Consiglio Nazionale delle Ricerche

ISTITUTO DI ELABORAZIONE DELLA INFORMAZIONE

PISA

Apparati per la sintesi di segnali audio

Tecniche per la gestione in tempo reale del Sintetizzatore Audio Digitale MP3A

G.Bertini, P.Chiodaroli, L.Tarabella, S.Toni

Nota interna B4-62
Dicembre 1988

Stampato in proprio dal Servizio Tecnografico
dell'Istituto di Elaborazione della Informazione - CNR - Pisa

Apparati per la sintesi di segnali audio

"Tecniche per la gestione in tempo reale del Sintetizzatore Audio Digitale MP3A".

G.Bertini, P.Chiodaroli, L.Tarabella, S.Toni

Nota Interna B4-64
Dicembre 1988

Indice

Introduzione

In questa nota vengono descritte le tecniche di gestione di un modulo denominato **MP3A** atto alla sintesi numerica di segnali audio in tempo reale basato sul microprocessore TMS32010 (DSP) prodotto dalla Texas Instruments, e l'interfaccia al bus di un PC-IBM (XT, AT o compatibile). Il modulo MP3A come verrà illustrato svolge il suo compito in modo completamente autonomo ("stand-alone"). Quindi mediante l'impiego di più schede MP3A è possibile costituire un sistema di sintesi polifonico in cui, come nell'esempio mostrato in figura 1, ciascuna si occupa della sintesi di una "voce musicale". L'interfacciamento con l'Host, del resto molto semplice, permette infatti un agile controllo di un numero elevato di schede MP3A.

La scheda MP3A, realizzata nell'ambito della collaborazione tra il Reparto di Informatica Musicale del CNUCE (Pisa) e il Reparto di Elaborazione di Segnali e di Immagini dell'IEI (entrambi Istituti del CNR), è la naturale evoluzione di una precedente versione (MP3) già descritta in una nota interna CNUCE {1}, alla quale occorre fare riferimento per avere informazioni sull'architettura di base della scheda e sulle modalità di utilizzo tramite un Commodore C'64.

Infine sarà data la descrizione dettagliata dei meccanismi software implementati per ottenere l'esecuzione di un algoritmo complesso di sintesi audio .

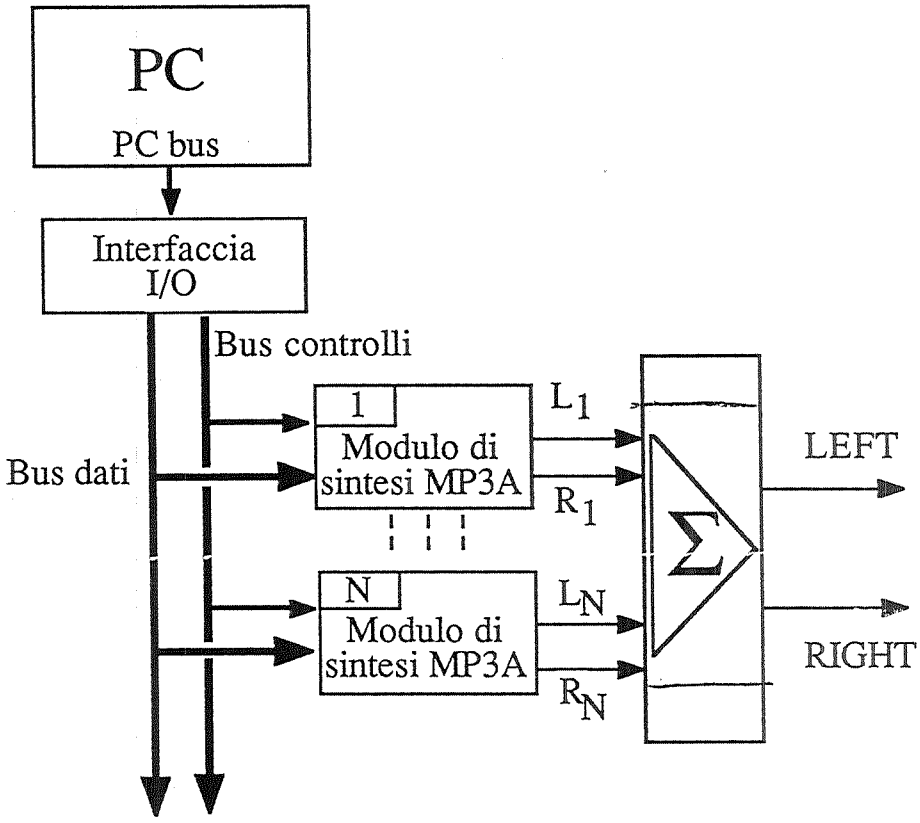


Fig. 1 - Sistema di sintesi polifonico con N moduli MP3A

1 - Richiami alle tecniche di implementazione della sintesi digitale del suono.

In generale il processo di generazione del suono (sintesi) può essere pensato scomposto in due sottoprocessi distinti: la realizzazione di "generatori audio", capaci di fornire segnali complessi con componenti comprese nella gamma di frequenze fra i 20 Hz e i 20 KHz, e la realizzazione dei "generatori di inviluppo", le cui uscite vengono utilizzate per controllare in vario modo i segnali prodotti dai generatori audio , con cadenze evolutive più basse.

Per quel che riguarda i metodi per l'implementazione dei generatori audio si può dire che fondamentalmente fanno uso di "oscillatori digitali" basati sul ben noto metodo della "lettura tabellare"{2},{3}; Nel seguito per evidenziare i problemi che si incontrano nella programmazione di dispositivi DSP verrà illustrato a titolo d'esempio come è stato realizzato un generatore audio che sfrutta, al fine di produrre un segnale dallo spettro particolarmente ricco, la tecnica di sintesi basata sulla Modulazione di Frequenza (FM){4}.

