ISTITUTO NAZIONALE DI FISICA NUCLEARE
Laboratori Nazionali di Frascati

LNF-78/10(R)
15 Febbraio 1978

G. Bologna e B. Liotta: UNA LIBRERIA DI MACRO ISTRUZIONI IML PER LA GESTIONE DI SISTEMI CAMAC CON MINICALCOLATORI PDP 11 E CRATE CONTROLLER JCC 11.

Servizio Documentazione
dei Laboratori Nazionali di Frascati
Cas. Postale 13 - Frascati (Roma)

INFN - Laboratori Nazionali di Frascati
Servizio Documentazione

LNF-78/10(R)
15 Febbraio 1978

G. Bologna[x] e B. Liotta[o]: UNA LIBRERIA DI MACRO ISTRUZIONI IML PER LA GESTIONE DI SISTEMI CAMAC CON MINICALCOLATORI PDP 11 E CRATE CONTROLLER J CC 11.

1. INTRODUZIONE

1.1 Generalità

Con il presente lavoro è stato realizzato il linguaggio IML (Intermediate Level Language) per la gestione di sistemi CAMAC, usando come linguaggio ospitante l'assembler MACRO-11 (5) (6) dei calcolatori PDP 11.

E' stata creata una libreria di macro-istruzioni per poter effettuare l'espansione degli statements; mediante il loro uso si possono realizzare programmi in IML per gestire sistemi CAMAC interfacciati tramite il modulo J CC 11 (3) (4) con calcolatori DEC (Digital Equipment Corporation) della famiglia PDP 11 provvisti di sistema operativo RSX-11 M (9) o RSX-11 D (9).

(x) - Attualmente al CERN
(o) - Ospite LNF per il periodo della realizzazione della tesi di laurea.

1.2 Caratteristiche della realizzazione

Le caratteristiche principali di questa realizzazione dell'IML sono :

1) Possibilità di esecuzione di un programma IML su ogni calcolatore della famiglia PDP 11. Si è usato infatti come linguaggio ospitante, l'assembler MACRO 11, adoperando solo le istruzioni riconoscibili ed eseguibili dall'intera famiglia PDP 11.

2) Possibilità d'uso dell'IML su calcolatori forniti di sistemi operativi RSX-11 M o RSX-11 D (salvo ulteriori estensioni ad altri sistemi operativi). Si sono usate infatti solo le direttive compatibili tra i due sistemi operativi.

3) Disponibilità dell'intero gruppo delle macro-istruzioni sotto forma di macro-librerie.

4) Ottimizzazione della memoria occupata, sia in fase di assemblaggio che in fase di esecuzione del programma. Si è cercato infatti di rendere il corpo di istruzioni della macro, il più corto possibile ottimizzando la logica di assemblaggio; facendo inoltre uso delle direttive di assemblaggio condizionale si è ottenuto lo scopo di generare, per ogni particolare chiamata macro, il minimo indispensabile di istruzioni base necessarie per la successiva corretta esecuzione dell'azione. Inoltre tutte le informazioni interne collegate ad ogni dichiarazione, vengono realizzate tramite direct-assignements con ulteriore risparmio di memoria. Il tutto , naturalmente, compatibilmente con le richieste di generalità che un tale tipo di linguaggio prescrive alla macro-istruzione.

5) Ottimizzazione del tempo di esecuzione del programma. Per gli statements dichiarativi, tutte le strutture per il calcolo degli indirizzi, il calcolo stesso e l'eventuale riserva di memoria, avvengono, tramite opportune diret

tive, in fase di assemblaggio; cosa che comporta un notevole risparmio di spazio e di tempo in fase di esecuzione. Come media, uno statement di azione singola consiste di tre o quattro istruzioni assembler.
Si è avuto cura inoltre, tramite l'uso delle tabelle dei tempi di esecuzione delle istruzioni, di determinare, compatibilmente con le esigenze di generalità del programma, la forma migliore di codice assembler ai fini dell'ottimizzazione temporale di esecuzione delle istruzioni.

6) Esecuzione delle operazioni di Input/Output, tramite istruzioni MOV, direttamente dall'interno delle macro-istruzioni. Questo tipo di gestione dello I/O è particolarmente utile quando di abbia a che fare con tempi di acquisizione critici in tempo reale, in quanto le richieste di I/O vengono soddisfatte immediatamente senza dover passare attraverso la normale procedura di accodamento. Generalmente, infatti, i programmi eseguono lo I/O, nel sistema operativo RSX-11, sottoponendo le richieste all'Executive (9), il quale è in grado di coordinare e gestire l'uso delle unità fisiche; l'Executive manda poi la richiesta di I/O al device driver appropriato accodandola conformemente alla priorità del programma richiedente. In questo modo, tra la richiesta di I/O e la sua esecuzione, passa un certo tempo che, se è accettabile nella maggior parte dei casi, può non esserlo quando le richieste di tempo reale siano molto stringenti. La soluzione attuata comporta anche una maggiore modularità dell'IML realizzato per ciò che riguarda sviluppi ulteriori su altri sistemi operativi, in quanto non è stato necessario realizzare un driver,per il particolare sistema operativo, per gestite lo I/O del CAMAC.

7) Esecuzione dei trasferimenti multipli, cioè degli Uni-device block transfers e delle Multi-device Actions, senza dover fare ricorso al dispositivo hardware aggiuntivo : Autonomous transfer controller (ATC type 081) (3a)

Dato che il crate controller JCC-11 non è in grado di effettuare tramite hardware i trasferimenti di blocchi di dati, si è preferito rendere possibili tali trasferimenti facendo ricorso alla simulazione software piuttosto che allo uso del dispositivo hardware aggiuntivo; questo per estendere l'uso della realizzazione dell'IML anche a quei sistemi sprovvisti dell'Autonomous transfer controller.

8) Controllo e segnalazione dettagliatissimi degli errori sintattici e semantici.
Facendo uso di opportune direttive MACRO 11, associate ad assemblaggi condizionali e completate con messaggi diagnostici appropriati, si è ottenuto, in fase di assemblaggio, il riconoscimento e la segnalazione dettagliata (basti pensare che il numero di messaggi diagnostici supera il centinaio), dei possibili errori, sintattici e semantici commessi dall'utente nella stesura del programma.
Questo facilita grandemente sia il lavoro di apprendimento che l'uso dell'IML. Ogni statement IML viene assemblato solo dopo che ogni argomento è stato scrupolosamente analizzato ed è risultato corretto sintatticamente e semanticamente, altrimenti vengono segnalati sul file.lst che verrà scritto dal calcolatore sul terminale designato dall'utente per l'output listing, i messaggi di errore corrispondenti.
Una descrizione della forma, ed un sommario di tali messaggi vengono riportati in appendice D.

9) La sintassi realizzata è compatibile, tranne qualche eccezione dovuta alla particolare configurazione CAMAC-interfaccia-calcolatore, con le specifiche date in (1); in (1) è comunque previsto che le realizzazioni dell'IML possano essere diverse, in una certa misura, dalla loro definizione. Le differenza sono riportate nelle descrizioni delle funzioni svolte dalle varie macro-istruzioni.

2. PROGRAMMAZIONE CON L'IML E PROCEDURE OPERATIVE

2.1 Programmazione

Un programma scritto in linguaggio IML, o che comunque ne sfrutti gli statements, consiste di due parti distinte:

1) Definizione, tramite i <declaration-statements>, dei nomi simbolici (<identifiers>) che verranno usati successivamente negli <action-statements>.
   In questa fase l'utente, in base alla configurazione hardware CAMAC, ed all'uso che intende farne, definisce gli identificatori, specificando per ognuno tutta quella serie di informazioni necessarie per un corretto assemblaggio ed una successiva corretta esecuzione delle azioni che ne faranno uso.
   Supponiamo, per esempio, di avere a disposizione un modulo CAMAC, inserito nella stazione 9 (N=9), del crate 3 (C=3) e del quale ci interessa leggere e memorizzare il dato contenuto nel registro al sottoindirizzo 8 (A=8); azione che può essere effettuata tramite lo statement di azione singola SA. Si tratta dunque di definire il registro CAMAC e la memoria, per esempio tramite gli statements LOCD (Local Device declaration) e LOCM (Local Memory declaration):

```
LOCD    REG8,H,<3,9,8>
LOCM    MEMORY,V,16
```

   LOCD e LOCM sono i nomi delle macro che sviluppano le dichiarazioni; REG8 è l'identificatore stabilito dall'utente ed al quale verranno associate le informazioni relative agli indirizzi del CSR e del registro CAMAC; H è un simbolo definito nell'IML e serve a specificare il tipo di device che si sta dichiarando; <3,8,9> sono rispettivamen

te il numero del crate, della stazione e del sottoindirizzo. MEMORY è l'identificatore stabilito dall'utente per la memoria; V è un simbolo definito nell'IML e serve a specificare che il tipo di memoria che si sta dichiarando è una Variabile; 16 è la lunghezza di parola CAMAC in numero di bits.

2) Richiesta dell'azione voluta tramite gli statements di azione, aventi come argomenti gli identificatori definiti precedentemente. In questa fase, l'utente, in base al tipo di azioni richieste all'hardware CAMAC scrive gli statements di azione IML opportuni, eventualmente integrando il programma con istruzioni o direttive assembler.
Concludendo l'esempio precedente si tratta ora di scrivere lo statement di azione singola:

```
SA      RD1,REG8,MEMORY
```

SA è il nome della macro che sviluppa le istruzioni per la successiva esecuzione dell'azione; RD1 è il codice IML corrispondente alla funzione CAMAC Read Group-1 Register; REG8 e MEMORY sono i due identificatori definiti in precedenza.
L'esempio citato assumerà la forma:

```
.MCALL   .CAMAC,LOCD,LOCM,SA
.CAMAC
LOCD     REG8,H,<3,9,8>
LOCM     MEMORY,V,16
SA       RD1,REG8,MEMORY
.END
```

In appendice E viene riportato un esempio di programma scritto in linguaggio IML.

Nota 1: Tutte le macro IML utilizzate nel programma dell'utente, vanno specificate tramite la direttiva MACRO 11:

```
.MCALL   macro-name,...
```

Es: `.MCALL .CAMAC,LOCD,LOCM,SA,CZ`

Nota 2: In un programma IML, la prima macro IML che compare deve essere la macro ausiliaria .CAMAC che va chiamata senza argomenti; serve alla definizione dei simboli usati nell'IML. Tali simboli sono quindi riservati e l'utente deve avere cura di non ridefinirli nel corso del suo programma. Il loro elenco è dato in appendice A.

Nota 3: Nelle macro di uni-device block transfers e multi-device actions è stata usata la facility offerta dal MACRO 11 per la creazione automatica di simboli locali ((10) 4.2.1, descrizione della macro). L'utente deve quindi rispettare le convenzioni del MACRO 11 sui local symbols; in particolare non deve usare simboli locali nel range da 64$ fino a 128$ inclusi.

2.2 Procedure operative

Dopo la stesura del programma e la sua scrittura su opportuno supporto, si passa alla fase dell'assemblaggio le cui modalità sono descritte di seguito per i due sistemi operativi.

2.2.1 Procedure operative sotto RSX-11 M e RSX-11 D

MACRO-11

Per iniziare il MACRO-11 si ricorre ad uno dei metodi de-

scritti nel cap. 8 di (5) ad es.:

```
MCR>MAC
MAC>file.OBJ,file.lst=IML/ML,file.MAC
```

IML/ML specifica che nell'assemblaggio è richiesto l'uso della user-defined library di nome IML; le macro,infatti, sono definite nella libreria IML.MLB. Il programmatore deve inoltre dichiarare ogni macro IML usata, tramite la direttiva MACRO-11: .MCALL.

TASK BUILDER

```
MCR>TKB
TKB>file.image/PR,file.map=file.OBJ
TKB>//
```

/PR è lo switch "privileged", necessario per accedere alla extended page.

3. STATEMENTS DICHIARATIVI

Gli statements dichiarativi hanno il compito di associare all'identificatore specificato dall'utente, tutta quella serie di informazioni necessarie per un corretto assemblaggio ed una successiva corretta esecuzione degli statements di azione che ne faranno uso.

La distinzione tra identificatore locale, globale o esterno viene fatta dal tipo di statement dichiarativo usato (Local declaration, Export declaration, Import declaration); questo per semplificare le macro dichiarative.

Per ognuno dei tre tipi di dichiarazioni sono disponibili quattro diversi statements a seconda che si voglia definire un device (external reference), la memoria (internal reference), un canale o una LAM.

Lo statement dichiarativo è costitutito dalla macro dichiarativa che, se occorre, riserva la memoria, e si occupa del calcolo degli indirizzi; al suo interno definisce una nuova macro, con lo stesso nome dell'identificatore che si sta dichiarando, e che contiene, sotto forma di direct assignements, le informazioni per il corretto assemblaggio delle macro di azione che faranno uso di quell'identificatore. Gli statements dichiarativi, ad eccezione della dichiarazione di memoria di tipo buffer, non contengono codice eseguibile; il codice generato è sempre protetto da un'istruzione di salto opportuna.

3.1 Dichiarazioni locali

Gli statements dichiarativi locali specificano tutte le caratteristiche di un identificatore al quale si potrà fare riferimento solo nel programma attuale.

