

BABUC ABC MANUALE DEL PROGRAMMATORE

Per strumenti versione 4 e 5

<i>Documento:</i>	MW6061	<i>Revisione:</i>	a
<i>Pag./TotPag.:</i>	1 / 28	<i>del:</i>	12/09/2002

SOMMARIO

1 INTRODUZIONE.....	3
2 RIFERIMENTI.....	3
3 PROTOCOLLO PROPRIETARIO LSI-LASTEM.....	3
3.1 INFORMAZIONI GENERALI.....	3
3.1.1 Protocollo di comunicazione.....	3
3.1.2 Livello fisico.....	4
3.1.3 Formato di memorizzazione e dei dati trasmessi.....	4
3.1.4 Tempi di trasmissione.....	5
3.2 IL FRAME.....	9
3.2.1 Calcolo del CRC.....	9
3.2.2 Significato di OpCode.....	10
3.3 LISTA DEI CODICI DI ERRORE.....	12
3.4 DEFINIZIONI ED ENUMERAZIONI.....	12
3.5 DEFINIZIONE DELLE STRUTTURE DATI.....	18
3.5.1 Struttura BBCTime.....	18
3.5.2 Struttura BC_SysInf.....	18
3.5.3 Struttura DataMemInf.....	19
3.5.4 Struttura RelMemHeader.....	19
3.5.5 Struttura ChMemHeader.....	20
3.5.6 Struttura ElabEvent.....	20
3.5.7 Struttura FECommon.....	21
3.5.8 Struttura FEMinMedMaxDvst[B/W/F].....	21
3.5.9 Struttura FEDTMinMedMaxDvst[B/W/F].....	21
3.5.10 Struttura FEEolo3.....	22
4 PROTOCOLLO MODBUS.....	23
4.1 PRESTAZIONI FUNZIONALI.....	23
4.2 LINEA FISICA DI COMUNICAZIONE.....	23
4.3 FRAME DI TRASMISSIONE.....	23
4.4 COMANDI E RICHIESTE MODBUS GESTITI DA BABUC ABC.....	24
4.4.1 Richiesta Read Input Status.....	24
4.4.2 Richiesta Read Input Register.....	25
4.4.3 Richiesta Reset Insorgenza Errore.....	26
4.4.4 Richiesta Approntamento Elaborazioni.....	26
4.4.5 Richiesta Trasmissione Elaborazioni.....	27
4.4.6 Richiesta Modifica Data/ora di Sistema.....	28
4.4.7 Calcolo del CRC.....	28

1 Introduzione

Babuc ABC supporta due tipi di protocolli di comunicazione: uno di tipo proprietario Lastem, ed uno di tipo MODBUS. Questo documento descrive i due protocolli in modo tale da permettere lo sviluppo di applicazioni di comunicazione con gli acquisitori Babuc ABC V. 4.02 da parte di personale tecnico non Lastem.

2 Riferimenti

[1] Babuc ABC Manuale operativo

3 Protocollo proprietario LSI-Lastem

3.1 Informazioni generali

3.1.1 Protocollo di comunicazione

Le stazioni di acquisizione si comportano come slave rispetto al sistema master che le interroga; esse quindi non possono mai intraprendere di propria iniziativa un trasferimento dati.

La richiesta e la trasmissione dei dati fra il master e lo slave avvengono attraverso l'invio sulla linea di comunicazione seriale di *frame* binari, che uniscono più informazioni all'interno di un unico pacchetto di caratteri, trasmessi l'uno di seguito all'altro.

Il frame possiede le informazioni che permettono di destinare una richiesta di dati verso una stazione piuttosto di un'altra, di allegare informazioni supplementari alla richiesta, di contenere i dati veri e propri da trasmettere e di effettuare il controllo sulla validità dei dati ricevuti.

Il master effettua sempre richieste che vengono inviate all'interno di un unico frame; la risposta da parte dello slave può giungere invece con uno o più frame, in quanto i dati trasmessi possono essere di numero maggiore rispetto alle informazioni che un singolo frame può contenere (trasmissioni *multiframe*). L'invio delle richieste da parte del master avviene una alla volta; per ogni richiesta inviata si ha sempre almeno un frame di risposta da parte dello slave.

Alla ricezione di un frame da parte del master non necessariamente deve corrispondere la trasmissione della risposta da parte dello stesso. Nel caso in cui lo slave produca una trasmissione di dati che sono contenuti entro un solo frame, il master può fare a meno di rispondere allo slave, in quanto quest'ultimo non rimane in attesa di conferma della ricezione avvenuta; esso è comunque sempre in grado di inviare il frame precedentemente trasmesso, ogniqualvolta il master lo richieda. Nel caso in cui lo slave produca una trasmissione di dati che debbano essere contenuti all'interno di più frames (trasmissione *multiframe*), ad ogni frame ricevuto dal master deve corrispondere, da parte di quest'ultimo, l'invio di un frame di conferma di ricezione avvenuto (*Ack*); questo produce l'invio, da parte dello slave, del frame successivo. La condizione viene ripetuta fino a quando il master non riceve un frame contenente il segnale di ultimo frame trasmesso (*LastFrame*); a questo punto il master è libero di rispondere o meno.

La sicurezza della trasmissione e la correttezza dei dati ricevuti sono garantite dal carattere di controllo CRC a 16 bit, calcolato sulla base del contenuto del frame trasmesso. In caso di non corrispondenza fra CRC ricevuto e CRC calcolato, il master esegue la richiesta di ritrasmissione del frame, mediante apposito comando. Se lo slave non riceve correttamente il frame, non segnala l'anomalia di ricezione, per non influenzare eventuali comunicazioni in corso fra il master ed un altro slave. Nel caso in cui il master non riceva risposta dallo slave entro un certo tempo predefinito, o la risposta gli giunga errata, esso è libero di richiedere per un numero di volte indefinito la ritrasmissione del frame da parte dello slave.

Documento:	MW6061	Revisione:	a
Pag./TotPag.:	3 / 28	del:	12/09/2002

I tre caratteri di sincronismo trasmessi all'inizio del frame garantiscono che il frame venga ricevuto correttamente, anche in presenza di caratteri spuri dovuti, per esempio, all'attivazione delle portanti radio. Il protocollo di comunicazione è in grado di riconoscere correttamente il frame anche se viene ricevuto uno solo dei tre caratteri di sincronismo.

Il tempo massimo di risposta dello slave a qualsiasi richiesta è dell'ordine di 3 secondi. Il master deve attendere quindi almeno per questo periodo prima di considerare non ricevuta la risposta dallo slave.

3.1.2 Livello fisico

Il protocollo fisico di trasmissione avviene con bit rate compresi fra 1200 e 19200 (inclusi), 8 bit per carattere, 1 bit di stop e nessuna parità. I frame trasmessi possono avere lunghezze comprese fra qualche byte e 2048 bytes. I dati trasmessi con formato binario a 16 o 32 bit hanno il byte più significativo nell'indirizzo minore ed il byte meno significativo all'indirizzo maggiore. I valori floating point sono a 32 bit e rispettano lo standard IEEE-754.

Il protocollo gestisce correttamente connessioni point to point e connessioni multipoint fra un master e più stazioni slave (fino ad un massimo di 32 su linea RS-485).

Lo strumento Babuc ABC può pilotare convenientemente il segnale RTS (*request to send*) in modo tale da consentire trasmissioni half duplex mediante linee di comunicazione via radio a singola frequenza. Questo segnale deve essere accoppiato al sistema radio-trasmittente in modo tale da pilotare adeguatamente il segnale di attivazione portante in fase di trasmissione.

Il segnale RTS viene abilitato prima di iniziare la trasmissione del frame e viene opportunamente disabilitato al termine della trasmissione. Il tempo che intercorre tra l'abilitazione del segnale e l'inizio della trasmissione del frame è modificabile da un minimo di 100 ms ad un massimo di 990 ms. con passi di 100 ms.

3.1.3 Formato di memorizzazione e dei dati trasmessi

La memoria del sistema Babuc ABC può contenere uno o più *rilievi*; il rilievo è un insieme di elaborazioni/eventi che vengono raggruppati in modo da essere poi decodificati in modo univoco. La decodifica dei dati avviene mediante alcune *testate*, memorizzate al momento dell'apertura del rilievo in memoria; queste testate contengono le informazioni generali sul rilievo (struttura *RelMemHeader*) e le informazioni sulle elaborazioni e sui canali di acquisizione/preelaborazione che hanno prodotto le elaborazioni stesse (struttura *ChMemHeader*). Ogni rilievo contiene una testata di rilievo *RelMemHeader* ed una o più testate di canale *ChMemHeader*.

Il sistema permette la memorizzazione dei dati in forma lineare o circolare; il formato lineare produce la memorizzazione dei dati fino al termine fisico della memoria, dopodiché la memorizzazione viene arrestata e prodotto un errore di memoria piena; la memorizzazione circolare consente al sistema di ricoprire i dati più vecchi con i più giovani; questo avviene quando lo spazio fisico in memoria viene terminato, per cui le nuove elaborazioni o eventi vengono memorizzati ripartendo all'inizio del rilievo, dopo l'ultima testata di canale.

