```
Reference oscillator frequency      40MHz
Symbol rate                         4ksps
Frequency deviation                 1kHz
RF frequency                        228.4MHz


#define IOCFG3_REG                          0x00
#define IOCFG3                              0x1A

#define IOCFG2_REG                          0x01
#define IOCFG2                              0x06

#define IOCFG0_REG                          0x03
#define IOCFG0                              0x02        //TXFIFO_THR

#define SYNC3_REG                           0x04
#define SYNC3                               0x93        //Sync word 3 = 0x93

#define SYNC2_REG                           0x05
#define SYNC2                               0x0B        //Sync word 2 = 0x0B

#define SYNC1_REG                           0x06
#define SYNC1                               0x51        //Sync word 1 = 0x51

#define SYNC0_REG                           0x07
#define SYNC0                               0xDE        //Sync word 0 = 0xDE

#define SYNC_CFG1_REG                       0x08
#define SYNC_CFG1                           0x18

#define DEVIATION_M_REG                     0x0A
#define DEVIATION_M                         0xD2        //Freq deviation 1kHz.

#define MODCFG_DEV_E_REG                    0x0B
#define MODCFG_DEV_E                        0x08        //2GFSK, normal mode. DEV_E = 0.

#define PREAMBLE_CFG1_REG                   0x0D
#define PREAMBLE_CFG1                       0x18        //Pattern 0xAA.Bytes = 4.
```

```
#define CHAN_BW_REG                     0x11
#define CHAN_BW                         0x08        //Decimation factor 20.

#define SYMBOL_RATE2_REG                0x14        //Data rate
#define SYMBOL_RATE2                    0x5A        //4ksps. SRATE_E = 5.

#define SYMBOL_RATE1_REG                0x15        //Data rate
#define SYMBOL_RATE1                    0x36        //4ksps

#define SYMBOL_RATE0_REG                0x16        //Data rate
#define SYMBOL_RATE0                    0xE2        //4ksps

#define FIFO_CFG_REG                    0x1E
#define FIFO_CFG                        0x78        //120bytes in TX FIFO.

#define FS_CFG_REG                      0x21
#define FS_CFG                          0x18        //205MHz to 240MHz. LO divider = 16.

#define PKT_CFG1_REG                    0x27
#define PKT_CFG1                        0x04        //CRC enabled. Status byte not appended.

#define PKT_CFG0_REG                    0x28
#define PKT_CFG0                        0x40        //Infinite packet length mode.

#define PA_CFG0_REG                     0x2D
#define PA_CFG0                         0x7E        //Tx up sampler factor = 64.

#define PKT_LEN_REG                     0x2E
#define PKT_LEN                         0xF6        //246 bytes.
#define PKT_LEN_UNMODULATED             0x00

#define FREQ2_REG                       0x0C        //Extended register.
#define FREQ2                           0x5B        //228.4MHz

#define FREQ1_REG                       0x0D        //Extended register.
#define FREQ1                           0x5C        //228.4MHz

#define FREQ0_REG                       0x0E        //Extended register.
#define FREQ0                           0x28        //228.4MHz
```

```
#define FS_SPARE_REG                    0x22        //Extended register.
#define FS_SPARE                        0xAC

#define TXFIRST_REG                     0xD3        //Extended register.
#define TXFIRST                         0x00        //Default.

#define TXLAST_REG                      0xD5        //Extended register.
#define TXLAST                          0x00        //Default.


#define PARTNUMBER                      0x8F        //Extended register.
#define PARTVERSION                     0x90        //Extended register.

#define PACKET_LENGTH                   126         //Bytes
#define EXTENDED_REG                    0x2F
#define DIRECT_TX_FIFO_REG              0x3E        //Tx FIFO direct access.
#define SRES                            0x30        //Reset the frequency sysnthesiser.
#define SFSTXON                         0x31        //Enable and calibrate frequency sysnthesiser.
#define SCAL                            0x33        //Calibrate frequency synthesiser.
#define STX                             0x35        //Enable TX.
#define SIDLE                           0x36        //Disable TX.
#define SFTX                            0x3B        //Flush TX FIFO.
#define SNOP                            0x3D        //No operation. Get chip status byte.
