8　タイマ割り込みを使ってみよう

割り込み（Interrupt）とは，言葉の意味の通り，ある作業中に割り込むことである．マイコンにおいてはとても重要な機能の一つである．例えば，インスタントカップ麺にお湯を入れて 3 分間待ってから食べることを想像してみよう．お湯を入れてカップ麺ができるまでの 3 分間，時計の針だけを見つめ続けて，他には何にもせずに待ち続ける人はほとんどいないだろう．たいていの人は，お湯を入れてタイマを 3 分に設定し，タイマのベルが鳴るまでは，テレビを見たり，本を読んだりと自分が好きなことをして過ごすと思う．そして，ベルが鳴ったら，今していたことを中断し，ラーメンを食べ始める．この例のような，ある作業中にベルが鳴ったら他の作業を開始することを割り込み処理という．割り込み処理は，マイコンを効率よく使用するためのプログラミングで欠かせない存在である．

OpenCM 内では，実世界のタイマと同じ役割をするタイマ割り込み機能が内蔵されている．ここでは，サンプルを通してタイマの基本的な使い方を学ぶ．[ファイル]-[スケッチの例]-[05.Interrupt]から d_Timer_Interrupt_LED を選択して開いてみよう．

以下は、d_Timer_Interrupt_LED のソースコードである。

```
#define LED_RATE 100000    // in microseconds; should give 0.5Hz toggles

HardwareTimer Timer(1);

void setup() {
  // Set up the LED to blink
  pinMode(BOARD_LED_PIN, OUTPUT);

  // Pause the timer while we're configuring it
  Timer.pause();

  // Set up period
  Timer.setPeriod(LED_RATE); // in microseconds

  // Set up an interrupt on channel 1
  Timer.setMode(TIMER_CH1, TIMER_OUTPUT_COMPARE);
  Timer.setCompare(TIMER_CH1, 1);  // Interrupt 1 count after each update
  Timer.attachInterrupt(TIMER_CH1, handler_led);

  // Refresh the timer's count, prescale, and overflow
  Timer.refresh();

  // Start the timer counting
  Timer.resume();
}

void loop() {
  // Nothing! It's all in the handler_led() interrupt:
}

```

```
41  void handler_led(void) {
42    toggleLED();
43  }
```

まず，13 行目の HardwareTimer Timer(1);は，HardwareTimer というクラスを利用するという宣言である．この宣言によって Timer というインスタンスが生成される．Timer(1)の 1 は，OpenCM に 4 個用意されているタイマのうち，1 番のタイマを Timer に割り当てるという意味である．プログラミングの初心者または苦手な人は，ひとまずタイマを使うためのおまじないと思っても構わない．

次に setup()内であるが，19～34 行目でタイマの設定を行っている．これまでの例で setup()内でピンを設定していたのと同じである．13 行目で生成された Timer の様々なメソッドを呼び出すことで，タイマの設定が行える．このサンプルでは Timer は 1 番のタイマ（タイマ 1）に割り当てているので，以下のメソッドは全てタイマ 1 に関する命令である．

タイマの設定の詳細を説明する前に，OpenCM のタイマにおけるチャンネルについて説明しておく．チャンネルとは，タイマの動きに応じてアクションを起こす際の基準のことである．OpenCM ではタイマ 1 つに 4 つのチャンネルが存在し，タイマ 1 つで 4 つのアクションを起こすことができる．

以下に，setup()内で行われているタイマ設定部分をまとめる．

Timer.pause();

タイマを停止する．設定を行うにあたっては，タイマを停止しておく．

Timer.setPeriod(LED_RATE);

タイマの時間をセットする．単位はマイクロ秒である．

LED_RATE は 11 行目で#define LED_RATE 100000 と宣言されており，100000 マイクロ秒，すなわち 0.1 秒とセットしたことになる．この場合，タイマは 0.1 秒まで計測を行い，0.1 秒が過ぎるとまた 0 からタイマがスタートする．Robotis Inc.によると，タイマにセットされる時間間隔として最低でも 8000 以上，すなわち 0.008 秒以上を推奨している．

Timer.setMode(TIMER_CH1, TIMER_OUTPUT_COMPARE);

タイマのモードを設定する．

ここではタイマ 1 のチャンネル1を TIMER_OUTPUT_COMPARE モードに設定するという意味である．TIMER_OUTPUT_COMPARE モードとは，Timer.setPeriod で設定した割り込み間隔を 65536 等分した時間ごとに値が増加するカウンタカウントが増加し，そのカウントが setCompare で設定した値になった時に割り込みを発生させるモードである．

Timer.setCompare(TIMER_CH1, 1);

