

- [首頁](#)
- [文檔](#)
- [問答](#)
- [知識](#)
- [專題](#)

嵌入式系統



CRC 演算法原理及C 語言實現

[admin](#) @ 2014-03-25 , reply:0

Tags:

推薦 6 分享

概述

摘要：本文從理論上推導出CRC演算法實現原理，給出三種分別適應不同計算機或微控制器硬體環境的C語言程序。讀者更能根據本演算法原理，用不同的語言編寫出獨特風格更加實用的CRC計算程序。1引言



摘要：本文從理論上推導出CRC 演算法實現原理，給出三種分別適應不同計算機或微控制器硬體環境的C 語言程序。讀者更能根據本演算法原理，用不同的語言編寫出獨特風格更加實用的CRC 計算程序。

1 引言

循環冗餘碼CRC 檢驗技術廣泛應用於測控及通信領域。CRC 計算可以靠專用的硬體來實現，但是對於低成本的微控制器系統，在沒有硬體支持下實現CRC 檢驗，關鍵的問題就是如何通過軟體來完成CRC 計算，也就是CRC 演算法的問題。

這裡將提供三種演算法，它們稍有不同，一種適用於程序空間十分苛刻但CRC 計算速度要求不高的微控制器系統，另一種適用於程序空間較大且CRC 計算速度要求較高的計算機或微控制器系統，最後一種是適用於程序空間不太大，且CRC 計算速度又不可以太慢的微控制器系統。

2 CRC 簡介

CRC 校驗的基本思想是利用線性編碼理論，在發送端根據要傳送的k 位二進位碼序列，以一定的規則產生一個校驗用的監督碼（既CRC 碼）r 位，並附在信息後邊，構成一個新的二進位碼序列數共(k+r)位，最後發送出去。在接收端，則根據信息碼和CRC碼之間所遵循的規則進行檢驗，以確定傳送中是否出錯。

16 位的CRC 碼產生的規則是先將要發送的二進位序列數左移16 位（既乘以 2^{16} ）后，再除以一個多項式，最後所得到的餘數既是CRC 碼，如式（2-1）式所示，其中B(X)表示n 位的二進位序列數，G(X)為多項式，Q(X)為整數，R(X)是餘數（既CRC 碼）。

$$\frac{B(X) \cdot 2^{16}}{G(X)} = Q(X) + \frac{R(X)}{G(X)} \quad (2-1)$$

求CRC碼所採用模2 加減運演算法則，既是不帶進位和借位的按位加減，這種加減運算實際上就是邏輯上的異或運算，加法和減法等價，乘法和除法運算與普通代數式的乘除法運算是一樣，符合同樣的規律。生成CRC碼的多項式如下，其中CRC-16 和CRC-CCITT產生16 位的CRC碼，而CRC-32 則產生的是32 位的CRC碼。本文不討論32 位的CRC演算法，有興趣的朋友可以根據本文的思路自己去推導計算方法。

CRC-16：（美國二進位同步系統中採用） $G(X) = X^{16} + X^{15} + X^2 + 1$

CRC-CCITT：（由歐洲CCITT 推薦） $G(X) = X^{16} + X^{12} + X^5 + 1$

CRC-32： $G(X) = X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X^1 + 1$

接收方將接收到的二進位序列數（包括信息碼和CRC 碼）除以多項式，如果餘數為0，則說明傳輸中無錯誤發生，否則說明傳輸有誤，關於其原理這裡不再多述。用軟體計算CRC 碼時，接收方可以將接收到的信息碼求CRC 碼，比較結果和接收到的CRC 碼是否相同。

3 按位計算CRC

對於一個二進位序列數可以表示為式(3-1):

$$B(X) = B_n \cdot 2^n + B_{n-1} \cdot 2^{n-1} + \dots + B_1 \cdot 2 + B_0 \quad (3-1)$$

求此二進位序列數的CRC 碼時，先乘以 2^{16} 后（既左移16 位），再除以多項式 $G(X)$ ，所得的餘數既是所要求的CRC 碼。如式(3-2)所示：

$$\frac{B(X) \cdot 2^{16}}{G(X)} = \frac{B_n \cdot 2^{16}}{G(X)} \cdot 2^n + \frac{B_{n-1} \cdot 2^{16}}{G(X)} \cdot 2^{n-1} + \dots + \frac{B_1 \cdot 2^{16}}{G(X)} \cdot 2 + \frac{B_0 \cdot 2^{16}}{G(X)} \quad (3-2)$$

