```
// ============================================================================ BOF
// FILE FUNC:  receive a sequence of ASCII characters over the cc2500
//             (at 3-chars per RF packet). Then print them on the host PC
// ----------------------------------------------------------------------------

#include "stdint.h"               // MSP430 data type definitions
#include "msp430x22x4.h"          // Mammoth MSP430-specific macros & defs
#include "wireless-v2.h"          // Mammoth cc2500 setup & function defs


//-----------------------------------------------------------------------------
// Func:  Packet Received ISR:  triggered by falling edge of GDO0.
//        Parses pkt & prints entire packet data bytes to the host PC.
// Args:  None
// Retn:  None
//-----------------------------------------------------------------------------
#pragma vector=PORT2_VECTOR       // cc2500 pin GDO0 is hardwired to P2.6
__interrupt void PktRxedISR(void)
{
  // Buffer Len for rxPkt = only data bytes;
  // pkt size byte not incl b/c it is stripped off by RFReceivePacket func.
  // IF address check was enabled, THEN would need 1 more byte for device addr

  static uint8_t len = 4;         // Packet Len = 3 bytes (data only)
  uint8_t status[2];              // Buffer to store pkt status bytes
  uint8_t crcOk = 0;              // Flag that pkt was received w/ good CRC

  if(TI_CC_GDO0_PxIFG & TI_CC_GDO0_PIN)           // chk GDO0 bit of P2 IFG Reg
  { crcOk = RFReceivePacket(rxPkt, &len, status);  // Get packet from cc2500
    TI_CC_GDO0_PxIFG &= ~TI_CC_GDO0_PIN;          // Reset GDO0 IRQ flag
  }

  if(crcOk)                             // If RXed pkt is valid
    P1OUT ^= 0x03;                      // Pkt RXed => Toggle LEDs

  TI_CC_SPIStrobe(TI_CCxxx0_SIDLE);     // Set cc2500 to idle mode.
  TI_CC_SPIStrobe(TI_CCxxx0_SRX);       // Set cc2500 to RX mode.
                                        // AutoCal @ IDLE to RX Transition

  __bic_SR_register_on_exit (CPUOFF);   // wake CPU on exit so it can print


//-----------------------------------------------------------------------------
// FUNC:  Setup MSP430 Ports & Clocks, reset & config cc2500
// ARGS:  none
// RETN:  none
//-----------------------------------------------------------------------------
void SetupAll(void)
{
  volatile uint16_t delay;
```

```
  for(delay=0; delay<650; delay++);     // Empirical: cc2500 Pwr up settle

  // set up clock system
  BCSCTL1 = CALBC1_8MHZ;                // set DCO = 8MHz
  DCOCTL  = CALDCO_8MHZ;                // set DCO = 8MHz
  BCSCTL2 |= DIVS_3;                    // SMCLK = MCLK/8 = 1MHz

  // LED Port config
  P1DIR |=  0x03;                       // Set both LED pins to output
  P1OUT &= ~0x03;                       // Clear both LEDs

  // Wireless Initialization
  P2SEL = 0;                            // P2.6, P2.7 = GDO0, GDO2 (GPIO)
  TI_CC_SPISetup();                     // Initialize SPI port
  TI_CC_PowerupResetCCxxxx();           // Reset cc2500
  writeRFSettings();                    // Send RF settings to config regs

  TI_CC_GDO0_PxIES |=  TI_CC_GDO0_PIN;  // IRQ on GDO0 fall. edge (end of pkt)
  TI_CC_GDO0_PxIFG &= ~TI_CC_GDO0_PIN;  // Clear  GDO0 IRQ flag
  TI_CC_GDO0_PxIE  |=  TI_CC_GDO0_PIN;  // Enable GDO0 IRQ
  TI_CC_SPIStrobe(TI_CCxxx0_SRX);       // Init. cc2500 in RX mode.

  TI_CC_SPIWriteReg(TI_CCxxx0_CHANNR, 0);    // Set Your Own Channel Number
                                             // AFTER writeRFSettings (???)

  for(delay=0; delay<650; delay++);     // Empirical: Let cc2500 finish setup

  P1OUT = 0x02;                         // Setup done => Turn on green LED
}

// -------------------------------------------------------------------------
void main()
{ WDTCTL = WDTPW + WDTHOLD;         // halt watchdog

  volatile uint8_t rxPkt[4];        // Buffer to store received pkt payload
  uint8_t str[8];                   // Buffer to parse payload into ASCII str
  uint8_t i = 0;                    // Index for rxPkt
  uint8_t j = 0;                    // Index for str
  SetupAll();                       // Setup clocks, ports & cc2500 settings
  while (1)
  {
    __bis_SR_register(CPUOFF + GIE);                      // sleep til Pkt RX
    for (i = 0; i < 4; i ++)
    {
      j = i + i;                                  // Calc. index of str
      str[j]   = rxPkt[i] & 0x0F;                 // Get first nybble of byte
      str[j+1] = (rxPkt[i] & 0xF0 >> 4) & 0x0F;  // Get second nybble of byte
    }
```

```
    // Convert hex to corresponding ASCII value
    for (j = 0; j < 8; j ++)
    {
      if (str[j] < 10)  // if the nybble is a number
      {
        str[j] += 0x30; // add 48
      }
      else              // if the nybble is a letter
      {
        str[j] += 0x37; // add 55
      }
    }

    printf ("%c%c%c%c%c%c%c%c\n", str[0], str[1], str[2], str[3],
                                  str[4], str[5], str[6], str[7]);// print str
  }
}

// ============================================================================ EOF
```