Nel caso di memorizzazione circolare, lo spazio a disposizione per effettuare la *circolarità* della memorizzazione è compreso fra il termine dell'ultima testata di canale del rilievo corrente e il termine fisico della memoria; per questo motivo, se la memoria dati contiene altri rilievi, precedentemente memorizzati, e che hanno utilizzato solo una parte della memoria a disposizione, l'ultimo rilievo può effettuare la memorizzazione dei dati in forma circolare solo sullo spazio di memoria rimanente, senza alterare il contenuto dei rilievi precedenti; una situazione limite è rappresentata dalla possibilità di non poter effettuare alcuna memorizzazione, nel caso in cui il rilievo o i rilievi precedenti abbiano occupato tutto lo spazio fisico di memoria.

Tutte le elaborazioni o eventi che vengono memorizzati in memoria appartengono a svariati tipi di strutture; ognuna di queste include all'interno una struttura di tipo *FECCommon*, che permette il riconoscimento dei dati che seguono la struttura *FECCommon* stessa; la funzione di conversione che deve convertire i dati binari ricevuti dall'acquisitore procede quindi alla decodifica degli stessi partendo dapprima dalla struttura *FECCommon* e decifrando poi l'elaborazione seguente; quindi si deve spostare alla struttura *FECCommon* successiva e così via fino al termine dei dati ricevuti.

Tra una elaborazione e l'altra, può essere memorizzata una struttura di tipo *FinderMemHeader*, utilizzata internamente dal sistema per snellire alcune operazioni di ricerca dei dati; queste strutture, al fine ottenere una corretta conversione, devono essere ignorate.

Segue una rappresentazione schematica del contenuto della memoria con due rilievi memorizzati (*FEElab* intende un qualsiasi tipo di elaborazione o evento):

RelMemHeader	
ChMemHeader	
ChMemHeader	
...	
ChMemHeader	
FEElab	
FEElab	
...	
FEElab	
FinderMemHeader	
FEElab	
FEElab	
RelMemHeader	
ChMemHeader	
ChMemHeader	
...	
ChMemHeader	
FEElab	
FEElab	
...	
FEElab	
FEElab	

<i>Testata del rilievo</i>	
<i>Testata del canale</i>	<i>blocco</i>
	<i>testate</i>
	<i>canali</i>
<i>Elaborato</i>	
	<i>blocco</i>
	<i>elaborati</i>
<i>Blocco utilizzato per</i>	
<i>ricerca rapida dei dati</i>	
<i>Testata del rilievo</i>	
<i>Testata del canale</i>	<i>blocco</i>
	<i>testate</i>
	<i>canali</i>
<i>Elaborato</i>	
	<i>blocco</i>
	<i>elaborati</i>

I codici di richiesta a disposizione permettono di ottenere la trasmissione di tutte le testate di rilievo (una o più strutture *RelMemHeader*), la trasmissione di una testata di rilievo, di cui è fornito l'indice, completa delle testate di canale (una o più strutture *ChMemHeader*) e la trasmissione dei dati elaborati, a partire da una certa data/ora.

3.1.4 Tempi di trasmissione

Il tempo dedicato alla trasmissione dei dati dallo strumento Babuc ABC al sistema master esterno è dipendente dai seguenti fattori:

- velocità di trasmissione (bit rate)
- lunghezza delle informazioni da trasmettere
- overhead imposto dalla struttura del frame
- tempo di risposta dello strumento

3.1.4.1 Velocità di trasmissione

La velocità di trasmissione, espressa in bps (bit per second), fornisce la quantità di bit che, nell'unità di tempo di un secondo, possono transitare sulla linea di comunicazione. In condizioni ideali, e quindi in presenza di un flusso continuo di dati, la quantità di bytes trasmessi sulla linea di comunicazione è ricavabile dalla seguente espressione:

$$\frac{BitRate}{BitsPerByte + NrStartBit + NrStopBit}$$

dove:

BitRate: velocità di comunicazione, espressa in bps
BitsPerByte: numero di bits che compongono un byte (8)
NrStartBit: numero di bit di start (per Babuc ABC: 1)
NrStopBit: numero di stop bit (per Babuc ABC: 1)

3.1.4.2 Lunghezza delle informazioni da trasmettere

Questa informazione è dipendente dal tipo di informazioni che vengono inviate sulla linea di comunicazione. Considerando ad esempio la trasmissione dei valori istantanei acquisiti in un dato istante dallo strumento, la dimensione dei dati trasmessi corrisponde al numero di canali di acquisizione moltiplicato 4, corrispondente al numero di bytes che compongono un singolo numero in formato floating point.

Seguono alcuni esempi in cui si elencano i bytes trasmessi per configurazioni tipiche dello strumento Babuc ABC

Tipo di trasmissione e configurazione dello strumento	Nr. bytes
Trasmissione dei valori istantanei da uno strumento configurato con 10 canali	40
Trasmissione dei dati elaborati orari e giornalieri di un giorno da uno strumento configurato con 10 canali ed elaborazioni <i>EEMinMedMaxDvst</i> di tipo <i>word</i>	≈4000
Trasmissione dei dati elaborati orari e giornalieri di un mese da uno strumento configurato con 10 canali ed elaborazioni <i>EEMinMedMaxDvst</i> di tipo <i>word</i>	≈120000
Trasmissione dei dati elaborati orari e giornalieri di un giorno da uno strumento configurato con 5 canali ed elaborazioni <i>EEMinMedMaxDvst</i> di tipo <i>word</i> per 4 canali ed elaborazioni <i>EEEolo3</i> per un canale di direzione del vento	≈2600
Trasmissione dei dati elaborati orari e giornalieri di un mese da uno strumento configurato con 5 canali ed elaborazioni <i>EEMinMedMaxDvst</i> di tipo <i>word</i> per 4 canali ed elaborazioni <i>EEEolo3</i> per un canale di direzione del vento	≈78000
Trasmissione della configurazione di funzionamento dello strumento (parametri di sistema, di rilievo, libreria dei codici operativi, calibrazione degli amplificatori, etc.):	≈16000
Trasmissione di valori istantanei raggruppati su elaborazioni <i>EE10Ist</i> di tipo <i>word</i> per 8 canali (80 valori in totale) con tempi di acquisizione a scelta	216
Trasmissione di 80 elaborazioni <i>EEMinMedMaxDvst</i> di tipo <i>word</i> ripartite su vari canali	1280

3.1.4.3 Overhead imposto dalla struttura del frame

Il frame di trasmissione consente la regolazione del traffico delle informazioni sulla linea di comunicazione; esso aggiunge, ai veri dati trasmessi, alcune informazioni supplementari.

L'influenza della lunghezza della struttura del frame sulla trasmissione dipende dalla natura, e quindi dalla dimensione, dei dati trasmessi, e dalla suddivisione in frame dell'insieme delle informazioni che lo strumento trasmette al sistema master.

La scelta della lunghezza del frame, selezionabile dall'utente, è determinata principalmente dal tipo di media trasmissivo utilizzato per la comunicazione e quindi dalla qualità e affidabilità della linea di trasmissione. E' quindi opportuno utilizzare frame di lunghezza corrispondente a 512 bytes, fino ad arrivare a 1024 bytes, per linee di comunicazione dedicate o a bassa velocità, e lunghezze di frame limitate a 32 o 64 bytes per linee di comunicazione particolarmente disturbate.

Per le trasmissioni via radio è necessario diminuire progressivamente la lunghezza del frame fino ad raggiungere la minore probabilità di incorrere in disturbi sulla portante radio nel periodo in cui viene spedito il frame. La lunghezza non deve però essere eccessivamente limitata per non influire negativamente sul tempo di trasmissione dell'insieme di dati da inviare: ogni inizio frame è difatti caratterizzato da un tempo di attesa, necessario al fine di garantire il perfetto sincronismo dei due apparati radio, ogni qualvolta viene invertita la comunicazione fra stazione trasmittente e stazione ricevente.

3.1.4.4 Tempo di risposta dello strumento

Lo strumento Babuc ABC risponde ad un insieme di codici di richiesta dati o di comando in tempi differenti; ciò dipende da quanto tempo lo strumento impiega a recuperare le informazioni contenute nella sua memoria, o ad elaborare i comandi ricevuti.

In generale il tempo di risposta è pressoché immediato per la quasi totalità delle richieste accettate. Fa eccezione la trasmissione dei dati elaborati, a partire da una data/ora imposta dal sistema master; questo tipo di comando richiede da parte dello strumento l'esecuzione di una funzione di ricerca all'interno della memoria dati, necessaria al fine di trasmettere solamente le informazioni desiderate. Questa ricerca viene conclusa in un tempo variabile, dipendente da 3 fattori:

- numero di elaborazioni memorizzate nella memoria dati
- dimensione della memoria dati
- distanza fisica della prima elaborazione con data/ora corrispondente a quella richiesta rispetto alla fine del rilievo

La ricerca avviene partendo dall'elaborazione più giovane e proseguendo verso l'inizio del rilievo, dove sono presenti le elaborazioni con data/ora più vecchia; in questo modo sono avvantaggiate le ricerche con data/ora prossima al tempo reale e svantaggiate le ricerche con data/ora più vecchia.