可以設：

$$\frac{B_n \cdot 2^{16}}{G(X)} = Q_n(X) + \frac{R_n(X)}{G(X)} \quad (3-3)$$

其中 $Q_n(X)$ 為整數， $R_n(X)$ 為16 位二進位餘數。將式(3-3)代入式(3-2)得：

$$\begin{aligned} \frac{B(X) \cdot 2^{16}}{G(X)} &= \left\{ Q_n(X) + \frac{R_n(X)}{G(X)} \right\} \cdot 2^n + \frac{B_{n-1} \cdot 2^{16}}{G(X)} \cdot 2^{n-1} + \dots + \frac{B_1 \cdot 2^{16}}{G(X)} \cdot 2 + \frac{B_0 \cdot 2^{16}}{G(X)} \\ &= Q_n(X) \cdot 2^n + \left\{ \frac{R_n(X) \cdot 2}{G(X)} + \frac{B_{n-1} \cdot 2^{16}}{G(X)} \right\} \cdot 2^{n-1} + \dots + \frac{B_1 \cdot 2^{16}}{G(X)} \cdot 2 + \frac{B_0 \cdot 2^{16}}{G(X)} \end{aligned} \quad (3-4)$$

再設：

$$\frac{R_n(X) \cdot 2}{G(X)} + \frac{B_{n-1} \cdot 2^{16}}{G(X)} = Q_{n-1}(X) + \frac{R_{n-1}(X)}{G(X)} \quad (3-5)$$

其中 $Q_{n-1}(X)$ 為整數， $R_{n-1}(X)$ 為16 位二進位餘數，將式(3-5)代入式(3-4)，如上類推，最後得到：

$$\frac{B(X) \cdot 2^{16}}{G(X)} = Q_n(X) \cdot 2^n + Q_{n-1}(X) \cdot 2^{n-1} + Q_{n-2}(X) \cdot 2^{n-2} + \dots + Q_0(X) + \frac{R_0(X)}{G(X)} \quad (3-6)$$

根據CRC 的定義，很顯然，十六位二進位數 $R_0(X)$ 既是我們要求的CRC 碼。式(3-5)是編程計算CRC 的關鍵，它說明計算本位后的CRC 碼等於上一位CRC 碼乘以2 后除以多項式，所得的餘數再加上本位值除以多項式所得的餘數。由此不難理解下面求CRC 碼的C 語言程序。 $*ptr$ 指向發送緩衝區的首位元組， len 是要發送的總位元組數， $0x1021$ 與多項式有關。

```
unsigned int cal_crc(unsigned char *ptr, unsigned char len) {
    unsigned char i;
    unsigned int crc=0;
    while(len--!=0) {
        for(i=0x80; i!=0; i/=2) {
            if((crc&0x8000)!=0) {crc*=2; crc^=0x1021;} /* 余式CRC 乘以2 再求CRC */
            else crc*=2;
            if((*ptr&i)!=0) crc^=0x1021; /* 再加上本位的CRC */
        }
        ptr++;
    }
    return(crc);
}
```

按位計算CRC 雖然代碼簡單，所佔用的內存比較少，但其最大的缺點就是一位一位地計算會佔用很多的處理器處理時間，尤其在高速通訊的場合，這個缺點更是不可容忍。因此下面再介紹一種按位元組查錶快速計算CRC 的方法。

4 按位元組計算CRC

不難理解，對於一個二進位序列數可以按位元組表示為式(4-1)，其中 $B_n(X)$ 為一個位元組(共8 位)。

$$B(X) = B_n(X) \cdot 2^{8n} + B_{n-1}(X) \cdot 2^{8(n-1)} + \dots + B_1(X) \cdot 2^8 + B_0(X) \quad (4-1)$$

求此二進位序列數的CRC 碼時，先乘以 2^{16} 后（既左移16 位），再除以多項式 $G(X)$ ，所得的餘數既是所要求的CRC 碼。如式(4-2)所示：

$$\frac{B(X) \cdot 2^{16}}{G(X)} = \frac{B_n(X) \cdot 2^{16}}{G(X)} \cdot 2^{8n} + \frac{B_{n-1}(X) \cdot 2^{16}}{G(X)} \cdot 2^{8(n-1)} + \dots + \frac{B_0(X) \cdot 2^{16}}{G(X)} \quad (4-2)$$