3.1.1 LOCD Dichiarazioni locali di device

Formato:

```
LOCD    (<h-name>,H,<cna>)
       /(<p-name>,P[,<cna>])
       /(<g-name>,G,<#-cna>/CONT,<cna>...)
       /(<c-name>,C,<#-cna>,<cna-f>,<cna-l>,<cna-i>)
       /(<q-name>,Q,<cna-f>,<cna-l>)
```

Nota:

Per gli argomenti<cna>, le parentesi angolari sono elemen

ti sintattici obbligatori.
I simbolo usati sono definiti sotto; per la definizione degli elementi sintattici vedere l'appendice A.

H = indirizzo-CAMAC costante, cioè indirizzo Hardware

P = indirizzo-CAMAC variabile, cioè Pointer

G = lista di indirizzi dati,Given

C = lista di indizi calcolati, Calculated

Q = Q-scan set

CONT = CONTinuazione di una LOCD del tipo G

Esempi:

```
LOCD   DISPL,H,<3,10,0>
LOCD   UNIT,P,<3,3,2>
LOCD   RTA,G,8,<1,2,0>,<1,8,10>,<3,5,2>,<2,5,1>,<1,1,0>,<4,5,3>
LOCD   RTA,G,CONT,<10,20,5>,<7,16,9>
LOCD   CALC1,C,18,<1;10,0>,<3,20,4>,<2,5,2>
LOCD   SCANQ,Q,<3,0,0>,<3,10,15>
```

Funzioni svolte e modalità d'uso

In base al simbolo specificato (H,P,G,C,Q), vengono effettuati,in fase di assemblaggio, i calcoli degli indirizzi dati, riservando, tranne nel caso H, la memoria opportuna, e associando poi tali informazioni all'identificatore dichiarato dall'utente. Le eventuali locazioni di memoria riservate sono protette da istruzioni di salto, per evitare possibili interpretazioni errate in fase di esecuzione.

H: Tramite i valori specificati di C,N ed A, viene effettuato il calcolo dell'indirizzo ed il risultato è posto sot-

to forma di direct-assignement:

h-name=cna

P: Se è specificato<cna>, viene effettuato il calcolo degli indirizzi del CSR, del device e del DHR, ed i risultati sono memorizzati nell'ordine in tre successive locazioni di memoria; se <cna> non è specificato, nelle tre locazioni viene posto il valore 0.

```
            BR      .+10
p-name:     .WORD   CSR,CNA,DHR
```

G: Per ogni gruppo di valori <cna> specificati, vengono calcolati gli indirizzi del CSR e del device, ponendo i risultati, uno dopo l'altro, in locazioni di memoria successive. L'identificatore punta alla prima di queste locazioni nella quale si trova il numero di elementi specificati.
Per poter dichiarare liste di indirizzi con più di 5 o 6 elementi, occorrerebbe continuare lo statement della LOCD nella riga successiva, (dato che la lunghezza massima di uno statement MACRO-11 è di 80 colonne), ma non essendo questo consentito si è resa necessaria la realizzazione di una LOCD di tipo G di appoggio. La forma è la stessa della LOCD normale, ma ora, al posto dell'argomento <#cna> , va posto il simbolo CONT. Questo nuovo statement si deve trovare nella riga immediatamente seguente la LOCD normale che si vuole controllare; sono ammesse più di una continuazione. Naturalmente il valore <#cna> della LOCD di testa deve riflettere il numero effettivo totale degli indirizzi specificati.

```
            JMP     .+...
g-name:     .WORD   CSR1,CNA1
            .WORD   CSR2,CNA2
            .WORD   CSR3,CNA3
                ...
```

C: A partire dai valori di<cna-f> (first), viene calcolata la lista di indirizzi, incrementando nell'ordine A,N e C con i valori <cna-i> (increment), finchè non si raggiunge <cna-l> (last). I valori di <cna-l> devono essere maggiori o uguali a quelli di <cna-f.> Il calcolo degli indirizzi segue un criterio di simmetria ed il risultato finale è analogo al caso G.

Q: Questo tipo di dichiarazione (Q-scan set)può essere usato solo dalla macro di azione MAD.
A partire dai valori di <cna-f> (first) e <cna-l> (last) viene calcolata e memorizzata una lista composta di 6 locazioni, contenenti nell'ordine i valori di: CSR,CNA-F,CN,CN,DHR,CNA-L.
Con un Q-scan set si può coprire al massimo un crate per cui, se l'utente vuole fare lo scanning di più crates, deve dichiarare un Q-scan set per ognuno di essi, ed usare poi una macro MAD per ogni Q-scan set.
Se il programmatore specifica il numero C-last, diverso da C-first, la macro, automaticamente forza C-last a riflettere lo stesso valore di C-first, segnalando la circostanza con il messaggio di avvertimento (non di errore!):

.PRINT ; >>>>> : THE CRATE NUMBERS IN Q-SCAN SET MUST BE IDENTICAL. e continuando l'assemblaggio con il nuovo valore di C-last.

```
                BR      .+16
q-name:         .WORD   CSR,CNA-F,CN
                .WORD   CN,DHR,CNA-L
```

in questa realizzazione dell'IML non si è inclusa la versione subscripted.

3.1.2 LOCM Dichiarazioni locali di memoria

Formato:

```
LOCM    (<k-name>,K,<word-lenght>,<literal>)
       /(<v-name>,V,<word-lenght>)
       /(<vl-name>,VL)
       /(<a-name>,A,<word-lenght>,<array-lenght>)
       /(<b-name>,B,<buffer>,<word-lenght>,
                 <buffer-lenght>[,<label-name>])
```

I simboli usati sono definiti sotto; per la definizione degli elementi sintattici vedere l'appendice A.

K = parola-CAMAC Costante
V = parola-CAMAC Variabile
VL = Variabile Logica
A = lista di parole-CAMAC,Array
B = Buffer

Esempi:

```
LOCM    MEM1,K,16,2750
LOCM    MEM2,V,24
LOCM    MEM3,VL
LOCM    MEM4,A,24,512
LOCM    MEM5,B,MEMB,16,64
```

Funzioni svolte e modalità d'uso

In base al simbolo specificato (K,V,VL,A,B) vengono associate all'identificatore dichiarato dall'utente, le informazioni fornite e, tranne nel caso K,si riserva la memoria necessaria, proteggendola, tramite opportune istruzio

ni di salto, da interpretazioni errate in fase di esecuzione.

K: Si fa corrispondere all'identificatore il valore specificato in <literal>, nella forma di direct assignement. La word-lenght deve essere sempre uguale a 16 (16 bit è la lunghezza massima di una parola PDP 11).

```
k-name=literal
```

V: A seconda della lunghezza di parola specificata, 16 o 24 bit, si riservano una o due locazioni di memoria, puntate dall'identificatore.

```
            BR      .+4
v-name:     .WORD   0
```

VL: Il trattamento di questo caso è analogo al caso V con word-lenght di 16 bit.

A: Consente di riservare con un unico identificatore un blocco di parole di memoria. Viene riservato un blocco opportuno di locazioni a seconda della word-lenght e dell'array-lenght. La prima locazione contiene il numero di elementi del blocco ed è identificabile tramite il nome simbolico <a-name>.
In questa realizzazione dell'IML non si è inclusa la versione subscripted AS.

```
            JMP     .+...
a-name:     .WORD   array-lenght
            .BLKW
```

B: Viene creato un buffer-header e si riserva un blocco opportuno di locazioni, la prima delle quali è puntata dalla label <buffer>, a seconda della word-lenght e della buffer-lenght.
Il buffer-header consiste in:

a) quattro locazioni di memoria contenenti rispettivamente:

1) il numero di elementi del buffer
2) la Posizione Corrente di Scrittura del Buffer
3) la Posizione Corrente di Lettura del Buffer
4) lo STATO del buffer.

Le parole 2) e 3) contengono l'indirizzo della prossima locazione disponibile rispettivamente in scrittura e in lettura.

b) il codice delle istruzioni di due subroutines di servizio del buffer, una per le operazioni di lettura ed una per quelle di scrittura sul buffer.

Da quanto risulta al punto b), in questo caso il codice generato è di tipo misto, cioè dichiarativo ed esecutivo.

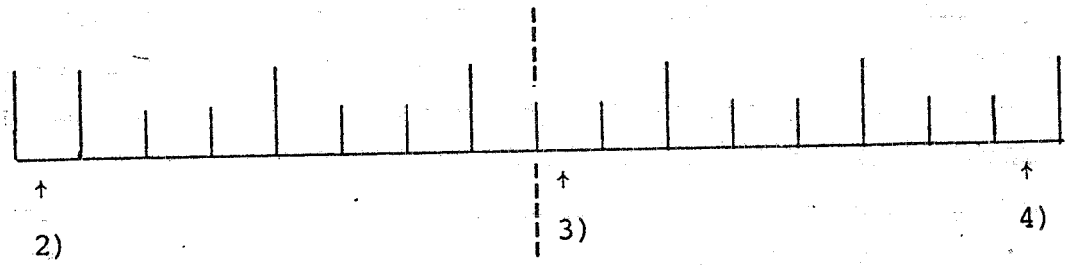
L'accesso al buffer avviene sempre in maniera sequenziale e dopo ogni operazione vengono effettuati dei controlli atti a garantire il corretto funzionamento.
Per ciò che concerne le operazioni di lettura da moduli CAMAC (cioè di scrittura dei dati letti sul buffer), si controlla che non venga superata la fine del buffer tenendo aggiornata la Posizione Corrente di Scrittura sul Buffer (P.C.S.B); In caso contrario, nella P.C.S.B si scrive l'indirizzo di partenza del buffer, la condizione d'errore viene segnalata nella parola di stato del buffer (bit 15=1) e l'azione viene terminata.
Nel caso l'utente abbia specificato nella dichiarazione LOCM del buffer, anche l'argomento <label-name>, il controllo del programma passerà ora, tramite l'istruzione

```
JSR     PC,<label-name>
```

alla routine di servizio dell'utente, avente come entry-point label-name. Tale routine dovrà terminare, se l'utente vuole che il controllo del programma ritorni all'istruzione successiva alla azione che ha causato il riempimento del buffer, con l'istruzione

```
RTS     PC
```

Per le operazioni di scrittura sui moduli CAMAC (cioè di lettura dati dal buffer), la sequenza delle operazioni è la seguente: si controlla che non venga superata la locazione in cui è stato memorizzato l'ultimo dato dell'ultima operazione di lettura CAMAC, e che non venga superata la fine del buffer. Nel primo caso, appena si raggiunge l'ultimo dato, viene aggiornata la Posizione Corrente di Lettura sul Buffer (P.C.L.B), si segnala la condizione nella parola di stato del buffer (bit 7=1), e l'azione in corso viene terminata. Nel secondo caso, quando si supera la fine del buffer, si riposiziona la P.C.L.B all'indirizzo di partenza del buffer, si segnala la condizione nella status del buffer (bit 0=1), e l'azione viene terminata.



Parola di stato del buffer

La parola di STATO del buffer contiene, istante per istante, l'informazione relativa alla situazione del buffer e può essere esaminata all'indirizzo <b-name+6> per controllare il risultato dell'azione che ha fatto uso del buffer. I possibili valori che la parola di STATO può assumere sono (in ottale) :

1) 0 : all'interno di ogni azione che implica l'uso del buffer, viene preventivamente azzerato lo STATO, per cui, se alla fine dell'azione lo STATO è ancora 0, significa che non si sono verificate condizioni anomale.

2) 100000 : buffer pieno. Durante lo svolgimento di una azione di scrittura sul buffer, è stata raggiunta l'ultima locazione disponibile,per cui la azione è stata terminata (la P.C.S.B contiene ora l'indirizzo iniziale).

3) 200 : in operazioni di lettura sul buffer è stata raggiunta l'ultima parola precedentemente scritta con altre operazioni di scrittura, per cui l'azione viene terminata (la P.C.L.B. contiene ora l'indirizzo di quella parola).

4) 1 : in operazioni di lettura sul buffer è stata raggiunta la fine del buffer,per cui l'azione viene terminata (la P.C.L.B. contiene ora l'indirizzo iniziale).

Se si vuole un controllo completo sul risultato di un'azione si deve controllare anche la parola di stato del canale (qualora l'azione ne faccia uso).

3.1.3 LOCC Dichiarazione locale di canale

Formato:

```
    LOCC    <ch-name>,<repeat>,<tally>,<status>

<repeat>::=<k-name>/<v-name>
<tally> ::=<v-name>
```

Esempio:

```
    ...
    LOCM    REP1,V,16
    LOCM    TAL1,V,16
    ...
```

```
    LOCC    CANAL1,REP1,TAL1,STAT
    ...
STAT: .WORD   0
    ...
    MOV     #64,REP1
    ...
```

Funzioni svolte e modalità d'uso

Lo scopo del canale è di controllare, tramite l'argomento <repeat>,il numero di volte che un'azione deve essere ripetuta, e di fornire all'utente, nella variabile status, le informazioni relative allo stato attuale ed alla conclusione dell'azione stessa.
L'utente, prima di chiamare questa macro, deve aver dichiarato <repeat> e <tally> in opportuni statements dichiarativi di memoria; inoltre nel corso del programma (o in programmi assemblati separatamente,ma che verranno legati, al momento del task-building, con quello attuale) deve venire riservata una parola, identificata dalla label <status>.

In questa realizzazione dell'IML non si è incluso l'argomento opzionale <channel>; non viene infatti adoperato un canale hardware.

