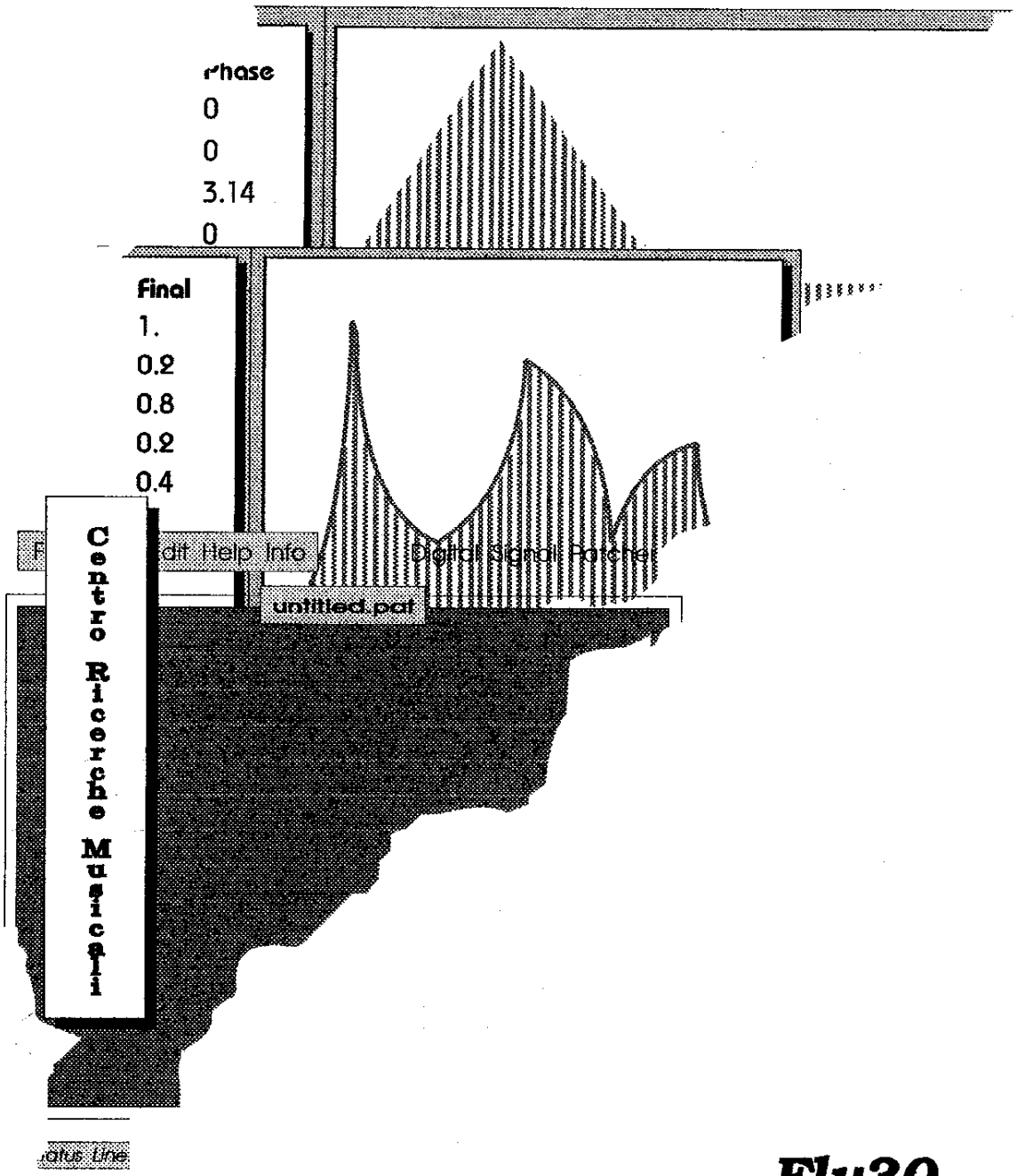


Fly30 - Digital Signal Patcher (ver. 3.1): Manuale di riferimento

SISTEMA FLY30
DIGITAL SIGNAL PATCHER

MANUALE DI RIFERIMENTO

Fly30 - Digital Signal Patcher ver. 3.1

MANUALE DI RIFERIMENTO

1. Requisiti di funzionamento

Per un corretto funzionamento, il sistema ***Fly30*** richiede i seguenti requisiti di base hardware/software:

- PC IBM o compatibile con processore 286 : xx86
- RAM:2 Mbyte
- disco fisso: 20 Mbyte
- scheda grafica: VGA
- scheda di elaborazione numerica dei segnali SPIRIT30 della SONITECH INTERNATIONAL INC.
- sistema operativo MS-DOS, ver.5.0

2. I comandi ed i moduli di DIGITAL SIGNAL PATCHER

In questo manuale verranno esaminati in dettaglio significati, funzioni e modalità d'uso di tutti i comandi e moduli contenuti nei 5 menu. La trattazione di questi ultimi e dei comandi più complessi verrà corredata da figure e facili esempi applicativi.

In questo capitolo i comandi e moduli verranno trattati all'interno e nell'ordine in cui si presentano aprendo i *menu a tendina.*

Ogni comando può essere richiamato aprendo il relativo menu a tendina e selezionandolo con il mouse o con le frecce direzionali oppure semplicemente digitando la lettera evidenziata corrispondente.

Prima di entrare nello specifico, è necessario comunque chiarire in via preliminare l'esatta interpretazione e la giusta sintassi con cui rispondere, nella "*Linea di stato*", ad alcune richieste di input da parte del programma relativamente a certi tipi di comandi.

I comandi di gestione-file **NEW, GET, PUT, LOAD, SAVE** del menu **FILE** chiedono all'utente di fornire il nome-file con cui salvare o caricare il patch desiderato. La richiesta <NOME-FILE> va presa alla lettera solo nel caso in cui si voglia archiviare o richiamare un file appartenente alla directory corrente; altrimenti, se questo si trova in una sottodirectory, digitare: *<nome sottodirectory>\<nome-file>*; se il file si trova o si vuole salvare in tutt'altra directory rispetto a quella corrente, bisogna prima digitare l'intero *<nome-percorso>* secondo la classica sintassi DOS: *<unità disco>:\<nome-percorso>\<nome-file>*. Analogamente, il comando **DIR** vuole il simbolo "." nel primo caso, il semplice *<nome sottodirectory>* nel secondo, la digitazione dell'intero *<nome percorso>* nel terzo caso.

Attivando comandi quali **VAL**, che chiedono all'utente l'immissione di un valore numerico, nella "*Linea di stato*" compare anche (tra i simboli "< >") il valore attualmente associato al parametro selezionato.

3. Menu FILE

NEW

E' il comando di *reset* del DIGITAL SIGNAL PATCHER: riporta quindi il programma allo stato iniziale cancellandone dalla memoria i patch correnti, le tabelle e le forme d'onda create, nonché i valori impostati. Prima di effettuare tali operazioni il programma chiede conferma della cancellazione invitando a salvare eventualmente i cambiamenti apportati, tramite il messaggio <FILE HAS BEEN MODIFIED, IGNORE THE CHANGES? **<RETURN>** = YES, **<ESC>** = NO>.

DIR

Consente la visualizzazione dell'elenco dei nomi-file con estensione ***.pat*** contenuti in una directory; il nome della directory desiderata va digitato in risposta al messaggio: <DIRECTORY:>. Se la directory prescelta contiene un numero di file maggiore di quanti una videata possa ospitarne (4 colonne x 7 righe), dare <INVIO> per visualizzare la pagina successiva. Si consiglia salvare i file di lavoro ***.pat*** in una directory diversa da quella in cui si trovano i file eseguibili. Così facendo, risulta molto più semplice ritrovare il proprio lavoro.

Importante: se, per qualche problema, si perde il controllo del sistema, ad esempio usando i comandi **LOAD, SAVE, DIR** nell'accesso ad un drive vuoto, si deve eseguire la seguente procedura:

Al messaggio <DISPOSITIVO NON PRONTO, ANNULLA, RIPROVA, TRALASCIA?>, rispondere digitando **<T>** (tralascia), premere quindi il tasto di invio ed infine premere il tasto **<F3>** seguito da **<R>** (redraw screen).

GET

Permette il caricamento di un elaborato grafico sotto forma di "modulo" da inserire all'interno del patch di lavoro. Ciò significa che, al contrario di quanto avviene con il comando **LOAD** (vedi oltre), all'atto del caricamento il contenuto attuale della memoria non viene azzerato ed il "modulo" richiamato con **GET** viene aggiunto al patch corrente. Inoltre, il contenuto del modulo è di natura esclusivamente grafica, non comprendendo quindi né le tabelle, le forme d'onda, gli inviluppi, i valori, né i file di conversione normalmente associati ad un file caricato come patch tramite **LOAD**.

GET richiede l'uso del mouse e del tasto funzionale **<F1>**: posizionare il puntatore del mouse nel punto dell' "*Area di lavoro*" in cui si desidera far apparire il modulo da caricare; aprire il menu file con il tasto **<F1>**, selezionando il comando **GET** con l'apposito tasto cursore; rispondere in modo appropriato al messaggio <FILE NAME:>: il disegno del modulo apparirà sullo schermo a partire dalla posizione del puntatore-mouse. Si tenga presente che quest'ultima indica anche il "punto d'origine grafica" con cui il modulo era stato precedentemente salvato tramite comando **PUT**; se, ad esempio, tale modulo era stato salvato posizionando il puntatore del mouse nell'angolo in alto a sinistra del disegno (posizione vivamente consigliata), il puntatore del mouse al momento del comando **GET** indicherà l'angolo in alto a sinistra del grafico da riprodurre.

PUT

E' il reciproco del comando **GET**: consente infatti il salvataggio di un disegno sotto forma di "modulo" e non sotto forma di patch; non vengono quindi salvate le forme d'onda, gli inviluppi ed i file di conversione associati, come invece avviene utilizzando il comando **SAVE**.

Si consiglia di archiviare i file contenenti "moduli" in una apposita directory (ad esempio una directory "**MOD**"), onde distinguerli dai file salvati come patch.

L'uso del comando **PUT** è analogo a quello del comando **GET** (vedi sopra); si ricorda che la posizione prescelta per il puntatore-mouse al momento del salvataggio del modulo, sarà il "punto d'origine grafica" a partire dal quale, con il comando **GET**, il programma riprodurrà il disegno nell' "*Area di lavoro*".

