<http://www.aiuxian.com/article/p-1652764.html>

# stm32f4驱动enc28j60（吐槽篇）

分类： [单片机](http://www.aiuxian.com/catalog/p-241096.html)  |  标签: enc28j60,STM32F4Discovery  |  作者： wzs298 [相关](http://www.aiuxian.com/relative/p-1652764.html)  |  发布日期 : 2014-09-28  |  热度 : 320°

鼓捣enc28j60这个模块过程还是比较曲折的！

买这个模块的时候，卖家只附送了51的驱动工程。或者说，就51的工程能用！

在51上，一编译一下载就搞定了！

========================吐槽下==============================

在F4这个平台上呢！因为enc28j60.c这个文件的问题，导致我创建了20多份不同的工程来测试！比较坑爹！

刚开始的时候，想移植51的这个文件，发现改动量太大了，改了一下，测试不成功就放弃之！

再拿野火stm32f103的来移植，还是不成功。发现也有网友有这问题，我跟踪调试，发现mac初始化一直不成功！

只好去拿官方库里面的 lwip 的工程来改。还是不行！ 这下不淡定了！

打算去收集资料！

百度关键字 stm32f4 enc28j60 ，几乎没可用的资料！有的也是用手指头都能数的完的求助贴！

谷歌关键字 stm32f4 enc28j60 ，有惊喜！能看到几个视频，而且要看还得翻墙！

那就翻吧！废了九牛二虎之力，找到了三份源码！其中日本那份下载不了，另外两份倒是下载了。

但是打开一看，又纠结了，一份是linux下的工程。另一份不知道拿什么开发工具写的。不过这两份源码倒是能用！

----------------------------------------------------------------------------------------------------

重新看了一遍enc28j60的手册，拿之前lcd的工程加上国外的源码重新写吧！

前前后后弄了两天，终于出来了！深感累到不行！

============================END=============================

说点有用的，enc28j60与f4通信比较简单。使用SPI，也就需要四根线就行了。

我的接法是：

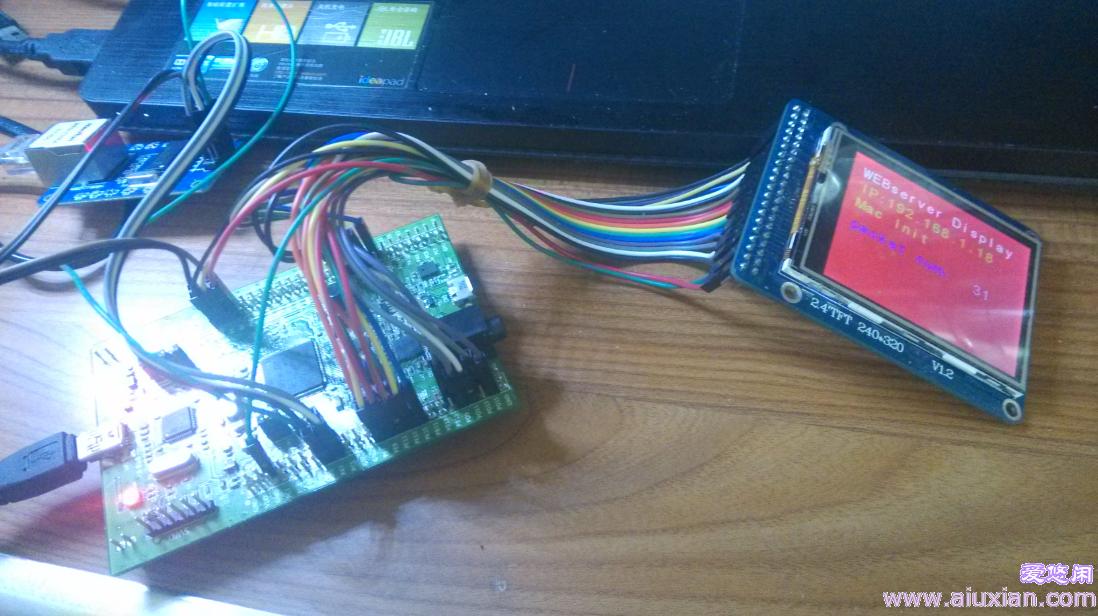
PA4 -------------- CS

PA5 -------------- SCK

PA6 -------------- SO

PA7 -------------- SI

来张图：





ENC28J60.C

|  |  |  |
| --- | --- | --- |
| 001 | #include "ENC28J60.h" | |
| 002 | #include "spi.h" |

|  |  |
| --- | --- |
| 003 | #include "lcd.h" |
| 004 | static uint8\_t Enc28j60Bank; | |

|  |  |  |
| --- | --- | --- |
| 005 | static uint16\_t gNextPacketPtr; | |
| 006 | static uint8\_t erxfcon; |

|  |  |
| --- | --- |
| 007 |  |
| 008 |  |

|  |  |  |
| --- | --- | --- |
| 009 | unsigned char ENC28J60\_SendByte(unsigned char dt) | |
| 010 | { |

|  |  |  |
| --- | --- | --- |
| 011 | while(SPI\_I2S\_GetFlagStatus(SPI1, SPI\_I2S\_FLAG\_TXE) == RESET); | |
| 012 |  |

|  |  |  |
| --- | --- | --- |
| 013 | SPI\_I2S\_SendData(SPI1, dt); | |
| 014 |  |

|  |  |  |
| --- | --- | --- |
| 015 | while(SPI\_I2S\_GetFlagStatus(SPI1, SPI\_I2S\_FLAG\_RXNE) == RESET); | |
| 016 |  |

|  |  |  |
| --- | --- | --- |
| 017 | return SPI\_I2S\_ReceiveData(SPI1); | |
| 018 | } |

|  |  |
| --- | --- |
| 019 |  |
| 020 | uint8\_t enc28j60ReadOp(uint8\_t op, uint8\_t address) | |

|  |  |
| --- | --- |
| 021 | { |
| 022 | uint8\_t temp; | |

|  |  |
| --- | --- |
| 023 | enableChip; |
| 024 | // issue read command | |

|  |  |  |
| --- | --- | --- |
| 025 | ENC28J60\_SendByte(op | (address & ADDR\_MASK)); | |
| 026 | temp = ENC28J60\_SendByte(0xFF); |

|  |  |
| --- | --- |
| 027 | if (address & 0x80) |
| 028 | temp = ENC28J60\_SendByte(0xFF); | |

|  |  |
| --- | --- |
| 029 |  |
| 030 | // release CS | |

|  |  |
| --- | --- |
| 031 | disableChip; |
| 032 | return temp; | |

|  |  |
| --- | --- |
| 033 | } |
| 034 |  | |

|  |  |  |
| --- | --- | --- |
| 035 | void enc28j60WriteOp(uint8\_t op, uint8\_t address, uint8\_t data) | |
| 036 | { |

|  |  |
| --- | --- |
| 037 | enableChip; |
| 038 | ENC28J60\_SendByte(op | (address & ADDR\_MASK)); | |

|  |  |  |
| --- | --- | --- |
| 039 | ENC28J60\_SendByte(data); | |
| 040 | disableChip; |