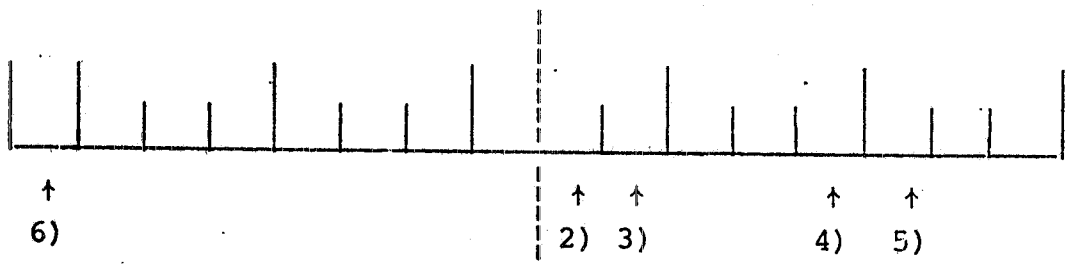
Un canale in cui <repeat>::=<k-name> causa, nelle azioni che ne fanno uso, un numero di ripetizioni sempre costante; mentre uno in cui <repeat>::=<v-name>, causa un numero variabile di ripetizioni, in dipendenza del valore attuale di <repeat> (per esempio, se prima di una azione l'utente modifica il contenuto di <repeat> ad sempio tramite l'istruzione assembler: MOV #128.,<repeat>
l' azione che usa il canale troverà ora in repeat quel valore, e ripeterà 128 volte l'azione).

La variabile tally viene usata, durante lo svolgimento di un'azione MNQ, per memorizzare l'indirizzo del primo registro che ha dato risposta Q=1.
Viene inoltre usata per memorizzare,alla fine di un'azio-

ne MAD, l'indirizzo dell'ultimo registro coinvolto nell'azione.

```
                    BR        .+12
                    .WORD     <k-name>
<ch-name>:          .WORD      .-2,<tally>,<status>
                      ...
                    BR        .+10
<ch-name>:          .WORD     <repeat>,<tally>,<status>
```

Per poter trattare i due casi: <repeat>::=<k-name>/<v-name> nella stessa maniera, la riservazione di memoria viene fatta come indicato sopra.



Parola di stato del canale

La parola di stato del canale contiene, istante per istante, l'informazione relativa alla situazione attuale della azione che sta usando il canale; può essere esaminata (all'indirizzo <status>) per controllare il risultato finale dell'azione. I possibili valori in ottale che essa può assumere sono:

1) 0 : all'inizio di ogni azione che fa uso di un cannale, la relativa parola di stato viene azzerata, e se tale valore viene ritrovato alla fine dell'azione vuol dire che l'azione è terminata

prima del previsto. Per esempio se si sta usando un buffer e questo si riempie prima che venga raggiunto il numero di ripetizioni richiesto con <repeat>; notare che la condizione di buffer pieno è rilevabile dalla parola di stato del buffer: bit 15 (segno negativo) impostato.

2)3)4) : prima di dare inizio all'azione, vengono effettuati (in fase di run) dei controlli per stabilire se il numero di ripetizioni richieste è compatibile con il numero di elementi degli argomenti di device e di memoria. Se i controlli effettuati danno esito negativo, l'azione non viene neppure iniziata e nella parola di stato viene impostato ad uno il bit opportuno. (Naturalmente, nel caso Uni-device, il controllo sul devive non viene effettuato, mentre, per funzioni CAMAC del tipo operate, non viene effettuato quello sulla memoria).
Le condizioni cui devono sottostare questi elementi sono:

$$repeat > 0$$
$$memoria \geq repeat$$
$$device \geq repeat$$

mentre la relazione in cui stanno tra loro memoria e device non ha importanza, sempre che siano soddisfatte le due con repeat. Non si può infatti chiedere la ripetizione di un'operazione se la memoria o gli indirizzi CAMAC specificati sono terminati; mentre si può invece per es.: dato un device di 200 registri, volerne leggere solo 100 (repeat=100 device), è memorizzarne i dati in un array di memoria di 100 parole (memoria=100=repeat<device).
I possibili risultati di questi controlli daranno i valori 2),3),4) alla parola di stato.

2) 200 : repeat > device
- il numero di ripetizioni richieste è maggiore del numero di registri CAMAC.

3) 100 : repeat > memoria
il numero di ripetizioni richieste è maggiore del numero di elementi di memoria disponibili.

4) 10 : repeat $\leq$ 0
il numero di ripetizioni richieste è negativo o nullo.

5) 4 : durante l'esecuzione di un address scan mode (macro MAD) è stato raggiunto l'ultimo indirizzo del Q-scan set senza che il repeat count sia stato raggiunto.

6) 100000 : done (fatto)
i controlli eseguiti hanno dato esito positivo e l'azione è stata eseguita correttamente fino alla fine.

3.1.4 LOCL Dichiarazioni locali di LAM

Formato:

```
LOCL    (<l-name>,SUB,<cna>[,<base-address>,
         <demand-line>])
       /(<l-name>,BIT,<bit-posn>,<cn>
        [<base-address>,<demand-line>])
```

<base-address> ::= indirizzo di base
<demand-line> ::= linea di domanda D_i
} (3),(4)

Nota: per gli argomenti<cna> e <cn>, le parentesi angolari sono elementi sintattici obbligatori.

Esempio:

```
LOCL    LAM1,SUB,<5,10,9>
LOCL    LAM2,BIT,13,<5,2>,274,5
```

Funzioni svolte e modalità d'uso

Tramite i simboli SUB e BIT si differenzia il tipo di LAM che si sta dichiarando e precisamente: SUB andrà usato per una LAM di tipo A, e BIT per una LAM di tipo I (vedi Cap. 5.).

SUB: viene generato un direct assignement nella forma

l-name=cna

inoltre si sfrutta la macro interna per trasmettere all'azione il valore del CSR.

BIT: viene generato un direct assignement nella forma

l-name=cn

e nella macro interna viene calcolato, a partire da <bit-posn>, che deve avere un valore compreso tra 1 e 16, il numero ottale corrispondente alla posizione specificata nella parola. Questa informazione verrà poi passata alle azioni che ne faranno uso.

In entrambi i casi, se la LAM che si sta dichiarando dovrà essere usata in uno statement UBL (Uni-device Block transfer, LAM synchronised), occorre specificare tra gli argomenti anche il base-address e la demand-line. Per la particolare struttura del crate controller JCC 11 (vedi (3),(4)), ad una data LAM del crate viene fatto corrispondere un vettore d'interruzione il cui indirizzo dipende dalla disposizione di alcuni jumper all'interno del modulo JCC 11. L'utente dovrà quindi specificare la configurazione dei jumper interni, cioè del base-address e della linea D_i corrispondente alla LAM che si sta dichiarando, del suo crate controller. Il calcolo del vettore di interruzione viene effettuato dalla macro e

l'informazione relativa viene trasmessa tramite la macro interna.

Non è stato necessario riservare parole di memoria nè effettuare calcoli in fase di run.

Questa LOCL differisce dalla definizione data in (1) poichè in quella compare come argomento opzionale solo <gl-number>.

3.2 Dichiarazioni export

Gli statements dichiarativi export hanno il compito di rendere globali, cioè disponibili per programmi assemblati separatamente, gli identificatori dichiarati nel programma corrente. Il formato è analogo a quello delle dichiarazioni locali con la sola differenza che gli identificatori sono ora resi globali. Per fare ciò le macro dichiarative export, prima di chiamare internamente le corrispondenti macro locali con gli argomenti uguali a quelli attuali, rendono globale, tramite la direttiva .GLOBL, l'identificatore specificato dall'utente.
I simboli usati sono definiti neiparagrafi che descrivono le corrispondenti dichiarazioni locali.

3.2.1 EXPD Dichiarazioni Export di Device

Formato:

```
EXPD    (<h-name>,H,<cna>)
       /(<p-name>,P[,<cna>])
       /(<g-name,G,<#-cna>/CONT,<cna>...)
       /(<c-name>,C,<#-cna>,<cna-f>,<cna-l>,<cna-i>)
       /(<q-name>,Q,<cna-f>,<cna-l>)
```

3.2.2 EXPM Dichiarazioni Export di Memoria

Formato:

```
EXPM    (<k-name>,K,<word-lenght>,<literal>)
       /(<v-name>,V,<word-lenght>)
       /(<vl-name>,VL)
       /(<a-name>,A,<word-lenght>,<array-lenght>)
       /(<b-name>,B,<buffer>,<word-lenght>,
         <buffer-lenght>[,<label-name>])
```

3.2.3 EXPC Dichiarazioni Export di Canale

Formato:

```
EXPC    <ch-name>,<repeat>,<tally>,<status>
```

3.2.4 EXPL Dichiarazioni Export di LAM

Formato:

```
EXPL    (<l-name>,SUB,<cna>[,<base-address>,
        <demand-line>])
       /(<l-name>,BIT,<bit-posn>,<cn>
        [<base-address>,<demand-line>])
```

Questo statement è stato inserito solo per rispettare la definizione generale dell'IML in quanto è esattamente uguale al corrispondente statement locale (LOCL). Per l'utente è comunque più comodo usare sempre e solo lo statement locale LOCL.

3.3 Dichiarazioni Import

Gli statements dichiarativi import hanno il compito di rendere disponibili nel programma corrente, tutte le

informazioni necessarie ad un corretto assemblaggio delle macro di azione che fanno uso di identificatori già dichiarati come export in altri programmi, assemblati separatamente, ma che verranno legati a quello attuale in fase di Task building.
In queste dichiarazioni non è quindi necessario, in genere, fornire tutte le stesse informazioni delle corrispondenti export; per esempio, nel caso di una dichiarazione di memoria di un buffer, tutto il buffer header, le routines di servizio e la memoria saranno già stati riservati nell'altro programma con entry points globali.
I simboli usati sono definiti nei paragrafi che descrivono le corrispondenti dichiarazioni locali.

3.3.1 IMPD Dichiarazioni Import di Device

Formato:

```
IMPD    (<h-name>,H,<c>)
       /(<p-name>,P)
       /(<g-name>,G)
       /(<c-name>,C)
       /( q-name>,Q)
```

Esempio:

```
IMPD    DISPL,H,3
IMPD    DEVP,P
IMPD    GIVAD,G
IMPD    ADCALC,C
IMPD    QSCAN,Q
```

Nota: In questo caso la realizzazione dell'IML si differenzia dalla definizione data in (1) e precisamente, nel caso H occorre specificare l'argomento <c>; mentre nei casi G e C non occorre specificare gli argomenti <#cna>.

3.3.2 IMPM Dichiarazioni Import di Memoria

Formato:

```
IMPM    (<k-name>,K,<word-lenght>)
       /(<v-name>,V,<word-lenght>)
       /(<vl-name>,VL)
       /(<a-name>,A,<word-lenght>)
       /(<b-name>,B,<word-lenght>)
```

Esempio:

```
IMPM    CONST1,K,16
IMPM    AZUL,V,24
IMPM    LOGVAR,VL
IMPM    MEM,A,16
IMPM    BUFFER,B,16
```

3.3.3 IMPC Dichiarazione Import di Canale

Formato:

```
IMPC    <ch-name>
```

Esempio:

```
IMPC    CHAN2
```

Nota: In questa realizzazione dell'IML non occorre specificare l'argomento <r-type> come da (1).

3.3.4 IMPL Dichiarazione Import di LAM

Formato:

```
IMPL    (<l-name>,SUB,<cna>[,<base-address>,
         <demand-line>])
       /(<l-name>,BIT,<bit-posn>,<cn>
        [<base-address>,<demand-line>])
```

Esempio:

```
IMPL    LAM7,SUB,<3,7,0>,274,7
IMPL    ECC,BIT,3,<3,10>
```

Modalità d'uso

Analoghe a quelle descritte nel paragrafo riguardante la LOCL (4.1.4). Questo statement dichiarativo è esattamente uguale a quello locale ed è stato introdotto solo per completezza con la definizione (1) dell'IML. Per l'utente è comunque più comodo usare sempre e solo lo statement locale LOCL.

4. STATEMENTS DI AZIONE DI MODULI

Gli statements di azione di moduli specificano uno o più comandi da eseguire sull'hardware CAMAC. Generalmente si tratta di azioni su registri o altro a livello di sottoindirizzo nei moduli dell'utente.

Conformemente al modo di accesso al CAMAC, gli statements causeranno una delle seguenti:

- Azione singola
  viene eseguito un ciclo CAMAC e se la funzione CAMAC era nel gruppo read o write viene trasferita una parola di dati.

- Uni-device block transfer
  viene trasferito un blocco di dati tra un singolo indirizzo CAMAC e la memoria del calcolatore.

- Multi-device action
  viene eseguita una serie di azioni singole, su indirizzi CAMAC differenti usando la stessa funzione CAMAC. Se tale funzione è nel gruppo read write, viene trasferito un blocco di dati tra il CAMAC e la memoria del calcolatore, una parola CAMAC da o per ciascun indirizzo CAMAC.

Si riporta qui, per comodità del lettore, la definizione, come da appendice A, degli elementi sintattici:
<f-read>, <f-write>, <f-operate>.

```
<f-read>     ::= (F00/RD1)/(F01/RD2)/(F02/RC1)/(F03/RCM)
                 /F04/F05/F06/F07

<f-write>    ::= (F16/WT1)/(F17/WT2)/(F18/SS1)/(F19/SS2)
                 /F20/(F21/SC1)/F22/(F23/SC2)

<f-operate>  ::= (F08/TLM)/(F09/CL1)/(F10/CLM)/(F11/CL2)
                 /F12/F13/F14/F15
                 /(F24/DIS)/(F25/XEQ)/(F26/ENB)/(F27/TST)
                 /F28/F29/F30/F31
```

4.1 Azioni singole

Gli statements IML di azione singola rappresentano il modo fondamentale di richiesta di un'azione CAMAC; possono essere usati con qualsiasi funzione ed operare su qualunque indirizzo CAMAC.
L'azione singola esegue un ciclo CAMAC e, se la funzione specificata è di tipo read o write, trasferisce una parola CAMAC nella o dalla memoria specificata.