可以設：

$$\frac{B_n(X) \cdot 2^{16}}{G(X)} = Q_n(X) + \frac{R_n(X)}{G(X)} \quad (4-3)$$

其中 $Q_n(X)$ 為整數， $R_n(X)$ 為16 位二進位餘數。將式(4-3)代入式(4-2)得：

$$\begin{aligned}\frac{B(X) \cdot 2^{16}}{G(X)} &= [Q_n(X) + \frac{R_n(X)}{G(X)}] \cdot 2^{8n} + \frac{B_{n-1}(X) \cdot 2^{16}}{G(X)} \cdot 2^{8(n-1)} + \dots + \frac{B_0(X) \cdot 2^{16}}{G(X)} \\ &= Q_n(X) \cdot 2^{8n} + \left\{ \frac{R_n(X) \cdot 2^8}{G(X)} + \frac{B_{n-1}(X) \cdot 2^{16}}{G(X)} \right\} \cdot 2^{8(n-1)} + \dots + \frac{B_0 \cdot 2^{16}}{G(X)}\end{aligned}\quad (4-4)$$

因為：

$$\begin{aligned}R_n(X) \cdot 2^8 &= [R_{nH8}(X) \cdot 2^8 + R_{nL8}(X)] \cdot 2^8 \\ &= R_{nH8}(X) \cdot 2^{16} + R_{nL8}(X) \cdot 2^8\end{aligned}\quad (4-5)$$

其中 $R_{nH8}(X)$ 是 $R_n(X)$ 的高八位， $R_{nL8}(X)$ 是 $R_n(X)$ 的低八位。將式 (4-5) 代入式 (4-4)，經整理後得：

$$\frac{B(X) \cdot 2^{16}}{G(X)} = Q_n(X) \cdot 2^{8n} + \left\{ \frac{R_{nL8}(X) \cdot 2^8}{G(X)} + \frac{[R_{nH8}(X) + B_{n-1}(X)] \cdot 2^{16}}{G(X)} \right\} \cdot 2^{8(n-1)} + \dots + \frac{B_0 \cdot 2^{16}}{G(X)}\quad (4-6)$$

再設：

$$\frac{R_{nL8}(X) \cdot 2^8}{G(X)} + \frac{[R_{nH8}(X) + B_{n-1}(X)] \cdot 2^{16}}{G(X)} = Q_{n-1}(X) + \frac{R_{n-1}(X)}{G(X)}\quad (4-7)$$

其中 $Q_{n-1}(X)$ 為整數， $R_{n-1}(X)$ 為 16 位二進位餘數。將式(4-7)代入式(4-6)，如上類推，最後得：

$$\frac{B(X) \cdot 2^{16}}{G(X)} = Q_n(X) \cdot 2^{8n} + Q_{n-1}(X) \cdot 2^{8(n-1)} + \dots + Q_0(X) + \frac{R_0(X)}{G(X)}\quad (4-8)$$