LOAD

Comando per il caricamento di un patch (comprese le tabelle, le forme d'onda, gli inviluppi ed i file di conversione associati).

Dopo la sua selezione appare la richiesta <FILE NAME:> nella "*Linea di stato*".

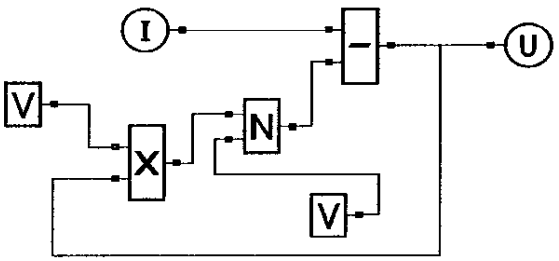
Ad ogni richiesta di un nuovo caricamento, **LOAD** azzera il contenuto attuale della memoria, cancellando da essa il patch corrente. Prima di far questo e per proteggere l'utente da cancellazioni accidentali di patch, il programma visualizza il messaggio già esaminato con il comando **NEW.**

SAVE

Comando per il salvataggio del patch corrente. Consente di rinominare eventualmente il file di lavoro visualizzando il messaggio <NOME FILE:>. Se in risposta viene digitato un *nome-file* preesistente, chiede conferma impedendo aggiornamenti non voluti al file stesso.

Si tenga presente che, per default, **SAVE** archivia il patch attuale ed i file di conversione ad esso associati nella directory corrente, dando loro l'estensione ***.pat.***

Il comando SAVE, nell'effettuare il salvataggio del patch su disco rigido, genera 5 file associati che hanno stesso nome ma diversa estensione: il primo è il file necessario al programma per il disegno del patch sullo schermo ed ha estensione ***.pat***. Esso contiene i numeri di riferimento e i valori assunti da ciascun modulo utilizzato, le coordinate grafiche [x,y] che ne identificano la posizione, nonchè le coordinate e i numeri di riferimento associati a tutte le connessioni tra i moduli stessi. A titolo di esempio, riportiamo di seguito patch grafico e relativi file di conversione per la realizzazione di un semplice eco tramite il programma DSP.



Nella figura soprastante è riportato il disegno risultante dal file ***Eco.pat***, il cui listato viene mostrato di seguito:

```
13 225 235 0.75
2 270 240 1.7e+308
13 270 270 10000
9 310 240 16
1 365 210 1.7e+308
15 430 210 1
14 145 105 1
-1
0 365 210 410 210 -1
0 365 210 395 270 245 315 230 275 250 245 -1
1 310 240 345 215 -1
```

```
2 270 270 290 245 -1
3 270 240 275 235 290 235 -1

4 225 235 250 235 -1

5 145 105 -1

5 145 105 340 105 345 205 -1

-1
```

Questo è un file in codice ASCII in realtà non visibile e non manipolabile dall'utente; esso viene semplicemente utilizzato dal programma DSP ogni volta che, tramite comando LOAD, gli si richieda il caricamento di un file grafico.

Il secondo file generato dal comando SAVE ha estensione ***.mlf*** (Music Language ***Fly30***) e contiene la traduzione del corrispettivo file ***.pat*** in un metalinguaggio ***Fly30*** il quale verrà a sua volta utilizzato da un apposito compilatore nel caso in cui si richieda al programma di connettere più patch per formarne uno generale.

Ecco il listato del file ***Eco.mlf***:

```
DEFPATCH pat\eco.pat;
V4=Val(0.75);
V3=Mul(V4,V0);
V2=Val(10000);
V1=Dln(V3,V2,16);
V0=Sub(V5,V1);
Out(V0,1);
V5=Inp(1);
ENDPATCH pat\eco.pat;
```

Anche il file *.mlf* rimane invisibile all'utente, non essendo di alcuna utilità la sua manipolazione diretta.

Il file *.asm* è il terzo file generato dal programma a partire dal comando SAVE e rappresenta la vera e propria traduzione "automatica" in codice Assembler 320C30 del patch disegnato. L'utente programmatore esperto può intervenire sul file *.asm* per ritoccarlo, per migliorarne l'ottimizzazione o nel caso preferisca lavorare direttamente in codice Assembler, al di fuori dell'interfaccia grafica del DSP, mentre l'utente medio, come accadeva per il contenuto dei file *.pat* e *.mlf*, può tranquillamente ignorarne l'esistenza e, anzi, gli vengono vivamente sconsigliate operazioni di editing e di manipolazione su file assembler generati automaticamente dal programma.

Di seguito viene riportato il listato del file *Eco.asm*:

```
; TMS320C30 ASSEMBLY SOURCE pat\eco.asm GENERATED BY
DSP PROGRAM
;
; V4=Val(0.75);
; V3=Mul(V4,V0);
                LDF   *+AR0(4),R0
                MPYF *+AR0(0),R0
                STF   R0,*+AR0(3)
; V2=Val(10000);
; V1=Dln(V3,V2,16);
                LDF   *+AR0(2),R3
                LDF   *+AR0(6),R0
                ADDF 1.0,R0
                SUBF3         R3,R0,R2
                LDFNN         R2,R0
                STF   R0,*+AR0(6)
                FIX   R0,R1
                FLOAT         R1,R2
```

```
                SUBF3       R2,R0,R2
                AND   0FFFFH,R1
                LDI   R1,IR0
                ADDI  01H,R1
                AND   0FFFFH,R1
                LDI   R1,IR1
                LDI   *+AR2(16),AR3
                LDF   *+AR3(IR0),R0
                LDF   *+AR3(IR1),R1
                SUBF  R0,R1
                MPYF  R2,R1
                ADDF  R1,R0
                LDF   *+AR0(3),R1
                STF   R1,*+AR3(IR0)
                STF   R0,*+AR0(1)
; V0=Sub(V5,V1);
                LDF   *+AR0(5),R0
                SUBF  *+AR0(1),R0
                STF   R0,*+AR0(0)
; Out(V0,1);
                LDF   *+AR0(0),R0
                ADDF  *+AR1(4),R0
                STF   R0,*+AR1(4)
; V5=Inp(1);
                LDF   *+AR1(0),R0
                STF   R0,*+AR0(5)
;
                .DATA
;
ADLN0           .WORD       012000H
```

```
;
                .SECT "VALUES"
;
CONNI           .WORD       6
CONN0           .FLOAT      0
CONN1           .FLOAT      0
CONN2           .FLOAT      10000.0000000000
                .GLOBL      CONN2
CONN3           .FLOAT      0
CONN4           .FLOAT      0.750000000000000
                .GLOBL      CONN4
CONN5           .FLOAT      0
PHAS0           .FLOAT      0
;
                .SECT "DELAYS"
;
;
; END SOURCE pat\eco.asm
;
```

Infine, nei file con estensione *.env* e *.waw* il programma memorizza rispettivamente tutti i parametri relativi agli inviluppi (n. 16 max.) e alle forme d'onda (n. 16 max.) create dall'utente tramite apposito editor all'interno del proprio patch di lavoro (vedi comando **TABLES** nel menu **EDIT**) con il programma DSP.

I listati seguenti mostrano un possibile contenuto dei file ***Eco.env*** ed ***eco.waw***:

ECO.ENV

```
1  1  1
1  1  0
1  0  0
```

ECO.WAW

```
1000  0
0  0
0  0
```

```
1  0  0                    1500  3.14
1  0  0                    0  0
1  0  0                    0  0
1  0  0                    0  0
1  0  0                    0  0
                           0  0
1  1  1                    0  0
1  1  0
1  0  0                    2000  3.14
1  0  0                    1200  0
1  0  0                    0  0
1  0  0                    0  0
1  0  0                    0  0
1  0  0                    0  0
                           0  0
..  ..  ..                 0  0
..  ..  ..                 0  0
..  ..  ..                 0  0
..  ..  ..
..  ..  ..                 ..  ..
```

Il file ***Eco.env*** contiene i 16 gruppi di 8x3 valori che identificano ciascuno un inviluppo creato dall'utente in fase di editing; infatti, per ognuno degli otto segmenti che compongono un inviluppo, il file *.env* riporta i tre

valori che lo caratterizzano: rispettivamente il valore di curvatura, di durata e il valore finale.

Ogni gruppo di 10x2 valori nel file ***Eco.waw*** riguarda invece i parametri associati a ciascuna tabella per forme d'onda e cioè, per ognuna delle 10 armoniche che la compongono, l'ampiezza (espressa in valore assoluto) e la fase (espressa in radianti).

UPDT

Consente l'aggiornamento del patch corrente, salvando il file di lavoro nella nuova versione ma conservandone il vecchio nome. E' quindi analogo al comando precedente, salvo il fatto che non prevede ridenominazione dell'algoritmo.

HCPY

Comando per l'hard copy su stampante del disegno corrente. A differenza del risultato ottenibile premendo il tasto di stampa sulla tastiera alfanumerica, il comando **HCPY** effettua esclusivamente la stampa del disegno presente sullo schermo ed il relativo nome-file (stampa cioè il contenuto dell'area (C) e dell'area (B) nella fig.1).

PLAY

E' il comando per l'esecuzione in tempo reale dell'elaborato grafico corrente.

Prima di avviare l'esecuzione, il programma si preoccupa di verificare se il patch contenga o meno moduli di ritardo (*Delays*), tabelle con forme d'onda e inviluppi. In assenza di tali elementi, il Digital Signal Patcher compie direttamente tutte le necessarie operazioni di compilazione e linking (vengono infatti visualizzati sullo schermo i messaggi informativi <COMPILING PATCH> e <LINKING PATCH>), generando due file eseguibili: uno, temporaneo, denominato ***Src30.out*** ed uno permanente con nome identico al file-patch di riferimento, ma con estensione ***.out***; quest'ultimo viene memorizzato su disco rigido nella directory corrente.

Nel caso in cui ci sia presenza di moduli delays il programma, prima di avviare l'esecuzione, azzera il contenuto della memoria delay ed avverte di ciò visualizzando il messaggio <CLEAR DELAY MEMORY>. Se invece il patch contiene degli oscillatori memorizzati in tabelle create o modificate dopo il caricamento del patch stesso, al momento del **PLAY** il programma deve generare le forme d'onda corrispondenti ed inviarle alla scheda su cui risiede il processore numerico; di tale operazione si è avvisati quando appare il messaggio <LOADING WAVEFORM N. XX>, il che significa che il Digital Signal Patcher sta generando un file di numeri *floating point* in un formato accettabile dalla scheda stessa. L'identico procedimento viene seguito per le tabelle contenenti inviluppi.

