```
//=========================================================================== BOF
// FILE FUNC: Generate a sequence of bytes & TRANSMIT them out the cc2500.
//            Subsequent Transmissions are triggered by a roll-over timer.
//---------------------------------------------------------------------------

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

typedef union
{ uint8_t  u8[4];
  uint32_t u32;
} Uint32Payload_t;

//---------------------------------------------------------------------------
// Func:  Timer A ISR  - Just Pull CPU out of sleep mode to retn to next cmd
// Args:  None
// Retn:  None
//---------------------------------------------------------------------------
#pragma vector=TIMERA0_VECTOR
__interrupt void TimerAISR(void)
{
  __bic_SR_register_on_exit(CPUOFF);     // wake up on exit

} // end function "TimerAISR"


// ---------------------------------------------------------------------------
// Func:  Forms appropriate data packet and uses canned wireless function
//        RFSendPacket to send the data to the CC2500 chip and have it
//        transmit the data.
// Args:  none
// Retn:  none
// ---------------------------------------------------------------------------
void TxRfPacket(void)
{
   P1OUT ^= 0x03;                      // Start of TX => toggle LEDs
   static uint8_t pktLen = 5;          // Fix Packet size
   static Uint32Payload_t pld.u32 = 0x0000000F;
   static uint8_t pktData[5] = {0x04,pld.u8[0],pld.u8[1],pld.u8[2],pld.u8[3]};
   // Contents:   pktData[4] = {Pyld leng, data, data, data, data}
   // Note: If addr-mode was enabled, then 2nd byte would be dev. addr, not data

   RFSendPacket(pktData, pktLen);      // Activate TX mode & transmit the packet

   // Double the value of the payload until it equals 0xF0000000
   if (pld.u32 == 0xF0000000)
   {
     pld.u32 = 0x0000000F;
```

```
   }
   else
   {
     pld.u32 *= 2;
   }

   TI_CC_SPIStrobe(TI_CCxxx0_SIDLE);   // Set cc2500 to IDLE mode.
                                       // TX mode re-activates in RFSendPacket
                                       // w/ AutoCal @ IDLE to TX Transition
}


//-------------------------------------------------------------------------------
// Func:  Setup MSP430 Ports & Clocks, and reset & config cc2500 chip
// 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

  // TimerA config
  TACTL   = TASSEL_2 |ID_3 | MC_1;      // TA uses SMCLK/8, in Up mode
  TACCR0  = 60000;                      // ~480 msec @  1/8 MHz
  TACCTL0 = CCIE;                               // enable TA CCR0 IRQ

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

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

  TI_CC_SPIWriteReg(TI_CCxxx0_CHANNR,  0);  // Set Your Own Channel Number
                                            // only 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
  SetupAll();                // Setup all ports and wireless settings
  while(1)                   // Tx one data packet at each timer A IRQ
  {
    TxRfPacket();                       // Tx one data packet
    __bis_SR_register(CPUOFF + GIE);    // sleep until next timer A IRQ
  }
}
// ======================================================================== EOF
```