Considerando la condizione limite maggiormente sfavorevole, ovvero lo strumento avente connessa la memory card di maggiori dimensioni, completamente piena di elaborazioni e la stazione master che richiede i dati con data/ora corrispondente all'elaborazione più vecchia, Babuc ABC produce la trasmissione del primo frame entro 3 secondi; i frames successivi vengono inviati, dopo la ricezione del frame di *Ack*, in modo praticamente istantaneo; sono necessari solo alcuni decimi di secondo, necessari alla composizione dei frames successivi, se questi hanno dimensioni cospicue, oltre 512 bytes.

Considerando quanto sopra esposto, si elencano alcuni casi reali di trasmissione dati e relativi tempi di occupazione della linea di comunicazione. Si considera l'utilizzo di una linea di comunicazione di buona qualità.

Tipo di trasmissione	Bit Rate (bps)	Dimensione del frame (bytes)	Tempo impiegato (mm:ss)
Trasmissione dei valori istantanei da uno strumento con 10 canali	9600	1024	< 00:01
come sopra	1200	64	< 00:01
come sopra via radio, con tempo di attivazione portante di 500 ms	1200	128	< 00:02
Trasmissione dei dati elaborati orari e giornalieri di un giorno da uno strumento configurato con 10 canali ed elaborazioni <i>EEMinMedMaxDvst</i> di tipo <i>word</i>	9600	1024	< 00:07
come sopra	1200	64	< 00:50
come sopra via radio, con tempo di attivazione portante di 500 ms	1200	128	< 01:15
Trasmissione dei dati elaborati orari e giornalieri di un mese da uno strumento configurato con 10 canali ed elaborazioni <i>EEMinMedMaxDvst</i> di tipo <i>word</i>	9600	1024	< 03:30
come sopra	1200	64	< 25:00
come sopra via radio, con tempo di attivazione portante di 500 ms	1200	128	< 40:00
Trasmissione dei dati elaborati orari e giornalieri di un giorno da uno strumento configurato con 5 canali ed elaborazioni <i>EEMinMedMaxDvst</i> di tipo <i>word</i> per 4 canali ed elaborazioni <i>EEEolo3</i> per un canale di direzione del vento	9600	1024	< 00:04
come sopra	1200	64	< 00:30
Trasmissione dei dati elaborati orari e giornalieri di un mese da uno strumento configurato con 5 canali ed elaborazioni <i>EEMinMedMaxDvst</i> di tipo <i>word</i> per 4 canali ed elaborazioni <i>EEEolo3</i> per un canale di direzione del vento	9600	1024	< 02:00
come sopra	1200	64	< 15:00
Trasmissione della configurazione di funzionamento dello strumento (parametri di sistema, di rilievo, libreria dei codici operativi, calibrazione degli amplificatori, etc.):	9600	1024	< 00:30
come sopra	1200	64	< 03:20
Trasmissione di valori istantanei raggruppati su elaborazioni <i>EE10Ist</i> di tipo <i>word</i> per 8 canali (80 valori in totale) con tempi di acquisizione a scelta	9600	1024	< 00:01
Trasmissione di 80 elaborazioni <i>EEMinMedMaxDvst</i> di tipo <i>word</i> ripartite su vari canali	9600	1024	< 00:02

3.2 Il frame

Il frame è composto dalle seguenti informazioni (ne viene data codifica in linguaggio C perchè maggiormente diffuso per lo sviluppo di questo tipo di applicazioni):

Elemento	Significato
unsigned char <i>TheSyn</i> [3]	caratteri di sincronismo di inizio frame, il primo contenente il valore hex FD, i successivi il valore hex FF
unsigned char <i>ID</i>	<i>ID</i> della stazione slave; valore compreso fra 2 e 254; verso il master l' <i>ID</i> assume valore 1
unsigned char <i>NrFrame</i>	numero del frame corrente trasmesso; normalmente sempre 0 per trasmissioni che avvengono mediante un solo frame; per trasmissioni multiframe il valore di <i>NrFrame</i> viene incrementato ad ogni frame trasmesso da parte dello slave.
unsigned short <i>FrameLength</i>	indica la quantità di informazioni contenute all'interno di un frame; esse sono rappresentate da tutti i bytes inclusi fra <i>TheSyn</i> [0] e <i>EOT</i> (inclusi).
unsigned char <i>OpCode</i>	indica il codice operativo della richiesta; vedi §Significato di <i>OpCode</i>
<i>Dati</i>	numero variabile di bytes contenenti i dati da trasmettere; questo campo può essere vuoto
unsigned short <i>CRC</i>	<i>CRC</i> a 16 bit dei dati trasmessi fra <i>ID</i> e l'ultimo bytes di <i>Dati</i> (inclusi)
unsigned char <i>EOT</i>	identifica la fine del frame; contiene sempre il valore ASCII <i>EOT</i>

3.2.1 Calcolo del CRC

Il CRC viene calcolato in base al polinomio

$$X^{16}+X^{15}+X^2+1$$

Viene qui di seguito riportata la funzione utilizzata per il calcolo del CRC a 16 bit:

```

unsigned short CRC16(unsigned char *PtrBuf, unsigned short NrChar)
{
    unsigned short UnsValA,
                  UnsValB,
                  j;
    unsigned char CharValA,
                  i,
                  PEBit;

    for (j=0, UnsValB=0; j<NrChar; j++) {
        CharValA = (*PtrBuf++) ^ (unsigned char)UnsValB;
        UnsValB = UnsValB >> 8;
        for (i=0, PEBit=0; i<8; i++)
            PEBit ^= (CharValA >> i) & 0x01;
        if (PEBit)
            UnsValB ^= 0xC001;
        UnsValA = (unsigned)CharValA << 6;
        UnsValB ^= UnsValA;
        UnsValA = UnsValA << 1;
        UnsValB ^= UnsValA;
    }
    return UnsValB;
}
    
```

Dove: *PtrBuf*: puntatore al buffer dati

NrChar: numero di caratteri presenti nel buffer dati

Come menzionato nel paragrafo *Protocollo di comunicazione* i dati con formato binario a 16 o 32 bit, trasmessi dalle stazioni, hanno il byte più significativo nell'indirizzo minore ed il byte meno significativo all'indirizzo maggiore; è quindi necessario invertire la sequenza dei bytes che compongono tali dati, una volta ricevuti. Il calcolo del CRC da parte del master in ricezione ed in trasmissione va effettuato rispettivamente prima della conversione e dopo la conversione dei valori a 16 e 32 bit.

3.2.2 Significato di OpCode

Il valore di *OpCode* identifica il tipo della richiesta inviata oppure il significato del frame trasmesso.

Una serie di codici è dedicata alla gestione ed al condizionamento della comunicazione. Per esempio, se il master riceve un frame che, a seguito del controllo del CRC, risulta essere errato, provvederà all'invio di un frame avente il codice di richiesta impostato a *SendLastFrame*; gli altri codici sono raggruppati come facenti parte del gruppo dei codici di invio comandi o richiesta dati.

3.2.2.1 Codici di condizionamento della trasmissione

Valore	Codice	Campo Data trasmesso da Master	Campo Data ricevuto da Master	Significato
0x00	<i>Alert</i>	---	byte 0: significato dell'errore	identifica una condizione di errore accaduta durante il trasferimento dei dati; si veda in seguito per la lista dei possibili codici di errore trasmessi; questo frame viene sempre trasmesso dallo slave verso il master
0x01	<i>NotAck</i>	---	---	la richiesta inviata è stata riconosciuta valida come destinazione (master o slave) ma il campo OpCode del frame non è stato riconosciuto valido
0x02	<i>Ack</i>	---	---	la richiesta è stata ricevuta e correttamente gestita; questo frame viene trasmesso da slave a master nel caso in cui quest'ultimo abbia inviato un comando a cui non deve seguire, da parte dello slave, alcuna trasmissione di dati, oppure da master a slave ogni qualvolta il master riceva un pacchetto dati trasmesso all'interno di una sequenza multiframe; in questo caso l'Ack ricevuto dallo slave abilita lo stesso a trasmettere il frame successivo.
0x03	<i>SendLastFrame</i>	---	---	il ricevente richiede la ritrasmissione dell'ultimo frame da parte del trasmittente; questo frame viene normalmente inviato quando il frame ricevuto risulta non valido (CRC errato, lunghezza errata o ID non corrispondenti) e quindi deve essere ritrasmesso.
0x04	<i>EndTrasm</i>	---	---	il master richiede che la trasmissione multiframe da parte dello slave debba essere terminata; questo frame viene normalmente utilizzato al posto del frame <i>Ack</i> , se il master incontra qualche errore che impedisca la ricezione dei frames successivi (problemi di allocazione di memoria, disco pieno, etc.).
0x05	<i>LastFrame</i>	---	---	questo frame viene trasmesso dallo slave al termine di una trasmissione multiframe ed indica che il frame corrente è l'ultimo della serie trasmessa.