很顯然，十六位二進位數 $R_0(X)$ 既是我們要求的 CRC 碼。

式(4-7)是編寫按位元組計算 CRC 程序的關鍵，它說明計算本位元組後的 CRC 碼等於上一位元組余式 CRC 碼的低 8 位左移 8 位後，再加上上一位元組 CRC 右移 8 位（也既取高 8 位）和本位元組之和後所求得的 CRC 碼，如果我們把 8 位二進位序列數的 CRC 全部計算出來，放如一個表裡，採用查表法，可以大大提高計算速度。由此不難理解下面按位元組求 CRC 碼的 C 語言程序。*ptr 指向發送緩衝區的首位元組，len 是要發送的總位元組數，CRC 余式表是按 0x11021 多項式求出的。

```
unsigned int cal_crc(unsigned char *ptr, unsigned char len) {
    unsigned int crc;
    unsigned char da;
    unsigned int crc_ta[256]={ /* CRC 余式表 */
        0x0000, 0x1021, 0x2042, 0x3063, 0x4084, 0x50a5, 0x60c6, 0x70e7,
        0x8108, 0x9129, 0xa14a, 0xb16b, 0xc18c, 0xd1ad, 0xe1ce, 0xf1ef,
        0x1231, 0x0210, 0x3273, 0x2252, 0x52b5, 0x4294, 0x72f7, 0x62d6,
        0x9339, 0x8318, 0xb37b, 0xa35a, 0xd3bd, 0xc39c, 0xf3ff, 0xe3de,
        0x2462, 0x3443, 0x0420, 0x1401, 0x64e6, 0x74c7, 0x44a4, 0x5485,
        0xa56a, 0xb54b, 0x8528, 0x9509, 0xe5ee, 0xf5cf, 0xc5ac, 0xd58d,
        0x3653, 0x2672, 0x1611, 0x0630, 0x76d7, 0x66f6, 0x5695, 0x46b4,
        0xb75b, 0xa77a, 0x9719, 0x8738, 0xf7df, 0xe7fe, 0xd79d, 0xc7bc,
        0x48c4, 0x58e5, 0x6886, 0x78a7, 0x0840, 0x1861, 0x2802, 0x3823,
        0xc9cc, 0xd9ed, 0xe98e, 0xf9af, 0x8948, 0x9969, 0xa90a, 0xb92b,
        0x5af5, 0x4ad4, 0x7ab7, 0x6a96, 0x1a71, 0x0a50, 0x3a33, 0x2a12,
        0xdbfd, 0xcbdc, 0xfbbf, 0xeb9e, 0x9b79, 0x8b58, 0xbb3b, 0xab1a,
        0x6ca6, 0x7c87, 0x4ce4, 0x5cc5, 0x2c22, 0x3c03, 0x0c60, 0x1c41,
        0xedae, 0xfd8f, 0xcdec, 0xddcd, 0xad2a, 0xbd0b, 0x8d68, 0x9d49,
        0x7e97, 0x6eb6, 0x5ed5, 0x4ef4, 0x3e13, 0x2e32, 0x1e51, 0x0e70,
        0xff9f, 0xefbe, 0xdfdd, 0xcffc, 0xbf1b, 0xaf3a, 0x9f59, 0x8f78,
        0x9188, 0x81a9, 0xb1ca, 0xa1eb, 0xd10c, 0xc12d, 0xf14e, 0xe16f,
        0x1080, 0x00a1, 0x30c2, 0x20e3, 0x5004, 0x4025, 0x7046, 0x6067,
        0x83b9, 0x9398, 0xa3fb, 0xb3da, 0xc33d, 0xd31c, 0xe37f, 0xf35e,
        0x02b1, 0x1290, 0x22f3, 0x32d2, 0x4235, 0x5214, 0x6277, 0x7256,
        0xb5ea, 0xa5cb, 0x95a8, 0x8589, 0xf56e, 0xe54f, 0xd52c, 0xc50d,
        0x34e2, 0x24c3, 0x14a0, 0x0481, 0x7466, 0x6447, 0x5424, 0x4405,
        0xa7db, 0xb7fa, 0x8799, 0x97b8, 0xe75f, 0xf77e, 0xc71d, 0xd73c,
        0x26d3, 0x36f2, 0x0691, 0x16b0, 0x6657, 0x7676, 0x4615, 0x5634,
        0xd94c, 0xc96d, 0xf90e, 0xe92f, 0x99c8, 0x89e9, 0xb98a, 0xa9ab,
        0x5844, 0x4865, 0x7806, 0x6827, 0x18c0, 0x08e1, 0x3882, 0x28a3,
        0xcb7d, 0xdb5c, 0xeb3f, 0xfb1e, 0x8bf9, 0x9bd8, 0xabbb, 0xbb9a,
        0x4a75, 0x5a54, 0x6a37, 0x7a16, 0x0af1, 0x1ad0, 0x2ab3, 0x3a92,
        0xfd2e, 0xed0f, 0xdd6c, 0xcd4d, 0xbdaa, 0xad8b, 0x9de8, 0x8dc9,
        0x7c26, 0x6c07, 0x5c64, 0x4c45, 0x3ca2, 0x2c83, 0x1ce0, 0x0cc1,
```

```

0xef1f, 0xff3e, 0xcf5d, 0xdf7c, 0xaf9b, 0xbfb, 0x8fd9, 0x9ff8,
0x6e17, 0x7e36, 0x4e55, 0x5e74, 0x2e93, 0x3eb2, 0x0ed1, 0x1ef0
};
crc=0;
while(len--!=0) {
da=(uchar) (crc/256); /* 以8 位二進位數的形式暫存CRC 的高8 位 */
crc<=8; /* 左移8 位, 相當於CRC 的低8 位乘以28 */
crc^=crc_ta[da*ptr]; /* 高8 位和當前位元組相加后再查表求CRC , 再加上以前的CRC */
ptr++;
}
return(crc);
}