Esistono tre tipi di azioni singole:

SA (Single Action)

SJQ (Single action Jumping on Q=1)

SJNQ (Single action Jumping on Q=0)

per ognuna di esse viene generato, tramite opportuni assemblaggi condizionali, il codice di istruzioni ottimizzato in relazione agli argomenti attuali specificati dall'utente. Come media, il numero di istruzione assembler generate per un'azione singola, è di tre o quattro, a seconda della lunghezza di parola trattata.

4.1.1 SA Single Action

Formato:

```
SA      (<f-read>,<dev-1>,<mem-1>/<dev-1>)
       /(<f-write>,<dev-1>,<mem-2>/<dev-1>)
       /(<f-operate>,<dev-1>)

<dev-1>  ::= <h-name>/<p-name>
<mem-1>  ::= <v-name>/<b-name>
<mem-2>  ::= <k-name>/<v-name>/<b-name>
```

Esempio:

```
SA      FO1,ADC,VAR
SA      WT1,ADC,VAR
SA      F25,PAR
```

Funzioni svolte e modalità d'uso

La SA è la macro di azione singola di base, può essere adoperata con tutte le funzioni CAMAC, read, write e operate ed esegue un ciclo CAMAC accompagnato, nel caso di funzioni read o write, dal trasferimento di dati sulle corrispondenti linee.
Data la particolare costituzione dell'interfaccia JCC 11, sono ammesse azioni singole tra registri CAMAC che stiano

sullo stesso crate. Se l'utente vuole quindi eseguire una operazione di lettura da registro e trasferire il dato letto su di un altro registro nello stesso crate con f-read, e viceversa con f-write, dovrà usare nella singola azione, al posto dell'argomento <mem-1> o <mem-2>, un argomento del tipo <dev-1>. Se il numero del crate del primo device non è uguale a quello del secondo, solo nel caso di due devices di tipo H l'errore viene segnalato dal messaggio:
.ERROR ; >>>>> "dev-mem": ILLEGAL INTER-CAMAC TRANSFER.
negli altri casi verrà assunto che i registri stiano ambedue sul crate del primo.
Non è stata realizzata la SA per f-status, data la scelta possibile consentita in (1) A5.1.

4.1.2 SJQ (Single action Jumping on Q=1)

Formato:

```
SJQ   (<f-read>,<dev-1>,<mem-1>/<dev-1>,<label-name>)
     /(<f-write>,<dev-1>,<mem-2>/<dev-1>,<label-name>)
     /(<f-operate>, < dev-1 >,<label-name>)

<dev-1>  ::=  <h-name>/<p-name>
<mem-1>  ::=  <v-name>/<b-name>
<mem-2>  ::=  <k-name>/<v-name>/<b-name>
```

Esempio:

```
         SJQ   FO3,MOD,ADC,LAB1
         SJQ   F27,SCAL,LAB2
         ...
LAB1:    ...
         ...
LAB2:    ...
```

Funzioni svolte e modalità d'uso

La SJQ è una macro di azione singola nella quale viene eseguito un salto alla locazione specificata dallo argomento < label-name> se l'operazione CAMAC eseguita ha dato risposta Q=1.

Valgono per essa le stesse avvertenze date per la SA.

4.1.3 SJNQ (Single action Jumping on Q=0)

Formato:

```
SJNQ   (<f-read>,<dev-1>,<mem-1>/<dev-1>,<label-name>)
      /(<f-write >,<dev-1>,<mem-2>/<dev-1>,<label-name>)
      /(<f-operate >,<dev-1> ,<label-name>)

<dev-1>  ::= <h-name>/<p-name>
<mem-1>  ::= <v-name>/<b-name>
<mem-2>  ::= <k-name>/<v-name>/<b-name>
```

Esempio:

```
         SJNQ   FO5,PXP,MEM7,SAL1
         SJNQ   F1O,ALM,SAL2
         ...
SAL1:    ...
         ...
SAL2:    ...
```

Funzioni svolte e modalità d'uso

La SJNQ è una macro di azione singola nella quale viene eseguito un salto alla locazione specificata dall'argomento label-name se l'operazione CAMAC che ha avuto luogo ha dato risposta Q=O.
Valgono per essa le stesse avvertenze date per la SA.

4.2 Uni-device Block Transfers

Gli statements uni-device block transfer vengono usati con funzioni CAMAC read o write per trasferire un blocco di parole CAMAC, una alla volta, tra un singolo registro e la memoria del calcolatore.
Una volta partita l'azione, il trasferimento di dati terminerà per una delle seguenti cause:

1) raggiungimento del repeat-count (numero di ripetizioni richieste tramite il canale attualmente specificato).
2) il verificarsi di una condizione di errore nel buffer (se la memoria specificata è di questo tipo).
3) Una condizione di errore.

Il modo in cui è terminata l'azione è registrato nella variabile status del canale.
Esistono tre tipi di uni-device block transfers:

UBL (Uni-device Block transfers, LAM-synchronised)

UBC (Uni-device Block transfers, Controller-synchronised)

UBR (Uni-device Block transfers, Repeat while not Q)

Per ognuno di essi viene generato, tramite opportuni assemblaggi condizionali, il codice di istruzioni ottimizzato in relazione agli argomenti attuali specificati dall'utente.
Come si è già avuto modo di dire, i block transfers vengono eseguiti tramite simulazione software del dispositivo hardware Autonomous Transfer Controller. Per questo motivo nelle azioni non compare l'argomento opzionale <return> (vedi (1): A.6 e Nota a pag. 11).

4.2.1 UBL Uni-device Block transfers, LAM synchronised

Formato:

UBL <f-read>/<f-write>,<dev-1>,<mem-3>,<l-name>,
<ch-name>,<psw>

<dev-1> ::= <h-name>/<p-name>
<mem-3> ::= <a-name>/<b-name>
<psw> ::= processor status word (con cui si vuole sia servita la LAM)

Esempio:

UBL FOO,PARUN,ARRAY,LAMPAR,CANAL,300

Funzioni svolte e modalità d'uso

Nello statement di trasferimento di blocchi UBL, il modulo specificato genera una LAM quando è pronto a partecipare ad una operazione di trasferimento di dati; viene trasferito un dato in sincronismo con ogni LAM generata.
La LAM di sincronizzazione corrispondente al device, genera un'interruzione con vettore specificato dall'utente nella precedente dichiarazione dell'identificatore < l-name > tramite il base address e la demand line (vedi 3.1.4), e con nuova processor status word assegnata dall'utente tra gli argomenti della UBL (<psw>).
La UBL provvede da sola a tutte le operazioni necessarie alla preparazione e gestione dell'interruzione; in particolare, seguendo le norme CAMAC (2), per abilitare la sorgente LAM, vengono usate le funzioni F26 all'indirizzo LAM-A, oppure F19 in A(13) (LAM MASK REGISTER) all'indirizzo LAM-I; Per disabilitare la sorgente LAM si usano le funzioni F24 all'indirizzo LAM-A o F23 in A(13) all'indirizzo LAM-I; per azzerare la sorgente LAM vengono usate le funzioni F10 allo indirizzo LAM-I, oppure F23 in A(12) (LAM STATUS REGISTER) all'indirizzo LAM-I.
L'utente deve quindi assicurarsi che il modulo su cui vuole eseguire l'azione UBL, risponda ai requisiti CAMAC standard.

Nota:

Il contenuto dei registri CSR,DMR e del vettore di interruzione vengono salvati nella stack in entrata alla UBL, dato che durante lo svolgimento dell'azione occorre modificarli; prima di uscire dalla UBL verranno ripristinati i contenuti originali.

In uscita dalla UBL la sorgente LAM è azzerata e disabilitata.

Se più sorgenti LAM sono connesse alla stessa Demand line del DMR, non occorre disabilitare quelle che non interessano nel caso attuale, in quanto l'acquisizione nella routine di servizio della LAM avverrà solo se l'interruzione proviene dalla sorgente LAM corrispondente al modulo specificato dall'utente.

4.2.2 UBC (Uni-device Block transfers, Controller synchronised)

Formato:

```
UBC   <f-read>/<f-write>,<dev-1>,<mem-3>,<ch-name>

<dev-1>   ::= <h-name>/<p-name>
<mem-3>   ::= <a-name>/<b-name>
```

Esempio:

```
UBC    FO1,REG,BUF,CAN1
```

Funzioni svolte e modalità d'uso

Nello statement di trasferimento di blocchi UBC, i trasferimenti dei dati vengono sincronizzati dal computer, si assume cioè che il rateo di disponibilità dei dati sia maggiore di quello massimo di trasferimento del calcolatore.

4.2.3 UBR (Uni-device Block transfers,Repeat while not Q)

Formato:

UBR <f-read>/<f-write>,<dev-1>,<mem-3>,<ch-name>

<dev-1> ::= <h-name>/<p-name>
<mem-3> ::= <a-name>/<p-name>

Esempio:

UBR F16,APP3,MARR3,CHAN

Funzioni svolte e modalità d'uso

Nello statement di trasferimento di blocchi UBR, viene realizzato il REPEAT MODE, definito in (2), 5.4.3.2 . Tale modo viene usato quando i trasferimenti di dati per o da un registro devono avvenire quando il registro è pronto. Un modulo che contenga un registro al quale si vuole accedere in REPEAT MODE deve quindi generare Q=1 durante un'operazione di lettura o scrittura se il registro è pronto a partecipare ad un trasferimento di dati, Q=0 se il registro non è pronto.

4.3 Multi-device actions

Tramite questi statements viene eseguita,su ciascuno della serie di registri CAMAC specificati,l'azione singola richiesta tramite la funzione CAMAC.
La serie di azioni terminerà per una delle seguenti cause:

1) raggiungimento del repeat-count (numero di ripetizioni richieste tramite il canale attualmente specificato).
2) Il verificarsi, nel caso di funzioni read o write che usino un buffer, di una condizione di errore nel buffer.

3) Una condizione d'errore.

Il modo in cui è terminata l'azione è registrato nella variabile status del canale.
Esistono tre tipi di multi-device actions:

MA	(Multi-device Action)
MNQ	(Multi-device action, repeat while Not Q)
MAD	(Multiple Address scan)

per ognuna di esse viene generato, tramite opportuni assemblaggi condizionali , il codice di istruzioni ottimizzato in relazione agli argomenti attuali specificati dall'utente.
Come si è già avuto modo di dire, le azioni multiple vengono effettuate tramite simulazione software del dispositivo hadrware Autonomous Transfer Controller. Per questo motivo nelle azioni non compare l'argomento opzionale
< return > (vedi (1): A.7 e Nota a pag. 11),

4.3.1 MA (Multi-device Action)

Formato:

```
MA   (<f-read>/<f-write>,<dev-2>,<mem-3>,<ch-name>)
    /(<f-operate>,<dev-2>,<ch-name>)

<dev-2> ::= <g-name>/<c-name>
<mem-3> ::= <a-name>/<b-name>
```

Esempio:

```
MA   F17,MODUL5,MEMORY,CANALE
MA   F10,MOD3,CAN
```

Funzioni svolte e modalità d'uso

La MA è l'azione multi-device di base; con essa si possono usare tutte le funzioni CAMAC e se la funzione è del tipo read o write, avrà luogo una serie di trasferimenti di dati (un dato da o per ogni registro) da registro CAMAC a memoria o viceversa.

4.3.2 MNQ (Multi-device action,repeat while Not Q)

Formato:

MNQ (TLM/FO8/TST/F27),<dev-2>,<ch-name>

<dev-2> ::= <g-name>/<c-name>

Esempio:

MNQ TLM,GIVLAM,CHAN

Funzioni svolte e modalità d'uso

La MNQ è uno statement di azione che termina dopo la prima operazione con risposta Q=1; può venire quindi usata tutte quelle volte in cui è richiesta una rapida ricerca di quale, della serie dei registri specificati,ha posto ad 1 la LAM.
Le uniche funzioni CAMAC utilizzabili in questa azione sono le due funzioni operate: FO8/TLM (test look-at-me) o F27/TST (test status).
Il risultato dell'azione, cioè quale degli indirizzi esaminati ha dato risposta Q=1, viene memorizzato, e può quindi essere letto, nella variabile tally del canale usato (un valore zero in tally dopo l'esecuzione di una MNQ, significa che nessuno dei registri esaminati ha la LAM impostata.

4.3.3 MAD (Multiple-Address scan)

Formato:

```
MAD    (<f-read>/<f-write>,<dev-3>,<mem-3>,<ch-name>)
      /(<f-operate>,<dev-3>,<ch-name>)
<dev-3>  ::=  <q-name>
<mem-3>  ::=  <a-name>/<b-name>
```

Esempio:

```
MAD    FO2,RTM,MEMA,CH3
MAD    F25,QSC,CANALE
```

Funzioni svolte e modalità d'uso

La MAD realizza l'ADDRESS SCAN MODE descritto in (2) 5.4.3.1, che viene usato per trasferimenti di dati per o da un gruppo di moduli che non occupino necessariamente stazioni successive o tutti i sottoindirizzi.
Viene adoperato lo stato della Q,durante ogni operazione, per determinare il numero della stazione ed il sottoindirizzo per la prossima operazione. Quando Q=1, il sottoindirizzo viene incrementato, con eventuale riporto nel numero della stazione. Quando Q=0 si pone il sottoindirizzo ad A(0), e viene incrementato il numero della stazione. Questo permette di lasciare stazioni vuote all'interno di un gruppo di moduli. La MAD può essere usata solo con device di tipo Q-scan set e con funzioni read, write e operate, ma non funzioni read status. per ciò che concerne le funzioni operate potrebbero sorgere complicazioni con i moduli, dato che in (2) l'algoritmo ADDRESS SCAN MODE è stato definito solo per l'uso con funzioni che trasferiscono dati.
Nota: comunque finisca l'azione, l'indirizzo del registro su cui è avvenuta l'ultima azione, viene scritto nella varabile tally del canale usato.

