```
+-----------------------------------------------------------+
| Packetizer                                                |
+-----------------------------------------------------------+
| Versione: 0.11 Beta                                       |
| Revisione Documento: 2014-11-13 (creato il 2014-02-15)    |
| Autore: Roberto Gatti                                     |
|                                                           |
+-----------------------------------------------------------+
|                            (c) 2014 Movie Engineering Srl |
+-----------------------------------------------------------+
```

INTRODUZIONE

Estrae un pacchetto dati strutturato da un flusso di byte in arrivo.

Al componente bisogna fornire due funzioni di call-back per validare l'header del pacchetto (byte di inizio, eventuali lunghezze, indirizzi, codici, ecc.) e il pacchetto stesso (ad es. calcolare le checksum, crc, ecc.)

I dati in arrivo possono essere anche fuori sync (cioè non iniziare dall'inizio del pacchetto ma con dati parziali), una volta impostato il carattere di inizio pacchetto il componente pensa a risincronizzarsi perdendo meno byte e pacchetti possibile.

La funzione che costruisce il pacchetto riceve un byte alla volta e restituisce un valore true alla chiamata contenente l'ultimo byte di un pacchetto valido.

Questo componente ha i seguenti limiti:
- L'header dei pacchetti in arrivo deve essere di lunghezza fissa (anche 1 solo byte, sop)
- L'inizio dei pacchetti è segnalato da un singolo byte (ma si può sempre validare il tutto nella funzione di validazione del pacchetto intero o dell'header)
- I pacchetti possono essere di lunghezza variabile, ma deve essere indicata nell'header (oppure estrapolata dal tipo di pacchetto, il tipo deve essere nell'header)

PARAMETRI DEL COMPONENTE PSoC

HEADER_LENGTH : Lunghezza dell'header del pacchetto comprensiva di primo carattere (start of packet). Deve essere almeno 1. La header deve avere lunghezza fissa e all'interno di essa ci può essere qualche parametro che definisce la lunghezza del pacchetto di cui fa parte.

MAX_PACKET_LENGTH : Uno stesso start of packet può rappresentare pacchetti di lunghezza diversa (definiti nei restanti byte della header). Questo parametro rappresenta la massima lunghezza del pacchetto più lungo (header compresa).

START_OF_PACKET : Singolo byte che rappresenta l'inizio di un pacchetto (quindi dell'header di pacchetto che potrebbe essere formata da solo questo byte).
Questo parametro può essere cambiato a run-time con l'apposita funzione, ma un componente può avere un solo start of packet alla volta.

MECCANISMI O PRINCIPI GENERALI DI FUNZIONAMENTO

Il packetizer semplifica tutta la gestione della ricezione di pacchetti di dati già in ordine (non spezzati e disordinati). Tipicamente su protocolli seriali o simili.

Il flusso di dati in ingresso al componente viene analizzato un byte alla volta e una macchina a stati interna fa in modo che vengano fatte delle scelte per minimizzare i dati persi e risincronizzare velocemente nel caso in cui la ricezione cominci a metà pacchetto.

Il componente si occupa inoltre di validare (mediante funzioni di call-back fornite

all'inizializzazione) sia la parte di header (che può contenere la lunghezza effettiva del pacchetto in arrivo) sia l'intero pacchetto (di solito con un controllo di checksum e di sequenza). La logica della validazione viene lasciata all'utente del componente che deve necessariamente definire e fornire le due funzioni di cui sopra.

Il componente tiene conto di varie statistiche come il numero di pacchetti non validi.

La risincronizzazione viene fatta in modo intelligente: un byte di inizio pachetto è un valore come gli altri e potrebbe capitare che, se la ricezione cominciasse a metà pacchetto, il carattere ricevuto non sia l'inizio pachetto ma solo un valore interno. In questo caso al momento della validazione fallita viene tenuto conto di eventuali altri start of packet interni e i dati ricevuti vengono traslati in modo da effettuare una nuova validazione a partire da questo nuovo start of packet. In questo modo i pachetti si risincronizzano nel minor tempo possibile.

DESCRIZIONE API

La API del componente non è nient'altro che un "involucro" di chiamate alle funzioni della libreria "ME_lib_packetizer.*" e di allocazione dei buffer di memoria richiesti in base ai parametri del componente stesso.

Variabili esportate

Nessuna.

Funzioni Esportate

Init : inizializza il componente impostando le funzioni di validazione.

NOTA: questa funzione resituisce un puntatore all'array di byte in cui viene costruito il pacchetto ricevuto. E' necessario quindi salvarlo per poter accedere ai dati quando la funzione BuildPacket restituisce true.

Reset : ripristina il componente e i contatori interni.

BuildPacket : funzione di costruzione e analisi del pacchetto. Viene chiamata ad ogni byte ricevuto (in modo che sia non bloccante) e si occupa di tutto il processo di individuazione dell'inizio e fine pacchetto e controllo checksum quando ha ricevuto il numero di byte necessario. Riallinea automaticamente in caso di perdita di sincronizzazione nella ricezione (mezzi pacchetti e simili). Restituisce true solo quando un pacchetto è riconosciuto come valido e lo si può leggere all'indirizzo puntato dal puntatore restituito dalla funzione Init.

SetStartOfPacket (byte) : imposta il carattere di inizio pacchetto nel caso fosse necessario modificarlo a runtime.

PacketStatus : stato attuale della macchina a stati interna del componente.

PACKET_WAITING	: Stanno arrivando byte ma non è ancora arrivato un sop
PACKET_RESYNCING	: La validazione ha dato KO e c'è uno sop nei dati ricevuti
PACKET_BUILDING	: E' arrivato uno sop e sta accodando i byte in arrivo
PACKET_OK	: La funzione di validazione ha dato ok.
PACKET_ERROR	: C'è stato un errore da qualche parte (usare Reset)

InvalidPackets (bool) : conteggio di pacchetti che hanno dato errore nella funzione di validazione. Se viene passato true il conteggio viene azzerato.

ESEMPIO DI UTILIZZO

==========

```
    ----------------------------------
     Esempi di funzioni di validazione
    ----------------------------------

    BYTE PNISPS_HeaderValidator(BYTE *packet)
    {
        BYTE length = 0 ;

        if (packet[0] == 0x24)        // Il pacchetto che inizia con 0x24 ha lunghezza fissa 26
        {
            length = 26 ;
        }

        return length ;
    }


    BOOL PNISPS_PacketValidator(BYTE *packet, BYTE length)
    {
        BOOL valid = false ;

        if ( ( packet[00] == 0x24 ) &&                  // Formato richiesto dal pacchetto
             ( packet[24] == 0x0D ) &&                  // Inizio : 0x24
             ( packet[25] == 0x0A ) )                   // Fine : 0x0D, 0x0A
        {
            BYTE sumA, sumB ;

            fletcher16( &sumA, &sumB, packet, 22);          // Calcolo della checksum

            valid = ( ( sumA == packet[22] ) && ( sumB == packet[23] )) ;
        }

        return valid ;
    }


    --------------------------------------
     Esempio di ciclo di attesa pacchetti
    --------------------------------------

    while (   (PNISPS_CBUFFER_AnyData()!=0)
           && (!valid) )
    {
        valid = PNISPS_PACKETIZER_BuildPacket(PNISPS_CBUFFER_GetByte());
        if (PNISPS_status == SP_WAITING)
           PNISPS_status = SP_OK ;
    }
```