Usando la FM, ma anche con altre tecniche, per produrre i valori numerici da applicare ai convertitori D/A viene eseguita ciclicamente una routine con frequenza pari alla *frequenza di campionamento*. Pertanto, se si desidera ottenere segnali con contenuto frequenziale tipico dell'HI-FI, è necessario produrre i campioni numerici ad un tasso di almeno 32000 al secondo (cioè con una frequenza di campionamento di almeno 32KHz); ne risulta che il tempo massimo per l'esecuzione dell'algoritmo è di circa **30 µsec.**

Per quanto riguarda invece un generatore di inviluppo, il suo scopo è quello di "modulare" il valore di alcuni parametri di un oscillatore audio come ad esempio l'ampiezza di un singolo oscillatore audio, *l'indice di modulazione* (I) della già citata tecnica FM, oppure il volume del segnale in uscita.

In ogni caso, per qualunque parametro di inviluppo, è ampiamente dimostrato dall'esperienza che è sufficiente aggiornare il valore ad intervalli temporali dell'ordine di **1 millisecondo**; la frequenza di aggiornamento dei parametri è detta spesso *frequenza di frame*.

La generazione degli inviluppi per via digitale può essere realizzata in maniera sostanzialmente simile alla generazione algoritmica di segnali audio; in questo caso, essendo più lenta la frequenza di lavoro, oltre alla tecnica della scansione tabellare è possibile anche generare i campioni dell'inviluppo

direttamente a partire da un determinato calcolo matematico, ad esempio incrementando, ad ogni intervallo di frame, il valore del parametro da "inviluppare" fino ad un certo massimo, mantenendolo poi per un certo intervallo di tempo e quindi decrementandolo nuovamente e così via in modo ripetitivo.

Da quanto premesso,si vede che siamo di fronte a due processi concorrenti che, negli apparati di sintesi sviluppati in precedenza, venivano eseguiti da due distinte unità di elaborazione. La soluzione adottata nella scheda MP3A permette ai due processi di essere portati avanti dalla stessa unità di calcolo, basata appunto sul TMS32010.

Come avviene di solito, per realizzare questa concorrenza , si ricorre al meccanismo delle interruzioni di programma. La differente cadenza temporale che caratterizza i due processi suggerisce di sfruttare questa tecnica eseguendo un processo in "background" nel programma principale, e l'altro durante i driver attivati ad ogni interruzione.

Naturalmente al processo che ha durata minore e tempi più stretti, cioè quello di sintesi dei campioni del segnale, si dà la priorità maggiore, e si fà quindi eseguire dal driver di interruzione. La generazione degli inviluppi, invece, viene effettuata all'interno del programma principale,in modo distribuito nel tempo, in intervalli di ampiezza opportuna, che devono essere lasciati liberi dal processo di sintesi. La cadenza temporale del processo di generazione degli inviluppi è ottenuta tramite un meccanismo software di sincronizzazione con il processo a priorità più alta, che è periodico. Nei paragrafi seguenti verrà descritta l'implementazione di tale meccanismo.

2 - Architettura e funzionamento del modulo di sintesi MP3A

L'MP3A come vedremo meglio in seguito funziona secondo questi principi: una volta caricati programmi e tabelle per il calcolo dei campioni, può essere considerata come un modulo "stand alone" e riceve dall'Host solo i dati riguardanti i parametri tipicamente musicali come l'altezza dei suoni (frequenza della nota musicale), tipo di timbrica (indice di modulazione), effetti spaziali, ecc. Naturalmente la cadenza per l'invio di questi dati varia dal tipo di applicazione, cioè dal "genere" di brano musicale che l'Host sta trattando: in effetti anche questo può essere considerato un terzo processo che si svolge in concorrenza con gli altri due. Tenendo conto però della frequenza massima di invio (qualche centinaio di hertz) e delle caratteristiche dell'apparato uditivo son tollerati sfasamenti temporali "locali" anche di frazioni di millisecondo.

Lo schema con i componenti e i blocchi funzionali principali sia della scheda di sintesi che dell'interfaccia con il PC è riportato in fig.2; i componenti dell'interfaccia sono montati in una scheda separata inserita in uno degli slot disponibili all'interno del PC.

Sostanzialmente il modulo di sintesi è composto da:

- un microprocessore TMS320/10 della prima generazione dei dispositivi DSP della Texas Instruments. La scelta di tale componente è già stata discussa in precedenti lavori. Ricordiamo solo che la maggior parte delle istruzioni vengono eseguite in 200 ns, compreso quella della moltiplicazione di 16x16 bits; possiede una ram interna di 164 word e può indirizzare 4kw di ram esterna.
- una ram 4064 word di 16 bits per programmi, dati e tabelle.
- una rom di 32 word di 16 bits che contiene il programma di boot-strap per il caricamento di programmi e dati dall'Host verso la scheda di sintesi.
- due porte di uscita a 16 bit verso i D/A.
- due porte di ingresso per il passgio dei dati da Host verso la MP3A.
- un insieme di circuiti per la logica di controllo di tutti i componenti della scheda.

Si fa notare che rispetto alla precedente versione della scheda (MP3), sulla nuova versione MP3A è stato introdotto il reset via software ed attivato l'interrupt sul TMS320/10, supportati da due appositi registri di sincronizzazione.