5. STATEMENTS DI AZIONE DI LAM

Tramite questi statements si possono gestire, abilitare, disabilitare, azzerare ed esaminare le LAM.
Nelle pagine seguenti verranno usati i termini LAM-A e LAM-I e verrà fatto riferimento ai registri all'interno dei moduli che contengono informazioni per le LAM. Per ciò che riguarda LAM-A e LAM-I, si riporta qui la descrizione che ne viene fatta in (1) (3.9.4-3.9.5).

LAM-A

LAM-A è una LAM cui si accede a livello di sottoindirizzo. Ha come componenti di indirizzo C,N ed A.

LAM-I

LAM-I è una LAM cui si accede tramite bit-position in registri del gruppo 2 (vedi (2) 5.4.1.2).Ha come componenti di indirizzo C,N ed I.

Notare che per accedere ad un indirizzo LAM-I il sottoindirizzo usato in una azione all'atto del run dipende dalla azione che si esegue.
Notare anche che la bit-position non fa partedella struttura dell'indirizzo a livello hardware; l'accesso al bit avviene tramite i codici F 18,19,21,23 usando una parola CAMAC con il bit appropriato impostato ad uno e tutti i rimanenti bit a zero (2) 6.3.
Per i registri si ricorda che i moduli possono contenere registri per le informazioni delle LAM.Questi registri non sono obbligatori ma, se inclusi, vi si deve poter accedere come registri del gruppo 2 ai seguenti sottoindirizzi:

LAM STATUS REGISTER	A(12)
LAM MASK REGISTER	A(13)
LAM REQUEST REGISTER	A(14)

Le bit-positions corrispondenti dovrebbero essere associate con la stessa sorgente di LAM.

Una volta che l'utente ha dichiarato opportunamente una LAM, non occorre differenziare le azioni tra LAM con accesso al sottoindirizzo e LAM con accesso alla bit-position.

```
                          ENL     <l-name>
                          DISL    <l-name>
                          CLRL    <l-name>
                          IFL     <l-name>,<label-name>
                          IFNL    <l-name>,<label-name>
<lam-action>::=           IFS     <l-name>,<label-name>
                          IFNS    <l-name>,<label-name>
                          RLS     <cn>,<mem-1>
                          RLR     <cn>,<mem-1>
                          RLM     <cn>,<mem-1>
                          WLM     <cn>,<mem-2>

<mem-1>::=<v-name>/<b-name>
<mem-2>::=<k-name>/<v-name>/<b-name>
```

5.1 ENL - ENABLE LAM

La macro ENL abilita la LAM specificata dall'identificatore <l-name>, che l'utente deve aver dichiarato precedentemente in uno statement dichiarativo di LAM. A seconda del tipo di LAM verrà eseguita o F26 (ENABLE) all'indirizzo LAM-A, o F19 (SELECTIVE SET GROUP 2 REGISTER) in A (13) all'indirizzo LAM-I.

5.2 DISL - DISABLE LAM

La macro DISL disabilita la LAM specificata. A seconda del tipo di LAM verrà eseguita o F24 (DISABLE) all'indirizzo LAM-A, oppure F23 (SELECTIVE CLEAR GROUP 2 REGISTER) in A(13) allo indirizzo LAM-I.

5.3 CLRL - CLEAR LAM

La macro CLRL azzera la LAM specificata. A seconda del tipo di LAM verrà eseguita o F10 (CLEAR LOOK-AT-ME) all'indirizzo LAM-A, oppure F23 (SELECTIVE CLEAR GROUP 2 REGISTER) in A(12) all'indirizzo LAM-I.

5.4 IFL - JUMP IF LAM

La macro IFL esegue un salto alla label indicata se la LAM specificata è impostata ad uno. A seconda del tipo di LAM verrà eseguita F08 (TEST LOOK-AT-ME) all'indirizzo LAM-A, oppure verrà controllata la bit-position nel registro A(14) all'indirizzo LAM-I

5.5 IFNL - JUMP IF NOT LAM

La macro IFNL è la negata della IFL, esegue cioè il salto alla label specificata se la LAM dichiarata non è impostata ad uno.

5.6 IFS - JUMP IF STATUS

La macro IFS esegue un salto alla label indicata se lo stato della LAM specificata è ad uno. In dipendenza del tipo di LAM verrà eseguita F27 (TEST STATUS) all'indirizzo LAM-A, oppure verrà controllata la bit-position nel registro A(12) allo indirizzo LAM-I.

5.7 IFNS - JUMP IF NOT STATUS

La macro IFNS è la negata della IFS, salta cioè alla label specificata se lo stato della LAM dichiarata è zero.

5.8 RLS - READ LAM STATUS register

La macro RLS trasferisce al riferimento interno dato la parola CAMAC letta usando F01 in A(12) all'indirizzo del modulo.

5.9 RLR - READ LAM REQUEST register

La macro RLR trasferisce al riferimento interno dato la parola CAMAC letta usando F01 in A(14) all'indirizzo del modulo.

5.10 RLM - READ LAM MASK register

La macro RLM trasferisce al riferimento interno dato la parola CAMAC letta usando F01 in A(13) all'indirizzo del modulo.

5.11 WLM - WRITE LAM MASK register

La macro WLM scrive la parola CAMAC dal riferimento interno dato, in A(13) all'indirizzo del modulo.

6. STATEMENTS DI AZIONE DI SISTEMA

Crate controller actions

Tramite questi statements si possono specificare comandi o controlli da eseguire sul crate controller.
Lo scopo di questi statements è di rendere operative tutte le azioni eseguibili dai registri dell'interfaccia, di poter leggere e scrivere i registri e di trasferire il controllo del programma ad una istruzione specificata da una label, al verificarsi di certe condizioni nel crate.
Gli statements che richiedono la modifica del contenuto di un registro, lo fanno a livello di posizione di bit, senza modificare i rimanenti bit, ad eccezione naturalmente, dei due statements aggiuntivi: WCSR (Write CSR) e WCGL (Write Crate Graded LAM), che scrivono sui registri la parola specificata dall'utente.
Si è potuta notare la convenienza di poter disporre di una istru-

zione che scriva più bit alla volta su un dato registro del crate controller, e siccome la definizione semantica del linguaggio IML (1) non prevedeva, fra gli statements di azione sul crate controller una tale possibilità, sono stati creati due nuovi statements (WCSR, WCGL) per poter scrivere i due registri CSR e DMR; per completezza è stato introdotto anche lo statement RCSR per poter leggere il registro CSR (lo statement RCGL, per la lettura del registro DMR, era previsto).

<crate-action>::=

CZ	<c>
CC	<c>
SETCI	<c>
CLRCI	<c>
ENCD	<c>
ENCX	<c>
DISCD	<c>
DISCX	<c>
IFCI	<c>,<label-name>
IFNCI	<c>,<label-name>
IFCD	<c>,<label-name>
IFNCD	<c>,<label-name>
IFGL	<c>,<label-name>
IFNGL	<c>,<label-name>
RCSR	<c>
WCSR	<c>
RCGL	<c>
WCGL	<c>

6.1 CZ - INITIALISE CRATE

La macro CZ fa partire un ciclo CAMAC automatico di inizializzazione, generando il segnale Z (Initialise), sul crate specificato (viene impostato il bit Z, nove, del CSR).

6.2 CC - CLEAR CRATE

La macro CC fa partire un ciclo CAMAC automatico di azzeramento

generando il segnale C (Clear), sul crate spacificato (viene impostato il bit C, otto, del CSR).

6.3 SETCI - SET the "INHIBIT bus" in the crate

La macro SETCI genera lo stato uno sulla linea-inhibit (I) del crate specificato (viene attivato il bit I-FF, cinque, del CSR).

6.4 CLRCI - CLEAR the "INHIBIT bus" in the crate

La macro CLRCI è la negata della SETCI; la linea-inhibit (I) del crate specificato viene posta nello stato zero (viene azzerato il bit I-FF, cinque, del CSR).

6.5 ENCD - ENABLE crate DEMAND

La macro ENCD abilita il sistema di interruzioni nel crate specificato(viene posto ad uno il bit D-ENB, sei, del CSR).

6.6 DISCD - DISABLE crate DEMAND

La macro DISCD è la negata della ENCD; disabilita il sistema di interruzioni del crate specificato (viene azzerato il bit D-ENB, sei, del CSR).

6.7 ENCX - ENABLE Crate X-error detection

La macro ENCX abilita la generazione di un'interruzione sul crate specificato, con vettore dipendente dal cablaggio interno del crate controller, quando la risposta X del dataway è zero e se D-ENB è ad uno (viene posto ad uno il bit X-ENB, dieci, del CSR). In (1): 6.4.2 e A9.2 questa azione era prevista solo per il system controller, ma nel caso dell'interfaccia J CC 11, non esistendo system controller, è stata realizzata per il crate controller.

6.8 DISCX - DISABLE Crate X-error detection

La macro DISCX è la negata della ENCX e disabilita la generazione dell'interruzione X (viene azzerato il bit X-ENB, dieci, del CSR).

6.9 IFCI - JUMP if crate INHIBIT SET

La macro IFCI esegue un salto alla label specificata se lo INHIBIT bus, nel crate dato, è impostato ad uno (viene controllato il bit I, dodici, del CSR).
In questa macro, come nelle seguenti, compare come parametro, oltre al numero del crate anche la label a cui si vuole che passi il controllo del programma nel caso sia soddisfatta la richiesta della macro.

6.10 IFNCI - JUMP if crate INHIBIT NOT SET

La macro IFNCI è la negata della IFCI, esegue cioè il salto alla label se l'Inihibit bus nel crate specificato non è posto ad uno (viene controllato il bit I, dodici, del CSR).

6.11 IFCD - JUMP if crate DEMAND ENABLED

La macro IFCD esegue un salto alla label specificata se il sistema di interruzione nel crate specificato è abilitato (viene controllato il bit D-ENB, sei, del CSR).

6.12 IFNCD - JUMP if crate DEMAND NOT ENABLED

La macro IFNCD è la negata della IFCD ed esegue il salto alla label specificata se nel crate stabilito il sistema di interruzione non è abilitato (viene controllato il bit D-ENB, sei, del CSR).

6.13 IFGL - JUMP if crate DEMAND

La macro IFGL esegue un salto alla label specificata se una qualsiasi LAM abilitata nel crate dato è posta ad uno (viene controllato il bit D, sette, del CSR).

6.14 IFNGL - JUMP if NOT crate DEMAND

La macro IFNGL è la negata della IFGL ed esegue il salto alla label specificata se nessuna delle LAM abilitate nel crate dato è posta ad uno (viene controllato il bit D,sette, del CSR).

6.15 RCSR - READ CONTROL and STATUS REGISTER

La macro RCSR trasferisce nella variabile<v-name> , il contenuto del registro CSR del crate controller specificato.

6.16 WCSR - WRITE CONTROL and STATUS REGISTER

La macro WCSR scrive nel registro CSR del crate controller specificato, la parola stabilita tramite <k-name> o <v-name>.

6.17 RCGL - READ CRATE GRADED LAM

La macro RCGL trasferisce nella variabile <v-name> il contenuto del registro DMR del crate controller specificato. Per consentire all'utente un controllo completo sullo stato del DMR, viene trasferita in <v-name> , sia l'informazione relativa ai bit di demand D_i (high-byte), sia quella relativa ai bit di mask M_i (low-byte). L'utente che voglia controllare solo una delle due informazioni può farlo tramite opportune istruzioni a livello di byte.

6.18 WCGL - WRITE CRATE GRADED LAM

La macro WCGL scrive nel registro DMR del crate controller desiderato, la parola specificata tramite <k-name> o <v-name>. Naturalmente, data la struttura del registro DMR, verranno interessati alla scrittura, solo i bit di mask M_i (low-byte).

7. MACRO AUSILIARIE

Si è resa necessaria la creazione di due macro ausiliarie da aggiungere a quelle descritte nella definizione dell'IML (1); si tratta delle macro .CAMAC e .CLINT. La prima è stata realizzata per motivi tecnici, la seconda per fornire un'ulteriore facilitazione all'utente.

7.1 .CAMAC

Formato:

```
.CAMAC
```

Funzioni svolte e modalità d'uso

In ogni programma IML, la prima macro IML che compare deve essere obbligatoriamente la macro ausiliaria .CAMAC. La .CAMAC non ha argomenti; tramite la sua espansione vengono resi noti all'assemblatore tutti i simboli usati dall'IML. Tali simboli sono privilegiati, e non devono quindi essere usati dall'utente per altri scopi. Essi sono costituiti dai simboli usati per il calcolo degli indirizzi del crate, da quelli del metalinguaggio <f-read>,<f-write>,<f-operate>, da quelli usati nelle dichiarative e dai due simboli di uso generale ERCONT ed EXPON2. Il loro elenco è dato in appendice A.

7.2 .CLINT Caricamento vettori Interrupt service routine

Formato:

```
.CLINT   <label-name>,<base-address>,<demand-line>,
         <psw>
```

<base-address>::= indirizzo di base
<demand-line> ::= linea di domanda D_i
<psw> ::= Processor Status Word, in ottale, che si vuole dare alla routine di servizio dell'interrupt.