|  |  |
| --- | --- |
| 041 | } |
| 042 |  | |

|  |  |
| --- | --- |
| 043 | void enc28j60PowerDown() { |
| 044 | enc28j60WriteOp(ENC28J60\_BIT\_FIELD\_CLR, ECON1, ECON1\_RXEN); | |

|  |  |  |
| --- | --- | --- |
| 045 | while(enc28j60Read(ESTAT) & ESTAT\_RXBUSY); | |
| 046 | while(enc28j60Read(ECON1) & ECON1\_TXRTS); |

|  |  |  |
| --- | --- | --- |
| 047 | enc28j60WriteOp(ENC28J60\_BIT\_FIELD\_SET, ECON2, ECON2\_PWRSV); | |
| 048 | } |

|  |  |
| --- | --- |
| 049 |  |
| 050 | void enc28j60PowerUp() { | |

|  |  |  |
| --- | --- | --- |
| 051 | enc28j60WriteOp(ENC28J60\_BIT\_FIELD\_CLR, ECON2, ECON2\_PWRSV); | |
| 052 | while(!enc28j60Read(ESTAT) & ESTAT\_CLKRDY); |

|  |  |
| --- | --- |
| 053 | } |
| 054 |  | |

|  |  |
| --- | --- |
| 055 |  |
| 056 | void enc28j60ReadBuffer(uint16\_t len, uint8\_t\* data) | |

|  |  |
| --- | --- |
| 057 | { |
| 058 | enableChip; | |

|  |  |  |
| --- | --- | --- |
| 059 | ENC28J60\_SendByte(ENC28J60\_READ\_BUF\_MEM); | |
| 060 | while (len--) { |

|  |  |  |
| --- | --- | --- |
| 061 | \*data++ = ENC28J60\_SendByte(0x00); | |
| 062 | } |

|  |  |
| --- | --- |
| 063 | disableChip; |
| 064 | // Remove next line suggested by user epam - not needed | |

|  |  |  |
| --- | --- | --- |
| 065 | //    \*data='\0'; | |
| 066 | } |

|  |  |
| --- | --- |
| 067 |  |
| 068 | static uint16\_t enc28j60ReadBufferWord() { | |

|  |  |
| --- | --- |
| 069 | uint16\_t result; |
| 070 | enc28j60ReadBuffer(2, (uint8\_t\*) &result); | |

|  |  |  |
| --- | --- | --- |
| 071 | return result; | |
| 072 | } |

|  |  |
| --- | --- |
| 073 |  |
| 074 |  |

|  |  |  |
| --- | --- | --- |
| 075 | void enc28j60WriteBuffer(uint16\_t len, uint8\_t\* data) | |
| 076 | { |

|  |  |
| --- | --- |
| 077 | enableChip; |
| 078 | ENC28J60\_SendByte(ENC28J60\_WRITE\_BUF\_MEM); | |

|  |  |
| --- | --- |
| 079 | while (len--) |
| 080 | ENC28J60\_SendByte(\*data++); | |

|  |  |
| --- | --- |
| 081 |  |
| 082 | disableChip; | |

|  |  |
| --- | --- |
| 083 | } |
| 084 |  | |

|  |  |  |
| --- | --- | --- |
| 085 | void enc28j60SetBank(uint8\_t address) | |
| 086 | { |

|  |  |
| --- | --- |
| 087 | if ((address & BANK\_MASK) != Enc28j60Bank) { |
| 088 | enc28j60WriteOp(ENC28J60\_BIT\_FIELD\_CLR, ECON1, ECON1\_BSEL1|ECON1\_BSEL0); | |

|  |  |
| --- | --- |
| 089 | Enc28j60Bank = address & BANK\_MASK; |
| 090 | enc28j60WriteOp(ENC28J60\_BIT\_FIELD\_SET, ECON1, Enc28j60Bank>>5); | |

|  |  |  |
| --- | --- | --- |
| 091 | } | |
| 092 | } |

|  |  |
| --- | --- |
| 093 |  |
| 094 | uint8\_t enc28j60Read(uint8\_t address) | |

|  |  |
| --- | --- |
| 095 | { |
| 096 | // set the bank | |

|  |  |  |
| --- | --- | --- |
| 097 | enc28j60SetBank(address); | |
| 098 | // do the read |

|  |  |  |
| --- | --- | --- |
| 099 | return enc28j60ReadOp(ENC28J60\_READ\_CTRL\_REG, address); | |
| 100 | } |

|  |  |
| --- | --- |
| 101 |  |
| 102 | void enc28j60WriteWord(uint8\_t address, uint16\_t data) { | |

|  |  |
| --- | --- |
| 103 | enc28j60Write(address, data & 0xff); |
| 104 | enc28j60Write(address + 1, data >> 8); | |

|  |  |
| --- | --- |
| 105 | } |
| 106 |  | |

|  |  |
| --- | --- |
| 107 | // read upper 8 bits |
| 108 | uint16\_t enc28j60PhyReadH(uint8\_t address) | |

|  |  |
| --- | --- |
| 109 | { |
| 110 | // Set the right address and start the register read operation | |

|  |  |
| --- | --- |
| 111 | enc28j60Write(MIREGADR, address); |
| 112 | enc28j60Write(MICMD, MICMD\_MIIRD); | |

|  |  |  |
| --- | --- | --- |
| 113 | delay\_us(15); | |
| 114 |  |

|  |  |
| --- | --- |
| 115 | // wait until the PHY read completes |
| 116 | while(enc28j60Read(MISTAT) & MISTAT\_BUSY); | |

|  |  |
| --- | --- |
| 117 |  |
| 118 | // reset reading bit | |

|  |  |  |
| --- | --- | --- |
| 119 | enc28j60Write(MICMD, 0x00); | |
| 120 |  |

|  |  |  |
| --- | --- | --- |
| 121 | return (enc28j60Read(MIRDH)); | |
| 122 | } |

|  |  |
| --- | --- |
| 123 |  |
| 124 |  |

|  |  |  |
| --- | --- | --- |
| 125 | void enc28j60Write(uint8\_t address, uint8\_t data) | |
| 126 | { |

|  |  |
| --- | --- |
| 127 | // set the bank |
| 128 | enc28j60SetBank(address); | |

|  |  |
| --- | --- |
| 129 | // do the write |
| 130 | enc28j60WriteOp(ENC28J60\_WRITE\_CTRL\_REG, address, data); | |

|  |  |
| --- | --- |
| 131 | } |
| 132 |  | |

|  |  |
| --- | --- |
| 133 |  |
| 134 | void enc28j60PhyWrite(uint8\_t address, uint16\_t data) | |

|  |  |
| --- | --- |
| 135 | { |
| 136 | // set the PHY register address | |

|  |  |  |
| --- | --- | --- |
| 137 | enc28j60Write(MIREGADR, address); | |
| 138 | // write the PHY data |

|  |  |
| --- | --- |
| 139 | enc28j60Write(MIWRL, data); |
| 140 | enc28j60Write(MIWRH, data>>8); | |

|  |  |
| --- | --- |
| 141 | // wait until the PHY write completes |
| 142 | while(enc28j60Read(MISTAT) & MISTAT\_BUSY){ | |