3.2.2.2 Codici di invio comandi o richiesta dati

Valore	Codice	Campo Data trasmesso da Master	Campo Data ricevuto da Master	Significato
0x06	<i>TrCnfSysStat</i>		strutture di tipo <i>BBCTime</i> e <i>BC SysInf</i>	ricezione da master delle informazioni di sistema: data attuale + configurazione sistema
0x08	<i>TrDataMemInf</i>		struttura di tipo <i>DataMemInf</i>	informazioni sulla configurazione e utilizzo della memoria dati
0x09	<i>SetDT</i>	struttura di tipo <i>BBCTime</i>		invio da master della data/ora corrente (attualmente non supportato)
0x0C	<i>TrAllChElab</i>		un valore float per ogni canale attivo, corrispondente al dato ultimo acquisito, già ingegnerizzato	ricezione da master degli ultimi dati acquisiti/preelaborati di tutti i canali attivi (risposta multiframe)
0x0D	<i>TrAllMemHeaders</i>		una o più strutture di tipo <i>RelMemHeader</i>	ricezione da master di tutte le testate di rilievo presenti nella memoria dati (risposta multiframe)
0x0E	<i>TrOneMemHeader</i>	bytes 0-1: indice del rilievo (0xff=ultimo rilievo); bytes 2-3: 0=trasmette solo la testata di rilievo; 1=trasmette la testata di rilievo e tutte le testate dei canali	una struttura di tipo <i>RelMemHeader</i> più una o più strutture <i>ChMemHeader</i> , se il parametro nei bytes 2-3 è uguale a 1.	Ricezione di una testata di rilievo selezionato più, a scelta, tutte le testate dei canali del rilievo stesso (risposta multiframe)
0x0F	<i>TrMemRel</i>	bytes 0-1: indice del rilievo (0xff=ultimo rilievo); bytes 2-3: 0=una elaborazione per frame; 1=più elaborazioni per frame; bytes 4-7: struttura <i>BBCTime</i> per selezione elaborazioni da questa data in poi; se questo vale 0, vengono trasmessi tutte le elaborazioni contenute nel rilievo	una o più strutture di elaborazione	ricezione da master di un rilievo (solo strutture di elaborazione); nel caso che il parametro nel campo <i>Data</i> contenuto nei bytes 2-3 sia uguale a 1, le elaborazioni vengono memorizzate nei frame in blocchi che non necessariamente terminano con la fine fisica di un'elaborazione; per questo motivo la conversione dei dati ricevuti può avvenire solamente al termine della trasmissione di tutti i frame (risposta multiframe)

3.3 Lista dei codici di errore

Seguono i valori ed i significati degli errori che vengono trasmessi nei frame di tipo *Alert*:

0x00	<i>NotAvailable:</i>	operazione/informazioni richieste non disponibili al momento.
0x01	<i>NotPermitted:</i>	operazione richiesta non permessa al momento.
0x02	<i>OutOfRange:</i>	indice di struttura fuori range.
0x03	<i>BadValue:</i>	valore non corretto.
0x04	<i>E2PError:</i>	errore di trasferimento informazioni nella memoria di configurazione.
0x05	<i>CriticalError:</i>	la richiesta ha provocato un errore critico non riconoscibile.
0x06	<i>IPCTimeOut:</i>	la richiesta ha causato un timeout fra il colloquio dei processori contenuti nella stazione.
0x07	<i>PADataSearchTimeOut:</i>	timeout scaduto durante la ricerca di informazioni nella memoria dati

3.4 Definizioni ed enumerazioni

Le definizioni e le enumerazioni che seguono vengono utilizzati per dimensionare e gestire alcune strutture dati in seguito definite. Si consideri comunque più aggiornata la definizione delle strutture direttamente contenute nei files sorgenti forniti con questo documento, in quanto in continua evoluzione.

```
enum ABCM {
    ABCM_DGB055, // modelli di strumento
    ABCM_DGB105, // strumento a 5 ingressi
    ABCM_DGB205, // strumento a 10 ingressi
    ABCM_DGB305, // strumento a 20 ingressi
    // strumento a 30 ingressi
};

enum AIM {
    AIMNone= 0x0, // maschere per invalidazioni su acquisizione
    AIMCheckDelta= 0x01, // nessun controllo
    AIMCheckMin= 0x02, // controlla delta con valore precedente
    AIMCheckMax= 0x04, // controllo su valore minimo ammesso
    AIMCheckFriend= 0x08, // controllo su valore massimo ammesso
    // controllo su presenza valore corretto canale 'compagno' (previsto;
    // attualmente non utilizzato)
};

enum ATDT {
    ATDTNoActive, // definizione tipi di trasmissione dati automatica via linea seriale (previsto;
    // attualmente non utilizzato)
    ATDTEventChan, // trasmissione non attivata
    ATDTAllIstChan, // trasmis. valore canale che ha generato l'evento
    ATDTAllElabChan, // trasmis. valori istantanei di tutti i canali al momento dell'evento
    ATDTSelIstChan, // trasmis. degli ultimi valori elaborati da tutti i canali al momento dell'evento
    // trasmis. valori istantanei dei canali selezionati per l'operazione al momento
    // dell'evento
    ATDTSelElabChan // trasmis. degli ultimi valori elaborati dei canali selezionati per l'operazione
    // al momento dell'evento
};
```

```

enum BR {
    BR300,           // non utilizzato in Babuc ABC
    BR600,           // non utilizzato in Babuc ABC
    BR1200,
    BR2400,
    BR4800,
    BR9600,
    BR19200
};

enum BT {           // tipo di batteria
    BTAlcaline,
    BTNichelCadmio,
    BTPiombo
};

enum BCEC {         // definizione dei contesti degli errori rilevati dal sistema
    BCECMem,        // errori rilevati dalla gestione del banco memorie statiche
    BCECInit,       // errori rilevati durante l'inizializzazione
    BCECIPC,        // errori IPC
    BCECGeneral,    // errori generali rilevati
    BCECTotContext // numero di contesti degli errori
};

enum BCFPT {
    BCFPTNothing    // Nessun significato
    BCFPTTR,        // Sensore a termoresistenza
    BCFPTOhm,       // Sensore a potenziometro
    BCFPTmV,        // Sensore con uscita in mV.
    BCFPTFrequency, // Segnale digitale a frequenza
    BCFPTCounter,   // Segnale digitale a contatore
    BCFPTGF,        // Sensore giunto freddo
    BCFPTBatt       // Valore di Batteria
};

enum BCLT {
    BCLTNothing,    // Nessuna linearizzazione
    BCLTPT100,
    BCLTNI100,
    BCLTTSB,
    BCLTTS1,
    BCLTSS1,
    BCLTTCJPTS,
    BCLTTCJDIN,
    BCLTTCK,
    BCLTTCS,
    BCLTTCT,
    BCLTTCE,
    BCLTOnOff,
    BCLTGF,
    BCLTMinAir60,   // Linearizzazione ventolina miniAIR60
    BCLTCO2,
    BCLTpH,
    BCLTCaricaBatt,
    BCLTTabLin1,    // tabelle di linearizzazione provvisorie in attesa dell'inserimento da PC
    BCLTTabLin2,
    BCLTTabLin3,
    BCLTTot        // Totale codici di linearizzazione
};

```

```

enum CDE {
    CDELast,
    CDEMin,
    CDEMed,
    CDEMax,
    CDEDevSta,
    CDEVar,
    CDETot
};

enum EE {
    EEDTMinMedMaxDvstIstB, // definizione elaborati ed eventi gestibili dal SSE; il suffisso ai nomi degli
    EEDTMinMedMaxDvstIstW, // elementi di questo enum B, W e F, indica che i dati vengono memorizzati
                                // utilizzando rispettivamente 1 byte (valore binario), 2 bytes (valore binario) e
                                // 4 bytes (valore floating point).
                                // minima, media, massima, deviazione standard, istantanea, min e max datate
                                // minima, media, massima, deviazione standard, istantanea, min e max
                                // datate
    EEDTMinMedMaxDvstIstF, // minima, media, massima, deviazione standard, istantanea, min e max datate
    EEMinMedMaxDvstB, // minima, media, massima, deviazione standard
    EEMinMedMaxDvstW, // minima, media, massima, deviazione standard
    EEMinMedMaxDvstF, // minima, media, massima, deviazione standard
    EEDTMinMaxTotB, // minima e massima datate e totale
    EEDTMinMaxTotW, // minima e massima datate e totale
    EEDTMinMaxTotF, // minima e massima datate e totale
    EEMinMaxTotB, // minima e massima e totale
    EEMinMaxTotW, // minima e massima e totale
    EEMinMaxTotF, // minima e massima e totale
    EEDTMinMaxB, // minima e massima datate
    EEDTMinMaxW, // minima e massima datate
    EEDTMinMaxF, // minima e massima datate
    EEMinMaxB, // minima e massima
    EEMinMaxW, // minima e massima
    EEMinMaxF, // minima e massima
    EEMedDvstB, // media e deviazione standard
    EEMedDvstW, // media e deviazione standard
    EEMedDvstF, // media e deviazione standard
    EEMedB, // media
    EEMedW, // media
    EEMedF, // media
    EETotB, // totale
    EETotW, // totale
    EETotF, // totale
    EEDurataMinB, // durata in minuti
    EEDurataMinW, // durata in minuti
    EEDurataMinF, // durata in minuti (la parte decimale intende i secondi)
    EEFreqDir1VelClass, // percentuali su 6 campi di velocita' di 1 settore
    EEFreqDir16VelClass, // percentuali su 6 campi di velocita' di 16 settori
    EEFreqDir18VelClass, // percentuali su 6 campi di velocita' di 18 settori
    EEFreqDir32VelClass, // percentuali su 6 campi di velocita' di 32 settori
    EEFreqDir36VelClass, // percentuali su 6 campi di velocita' di 36 settori
    EEEolo1, // settore prevalente: ampiezza angolo, bisettrice,
                                // dir media pesata, velocita' media e dvst
    EEEolo2, // genera sull'angolo giro: direzione media, velocita' media, dvst
    EEEolo3, // genera settore prevalente: bisettrice, direzione media, velocita' media,
                                // dvst della direzione, per il calcolo dir media, vel media, dvst standard

