and P92 output the H level first */
 9      P9OUTCR = 0x05;             /* P90 and P92 open drain output */
10      P9PU = 0x05;                /* The build-in pull-up resistor connected */
11      SERSEL = 0x08;
12
13      _DI();
14      EIRE = EIRE | 0x40;         /* Enable the   INTSIO1 */
15      _EI();
16
17      SIO1CR = 0x01;              /* LSB first, 8-bit transmit mode ,SIOCKS = fc/2^9
18                                     Data transmission at the Falling edge */
19  }
```

[3 行目 : SIO クロック許可]

低消費電力レジスタ 2 制御

POFFCR2 (0x0F76)

	7	6	5	4	3	2	1	0
Bit Symbol	-	-	RTCEN	-	-	-	SIO1EN	SIO0EN
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
リセット後	0	0	0	0	0	0	0	0

RTCEN	RTC 制御	0 1	Disable Enable
SIO1EN	SIO1 制御	0 1	Disable Enable
SIO0EN	SIO0 制御	0 1	Disable Enable

[17 行目 ： SIO 転送モードセット]

シリアルインタフェース制御レジスタ

*SIO1CR は SIO0CR と同じ構成です。

SIO0CR (0x001F)

	7	6	5	4	3	2	1	0
Bit Symbol	SIOEDG	SIOCKS			SIODIR	SIOS	SIOM	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
リセット後	0	0	0	0	0	0	0	0

SIOEDG	転送エッジ選択	0 1	0: 立上りエッジでデータの受信、立下りエッジでデータの送信 1: 立上りエッジでデータの送信、立下りエッジでデータの受信	
SIOCKS	シリアルクロックの選択		NORMAL1/2,IDLE1/2 モード	SLOW1/2、SLEEP1 モード
		000	$f_{cgck}/2^9$	-
		001	$f_{cgck}/2^6$	-
		010	$f_{cgck}/2^5$	-
		011	$f_{cgck}/2^4$	-
		100	$f_{cgck}/2^3$	-
		101	$f_{cgck}/2^2$	-
		110	$f_{cgck}/2$	$f_s/2^3$
		111	外部クロック入力	
SIODIR	転送フォーマット (MSB/LSB) の選択	0 1	LSB ファースト (ビット 0 から転送) MSB ファースト (ビット 7 から転送)	
SIOS	転送動作の開始 / 終了指示	0 1	0: 動作終了 (予約停止) 1: 動作開始	
SIOM	転送モードの選択&動作	00	動作停止 (強制停止)	
		01	8 ビット送信モード	
		10	8 ビット受信モード	
		11	8 ビット送受信モード	

5-6-3. SIO 開始処理

```
1  void SIOTXStart(void)
2  {
3      _asm(" SET (_SIO1CR).2 ");          /* Start SIO function */
4  }
```

[3 行目 ： SIO 開始]

5-6-4. SIO 先頭データセット処理

```
1  void SIOTXTrans(void)
2  {
3      SIO1BUF = cSIOTXDataL;              /* Write data into the buffer */
4  }
```

[3 行目 ： 送信バッファへ先頭データをセット]

5-6-5. SIO 停止処理

```
1  void SIOTXStop(void)
2  {
3      SIO1CR = 0x01;                    /* Stop SIO function */
4  }
```

[3 行目 ： SIO 停止]

5-6-6. 割り込み処理