```

```
CY_ISR(Tx_FIFO_thrshold_isr_Interrupt_Handler)
{
    Tx_FIFO_threshold_isr_ClearPending();
    RF_GPIO_0_ClearInterrupt();

    SPI_SpiUartPutArray(txBuffer, 122);                         //Load the data in the CC1125 TX FIFO.
    RF_Tx_Start();
}


int main(void)
{
    uint8_t  result;

    RF_SPI_Write(SIDLE, 0x00, 0x00, SPI_TRANSACTION_TYPE_WRITE);        //Set RF synthesiser in the IDLE state.

    while(Initialise_RF() == FALSE)
    {
        Fault_LED_Write(ILLUMINATED);
        CyDelay(DELAY_250ms);
        Debug_UartPutString("\n\rNon poi inizializzare sintetizzatore RF.\n\r");
    }

    Debug_UartPutString("\n\rSintetizzatore RF inizializzato con successo.\n\r");

    Manual_Calibration();

    result = RF_SPI_Read(EXTENDED_REG, FS_CHP_REG, SPI_TRANSACTION_TYPE_READ);
    result = result + 0x0C;
    RF_SPI_Write(EXTENDED_REG, FS_CHP_REG, result, SPI_TRANSACTION_TYPE_WRITE);

    RF_SPI_Write(EXTENDED_REG, FS_VCO4_REG, 0x0A, SPI_TRANSACTION_TYPE_WRITE);

    Tx_FIFO_threshold_isr_StartEx(Tx_FIFO_thrshold_isr_Interrupt_Handler);

    while(TRUE)
    {

    }
```

```
uint8_t        Initialise_RF(void)          //Use CC1125 default values for registers that are not listed below.
{
    uint16    index;

    RF_Reset_Write(RESET);                                                  //Reset the CC1125.
    CyDelay(DELAY_10ms);
    RF_Reset_Write(ACTIVE);

    Status.flags = MODULATED_CARRIER;

    txBuffer[0] = DIRECT_TX_FIFO_REG | 0x40;                                //Burst mode.
    txBuffer[1] = 0x00;                                                     //FIFO address 0.
    txBuffer[2] = PACKET_LENGTH;

    for(index = 3; index < PACKET_LENGTH + 4; index++)                      //Generate the data to be transmitted.
    {
        txBuffer[index] = 0x5A;
    }

    RF_SPI_Write(SFTX, 0x00, 0x00, SPI_TRANSACTION_TYPE_WRITE);

    SPI_SpiUartPutArray(txBuffer, PACKET_LENGTH + 3);                       //Load the data in the CC1125 TX FIFO.

    RF_SPI_Write(IOCFG3_REG, IOCFG3, 0x00, SPI_TRANSACTION_TYPE_WRITE);

    RF_SPI_Write(IOCFG2_REG, IOCFG2, 0x00, SPI_TRANSACTION_TYPE_WRITE);

    RF_SPI_Write(IOCFG0_REG, IOCFG0, 0x00, SPI_TRANSACTION_TYPE_WRITE);

    RF_SPI_Write(SYNC3_REG, SYNC3, 0x00, SPI_TRANSACTION_TYPE_WRITE);

    RF_SPI_Write(SYNC2_REG, SYNC2, 0x00, SPI_TRANSACTION_TYPE_WRITE);

    RF_SPI_Write(SYNC1_REG, SYNC1, 0x00, SPI_TRANSACTION_TYPE_WRITE);

    RF_SPI_Write(SYNC0_REG, SYNC0, 0x00, SPI_TRANSACTION_TYPE_WRITE);

    RF_SPI_Write(SYNC_CFG1_REG, SYNC_CFG1, 0x00, SPI_TRANSACTION_TYPE_WRITE);
```

```
RF_SPI_Write(DEVIATION_M_REG, DEVIATION_M, 0x00, SPI_TRANSACTION_TYPE_WRITE);

RF_SPI_Write(MODCFG_DEV_E_REG, MODCFG_DEV_E, 0x00, SPI_TRANSACTION_TYPE_WRITE);          //2GFSK.

