GenTable.exe　：　CRC 計算用の C 言語用テーブル（配列）作成コマンド

2018 年 6 月 4 日

動作環境

Windows のコマンドプロンプト

コマンドシンタックス

```
GenTable /{l | r}{7 | 16 | 32},<polynominal>[:W]
```

コマンドスイッチ

l	左送り（MSB から LSB の順番）
r	右送り（LSB から MSB の順番）
7,16,32	CRC ビット数
<polynominal>	CRC 多項式を表す 16 進数
W	ワード配列を出力（7 ビット指定のとき）

解説

C 言語で CRC を計算する場合 1 ビットづつ取り出して計算するのでは効率が悪いので 1 バイトづつ取り出してテーブル（C 言語の配列）の値を使って CRC 値を計算します。このとき使用するテーブルを CRC 多項式を与えて作成するユーティリティコマンドです。C 言語のソース形式文字列を標準出力へ出力します。理論的な根拠は以下の参考文献を参照してください。

以下に CRC-16(CRC-CCITT)多項式＝1+ X^5 + X^12 + X^16 で右送りの場合の例を挙げます。
この場合、以下のようにコマンドプロンプト内で使用します。

```
C:¥>GenTable /r16,8408
/*Generate CRC Table(V1.02)
        (C)Copyright K.Sasaki 1998 - 2002
*/
/* CRC16 lookup table of polynominal 0x8408 to shift right*/
typedef unsigned short u16;
const u16 crc16tab[256]={
/*   0 */ 0x0000,0x1189,0x2312,0x329b,0x4624,0x57ad,0x6536,0x74bf,
/*   8 */ 0x8c48,0x9dc1,0xaf5a,0xbed3,0xca6c,0xdbe5,0xe97e,0xf8f7,
/*  16 */ 0x1081,0x0108,0x3393,0x221a,0x56a5,0x472c,0x75b7,0x643e,
/*  24 */ 0x9cc9,0x8d40,0xbfdb,0xae52,0xdaed,0xcb64,0xf9ff,0xe876,
/*  32 */ 0x2102,0x308b,0x0210,0x1399,0x6726,0x76af,0x4434,0x55bd,
/*  40 */ 0xad4a,0xbcc3,0x8e58,0x9fd1,0xeb6e,0xfae7,0xc87c,0xd9f5,
/*  48 */ 0x3183,0x200a,0x1291,0x0318,0x77a7,0x662e,0x54b5,0x453c,
/*  56 */ 0xbdcb,0xac42,0x9ed9,0x8f50,0xfbef,0xea66,0xd8fd,0xc974,
```

```
/*  64 */ 0x4204,0x538d,0x6116,0x709f,0x0420,0x15a9,0x2732,0x36bb,
/*  72 */ 0xce4c,0xdfc5,0xed5e,0xfcd7,0x8868,0x99e1,0xab7a,0xbaf3,
/*  80 */ 0x5285,0x430c,0x7197,0x601e,0x14a1,0x0528,0x37b3,0x263a,
/*  88 */ 0xdecd,0xcf44,0xfddf,0xec56,0x98e9,0x8960,0xbbfb,0xaa72,
/*  96 */ 0x6306,0x728f,0x4014,0x519d,0x2522,0x34ab,0x0630,0x17b9,
/* 104 */ 0xef4e,0xfec7,0xcc5c,0xddd5,0xa96a,0xb8e3,0x8a78,0x9bf1,
/* 112 */ 0x7387,0x620e,0x5095,0x411c,0x35a3,0x242a,0x16b1,0x0738,
/* 120 */ 0xffcf,0xee46,0xdcdd,0xcd54,0xb9eb,0xa862,0x9af9,0x8b70,
/* 128 */ 0x8408,0x9581,0xa71a,0xb693,0xc22c,0xd3a5,0xe13e,0xf0b7,
/* 136 */ 0x0840,0x19c9,0x2b52,0x3adb,0x4e64,0x5fed,0x6d76,0x7cff,
/* 144 */ 0x9489,0x8500,0xb79b,0xa612,0xd2ad,0xc324,0xf1bf,0xe036,
/* 152 */ 0x18c1,0x0948,0x3bd3,0x2a5a,0x5ee5,0x4f6c,0x7df7,0x6c7e,
/* 160 */ 0xa50a,0xb483,0x8618,0x9791,0xe32e,0xf2a7,0xc03c,0xd1b5,
/* 168 */ 0x2942,0x38cb,0x0a50,0x1bd9,0x6f66,0x7eef,0x4c74,0x5dfd,
/* 176 */ 0xb58b,0xa402,0x9699,0x8710,0xf3af,0xe226,0xd0bd,0xc134,
/* 184 */ 0x39c3,0x284a,0x1ad1,0x0b58,0x7fe7,0x6e6e,0x5cf5,0x4d7c,
/* 192 */ 0xc60c,0xd785,0xe51e,0xf497,0x8028,0x91a1,0xa33a,0xb2b3,
/* 200 */ 0x4a44,0x5bcd,0x6956,0x78df,0x0c60,0x1de9,0x2f72,0x3efb,
/* 208 */ 0xd68d,0xc704,0xf59f,0xe416,0x90a9,0x8120,0xb3bb,0xa232,
/* 216 */ 0x5ac5,0x4b4c,0x79d7,0x685e,0x1ce1,0x0d68,0x3ff3,0x2e7a,
/* 224 */ 0xe70e,0xf687,0xc41c,0xd595,0xa12a,0xb0a3,0x8238,0x93b1,
/* 232 */ 0x6b46,0x7acf,0x4854,0x59dd,0x2d62,0x3ceb,0x0e70,0x1ff9,
/* 240 */ 0xf78f,0xe606,0xd49d,0xc514,0xb1ab,0xa022,0x92b9,0x8330,
/* 248 */ 0x7bc7,0x6a4e,0x58d5,0x495c,0x3de3,0x2c6a,0x1ef1,0x0f78
};
C:¥>
```