```

```

EEEolo4, // elaborato fittizio (non calcolato) settore prevalente: dir media, vel media,
// dvst della direzione, per il calcolo risultante dir media, vel media, dvst.
// Altri calcoli: bisettrice settore prevalente su 16 settori, perc. freq. dir. su
// 16 settori piu' calma, bisettrice settore prevalente.
// implementata per compatibilita' Anadata.

EE1IstB, // elaborato di 1 valore istantaneo
EE1IstW, // elaborato di 1 valore istantaneo
EE1IstF, // elaborato di 1 valore istantaneo
EE10IstB, // elaborato di 10 valori istantanei
EE10IstW, // elaborato di 10 valori istantanei
EE10IstF, // elaborato di 10 valori istantanei
EE60IstB, // elaborato di 60 valori istantanei
EE60IstW, // elaborato di 60 valori istantanei
EE60IstF, // elaborato di 60 valori istantanei
Free1, // codici liberi per future implementazioni
Free2,

....
Free29,
Free30,
EEStartEventElab, // define fittizio per delimitazione inizio elaborazioni ad evento
EEEvMinB, // evento di minimo (min)
EEEvMinW, // evento di minimo (min)
EEEvMinF, // evento di minimo (min)
EEEvMaxB, // evento di massimo (max)
EEEvMaxW, // evento di massimo (max)
EEEvMaxF, // evento di massimo (max)
EEEvDeltaB, // evento di scostamento (Delta)
EEEvDeltaW, // evento di scostamento (Delta)
EEEvDeltaF, // evento di scostamento (Delta)
EEEvPulseB, // evento di impulso (Pulse)
EEEvPulseW, // evento di impulso (Pulse)
EEEvPulseF, // evento di impulso (Pulse)
EEEvUserMessage, // evento di messaggio da utente
EETot, // numero totale di tipi definiti
EENothing, // non genera alcuna elaborazione
EERelMemHeader=250, // valore di elaborazione fittizio per riconoscimento testata di rilievo
EEFinderMemHeader, // valore di elaborazione fittizio per riconoscimento struttura
FinderMemHeader

EEJumpValue // valore di elaborazione fittizio per riconoscimento valore di ritorno a inizio
rilievo

};

enum EMEM { // definizione degli errori occorsi durante la gestione banco memorie statiche
EMNoError, // nessun errore
EMWriteTimeOut, // timeout durante la scrittura
EMRAWError, // errore di Read After Write
EMPointError = 0x04, // errore puntamento struttura in memoria statica
EMNoFindData = 0x08, // non e' stata trovata la struttura
EMReqNotAvlb = 0x10, // codice richiesta errato o non disponibile
EMMemoryFull = 0x20, // memoria piena
EMOutOfRange = 0x40, // indice struttura fuori range
EMBMSNotConnected = 0x80, // memocard statica non connesso
EMBMSNotFormatted = 0x100, // memocard statica non formattato
EMBMSLowBattery = 0x200, // batteria bassa su memocard
EMGeneralFailure = 0x400, // errore non riconosciuto
EMOpNotPermitted = 0x800 // operazione al momento non permessa
};

enum EIM { // maschere per invalidazioni su elaborazione

```

```

EIMNone= 0x0, // nessun controllo
EIMForceMin= 0x01, // controlla che il delta dei valori sia minore di...
EIMForceMax= 0x02, // controlla che il delta dei valori sia maggiore di...
EIMCheckValidData= 0x04 // controlla il numero di dati validi
};

enum EInit { // definizione degli errori occorsi durante l'inizializzazione del sistema
    EInoError, // nessun errore
    EInoProbe, // nessun sensore programmato
    EITooProbe, // troppi sensori programmati
    EIActTooFast = 0x04, // velocita' attivazione canali troppo elevata
    EINotValidProbe = 0x08, // una o piu' sonde programmate non valide
    EILowMemory = 0x10, // RAM memorizzazione temporanea dati insufficiente
    EITooManyCh = 0x20, // troppi canali configurati
    EILowBattery = 0x40, // livello batteria insufficiente
    EIRAWDataRAM = 0x80 // read after write errato nella RAM dati
};

enum EIPC { // definizione degli errori di tipo IPC
    EIPCNoError, // nessun errore riscontrato
    EIPCInvalidRequest, // richiesta IPC non valida
    EIPCInvalidContext, // contesto IPC non valido
    EIPCChLastElabOF = 0x04, // overflow su trasferimento struttura ChLastElab
    EIPCBadIndex = 0x08, // indice a struttura errato durante IPC
    EIPCTimeOut = 0x10, // timeout durante IPC
    EIPCRAWDRP = 0x20 // read after write errato nella RAM dual port
};

enum EGeneral { // definizione degli errori di tipo generico
    EGNoError, // nessun errore riscontrato
    EGBadSerFrameDim, // dimensione errata del frame seriale
    EGLowBattery // livello batteria insufficiente
};

enum FP { //
    FPTemperature, // Temperatura ambiente
    FPGravity, // Gravita'
    FPAtmPressure, // Pressione atmosferica
    FPOSLAltitude, // Altezza SLM
    FPLatitude, // latitudine
    FPLongitude, // longitudine
    FPSogliaEliof, // Soglia eliofania
    FPLimiteSI, // Limite Min/Max Stato SI
    FPCambioStato, // Limite del cambio di stato SI/NO
    FPLimiteNO, // Limite Min/Max Stato NO
    FPTot
};

enum ID { // definizione valori per ingegnerizzazione canale
    IDStartProbe,
    IDEndProbe,
    IDStartIng,
    IDEndIng,
    IDTotIngValue
};

```

```

enum IT {
    ITScaleLim,           // Ingegnerizzazione con limiti scale
    ITNothing,           // Nessuna ingegnerizzazione
    ITTot                 // Totale codici di ingegnerizzazione
};

enum SE {
    SEStandard,          // Elaborazione standard
    SEImpulsivo,         // Elaborazione sensori impulsivi
    SEStatoLogico,       // Elaborazione di stato SI/NO
    SEWind,              // Elaborazione per sensore di vento
    SEGas,               // Elaborazione per sensore di gas
    SESound,             // Elaborazione per sensore di suono
    SETot                // Nr totale di elaborazioni statistiche
};

enum TR {
    TRUserDep,           // Rilievo dipendente comandi avvio e fine dell'utente
    TRStartDT,           // Rilievo con avvio automatico in base data-ora impostata
    TREndDT,             // Rilievo con fine autom. in base data-ora impostata
    TRStartEndDT         // Rilievo con fine e avvio autom. in base data-ora impostata
};

enum SW {
    SWWaitingUsr,        // Macchina con rilievo in attesa di avviamento da operatore
    SWWaitingBat,        // rilievo gia' in corso ma sospeso, in attesa di riavviamento per ripristino
                        // tensione min. batteria
    SWWaitingDT,         // rilievo avviato in attesa di riconoscimento data-ora inizio rilievo
    SWRelOpening,        // Rilievo in fase di apertura
    SWRunning,           // Rilievo in corso
    SWEndRel,            // Rilievo terminato
    SWLowMem,            // Rilievo fermo causa fine spazio su E2PROM
    SWAutoPwrOff,        // Stato susseguente allo spegnimento automatico per batteria bassa
    SWCriticalErr        // Stato di funzionamento a seguito di errore critico di sistema
};

#define BC_MaxFisCh      30
#define BC_TotLogCh     60
#define MaxChElabEvent  5
#define MemFrmt          0xA5
#define MemNullRef       0L
#define NameLength       15
#define UnMisLength      7
#define RemarksLength    61
#define TotRefCh         6
#define VerLength        10
#define NumVelClass      6

```

3.5 Definizione delle strutture dati

3.5.1 Struttura BBCTime

La struttura *BBCTime* è semplicemente un tipo *unsigned long* ridefinito per comodità. Il valore contenuto esprime un tempo un numero di secondi, a partire dal 1 gennaio 1990 alle ore 00. Le funzioni per la conversione di questo valore in una struttura “più leggibile” sono contenute nei files sorgenti fornibili su richiesta dalla LSI.