|  |  |  |
| --- | --- | --- |
| 143 | delay\_us(15); | |
| 144 | } |

|  |  |
| --- | --- |
| 145 | } |
| 146 | /\* | |

|  |  |  |
| --- | --- | --- |
| 147 | static void enc28j60PhyWriteWord(byte address, word data) { | |
| 148 | enc28j60Write(MIREGADR, address); |

|  |  |  |
| --- | --- | --- |
| 149 | //enc28j60WriteByte(MIREGADR, address); | |
| 150 | enc28j60WriteWord(MIWRL, data); |

|  |  |  |
| --- | --- | --- |
| 151 | while (enc28j60ReadByte(MISTAT) & MISTAT\_BUSY) | |
| 152 | ; |

|  |  |
| --- | --- |
| 153 | } |
| 154 | \*/ | |

|  |  |  |
| --- | --- | --- |
| 155 | void enc28j60clkout(uint8\_t clk) | |
| 156 | { |

|  |  |  |
| --- | --- | --- |
| 157 | //setup clkout: 2 is 12.5MHz: | |
| 158 | enc28j60Write(ECOCON, clk & 0x7); |

|  |  |
| --- | --- |
| 159 | } |
| 160 |  | |

|  |  |  |
| --- | --- | --- |
| 161 | void enc28j60Init( uint8\_t\* macaddr ) | |
| 162 | { |

|  |  |  |
| --- | --- | --- |
| 163 | enableChip; // ss=0 | |
| 164 |  |

|  |  |
| --- | --- |
| 165 | // perform system reset |
| 166 | enc28j60WriteOp(ENC28J60\_SOFT\_RESET, 0, ENC28J60\_SOFT\_RESET); | |

|  |  |
| --- | --- |
| 167 | delay\_ms(50); |
| 168 | // check CLKRDY bit to see if reset is complete | |

|  |  |  |
| --- | --- | --- |
| 169 | // The CLKRDY does not work. See Rev. B4 Silicon Errata point. Just wait. | |
| 170 | //while(!(enc28j60Read(ESTAT) & ESTAT\_CLKRDY)); |

|  |  |
| --- | --- |
| 171 | // do bank 0 stuff |
| 172 | // initialize receive buffer | |

|  |  |  |
| --- | --- | --- |
| 173 | // 16-bit transfers, must write low byte first | |
| 174 | // set receive buffer start address |

|  |  |  |
| --- | --- | --- |
| 175 | gNextPacketPtr = RXSTART\_INIT; | |
| 176 | // Rx start |

|  |  |  |
| --- | --- | --- |
| 177 | enc28j60WriteWord(ERXSTL, RXSTART\_INIT); | |
| 178 | // set receive pointer address |

|  |  |  |
| --- | --- | --- |
| 179 | enc28j60WriteWord(ERXRDPTL, RXSTART\_INIT); | |
| 180 | // RX end |

|  |  |  |
| --- | --- | --- |
| 181 | enc28j60WriteWord(ERXNDL, RXSTOP\_INIT); | |
| 182 | // TX start |

|  |  |  |
| --- | --- | --- |
| 183 | enc28j60WriteWord(ETXSTL, TXSTART\_INIT); | |
| 184 | // TX end |

|  |  |  |
| --- | --- | --- |
| 185 | enc28j60WriteWord(ETXNDL, TXSTOP\_INIT); | |
| 186 | // do bank 1 stuff, packet filter: |

|  |  |
| --- | --- |
| 187 | // For broadcast packets we allow only ARP packtets |
| 188 | // All other packets should be unicast only for our mac (MAADR) | |

|  |  |
| --- | --- |
| 189 | // |
| 190 | // The pattern to match on is therefore | |

|  |  |
| --- | --- |
| 191 | // Type     ETH.DST |
| 192 | // ARP      BROADCAST | |

|  |  |  |
| --- | --- | --- |
| 193 | // 06 08 -- ff ff ff ff ff ff -> ip checksum for theses bytes=f7f9 | |
| 194 | // in binary these poitions are:11 0000 0011 1111 |

|  |  |  |
| --- | --- | --- |
| 195 | // This is hex 303F->EPMM0=0x3f,EPMM1=0x30 | |
| 196 |  |

|  |  |  |
| --- | --- | --- |
| 197 | //enc28j60Write(ERXFCON, ERXFCON\_UCEN|ERXFCON\_CRCEN|ERXFCON\_PMEN); | |
| 198 | //Change to add ERXFCON\_BCEN recommended by epam |

|  |  |
| --- | --- |
| 199 | //enc28j60Write(ERXFCON, ERXFCON\_UCEN|ERXFCON\_CRCEN|ERXFCON\_PMEN|ERXFCON\_BCEN); |
| 200 | erxfcon =  ERXFCON\_UCEN|ERXFCON\_CRCEN|ERXFCON\_PMEN|ERXFCON\_BCEN; |

|  |  |
| --- | --- |
| 201 | enc28j60Write(ERXFCON, erxfcon ); |
| 202 | enc28j60WriteWord(EPMM0, 0x303f); |

|  |  |  |
| --- | --- | --- |
| 203 | enc28j60WriteWord(EPMCSL, 0xf7f9); | |
| 204 | // |

|  |  |
| --- | --- |
| 205 | // do bank 2 stuff |
| 206 | // enable MAC receive | |

|  |  |  |
| --- | --- | --- |
| 207 | enc28j60Write(MACON1, MACON1\_MARXEN|MACON1\_TXPAUS|MACON1\_RXPAUS); | |
| 208 | // bring MAC out of reset |

|  |  |
| --- | --- |
| 209 | enc28j60Write(MACON2, 0x00); |
| 210 | // enable automatic padding to 60bytes and CRC operations | |

|  |  |  |
| --- | --- | --- |
| 211 | enc28j60WriteOp(ENC28J60\_BIT\_FIELD\_SET, MACON3, MACON3\_PADCFG0|MACON3\_TXCRCEN|MACON3\_FRMLNEN);  //|MACON3\_FULDPX); | |
| 212 | // set inter-frame gap (non-back-to-back) |

|  |  |
| --- | --- |
| 213 | enc28j60WriteWord(MAIPGL, 0x0C12); |
| 214 | // set inter-frame gap (back-to-back) | |

|  |  |
| --- | --- |
| 215 | enc28j60Write(MABBIPG, 0x12); |
| 216 | // Set the maximum packet size which the controller will accept | |

|  |  |  |
| --- | --- | --- |
| 217 | // Do not send packets longer than MAX\_FRAMELEN: | |
| 218 | enc28j60WriteWord(MAMXFLL, MAX\_FRAMELEN); |

|  |  |
| --- | --- |
| 219 | // do bank 3 stuff |
| 220 | // write MAC address | |

|  |  |  |
| --- | --- | --- |
| 221 | // NOTE: MAC address in ENC28J60 is byte-backward | |
| 222 | enc28j60Write(MAADR5, macaddr[0]); |

|  |  |
| --- | --- |
| 223 | enc28j60Write(MAADR4, macaddr[1]); |
| 224 | enc28j60Write(MAADR3, macaddr[2]); |

|  |  |
| --- | --- |
| 225 | enc28j60Write(MAADR2, macaddr[3]); |
| 226 | enc28j60Write(MAADR1, macaddr[4]); |