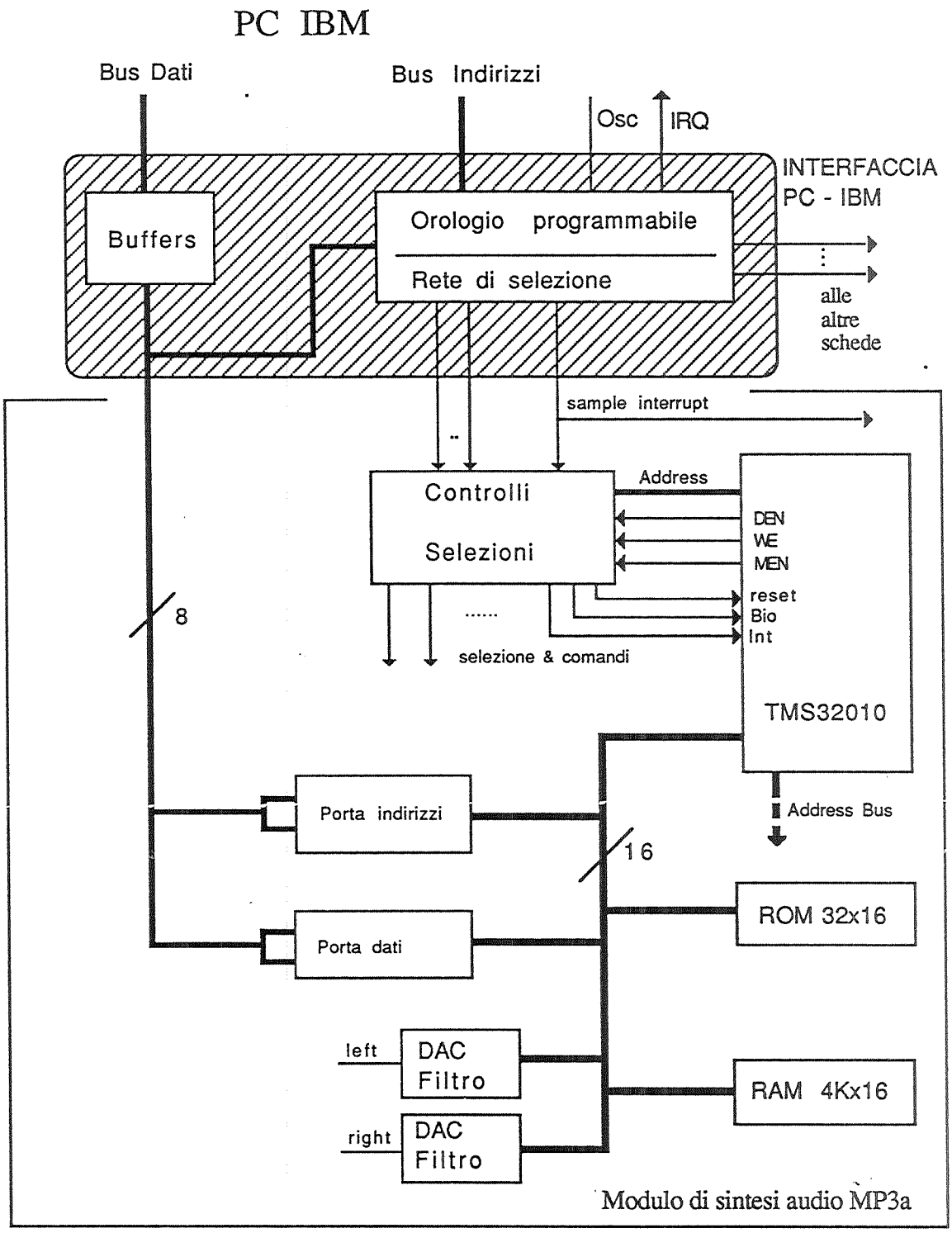


Fig.2 Schema a blocchi di un modulo MP3A.

Tramite l'uso dell'interrupt è possibile dare inizio e far eseguire il programma (caricato in fase di preparazione nella ram della scheda) che calcola l'iesimo campione numerico (passo iesimo del processo di sintesi) in base ai valori dei parametri forniti saltuariamente dall'Host.

Dal momento che il campione numerico deve pervenire al D/A ad intervalli rigidamente prefissati è utile avere un orologio hardware (opportunamente programmato) che fornisce il segnale di attivazione dell'interrupt stesso.

Tale dispositivo è stato implementato con il circuito 8254 della Intel, posto sulla scheda di interfaccia, e che ha al suo interno tre timer/counter indipendenti; l'uscita di uno di questi è collegata in modo da effettuare automaticamente una scrittura nel registro di interruzione ogni volta che termina una sequenza di conteggio. Ciò consente di disporre di una sorgente di sincronizzazione per il software che viene eseguito sulla scheda, necessaria per il tipo di gestione che si è implementato, come vedremo in seguito.

Poichè il timer determina l'istante in cui ha inizio l'esecuzione del driver di interrupt può risultare un riferimento di accuratezza sufficiente a garantire la costanza della frequenza di produzione dei campioni nella sintesi audio. Il Timer è programmabile a piacere via software e permette di ottenere a regime una esatta temporizzazione per via hardware e non mediante un software loop. Questa soluzione era già stata proposta in {5} e si era dimostrata funzionante: comunque la gestione dell'interruzione con il TMS32010 comporta l'adozione di alcune precauzioni che verranno in seguito discusse nei dettagli.

L'introduzione del reset del microprocessore via software permette all'Host di interrompere l'esecuzione di un programma sulla scheda per caricare, ad esempio, un nuovo algoritmo di sintesi.

Infine sono stati tolti i registri di ritorno dalla scheda verso il PC poichè è stata giudicata eccessivamente dispendiosa la realizzazione della logica di controllo necessaria per poter implementare una interfaccia bidirezionale completa non ritenuta necessaria per i nostri scopi.

3 - L'interfaccia fra il PC-IBM ed il sistema di sintesi.

