# Formato dei dati generati da XbusMaster Emulator

Il formato di ogni sequenza di byte ricevuta da Xbus master dipende, oltre che dalla modalità di output, dal numero di sensori collegati; un pacchetto inizia sempre con un preambolo di un byte (0xFA), ed è seguito da un BID (Bus Identifier, identifica mittente e destinatario dei messaggi) e da un MID (Message Identifier, identifica ogni specifico messaggio). Il MID nel caso dei dati sensoriali è 0x32.

Segue un quarto campo Length, che rappresenta la quantità di byte da leggere successivamente; se questo campo è uguale a 255 è attiva la modalità Extended-Length, quindi per conteggiare i byte da leggere viene usato un ulteriore numero a 16 bit rappresentato con il quarto e quinto byte (il pacchetto contiene quindi due byte in più). Ad esempio se riceviamo i dati da 5 sensori, avremo 5 \* 52 = 260 byte solo per i dati dei sensori, a cui andranno aggiunti gli altri byte. In Fig. 1 è mostrata la differenza tra un pacchetto normale e un pacchetto Extended-Length.

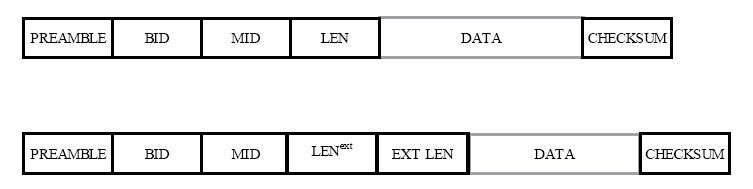


Fig.1

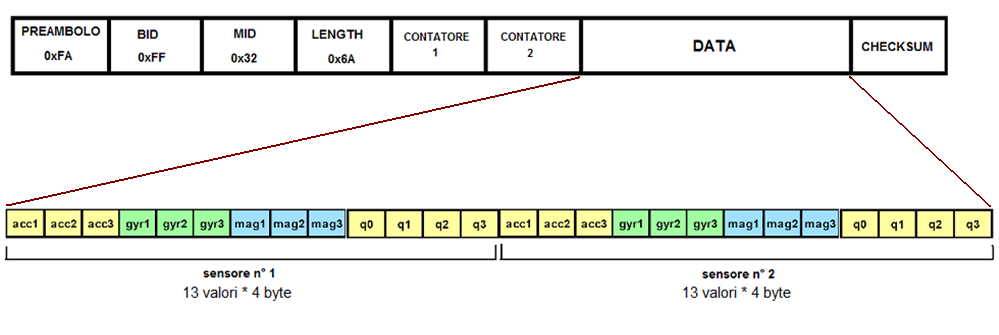


Fig.2

Successivamente ci sono due byte che fungono da contatore (in Fig. 3 sono compresi nel campo “DATA”); il valore contenuto è incrementato ad ogni pacchetto inviato, e azzerato ogni volta che Xbus master viene stoppato e fatto ripartire. Alla fine dei dati è presente un byte di Checksum, per l’individuazione di eventuali errori.

Nell’immagine di Fig. 2 è rappresentato un esempio di pacchetto per due sensori in formato calibrato con quaternioni:

## Ricezione, traduzione e memorizzazione dei dati

Realizziamo una funzione per ricevere e tradurre i dati in arrivo da tutti i sensori MTx; permette la ricezione dei dati provenienti da uno a dieci sensori, il massimo possibile per Xbus master, anche se l’hardware utilizzato nei nostri test ha utilizzato al massimo cinque sensori.

Per la connessione viene utilizzata una Socket con TcpListener; alla Socket viene associato un NetworkStream, e a questo si associa un BinaryReader. Il metodo ReadBytes() della classe BinaryReader permette di essere certi di ricevere il numero di byte indicato. Questa scelta è motivata dal fatto che non è detto che metodi come Socket.Receive() e NetworkStream.Read() ricevano sempre il numero di byte prefissato. Le cause possono essere problemi con le connessioni (Wi-Fi o Bluetooth), una difficoltà da parte del programma server nel processare i dati nei tempi necessari, o una difficoltà di Xbus master o di Motus nell’invio dei dati. Questo ha portato in alcuni test a perdere alcuni byte e talvolta anche interi pacchetti.