In tutti i casi sopra elencati, il risultato finale del comando **PLAY** è comunque la generazione del file *eseguibile* con estensione ***.out.***

> I file *eseguibili* con estensione ***.out*** possono anche essere direttamente caricati nella memoria del processore numerico, tramite l'apposito programma di caricamento denominato **Loader** (o **LDO**); con ciò è possibile avviare una esecuzione in tempo reale senza entrare affatto nel DIGITAL SIGNAL PATCHER; oppure lo si può comunque utilizzare (velocizzandone però le operazioni di compilazione) avviandolo con quelle particolari opzioni (sotto indicate) che inibiscono il caricamento di inviluppi e forme d'onda.

Come già anticipato nel paragrafo "Avvio di DIGITAL SIGNAL PATCHER", il programma può essere avviato in modalità diverse da quella *standard*

servendosi di particolari opzioni da digitarsi nella stessa riga del comando "DSP". Riesaminiamo brevemente le opzioni che coinvolgono direttamente l'esecuzione in tempo reale del patch di lavoro e quindi il comando in esame.

Le opzioni "/E" e "/W" inibiscono rispettivamente il caricamento degli inviluppi ed il caricamento delle forme d'onda all'atto del PLAY.

L'opzione "/I" consente l'esecuzione senza calcolo interpolatorio delle funzioni predefinite per i moduli OSC, TAB, DLS (vedi in menu **MODL**).

L'avvio con l'opzione "/O" consente invece, selezionando il comando PLAY, di rieseguire l'ultimo file *.out* creato dal programma. Dopo aver avviato il Digital Signal Patcher in questa modalità, l'operazione di compilazione relativa al PLAY è molto più veloce e sullo schermo appare quasi immediatamente il messaggio <READY TO START RET=YES, ESC=NO>, intendendo con questo che la riesecuzione dell'ultimo patch può partire all'istante. Se invece appare il messaggio <PATCH NOT COMPILED>, significa che il programma non ha trovato il file eseguibile *.out* e non può quindi lavorare in questa modalità.

SHELL

Consente di entrare nel sistema operativo DOS senza uscire dal programma. Per tornare in Digital Signal Patcher è sufficiente digitare "EXIT".

QUIT

Comando per l'uscita dal programma Digital Signal Patcher. Richiede eventualmente di salvare il lavoro corrente, visualizzando l'identico messaggio che appare con i comandi NEW e LOAD; chiede inoltre se,

uscendo dal programma, si desidera contemporaneamente il *reset* della memoria del processore numerico. Nel caso si risponda negativamente ed in presenza di un **PLAY** attivo, quest'ultima continuerà, anche dopo l'uscita dal programma, ad eseguire l'ultimo patch corrente.

4. Menu EDIT

JOIN

Comando per la connessione dei moduli elencati nel menu **MODL**. I collegamenti tra i moduli devono essere sempre del tipo: **uscita** modulo A → **ingresso** modulo B e mai viceversa. È possibile connettere più ingressi ad un'unica uscita, ma non è possibile il contrario. Per unire infatti più uscite ad un unico ingresso è necessario utilizzare il modulo-somma **ADD** del menu **MODL**.

Una volta attivato il comando **JOIN**, il puntatore del mouse assume la forma di una matita rossa, con la cui punta bisogna selezionare il piedino di **uscita** di un modulo (il piedino viene evidenziato a questo punto da un quadratino verde), quindi trascinare il puntatore verso **l'ingresso** del modulo da connettere (se la connessione è andata a buon fine, anche il piedino di quest'ultimo risulterà evidenziato).

Il comando **JOIN** rende visibili le connessioni tramite segmenti di retta di color arancio. Il percorso grafico di tali "fili" di collegamento può essere costituito da un'unica linea retta che congiunga un modulo all'altro, oppure da più segmenti consecutivi orizzontali e verticali (mai obliqui), laddove la complessità del disegno lo richieda. Per spezzare la retta è sufficiente selezionare con il mouse il punto dell' "*Area di lavoro*" che il singolo segmento deve raggiungere: questo verrà automaticamente disegnato dal programma. Con **JOIN** possono anche essere creati dei "fili volanti": quei fili, cioè, che non terminano in nessun ingresso di moduli. Ciò si rivela utile, ad esempio, nel caso in cui si voglia rimandare ad un secondo momento la continuazione del patch di lavoro, o gli si voglia annettere un sotto-patch precedentemente creato, richiamandolo tramite comando **GET** (vedi menu **FILE**).

È anche possibile unire tra loro i "fili volanti" o collegarli a ingressi e uscite dei moduli.

Infine, le connessioni indesiderate possono essere eliminate ricorrendo al comando **DELETE** del presente menu; in questo caso fare attenzione alla cancellazione di un modulo già connesso, perché i collegamenti che lo

riguardano non vengono automaticamente ripristinati tra i moduli ad esso adiacenti.

MOVE

Il comando **MOVE** va utilizzato per lo spostamento ed il posizionamento fine di moduli all'interno dell' "*Area di lavoro*".

È possibile muovere esclusivamente un modulo alla volta e, nel caso questo fosse stato precedentemente connesso, l'operazione non prevede il trascinamento automatico delle connessioni nella nuova posizione: queste ultime andranno infatti ripristinate manualmente.

Dopo aver scelto il comando **MOVE**, selezionare con il mouse il modulo desiderato: questo verrà cancellato dallo schermo; quindi portare il puntatore nella nuova posizione e selezionare di nuovo: il modulo riapparirà nel punto prescelto.

VALUE

VALUE assegna un "*valore associato*" ad uno qualsiasi dei moduli elencati nel menu **MODL**, ad eccezione dei moduli aritmetici **ADD, SUB** e **MUL** che non richiedono assegnazione di valore. Per ogni modulo, il valore attualmente memorizzato ed il messaggio con la richiesta del nuovo valore da assegnare, compaiono nella "*Linea di stato*" (area (D) nella Fig. 1).

Per utilizzare il comando **VALUE**: dopo averlo attivato dal menu (il puntatore appare come una matita gialla), selezionare con il mouse un modulo tra quelli presenti nell'algoritmo grafico di lavoro. Nella "*Linea di stato*" controllare ed eventualmente modificare il valore associato al modulo, mediante tastiera numerica o tasti cursore.

A seconda del tipo di modulo selezionato, il comando **VALUE** accetta esclusivamente dei valori che per il DIGITAL SIGNAL PATCHER risultino

congrui. Se viene digitato un valore che esula da quelli consentiti, il programma invia il messaggio: <BAD VALUE, RANGE IS x xxxx>; dare **<ESC>** o **<INVIO>** e digitare dei numeri corretti.

Esaminiamo di seguito, per ciascuno dei moduli relativamente ai quali è previsto l'uso di VALUE, gli ambiti entro cui i valori devono essere forniti ed il loro significato, rimandando al menu **MODL** per ulteriori chiarimenti sulla natura e funzione dei moduli stessi.

modulo PHS

Ricordando che questo modulo è in realtà un generatore d'onda a *dente di sega*, VALUE chiede che venga fornito il valore iniziale **assoluto** della fase, con un numero che quindi può variare da -32768 a 32767. Un valore al di fuori di questo ambito non dà comunque errore, in quanto viene internamente corretto.

moduli TAB, MEM, OSC

Viene richiesto il numero identificatore della tabella a cui si vogliono apportare modifiche. Il messaggio visualizzato nella "*Linea di stato*" è, per tutti e tre i moduli, <TAB NUMBER:>; dal momento che il numero massimo consentito di tabelle è 16, i valori ammissibili vanno da 1 a 16. In tutti e tre i moduli le fasi sono espresse in gradi.

modulo ENV

Anche nel caso del modulo ENV viene richiesto il numero identificatore dell'inviluppo prescelto, tramite messaggio <ENV NUMBER:>. Valori accettati: da 1 a 16.

modulo DLY (1 pos.)

Chiede il valore iniziale da assegnare alla memoria delay di 1 posizione, mediante messaggio <INIT VALUE:>. Valori consentiti: qualsiasi valore reale *floating point.*

moduli DLN (2^n pos.) e DLS (delay multiplo con incremento)

VALUE richiede qui il valore del ritardo in termini di unità logaritmiche in riferimento alle celle di memoria implicate. Così, ad esempio, se in risposta al messaggio <DELAY (log):> si digita il n. **"1"**, il significato sarà: 2^1 celle di memoria; digitando invece il n. **"10"**, si avrà una linea di ritardo di 2^{10} celle di memoria (posizioni). Anche qui i valori ammissibili vanno da 1 a 16.

modulo FLT

Per il filtro IIR di second'ordine, bisogna fornire il valore iniziale di innesco del filtro stesso; viene infatti visualizzato il messaggio <INIT VALUE:>. Valori accettati: qualsiasi valore *floating point.*

modulo RND

Tramite messaggio <RANDOM NUMBER:>, viene richiesto il valore di innesco del rumore; bisogna digitare un numero in *floating point* tanto più complesso quanto più si desidera un risultato sonoro vicino al *rumore bianco.*

modulo IFP

In questo caso viene richiesto il valore di soglia (K) da assegnare al primo ingresso (A): valori superiori a quello fornito determinano per l'uscita (U) del modulo il secondo ingresso (B), mentre valori inferiori a K determinano per (U) il terzo ingresso (C). Se ad esempio viene assegnato a K il valore 1000, ed essendo K1 il valore immesso nell'ingresso A, si avrà:

per K1 > = 1000	U = B
per K1 < 1000	U = C

Valori ammissibili: qualsiasi valore reale.

moduli INP e OUT

Sono accettabili valori da **"1"** a **"4"** e cioè tanti quanti (in presenza di due schede ADC/DAC) sono i canali di conversione disponibili. Normalmente, lavorando con un'unica scheda, si opera con i canali 1 e 2.