出力を画面でなくファイル（crc16.txt）に格納する場合は以下のようにします

```
C:¥>GenTable /r16,8408 > crc16.txt
C:¥>
```

以下によく使われる多項式とそれを表す<polynominal>（16 進数）の例を挙げます。

```
CRC-16: The HDLC(CRC-CCITT)
        1+ X^5 + X^12 + X^16
        1000 0100 0000 1000     (0x8408)        右送り
        0001 0000 0010 0001     (0x1021)        左送り

CRC-16(CRC-ANSI)
        X^16+X^15+X^2+1
        1010 0000 0000 0001     (0xa001)        右送り
        1000 0000 0000 0101     (0x8005)        左送り

CRC-7
        X^7+X^6+X^2+1
        0101 0001               (0x51)          右送り（ビット 6-0 が有効）
        1000 1010               (0x8a)          左送り（ビット 7-1 が有効）
```

CRC-32

X^32+X^26+X^23+X^22+X^16+X^12+X^11+X^10+X^8+X^7+X^5+X^4+X^2+X^1+1
1110 1101 1011 1000 1000 0011 0010 0000 (0xedb88320)　右送り
0000 0100 1100 0001 0001 1101 1011 0111 (0x04c11db7)　左送り

※ここで右送りとはLSB→MSBの順にビットを取り出して計算することで、左送りとはMSB→LSBの順にビットを取り出して計算することです。LSB(Least Significant Bit)はビット0、MSB(Most Significant Bit)はビット7を指します。

計算例

このコマンドで作成される参照テーブル（配列）を使ってCRCを計算するC言語のプログラムを以下に示します。

CRC7の計算例

CRC7（右送り）の配列を以下のコマンドで作成します。

```
C:¥>GenTable /r7,51 > crc7R.h
```

ここで作成されたcrc7R.hをインクルードし、以下の関数CalcCRC7RでCRCを計算します。CRCの初期値には主に(u8)0が使われます。

```
#include <stdio.h>
#include <stdlib.h>
#include "crc7R.h"
typedef unsigned char byte;

u8 CalcCRC7R(u8 crc, int n, byte c[])
{
	int i;
	for (i = 0; i < n; i++) {
		crc = crc7tab[((crc) ^ *c++) & 0xff];
	}
	return (crc);
}
```