Esempio:

```
        .CLINT    RSLADC,274,3,340
                  ...
RSLADC: "routine di servizio interrupt"
```

Funzioni svolte e modalità d'uso

La macro ausiliaria .CLINT ha il compito di legare e gestire tutti i parametri necessari ad un correttto caricamento del vettore d'interruzione corrispondente ad una data LAM, con l'indirizzo della routine di servizio dell'utente (label=name) e con la nuova processor status word (psw). Per la particolare struttura del crate controller J CC 11 (vedi appendice E) ad una data LAM del crate viene fatto corrispondere un vettore di interruzione il cui indirizzo dipende dalla disposizione di alcuni jumper all'interno del modulo J CC 11. L'utente dovrà quindi specificare la configurazione dei jumper interni, cioè del base-address e della linea D_i corrispondente alla LAM attuale, del suo crate controller.

- Il meta-linguaggio sintattico

Nelle definizioni sintattiche si è fatto ricorso al seguente meta-linguaggio.
Il meta-linguaggio contiene alcuni simboli come / ::= < (etc , che non compaiono nell'IML. Le parole contenute fra parentesi angolari, es: <digit>, fanno anch'esse parte del meta-linguaggio e non dell'IML. Le parentesi angolari sono invece elementi sintattici obbligatori quando si debbano specificare indirizzi CAMAC.
Segue ora una lista dei simboli del meta-linguaggio e del loro significato. Qualunque altro simbolo, non racchiuso da parentesi angolari, è autonomo e fa parte dell'IML.

::=	Significa "è definito da". Separa la parte sinistra da quella destra di una definizione.
'	Apre una stringa di caratteri che costituisce un simbolo del meta-linguaggio.
'	Termina una stringa di caratteri che costituisce un simbolo del meta-linguaggio.
/	Separa unità sintattiche alternative nella parte destra di una definizione.
[	Apre una opzione, cioé le unità sintattiche racchiuse fra parentesi quadre sono presenti come opzioni.
]	Termina una opzione.
(	Apre un gruppo di elementi che devono essere considerati come una singola unità sintattica per gli scopi della definizione.
)	Chiude un gruppo di elementi che devono essere considerati come una singola unità sintattica.
...	Significa che l'unità sintattica precedente può essere ripetuta nessuna o più volte.

- Definizione degli elementi sintattici

<a-name>	::=	<identifier>	Nome di un CAMAC-word array e suo macro-name.
<array-lenght>	::=	<constant>	Numero di elementi in un CAMAC-word array.
<b-name>	::=	<identifier>	Nome del blocco di controllo del buffer e suo macro-name.
<buffer>	::=	<identifier>	Nome del buffer.
<buffer-lenght>	::=	<constant>	Numero di parole-CAMAC che un buffer può contenere.
<c>	::=	<constant>,	Numero del crate (1÷10).
<cn>	::=	<constant>, <constant>,	Numeri del crate e della stazione (1÷10), (1÷23).
<cna>	::=	<constant>, <constant>, <constant>,	Numeri del crate, della stazione e del sottoindirizzo (1÷10), (1÷23), (0÷15).
<cna-f >	::=	<cna>	Primo valore di CNA (first).
<cna-l >	::=	<cna>	Ultimo valore di CNA (last).
<cna-i >	::=	<cna>	Valore di incremento per CNA.
<#cna >	::=	<constant >	Numero di elementi in un Given o Calculated address array.
<bit-posn>	::=	<constant>	Numero del bit in una parola-CAMAC (bit-position, 1÷16).
<c-name>	::=	<identifier>	Nome di un Calculated-address array e suo macro-name.
<ch-name>	::=	<identifier>	Nome del blocco di parametri del canale e suo macro-name.
<constant>	::=	<literal>/<identifier>	Costante intera.
<digit>	::=	0/1/2/3/4/5/6/7/8/9	
<f-read>	::=	(F00/RD1)/(F01/RD2) /(F02/RC1)/(F03/RCM) /F04/F05/F06/F07	Codici F-READ.
<f-write>	::=	(F16/WT1)/(F17/WT2) /(F18/SS1)/(F19/SS2) /F20/(F21/SC1) /F22/(F23/SC2)	Codici F-WRITE.

<f-operate >	::=	(F08/TLM)/(F09/CL1) /(F10/CLM)/(F11/CL2) /F12/F13/F14/F15 /(F24/DIS)/(F25/XEQ) /(F26/ENB)/(F27/TST)	Codici F-OPERATE.
<g-name>	::=	<identifier >	Nome di un Given-address array e suo macro-name.
<h-name>	::=	<identifier >	Nome di un indirizzo CAMAC costante (H) e suo macro-name.
<identifier >	::=	<letter>[(<letter>/<digit>)...]	
<k-name>	::=	<identifier >	Nome di una parola CAMAC costante o costante intera e suo macro-name.
<l-name>	::=	<identifier >	Nome del blocco di parametri LAM e suo macro-name.
<label>	::=	<label-name><label-terminator>	
<label-name >	::=	<identifier >	
<label-terminator>	::=	:	
<letter>	::=	A/B/C/D/E/F/G/H/I/J/K/L/M /N/O/P/Q/R/S/T/U/V/W/X/Y/Z	
<literal>	::=	<digit>...	Costante numerica.
<p-name>	::=	<identifier >	Nome di un indirizzo CAMAC Variabile e suo macro-name.
<q-name>	::=	<identifier >	Nome di un Q-scan set e suo macro-name.
<repeat>	::=	<k-name>/<v-name >	Parametro di repeat count del canale.
<st-delimiter>	::=	[<comment>]end of line >	
<status>	::=	<identifier >	Nome della variabile status.
<tally>	::=	<v-name>	Variabile tally nei parametri del canale.
<v-name>	::=	<identifier >	Nome di una Variabile intera o di una parola-CAMAC Variabile e suo macro-name.
<vl-name>	::=	<identifier >	Nome di una Variabile logica e suo macro-name.
<word-longht>	::=	<constant >	

TABELLA DEI SIMBOLI RISERVATI

CR1	F00	RD1	H
CR2	F01	RD2	P
CR3	F02	RC1	G
CR4	F03	RCM	C
CR5	F04	TLM	Q
CR6	F05	CL1	K
CR7	F06	CLM	V
CR8	F07	CL2	VL
CR9	F08	WT1	A
CR10	F09	WT2	B
	F10	SS1	SUB
	F11	SS2	BIT
	F12	SC1	ERCONT
	F13	SC2	EXPON2
	F14	DIS	
	F15	XEQ	
	F16	ENB	
	F17	TST	
	F18		
	F19		
	F20		
	F21		
	F22		
	F23		
	F24		
	F25		
	F26		
	F27		
	F28		
	F29		
	F30		
	F31		

APPENDICE B

<declaration>::=

<local-declaration>
/<export-declaration>::=

LOCD/EXPD (<h-name>,H,<cna>)
/(<p-name>,P[,<cna>])
/(<g-name>,G,<#-cna>/CONT,<cna>...)
/(<c-name>,C,<#-cna>,<cna-f>,<cna-l>,<cna-i>)
/(<q-name>,Q,<cna-f>,<cna-l>)

LOCM/EXPM (<k-name>,K,<word-lenght>,<literal>)
/(<v-name>,V,<word-lenght>)
/(<vl-name>,VL)
/(<a-name>,A,<word-lenght>,<array-lenght>)
/(<b-name>,B,<buffer>,<word-lenght>,<buffer-lenght>[,<label-name>])

LOCC/EXPC <ch-name>,<repeat>,<tally>,<status>

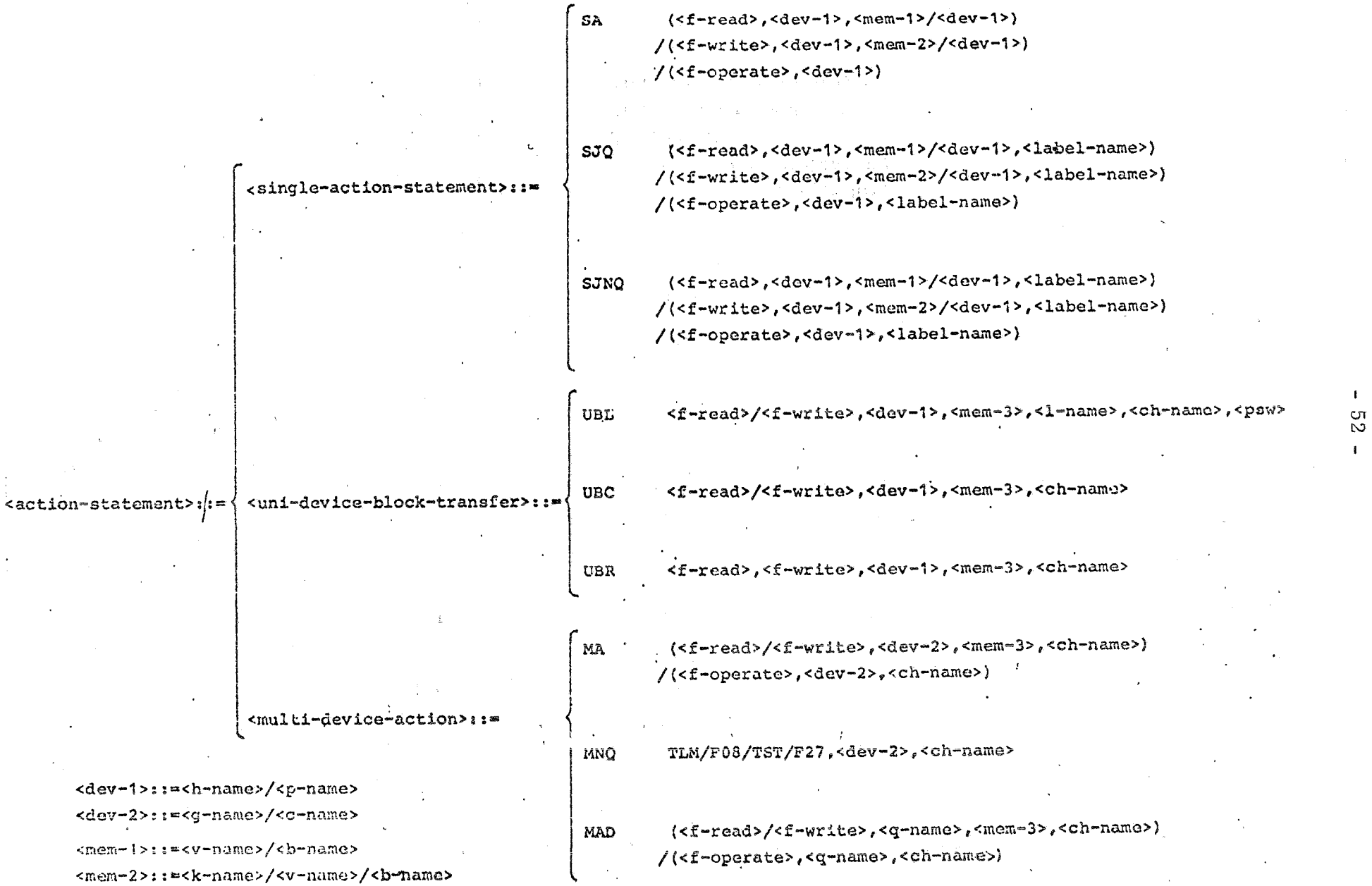
LOCL/EXPL/ (<l-name>,SUB,<cna>[,<base-address>,<demand-line>])
IMPL /(<l-name>,BIT,<bit-posn>,<cn>[,<base-address>,<demand-line>])

<import-declaration>::=

IMPD (<h-name>,H,<c>)
/(<p-name>,P)
/(<g-name>,G)
/(<c-name>,C)
/(<q-name>,Q)

IMPM (<k-name>,K,<word-lenght>)
/(<v-name>,V,<word-lenght>)
/(<vl-name>,VL)
/(<a-name>,A,<word-lenght>)
/(<b-name>,B,<word-lenght>)

IMPC <ch-name>


<action-statement>::=
<single-action-statement>::=
SA (<f-read>,<dev-1>,<mem-1>/<dev-1>)
/(<f-write>,<dev-1>,<mem-2>/<dev-1>)
/(<f-operate>,<dev-1>)
SJQ (<f-read>,<dev-1>,<mem-1>/<dev-1>,<label-name>)
/(<f-write>,<dev-1>,<mem-2>/<dev-1>,<label-name>)
/(<f-operate>,<dev-1>,<label-name>)
SJNQ (<f-read>,<dev-1>,<mem-1>/<dev-1>,<label-name>)
/(<f-write>,<dev-1>,<mem-2>/<dev-1>,<label-name>)
/(<f-operate>,<dev-1>,<label-name>)
<uni-device-block-transfer>::=
UBL <f-read>/<f-write>,<dev-1>,<mem-3>,<l-name>,<ch-name>,<psw>
UBC <f-read>/<f-write>,<dev-1>,<mem-3>,<ch-name>
UBR <f-read>,<f-write>,<dev-1>,<mem-3>,<ch-name>
<multi-device-action>::=
MA (<f-read>/<f-write>,<dev-2>,<mem-3>,<ch-name>)
/(<f-operate>,<dev-2>,<ch-name>)
MNQ TLM/F08/TST/F27,<dev-2>,<ch-name>
MAD (<f-read>/<f-write>,<q-name>,<mem-3>,<ch-name>)
/(<f-operate>,<q-name>,<ch-name>)
<dev-1>::=<h-name>/<p-name>
<dev-2>::=<g-name>/<c-name>
<mem-1>::=<v-name>/<b-name>
<mem-2>::=<k-name>/<v-name>/<b-name>
<mem-3>::=<a-name>/<b-name>