L'interfaccia è stata progettata con criteri di generalità in modo da rendere possibile l'adattamento al bus del PC di altri tipi di schede con differenti microprocessori. Infatti è stata usata anche per pilotare una scheda con il TMS320C25 {6}. Lo schema logico è riportato in figura 3. Fisicamente la scheda viene inserita in uno slot del PC-IBM dal quale prende l'alimentazione per gli integrati TTL utilizzati. I componenti principali sono:

- i buffers per i dati e gli indirizzi e i decodificatori per la generazione delle linee di selezione e dei segnali di "strobe" per scrivere nei registri attraverso cui è visibile la scheda . Tali registri sono mappati nello spazio di I/O del PC.
- il comparatore 74LS682 che esegue una funzione di confronto con la quale si rende facilmente "rilocabile" lo spazio di I/O occupato, tramite il settaggio di appositi "microinterruttori" che ne determinano la "maschera".
- l'orologio programmabile (il già citato timer/counter Intel 8254).

Per quest'ultimo componente si devono aggiungere altre informazioni oltre a quanto già detto nel paragrafo precedente. I registri mediante cui viene programmato il funzionamento di questo integrato sono stati mappati nello spazio di I/O del PC. Gli ingressi di due dei tre timer/counter, che si trovano nell'8254, sono collegati all'uscita di un flip flop che divide per due la frequenza del segnale "OSC" proveniente dal bus del PC. Si è utililizzato questo segnale, e non quello di clock della scheda di sintesi, perchè ha la stessa frequenza sia sullo slot del PC-XT che su quello del PC-AT e dei vari "compatibili" mentre la frequenza dei segnali di clock può variare a seconda del tipo di scheda. È stato necessario dividere per due la frequenza di OSC (originariamente di circa 14.31818 MHz) per riportarla nei limiti operativi dell'Intel 8254. L'uscita di uno dei timer (Out0) che stiamo considerando è portata verso la scheda di sintesi ed è collegata in modo da "settare" il registro delle interruzioni verso il TMS32010 ogni volta che scade un periodo di conteggio. L'uscita dell'altro timer (Out1) è invece portata come ingresso al terzo, così da poter ottenere intervalli di tempo piuttosto lunghi. L'uscita di quest' ultimo timer (Out2) viene infine collegata allo slot del PC (IRQx) in

SLOT PC IBM

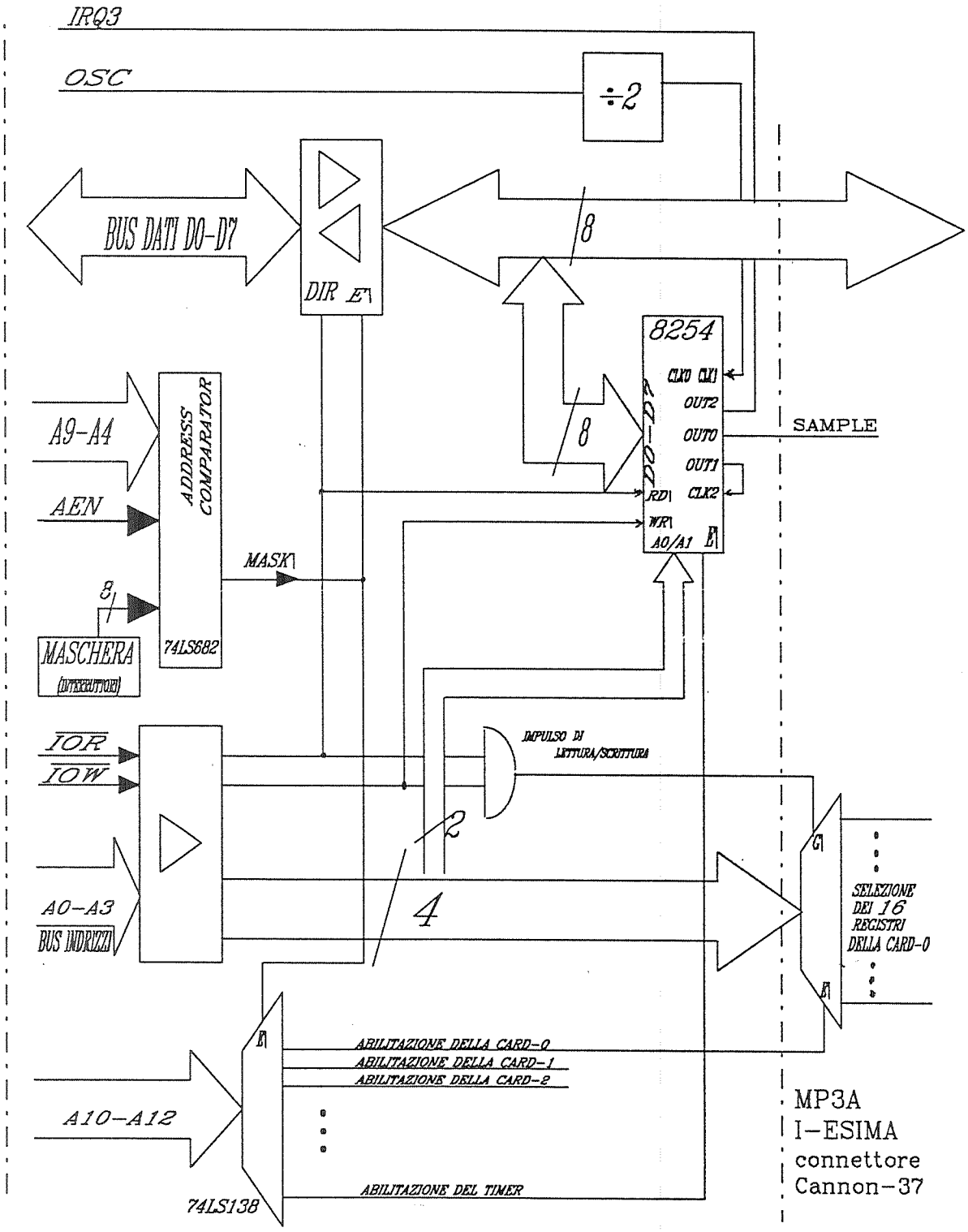


Fig.3 Schema logico dell'interfaccia con il bus dell'Host.

modo da poter generare una richiesta di interruzione anche verso il microprocessore dell'Host Computer. Questo nel caso che anche su di esso ci sia la necessità di temporizzare con precisione degli eventi (ad esempio nel caso di gestione degli inviluppi,o anche solo di una partitura musicale) da parte dell'Host Computer.