```

很顯然, 按位元組求CRC 時, 由於採用了查表法, 大大提高了計算速度。但對於廣泛運用的8位微處理器, 代碼空間有限, 對於要求256 個CRC 余式表 (共512 位元組的內存) 已經顯得捉襟見肘了, 但CRC 的計算速度又不可以太慢, 因此再介紹下面一種按半位元組求CRC 的演算法。

5 按半位元組計算

CRC同樣道理, 對於一個二進位序列數可以按位元組表示為式(5-1), 其中 $B_n(X)$ 為半個位元組(共4 位)。

$$B(X) = B_n(X) \cdot 2^{4n} + B_{n-1}(X) \cdot 2^{4(n-1)} + \dots + B_1(X) \cdot 2^4 + B_0(X) \quad (5-1)$$

求此二進位序列數的CRC 碼時, 先乘以 2^{16} 后 (既左移16 位), 再除以多項式 $G(X)$, 所得的餘數既是所要求的CRC 碼。如式(5-2)所示:

$$\frac{B(X) \cdot 2^{16}}{G(X)} = \frac{B_n(X) \cdot 2^{16}}{G(X)} \cdot 2^{4n} + \frac{B_{n-1}(X) \cdot 2^{16}}{G(X)} \cdot 2^{4(n-1)} + \dots + \frac{B_0(X) \cdot 2^{16}}{G(X)} \quad (5-2)$$

$$\text{可以設: } \frac{B_n(X) \cdot 2^{16}}{G(X)} = Q_n(X) + \frac{R_n(X)}{G(X)} \quad (5-3)$$

其中 $Q_n(X)$ 為整數, $R_n(X)$ 為16 位二進位餘數。將式(5-3)代入式(5-2)得:

$$\begin{aligned} \frac{B(X) \cdot 2^{16}}{G(X)} &= [Q_n(X) + \frac{R_n(X)}{G(X)}] \cdot 2^{4n} + \frac{B_{n-1}(X) \cdot 2^{16}}{G(X)} \cdot 2^{4(n-1)} + \dots + \frac{B_0(X) \cdot 2^{16}}{G(X)} \\ &= Q_n(X) \cdot 2^{4n} + \left\{ \frac{R_n(X) \cdot 2^4}{G(X)} + \frac{B_{n-1}(X) \cdot 2^{16}}{G(X)} \right\} \cdot 2^{4(n-1)} + \dots + \frac{B_0 \cdot 2^{16}}{G(X)} \end{aligned} \quad (5-4)$$

$$\begin{aligned} \text{因為: } R_n(X) \cdot 2^4 &= [R_{nH4}(X) \cdot 2^{12} + R_{nL12}(X)] \cdot 2^4 \\ &= R_{nH4}(X) \cdot 2^{16} + R_{nL12}(X) \cdot 2^4 \end{aligned} \quad (5-5)$$

其中 $R_{nH4}(X)$ 是 $R_n(X)$ 的高4 位, $R_{nL12}(X)$ 是 $R_n(X)$ 的低12 位。將式 (5-5) 代入式 (5-4), 經整理后得:

$$\frac{B(X) \cdot 2^{16}}{G(X)} = Q_n(X) \cdot 2^{4n} + \left\{ \frac{R_{nL12}(X) \cdot 2^4}{G(X)} + \frac{[R_{nH4}(X) + B_{n-1}(X)] \cdot 2^{16}}{G(X)} \right\} \cdot 2^{4(n-1)} + \dots + \frac{B_0 \cdot 2^{16}}{G(X)} \quad (5-6)$$

$$\text{再設: } \frac{R_{nL12}(X) \cdot 2^4}{G(X)} + \frac{[B_{nH4}(X) + B_{n-1}(X)] \cdot 2^{16}}{G(X)} = Q_{n-1}(X) + \frac{R_{n-1}(X)}{G(X)} \quad (5-7)$$