TABLES

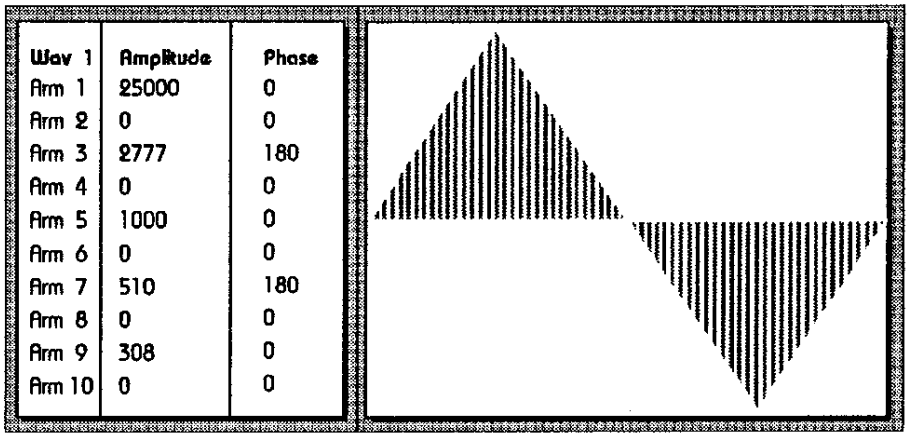
Questo comando apre un *editor* implementato esclusivamente per quei moduli i cui parametri possono essere gestiti tramite tabelle: **OSC, ENV, TAB, MEM.** Se si tenta di attivare l'*editor* di tabelle a partire da un modulo per il quale queste non siano previste, appare il messaggio <THIS MODULE NEEDS NO TABLE> ed il programma attende un **<ESC>** o **<INVIO>** prima di accettare la selezione di uno dei moduli sopra elencati.

Dopo aver scelto **TABLES** nel "*menu a tendina*", il puntatore del mouse appare come un'onda triangolare incorniciata, di color arancio. A questo punto, come avveniva per il comando **VALUE**, bisogna selezionare con il puntatore il modulo del quale si vuole editare la tabella. Fatto questo, alla "*Area di lavoro*" si sovrappone quasi completamente il *Menu* dell'*editor* che si riferisce alla tabella precedentemente assegnata dall'utente al modulo selezionato. Se, ad esempio, tramite comando **VALUE**, l'utente aveva assegnato a quel modulo la tabella n. 8, saranno il grafico e lo stato dei parametri relativi a quest'ultima a comparire nel *Menu* dell'*editor*.

In questa situazione, se lo si desidera e senza necessariamente percorrere a ritroso tutte le operazioni già effettuate, si possono editare in successione le altre 15 tabelle servendosi dei tasti funzionali **<F1>** e **<F2>**: il primo edita le tabelle successive a quella corrente, il secondo le precedenti.

Illustreremo ora i due tipi di *editor* disponibili: quello di tabelle per la creazione di forme d'onda e quello per inviluppi. La figura che segue mostra, a titolo esemplificativo, il *Menu editor* di una tabella per forme

d'onda e ci aiuterà, insieme alla successiva figura di un *editor* per inviluppi, a capire il funzionamento di questo strumento.



Come si vede, la parte sinistra del *Menu editor* è destinata ad accogliere, per un massimo di 10 armoniche, i valori di ampiezza e fase che l'utente deve inserire onde creare la forma d'onda desiderata. La visualizzazione di un periodo del segnale creato viene invece ospitata nella parte destra del *Menu.*

Le ampiezze vanno fornite in valori assoluti con numeri *floating point.* Si tenga presente che 32767 è il valore massimo dell'ampiezza **complessiva** che il convertitore DAC/ADC è in grado di accettare senza produrre fenomeni di *overflow.* Ciò significa che, se ad esempio un algoritmo prevede l'uso di 16 segnali sinusoidali in fase e della medesima ampiezza, a ciascuno di essi non potrà essere assegnato un valore superiore a **"2000"**. Normalmente, l'ampiezza minima che un segnale deve avere per essere percepibile uditivamente è intorno al valore **"1000"**; tale valore è quello presettato per *default* dal DIGITAL SIGNAL PATCHER su tutte e 16 le tabelle per forme d'onda contenenti, prima delle modifiche apportate dall'utente, altrettante sinusoidi.

Le fasi vanno invece espresse in **gradi.**

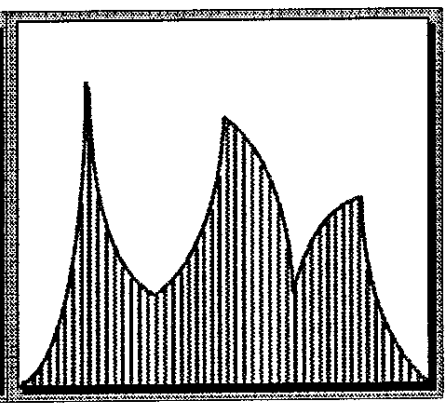
Per muoversi tra i campi che compongono le colonne "***Amplitude***" e "***Phase***" e per effettuare le selezioni, bisogna servirsi dei tasti cursore + <INVIO> o del mouse; una volta selezionato, il campo prescelto viene evidenziato e nella "*Linea di stato*" compare il messaggio <VAL:> : digitare il valore desiderato e premere di nuovo <INVIO>; il contenuto del campo verrà aggiornato e, di conseguenza, la rappresentazione grafica della forma d'onda.

Per uscire dall'*editor*, basta premere il tasto <ESC> e rispondere in modo appropriato al messaggio: <DOWN LOAD WAVEFORM? RET=YES, ESC=NO>, vale a dire: "vuoi caricare ora le tabelle nella memoria del DSP ?". Se si rinuncia al caricamento immediato premendo <ESC>, il programma lo effettuerà al momento del PLAY (a meno che DSP non sia stato avviato con l'opzione "/N"). Nell'effettuare l'una o l'altra delle scelte, si tenga presente che DIGITAL SIGNAL PATCHER impiega un certo tempo per completare questa operazione (nell'ordine di alcuni secondi), tanto più se le tabelle sono numerose e complesse.

Esaminiamo ora il secondo tipo di *editor* per tabelle: quello destinato alla creazione di inviluppi.

La figura sottostante, che ora commenteremo, ne mostra un esempio:

Env 1	Curve	Length	Final
Seg 1	1.0001	1.	1.
Seg 2	0.9999	1.	0.2
Seg 3	1.0001	1.	0.8
Seg 4	1.0001	1.	0.2
Seg 5	0.9999	1.	0.4
Seg 6	0.9999	1.	0.
Seg 7	1.	0.	0.
Seg 8	1.	0.	1.



il DIGITAL SIGNAL PATCHER è in grado di generare inviluppi applicabili ad un qualsivoglia parametro (ampiezza, frequenza, fase, valori all'ingresso di filtri, ecc.) e di ciclarli all'infinito oppure di interromperli dopo una prima esecuzione. Ogni inviluppo è costruito tramite segmenti di

ampiezza, curvatura e durata variabili ed in numero massimo di 8. È appunto l'elenco di tali segmenti quello che compare nella prima colonna del *Menu editor*. Nella seconda colonna l'utente deve invece introdurre i valori di curvatura relativi a ciascun segmento: il valore "**1**" sta ad indicare una retta, mentre (come si vede nella figura) valori superiori all'unità forniscono curvature esponenziali e valori inferiori generano curve logaritmiche. Trovandoci in regime *floating point*, si raccomanda di introdurre numeri sempre molto prossimi all' "**1**", nell'ordine di quelli esemplificati all'interno della figura, dalla quale si noterà che è sufficiente una variazione di "**0.0001**" per imprimere una curvatura notevole al segmento.

Nella colonna "***Length***" vanno indicate le durate in secondi di ciascun segmento. Si noti nell'esempio che, avendo voluto disegnare un inviluppo di soli 6 segmenti, si è usato l'artificio di porre a "**0**" la durata degli ultimi due: questo è l'unico modo per avere meno segmenti del numero massimo previsto. Naturalmente i valori di durata possono essere esclusivamente positivi.

La colonna "***Final***" ospita invece i valori finali da attribuire a ciascun segmento, con numeri *floating point* che possono essere sia positivi che negativi.

Infine, pur essendo il lettore di tabelle concepito per esecuzioni a ciclo continuo, è possibile creare inviluppi che vengano bloccati dopo una prima lettura della tabella. Ciò si ottiene ricorrendo ad un semplice artificio: è necessario che **l'ultimo** dei segmenti attivati sia un segmento "fittizio"; esso deve avere lunghezza = "**0**" e valore finale superiore a quello del segmento precedente. Osservando l'esempio riportato in figura, si noterà che il segmento 8 risponde a questi requisiti e che quindi siamo in un caso in cui la lettura della tabella non verrà ciclata.

L'uso operativo di questo secondo *editor* è analogo al precedente, così come è analogo il messaggio che appare una volta premuto il tasto **<ESC>** per uscire dall'*editor* stesso: <DOWN LOAD ENVELOPE? **RET** = YES, **ESC** = NO>, vale a dire: "Vuoi caricare **ora** gli inviluppi ?".

DELETE

DELETE effettua cancellazioni di moduli e connessioni, compiendo tali operazioni una alla volta.

Selezionare **DELETE** dal "*menu a tendina*": il puntatore del mouse si trasforma in un piccolo cancellino di colore fucsia. Selezionare i moduli e le connessioni da eliminare **sempre** con l'angolo superiore sinistro del puntatore-mouse.

SHOVAL

Visualizza sullo schermo all'interno del disegno corrente i numeri (di colore verde) corrispondenti ai valori assegnati dall'utente ai moduli tramite comando **VALUE.**

L'utilità delle cifre visualizzate appare evidente: esse costituiscono per l'utente una sorta di *memo*, relativamente alle operazioni effettuate con **VALUE**

SHOVAL è un comando del tipo ***on/off***: con una prima selezione viene attivato, con una seconda viene disattivato. Se, prima di aver disattivato il comando, si modificano degli elementi del patch, bisogna premere **<ESC>** affinché **SHOVAL** si aggiorni di conseguenza.

SHOCON

E' analogo al comando precedente.

Visualizza sullo schermo, all'interno del patch disegnato, i numeri di identificazione assegnati dal programma alle connessioni create dall'utente stesso. Questi numeri (di color arancio) rivestono interesse per l'utente esperto o "curioso" che voglia, ad esempio, rintracciare le singole

connessioni nel listato del file ***.pat***; ma soprattutto sono importanti per il programmatore che desideri interfacciare il Digital Signal Patcher con altri programmi, oppure per l'utente dei programmi applicativi ***Fly30.***

Per quanto riguarda il comportamento ***on/off*** di **SHOCON**, si veda il precedente comando.

REDRAW

Questo comando ridisegna lo schermo. **REDRAW** si rivela utile, ad esempio, nei casi in cui l'utente o l'Assemblatore 320C30 con cui il programma interagisce, abbiano "sporcato" la videata con caratteri estranei.