È stata prevista, nella realizzazione dell'interfaccia, la possibilità di gestire più schede di sintesi contemporaneamente: tuttavia l'interfaccia occupa solo 16 porte dello spazio di I/O del PC poichè si sono sfruttate le linee di indrizzi da A10 ad A15 disponibili sul bus. L'uso di queste linee permette teoricamente di collegare fino a 64 schede MP3A all'Host senza ulteriore occupazione di spazio di I/O.

La particolare struttura dell'interfaccia permette una gestione del colloquio con l'Host estremamente efficiente per il TMS32010, ossia senza sottrarre tempo significativo alle elaborazioni in tempo reale da supportare, condizione questa, assolutamente indispensabile per le frequenze di lavoro di nostro interesse.

4 - Tecnica di gestione dell'inviluppo.

Il programma riportato come esempio utilizza, anche per la generazione dell'inviluppo, il metodo della scansione tabellare. Il modello di inviluppo a cui si fa riferimento è una versione leggermente semplificata del tipico ADSR (Attack, Decay, Sustain, Release) utilizzato da molte tastiere e sistemi commerciali. La semplificazione consiste nel fatto di aver considerato come un unica fase l'insieme delle fasi di Attack e Decay e di aver considerato costante il livello di uscita nella fase di Sustain. In pratica la routine dispone di una tabella in cui sono immagazzinati dei valori che descrivono l'andamento dell'inviluppo (figura 4). Quando la routine riceve l'informazione che deve iniziare una nota (a questo proposito dall'esterno viene "settata" una variabile booleana detta "gate" che ad esempio rappresenta la situazione di "tasto premuto/tasto alzato" sulla tastiera di un organo), inizia la scansione della tabella con una certa "velocità", cioè con un certo valore dell'incremento che ad ogni passo viene sommato al puntatore alla tabella (fase di Attack-Decay). Giunta ad un certo punto della tabella, detto "nodo"(vedi la stessa figura), la

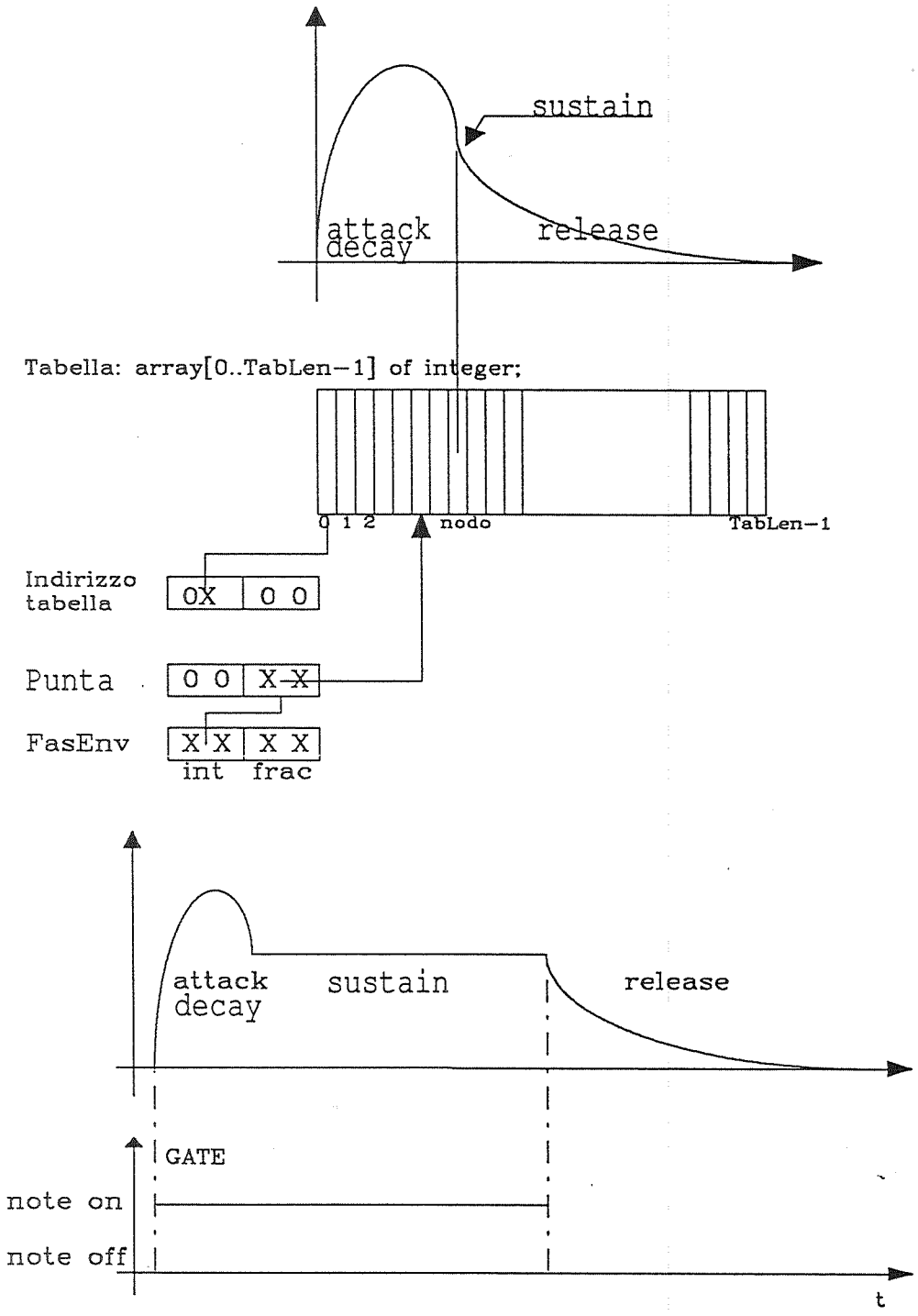


Fig.4 Meccanismo di generazione dell'inviluppo ADSR.