```
1  void _interrupt IntSIO1(void)
2  {
3      static UINT8_t i = 0;
4
5      i++;
6      if (i == 1)
7      {
8          while ((SIO1SR & 0x04) == 0x04);      /* TBFL = 0 */
9          SIO1BUF = cSIOTXDataH;
10     }
11     else if (i == 2)
12     {
13         SIOTXStop();
14     }
15     else
16     {
17         i = 0;
18     }
19 }
```

[8 行目 ： 送信バッファフルの確認]

シリアルインタフェースステータスレジスタ

SIO0SR (0x0020)	7	6	5	4	3	2	1	0
Bit Symbol	SIOF	SEF	OERR	REND	UERR	TBFL	-	-
Read/Write	R	R	R	R	R	R	R	R
リセット後	0	0	0	0	0	0	0	0

SIOF	シリアル転送動作状態モニタ	0 1	転送中でない 転送中
SEF	シフト動作状態モニタ	0 1	シフト動作中でない シフト動作中
OERR	受信オーバーランエラーフラグ	0 1	オーバーランエラー無し オーバーランエラーが少なくとも 1 回は発生した
REND	受信完了フラグ	0 1	前回受信データ読み出し後、データを受信していない 少なくとも 1 回のデータ受信が行われた
UERR	送信アンダーランエラーフラグ	0 1	送信アンダーランエラー無し 送信アンダーランエラーが少なくとも 1 回は発生した
TBFL	送信バッファフルフラグ	0 1	送信バッファは空 送信バッファに未送信データが格納されている

[12 行目 ： 次の出力データセット]

[13 行目 ： SIO 停止]

5-7. I2C 出力 : [sbi.c]

5-7-1. 制御処理

```
 1  void sample_sbi(void)
 2  {
 3      UINT8_t i = 0;
 4      fSBITXCheck = FALSE;
 5      f10msCheck = FALSE;
 6      gCheckCnt = 10;
 7
 8      SBIDeviceInit();
 9      SBITXInit();
10      SBITXStart();
11      TBTInit();
12      TBTStart();
13      while (fSBITXCheck == FALSE)
14      {
15          WDCDR = 0x4E;                       /* Clear WDT counter */
16          if (f10msCheck == TRUE)
17          {
18              f10msCheck = FALSE;
19              i++;
20              if (i < sRunTimeCnt)
21              {
22                  SBITXStart();
23              }
24              else
25              {
26                  fSBITXCheck = TRUE;
27              }
28          }
29      }
30      SBITXStop();
31      TBTStop();
32  }
```

[8 行目 : I2C(SBI)初期化] ➡ 5-7-2

[9 行目 : I2C ポート／割り込み初期化] ➡ 5-7-3

[10 行目 : I2C 開始] ➡ 5-7-4

[30 行目 : I2C 停止] ➡ 5-7-5

5-7-2. 初期化処理

