```
+-----------------------------------------------------------+
| ME_Imu_CHRUM7 - Unità per Misure Inerziali                |
+-----------------------------------------------------------+
| Versione: 0.11 Beta                                       |
| Revisione Documento: 2015-03-10 (2015-01-12)              |
| Autore: Roberto Gatti                                     |
| Tipo: Componente PSoC                                     |
+-----------------------------------------------------------+
|                             (c) 2015 Movie Engineering Srl |
+-----------------------------------------------------------+
```

INTRODUZIONE

Questo componente fa da interfaccia con la libreria che comunica con il chip hardware Ch Robotics UM7 (Unità per Misure Inerziali) che usa giroscopi e accelerometri per dare l'orientamento spaziale assoluto dell'oggetto su cui è montato rispetto alle stelle fisse.

L'hardware comunica via seriale e quindi il componente si occupa di fornire un meccanismo di trasmissione/ricezione dati e di conversione in angoli.

L'hardware inoltre ha tutta una serie di registri interni di configurazione e informazione a cui è possibile accedere con le stesse funzioni.

Le strutture dati e le funzioni di base sono nella libreria comune e non nel componente. (si vedano i moduli ME_lib_imu_CHRUM7.*)

L'hardware ha due modalità di spedizione dati: su richiesta o continua ad una frequenza impostabile nei registri interni (broadcast), il componente usa questa secondo modalità per i dati che comunicano gli angoli di Eulero (unici dati chiesti per ora all'hardware). (Sebbene si possano impostare i rate di spedizione di altri dati, non vengono fornite funzioni ad hoc per estrapolarli dal pacchetto, bisogna scriversele).

La frequenza di spedizione autonoma degli angoli di eulero è impostabile da parametro del componente.

PARAMETRI DEL COMPONENTE PSoC

BROADCAST_FREQ : Frequenza in Hz con cui il componente hardware deve spedire i dati.
NOTA: la frequenza massima dipende dalla velocità della trasmissione seriale e dalla grandezza del pacchetto dati (che a sua volta dipende da quali dati vengono spediti)
Impostare a 0 significa nessuna trasmissione, 255 significa massima frequenza 255 Hz.

MECCANISMI O PRINCIPI GENERALI DI FUNZIONAMENTO

Il componente hardware può essere configurato in modo da spedire i pacchetti con i dati ad una frequenza prefissata (che sia compatibile con la dimensione del pacchetto e la velocità di trasmissione).

Siccome i dati sono spediti su interfaccia seriale UART in generale è possibile che la ricezione cominci in un punto indefinito del pacchetto (ad esempio cavo attaccato in seguito o porta seriale attivata dopo l'accensione).

Per ovviare a questi inconvenienti viene utilizzato il meccanismo di bufferizzazione in ingresso (mediante il sotto-componente CBUFFER che è un buffer circolare) e il riallineamento nella costruzione del pacchetto (mediante il sotto-componente PACKETIZER che si occupa di scartare i dati finquando non ne individua un pacchetto intero coerente).

I dati ricevuti poi sono convertiti negli angoli di Eulero (vedi datasheet).

DESCRIZIONE API

L'uso del componente si riduce alla sola inizializzazione e alla chiamata periodica di una delle funzioni CheckFor... passando un puntatore a variabile opportuna (definite nell'header file).

Nota: le chiamate a CheckFor... elaborano un solo carattere alla volta quindi devono essere fatte ad una frequenza maggiore o uguale a quella sufficiente a smaltire tutti i caratteri in arrivo.

Dopo aver inizializzato il componente si possono chiamare le funzioni di servizio per impostare i vari altri parametri o comandi.

Costanti esportate

Valori per baud rate (dalla libreria generale)

- CHRUM7_BAUD_9600
- CHRUM7_BAUD_14400
- CHRUM7_BAUD_19200
- CHRUM7_BAUD_38400
- CHRUM7_BAUD_57600
- CHRUM7_BAUD_115200
- CHRUM7_BAUD_128000
- CHRUM7_BAUD_153600
- CHRUM7_BAUD_230400
- CHRUM7_BAUD_256000
- CHRUM7_BAUD_460800
- CHRUM7_BAUD_921600

Valori per broadcast rate

- RATE_PROC_ACCEL
- RATE_PROC_GYRO
- RATE_PROC_MAG
- RATE_QUAT
- RATE_EULER
- RATE_POSITION
- RATE_VELOCITY
- RATE_POSE

Variabili esportate

_serial_errors_count : Conteggio errori hardware della porta seriale

Funzioni Esportate

_InitAndStart : Inizializza e fa partire il componente.
Imposta automaticamente e fa partire la routine di interrupt di ricezione seriale.
Dopo la chiamata di questa funzione bisogna iniziare a svuotare il buffer della seriale (con le funzioni CheckFor...) in modo da evitare overrun perché l'hardware inizia a spedire i dati che vengono raccolti nel buffer.
Nel caso di errori del componente chiamare nuovamente questa funzione per resettarlo.
Restituisce un puntatore al buffer interno dove ci sono i dati grezzi del pacchetto (nel caso servisse andare a indagare pacchetti particolari)

_WriteRegister : Manda un pacchetto con richiesta di scrittura di un valore in un registro ed attende di ricevere il pacchetto di ack.
Note:
- la funzione è bloccante, ma il compoente è saldato e quindi a parte bachi hardware o rotture generali non dovrebbe creare problemi.
- vengono ignorati e scartati eventuali pacchetti broadcast ricevuti nel frattempo

_ReadRegister : Manda un pacchetto con richiesta di lettura di un valore da un registro ed attende il pacchetto con il valore.
Note:
- la funzione è bloccante, ma il compoente è saldato e quindi a parte bachi hardware o rotture generali non dovrebbe creare problemi.
- vengono ignorati e scartati eventuali pacchetti broadcast ricevuti nel frattempo

_Stop : Sospende la ricezione dei pacchetti spegnendo l'interrupt in modo che non venga dato alcun errore di overrun.