La funzione server, una volta connessa la Socket, cerca l’esatta sequenza 0xFF – 0x32 (che identifica un pacchetto di dati; 0xFA non viene considerato in questo codice), e finché non riconosce l’esatta sequenza continua a leggere un byte alla volta. Successivamente viene analizzato il campo Length, che permette di riconoscere la quantità di byte successivi da leggere, e quindi di dedurre il numero di sensori collegati.

Se il campo Length è uguale a 255 (0xFF) significa che siamo nella modalità Extended Length (nel nostro caso questo equivale a dire che tutti e 5 i sensori sono connessi all’Xbus Master).

Con due cicli, uno per il numero di sensori e uno per i tredici valori, tutti i byte del pacchetto vengono tradotti in valori float, complemento a due, singola precisione (secondo lo standard IEEE 754), grazie al metodo BitConverter.ToSingle().

Una volta acceso l’Xbus master, non è possibile modificare il numero di sensori; per farlo è necessario spegnere l’hardware, ricollegare i sensori e riaccenderlo. La funzione di ricezione si aspetta quindi sempre pacchetti della stessa lunghezza, definita nell’analisi del primo pacchetto di dati (questo permette di risparmiare la ripetizione delle istruzioni illustrate in precedenza).

Segue parte del codice della funzione, in un esempio di semplice output dei dati a console (sono stati omessi particolari come Try-Catch e tutti i metodi come salvataggio su file, aggiornamento di text box, timer ecc).

byte[] len = new byte[2];

byte[] tem = new byte[3];

while (!(tem[0] == 0xFF && tem[1] == 0x32)) // cerca la sequenza FF-32

{

tem[0] = tem[1];

tem[1] = tem[2];

byte[] read = bin.ReadBytes(1);

tem[2] = read[0];

}

if (tem[2] != 0xFF) // modalità normale

{

byteToRead = tem[2]; // byte da leggere

}

else // modalità extended-length

{

len = new byte[2];

len = bin.ReadBytes(2);

byteToRead = (len[0] \* 256) + len[1]; // byte da leggere

}

byte[] data = new byte[byteToRead + 1];

data = bin.ReadBytes(byteToRead + 1); // lettura dei dati

if (tem[2] != 0xFF)

{

pacchetto = new byte[byteToRead + 4]; // creazione pacchetto

}

else

{

pacchetto = new byte[byteToRead + 6];

}

numSensori = (byteToRead - 2) / 52; // calcolo del numero di sensori

pacchetto[0] = 0xFF; // copia dei primi elementi

pacchetto[1] = 0x32;

pacchetto[2] = tem[2];

if (tem[2] != 0xFF)

{

data.CopyTo(pacchetto, 3); // copia dei dati

}

else

{

pacchetto[3] = len[0];

pacchetto[4] = len[1];

data.CopyTo(pacchetto, 5); // copia dei dati

}

List<List<double>> array = new List<List<double>>(); // salvataggio dati

int[] t = new int[maxSensori];

for (int x = 0; x < numSensori; x++)

{

array.Add(new List<double>()); // una lista per ogni sensore

t[x] = 5 + (52 \* x);

}

while (true)

{

for (i = 0; i < numSensori; i++)

{

byte[] temp = new byte[4];

for (int tr = 0; tr < 13; tr++)// 13 campi, 3 \* 3 + 4

{

if (numSensori < 5)

{

temp[0] = pacchetto[t[i] + 3]; // lettura inversa

temp[1] = pacchetto[t[i] + 2];

temp[2] = pacchetto[t[i] + 1];

temp[3] = pacchetto[t[i]];

}

else

{

temp[0] = pacchetto[t[i] + 5];

temp[1] = pacchetto[t[i] + 4];

temp[2] = pacchetto[t[i] + 3];

temp[3] = pacchetto[t[i] + 2];

}

valore = BitConverter.ToSingle(temp, 0); // conversione

array[i].Add(valore); // memorizzazione

t[i] += 4;

}

}

for (int x = 0; x < numSensori; x++)

{

t[x] = 5 + (52 \* x);

}

for (int j = 0; j < numSensori; j++)

{

for (int tr = 0; tr < 13; tr++)

{

// esempio output su console

Console.Write(array[j][tr] + "; ");

}

Console.WriteLine();

array[j].RemoveRange(0, 13); // cancellazione dati

}

Console.WriteLine();

if (numSensori < 5) // lettura pacchetto seguente

{

pacchetto = bin.ReadBytes(byteToRead + 4);

}

else

{

pacchetto = bin.ReadBytes(byteToRead + 6);

}

}