|  |  |  |
| --- | --- | --- |
| 227 | enc28j60Write(MAADR0, macaddr[5]); | |
| 228 | // no loopback of transmitted frames |

|  |  |  |
| --- | --- | --- |
| 229 | enc28j60PhyWrite(PHCON2, PHCON2\_HDLDIS); | |
| 230 | // switch to bank 0 |

|  |  |  |
| --- | --- | --- |
| 231 | enc28j60SetBank(ECON1); | |
| 232 | // enable interrutps |

|  |  |  |
| --- | --- | --- |
| 233 | enc28j60WriteOp(ENC28J60\_BIT\_FIELD\_SET, EIE, EIE\_INTIE|EIE\_PKTIE); | |
| 234 | // enable packet reception |

|  |  |  |
| --- | --- | --- |
| 235 | enc28j60WriteOp(ENC28J60\_BIT\_FIELD\_SET, ECON1, ECON1\_RXEN); | |
| 236 |  |

|  |  |  |
| --- | --- | --- |
| 237 | LCD\_String(20,80,"Mac Init",GREEN); | |
| 238 | } |

|  |  |
| --- | --- |
| 239 |  |
| 240 | // read the revision of the chip: | |

|  |  |  |
| --- | --- | --- |
| 241 | uint8\_t enc28j60getrev(void) | |
| 242 | { |

|  |  |
| --- | --- |
| 243 | uint8\_t rev; |
| 244 | rev=enc28j60Read(EREVID); | |

|  |  |
| --- | --- |
| 245 | // microchip forgot to step the number on the silcon when they |
| 246 | // released the revision B7. 6 is now rev B7. We still have |

|  |  |  |
| --- | --- | --- |
| 247 | // to see what they do when they release B8. At the moment | |
| 248 | // there is no B8 out yet |

|  |  |  |
| --- | --- | --- |
| 249 | if (rev>5) rev++; | |
| 250 | return(rev); |

|  |  |
| --- | --- |
| 251 | } |
| 252 |  | |

|  |  |  |
| --- | --- | --- |
| 253 | // A number of utility functions to enable/disable broadcast and multicast bits | |
| 254 | void enc28j60EnableBroadcast( void ) { |

|  |  |
| --- | --- |
| 255 | erxfcon |= ERXFCON\_BCEN; |
| 256 | enc28j60Write(ERXFCON, erxfcon); | |

|  |  |
| --- | --- |
| 257 | } |
| 258 |  | |

|  |  |  |
| --- | --- | --- |
| 259 | void enc28j60DisableBroadcast( void ) { | |
| 260 | erxfcon &= (0xff ^ ERXFCON\_BCEN); |

|  |  |  |
| --- | --- | --- |
| 261 | enc28j60Write(ERXFCON, erxfcon); | |
| 262 | } |

|  |  |
| --- | --- |
| 263 |  |
| 264 | void enc28j60EnableMulticast( void ) { | |

|  |  |
| --- | --- |
| 265 | erxfcon |= ERXFCON\_MCEN; |
| 266 | enc28j60Write(ERXFCON, erxfcon); | |

|  |  |
| --- | --- |
| 267 | } |
| 268 |  | |

|  |  |  |
| --- | --- | --- |
| 269 | void enc28j60DisableMulticast( void ) { | |
| 270 | erxfcon &= (0xff ^ ERXFCON\_MCEN); |

|  |  |  |
| --- | --- | --- |
| 271 | enc28j60Write(ERXFCON, erxfcon); | |
| 272 | } |

|  |  |
| --- | --- |
| 273 |  |
| 274 |  |

|  |  |
| --- | --- |
| 275 | // link status |
| 276 | uint8\_t enc28j60linkup(void) | |

|  |  |
| --- | --- |
| 277 | { |
| 278 | // bit 10 (= bit 3 in upper reg) | |

|  |  |  |
| --- | --- | --- |
| 279 | return(enc28j60PhyReadH(PHSTAT2) && 4); | |
| 280 | } |

|  |  |
| --- | --- |
| 281 |  |
| 282 | void enc28j60PacketSend(uint16\_t len, uint8\_t\* packet) | |

|  |  |
| --- | --- |
| 283 | { |
| 284 | // Check no transmit in progress | |

|  |  |  |
| --- | --- | --- |
| 285 | while (enc28j60ReadOp(ENC28J60\_READ\_CTRL\_REG, ECON1) & ECON1\_TXRTS) | |
| 286 | { |

|  |  |
| --- | --- |
| 287 | // Reset the transmit logic problem. See Rev. B4 Silicon Errata point 12. |
| 288 | if( (enc28j60Read(EIR) & EIR\_TXERIF) ) { |

|  |  |
| --- | --- |
| 289 | enc28j60WriteOp(ENC28J60\_BIT\_FIELD\_SET, ECON1, ECON1\_TXRST); |
| 290 | enc28j60WriteOp(ENC28J60\_BIT\_FIELD\_CLR, ECON1, ECON1\_TXRST); |

|  |  |  |
| --- | --- | --- |
| 291 | } | |
| 292 | } |

|  |  |
| --- | --- |
| 293 |  |
| 294 | // Set the write pointer to start of transmit buffer area | |

|  |  |
| --- | --- |
| 295 | enc28j60WriteWord(EWRPTL, TXSTART\_INIT); |
| 296 | // Set the TXND pointer to correspond to the packet size given | |

|  |  |
| --- | --- |
| 297 | enc28j60WriteWord(ETXNDL, (TXSTART\_INIT+len)); |
| 298 | // write per-packet control byte (0x00 means use macon3 settings) | |

|  |  |  |
| --- | --- | --- |
| 299 | enc28j60WriteOp(ENC28J60\_WRITE\_BUF\_MEM, 0, 0x00); | |
| 300 | // copy the packet into the transmit buffer |

|  |  |
| --- | --- |
| 301 | enc28j60WriteBuffer(len, packet); |
| 302 | // send the contents of the transmit buffer onto the network | |

|  |  |
| --- | --- |
| 303 | enc28j60WriteOp(ENC28J60\_BIT\_FIELD\_SET, ECON1, ECON1\_TXRTS); |
| 304 | // Reset the transmit logic problem. See Rev. B4 Silicon Errata point 12. | |

|  |  |
| --- | --- |
| 305 | } |
| 306 |  | |

|  |  |  |
| --- | --- | --- |
| 307 | // just probe if there might be a packet | |
| 308 | //uint8\_t enc28j60hasRxPkt(void) |

|  |  |
| --- | --- |
| 309 | //{ |
| 310 | //       return enc28j60ReadByte(EPKTCNT) > 0; | |

|  |  |  |
| --- | --- | --- |
| 311 | //} | |
| 312 |  |

|  |  |  |
| --- | --- | --- |
| 313 | // Gets a packet from the network receive buffer, if one is available. | |
| 314 | // The packet will by headed by an ethernet header. |

|  |  |  |
| --- | --- | --- |
| 315 | //      maxlen  The maximum acceptable length of a retrieved packet. | |
| 316 | //      packet  Pointer where packet data should be stored. |

|  |  |  |
| --- | --- | --- |
| 317 | // Returns: Packet length in bytes if a packet was retrieved, zero otherwise. | |
| 318 | uint16\_t enc28j60PacketReceive(uint16\_t maxlen, uint8\_t\* packet) |