<crate-action>::=

CZ	<c>
CC	<c>
SETCI	<c>
CLRCI	<c>
ENCD	<c>
ENCX	<c>
DISCD	<c>
DISCX	<c>
IFCI	<c>,<label-name>
IFNCI	<c>,<label-name>
IFCD	<c>,<label-name>
IFNCD	<c>,<label-name>
IFGL	<c>,<label-name>
IFNGL	<c>,<label-name>
RCSR	<c>
WCSR	<c>
RCGL	<c>
WCGL	<c>

<lam-action>::=

ENL	<l-name>
DISL	<l-name>
CLRL	<l-name>
IFL	<l-name>,<label-name>
IFNL	<l-name>,<label-name>
IFS	<l-name>,<label-name>
IFNS	<l-name>,<label-name>
RLS	<cn>,<mem-1>
RLR	<cn>,<mem-1>
RLM	<cn>,<mem-1>
WLM	<cn>,<mem-2>

<mem-1>::=<v-name>/<b-name>

<mem-2>::=<k-name>/<v-name>/<b-name>

APPENDICE C

Sommario delle macro IML in ordine alfabetico

Macro	Significato
CC	Clear Crate
CLRCI	CLEAR the "INHIBIT bus" in the crate
CLRL	CLEAR LAM
CZ	INITIALISE Crate
DISCD	DISABLE crate DEMAND
DISCX	DISABLE crate X
DISL	DISABLE LAM
ENCD	ENABLE crate DEMAND
ENCX	ENABLE crate X
ENL	ENABLE LAM
EXPC	DICHIARAZIONE EXPORT di CANALE
EXPD	DICHIARAZIONE EXPORT di DEVICE
EXPL	DICHIARAZIONE EXPORT di LAM
EXPM	DICHIARAZIONE EXPORT di MEMORIA
IFCD	JUMP if crate DEMAND ENABLED
IFCI	JUMP if crate INHIBIT SET
IFGL	JUMP if crate DEMAND
IFL	JUMP if LAM
IFNCD	JUMP if crate DEMAND NOT ENABLED
IFNCI	JUMP if crate INHIBIT NOT SET
IFNGL	JUMP if NOT crate DEMAND
IFNL	JUMP if NOT LAM
IFS	JUMP if LAM STATUS
IFNS	JUMP if NOT LAM STATUS

Macro	Significato
IMPC	DICHIARAZIONI IMPORT di CANALE
IMPD	DICHIARAZIONI IMPORT di DEVICE
IMPL	DICHIARAZIONE IMPORT di LAM
IMPM	DICHIARAZIONI IMPORT di MEMORIA
LOCC	DICHIARAZIONI LOCALI di CANALE
LOCD	DICHIARAZIONI LOCALI di DEVICE
LOCL	DICHIARAZIONI LOCALI di LAM
LOCM	DICHIARAZIONI LOCALI di MEMORIA
MA	MULTI-DEVICE ACTION
MAD	MULTIPLE-ADDRESS SCAN
MNQ	MULTI-DEVICE ACTION, REPEAT WHILE NOT Q
RCGL	READ CRATE GRADED LAM
RCSR	READ CONTROL and STATUS REGISTER
RLM	READ LAM MASK register
RLR	READ LAM REQUEST register
RLS	READ LAM STATUS register
SA	SINGLE ACTION
SETCI	SET the "INHIBIT bus" in the crate
SJNQ	SINGLE ACTION JUMPING on Q=0
SJQ	SINGLE ACTION JUMPING on Q=1
UBC	UNI-DEVICE BLOCK TRANSFERS, CONTROLLER-SYNCHRONISED
UBR	UNI-DEVICE BLOCK TRANSFERS, REPEAT WHILE NOT Q
UBL	UNI-DEVICE BLOCK TRANSFERS, LAM-SYNCHRONISED
WCGL	WRITE CRATE GRADED LAM
WCSR	WRITE CONTROL and STATUS REGISTER
WLM	WRITE LAM MASK register

Macro ausiliarie

.CAMAC	Definizione dei simboli
.CLINT	Caricamento vettori Interrupt service routine

APPENDICE D

La forma generale dei messaggi di errore, per le configurazioni che fanno uso di terminali con lunghezza di riga di 132 caratteri, è la seguente:

```
P  164  002470  000012  .ERROR  EXP  ; >>>>> "nome":"testo".
↑   ↑     ↑       ↑       ↑     ↑   ↑   ↑      ↑    ↑   ↑
1)  2)    3)      4)      5)    6)  7)  8)     9)  10)  11)
```

1) Codice di errore assembler associato dal MACRO-11 a tutte le direttive .ERROR.
2) Numero sequenziale dello statement in cui compare l'errore.
3) Valore attuale del location counter.
4) Valore dell'espressione 6), se specificata.
5) Direttiva .ERROR.
6) Espressione il cui valore verrà scritto in 4).
7) Separatore del campo commenti.
8) Caratteri per evidenziare l'errore.
9) Argomento errato.
10) Separatore.
11) Messaggio diagnostico, il più possibile autoconsistente, che spiega perchè l'argomento in questione è da considerarsi errato.

Nota: In alcuni casi, il messaggio di errore IML può essere accompagnato dai codici di errore del MACRO-11, che comunque, dopo la correzione dell'errore IML, scompariranno.

SOMMARIO DEI MESSAGGI DI ERRORE

Codice del messaggio	Informazioni supplementari / Significato
"A-first">"A-last" : A-LAST MUST BE GREATHER THAN OR EQUAL TO A-FIRST.	In una dichiarazione di device del tipo Calculated-address array, l'argomento <cna-l> deve essere maggiore o uguale all'argomento <cna-f>, altrimenti il calcolo non viene eseguito.
"C-first">"C-last" : C-LAST MUST BE GREATHER THAN OR EQUAL TO C-FIRST.	In una dichiarazione di device del tipo Calculated-address array, l'argomento <cna-l> deve essere maggiore o uguale all'argomento <cna-f>, altrimenti il calcolo non viene eseguito.
"F-CAMAC" : F-OPERATE IS ILLEGAL IN UNI-DEVICE-BLOCK TRANSFER.	Gli Uni-device Block transfers accettano solo funzioni CAMAC del tipo read o write, ma non operate.
"Symbol" : ILLEGAL EXTERNAL REF. IN DEVICE -DECLARATION.	Nelle LOCD e EXPD, il simbolo che differenzia il tipo di device che si sta dichiarando può essere solo: H,P,G,C,Q.

"Symbol" : ILLEGAL EXTERNAL-REFERENCE IN IMPD.

Nella IMPD, il simbolo che differenzia il tipo di device che si sta dichiarando può essere solo: H,P,G,C,Q.

- - - - -

"F-CAMAC" : ILLEGAL FUNCTION CODE.

L'argomento f-CAMAC deve essere conforme alla sintassi stabilita in appendice A.

- - - - -

"Device"-"Memory" : ILLEGAL INTER-CAMAC TRANSFER.

In una SA usata per trasferimenti tra registri CAMAC, dichiarati come H, tali registri si devono trovare sullo stesso crate.

- - - - -

"Symbol" : ILLEGAL INTERNAL-REFERENCE IN IMPM.

Nella IMPM, il simbolo che differenzia il tipo di memoria che si sta dichiarando può essere solo: K,V,VL,A,B.

- - - - -

"Symbol" : ILLEGAL INTERNAL-REFERENCE IN "macro-name"-MEMORY-DECLARATION.

Nelle LOCM ed EXPM, il simbolo che differenzia il tipo di memoria che si sta dichiarando può essere solo: K,V,VL,A,B.

- - - - -

"Word-lenght" : ILLEGAL OR MISSING WORD-LENGHT IN IMPM.

Nella IMPM, l'argomento word-lenght deve essere 16 o 24.

- - - - -

"Repeat" : ILLEGAL REPEAT MODE IN CHANNEL "ch-name" DECLARATION.

Nelle LOCC ed EXPC, l'argomento repeat deve essere stato dichiarato in precedenza come <k-name> o <v-name>.

- - - - -

"Tally" : ILLEGAL TALLY ARGUMENT IN CHANNEL "ch-name" DECLARATION.

Nelle LOCC ed EXPC, l'argomento tally deve essere stato dichiarato in precedenza come <v-name>.

- - - - -

"Word-lenght" : ILLEGAL WORD-LENGHT IN "macro-name"-MEMORY-DECLARATION.

Nelle LOCM ed EXPM, l'argomento <word-lenght> deve essere 16 o 24.

- - - - -

"Symbol" : IN LAM DECLARATION, MODE MUST BE SUB OR BIT.

Nelle LOCL, EXPL ed IMPL, il secondo argomento deve essere uno dei simboli: SUB o BIT.

- - - - -

"Number" : INVALID BIT-POSITION ARGUMENT.

La bit-position deve essere un numero compreso tra 1 e 16, estremi inclusi.

- - - - -

C,N,A : INVALID CRATE NUMBER.

Il valore specificato per C deve essere un numero compreso tra 1 e 10, estremi inclusi

- - - - -

"C" : INVALID CRATE NUMBER IN IMPD.	Nella IMPD di tipo H, il terzo argomentodeve rappresentare il numero del crate.
- - - - -	
"C" : INVALID OR MISSING CRATE NUMBER IN "macro-name".	Questo messaggio è comune a tutti gli statements di azione di sistema (CI, CC, SETCI, etc.). Per differenziare i vari casi, al posto di "macro-name" viene scritto il nome della macro fonte di errore.
- - - - -	
<C,N,A > : INVALID STATION NUMBER.	Il valore specificato per N deve essere un numero compreso tra 1 e 23, estremi inclusi.
- - - - -	
<C,N,A > : INVALID SUB-ADDRESS NUMBER.	Il valore specificato per A deve essere un numero compreso tra 0 e 15, estremi inclusi.
- - - - -	
MISSING ARRAY-LENGHT IN "macro-name"-MEMORY-DECLARATION.	Nelle LOCM ed EXPM di tipo Array, il quarto argomento deve rappresentare la array-lenght.
- - - - -	
MISSING BASE-ADDRESS NUMBER IN .CLINT.	Nella .CLINT, il secondo argomento deve rappresentare il numero del Base-address.
- - - - -	
MISSING BIT-POSITION ARGUMENT.	Quando richiesto, l'argomento sulla bit-position deve essere presente.
- - - - -	
MISSING BUFFER IN "name"-MEMORY DECLARATION.	Nelle LOCM ed EXPM di tipo buffer, il terzo argomento deve rappresentare il nome del buffer.
- - - - -	
MISSING BUFFER-LENGHT IN "name"-MEMORY-DECLARATION.	Nelle LOCM ed EXPM di tipo buffer, il quinto argomento deve rappresentare la buffer-lenght.
- - - - -	
MISSING CHANNEL REFERENCE.	L'azione in questione richiede tra gli argomenti anche il nome del canale.
- - - - -	
MISSING CH-NAME IN CHANNEL-DECLARATION.	Nelle LOCC ed EXPC, il primo argomento deve rappresentare il nome del canale.
- - - - -	
MISSING CH-NAME IN IMPC	Nella IMPC, il primo argomento deve rappresentare il nome del canale.
- - - - -	
MISSING DEMAND-LINE NUMBER IN .CLINT.	Nella .CLINT, il terzo argomento deve rappresentare il numero della Demand-line.
- - - - -	

MISSING DEVICE REFERENCE. — L'azione in questione richiede tra gli argomenti anche il nome del device.

- - - - -

MISSING EXTERNAL-REF. IN DEVICE-DECLARATION. — In una LOCD o EXPD il secondo argomento deve essere uno dei simboli: H,P,G,C,Q.

- - - - -

MISSING FUNCTION CODE. — L'azione in questione richiede tra gli argomenti anche la funzione-CAMAC.

- - - - -

MISSING INTERNAL-REFERENCE IN "name"-MEMORY DECLARATION. — Nelle LOCM ed EXPM, il secondo argomento deve essere uno dei simboli: K,V,VL,A,B.

- - - - -

MISSING INTERRUPT SERVICE ROUTINE ADDRESS IN.CLINT. — Nella .CLINT, il primo argomento deve rappresentare l'indirizzo della routine di servizio dell'interrupt.

- - - - -

MISSING LABEL NAME. — Nelle SJQ e SJNQ, l'argomento label-name è obbligatorio.

- - - - -

MISSING LABEL NAME IN "macro-name". — Questo messaggio è comune agli statements di azione di LAM e a quelli di azione di sistema. Per differenziare i vari casi, al posto di "macro-name" viene scritto il nome della macro fonte di errore.

- - - - -

MISSING LAM A-I SPECIFIER. — Nelle LOCL,EXPL ed IMPL, il secondo argomento deve essere uno dei dimboli SUB o BIT.

- - - - -

MISSING L-NAME IN "macro-name". — Questo messaggio è comune agli statements di azione di LAM: ENL, DISL, CLRL, etc. Per differenziare i vari casi, al posto di "macro-name" viene scritto il nome della macro fonte di errore.

- - - - -

MISSING L-NAME IN LAM DECLARATION. — Nelle LOCL,EXPL ed IMPL, il primo argomento deve rappresentare il nome della LAM.

- - - - -

MISSING L-NAME IN UBL. — Nella UBL, il quarto argomento deve essere l-name.