_CheckForPacket : Consuma un singolo byte dal buffer circolare del componente e inizia a costruire un pacchetto. Si risincronizza se la comunicazione viene avviata a metà pacchetto.
Restituisce TRUE solo quando ha ricevuto completamente un pacchetto ed ha verificato che sia valido.

_Error : Restituisce TRUE se c'è stato un errore di overflow o underflow del buffer circolare oppure se c'è stato un errore hardware di seriale.
In caso di errore per ripartire in modo pulito bisogna chiamare _InitAndStart.

_PacketAddress : Restituisce il campo 'address' del pacchetto presente nel buffer interno.
NOTA: il pacchetto DEVE essere valido (solo se l'ultima chiamata a CheckFor.. ha restituito TRUE).

_PrintPacket : Stampa su stdout il pacchetto presente nel buffer interno in formato leggibile.

Nota: gli ultimi due byte sono sempre la checksum

_ExecCommand : Esegue un comando del componente hardware (campo address >= 0xAA, vedi datasheet).

_RegisterIO : Scrive o legge da un singolo indirizzo dei registri del componente hardware (vedi datasheet).
Nota: nel caso di lettura spedisce indietro un pacchetto con il dato
Nota: se il parametro read è falso allora scrive value nel registro.

_SetBAUD : Imposta un nuovo valore di velocità di comunicazione.
Dopo questa chiamata bisogna cambiare velocità dal lato host altrimenti non si comunica.

_SetRate : Imposta la frequenza di broadcast autonomo di un singolo gruppo di dati
Ogni gruppo può essere spedito a frequenza da 0 (non spedito) a 255 Hz.
I rate dei gruppi sono indipendenti ed impostabili con diverse chiamate.
La funzione InitAndStart imposta il broadcast degli angoli di eulero in base al valore del parametro del componente psoc.
Nota : fare attenzione che la somma dei byte spediti per ciascun gruppo ognuno alla sua frequenza potrebbe eccedere la capacità di trasmissione del baud rate impostato. Questo porta ad un blocco delle trasmissioni.

_SetMisc : Imposta alcuni parametri generali
- zerogyro : cerca di misurare il bias dei giroscopi all'accensione (deve essere fermo durante l'accensione)
- quaternion_mode : esegue usando i quaternioni e non gli angoli di eulero
- use_magnetometer : usa il magnetometro durante gli aggiornamenti di stato

_SetGyroTrim : Imposta il bias dei giroscopi che verrà aggiunto a quello misurato durante il comando _ExecCommand(CHRUM7_ZERO_GYROS)
NOTA: al momento della chiamata di questa funzione non viene eseguita alcuna modifica ai dati in uscita, è necessario eseguire il comando di calibrazione.

_GetProcessedData : legge direttamente dai registri i dati elaborati ed aggiorna la struttura che gli viene passata con i dati freschi.

_PrintSettings : stampa le impostazioni attualmente sul sensore

_PrintData : legge e stampa tutti i dati

_PacketToEuler : Converte il pacchetto presente nel buffer interno in coordinate euleriane aggiornando la struttura passata per riferimento.
NOTA: il pacchetto DEVE essere valido (solo se l'ultima chiamata a CheckFor.. ha restituito TRUE.

_CheckForEulerPacket : come CheckForPacket, ma verifica che il pacchetto ricevuto contenga le coordinate euleriane e nel caso effettua la conversione e riempie la struttura passata per indirizzo.
NOTA: tutti gli altri pacchetti ricevuti vengono processati ma IGNORATI.

ESEMPIO DI UTILIZZO
==========

```
BYTE *packet_buffer_ptr ;
EULER_ANGLES e ;

void test_pacchetti(void)
{
    printf(CLEAR);
    while (1)
    {
//      /* Legge e stampa i pacchetti broadcast con gli angoli di eulero */
//         if (IMU_CheckForEulerPacket(&e))
//         {
//             printf(CRLF"OK %7.2f %7.2f %7.2f ",    e.phi, e.theta, e.psi);
//         }

//      /* Legge e stampa i pacchetti in formato generico */
//         if (IMU_CheckForPacket())
//         {
//             IMU_PrintPacket();
//         }

        /* Legge e stampa vari dati */
         printf(HOME);
         IMU_PrintData();
         CyDelay(10);

//         /* Esegue un comando... */
//         if (CONSOLE_GetChar() != 0)
//         {
//             IMU_ExecCommand(CHRUM7_GET_FW_REVISION);
```

```
//        }


    }
}

void main(void)
{
    setvbuf(stdout, 0, _IONBF, 0);          // No buffering per printf.

    CyGlobalIntEnable;

    packet_buffer_ptr = IMU_InitAndStart();

    CONSOLE_Start();
    printf("\r\nUM7 Test!\r\n");

    IMU_PrintSettings();

    IMU_SetRate(RATE_EULER, 1);
    IMU_SetRate(RATE_PROC_ACCEL, 0);
    IMU_SetRate(RATE_PROC_GYRO, 0);
    IMU_SetRate(RATE_PROC_MAG, 0);
    IMU_SetGyroTrim(10.23, 20.34, 30.45);
    IMU_PrintSettings();


    test_pacchetti();

    while(1)
    {
    }
}
```