|  |  |
| --- | --- |
| 319 | { |
| 320 | uint16\_t rxstat; | |

|  |  |
| --- | --- |
| 321 | uint16\_t len; |
| 322 | // check if a packet has been received and buffered | |

|  |  |
| --- | --- |
| 323 | //if( !(enc28j60Read(EIR) & EIR\_PKTIF) ){ |
| 324 | // The above does not work. See Rev. B4 Silicon Errata point 6. | |

|  |  |  |
| --- | --- | --- |
| 325 | if( enc28j60Read(EPKTCNT) ==0 ){ | |
| 326 | return(0); |

|  |  |  |
| --- | --- | --- |
| 327 | } | |
| 328 |  |

|  |  |  |
| --- | --- | --- |
| 329 | // Set the read pointer to the start of the received packet | |
| 330 | enc28j60WriteWord(ERDPTL, gNextPacketPtr); |

|  |  |  |
| --- | --- | --- |
| 331 | //enc28j60Write(ERDPTL, (gNextPacketPtr &0xFF)); | |
| 332 | //enc28j60Write(ERDPTH, (gNextPacketPtr)>>8); |

|  |  |
| --- | --- |
| 333 | // read the next packet pointer |
| 334 | gNextPacketPtr  = enc28j60ReadBufferWord(); | |

|  |  |
| --- | --- |
| 335 | //gNextPacketPtr  = enc28j60ReadOp(ENC28J60\_READ\_BUF\_MEM, 0); |
| 336 | //gNextPacketPtr |= enc28j60ReadOp(ENC28J60\_READ\_BUF\_MEM, 0)<<8; |

|  |  |  |
| --- | --- | --- |
| 337 | // read the packet length (see datasheet page 43) | |
| 338 | len = enc28j60ReadBufferWord() - 4; |

|  |  |
| --- | --- |
| 339 | //len = enc28j60ReadOp(ENC28J60\_READ\_BUF\_MEM, 0); |
| 340 | //len |= enc28j60ReadOp(ENC28J60\_READ\_BUF\_MEM, 0)<<8; | |

|  |  |
| --- | --- |
| 341 | //len-=4; //remove the CRC count |
| 342 | // read the receive status (see datasheet page 43) | |

|  |  |
| --- | --- |
| 343 | rxstat  = enc28j60ReadBufferWord(); |
| 344 | //rxstat  = enc28j60ReadOp(ENC28J60\_READ\_BUF\_MEM, 0); | |

|  |  |  |
| --- | --- | --- |
| 345 | //rxstat |= ((uint16\_t)enc28j60ReadOp(ENC28J60\_READ\_BUF\_MEM, 0))<<8; | |
| 346 | // limit retrieve length |

|  |  |
| --- | --- |
| 347 | if (len>maxlen-1){ |
| 348 | len=maxlen-1; | |

|  |  |
| --- | --- |
| 349 | } |
| 350 | // check CRC and symbol errors (see datasheet page 44, table 7-3): | |

|  |  |  |
| --- | --- | --- |
| 351 | // The ERXFCON.CRCEN is set by default. Normally we should not | |
| 352 | // need to check this. |

|  |  |
| --- | --- |
| 353 | if ((rxstat & 0x80)==0){ |
| 354 | // invalid | |

|  |  |  |
| --- | --- | --- |
| 355 | len=0; | |
| 356 | }else{ |

|  |  |  |
| --- | --- | --- |
| 357 | // copy the packet from the receive buffer | |
| 358 | enc28j60ReadBuffer(len, packet); |

|  |  |
| --- | --- |
| 359 | } |
| 360 | // Move the RX read pointer to the start of the next received packet | |

|  |  |
| --- | --- |
| 361 | // This frees the memory we just read out |
| 362 | enc28j60WriteWord(ERXRDPTL, gNextPacketPtr ); | |

|  |  |  |
| --- | --- | --- |
| 363 | //enc28j60Write(ERXRDPTL, (gNextPacketPtr &0xFF)); | |
| 364 | //enc28j60Write(ERXRDPTH, (gNextPacketPtr)>>8); |

|  |  |  |
| --- | --- | --- |
| 365 | // However, compensate for the errata point 13, rev B4: enver write an even address! | |
| 366 | if ((gNextPacketPtr - 1 < RXSTART\_INIT) |

|  |  |
| --- | --- |
| 367 | || (gNextPacketPtr -1 > RXSTOP\_INIT)) { |
| 368 | enc28j60WriteWord(ERXRDPTL, RXSTOP\_INIT); |

|  |  |
| --- | --- |
| 369 | //enc28j60Write(ERXRDPTL, (RXSTOP\_INIT)&0xFF); |
| 370 | //enc28j60Write(ERXRDPTH, (RXSTOP\_INIT)>>8); |

|  |  |
| --- | --- |
| 371 | } else { |
| 372 | enc28j60WriteWord(ERXRDPTL, (gNextPacketPtr-1)); | |

|  |  |
| --- | --- |
| 373 | //enc28j60Write(ERXRDPTL, (gNextPacketPtr-1)&0xFF); |
| 374 | //enc28j60Write(ERXRDPTH, (gNextPacketPtr-1)>>8); |

|  |  |
| --- | --- |
| 375 | } |
| 376 | // decrement the packet counter indicate we are done with this packet | |

|  |  |  |
| --- | --- | --- |
| 377 | enc28j60WriteOp(ENC28J60\_BIT\_FIELD\_SET, ECON2, ECON2\_PKTDEC); | |
| 378 | return(len); |

|  |  |  |
| --- | --- | --- |
| 379 |  | |
| 380 | } |

ENC28J60.H

[view source](http://www.aiuxian.com/article/p-1652764.html#viewSource)

[print](http://www.aiuxian.com/article/p-1652764.html#printSource)[?](http://www.aiuxian.com/article/p-1652764.html#about)

|  |  |
| --- | --- |
| 001 | #ifndef \_\_ENC28J60\_H |
| 002 | #define \_\_ENC28J60\_H |

|  |  |
| --- | --- |
| 003 |  |
| 004 | #include "stm32f4xx.h" | |

|  |  |  |
| --- | --- | --- |
| 005 | #include "Delay.h" | |
| 006 |  |

|  |  |
| --- | --- |
| 007 | #define disableChip  GPIO\_SetBits(GPIOA,GPIO\_Pin\_4); delay\_us(2); |
| 008 | #define enableChip   GPIO\_ResetBits(GPIOA,GPIO\_Pin\_4); delay\_us(2); | |

|  |  |
| --- | --- |
| 009 |  |
| 010 |  |

|  |  |
| --- | --- |
| 011 | // ENC28J60 Control Registers |
| 012 | // Control register definitions are a combination of address, | |

|  |  |  |
| --- | --- | --- |
| 013 | // bank number, and Ethernet/MAC/PHY indicator bits. | |
| 014 | // - Register address        (bits 0-4) |

|  |  |
| --- | --- |
| 015 | // - Bank number        (bits 5-6) |
| 016 | // - MAC/PHY indicator        (bit 7) | |

|  |  |
| --- | --- |
| 017 | #define ADDR\_MASK        0x1F |
| 018 | #define BANK\_MASK        0x60 |