3.5.2 Struttura BC_SysInf

La struttura *BC_SysInf* contiene le informazioni relative alla programmazione delle funzionalità del sistema.

DataMemInf TheDMI	informazioni sulla memoria connessa al sistema
unsigned char SSAProgramVer[VerLength]	stringa ASCII per l'identificazione della versione del programma di acquisizione;
unsigned char SSEProgramVer[VerLength]	stringa ASCII per l'identificazione della versione del programma di elaborazione;
unsigned char CnfProgramVer[VerLength]	stringa ASCII per l'identificazione della versione dei dati di configurazione;
unsigned char BitRate	esprime la velocità di trasferimento dati sulla porta di comunicazione seriale, a scelta fra i valori dell'enum <i>BR</i> ;
unsigned char NrMinAutoPowerOff	tempo in minuti relativi all'autospegnimento del display;
unsigned char FIBeeperActive	determina se il beeper viene attivato per ogni pressione sulla tastiera;
unsigned char ProtID	identificativo della stazione per le comunicazioni con protocollo di trasmissione dati su linea di comunicazione seriale;
unsigned char BatteryType	tipo di batteria di alimentazione installata, fra i valori offerti dall'enum <i>BT</i> ;
unsigned char AutoTxDataType	tipo di trasmissione automatica dei dati via linea di comunicazione seriale, in caso di scadenza del timer definito da <i>NrMinAutoDataTx</i> , a scelta fra i valori offerti dall'enum <i>ATDT</i> ;
unsigned char FIBinaryTx	definisce se la trasmissione dei dati via linea di comunicazione seriale avviene in formato binario; in caso contrario i dati possono essere inviati anche ad una stampante con porta seriale, connessa direttamente al sistema;
unsigned char FICircularMem	indica se la memorizzazione dei dati elaborati deve avvenire in modo circolare o lineare;
unsigned FactoryMatr	numero di matricola impresso in fabbrica;
unsigned UserMatr	numero di matricola selezionato dall'utente;
unsigned DimFrameTx	indica la quantità di dati che vengono trasmessi all'interno di un singolo frame di trasmissione; viene utilizzato per dimensionare i frame da trasmettere in modo da ottimizzare la quantità di dati inviati per frame rispetto alla qualità della linea di comunicazione; questo item può assumere i valori 32, 64, 256, 1024 e 2048;
unsigned NrMinAutoDataTx	numero di minuti per la trasmissione automatica dei dati, via linea di comunicazione seriale; se vale zero la trasmissione automatica con timer non è attivata;
unsigned long SysError[BCECTotContext]	errori di sistema, l'uno in OR logico con gli altri; <i>BCECTotContext</i> , ottenuto dall'enum <i>BCEC</i> , esprime i contesti degli errori; per ogni contesto è definito un enum che ne esprime tutti i valori possibili;
float BatteryLevel	ultimo livello di batteria, espresso in volt;
float FixedParams[FPTot]	parametri fissi inseriti dall'utente;

float ChanCoeffA[BC_MaxFisCh]	parametro <i>A</i> per calibrazione del canale fisico (un valore per ogni canale fisico);
float ChanCoeffB[BC_MaxFisCh]	parametro <i>B</i> per calibrazione del canale fisico (un valore per ogni canale fisico);
unsigned char RTSActPreTime	tempo di attivazione RTS in decimi di sec.
unsigned char FILineDriver	trasmissione seriale mediante line driver
unsigned long UserFuncEnableMask	maschera per abilitazione funzioni utente
unsigned char ABCModel	modello di strumento (enum ABCM)
char Free[25]	libero per espansioni future

3.5.3 Struttura DataMemInf

La struttura *DataMemInf* contiene le informazioni relative alla memoria dati del sistema (la memoria dati viene utilizzata per la memorizzazione delle elaborazioni e degli eventi sottoforma di rilievi).

unsigned char FIMemFrmt	indica se il banco memorie è già stato formattato; in caso positivo esso contiene il valore <i>MemFrmt</i> ; se non è connessa la memocard, questo elemento è comunque posto al valore di <i>MemFrmt</i>
unsigned char FIBMSConnected	flag di connessione memory card
unsigned char FIBMSInUse	indica se la memory card è correntemente utilizzata dallo strumento
unsigned char MemoryBatteryLevel	indicazione dello stato di carica della batteria della memory card
unsigned long MemAvlb	numero di bytes totali presenti nella memoria dati
unsigned long MemFree	numero di bytes liberi nella memoria dati
unsigned long ActWPtr	puntatore alla zona di memoria corrente di scrittura

3.5.4 Struttura RelMemHeader

La struttura *RelMemHeader* contiene le informazioni relative ad un rilievo memorizzato nella memoria dati del sistema.

unsigned char EEType	identificativo di testata di rilievo; contiene sempre il valore <i>EETRelMemHeader</i> ;
unsigned long PrevRMHAddr	indirizzo nella memoria statica della testata di rilievo precedente;
unsigned long NextRMHAddr	indirizzo nella memoria statica della testata di rilievo successiva;
unsigned long RelSize	dimensione del rilievo in bytes;
unsigned long FirstElabAddr	indirizzo della prima elaborazione, corrispondente alla più vecchia, in ordine temporale;
unsigned long NextElabAddr	indirizzo della prossima elaborazione che sarebbe stata scritta con il rilievo aperto; utilizzato per capire il termine della lettura (indirizzo di lettura corrispondente all'indirizzo indicato da questa variabile);
unsigned long FirstFMH	puntatore alla prima struttura di tipo <i>FinderMemHeader</i> memorizzata; se nessuna testata è stata memorizzata, questa variabile assume il valore di <i>MemNullRef</i> .
unsigned long LastFMH	puntatore all'ultima struttura di tipo <i>FinderMemHeader</i> memorizzata; se nessuna testata è stata memorizzata, questa variabile assume il valore di <i>MemNullRef</i> .
BBCTime StartDate	data d'inizio del rilievo;
BBCTime EndDate	data di termine del rilievo;
unsigned Index	indice interno della testata per reperimento dei dati, utilizzato per identificare univocamente il rilievo rispetto agli altri;

unsigned NrRel	numero di rilievo impostato dall'utente;
unsigned FactoryMatr	numero di matricola di fabbrica dello strumento che ha generato questo rilievo;
unsigned UserMatr	numero di matricola utente dello strumento che ha generato questo rilievo;
unsigned char NrChMemHeaders	numero di testate di canale che seguono questa testata di rilievo;
char Remarks[RemarksLen]	commento ASCII al rilievo impostato dall'utente;

3.5.5 Struttura ChMemHeader

La struttura *ChMemHeader* contiene le informazioni relative ad un canale memorizzato nella memoria dati del sistema.

unsigned char ChanElab	numero del sottocodice del canale dinamico, a scelta fra i valori offerti dall'enum <i>CE</i> ;
unsigned char NrFisCh	numero del canale fisico associato al canale;
unsigned char NrDecChar	numero di caratteri decimali per la visualizzazione dei valori istantanei ed elaborati;
unsigned char IndLogChan[TotRefCh]	indici ad altri canali di questo tipo utilizzati per reperire i dati utili alla preelaborazione o elaborazione del canale;
char Name[NameLength]	nome del canale;
char UnMis[UnMisLength]	unità di misura del canale;
unsigned long NrSecAct	tempo di attivazione del canale;
ElabEvent TabEE[MaxChElabEvent]	definizione delle elaborazioni o eventi generati dal canale;

3.5.6 Struttura ElabEvent

La struttura *ElabEvent* descrive le caratteristiche di un tipo di elaborazione o evento memorizzato dal sistema nella memoria dati.

unsigned char EEType	tipo di elaborato o evento, a scelta fra i valori offerti dall'enum <i>EE</i> ;
unsigned char AutoTxDataType	tipo di trasmissione automatica dei dati via linea di comunicazione seriale, in caso di attivazione dell'evento, a scelta fra i valori offerti dall'enum <i>ATDT</i> ;
unsigned char EIMask	maschera di selezione delle invalidazioni da applicare durante l'elaborazione dei dati, in OR tra i valori offerti dall'enum <i>EIM</i> ;
unsigned char InvMinValidData	numero minimo di dati validi in percentuale; utilizzato solo se il bit <i>EIMCheckNrErr</i> in <i>EIMask</i> è settato;
unsigned NrMinAct	numero di minuti di attivazione dell'elaborazione (non utilizzato se <i>EEType</i> corrisponde ad un evento, in quanto quest'ultimo viene attivato al manifestarsi dell'evento);
unsigned NrMinDataElab	numero di minuti per i quali vengono considerati validi i dati precedenti al tempo di attivazione dell'elaborazione, definito da <i>NrMinAct</i> ;
float ParamVal	valore del parametro per l'elaborazione o per l'evento;
float char InvMinValue	valore che esprime il delta minimo ammissibile fra tutti i dati utilizzati nell'elaborazione; utilizzato solo se il bit <i>EIMForceMin</i> in <i>EIMask</i> è settato;
float char InvMaxValue	valore che esprime il delta massimo ammissibile fra tutti i dati utilizzati nell'elaborazione; utilizzato solo se il bit <i>EIMForceMax</i> in <i>EIMask</i> è settato;

3.5.7 Struttura FECommon

La struttura *FECommon* descrive la parte comune a tutte le elaborazioni o eventi memorizzati nella memoria dati del sistema.

unsigned char EEType	tipo di elaborato o evento, a scelta fra i valori offerti dall'enum <i>EE</i> ;
unsigned char NrChan	numero di canale di riferimento, riferito alle testate di tipo <i>ChMemHeader</i> memorizzate all'inizio del rilievo;
unsigned char IndChanEE	indice alla struttura <i>TabEE</i> contenuta nella testata di canale di tipo <i>ChMemHeader</i> , utilizzata per ottenere le informazioni relative al tipo di elaborazione o evento;
BBCTime Date	data relativa all'istante dell'elaborazione o all'evento;

Seguono qualche definizione di strutture di elaborazione o evento; la descrizione completa può essere visionata direttamente nei files sorgenti forniti unitamente a questo documento. Il codice *[B/W/F]*, posto al termine del nome del tipo della struttura o all'inizio degli elementi della struttura, indica il tipo di memorizzazione prescelta; essa possiede i seguenti significati: B=byte, W=word, F=float.