```
1  void SBIDeviceInit(void)
2  {
3      POFFCR1 = 0x10;
4      while (P2PRD == 0x0C);
5      SBI0CR2 = 0x18;                  /* SBI mode */
6      SBI0CR1 = 0x15;
7      I2C0AR = 0x00;
8      SBI0CR2 = 0x18;                  /* Slave mode */
9  }
```

[3 行目 : SBI クロック許可]

低消費電力レジスタ 1 制御

POFFCR1 (0x0F75)

Bit Symbol	7	6	5	4	3	2	1	0
	-	-	-	SBI0EN	-	UART2EN	UART1EN	UART0EN
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
リセット後	0	0	0	0	0	0	0	0

SBI0EN	I2C0 制御	0 1	Disable Enable
UART2EN	UART2 制御	0 1	Disable Enable
UART1EN	UART1 制御	0 1	Disable Enable
UART0EN	UART0 制御	0 1	Disable Enable

[5 行目 : SBI モードセット]

シリアルバスインタフェース制御レジスタ 2

SBI0CR2 (0x0023)

Bit Symbol	7	6	5	4	3	2	1	0
	MST	TRX	BB	PIN	SBIM	-	SWRST	
Read/Write	W	W	W	W	W	R	W	
リセット後	0	0	0	1	0	0	0	

MST	マスタ / スレーブの選択	0: スレーブ 1: マスタ
TRX	送信 / 受信の選択	0: レシーバ 1: トランスミッタ
BB	スタート / ストップコンディションの発生	0: ストップコンディション発生 (MST, TRX, PIN が "1" のとき) 1: スタートコンディション発生 (MST, TRX, PIN が "1" のとき)
PIN	割り込みサービス要求の解除	0: – ("0" にクリアすることはできません) 1: 割り込みサービス要求の解除
SBIM	シリアルバスインタフェース動作モードレジスタ	0: ポートモード 1: シリアルバスインタフェースモード
SWRST	ソフトウエアリセット開始ビット	"10"、"01" の順に値を書き込むとソフトウエアリセットが発生

［ 6 行目 ： クロック設定 ］

シリアルバスインタフェース制御レジスタ 1

SBI0CR1 (0x0022)

	7	6	5	4	3	2	1	0
Bit Symbol	BC			ACK	NOACK	SCK		
Read/Write	R/W			R/W	R/W	R/W		
リセット後	0	0	0	0	0	0	0	0

		BC	ACK = 0 のとき		ACK = 1 のとき	
			データ転送のクロック数	データビット数	データ転送のクロック数	データビット数
BC	データビット数の選択	000:	8	8	9	8
		001:	1	1	2	1
		010:	2	2	3	2
		011:	3	3	4	3
		100:	4	4	5	4
		101:	5	5	6	5
		110:	6	6	7	6
		111:	7	7	8	7

		ACK	マスタモード	スレーブモード
ACK	アクノリッジのためのクロック発生 / カウントの選択	0:	アクノリッジのためのクロックを発生せず、データ転送終了で、INTSBI 割り込み要求を発生する。(非アクノリッジメントモード)	データ転送終了で INTSBI 割り込み要求を発生する。(非アクノリッジメントモード)
		1:	データ転送終了後、アクノリッジのためのクロックを発生し、INTSBI 割り込み要求を発生する。(アクノリッジメントモード)	データ転送終了、アクノリッジのためのクロックをカウントした後、INTSBI 割り込み要求を発生する。(アクノリッジメントモード)

		NOACK	マスタモード	スレーブモード
NOACK	スレーブアドレス一致検出、ゼネラルコール検出の許可 / 不許可選択	0:	Don't Care	スレーブアドレス一致検出、ゼネラルコール検出を許可する。
		1:	Don't Care	スレーブアドレス一致検出、ゼネラルコール検出を禁止する。

		SCK	t_{HIGH}(m/fcgck)	t_{LOW}(n/fcgck)	fscl@fcgck=8MHz
			m	n	
SCK	マスタ時のシリアルクロックの HIGH 時間、LOW 時間の選択 スレーブ時の SCL 端子解放までの時間選択	000:	9	12	381KHz
		001:	11	14	320KHz
		010:	15	18	242KHz
		011:	23	26	163KHz
		100:	39	42	99KHz
		101:	71	74	55KHz
		110:	135	138	29KHz
		111:	263	266	15KHz

［ 7 行目 ： I2C アドレス設定 ］

I^2C バスアドレスレジスタ

I2C0AR (0x0024)

	7	6	5	4	3	2	1	0
Bit Symbol	SA							ALS
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
リセット後	0	0	0	0	0	0	0	0

SA	スレーブアドレスの設定	スレーブモード時のスレーブアドレス
ALS	通信フォーマットの選択	0: I^2C バスモード 1: フリーデータフォーマット

5-7-3. I2C ポート／割り込み初期化

```
1  void SBITXInit(void)
2  {
3        _DI();
4        EIRH = EIRH | 0x80;
5        _EI();
6        _asm(" SET (_P2CR).3 ");
7        _asm(" SET (_P2FC).3 ");          /* Set P23 as the SBI data output port */
8        _asm(" SET (_P2CR).4 ");
9        _asm(" SET (_P2FC).4 ");          /* Set P24 as the SBI Clock output port */
10       P2DR = 0x18;
11 }
```

5-7-4. I2C(SBI)開始処理