|  |  |  |
| --- | --- | --- |
| 019 | #define SPRD\_MASK        0x80 | |
| 020 | // All-bank registers |

|  |  |
| --- | --- |
| 021 | #define EIE              0x1B |
| 022 | #define EIR              0x1C |

|  |  |
| --- | --- |
| 023 | #define ESTAT            0x1D |
| 024 | #define ECON2            0x1E |

|  |  |  |
| --- | --- | --- |
| 025 | #define ECON1            0x1F | |
| 026 | // Bank 0 registers |

|  |  |
| --- | --- |
| 027 | #define ERDPTL           (0x00|0x00) |
| 028 | #define ERDPTH           (0x01|0x00) |

|  |  |
| --- | --- |
| 029 | #define EWRPTL           (0x02|0x00) |
| 030 | #define EWRPTH           (0x03|0x00) |

|  |  |
| --- | --- |
| 031 | #define ETXSTL           (0x04|0x00) |
| 032 | #define ETXSTH           (0x05|0x00) |

|  |  |
| --- | --- |
| 033 | #define ETXNDL           (0x06|0x00) |
| 034 | #define ETXNDH           (0x07|0x00) |

|  |  |
| --- | --- |
| 035 | #define ERXSTL           (0x08|0x00) |
| 036 | #define ERXSTH           (0x09|0x00) |

|  |  |
| --- | --- |
| 037 | #define ERXNDL           (0x0A|0x00) |
| 038 | #define ERXNDH           (0x0B|0x00) |

|  |  |
| --- | --- |
| 039 | #define ERXRDPTL         (0x0C|0x00) |
| 040 | #define ERXRDPTH         (0x0D|0x00) |

|  |  |
| --- | --- |
| 041 | #define ERXWRPTL         (0x0E|0x00) |
| 042 | #define ERXWRPTH         (0x0F|0x00) |

|  |  |
| --- | --- |
| 043 | #define EDMASTL          (0x10|0x00) |
| 044 | #define EDMASTH          (0x11|0x00) |

|  |  |
| --- | --- |
| 045 | #define EDMANDL          (0x12|0x00) |
| 046 | #define EDMANDH          (0x13|0x00) |

|  |  |
| --- | --- |
| 047 | #define EDMADSTL         (0x14|0x00) |
| 048 | #define EDMADSTH         (0x15|0x00) |

|  |  |
| --- | --- |
| 049 | #define EDMACSL          (0x16|0x00) |
| 050 | #define EDMACSH          (0x17|0x00) |

|  |  |
| --- | --- |
| 051 | // Bank 1 registers |
| 052 | #define EHT0             (0x00|0x20) | |

|  |  |
| --- | --- |
| 053 | #define EHT1             (0x01|0x20) |
| 054 | #define EHT2             (0x02|0x20) |

|  |  |
| --- | --- |
| 055 | #define EHT3             (0x03|0x20) |
| 056 | #define EHT4             (0x04|0x20) |

|  |  |
| --- | --- |
| 057 | #define EHT5             (0x05|0x20) |
| 058 | #define EHT6             (0x06|0x20) |

|  |  |  |
| --- | --- | --- |
| 059 | #define EHT7             (0x07|0x20) | |
| 060 | #define EPMM0            (0x08|0x20) |

|  |  |
| --- | --- |
| 061 | #define EPMM1            (0x09|0x20) |
| 062 | #define EPMM2            (0x0A|0x20) |

|  |  |
| --- | --- |
| 063 | #define EPMM3            (0x0B|0x20) |
| 064 | #define EPMM4            (0x0C|0x20) |

|  |  |
| --- | --- |
| 065 | #define EPMM5            (0x0D|0x20) |
| 066 | #define EPMM6            (0x0E|0x20) |

|  |  |  |
| --- | --- | --- |
| 067 | #define EPMM7            (0x0F|0x20) | |
| 068 | #define EPMCSL           (0x10|0x20) |

|  |  |
| --- | --- |
| 069 | #define EPMCSH           (0x11|0x20) |
| 070 | #define EPMOL            (0x14|0x20) | |

|  |  |  |
| --- | --- | --- |
| 071 | #define EPMOH            (0x15|0x20) | |
| 072 | #define EWOLIE           (0x16|0x20) |

|  |  |  |
| --- | --- | --- |
| 073 | #define EWOLIR           (0x17|0x20) | |
| 074 | #define ERXFCON          (0x18|0x20) |

|  |  |  |
| --- | --- | --- |
| 075 | #define EPKTCNT          (0x19|0x20) | |
| 076 | // Bank 2 registers |

|  |  |
| --- | --- |
| 077 | #define MACON1           (0x00|0x40|0x80) |
| 078 | #define MACON2           (0x01|0x40|0x80) |

|  |  |
| --- | --- |
| 079 | #define MACON3           (0x02|0x40|0x80) |
| 080 | #define MACON4           (0x03|0x40|0x80) |

|  |  |
| --- | --- |
| 081 | #define MABBIPG          (0x04|0x40|0x80) |
| 082 | #define MAIPGL           (0x06|0x40|0x80) | |

|  |  |  |
| --- | --- | --- |
| 083 | #define MAIPGH           (0x07|0x40|0x80) | |
| 084 | #define MACLCON1         (0x08|0x40|0x80) |

|  |  |
| --- | --- |
| 085 | #define MACLCON2         (0x09|0x40|0x80) |
| 086 | #define MAMXFLL          (0x0A|0x40|0x80) | |

|  |  |
| --- | --- |
| 087 | #define MAMXFLH          (0x0B|0x40|0x80) |
| 088 | #define MAPHSUP          (0x0D|0x40|0x80) |

|  |  |
| --- | --- |
| 089 | #define MICON            (0x11|0x40|0x80) |
| 090 | #define MICMD            (0x12|0x40|0x80) |

|  |  |
| --- | --- |
| 091 | #define MIREGADR         (0x14|0x40|0x80) |
| 092 | #define MIWRL            (0x16|0x40|0x80) | |

|  |  |
| --- | --- |
| 093 | #define MIWRH            (0x17|0x40|0x80) |
| 094 | #define MIRDL            (0x18|0x40|0x80) |

|  |  |  |
| --- | --- | --- |
| 095 | #define MIRDH            (0x19|0x40|0x80) | |
| 096 | // Bank 3 registers |

|  |  |
| --- | --- |
| 097 | #define MAADR1           (0x00|0x60|0x80) |
| 098 | #define MAADR0           (0x01|0x60|0x80) |

|  |  |
| --- | --- |
| 099 | #define MAADR3           (0x02|0x60|0x80) |
| 100 | #define MAADR2           (0x03|0x60|0x80) |

|  |  |
| --- | --- |
| 101 | #define MAADR5           (0x04|0x60|0x80) |
| 102 | #define MAADR4           (0x05|0x60|0x80) |

|  |  |  |
| --- | --- | --- |
| 103 | #define EBSTSD           (0x06|0x60) | |
| 104 | #define EBSTCON          (0x07|0x60) |

|  |  |
| --- | --- |
| 105 | #define EBSTCSL          (0x08|0x60) |
| 106 | #define EBSTCSH          (0x09|0x60) |