scansione si ferma ed il valore dell'inviluppo viene mantenuto costante (fase di Sustain) fino a che non giunge l'informazione che la nota è finita, cioè la variabile "gate" viene resettata. A questo punto la scansione riprende (fase di release), con velocità, in generale, diversa da quella con cui è stata effettuata la prima fase, fino a raggiungere la fine della tabella, superata la quale l'inviluppo viene mantenuto all'ultimo valore letto in tabella (che può, per inciso, essere anche diverso da zero).

Dall'esterno si può esercitare un certo controllo su questo processo modificando il valore di alcune variabili che, per la loro caratteristica di essere visibili anche all'Host Computer, sono dette "globali".

Come già detto esiste una prima variabile globale chiamata "GATE", mediante la quale l'Host controlla l'istante di attacco della nota e l'istante in cui ha inizio il rilascio. Oltre a questa, sono state definite le variabili globali: "NODO","ATTSPE" e "RELSPE" che consentono rispettivamente di variare la posizione del punto in cui la tabella, utilizzata per la generazione dell'inviluppo, viene suddivisa nelle fasi di ATTACK-DECAY e di RELEASE e la velocità con cui vengono scandite le due parti della tabella.

Oltre a queste variabili condivise con l'Host, la routine di generazione dell'inviluppo fa uso anche di alcune variabili locali. Esse sono: RELEAS, SUST e PAUSA che assieme con GATE consentono di stabilire in quale fase della scansione della tabella ci si trova.

La tabella dei valori dell'inviluppo è, nel programma illustrato, un' array di lunghezza TABLEN (costante e pari a 256). L'indice mediante il quale si individuano gli elementi dell'array è la variabile PUNTA; poichè il TMS32010 accede solo a word (16 bit) anche PUNTA è su 16 bit: gli 8 MSB sono sempre degli zeri. La fase di scansione della tabella è FASENV i cui 8 MSB rappresentano la parte intera mentre gli 8 LSB sono considerati frazionari così da poter avere incrementi non interi. Da FASENV si passa a PUNTA semplicemente prendendo gli 8 MSB, per passare da PUNTA all'indirizzo fisico si appendono in testa altri 4 bit che rappresentano l'indirizzo iniziale della tabella. Questo consente di scandire contemporaneamente più tabelle: è sufficiente per questo concatenare a PUNTA indirizzi di tabelle diverse. È possibile quindi "inviluppare" più parametri, con la sola limitazione che per tutti la scansione avviene con la medesima fase. Nel caso che più parametri abbiano andamento simile ma su scala diversa si è introdotta la moltiplicazione

degli inviluppi per un fattore di scala senza dover ricorrere a tabelle diverse. Il compito di PUNTA e FASENV vengono ulteriormente chiariti dalla figura 4, mentre il diagramma di flusso della routine che realizza l'inviluppo ADSR, implementato poi in Assembler TMS32010 è riportato in figura 5.

Seguitiamo ora con delle considerazioni temporali. Supponendo di fissare un tipico intervallo di campionamento TC =31 microsecondi, lavorando in assembler TMS320/10 si riescono a realizzare all'incirca 7 o 8 oscillatori audio più qualche altra operazione aggiuntiva ; inoltre si può supporre di dover gestire al massimo una decina di inviluppi contemporaneamente.

Per generare un singolo valore di un inviluppo occorrono circa 7-8 microsecondi (con l'algoritmo illustrato e in assembler TMS32010), ai quali bisogna sommare un altro microsecondo per ogni ulteriore parametro;.quindi occorrerebbero un centinaio di microsecondi per aggiornare i valori di tutti gli inviluppi. Non è corretto interrompere saltuariamente e far shiftare il processo di generazione dei campioni per un tempo così lungo, pena l'introduzione di rumore.

.Perciò si è adottata la soluzione di rinunciare ad un piccola parte del tempo riservata all'algoritmo di sintesi (in pratica si perde all'incirca l'equivalente di un oscillatore) e di sfruttare tale parte per il calcolo degli inviluppi, da svolgersi così a "fette di tempo" e completarsi in intervalli, come abbiamo già detto, al massimo di circa 1 millisecondo.

Con riferimento alla fig.6, tenendo conto delle considerazioni fatte sopra, si deduce che viene lasciato un tempo di circa 28 microsecondi all'algoritmo di sintesi (Td), equivalente alla durata del driver di interruzione e di un intervallo di circa 3 microsecondi per il calcolo degli inviluppi (Tc - Td).

D'altra parte proprio la differente cadenza temporale che caratterizza i due processi, come già anticipato nel par.1, suggerisce di sfruttare il meccanismo di interruzione del microprocessore per eseguire il processo di sintesi con la massima priorità e a sincronizzare al tempo stesso la routine di gestione degli inviluppi all'interno del programma principale che viene così eseguita in "background".

Per generare gli intervalli di "frame", cioè gli intervalli durante i quali l'inviluppo rimane costante, viene utilizzato un timer "software" attivato dalla sola sorgente di temporizzazione di cui disponiamo (l'orologio che invia le interruzioni con periodo Tc).