MODIFY

È il comando per la modifica il tempo reale dei valori in *floating point* assegnati: agisce quindi esclusivamente sui valori dei moduli **VAL**. **MODIFY** può essere attivato solo successivamente al comando **PLAY**; se non si fa questo, viene visualizzato il messaggio di errore <YOU CAN MODIFY ONLY AFTER PLAY>.

Una volta prescelti i comandi **PLAY** e **MODIFY** dal menu selezionare quel modulo **VAL** di cui si intende modificare il valore in tempo reale (è possibile agire solo su un modulo per volta). Nella "*Linea di stato*" compaiono ora due elementi: <Val:> (sta per "valore") e <Inc:> (sta per "incremento"). Il valore può essere incrementato o decrementato con i tasti cursore superiore e inferiore; con i tasti cursore destro e sinistro, viene invece aumentato o diminuito lo *step incrementale* "**Inc**" con cui "**Val**" viene modificato. L'unità di misura con cui il comando modifica i valori è esponenziale, dato che con **MODIFY** si è voluto creare soprattutto un *trimmer* agile, veloce e di immediato riscontro uditivo, più che un "selettore fine" dei valori parametrici.

Una volta ottimizzato il parametro corrente, è sufficiente premere <INVIO> per memorizzarne il valore e passare al parametro successivo.

Uscendo da **MODIFY** al solito con il tasto <ESC> o il tasto-deselezione del mouse, il programma chiede se si desidera aggiornare il patch con i nuovi valori introdotti mediante questo comando, visualizzando: <UPDATE MODIFY VALUES? **RET** = YES, **ESC** = NO>. Se si risponde affermativamente, tutti i moduli **VAL** vengono aggiornati.

SWITCH

Permette di controllare i selettori di programma direttamente all'interno di DIGITAL SIGNAL PATCHER. Per *default* tutti i valori sono **off.**

NODWLDWAV(w)		off
NODWLDENV	(e)	off
NOCOMPILE	(o)	off
NOINTERP	(i)	off
FREQOSC	(f)	off
TIMEDLN	(t)	off

NODWLOWAV= se è posto ad **on** non carica le forme d'onda nella scheda all'atto del **PLAY.**

NODWLOENV = se è posto ad **on** non carica gli inviluppi nella scheda all'atto del **PLAY.**

NOCOMPILE = se è posto ad **on** non compila nuovamente il disegno del patch ma manda in esecuzione l'ultimo file compilato all'atto del **PLAY.**

NOINTERP = se è posto ad **on** disabilita l'interpolazione per i moduli **OSC, TAB, DLN, DLS** all'atto del **PLAY.**

FREQOSC = se è posto ad **on** il primo ingresso del modulo **OSC** (frequenza) viene espresso in **Hertz** e non in step incrementale.

TIMEDLN = se è posto ad **on** esprime il ritardo del modulo **DLN** in **millisecondi** e non in *celle di ritardo*.

Lo stato degli SWITCH viene salvato e memorizzato insieme al file con estensione ***.pat.***

ADJCON

Elimina eventuali imperfezioni nel disegno delle connessioni.

CHECK

Mostra una tabella in cui sono visualizzate alcune informazioni sul patch corrente.

CHECK RESULTS		PATCH
TOTAL CONNs	:	#
Float Input	:	#
Float Output	:	#
Next Handle	:	#
Void Handle	:	#
Multi Handle	:	#
TOTAL MODULs	:	#
Float Input	:	#

```
Float Output      : #
TOTAL ASMLINE     : #

TOTAL DSPTIME     : #
```

TOTAL CONNs = riporta il numero totale di connessioni contenute nel patch.

Float Input = indica le connessioni con ingresso non collegato.

Float Output = indica le connessioni con uscita non collegata.

Next Handle = indica il numero della prossima connessione che verrà disegnata.

Void Handle = rappresenta il numero degli *handle* vuoti poichè cancellati.

Multi Handle in uscita. = indica che alcuni moduli hanno più connessioni

TOTAL MODULs patch. = esprime il numero totale di moduli contenuti nel

Float Input moduli. = indica il numero di ingressi non collegati di

Float Output moduli. = indica il numero di uscite non collegate di

TOTAL ASMLINE = esprime il numero totale di istruzioni *assembler* del patch corrente.

TOTAL DSPTIME = indica il tempo totale di calcolo del patch espresso in nanosecondi. Questo tempo deve essere inferiore al periodo di campionamento (1/Fc= 1/25KHz = 40000 nanosecondi). In caso contrario si provoca il dimezzamento della frequenza di campionamento e quindi delle frequenze generate.

BOARD

Seleziona il numero di scheda se l'utente ne possiede più di una. Per un corretto funzionamento deve essere posta la seguente variabile di *environment:*

SET SPIRIT30 = <*ind1*, *ind2*, ...*indn*>

dove *ind1*, *ind2*,..*indn* sono gli indirizzi delle singole schede.

Se si possiede una sola scheda si può omettere la variabile di *environment* e per *default* l'indirizzo della scheda è **" 300 "**

DELORF

Attenzione!

Cancella **moduli** e **connessioni** che non hanno alcun tipo di collegamento.

DELCOF

Cancella tutte le **connessioni** che hanno un ingresso o un'uscita non collegati.

DELMOF

Cancella tutti i **moduli** che hanno un ingresso o un'uscita non collegati.

DELFLO

Attenzione!

Cancella **moduli** e **connessioni** che hanno un ingresso o un'uscita non collegati.

IMPORTANTE: I comandi **DELORF, DELCOF, DELMOF, DELFLO** non possono essere richiamati con un tasto ma solamente selezionati con il mouse.

5. Menu MODL

Prima di passare alla descrizione dettagliata dei moduli disponibili nel DIGITAL SIGNAL PATCHER (tutti contenuti nel presente menu), è utile ricordare che:

- la frequenza di lavoro dei convertitori ADC/DAC è fissata per *default* a 22.05 KHz.

- Tutte le operazioni aritmetiche vengono effettuate in *floating point*, con una precisione numerica pari a 2^{-38}; tali operazioni non danno luogo ad *overflow*. Tutti i moduli che utilizzano lettura tabellare effettuano una interpolazione lineare con 24 bit di precisione; l'interpolazione può essere inibita per i moduli oscillatori **DLS**, **OSC** e **TAB** avviando DIGITAL SIGNAL PATCHER in modalità "/I";

- il programma prevede l'uso di tabelle editabili dall'utente per i moduli **TAB**, **MEM**, **OSC**, **ENV**; tali tabelle, al momento della generazione, hanno lunghezza pari a $2^{12} = 4096$ punti;

- ad eccezione dei tre moduli aritmetici **ADD**, **SUB** e **MUL**, ogni modulo del presente menu necessita, oltre ai valori da assegnare ai suoi ingressi e uscita, di un "*valore associato*" da fornire tramite comando **VALUE** del menu **EDIT**: dal momento che il significato di tale valore varia a seconda della funzione svolta dal singolo modulo, per ulteriori dettagli in merito si faccia riferimento alla casistica riportata nella sezione dedicata a questo comando;

- scorrendo la descrizione dettagliata dei moduli DSP, il lettore noterà che molti di essi richiedono in input l'immissione di valori espressi in unità di misura poco "canoniche" per l'utente musicista. Valga un esempio per tutti: un oscillatore (o altro modulo analogo) viene impiegato dalla maggior parte dei programmi per la sintesi del suono

nella generazione di segnali audio periodici; di conseguenza all'utente viene normalmente richiesta l'immissione dei valori di frequenza (espressi in Hz) e di ampiezza (espressa in dB) nei rispettivi ingressi dell'oscillatore stesso.

Non è così per il modulo OSC avviando il DSP in modalità *standard*; tale modulo, richiedendo in ingresso un valore di step di lettura tabellare (in luogo della frequenza espressa in Hz) ed un valore di amplificazione (in luogo dell'ampiezza espressa in dB), consente un uso molto più esteso e creativo di questo strumento: ad esempio è possibile impiegare il modulo OSC per la rilettura, con fattore di amplificazione e step incrementale variabili, di una forma d'onda acquisita e memorizzata tramite modulo MEM; appare evidente che, in questo caso, avere all'ingresso di OSC frequenza in Hz e ampiezza in dB sarebbe, oltre che teoricamente fuorviante, anche notevolmente scomodo.

Tale impostazione è perfettamente in linea con la filosofia di progettazione di DIGITAL SIGNAL PATCHER, con cui si è voluto creare uno strumento soprattutto destinato a favorire la più ampia sperimentazione algoritmica possibile, proprio consentendo la massima flessibilità d'uso delle sue componenti modulistiche.

Nel sistema ***Fly30*** le fasi di elaborazione di una partitura musicale e di una sua esecuzione in tempo reale o differito sono infatti affidate a programmi applicativi "satelliti" ma indipendenti rispetto a DSP, appositamente progettati per "musicare" ciò che con DIGITAL SIGNAL PATCHER sia stato in precedenza algoritmicamente testato e sperimentato. Lavorando all'interno di questo programma, l'utente musicista avrà comunque la possibilità di ricondurre parametri, valori ed unità di misura in un ambito a lui più noto: nella descrizione dettagliata dei moduli DSP, laddove è necessario, vengono infatti riportate delle semplici formule di conversione dei valori e la trattazione di ogni modulo è corredata da facili esempi che ne illustrano la polifunzionalità.

Comunque, proprio riferendoci all'esempio sopra riportato, l'utente musicista potrà semplificare il proprio lavoro avviando il programma in modalità **"/F"** che, ricordiamo, consente di assegnare agli appositi ingressi dei moduli oscillatori un valore di frequenza espresso in Hz,

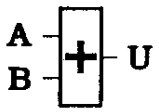
piuttosto che un valore assoluto di incremento (si vede anche il comando SWITCH del menu **EDIT**).

Ciò premesso, passiamo ora alla descrizione dettagliata dei moduli DSP contenuti nel presente menu, nell'ordine in cui si presentano al suo interno aprendo la relativa finestra a tendina.

ADD

ADD è, insieme a SUB e MUL, uno dei tre moduli aritmetici di cui dispone il DSP.