3.5.8 Struttura FEMinMedMaxDvst[B/W/F]

Struttura dati di definizione dell'elaborazione finale di tipo *EEMinMedMaxDvst*.

FECommon Common	parte comune dell'elaborato/evento;
unsigned char DispPercent	percentuale dei dati corretti utilizzati per costituire l'elaborazione; il valore è un intero senza virgola;
B/W/F Min	valore minimo assunto dall'elaborazione;
B/W/F Med	valore medio assunto dall'elaborazione;
B/W/F Max	valore massimo assunto dall'elaborazione;
B/W/F Devst	valore della deviazione standard assunto dall'elaborazione;

3.5.9 Struttura FEDTMinMedMaxDvst[B/W/F]

Struttura dati di definizione dell'elaborazione finale di tipo *EEDTMinMedMaxDvst*.

FECommon Common	parte comune dell'elaborato/evento;
unsigned char DispPercent	percentuale dei dati corretti utilizzati per costituire l'elaborazione; il valore è un intero senza virgola;
BBCTime MinDT	data dell'evento di minima;
BBCTime MaxDT	data dell'evento di massima;
B/W/F Min	valore minimo assunto dall'elaborazione;
B/W/F Med	valore medio assunto dall'elaborazione;
B/W/F Max	valore massimo assunto dall'elaborazione;
B/W/F Devst	valore della deviazione standard assunto dall'elaborazione;

3.5.10 Struttura FEEolo3

Struttura dati di definizione dell'elaborazione finale di tipo *EEEolo3*.

FECCommon Common	parte comune dell'elaborato/evento;
unsigned char DispPercent	percentuale dei dati corretti utilizzati per costituire l'elaborazione; il valore e' un intero senza virgola;
unsigned BisetSettPrev	bisettrice del settore prevalente;
unsigned FreqDir[NumSetDir16]	distribuzione percentuale delle frequenze su 16 settori;
unsigned FreqCalma	percentuale calma di vento;
unsigned DirMedSettPrev	direzione media pesata del settore prevalente;
unsigned VelMedSettPrev	velocità media nel settore prevalente;
unsigned DevStndDirSettPrev	deviazione standard della direzione nel settore prevalente;
unsigned DirMedRis	direzione media risultante;
unsigned VelMedRis	velocità media risultante;
unsigned DevStndDir	deviazione standard della direzione secondo l'algoritmo di Nelson;

4 Protocollo MODBUS

4.1 Prestazioni funzionali

Babuc ABC si configura come unità *slave* rispetto all'unità di comunicazione esterna. Per questo motivo non è permesso che Babuc ABC intraprenda un inizio di trasmissione dati, senza che sia stato trasmesso un comando dall'unità *master*.

I dati elaborati e quindi trasmessi da Babuc ABC possono essere rappresentati da tipi *unsigned char*, *unsigned* (a 16 bit), *unsigned long* e *float* (32 bit a standard IEEE-754); il formato di memorizzazione dei tipi *unsigned* ed *unsigned long* utilizza il byte più significativo (MSB) all'indirizzo inferiore, ed il byte meno significativo (LSB) all'indirizzo superiore.

4.2 Linea fisica di comunicazione

La comunicazione può avvenire mediante standard RS232; sono supportati dall'unità solamente i pin di *RX*, *TX* e *Ground*. L'unità si trova sempre in stato di pronto.

Le velocità di comunicazione supportate sono da 1200 a 19200 bps. Il formato dei dati è 8 data bit, 1 stop bit, nessuna parità.

4.3 Frame di trasmissione

Le caratteristiche del frame di trasmissione, utilizzato per l'invio dei comandi e la ricezione dei dati, è strutturato come da tabella seguente:

Numero bit	Nome	Significato
8	Addr	Indirizzo periferica
8	Func	Codice funzione
n*8	Data	Parametri o dati
16	CRC16	Checksum

Il campo *Addr* permette la selezione di una singola periferica tra tutte quelle connesse sullo stesso media di comunicazione; il protocollo nativo di Babuc ABC contiene già un elemento utilizzato per lo stesso scopo, chiamato *ID di protocollo*; questo ID viene quindi utilizzato anche come *Addr* per MODBUS. Per questioni interne al protocollo nativo di Babuc ABC, il valore che può assumere l'ID di protocollo, e quindi il campo *Addr*, va da 0x02 a 0xFE.

Il campo *Data* può contenere, come primi 8 bits, un valore di *sottocodice*, utilizzato quando il campo *Func* assume alcuni particolari valori. Il sottocodice quindi è una particolare selezione di un certo comando espresso da *Func*.

Il campo *CRC16* contiene, in formato high-low, un valore di check a ridondanza ciclica. Il calcolo avviene partendo dal primo byte del frame (*Addr*) fino ad arrivare all'ultimo byte componente il campo *Data*. Il campo *CRC16* permette di determinare un eventuale errore di ricezione di un intero frame da parte dell'unità ricevente; nel caso in cui il frame giunga errato a Babuc ABC, quest'ultimo non risponde in alcun modo, in quanto l'errore potrebbe presentarsi nel campo *Addr*, e quindi il Babuc ricevente non è quello effettivamente richiesto per la comunicazione; nel caso in cui il frame errato giunga al *master*, quest'ultimo effettua nuovamente la richiesta; non è previsto un frame con codice *Func* utilizzato per la ritrasmissione dell'ultimo frame inviato da Babuc. Questo campo ha i bytes basso e alto invertiti.

La lunghezza massima del frame inviabile a Babuc ABC è di 256 bytes; la lunghezza massima del frame che Babuc invia all'esterno è di 2048 bytes.

4.4 Comandi e richieste MODBUS gestiti da Babuc ABC

Vengono elencati di seguito i comandi e le richieste MODBUS che Babuc ABC può gestire; per ognuno di essi viene descritto il contenuto del campo *Data*, quando utilizzato.

4.4.1 Richiesta Read Input Status

Significato

Questo comando Modbus viene interpretato da Babuc ABC come richiesta di lettura del suo stato del sistema; questa richiesta viene utilizzata per conoscere eventuali stati di errore dell'apparecchiatura; vengono inoltre fornite informazioni relativamente allo stato di acquisizione.

I bit disponibili all'interrogazione da parte del master sono i seguenti:

- bit 0 1=Errore durante la scrittura nella memoria dati (timeout o Read After Write);
- bit 1 1=Errore di gestione della memoria dati (puntatore inconsistente, richiesta non disponibile o non permessa, indice struttura fuori limite, errore non riconosciuto);
- bit 2 1=Memoria dati non connessa o non formattata;
- bit 3 1=Memoria dati piena;
- bit 4 1=Batteria di backup della memoria dati scarica;
- bit 5 1=Livello della batteria principale insufficiente;
- bit 6 1=Errore di intercomunicazione fra i due processori (IPC timeout, richiesta o contesto IPC non validi, indice struttura fuori range);
- bit 7 1=Errore nell'HW IPC (Read After Write in RAM dual port);
- bit 8 1=Rilievo fermo;
- bit 9 1=Rilievo in corso;
- bit 10 1=Rilievo in attesa rientro tensione di batteria principale;

Formato di richiesta

Valore campo *Func*:
0x02;

Valore campo *Data*:

bytes 0-1: numero del primo bit, con formato high-low, da cui partire ad eseguire la lettura; il valore del primo canale è rappresentato da 0x0000;

bytes 2-3: numero di bit da leggere, con formato high-low, partendo da quello specificato dai bytes 0-1; questo numero può assumere valore da 0x0001 ad 0x000b; nel caso esso assuma valore 0x0000, Babuc ABC non risponde alla richiesta; nel caso assuma valore superiore a 0x000b, vengono spediti solo i bit di stato effettivamente presenti.