RF_SPI_Write(PREAMBLE_CFG1_REG, PREAMBLE_CFG1, 0x00, SPI_TRANSACTION_TYPE_WRITE);

RF_SPI_Write(CHAN_BW_REG, CHAN_BW, 0x00, SPI_TRANSACTION_TYPE_WRITE);

RF_SPI_Write(SYMBOL_RATE2_REG, SYMBOL_RATE2, 0x00, SPI_TRANSACTION_TYPE_WRITE);          //Symbol rate.

RF_SPI_Write(SYMBOL_RATE1_REG, SYMBOL_RATE1, 0x00, SPI_TRANSACTION_TYPE_WRITE);          //Symbol rate.

RF_SPI_Write(SYMBOL_RATE0_REG, SYMBOL_RATE0, 0x00, SPI_TRANSACTION_TYPE_WRITE);          //Symbol rate.

RF_SPI_Write(FIFO_CFG_REG, FIFO_CFG, 0x00, SPI_TRANSACTION_TYPE_WRITE);

RF_SPI_Write(FS_CFG_REG, FS_CFG, 0x00, SPI_TRANSACTION_TYPE_WRITE);                    //205MHz to 240MHz.

RF_SPI_Write(PKT_CFG1_REG, PKT_CFG1, 0x00, SPI_TRANSACTION_TYPE_WRITE);

RF_SPI_Write(PKT_CFG0_REG, PKT_CFG0, 0x00, SPI_TRANSACTION_TYPE_WRITE);

RF_SPI_Write(PKT_LEN_REG, PKT_LEN, 0x00, SPI_TRANSACTION_TYPE_WRITE);

RF_SPI_Write(EXTENDED_REG, FREQ2_REG, FREQ2, SPI_TRANSACTION_TYPE_WRITE);

RF_SPI_Write(EXTENDED_REG, FREQ1_REG, FREQ1, SPI_TRANSACTION_TYPE_WRITE);

RF_SPI_Write(EXTENDED_REG, FREQ0_REG, FREQ0, SPI_TRANSACTION_TYPE_WRITE);

RF_SPI_Write(EXTENDED_REG, FS_SPARE_REG, FS_SPARE, SPI_TRANSACTION_TYPE_WRITE);

RF_SPI_Read(EXTENDED_REG, PARTVERSION, SPI_TRANSACTION_TYPE_READ);
sprintf(Debug_message_array, "\n\rVersione sintetizzatore RF 0x%x\n\r", SPI_RX_data[2]);
Debug_UartPutString(Debug_message_array);

RF_SPI_Read(EXTENDED_REG, PARTNUMBER, SPI_TRANSACTION_TYPE_READ);
```

```
    if(SPI_RX_data[2] == CC1125_PARTNUMBER)
    {
        return TRUE;
    }

    return FALSE;
}



void        RF_Tx_Start(void)
{
    if(Status.flags == MODULATED_CARRIER)
    {
        RF_SPI_Write(EXTENDED_REG, TXFIRST_REG, 0x00, SPI_TRANSACTION_TYPE_WRITE);
        RF_SPI_Write(EXTENDED_REG, TXLAST_REG, PACKET_LENGTH+1, SPI_TRANSACTION_TYPE_WRITE);
    }

    RF_SPI_Write(STX, 0x00, 0x00, SPI_TRANSACTION_TYPE_WRITE);
    Debug_UartPutString("\n\rLa trasmissione RF e iniziata.\n\r");
}



void        RF_Tx_Stop(void)
{
    RF_SPI_Write(SIDLE, 0x00, 0x00, SPI_TRANSACTION_TYPE_WRITE);              //Ferma Tx.
    RF_SPI_Write(PKT_CFG2_REG, PKT_CFG2, 0x00, SPI_TRANSACTION_TYPE_WRITE);  //Setup for data transmission.
Different for unmodulated carrier.
    RF_SPI_Write(EXTENDED_REG, CFM_DATA_CFG_REG, CFM_DATA_CFG, SPI_TRANSACTION_TYPE_WRITE); //Setup for data
transmission. Different for unmodulated carrier.
    Status.flags = MODULATED_CARRIER;
    Debug_UartPutString("\n\rLa trasmissione RF e fermata.\n\r");
}
```