|  |  |  |
| --- | --- | --- |
| 107 | #define MISTAT           (0x0A|0x60|0x80) | |
| 108 | #define EREVID           (0x12|0x60) |

|  |  |  |
| --- | --- | --- |
| 109 | #define ECOCON           (0x15|0x60) | |
| 110 | #define EFLOCON          (0x17|0x60) |

|  |  |
| --- | --- |
| 111 | #define EPAUSL           (0x18|0x60) |
| 112 | #define EPAUSH           (0x19|0x60) |

|  |  |
| --- | --- |
| 113 | // PHY registers |
| 114 | #define PHCON1           0x00 | |

|  |  |
| --- | --- |
| 115 | #define PHSTAT1          0x01 |
| 116 | #define PHHID1           0x02 | |

|  |  |
| --- | --- |
| 117 | #define PHHID2           0x03 |
| 118 | #define PHCON2           0x10 |

|  |  |
| --- | --- |
| 119 | #define PHSTAT2          0x11 |
| 120 | #define PHIE             0x12 | |

|  |  |  |
| --- | --- | --- |
| 121 | #define PHIR             0x13 | |
| 122 | #define PHLCON           0x14 |

|  |  |
| --- | --- |
| 123 |  |
| 124 | // ENC28J60 ERXFCON Register Bit Definitions | |

|  |  |  |
| --- | --- | --- |
| 125 | #define ERXFCON\_UCEN     0x80 | |
| 126 | #define ERXFCON\_ANDOR    0x40 |

|  |  |
| --- | --- |
| 127 | #define ERXFCON\_CRCEN    0x20 |
| 128 | #define ERXFCON\_PMEN     0x10 | |

|  |  |
| --- | --- |
| 129 | #define ERXFCON\_MPEN     0x08 |
| 130 | #define ERXFCON\_HTEN     0x04 |

|  |  |
| --- | --- |
| 131 | #define ERXFCON\_MCEN     0x02 |
| 132 | #define ERXFCON\_BCEN     0x01 |

|  |  |  |
| --- | --- | --- |
| 133 | // ENC28J60 EIE Register Bit Definitions | |
| 134 | #define EIE\_INTIE        0x80 |

|  |  |
| --- | --- |
| 135 | #define EIE\_PKTIE        0x40 |
| 136 | #define EIE\_DMAIE        0x20 |

|  |  |
| --- | --- |
| 137 | #define EIE\_LINKIE       0x10 |
| 138 | #define EIE\_TXIE         0x08 | |

|  |  |  |
| --- | --- | --- |
| 139 | #define EIE\_WOLIE        0x04 | |
| 140 | #define EIE\_TXERIE       0x02 |

|  |  |
| --- | --- |
| 141 | #define EIE\_RXERIE       0x01 |
| 142 | // ENC28J60 EIR Register Bit Definitions | |

|  |  |
| --- | --- |
| 143 | #define EIR\_PKTIF        0x40 |
| 144 | #define EIR\_DMAIF        0x20 |

|  |  |
| --- | --- |
| 145 | #define EIR\_LINKIF       0x10 |
| 146 | #define EIR\_TXIF         0x08 | |

|  |  |  |
| --- | --- | --- |
| 147 | #define EIR\_WOLIF        0x04 | |
| 148 | #define EIR\_TXERIF       0x02 |

|  |  |
| --- | --- |
| 149 | #define EIR\_RXERIF       0x01 |
| 150 | // ENC28J60 ESTAT Register Bit Definitions | |

|  |  |  |
| --- | --- | --- |
| 151 | #define ESTAT\_INT        0x80 | |
| 152 | #define ESTAT\_LATECOL    0x10 |

|  |  |
| --- | --- |
| 153 | #define ESTAT\_RXBUSY     0x04 |
| 154 | #define ESTAT\_TXABRT     0x02 |

|  |  |
| --- | --- |
| 155 | #define ESTAT\_CLKRDY     0x01 |
| 156 | // ENC28J60 ECON2 Register Bit Definitions | |

|  |  |
| --- | --- |
| 157 | #define ECON2\_AUTOINC    0x80 |
| 158 | #define ECON2\_PKTDEC     0x40 | |

|  |  |
| --- | --- |
| 159 | #define ECON2\_PWRSV      0x20 |
| 160 | #define ECON2\_VRPS       0x08 | |

|  |  |  |
| --- | --- | --- |
| 161 | // ENC28J60 ECON1 Register Bit Definitions | |
| 162 | #define ECON1\_TXRST      0x80 |

|  |  |
| --- | --- |
| 163 | #define ECON1\_RXRST      0x40 |
| 164 | #define ECON1\_DMAST      0x20 |

|  |  |
| --- | --- |
| 165 | #define ECON1\_CSUMEN     0x10 |
| 166 | #define ECON1\_TXRTS      0x08 | |

|  |  |  |
| --- | --- | --- |
| 167 | #define ECON1\_RXEN       0x04 | |
| 168 | #define ECON1\_BSEL1      0x02 |

|  |  |
| --- | --- |
| 169 | #define ECON1\_BSEL0      0x01 |
| 170 | // ENC28J60 MACON1 Register Bit Definitions | |

|  |  |
| --- | --- |
| 171 | #define MACON1\_LOOPBK    0x10 |
| 172 | #define MACON1\_TXPAUS    0x08 |

|  |  |  |
| --- | --- | --- |
| 173 | #define MACON1\_RXPAUS    0x04 | |
| 174 | #define MACON1\_PASSALL   0x02 |

|  |  |
| --- | --- |
| 175 | #define MACON1\_MARXEN    0x01 |
| 176 | // ENC28J60 MACON2 Register Bit Definitions | |

|  |  |  |
| --- | --- | --- |
| 177 | #define MACON2\_MARST     0x80 | |
| 178 | #define MACON2\_RNDRST    0x40 |

|  |  |
| --- | --- |
| 179 | #define MACON2\_MARXRST   0x08 |
| 180 | #define MACON2\_RFUNRST   0x04 |

|  |  |
| --- | --- |
| 181 | #define MACON2\_MATXRST   0x02 |
| 182 | #define MACON2\_TFUNRST   0x01 |

|  |  |  |
| --- | --- | --- |
| 183 | // ENC28J60 MACON3 Register Bit Definitions | |
| 184 | #define MACON3\_PADCFG2   0x80 |

|  |  |
| --- | --- |
| 185 | #define MACON3\_PADCFG1   0x40 |
| 186 | #define MACON3\_PADCFG0   0x20 |

|  |  |
| --- | --- |
| 187 | #define MACON3\_TXCRCEN   0x10 |
| 188 | #define MACON3\_PHDRLEN   0x08 |

|  |  |
| --- | --- |
| 189 | #define MACON3\_HFRMLEN   0x04 |
| 190 | #define MACON3\_FRMLNEN   0x02 |

|  |  |
| --- | --- |
| 191 | #define MACON3\_FULDPX    0x01 |
| 192 | // ENC28J60 MICMD Register Bit Definitions | |

|  |  |
| --- | --- |
| 193 | #define MICMD\_MIISCAN    0x02 |
| 194 | #define MICMD\_MIIRD      0x01 | |