割り込みを発生させるタイマのカウント数（目標値）を設定する．

このサンプルでは 1 と設定したため，カウントが1のタイミングでチャンネル 1 による割り込みが発生する．もし 5 と設定したら 5 までカウントしてから割り込みが発生する．特殊なことをするのでなければ，基本的には1にセットしておけばよい．

Timer.attachInterrupt(TIMER_CH1, handler_led);

チャンネル 1 による割り込みが発生したら handler_led という関数を呼び出す．

Timer.refresh();

タイマのカウント数を 0 に戻す．

以前のカウント値が残っている可能性があるため，0 にリセットしてから利用する．

Timer.resume();

Timer.pause()で止めたタイマの動作を再開させる．

以上の設定は複雑に見えるが，ほとんどはおまじないのようなものであり，Timer.setPeriod()関数でタイマの時間間隔（割り込み発生周期）を設定することと，Timer.attachInterrupt()関数で割り込み発生時に呼び出す関数を設定することを理解していれば，問題なく自由に使用できるはずである．

void handler_led()関数内の 42 行目，toggleLED();という命令は，D14（BOARD_LED_PIN）の状態を反転する命令であり，HIGH であれば LOW に，LOW であれば HIGH に変更される．そのため，D14 に繋がっている OpenCM 上の LED が点灯していれば消灯し，消灯していれば点灯する．

最後に，タイマの時間間隔とカウントの違いについて，補足説明しておく．まずタイマの時間間隔とは，どのくらいの間隔で割り込みを発生させるか（割り込み発生周期）を決めるもので，設定すると設定した時間間隔でタイマが動く．カウントとは，設定した時間間隔を 65535 等分した時間ごとに値が1ずつ増えるもので，タイマがスタートしてから設定した時間間隔が経過すると 65535 になる．例えば，カウントが 4 のときに割り込みを発生するように設定すると，カウントが 4 になるたびに割り込みが発生する．ここで注意が必要なのは，カウントは割り込みのタイミングを決めるためのものであり，カウント数を変更しても割り込み周期は変わらないことである．また，1 カウントに必要な時間は，割り込み発生周期を 65536 で割った時間となるため，割り込み発生周期が異なれば 1 カウントの時間の長さも異なることにも注意が必要である．ただし，単に一定間隔で割り込みを発生したければ，カウントの値にはあまり意味がなく，1 にしておけば良い．

9　キーボードから入力を受けてみよう

「4. OpenCM で「Hello World!!」してみよう」では，SerialUSB.print 命令を用いて OpenCM から PC に USB ケーブルを通して文字列を送信してみた．ここでは，その逆方向の通信である，PC 側のキーボードから入力された文字または文字列を OpenCM で受信するプログラムを紹介する．[ファイル]-[スケッチの例]-[03.Communication]から c_SerialUSB_Echo_Interrupt を選択して開いてみよう．

プログラムをコンパイルしダウンロードできたら，シリアルモニタを開き，図 8.1 に示すように入力窓に文字列を入力して送信ボタンを押すと，OpenCM に文字列が送信される．サンプルプログラムがちゃんと動いていれば，図 8.2 のような結果が得られるはずである．