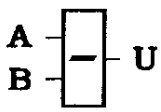
Il modulo ADD fornisce in uscita (U) la somma dei valori connessi ai suoi due ingressi (A) e (B). Simbolo grafico ed operazione effettuata sono i seguenti:



$$U = A + B$$

SUB

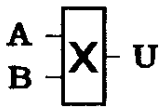
Effettua la sottrazione tra i valori provenienti da moduli la cui uscita sia connessa ai suoi ingressi (A) e (B); è rappresentato ed opera nel modo seguente:



$$U = A - B$$

MUL

Modulo per la moltiplicazione di due valori provenienti da moduli collegati ai suoi due ingressi (A) e (B). Simbolo grafico e funzione:



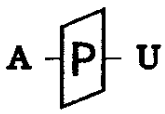
$$U = A * B$$

PHS

Il modulo **PHS** è un generatore di rampa, ossia un oscillatore che produce in uscita (U) un'onda a dente di sega, avendo fornito all'ingresso (A) un valore costante o variabile che rappresenta lo step incrementale della rampa stessa. In particolare:

per A > 0	U varia fra 0 e 65535
per A < 0	U varia fra -65535 e 0

Il simbolo grafico di **PHS** è:



PHS, come tutti i moduli seguenti, necessita anche di un "*valore associato*" da fornire tramite comando **VALUE** del menu **EDIT**. In questo caso, tale valore rappresenta il valore iniziale (assoluto) della rampa.

L'uso più semplice che si possa fare di questo oscillatore è quello di impiegarlo come semplice generatore di un'onda a dente di sega,

collegandone direttamente l'uscita ad un modulo **OUT** per inviare il segnale al DAC:

In questo caso si desidererà sicuramente attribuire all'ingresso (A) del modulo **PHS** (ingresso che è qui collegato ad un modulo **VAL**) il significato di frequenza dell'oscillazione stessa.

Si deve quindi tener presente la relazione che, nella fattispecie, lega step incrementale a frequenza espressa in Hz:

$$Si = \frac{f \cdot 2^{16}}{22050}$$

dove:

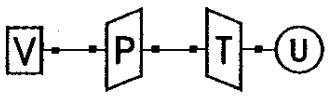
f = frequenza in Hz;

2^{16} = lunghezza della tabella utilizzata da questo modulo per memorizzare la forma d'onda;

22050 = frequenza di campionamento *default* con cui opera il DSP. Se ad esempio si desidera una frequenza di oscillazione di 200 Hz, il valore da attribuire ad (A) sarà:

$$\frac{200 \cdot 65536}{22050} = 594.431$$

L'esempio che segue mostra invece come **PHS** si renda necessario per l'attribuzione degli indirizzi per il modulo **MEM** o **TAB**. Tale uso rende effettivamente conto del nome attribuito a **PHS**: quello di "modulo fase":



il modulo **PHS** è posto qui all'ingresso di un modulo **TAB** (vedi) e consente di leggerne il contenuto con continuità a partire da un indirizzo definito. Essendo il modulo **TAB** un gestore di tabelle di lunghezza $2^{12} = 4096$ punti, il valore posto all'ingresso di **PHS** dovrà tener conto di tale dimensionamento: il valore dello step incrementale da immettere all'ingresso (A) del modulo **PHS** sarà quindi legato alla frequenza (in Hz) dalla seguente relazione:

$$Si = \frac{f \cdot 2^{12}}{22050}$$

TAB

Il modulo **TAB** va utilizzato per la generazione di forme d'onda tramite tabella pre-editata. L'editing di tabelle va effettuato servendosi del comando **TABLES** (menu **EDITOR**). Per attribuire una determinata tabella pre-editata a quel determinato modulo **TAB** che si incaricherà di leggerne la forma d'onda al momento dell'esecuzione del patch, bisogna fornire a **TAB** un "*valore associato*" (vedi comando **VALUE**) che corrisponda al numero identificatore della tabella editata.

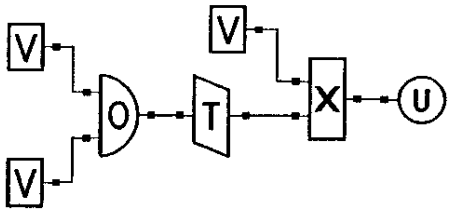
Il modulo **TAB** ha un unico ingresso (A) e fornisce in uscita (U) il seguente risultato, che riportiamo insieme al simbolo grafico:

U = TAB [A]

dove (A) rappresenta l'indirizzo di lettura in tabella. In uscita (U) si avrà quindi la lettura del contenuto pre-editato della tabella, indicizzato dall'ingresso (A).

Dal momento che la dimensione della tabella è uguale a 4096 punti, il valore di (A) in *floating point* viene trasformato in indirizzo intero a 12 bit; i valori letti nella tabella sono invece interpolati linearmente a 24 bit, a meno che non si stia lavorando in modalità **"/I"**.

Un impiego elementare di **TAB** è stato già presentato trattando del modulo **PHS**; di seguito ne viene invece evidenziato l'uso all'interno di un classico algoritmo di *distorsione non lineare*:



in questo caso un segnale periodico generato con modulo **OSC** viene distorto secondo la forma d'onda memorizzata in **TAB**, quindi moltiplicato per un valore costante di offset, prima di essere inviato al modulo **OUT** per l'uscita al convertitore.

MEM

Consente di memorizzare in una tabella di 4096 punti la forma d'onda di un segnale associato al primo dei suoi due ingressi: l'ingresso (A). Nel secondo ingresso (B) va invece immesso un valore costante o variabile che indica l'indirizzo di scrittura in tabella. Il segnale connesso all'ingresso (A) può venir generato digitalmente all'interno di **DSP** o tramite *tools* ***Fly30***, oppure può essere un segnale analogico convertito in digitale tramite apposito software (ad esempio tramite il programma **PCM** che è anch'esso parte integrante del sistema ***Fly30***). In entrambi i casi, comunque, la forma d'onda del segnale va editata tramite comando

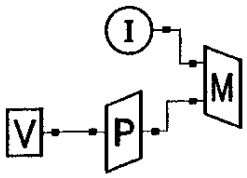
TABLES del menu **EDITOR**: infatti il "*valore associato*" al modulo MEM è il numero identificatore, compreso tra 1 e 16, della tabella editata.

Il simbolo grafico di MEM e la sua funzione sono:

TAB[B] = A

Si noti che MEM non dispone di un'uscita direttamente collegabile ad altro modulo: vedremo tra poco in quale altro modo sia possibile utilizzarne il risultato all'interno di un patch.

Nell'esempio seguente all'ingresso (A) di MEM è collegato un segnale analogico tramite modulo INP; contemporaneamente, il modulo fase posto all'ingresso (B) consente una scrittura continua con fase assegnata del contenuto introdotto in (A), scrittura che viene conservata in un vettore di 4096 punti:



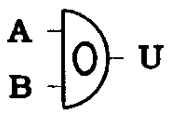
tipicamente, un tale algoritmo di scrittura tabellare andrà associato ad un corrispettivo algoritmo di lettura per produrre un risultato in uscita. Per far questo si può ad esempio utilizzare un modulo TAB, facendo attenzione che quest'ultimo abbia lo stesso "*valore associato*" (numero identificatore di tabella) del corrispettivo modulo MEM. A titolo esemplificativo, è possibile associare in un unico patch l'algoritmo di lettura tabellare qui presentato con il secondo esempio riportato per il modulo PHS, naturalmente assegnando a MEM e TAB lo stesso numero di tabella. Il risultato sarà che, con le modalità dovute ai valori connessi ai rispettivi ingressi, il modulo TAB consentirà la lettura ciclica del contenuto memorizzato in MEM e, tramite modulo OUT, il risultato verrà inviato al convertitore.

OSC

Il modulo OSC è un oscillatore, ossia un classico generatore di segnali periodici. Un oscillatore ripete ciclicamente una forma d'onda, memorizzata in tabella, con ampiezza e step di lettura (oppure frequenza) fissati di volta in volta dall'utente, il quale deve per questo fornire gli opportuni valori costanti o variabili ai due ingressi (A) e (B) del modulo stesso. Anche OSC utilizza tabelle di 4096 punti editabili tramite comando TABLES ed identificate da un numero compreso tra 1 e 16.

In modalità *standard* le forme d'onda disegnate in tali tabelle verranno generate tramite interpolazione, mentre in modalità "/I" il calcolo interpolatorio viene inibito.

Il disegno del modulo OSC è:



dove l'ingresso (A) rappresenta lo step di lettura della tabella, se il programma è stato avviato in modalità standard; (B) rappresenta il valore di amplificazione dei valori forniti al momento dell'editing della tabella stessa. In uscita (U) si avrà:

$$U = TAB[fase] \cdot B$$

dove:

$fase = (\text{fase} + A)_{\text{mod } 4096}$.

Per quanto riguarda la fase, dunque, il modulo opera sommandone il valore a quello fornito all'ingresso (A).

Dal momento che, nella maggioranza dei casi, l'utente utilizzerà l'ingresso (A) come ingresso per la frequenza da attribuire al segnale in uscita,

indichiamo la semplice relazione che anche qui lega step incrementale a frequenza espressa in Hz:

$$Si = \frac{f \cdot 4096}{22050}$$

dove:

f = frequenza di oscillazione in Hz;

$4096 = 2^{12}$ = lunghezza prefissata della tabella;

22050 = frequenza di campionamento *default* nel DIGITAL SIGNAL PATCHER. Così, ad esempio, se si desidera un segnale che oscilli a 200 Hz, bisognerà fornire in (A) il valore incrementale:

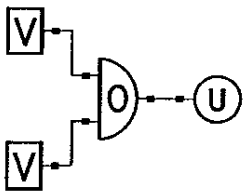
$$\frac{200 \cdot 4096}{22050} = 37.152$$

Non volendo effettuare manualmente tale calcolo di conversione, ricordiamo che l'utente può servirsi dell'opzione **"/F"** all'atto dell'avvio di DSP o tramite comando **SWITCH** del menu **EDIT**. Tale modalità consente, nel caso specifico, di immettere nell'ingresso (A) del modulo **OSC** direttamente un valore espresso in Hz.

Per quanto riguarda il valore di amplificazione da immettere in (B), questo va espresso come di consueto in *floating point* e può essere costante o variabile. Nel primo caso si collegherà (B) ad un modulo **VAL**, nel secondo si utilizzerà normalmente un generatore d'inviluppo (vedi modulo **ENV**).

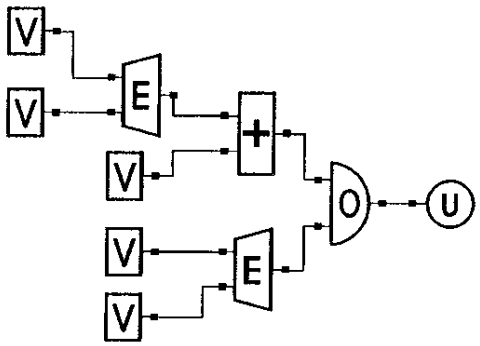
Infine, come per gli altri gestori di tabelle, anche **OSC** richiede un "*valore associato*" che indichi il numero di tabella in cui si sarà editata la forma d'onda da generare.