```
1  void SBITXStart(void)
2  {
3        UINT8_t temp;
4
5        temp = cSlaveAddr + cSBI_WRITE;
6        SBI0CR1 = 0x15;
7        while ((SBI0SR2 & 0x20) == 0x20);
8        SBI0DBR = temp;
9        SBI0CR2 = 0xF8;
10 }
```

[7 行目 ： バスフリー確認]

シリアルバスインタフェースステータスレジスタ 2

SBI0SR2 (0x0023)

	7	6	5	4	3	2	1	0
Bit Symbol	MST	TRX	BB	PIN	AL	AAS	AD0	LRB
Read/Write	R	R	R	R	R	R	R	R
リセット後	0	0	0	1	0	0	0	*

MST	マスタ / スレーブ選択状態モニタ	0: スレーブ 1: マスタ
TRX	トランスミッタ / レシーバ選択状態モニタ	0: レシーバ 1: トランスミッタ
BB	バス状態モニタ	0: バスフリー 1: バスビジー
PIN	割り込みサービス要求状態モニタ	0: 割り込みサービス要求中 1: 割り込みサービス要求解除中
AL	アービトレーションロスト検出モニタ	0:− 1: アービトレーションロスト検出
AAS	スレーブアドレス一致検出モニタ	0:− 1: スレーブアドレス一致またはゼネラルコール検出
AD0	ゼネラルコール検出モニタ	0:− 1: ゼネラルコール検出
LRB	最終受信ビットモニタ	0: 最終受信ビット "0" 1: 最終受信ビット "1"

[8 行目 ： 最初のデータ(アドレス)セット]

[9 行目 ： I2C 転送開始]

シリアルバスインタフェース制御レジスタ 2

SBI0CR2 (0x0023)

	7	6	5	4	3	2	1	0
Bit Symbol	MST	TRX	BB	PIN	SBIM	-	SWRST	
Read/Write	W	W	W	W	W	R	W	
リセット後	0	0	0	1	0	0	0	

MST	マスタ / スレーブの選択	0: スレーブ 1: マスタ
TRX	送信 / 受信の選択	0: レシーバ 1: トランスミッタ
BB	スタート / ストップコンディションの発生	0: ストップコンディション発生 (MST, TRX, PIN が "1" のとき) 1: スタートコンディション発生 (MST, TRX, PIN が "1" のとき)
PIN	割り込みサービス要求の解除	0: – ("0" にクリアすることはできません) 1: 割り込みサービス要求の解除
SBIM	シリアルバスインタフェース動作モードレジスタ	0: ポートモード 1: シリアルバスインタフェースモード
SWRST	ソフトウエアリセット開始ビット	"10" 、"01" の順に値を書き込むとソフトウエアリセットが発生

5-7-5. SBI 停止処理

```
1  void SBITXStop(void)
2  {
3      SBI0CR2 = 0xD8;
4      while ((SBI0SR2 & 0x20 ) == 0x20);
5  }
```

[3 行目 ： SBI 停止]

[4 行目 ： バスフリー確認]

5-7-6. 割り込み処理

```
1  void _interrupt IntSBI0(void)
2  {
3      static UINT8_t i = 0;
4
5      i++;
6      if (i == 1)
7      {
8          if ((SBI0SR2 & 0x01) == 0x00)
9          {
10             SBI0DBR = cSBITXData;
11         }
12     }
13     else
14     {
15         i = 0;
16         SBITXStop();
17         SBI0CR2 = 0x1A;
18         SBI0CR2 = 0x19;
19     }
20 }
```

[8 行目 ： 最終ビット確認]

[10 行目 ： 次回データセット]

[16 行目 ： SBI 停止]

[17-18 行目 ： SBI リセット (1 回目 ： SWRST=10、2 回目 ： SWRST=01)]

5-8. 10 ビット A/D 入力 : [adc.c]

5-8-1. 制御処理