其中 $Q_{n-1}(X)$ 為整數, $R_{n-1}(X)$ 為16 位二進位餘數。將式(5-7)代入式(5-6), 如上類推, 最後得:

$$\frac{B(X) \cdot 2^{16}}{G(X)} = Q_n(X) \cdot 2^{4n} + Q_{n-1}(X) \cdot 2^{4(n-1)} + \dots + Q_0(X) + \frac{R_0(X)}{G(X)} \quad (5-8)$$

很顯然, 十六位二進位數 $R_0(X)$ 既是我們要求的CRC 碼。

式(5-7)是編寫按位元組計算CRC 程序的關鍵, 它說明計算本位元組后的CRC 碼等於上一位元組CRC碼的低12 位左移4 位后, 再加上上一位元組余式CRC 右移4 位 (也既取高4 位) 和本位元組之和后所求得的CRC 碼, 如果我們把4 位二進位序列數的CRC 全部計算出來, 放在一個表裡, 採用查表法, 每個位元組算兩次 (半位元組算一次), 可以在速度和內存空間取得均衡。由此不難理解下面按半位元組求CRC 碼的C 語言程序。*ptr 指向發送緩衝區的首位元組, len 是要發送的總位元組數, CRC 余式表是按 $0x11021$ 多項式求出的。

```

unsigned cal_crc(unsigned char *ptr, unsigned char len) {
unsigned int crc;
unsigned char da;
unsigned int crc_ta[16]={ /* CRC 余式表 */
0x0000,0x1021,0x2042,0x3063,0x4084,0x50a5,0x60c6,0x70e7,

```

```

0x8108,0x9129,0xa14a,0xb16b,0xc18c,0xd1ad,0xe1ce,0xf1ef,
}
crc=0;
while(len--!=0) {
da=((uchar)(crc/256))/16; /* 暫存CRC 的高四位 */
crc<<=4; /* CRC 右移4 位, 相當於取CRC 的低12 位) */
crc^=crc_ta[da^(*ptr/16)]; /* CRC 的高4 位和本位元組的前半位元組相加后查表計算CRC ,
然後加上上一次CRC 的餘數 */
da=((uchar)(crc/256))/16; /* 暫存CRC 的高4 位 */
crc<<=4; /* CRC 右移4 位, 相當於CRC 的低12 位) */
crc^=crc_ta[da^(*ptr&0x0f)]; /* CRC 的高4 位和本位元組的後半位元組相加后查表計算
CRC ,
然後再加上上一次CRC 的餘數 */
ptr++;
}
return(crc);
}

```

5 結束語

以上介紹的三種求CRC 的程序, 按位求法速度較慢, 但佔用最小的內存空間; 按位元組查表求CRC 的方法速度較快, 但佔用較大的內存; 按半位元組查表求CRC 的方法是前兩者的均衡, 即不會佔用太多的內存, 同時速度又不至於太慢, 比較適合8 位小內存的單片機的應用場合。以上所給的C 程序可以根據各微處理器編譯器的特點作相應的改變, 比如把CRC 余式表放到程序存儲區內等。

Google 提供的廣告

Crc

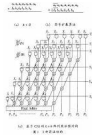
算法

所得計算

- [PCB技術大全](#)
- [OrCAD Capture 原理圖對話框中英對照](#)
- [protel技術大全](#)
- [Keil C51開發系統基本知識 \(1 \)](#)
- [嵌入式C語言測試題](#)
- [protel 99 se教程\(原理圖設計\)](#)
- [Keil C51使用詳解2](#)
- [基於精簡TCP/IP協議棧的信息家電網路伺服器](#)
- [用U-BOOT構建嵌入式系統的引導裝載程序](#)
- [在51系列單片機上移植uCOS-II](#)
- [C/C++編程新手錯誤語錄](#)
- [「Duplicate Pin Name found on Package」錯誤](#)
- [樓宇自動化控制網路數據通信協議BACnet](#)
- [PROTEL DXP創建原理圖器件](#)
- [cadence原理圖設計簡介](#)
- [Pxe Win98完全安裝手冊\(下\)](#)
- [VxWorks中文FAQ](#)
- [FrameBuffer 原理、實現與應用](#)
- [Linux 下 C 語言編程](#)
- [uboot移植到S3C44B0X開發板的經歷](#)
- [MAX+PLUS II快速入門](#)
- [自定製Nios處理器的FFT演算法指令](#)
- [打造SQL Server2000的安全策略教程](#)
- [AVR-GCC里定義的API](#)
- [CYPRESS FX2\(USB2.0 單片機\)讀書筆記](#)
- [Linux環境下基於I2C匯流排的EEPROM 驅動程序](#)
- [VxWorks嵌入式操作系統的真FFS文件系統驅動開發](#)
- [使用compilib命令編譯Xilinx的ModelSim模擬庫](#)
- [一個簡單的匯流排輪詢仲裁器Verilog代碼](#)
- [Linux系統下的ELF文件分析](#)