Formato di risposta

Valore campo *Func*:
0x02;

Valore campo *Data*:

byte 0: numero di bytes spediti; esso dipende dalla quantità di bit richiesti da master; Babuc ABC invia al massimo 2 bytes (11 bit totali); in questo caso il primo byte contiene i primi 8 bit, il secondo i restanti 3 (di questo byte vengono utilizzati i soli bit 0-2).

bytes successivi: sequenza di bit selezionati dalla richiesta inviata da master.

4.4.2 Richiesta Read Input Register

Significato

Questo comando Modbus viene interpretato da Babuc ABC come richiesta di lettura dai canali di acquisizione dell'ultimo valore istantaneo, per ciascun canale attivo nel rilievo; da notare che Babuc ABC può avere più canali di acquisizione rispetto al numero di sensori realmente connessi, in quanto è in grado di gestire anche ulteriori grandezze derivate dai sensori stessi; un esempio che chiarisce il concetto riguarda la possibilità di acquisire due sensori di temperatura, una ambientale ed una di bulbo umido; a queste due grandezze è possibile accoppiarne una terza che rappresenta l'umidità relativa, direttamente calcolata dalle due temperature mediante il sistema psicrometrico; in questo caso quindi, avendo due canali fisici in ingresso (le due temperature) si hanno in uscita tre canali (due temperature + l'umidità relativa).

Questa richiesta non esegue distinzioni fra canali realmente acquisiti (*grandezze primarie*) e canali calcolati (*grandezze secondarie*).

Formato di richiesta

Valore campo *Func*:

0x04;

Valore campo *Data*:

bytes 0-1: numero del canale, con formato high-low, da cui partire ad eseguire la lettura; il valore del primo canale è rappresentato da 0x0000;

bytes 2-3: numero di canali da leggere, con formato high-low, partendo da quello specificato dai bytes 0-1; questo numero può assumere valore da 0x0001 al numero di canali gestiti dalla stazione; nel caso esso assuma valore 0x0000, Babuc ABC non risponde alla richiesta; nel caso assuma valore superiore al numero di canali gestito in quel momento dallo strumento, vengono spediti solo i canali effettivamente configurati.

Formato di risposta

Valore campo *Func*:

0x04;

Valore campo *Data*:

byte 0: numero di bytes componenti **totalmente** il campo *Data* meno 1;

bytes successivi: contengono sequenzialmente il valore di tutti i canali selezionati dalla richiesta; i dati sono espressi in formato *unsigned*.

Il valore di errore viene definito come 0xffff, mentre il dato non ancora acquisito viene indicato da 0xffff. Qualora il dato del canale non assuma questi due valori, deve essere sottratto a 0x8000 e diviso per una potenza di 10 in base al numero di cifre decimali utilizzate per il canale. Per esempio, per quanto riguarda la temperatura, che viene espressa con due decimali, si richiede che il valore intero venga diviso per 100 da cui si ottiene quindi un valore a virgola mobile con due decimali.

Segue un esempio di canali acquisiti con relative scale e precisioni:

Canale	Nome	Unità di misura	Scala	Precisione (cifre decimali)
1	Temperatura aria	°C.	-30÷70	2
2	Umidità relativa	%	0÷100	1
3	Direzione vento	° angolare	0÷360	0
4	Press. atmosferica	mBar	850÷1050	1
5	Radiazione globale	w/m ²	-150÷1500	0
6	Livello	m	0÷10	2
9	Velocità aria	m/s	0÷50	1
10	Precipitazione	mm	0÷100	1

4.4.3 Richiesta Reset Insorgenza Errore

Significato

Azzera gli eventuali errori riscontrati dal sistema e trasmessi con la richiesta *Read Input Status*. Questo comando azzera tutti gli stati di errore presenti nel sistema. I bits 8, 9 e 10 (stati del rilievo) non vengono influenzati da questo comando.

Formato di richiesta

Valore campo *Func*:
0x46;

Valore campo *Data*:

byte 0: sottocodice della richiesta, posto al valore 0x02;
byte 1: posto sempre al valore 0x00
byte 2: posto sempre al valore 0x01

Formato di risposta

Valore campo *Func*:
0x46;

Valore campo *Data*:

byte 0: sottocodice della richiesta, posto al valore 0x02;

4.4.4 Richiesta Approntamento Elaborazioni

Significato

Richiede a Babuc ABC di preparare un buffer dati elaborati per la successiva trasmissione, richiesta mediante altro comando; vengono fornite la data/ora di inizio e la data/ora di fine di ricerca; Babuc ABC esegue la selezione delle elaborazioni richieste nella memoria dati e prepara un buffer con i dati richiesti; la dimensione del buffer è fissa a 1024 bytes. In seguito a questa richiesta, Babuc ABC risponde immediatamente prima dell'inizio della ricerca dei dati in memoria, per non far aspettare inutilmente il sistema host.

Formato di richiesta

Valore campo *Func*:
0x46;

Valore campo *Data*:

byte 0: sottocodice della richiesta, posto al valore 0x04;
bytes 1-4: data/ora di inizio dati richiesti, espressa in secondi dal 01/01/90
bytes 5-8: data/ora di fine dati richiesti, espressa in secondi dal 01/01/90

Formato di risposta

Valore campo *Func*:
0x46;

Valore campo *Data*:

byte 0: sottocodice della richiesta, posto al valore 0x04;

4.4.5 Richiesta Trasmissione Elaborazioni

Significato

Richiede a Babuc ABC di trasmettere il buffer dati approntato precedentemente dalla richiesta *Approntamento Elaborazioni*. Per l'invio di questa richiesta, il sistema host deve attendere almeno 5 secondi rispetto alla richiesta *Approntamento Elaborazioni*, per dare la possibilità a Babuc ABC di terminare la ricerca dei dati e di riempire il buffer di trasmissione. E' obbligatorio che questa richiesta sia preceduta dal comando *Approntamento Elaborazioni*.

Formato di richiesta

Valore campo *Func*:

0x46;

Valore campo *Data*:

byte 0: sottocodice della richiesta, posto al valore 0x03;

byte 1: posto sempre al valore 0x00

byte 2: posto sempre al valore 0x01

Formato di risposta

Valore campo *Func*:

0x46;

Valore campo *Data*:

byte 0: sottocodice della richiesta, posto al valore 0x03;

bytes 1-2: valore unsigned short che esprime la dimensione del blocco di elaborazioni che seguono;

bytes 3-n: dati elaborati compresi nel periodo specificato dalla richiesta *Approntamento elaborazioni*. Siccome il buffer di trasmissione ha un limite di lunghezza fisica, può accadere che i dati ivi contenuti non arrivino alla data/ora finale specificata; per questo motivo il sistema host deve calcolare le due date in modo tale da prevedere che i dati possano essere contenuti nel buffer, oppure effettuando richieste successive con data/ora d'inizio corrispondente alla data/ora dell'ultimo elaborato inviato da Babuc ABC nel frame precedente.

Seguono esempi di strutture dati utilizzate per la trasmissione delle elaborazioni al sistema esterno. Ulteriori tipi di elaborazioni possono essere utilizzati, se lo strumento viene configurato in modo differente rispetto allo standard.

La struttura *FECCommon* è contenuta in ogni elaborazione:

```
typedef struct {
    unsigned char EEType,
                NrChan,
                IndChanEE;
    BBCTime     Date;
} FECCommon;
```

Struttura utilizzata per memorizzare l'elaborazione di tutti i canali tranne quello della direzione del vento:

```
typedef struct {
    FECCommon     Common;
    unsigned char DispPercnt;
    unsigned      Min,
                Med,
                Max,
                Devst;
} FEMinMedMaxDvstW;
```

Struttura di elaborazione per la direzione del vento:

```
typedef struct {
    FECommon          Common;
    unsigned char DispPercnt;
    unsigned          AmpSettPrev,
                    BisetSettPrev,
                    DirMedSettPrev,
                    VelMedSettPrev,
                    DevStndDirSettPrev;
} FEEolo1;
```

L'ultima struttura *FECommon* memorizzata nel buffer dati ha il campo *EEType* posto ad *EENothing*; in questo modo è possibile determinare il termine delle elaborazioni poste nel buffer.

4.4.6 Richiesta Modifica Data/ora di Sistema

Significato

Richiede a Babuc ABC di modificare la data/ora di sistema in base al contenuto del messaggio inviato.

Formato di richiesta

Valore campo *Func*:

0x43;

Valore campo *Data*:

byte 0: sottocodice della richiesta, posto al valore 0x0A;

byte 1: secondi

byte 2: minuti

byte 3: ora

byte 4: giorno

byte 5: non utilizzato

byte 6: mese

byte 7: anno (ultime due cifre: 1995=95, 2001 = 1)

byte 8: non utilizzato

byte 9: non utilizzato

Formato di risposta

Valore campo *Func*:

0x43;

Valore campo *Data*:

byte 0: sottocodice della richiesta, posto al valore 0x0A;

4.4.7 Calcolo del CRC

Per il calcolo del CRC fare riferimento al §Calcolo del CRC. Si tenga presente però che il valore di inizializzazione del calcolo del CRC (nell'esempio proposto è contenuto nella variabile *UnsValB*) per il protocollo MODBUS è 0xFFFF.