```
 1  void sample_adc(void)
 2  {
 3      UINT8_t i = 0;
 4
 5      fADCCheck = FALSE;
 6      f2msCheck = FALSE;
 7      fOnceCheck = FALSE;
 8      gCheckCnt = 2;
 9
10      ADCInit();
11      ADCStart();
12      TBTInit();
13      TBTStart();
14      while (fADCCheck == FALSE)
15      {
16          WDCDR = 0x4E;                       /* Clear WDT counter */
17          if (f2msCheck == FALSE)
18          {
19              if (fOnceCheck == FALSE)
20              {
21                  while ((ADCCR2 & 0x80) == 0x00);
22                  ADCGetData();
23                  fOnceCheck = TRUE;
24              }
25          }
26          else
27          {
28              f2msCheck = FALSE;
29              i++;
30              if (i > sRunTimeCnt)
31              {
32                  fADCCheck   = TRUE;
33              }
34              else
35              {
36                  ADCStart();
37                  fOnceCheck = FALSE;
38              }
39          }
40      }
41      ADCStop();
42      TBTStop();
43  }
```

[10 行目 : ADC 初期化] ➡ 5-8-2

[11 行目 : AD 変換開始] ➡ 5-8-3

[21 行目 : 変換終了確認]

[22 行目 : 変換値取り込み] ➡ 5-8-4

[41 行目 : ADC 停止] ➡ 5-8-5

5-8-2. 初期化処理

```
1   void ADCInit(void)
2   {
3       P5CR = 0x00;
4       _asm(" SET (_P5FC).0 ");                /* P50 as the AIN input port */
5
6       PBCR = 0xff;
7       PBDR = 0xff;
8
9       ADCCR1 = 0x38;                          /* Single mode and AIN8 */
10      ADCCR2 = 0x01;                          /* 78/fcgck */
11  }
```

[9 行目 ： 変換モード／チャンネルセット]

AD コンバータ制御レジスタ 1

ADCCR1 (0x0034)	7	6	5	4	3	2	1	0
Bit Symbol	ADRS	AMD		AINEN	SAIN			
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
リセット後	0	0	0	0	0	0	0	0

ADRS	AD 変換開始	0: — 1: AD 変換開始
AMD	AD 動作モード	00: AD 動作ディセーブル、AD 動作強制停止 01: シングルモード 10: Reserved 11: リピートモード
AINEN	アナログ入力制御	0: アナログ入力ディセーブル 1: アナログ入力イネーブル
SAIN	アナログ入力チャネル選択	0000: AIN0 0001: AIN1 0010: AIN2 0011: AIN3 0100: AIN4 0101: AIN5 0110: AIN6 0111: AIN7 1000: AIN8 1001: AIN9 1010: AIN10 1011: AIN11 1100: AIN12 1101: AIN13 1110: AIN14 1111: AIN15

[10 行目 ： 変換時間セット]

AD コンバータ制御レジスタ 2

ADCCR2 (0x0035)	7	6	5	4	3	2	1	0
Bit Symbol	EOCF	ADBF	-	-	"0"	ACK		
Read/Write	R	R	R	R	W	R/W	R/W	R/W
リセット後	0	0	0	0	0	0	0	0

EOCF	AD 変換終了フラグ	0: 変換前または変換中 1: 変換終了
ADBF	AD 変換 BUSY フラグ	0: AD 変換停止中 1: AD 変換実行中
ACK	AD 変換時間選択 （変換時間例は下記表を参照してください）	000: 39/fcgck 001: 78/fcgck 010: 156/fcgck 011: 312/fcgck 100: 624/fcgck 101: 1248/fcgck 110: Reserved 111: Reserved

5-8-3. ADC 開始処理

```
1  void ADCStart(void)
2  {
3      _asm(" SET (_ADCCR1).7 ");
4  }
```

[3 行目 : AD 変換開始]

5-8-4. ADC データ取り込み処理