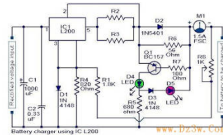
你的英語有幾分？立即檢測



32位乘法器性能比較

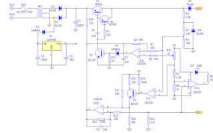


兼職接案賺獎金,快上PRO360



國外12V鉛酸蓄電池充電電路圖

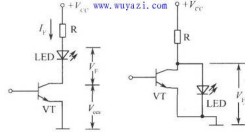
廣告 TutorABC



自製恆流限壓式鉛酸電池充電器電路

cocdig.com

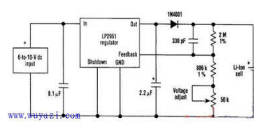
cocdig.com



簡易的直流LED驅動電路

cocdig.com

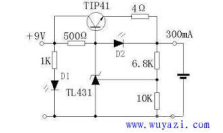
廣告 PRO 360



單節鋰電池充電器電路圖

cocdig.com

cocdig.com



簡單實用的鋰電池充電器電路

cocdig.com

[[admin](#) via 研發互助社區] **CRC 演算法原理及C 語言實現**已經有19425次圍觀

0条评论

排序方式 **最新**



添加评论...

Facebook 评论插件

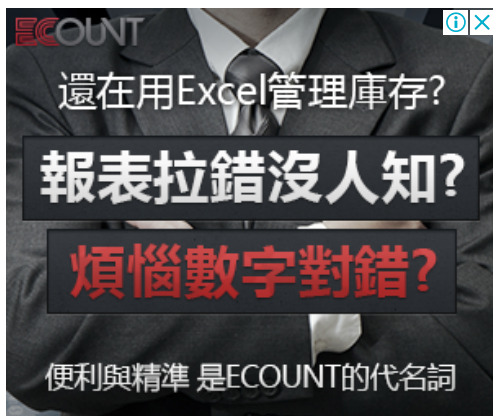
本文地址：<http://cocdig.com/docs/show-post-42378.html>

推薦 6 分享

- 搜.文檔.資訊.知識.專題
- 搜尋

熱門文章

1. [1602字元液晶詳細資料和實例](#)
2. [stm32固件庫3.3版本在stm3210e-eval開發板上的移植](#)
3. [一個按鍵的多次擊鍵組合判別技巧彙編程序](#)
4. [s3c2410移植MPlayer到linux2.6](#)
5. [Google Android內核編譯教程](#)
6. [將Android移植到FS2410開發板上](#)
7. [rt73 USB無線網卡驅動在armlinux平台上的移植](#)
8. [PIC軟體串列非同步通信三倍速採樣法設計](#)
9. [基於PIC單片機的SPWM控制技術](#)
10. [靜態編譯web server Appweb\(帶Matrixssl支持\) For ARM9、linux](#)



最新文章

1. [單片機編程經驗](#)
2. [S3C44B0存儲器的BANK設計和控制](#)
3. [S3C44B0 寄存器描述](#)
4. [S3c44b0 RTC程序](#)
5. [S3C44b0引導註釋](#)
6. [S3C2440應用筆記](#)
7. [DSP中斷向量表FAQ](#)
8. [DSP 54x串口FAQ](#)
9. [DSP與存儲器FAQ](#)
10. [DSP 5402時鐘與定時器中斷FAQ](#)



[关于我们](#) 聯繫郵箱： 站點： [研發互助社區](#) ©2014-2018 版權所有 部份內容來源於互聯網，僅供參考，專業問題請諮詢專家。