L'utilizzo più elementare di **OSC** può essere esemplificato come segue:



in cui l'oscillatore è semplicemente controllato in ampiezza e frequenza da due valori costanti e il segnale risultante è direttamente inviato in uscita.

Il secondo esempio mostra invece un oscillatore con inviluppo di ampiezza, realizzato tramite ENV, e con frequenza glissante ugualmente controllata tramite un ENV alla cui uscita viene sommato un valore di offset:



ENV

Modulo destinato alla generazione di tabelle contenenti funzioni di inviluppo. Per editare tali funzioni va di nuovo utilizzato il comando TABLES del menu **EDITOR** mentre, come "*valore associato*", va fornito il numero identificatore della tabella desiderata.

ENV dispone di due ingressi (A e B) e un'uscita (U):

Per l'ingresso (A) è richesta l'immissione di un fattore moltiplicativo per la velocità di esecuzione dell'inviluppo; in (B) va indicato un fattore moltiplicativo per l'ampiezza della funzione stessa. In particolare:

per A = 1	si ottiene una velocità di esecuzione inalterata rispetto a quella fornita nell'editing di tabella
per A > 1	si ottiene una accelerazione di esecuzione pari al fattore fornito
per A < 1	si ottiene una decelerazione di esecuzione pari al fattore fornito;

analogamente:

per B = 1	si ha un'ampiezza della funzione inalterata rispetto alla tabella editata
per B > 1	si ha un incremento di ampiezza pari al fattore fornito
per B < 1	si ha un decremento di ampiezza pari al fattore fornito

Per una valida esemplificazione sull'uso del modulo ENV all'interno di un algoritmo, ci si può riferire al secondo degli esempi illustrati per il modulo OSC.

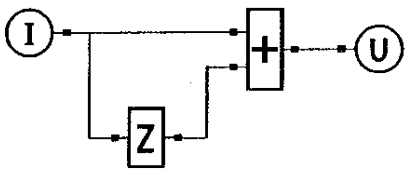
DLY

Modulo per la realizzazione di una linea di ritardo composta da una sola cella di memoria. Per un segnale posto all'ingresso (A) di questo modulo, si avrà dunque in uscita (U) l'identico segnale ritardato di 1 campione. Il suo simbolo grafico è rappresentato da:

Il "*valore associato*" per questo modulo indica il valore iniziale della memoria *delay*, ossia il valore da attribuire in partenza a quell'unica cella di memoria.

Il modulo **DLY**, così come i due successivi **DLN** e **DLS**, può essere utilizzato in vari modi: per realizzare segnali di controllo, per filtrare o reverberare segnali analogici, ecc.

Un utile impiego del modulo **DLY** può essere quello che lo vede utilizzato nella costruzione di un semplice filtro FIR del primo ordine:



in cui, essendo (I) il segnale in ingresso ed (U) il risultato in uscita del filtro, l'operazione svolta è del tipo:

$$U_n = I_n + I_{n-1}$$

DLN

La funzione svolta da **DLN** è analoga a quella del modulo precedente. La differenza consiste però nella possibilità di realizzare linee di ritardo composte da un numero di celle di memoria variabili da 1 a 2^n, vale a dire che il segnale posto all'ingresso (A) del modulo verrà ritardato di un numero di campioni in uscita (U), pari a (B). Il simbolo grafico di questo *delay multiplo* è:

Il "*valore associato*" al modulo è il valore di "**n**", ossia il numero massimo dei campioni di ritardo in termini di unità logaritmiche.

In (B) va dunque inserito un valore *floating point* (maggiore di 0 e minore di 2^n) corrispondente al numero di celle di memoria che si desidera effettivamente impiegare al momento.

(B) è legato al tempo di ritardo (in millisecondi) dalla seguente relazione:

$$T(ms) = \frac{Cm \cdot 1000}{Fc}$$

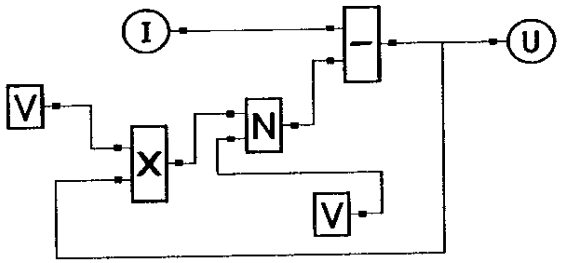
dove:

Cm = numero di celle di memoria che costituiscono la linea di ritardo;

Fc = frequenza di campionamento = 22050 per *default* in DIGITAL SIGNAL PATCHER.

Non volendo effettuare manualmente tale calcolo, l'utente può avviare il programma in una modalità introdotta proprio per lo scopo: la modalità "/T", selezionabile anche tramite comando **SWITCH** del menu **EDIT**: in tal caso, il valore in (B) va espresso in millisecondi.

Essendo il modulo DLN tipicamente impiegato nella realizzazione di echi, per esemplificare il suo uso si può fare riferimento al file ***Eco.pat*** già esaminato in occasione del comando SAVE (menu **FILE**):



in cui ad un segnale analogico viene, in modo continuo, sottratto il suo ritardo moltiplicato per un fattore di guadagno. L'entità di tale fattore (introdotto nel modulo VAL) deciderà del tempo di decadimento dell'eco stesso.

Nell'introdurre numerose o lunghe linee di ritardo all'interno di un patch, si tenga presente che, a causa delle caratteristiche hardware del superprocessore impiegato, non è possibile implicare più di 88 Kbytes di memoria RAM; vale a dire che non si possono realizzare linee di ritardo all'interno di un patch le quali, complessivamente, impieghino più di 90112 celle di memoria. Ad esempio, disegnando un algoritmo comprendente 2 moduli DLN per avere 2 linee di ritardo di identiche dimensioni, i rispettivi "*valori associati*" non potranno essere entrambi uguali a 16: infatti ciò comporterebbe l'uso di $2^{16} + 2^{16}$ celle di memoria = 131072 celle di memoria RAM.

DLS

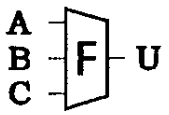
E' un *delay multiplo con incremento*. Svolge cioè funzioni analoghe a quelle dei due moduli precedenti, con in più la possibilità di introdurre in un secondo ingresso (B) un valore che rappresenta lo step incrementale per la lettura delle 2^n celle di memoria impiegate:

Evidentemente, se a (B) viene assegnato un valore pari a 1, il risultato in uscita (U) sarà identico a quello ottenuto con il modulo **DLN**. Se invece, ad esempio, sostituiamo nel patch ***Eco.pat*** (vedi sopra) il modulo **DLS** al modulo **DLN**, assegnando a (B) il valore 2, verrà letto solo un campione ogni due del segnale originario e si avrà in uscita un segnale echizzato con frequenza doppia e durata dimezzata rispetto all'originale.

Ricordiamo che, analogamente a quanto avviene per i moduli **OSC** e **TAB**, la funzione calcolata tramite **DLS** può essere generata senza che il programma effettui alcuna interpolazione; perché ciò avvenga bisogna lavorare con l'opzione **"/I"**.

FLT

Modulo per la realizzazione di filtri IIR di second'ordine (*all poles filters*). Vuole in ingresso (A) un segnale e negli ingressi (B) e (C) i parametri che rappresentano i coefficienti (a, b) del filtro; in uscita (U) si avrà il segnale filtrato. Simbolo grafico:



Il modulo **FLT** effettua il seguente calcolo:

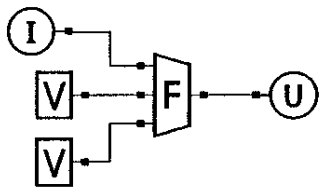
$$U_n = A_n + BU_{n-1} + CU_{n-2}$$

dove:

"n" rappresenta il numero d'ordine del campione.

Il *"valore associato"* al modulo **FLT** rappresenta il valore iniziale di ingresso del filtro, ossia il valore iniziale per (U):

Di seguito viene raffigurato l'algoritmo di un semplice filtro del secondo ordine:



perché **FLT** possa effettivamente rappresentare un filtro di secondo ordine, si tengano presenti le seguenti regole:

- il valore in (B) deve essere compreso tra -2 e +2;
- il valore in (C) deve essere compreso tra -1 e 0;
- inoltre: $B \leq 2 \cdot \sqrt{-C}$

Molti algoritmi noti, basati su tecniche di sintesi sottrattiva, utilizzano più filtri di 2° ordine posti in serie o in parallelo, a ciascuno dei quali viene affidata la gestione di un *formante* dello spettro del segnale da generare o elaborare. In tali algoritmi vengono ritenuti significativi parametri quali "Frequenza di risonanza" (F) e "Larghezza di banda" (BW) per operare su ciascun filtro, piuttosto che gestire direttamente i suoi due coefficienti (a, b) corrispondenti, nel nostro caso, agli ingressi (B) e (C) di **FLT**. La relazione che lega i coefficienti (a, b) del filtro di 2° ordine ai parametri F e BW, non è purtroppo semplice: ne indichiamo comunque la formula di calcolo:

$$B = 2e^{\frac{-2BW}{Fc}} \cdot \cos(\frac{2\pi F}{Fc})$$

$$C = -e^{\frac{-4BW}{Fc}}$$

dove:

B e *C* sono i rispettivi di **FLT**, equivalenti ai coefficienti (*a* e *b*);

Fc = frequenza di campionamento = *22050* per *default* in DIGITAL SIGNAL PATCHER (vedi anche il comando **FORMULAS** nel menu **HELP**).

RND

Modulo per la generazione di un segnale random.

Simbolo grafico:

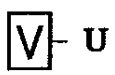
dove (U) è un segnale caotico con valori compresi tra -32768 e 32767.

Il "*valore associato*" da fornire tramite comando **VALUE** rappresenta il valore di innesco del rumore. Deve essere un numero in *floating point* tanto più complesso quanto più si desidera produrre un segnale vicino al *rumore bianco.*

VAL

Modulo per l'assegnazione di un valore costante. Tale valore è modificabile in tempo reale tramite comando **MODIFY**, dopo aver attivato il comando **PLAY**.