```
1  void ADCGetData(void)
2  {
3      UINT8_t result1;
4      UINT8_t result2;
5
6      result1 = ADCDRL;
7      result2 = ADCDRH;
8      result2 = result2 << 6;
9      result1 = result1 >> 2;
10     result2 = result2 + result1;          /* output the top 8 bits in 10 bits */
11     PBDR = result2;
12 }
```

[6-7 行目 : 変換データ取り込み]

5-8-5. ADC 停止処理

```
1  void ADCStop(void)
2  {
3      ADCCR1 = 0x00;
4  }
```

[3 行目 : AD 変換停止]

5-9. 0.5 秒信号出力 : [rtc.c]

5-9-1. 制御処理

```
1  void sample_rtc(void)
2  {
3      fRTCCheck = FALSE;
4      sRTCCnt = 10;
5
6      RTCInit();
7      RTCStart();
8      while (fRTCCheck == FALSE)
9      {
10         WDCDR = 0x4E;                /* Clear WDT counter */
11     }
12     RTCStop();
13 }
```

[6 行目 : RTC 初期化] ➡ 5-9-2

[7 行目 : RTC 開始] ➡ 5-9-3

[12 行目 : RTC 停止] ➡ 5-9-4

5-9-2. 初期化処理

```
1  void RTCInit(void)
2  {
3      POFFCR2 = 0x20;
4      _DI();
5      EIRH = EIRH | 0x08;
6      _EI();
7
8      P8CR = 0x10;
9      P8DR = 0x10;                 /* P84 as the output */
10
11     RTCCR = 0x02;                /* 0.5s cycle */
12 }
```

[3 行目 : RTC クロック許可]

低消費電力レジスタ 2 制御

POFFCR2 (0x0F76)	7	6	5	4	3	2	1	0
Bit Symbol	-	-	RTCEN	-	-	-	SIO1EN	SIO0EN
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
リセット後	0	0	0	0	0	0	0	0

RTCEN	RTC 制御	0 1	Disable Enable
SIO1EN	SIO1 制御	0 1	Disable Enable
SIO0EN	SIO0 制御	0 1	Disable Enable

[11 行目 ： RTC 周期セット]

時計専用タイマ制御レジスタ

RTCCR (0x0FC8)

	7	6	5	4	3	2	1	0
Bit Symbol	-	-	-	-	RTCSEL			RTCRUN
Read/Write	R	R	R	R	R/W			R/W
リセット後	0	0	0	0	0	0	0	0

RTCSEL	割り込み発生周期選択	000 : 2^{15}/fs (1.000 [s] @fs=32.768kHz) 001 : 2^{14}/fs (0.500 [s] @fs=32.768kHz) 010 : 2^{13}/fs (0.250 [s] @fs=32.768kHz) 011 : 2^{12}/fs (125.0 [ms] @fs=32.768kHz) 100 : 2^{11}/fs (62.50 [ms] @fs=32.768kHz) 101 : 2^{10}/fs (31.25 [ms] @fs=32.768kHz) 110 : 2^{9}/fs (15.62 [ms] @fs=32.768kHz) 111 : 2^{8}/fs (7.81 [ms] @fs=32.768kHz)
RTCRUN	時計専用タイマ動作の許可／禁止	0 : 禁止 1 : 許可

5-9-3. タイマ開始処理

```
1  void RTCStart(void)
2  {
3      RTCCR = 0x03;
4  }
```

[3 行目 ： タイマ開始]

5-9-4. タイマ停止処理

```
1  void RTCStop(void)
2  {
3      RTCCR = 0x02;
4  }
```

[3 行目 ： タイマ停止]

5-9-5. 割り込み処理

```
1  void _interrupt IntRTC(void)
2  {
3      static UINT8_t i = 0;
4
5      i++;
6      if (i >= sRTCCnt)
7      {
8          fRTCCheck = TRUE;
9          i = 0;
10     }
11     else
12     {
13         P84 = ~P84;
14     }
15 }
```