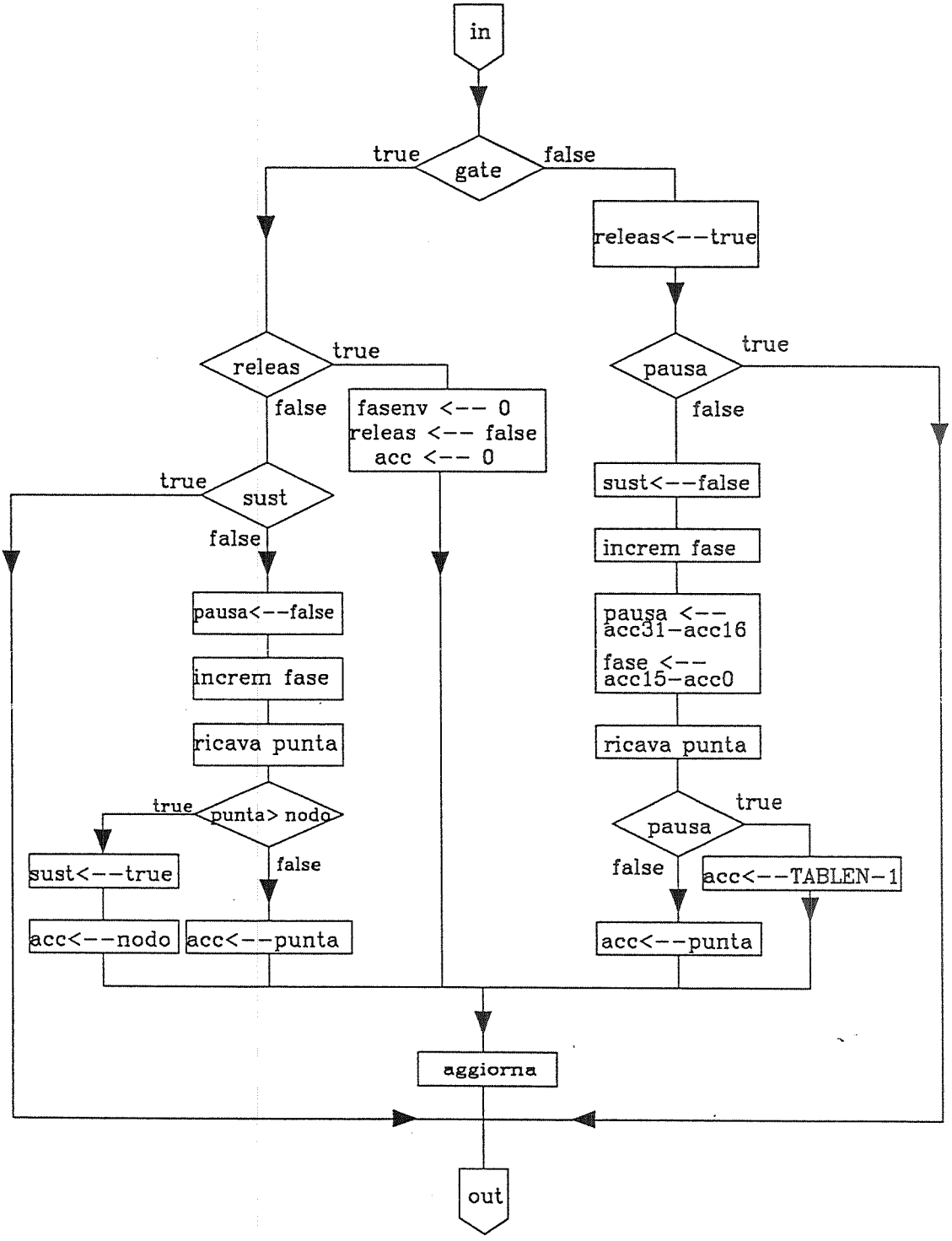


Fig.5 Diagramma di flusso della routine che realizza l'inviluppo ADSR.