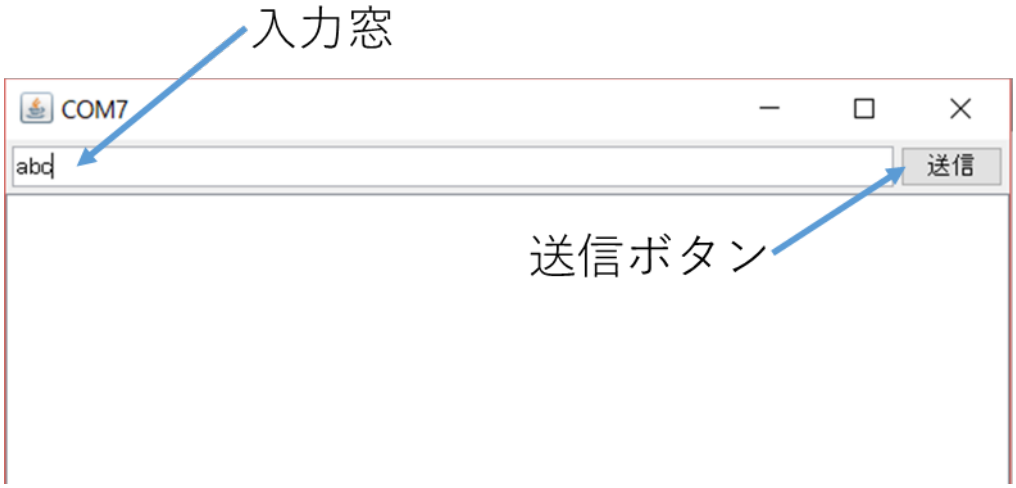

図 8.1 シリアルモニタへの文字列の入力

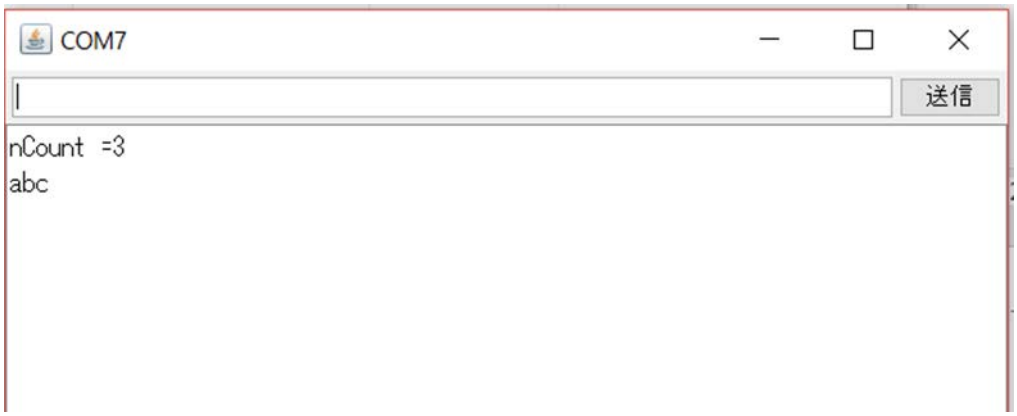

図 8.2 サンプルプログラムの結果

以下に c_SerialUSB_Echo_Interrupt のソースコードを示す.

```
15  void setup(){
16    //You can attach your serialUSB interrupt
17    //or, also detach the interrupt by detachInterrupt(void) method
18    SerialUSB.attachInterrupt(usbInterrupt);
19    pinMode(BOARD_LED_PIN, OUTPUT);  //toggleLED_Pin_Out
20  }
21
22  void usbInterrupt(byte* buffer, byte nCount){
23    SerialUSB.print("nCount =");
24    SerialUSB.println(nCount);
25    for(unsigned int i=0; i < nCount;i++)  //printf_SerialUSB_Buffer[N]_receive_Da
26      SerialUSB.print((char)buffer[i]);
27    SerialUSB.println("");
28  }
29
30  void loop(){
31    toggleLED();
32    delay(100);
33
34  }
```

setup()内の 18 行目, SerialUSB.attachInterrupt(usbInterrupt);は usb からの受信が検出された場合に割り込みを発生し, usbInterrupt()関数を呼び出すように設定する命令である.

loop()内では, 0.1 秒(100 ミリ秒)ごとに OpenCM 上の LED の点灯状態を反転させている. PC から USB ケーブルを通して何かを受信したら, 18 行目で設定した通り, void usbInterrupt(byte* buffer, byte nCount)関数が呼び出される. このとき, 受信内容が引数としてこの関数に渡される. 具体的には配列として文字列が渡され, buffer[] に受信した文字列が入り, nCount には受信した文字列の大きさ(文字数)が入る. usbInterrupt()関数内では, OpenCM が受信した文字列とその長さを, SerialUSB.print()関数と SerialUSB.println()関数を用い, USB を通してPCに送信している.

（応用課題）

SerialUSB.print((char)buffer[i]);を SerialUSB.print((char)(buffer[i]+1));に変更して実行結果を確認してみよう．さらに SerialUSB.print(buffer[i]);に変更して実行結果を確認してみよう．実行結果の確認する際には，下の ASCII テーブルを参考してほしい．

ASCII value	Character	ASCII value	Character	ASCII value	Character
032	(space)	064	@	096	
033	!	065	A	097	a
034	"	066	B	098	b
035	#	067	C	099	c
036	$	068	D	100	d
037	%	069	E	101	e
038	&	070	F	102	f
039	'	071	G	103	g
040	(	072	H	104	h
041	)	073	I	105	i
042	*	074	J	106	j
043	+	075	K	107	k
044	,	076	L	108	l
045	-	077	M	109	m
046	.	078	N	110	n
047	/	079	O	111	o
048	0	080	P	112	p
049	1	081	Q	113	q
050	2	082	R	114	r
051	3	083	S	115	s
052	4	084	T	116	t
053	5	085	U	117	u
054	6	086	V	118	v
055	7	087	W	119	w
056	8	088	X	120	x
057	9	089	Y	121	y
058	:	090	Z	122	z
059	;	091	[	123	{
060	<	092	\	124	|
061	=	093	]	125	}
062	>	094	^	126	~
063	?	095	_	127	⌂