Simbolo grafico:

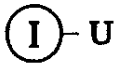


dove in uscita (U) si ha il valore costante da fornire come "*valore associato*" mediante comando **VALUE**.

INP

Modulo direttamente collegato al convertitore ADC per l'acquisizione digitale di segnali analogici audio provenienti dall'esterno del DIGITAL SIGNAL PATCHER.

Il simbolo grafico del modulo input è:

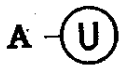


dove (U) rappresenta il segnale proveniente dal convertitore ADC. Il "*valore associato*" deve essere : **"1"** o **"2"** (se si lavora con una sola scheda di conversione), oppure sarà compreso tra **"1"** e **"4"** (se si lavora con due schede in quadrifonia) e sta ad indicare il numero di canale di acquisizione ADC.

OUT

Modulo di uscita del segnale processato con DIGITAL SIGNAL PATCHER, direttamente collegato con il convertitore DAC per l'ascolto del suono. E' tipicamente il modulo terminale di qualsiasi patch DSP; dispone quindi solo di connessione di ingresso (A).

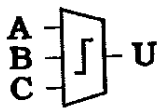
Il simbolo grafico del modulo output è:



Come per il modulo **INP**, il "*valore associato*" è il numero di canale di conversione, in questo caso del convertitore DAC; anche per **OUT** i valori consentiti sono: "**1**" o "**2**" in stereofonia, da "**1**" a "**4**" in quadrifonia.

IFP

IFP è un *selettore logico*: vale a dire che, a seconda del valore assunto dal primo dei suoi tre ingressi (A) rispetto ad un valore soglia (T), esso determinerà per l'uscita (U) il contenuto dell'ingresso (B) oppure il contenuto dell'ingresso (C):

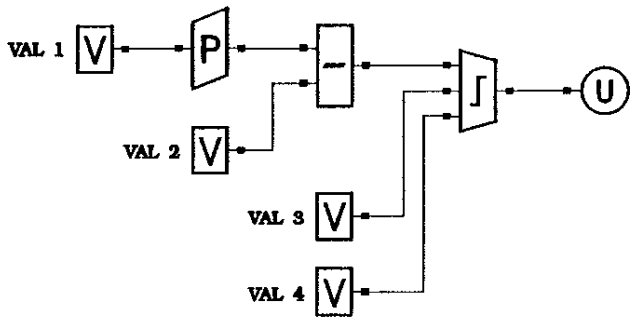


l'operazione effettuata è la seguente:

se $A \geq T$ allora $U = B$ altrimenti $U = C$

il "*valore associato*" da fornire con VALUE è proprio il valore di soglia (T).

L'uso di IFP può essere esemplificato esaminandolo all'interno di un algoritmo generatore di impulsi con *duty cycle* variabile:



Per commentare l'algoritmo illustrato, denominiamo i moduli VAL implicati, rispettivamente: VAL1, VAL2, VAL3, VAL4 e associamo al modulo IFP il valore soglia (T) = 0. Come si vede, all'ingresso (A) del modulo IFP viene collegato un modulo fase (PHS), che sappiamo produce in uscita un segnale rampa i cui estremi oscillano tra 0 e 65535. A tale segnale viene sottratto VAL2, la cui funzione vedremo tra poco.

All'ingresso (B) viene associato VAL3 cui daremo valore = 0, mentre a VAL4, collegato all'ingresso (C), forniremo valore = 1. Ora, ponendo VAL2 = 0, avremo in (A) esclusivamente valori positivi compresi tra 0 e 65535: vale a dire che IFP selezionerà per l'uscita sempre e solo (B) = VAL3 = 0.

Se invece aumentiamo VAL2 ponendolo, ad esempio, a 21845, si avrà in (A) una rampa che oscilla tra -21845 e 43690 producendo, per un terzo del suo periodo, valori negativi; in concomitanza di questi ultimi IFP selezionerà per l'uscita l'ingresso (C) = VAL4 = 1, generando quindi un impulso ogni periodo, di durata equivalente ad un terzo dell'intero ciclo. In altri termini si è prodotto così un *duty cycle* del 33%. Analogamente, per VAL2 = 32767 si avrà un *duty cycle* del 50% mentre, per VAL2 = 65535, un *duty cycle* del 100%.

6. Menu HELP

GENERAL

Descrive in generale il funzionamento di DIGITAL SIGNAL PATCHER.

KEYWORDS

Riassume in due tabelle i comandi da tastiera di DIGITAL SIGNAL PATCHER.

KEYBOARD

Riassume i principali comandi relativi ai tasti alfanumerici.

FORMULAS

Riassume le formule presenti in DIGITAL SIGNAL PATCHER e utilizzabili dall'utente.

Table reading moduls (letture tabellari)

Freq/Fc = Step/TabLeng

Freq/Fc = Phase/2Π

dove:

Freq = frequenza

Fc = frequenza di campionamento

TabLeng = lunghezza della tabella (4096 campioni).

Esprimono rispettivamente la relazione tra frequenza e step incrementale e tra frequenza e fase.

Second order filters (filtri del secondo ordine)

$-1 < C < 0;\ |B| < 2\sqrt{(-B)}$

Esprimono l'intervallo di stabilità della funzione per i valori di C e B rispettivamente.

$M = \mathrm{Exp}(-2Bw/Fc)$

$B = 2M\mathrm{Cos}(2\Pi Fr/Fc)$

$C = -M^2 = -\mathrm{Exp}(-4Bw/Fc)$

$M = \mathrm{Sqrt}(-C)$

$Fr = Fc/2\Pi * \mathrm{ArcCos}(B/2M)$

$Bw = -Fc/2 * \ln(M)$

dove:

M = variabile d'appoggio

B = coefficiente del primo ordine del filtro

C = coefficiente del secondo ordine del filtro

Bw = banda passante

Fr = frequenza di centrobanda

Fc = frequenza di campionamento.

Le prime tre formule esprimono il passaggio da frequenza e banda passante ai coefficienti B e C , le ultime tre il passaggio inverso cioè dai coefficienti B e C alla frequenza e alla banda passante.

Delay memory/time (numero di celle di ritardo/tempo)

Time(ms) = Mem*1000/Fc

dove:

Mem = numero di celle di memoria

Fc = frequenza di campionamento.

La formula serve per conoscere il tempo di ritardo espresso in millisecondi a partire dal numero di celle di memoria.

MODULES

Informa in modo conciso sul comportamento di tutti i moduli utilizzati in DIGITAL SIGNAL PATCHER.

Add: U = A+B

Somma: l'uscita U è uguale alla somma dei due ingressi A e B.

Sub: U = A-B

Sottrazione: l'uscita U è uguale alla differenza diegli ingressi A e B.

Mul: U = A*B

Moltiplicazione: l'uscita U è uguale al prodotto degli ingressi A e B.

Phs: U = U+A

Fase: l'uscita U è uguale alla somma (modulo 65536) dell'uscita precedente U con l'ingresso A.

Tab: U = Twf[A]

Tabella: l'uscita U legge i valori della forma d'onda all'indirizzo A.

Mem: Twf[B] = A

Memoria: all'indirizzo B della tabella viene scritto il valore A.

Osc: P = P+A

U = BTwf[P]

Oscillatore: la prima formula determina la fase P (fase = fase + step) di modulo 4096, la seconda invece determina l'uscita U che risulta uguale all'ampiezza B per il valore della tabella all'indirizzo P.

Env: U = BTen[A]

Inviluppo: L'uscita U è uguale al prodotto dell'ampiezza B per la tabella inviluppo *Ten* controllato in velocità da A.

Dly: U = Aprec

Ritardo di un campione: l'uscita U risulta uguale all'ingresso precedente.

Dln: U = Ak(B)prec

Ritardo di n campioni: l'uscita U risulta uguale all'ingresso B volte precedente.

Dls: U = Akprec(B)

Ritardo di n campioni controllabile in velocità: l'uscita U risulta uguale all'ingresso k volte precedente, B è il controllo di velocità.

Flt: U = A+BUp+CUpp

Filtro del secondo ordine: l'uscita U risulta uguale alla somma dell'ingresso A, di B volte l'uscita precedente Up e di C volte l'uscita due volte precedente Upp.

Rnd: U = Random

Valore casuale: l'uscita U ha un valore casuale compreso tra -32768 e +32767.

Val: U = Value

Valore determinato: l'uscita U ha un valore assegnato.

Inp: U = Adc(Chn)

Ingresso: l'uscita U assume il valore preso da uno dei canali ADC (analogico).

Out: Dac(Chn) = A

Uscita: ad A viene assegnato uno dei canali DAC (digitale).

Ifp: If A>thrs
then U = B
else U = C

Selettore: Se l'ingresso di selezione A è maggiore di una certa soglia *thrs* allora l'uscita U sarà uguale all'ingresso B altrimenti assumerà il valore dell'ingresso C.

APPENDICE

Eventuali problemi ed errori

Nel caso si verifichino i seguenti inconvenienti all'avvio o durante il funzionamento di DIGITAL SIGNAL PATCHER, elenchiamo per ciascuno di essi le possibili cause ed indichiamo le eventuali soluzioni:

1. Il programma "DSP.EXE" non viene eseguito:

- la scheda su cui è installato il superprocessore DSP è mancante;
- il programma è stato installato in modo scorretto;
- i files necessari al funzionamento del programma sono mancanti;
- si sta utilizzando una copia non autorizzata del programma.

Eseguire il programma "RUN.BAT" e leggere il messaggio di errore che verrà visualizzato sul monitor.

2. La scheda DSP è presente ma non viene riconosciuta dal sistema *Fly30*:

- la scheda DSP non è inserita correttamente;
- la scheda DSP non è stata inizializzata.

3. Non viene emesso alcun suono all'atto del PLAY:

- la scheda DSP non è stata inizializzata;
- l'algoritmo in uso non produce o non prevede segnale audio;
- non sono state caricate le tabelle per inviluppi o per forme d'onda;
- le connessioni sulla scheda ADC/DAC sono errate.

Nel primo caso uscire da Digital Signal Patcher ed eseguire il programma "INIT.BAT"; negli altri casi ricontrollare le operazioni effettuate prima di reinizializzare la scheda.

4. La compilazione del patch di lavoro produce errori:

- non è presente il programma *Assemblatore* e *Linker* TMS320C30;
- L'*Assemblatore* e *Linker* TMS320C30 non è nel "PATH" corretto.

Installare il programma *Assemblatore* e *Linker* del TMS320C30 in una directory visibile dal sistema operativo DOS.