CRC7（左送り）の場合の関数を以下に示します。

```
u8 CalcCRC7L(u8 crc, int n, byte c[])
{
	int i;
	for (i = 0; i < n; i++) {
		crc = (crc7tab[((crc) ^ *c++) & 0xff]);
	}
	return (crc);
}
```

このコマンドが生成するC言語の参照配列は以下のようになります。

```
typedef unsigned char u8;
const u8 crc7tab[256]={
```

```
        …
};
```

タイプ u8 が 8 ビットであればこのままでよいのですが、CPU によっては C 言語の char タイプが 16 ビットのものがあります（2000 年ごろ関わった Texas Instruments の TMS320C54x シリーズはこのタイプの CPU だった）。このような場合、char に 1 個の CRC7 を格納する配列では上位 8 ビットが無駄に消費されます。そこで 16 ビットに CRC7 を 2 個格納するとメモリ効率が良くなります。コマンドスイッチに W を指定すると配列は以下のようになり、1 要素（16 ビット）に 2 個の CRC7 を格納します。

```
typedef unsigned short u16;
static u16 crc7tab[256/2]={
        …
};
```

CRC16 の計算例

CRC16（右送り）の配列を以下のコマンドで作成します。

```
C:¥>GenTable /r16,8408 > crc16R.h
```

ここで作成された crc16R.h をインクルードし、以下の関数 CalcCRC16R で CRC を計算します。CRC の初期値には主に(u16)0 が使われます。

```
#include <stdio.h>
#include <stdlib.h>
#include "crc16R.h"
typedef unsigned char byte;

u16 CalcCRC16R(u16 crc, int n, byte c[])
{
        while (--n >= 0){
                crc = (crc >> 8) ^ crc16tab[(byte)(crc ^ *c) & 0xff];
                ++c;
        }
        return crc;
}
```

CRC16（左送り）の場合の関数を以下に示します。

```
u16 CalcCRC16L(u16 crc, int n, byte c[])
{
        while (--n >= 0){
                crc = (crc << 8) ^ crc16tab[(byte)(crc >> (16 - 8)) ^ (*c)];
                ++c;
        }
        return crc;
}
```

CRC32 の計算例

CRC32（右送り）の配列を以下のコマンドで作成します。

```
C:¥>GenTable /r32, edb88320 > crc32R.h
```

ここで作成された crc32R.h をインクルードし、以下の関数 CalcCRC32R で CRC を計算します。CRC の初期値には主に(u32)-1 が使われます。

```
#include <stdio.h>
#include <stdlib.h>
#include "crc32R.h"
typedef unsigned char byte;

u32 CalcCRC32R(u32 crc, int n, byte c[])
{
	while (--n >= 0){
		crc = (crc >> 8) ^ crc32tab[(byte)(crc ^ *c) & 0xff];
		++c;
	}
	return crc;
}
```

IETF（インターネット技術タスクフォース）の RFC(Request for Comments)2083 を参照すると CRC32 の初期値はすべてのビットを１とし、計算結果の１の補数を CRC として使うとありますので、以下のような関数 crc で計算することになります。

```
u32 crc(int n, byte c[])
{
	return CalcCRC32R (0xffffffffL, n, c) ^ 0xffffffffL;
	//または　return ~CalcCRC32R ((u32)-1, n, c);
}
```

CRC32（左送り）の場合の関数を以下に示します。

```
u32 CalcCRC32L(u32 crc, int n, byte c[])
{
	while (--n >= 0){
		crc = (crc << 8) ^ crc32tab[(byte)(crc >> (32 - 8)) ^ (*c)];
		++c;
	}
	return crc;
}
```

参考文献

C 言語で書くアルゴリズム 著者 Andrew Binstock/John Rex、訳者 岩谷宏、発行 ソフトバンク株式会社

C 言語による最新アルゴリズム辞典 著者 奥村晴彦、発行 株式会社 技術評論者

ダウンロード

ダウンロードはこちらから。
http://cmtoy.jp/mytool/crc/gentable.htm