|  |  |  |
| --- | --- | --- |
| 195 | // ENC28J60 MISTAT Register Bit Definitions | |
| 196 | #define MISTAT\_NVALID    0x04 |

|  |  |
| --- | --- |
| 197 | #define MISTAT\_SCAN      0x02 |
| 198 | #define MISTAT\_BUSY      0x01 |

|  |  |  |
| --- | --- | --- |
| 199 | // ENC28J60 PHY PHCON1 Register Bit Definitions | |
| 200 | #define PHCON1\_PRST      0x8000 |

|  |  |
| --- | --- |
| 201 | #define PHCON1\_PLOOPBK   0x4000 |
| 202 | #define PHCON1\_PPWRSV    0x0800 | |

|  |  |
| --- | --- |
| 203 | #define PHCON1\_PDPXMD    0x0100 |
| 204 | // ENC28J60 PHY PHSTAT1 Register Bit Definitions | |

|  |  |
| --- | --- |
| 205 | #define PHSTAT1\_PFDPX    0x1000 |
| 206 | #define PHSTAT1\_PHDPX    0x0800 |

|  |  |
| --- | --- |
| 207 | #define PHSTAT1\_LLSTAT   0x0004 |
| 208 | #define PHSTAT1\_JBSTAT   0x0002 |

|  |  |  |
| --- | --- | --- |
| 209 | // ENC28J60 PHY PHCON2 Register Bit Definitions | |
| 210 | #define PHCON2\_FRCLINK   0x4000 |

|  |  |  |
| --- | --- | --- |
| 211 | #define PHCON2\_TXDIS     0x2000 | |
| 212 | #define PHCON2\_JABBER    0x0400 |

|  |  |  |
| --- | --- | --- |
| 213 | #define PHCON2\_HDLDIS    0x0100 | |
| 214 |  |

|  |  |  |
| --- | --- | --- |
| 215 | // ENC28J60 Packet Control Byte Bit Definitions | |
| 216 | #define PKTCTRL\_PHUGEEN  0x08 |

|  |  |
| --- | --- |
| 217 | #define PKTCTRL\_PPADEN   0x04 |
| 218 | #define PKTCTRL\_PCRCEN   0x02 |

|  |  |  |
| --- | --- | --- |
| 219 | #define PKTCTRL\_POVERRIDE 0x01 | |
| 220 |  |

|  |  |
| --- | --- |
| 221 | // SPI operation codes |
| 222 | #define ENC28J60\_READ\_CTRL\_REG       0x00 | |

|  |  |  |
| --- | --- | --- |
| 223 | #define ENC28J60\_READ\_BUF\_MEM        0x3A | |
| 224 | #define ENC28J60\_WRITE\_CTRL\_REG      0x40 |

|  |  |
| --- | --- |
| 225 | #define ENC28J60\_WRITE\_BUF\_MEM       0x7A |
| 226 | #define ENC28J60\_BIT\_FIELD\_SET       0x80 |

|  |  |
| --- | --- |
| 227 | #define ENC28J60\_BIT\_FIELD\_CLR       0xA0 |
| 228 | #define ENC28J60\_SOFT\_RESET          0xFF | |

|  |  |
| --- | --- |
| 229 |  |
| 230 |  |

|  |  |  |
| --- | --- | --- |
| 231 | // The RXSTART\_INIT should be zero. See Rev. B4 Silicon Errata | |
| 232 | // buffer boundaries applied to internal 8K ram |

|  |  |  |
| --- | --- | --- |
| 233 | // the entire available packet buffer space is allocated | |
| 234 | // |

|  |  |
| --- | --- |
| 235 | // start with recbuf at 0/ |
| 236 | #define RXSTART\_INIT     0x0 | |

|  |  |
| --- | --- |
| 237 | // receive buffer end |
| 238 | #define RXSTOP\_INIT      (0x1FFF-0x0600-1) | |

|  |  |  |
| --- | --- | --- |
| 239 | // start TX buffer at 0x1FFF-0x0600, pace for one full ethernet frame (~1500 bytes) | |
| 240 | #define TXSTART\_INIT     (0x1FFF-0x0600) |

|  |  |
| --- | --- |
| 241 | // stp TX buffer at end of mem |
| 242 | #define TXSTOP\_INIT      0x1FFF | |

|  |  |
| --- | --- |
| 243 | // |
| 244 | // max frame length which the conroller will accept: | |

|  |  |  |  |
| --- | --- | --- | --- |
| 245 | #define        MAX\_FRAMELEN        1500        // (note: maximum ethernet frame length would be 1518) | | |
| 246 | | //#define MAX\_FRAMELEN     600 |

|  |  |
| --- | --- |
| 247 |  |
| 248 | uint8\_t enc28j60ReadOp(uint8\_t op, uint8\_t address); | |

|  |  |
| --- | --- |
| 249 |  |
| 250 |  |

|  |  |
| --- | --- |
| 251 | // functions |
| 252 | extern uint8\_t enc28j60ReadOp(uint8\_t op, uint8\_t address); | |

|  |  |  |
| --- | --- | --- |
| 253 | extern void enc28j60WriteOp(uint8\_t op, uint8\_t address, uint8\_t data); | |
| 254 | extern void enc28j60ReadBuffer(uint16\_t len, uint8\_t\* data); |

|  |  |  |
| --- | --- | --- |
| 255 | extern void enc28j60WriteBuffer(uint16\_t len, uint8\_t\* data); | |
| 256 | extern void enc28j60SetBank(uint8\_t address); |

|  |  |
| --- | --- |
| 257 | extern uint8\_t enc28j60Read(uint8\_t address); |
| 258 | extern void enc28j60Write(uint8\_t address, uint8\_t data); | |

|  |  |  |
| --- | --- | --- |
| 259 | extern void enc28j60PhyWrite(uint8\_t address, uint16\_t data); | |
| 260 | extern void enc28j60clkout(uint8\_t clk); |

|  |  |
| --- | --- |
| 261 | extern void enc28j60SpiInit(void);////// |
| 262 | extern void enc28j60Init(uint8\_t\* macaddr); | |

|  |  |
| --- | --- |
| 263 | extern void enc28j60PacketSend(uint16\_t len, uint8\_t\* packet); |
| 264 | extern uint16\_t enc28j60PacketReceive(uint16\_t maxlen, uint8\_t\* packet); | |

|  |  |  |
| --- | --- | --- |
| 265 | extern uint8\_t enc28j60getrev(void); | |
| 266 |  |

|  |  |  |
| --- | --- | --- |
| 267 | extern uint8\_t enc28j60hasRxPkt(void); | |
| 268 | extern uint8\_t enc28j60linkup(void); |

|  |  |
| --- | --- |
| 269 | extern void enc28j60EnableBroadcast( void ); |
| 270 | extern void enc28j60DisableBroadcast( void ); | |

|  |  |
| --- | --- |
| 271 | extern void enc28j60EnableMulticast( void ); |
| 272 | extern void enc28j60DisableMulticast( void ); | |

|  |  |  |
| --- | --- | --- |
| 273 | extern void enc28j60PowerDown(void); | |
| 274 | extern void enc28j60PowerUp(void); |

|  |  |
| --- | --- |
| 275 |  |
| 276 | #endif /\* \_\_ENC28J60\_H \*/ | |