- - - - -

MISSING MEMORY REFERENCE — L'azione in questione richiede tra gli argomenti anche il nome della memoria.

- - - - -

MISSING MEMORY REF. IN "macro-name". — Questo messaggio è comune agli statements di azione di LAM: RLS, RLR, RLM, RLN. Per differenziare i vari casi, al posto di "macro-name" viene scritto il nome della macro fonte di errore.

- - - - -

MISSING NAME IN DEVICE-DECLARATION.

Nelle LOCD ed EXPD, il primo argomento deve rappresentare il nome del device.

- - - - -

MISSING NAME IN IMPD.

Nella IMPD, il primo argomento deve rappresentare il nome del device.

- - - - -

MISSING NAME IN IMPM.

Nella IMPM, il primo argomento deve rappresentare il nome della memoria.

- - - - -

MISSING NAME IN MEMORY-DECLARATION.

Nelle LOCM ed EXPM, il primo argomento deve rappresentare il nome della memoria.

- - - - -

MISSING NUMBER OF ELEMENTS IN CALC. ADDR. ARRAY.

In una LOCD o EXPD di tipo Calculated-address array, il terzo argomento deve rappresentare il numero di elementi che si stanno dichiarando.

- - - - -

MISSING NUMBER OF ELEMENTS IN GIV. ADDR. ARRAY.

In una LOCD o EXPD di tipo Given-address array, il terzo argomento deve rappresentare il numero di elementi che si stanno dichiarando.

- - - - -

MISSING PROCESSOR STATUS WORD IN UBL.

Nella UBL, il sesto argomento deve rappresentare la Processor Status Word con la quale l'utente vuole che sia servita la LAM.

- - - - -

MISSING PSW IN .CLINT.

Nella .CLINT il quarto argomento deve rappresentare la Processor Status Word da assegnare alla routine di servizio dell'interrupt.

- - - - -

MISSING REPEAT ARGUMENT IN CHANNEL "ch-name" DECLARATION.

Nelle LOCC ed EXPC, il secondo argomento deve rappresentare la costante o la variabile repeat.

- - - - -

MISSING STATUS ARGUMENT IN CHANNEL "ch-name"

Nelle LOCC ed EXPC, il quarto argomento deve rappresentare la variabile status.

- - - - -

MISSING TALLY ARGUMENT IN CHANNEL "ch-name".

Nelle LOCC ed EXPC, il terzo argomento deve rappresentare la variabile tally.

- - - - -

"F-CAMAC" : MNQ ACCEPTS ONLY TLM OR TST FUNCTIONS.

Nella MNQ, le uniche funzioni-CAMAC accettabili sono: TLM/F08 o TST/F27.

- - - - -

"N-first">"N-last" : N-LAST MUST BE GREATHER THAN OR EQUAL TO A FIRST.

In una dichiarazione di device del tipo Calculated-address array, l'argomento <cna-l> deve essere maggiore o uguale all'argomento <cna-f>, altrimenti il calcolo non viene eseguito.

- - - - -

"N.of declared elements" : N.OF CALC. ELEM. DIFFERENT FROM N. OF DECL. ELEM.

In una LOCD o EXPD di tipo Calculated, il numero degli elementi calcolati è risultato essere diverso da quello dichiarato dall'utente. Il valore calcolato viene scritto, in linea con il messaggio di errore, dopo il valore del location counter attuale.

- - - - -

PREVIOUS L-NAME DECLARATION WITHOUT BASE-ADDRESS AND DEMAND LINE.

Nella UBL, il quarto argomento deve essere stato dichiarato in precedenza come l-name, con gli argomenti opzionali base-address e demand-line.

- - - - -

"ch-name" : UNDEFINED CHANNEL REFERENCE.

Il nome del canale specificato nella azione in questione, non è stato dichiarato correttamente in precedenza.

- - - - -

"F-CAMAC" : UNDEFINED FUNCTION CODE.

L'argomento funzione-CAMAC deve essere conforme alla definizione data in appendice A.

- - - - -

"l-name": UNDEFINED l-NAME IN "macro-name".

Questo messaggio è comune agli statements di azione di LAM: ENL, DISL, CLRL, etc. Per differenziare i vari casi, al posto di "macro-name" viene scritto il nome della macro fonte di errore.

- - - - -

"name" : UNDEFINED OR ILLEGAL DEVICE REFERENCE.

Il nome del device specificato nell'azione in questione, non è stato dichiarato correttamente in precedenza oppure non è di tipo accettabile nella azione.

- - - - -

"name" : UNDEFINED OR ILLEGAL L-NAME IN UBL.

Nella UBL, il quarto argomento deve essere stato dichiarato correttamente in precedenza come l-name.

- - - - -

"name" : UNDEFINED OR ILLEGAL MEMORY-REFERENCE.

Il nome della memoria specificata nella azione in questione, non è stato dichiarato correttamente in precedenza oppure non è di tipo accettabile dall'azione.

- - - - -

"name" : UNDEFINED OR ILLEGAL MEMORY REF. IN "macro-name".

Questo messaggio è comune agli statements di azione di LAM: RLS, RLR, RLM, WLM. Per differenziare i vari casi, al posto di "macro-name" viene scritto il nome della macro fonte di errore.

- - - - -

"name" : UNNECESSARY MEMORY REF. WITH F-OPERATE.

Nelle SA, SJQ ed SJNQ con funzioni CAMAC di tipo operate, non bisogna specificare l'argomento di memoria.

```
                                        .TITLE  ACQUISIZIONE ADC IN IML
                                        .MCALL  EXIT$S,QIO$S,WTSE$S,CAMAC ,LOCD,LOCM,LOCL,LOCC,.CLINT
                                        .MCALL  CZ,WCGL,ENCD,ENCX,ENL,DISL,CLRL,SA,MA
000000                                  ..CAMAC
                                        ;
                                        ;
000000                                  .PSECT  ADCIML
                                        ;
                                        ;
000000                                  LOCD    ADC,P,<1,3,0>                           ; INDIRIZZO ADC
000010                                  LOCL    ADCLAM,SUB,<1,3,0>                      ; INDIRIZZO LAM ADC
000010                                  LOCD    ADCADR,C,12,<1,3,0>,<1,3,11>,<1,1,1>    ; INDIRIZZI 12 CANALI ADC
000076                                  LOCM    DATBUF,A,16,12                          ; MEMORIA PER DATI
000134                                  LOCM    PIEBUF,A,16,12                          ; MEMORIA PER PIEDISTALLI
000172                                  LOCM    D38,K,16,204                            ; MASCHERA PER D3,D8 IN DMR
000172                                  LOCM    REP,V,16
000176                                  LOCM    TAL,V,16
000202                                  LOCC    CAN,REP,TAL,STAT                        ; CANALE
000212  000000                  STAT:   .WORD
000214  000000                  LAMFL:  .WORD
                                        ;
                                        ;       INIZIALIZZAZIONE
                                        ;
000216  000002  000104' 000142' TABLE:  .WORD   2,<DATBUF+2>,<PIEBUF+2>
000224  012767  000012  177742  INI::   MOV     #12,REP                                 ; NUMERO RIPETIZIONI=12
000232                                  .CLINT  LAMAD,274,3,140300                      ; CARIC. VETT. INT. PER D3; USER,PRI 6
000246                                  .CLINT  XADC,274,8,140300                       ; CARIC. VETT. INT. PER D8; USER, PRI 6
000262                                  CZ      1                                       ; CAMAC Z
000270                                  WCGL    1,D38                                   ; ABILITAZIONE D3, D8 NEL DMR
                                        ;
                                        ;       ABILITAZIONE
                                        ;
000276  005067  177712          ABILI:  CLR     LAMFL
000302                                  SA      F02,ADC                                 ; CLEAR SCALERS
000322                                  ENL     ADCLAM                                  ; ENABLE LAM
000342                                  ENCD    1                                       ; D-ENABLE
000350  005767  177640                  TST     LAMFL                                   ; ATTESA INTERRUZIONE
000354  001775                         BEQ     .-4
000356  000747                         BR      ABILI
                                        ;
                                        ;       SERVIZIO LAM
                                        ;
000360                          LAMAD:  DISL    ADCLAM                                  ; DISABLE LAM
000400                                  CLRL    ADCLAM                                  ; CLEAR LAM
000420                                  MA      F02,ADCADR,DATBUF,CAN                   ; MISURA DATI
000562                                  SA      F21,ADC                                 ; RESET ALL SCALERS
000602                                  MA      F02,ADCADR,PIEBUF,CAN                   ; MISURA PIEDISTALLI
000744  005267  177244                  INC     LAMFL
000750  012705  000216'                 MOV     #TABLE,R5
000754  004767  000000G                 JSR     PC,OUT2                                 ; ROUTINE STAMPA
000760  000002                         RTI
                                        ;
                                        ;
000762                          XADC:   QIO$S   #IO.WLB,#46,#6,,,,<MESX,#40,,#40>       ; SERVIZIO LAM-X
001034                                  WTSE$S  #6
001046                                  EXIT$S
001054      052     052     052 MESX:   .ASCII  /*** COMANDO NON ACCETTATO . STOP . ***/
```

```
ACQUISIZIONE ADC IN IML MACRO M10    01-SEP-77 15:37  PAGE 1-1


        001057      040      040      103
        001062      117      115      101
        001065      116      104      117
        001070      040      116      117
        001073      116      040      101
        001076      103      103      105
        001101      124      124      101
        001104      124      117      040
        001107      056      040      123
        001112      124      117      120
        001115      040      056      040
        001120      052      052      052
    58          000001                        .END
```

```
ACQUISIZIONE ADC IN IML MACRO M10    01-SEP-77 15:37  PAGE 1-2
SYMBOL TABLE

A      = 000020          CR6    = 154000          F08    = 000010
ABILI    000276R   002   CR7    = 152000          F09    = 000011
ADC    = 000002R   002   CR8    = 150000          F10    = 000012
ADCADR   000014R   002   CR9    = 146000          F11    = 000013
ADCLAM= 164140           DATBUF   000102R   002   F12    = 000014
B      = 000000          D38    = 000204          F13    = 000015
BIT    = 000000          ERCONT= 000000           F14    = 000016
C      = 000123          EXPON2= 000014           F15    = 000017
CAN      000204R   002   F00    = 000000          F16    = 000020
CH     = 000123          F01    = 000001          F17    = 000021
CR1    = 164000          F02    = 000002          F18    = 000022
CR10   = 144000          F03    = 000003          F19    = 000023
CR2    = 166000          F04    = 000004          F20    = 000024
CR3    = 160000          F05    = 000005          F21    = 000025
CR4    = 162000          F06    = 000006          F22    = 000026
CR5    = 156000          F07    = 000007          F23    = 000027

. ABS.   000000    000
         000000    001
ADCIML   001123    002
ERRORS DETECTED:  0

FREE CORE:  0. WORDS
ADCIML,TT!=IML/ML,ADCIML
>
```

```
F24    = 000030          OUT2   = ****** GX
F25    = 000031          P      = 000123
F26    = 000032          PIEBUF   000140R   002
F27    = 000033          Q.     = 000000
F28    = 000034          REP      000174R   002
F29    = 000035          STAT *   000212R   002
F30    = 000036          SUB    = 164000
F31    = 000037          TABLE    000216R   002
G      = 000000          TAL      000200R   002
H      = 000000          V      = 000000
INI      000224RG  002   VL     = 000000
IO.WLB= ****** GX        XADC     000762R   002
K      = 000000          $$$ARG= 000002
LAMAD    000360R   002
LAMFL    000214R   002
MESX     001054R   002
```

Esempio di programma che effettua la gestione e l'acquisizione dati da un ADC (Analogous to Digital Converter), utilizzando le macro IML.

APPENDICE F

RIFERIMENTI BIBLIOGRAFICI

(1) CAMAC - The definition of IML (A language for use in CAMAC systems)

ESONE , IML / 01 , october 1974
Published by the ESONE Committee.

(2) CAMAC - A modular instrumentation system for data handling (revised description and specification, 1972)

EURATOM report EUR 4100$_e$, 1972
U.S. AEC report TID - 25875

(3) CC 11 - CAMAC crate-PDP 11 interface-Type 116

CERN - NP CAMAC note 43-00, may 1972

(3a) ATC 081 - Autonomous Transfer Controller for PDP 11 Type 081

CERN CAMAC Note 40-01 Sept. 1975

(4) J CC 11 - Notice tecnique coupleur de chassis CAMAC PDP 11 type J CC 11 octobre 1975

SCHLUMBERGER instruments et systemes
departement instrumentation nucleaire
BP 47 - 92222 - Bagneux (France)
Bureaux et Laboratoires
57, Rue de Paris - 92222 - Bagneux (France)

(5) RSX-11 M - MACRO-11 Reference manual/RSX-11 M V2
May 1975, DEC-11-OMMAA - B - D

(6) RSX-11 D - MACRO-11 Reference manual/RSX-11 D V6

(7) PDP 11 - Processor handbook

(8) PDP 11 - Peripherals handbook

(9) RSX-11 M - Manuali del Sistema Operativo
RSX-11 D

(5), (6), (7), (8), (9):

Digital Equipment Corporation, USA Maynard, Massachusetts

(10) Realizzazione del linguaggio IML per la gestione di sistemi CAMAC con calcolatori PDP 11 e interfaccia J CC 11.
Tesi di laurea in fisica, realizzata presso i Laboratori Nazionali di Frascati, Istituto Nazionale di Fisica Nucleare, da B. Liotta; relatore G. Bologna.
Università degli Studi di Roma, piazzale delle Scienze 5

INDICE