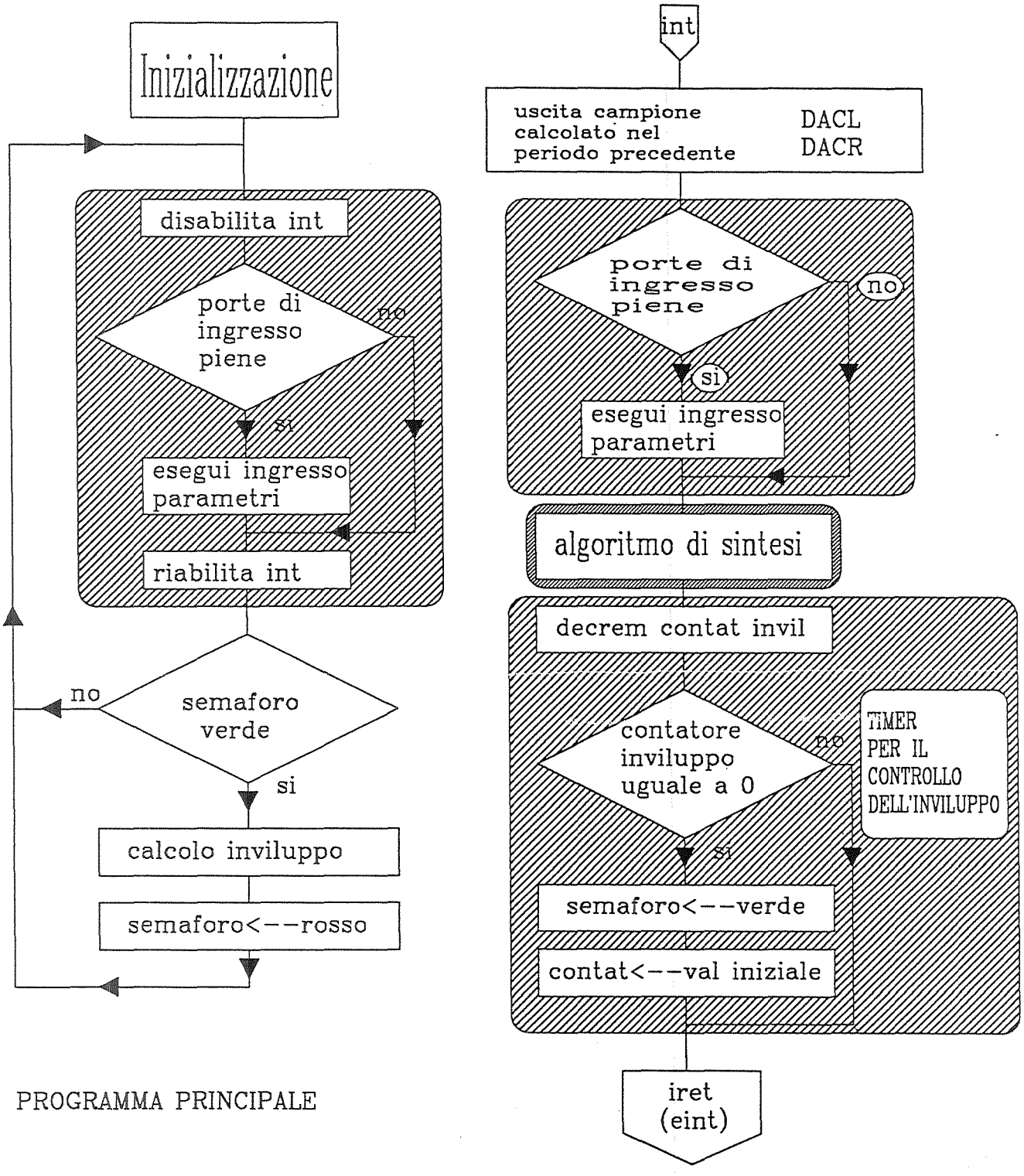


Fig.6 Diagrammi di flusso del programma principale (gestione inviluppi) e del driver di interruzione (algoritmo di sintesi).

Utilizziamo a questo proposito un contatore il cui conteggio iniziale viene decrementato ad ogni intervallo di campionamento (cioè ogni volta che va in esecuzione il driver di interruzione su comando dell'orologio hardware): quando il contatore raggiunge lo zero viene messo verde un semaforo. La costante che viene caricata ogni volta come conteggio iniziale dell'orologio è contenuta nella variabile globale QUANTI. Il programma principale, che si era fermato al semaforo rosso, riparte e genera un nuovo valore per ogni parametro inviluppato, dopo di che rimette a posto il semaforo (di nuovo rosso) e riparte da capo.

Per la generazione degli inviluppi, il Programma dispone dell'intervallo di tempo QUANTI*Tc. Il programmatore deve fare in modo che QUANTI*Tc sia all'incirca 1ms. Nel caso che il valore di QUANTI venga diminuito (aumentato) naturalmente si deve controllare se non sia necessario diminuire la durata del driver di interruzione (togliendo istruzioni) oppure nel caso opposto, se ciò è desiderato, aggiungere delle istruzioni.

Il nuovo meccanismo di generazione del suono adottato nella scheda MP3A risulta molto efficiente anche nei confronti del problema del rinnovo parametri da Host. Infatti il programma che gira sull'Host invia i valori delle variabili globali (GATE, Frequenze, indici di modulazione ecc.) con il solo vincolo di attendere un intervallo maggiore di Tc fra l'invio di due successivi parametri. Occorre perciò "testare" il BIO all'interno del driver di interruzione così che continui a valere l'ipotesi su cui si basa il pilotaggio da parte dell'Host. La routine di servizio per l'ingresso dei dati da Host può anche essere posta nel programma principale se si vuole avere la possibilità di spedire dei parametri anche quando non si ha generazione di campioni e l'orologio che genera le interruzioni è fermo.

In generale si deve osservare che la risorsa "BIO" (flag dello stato delle porte di I/O con l'Host) è condivisa dai due processi, pertanto il test sullo stato di questo flag e l'eventuale aquisizione di nuovi valori deve essere svolta ad interruzioni disabilitate (sezione critica) se eseguita nel programma principale (il driver di interruzione viene eseguito in modo non interrompibile).

La disabilitazione delle interruzioni nel programma principale può, in generale, comportare saltuariamente, ed in maniera imprevedibile, una deviazione dell'istante di uscita dei campioni verso i DAC rispetto al caso ideale in cui la frequenza di campionamento è rigorosamente prefissata.

L'inconveniente fortunatamente non è frequente e la deviazione, che può essere valutata in base alla durata delle sezioni critiche nel programma principale, non può essere apprezzata dal nostro apparato uditivo. La stessa architettura del TMS32010 comporta l'introduzione di altre sezioni critiche ben più rilevanti. Infatti il driver di interruzione si preoccupa solo di salvare l'accumulatore mentre distrugge il contenuto dei registri T e P (perchè il set di istruzioni del TMS32010 non consente di salvarlo con la snellezza necessaria a mantenere accettabili i tempi di esecuzione del driver). Occorre quindi disabilitare le interruzioni, nella routine di gestione dell'inviluppo, quando si vuol fare affidamento sul contenuto di tali registri. Nel programma realizzato, ad esempio, una volta terminata la scansione tabellare e procurati i valori degli inviluppi questi vengono moltiplicati, come si è detto, per dei fattori di scala. In queste sezioni del programma le interruzioni vanno disabilitate.

Nonostante questa solo apparente maggiore complicazione, la tecnica di gestione descritta risulta assai efficiente per una implementazione con un solo microprocessore rispetto alla soluzione descritta in {7} che prevedeva due microprocessori cooperanti.

Per finire, prima di illustrare in breve l'algoritmo di sintesi e di presentare il listato del programma esemplificativo, si vuole far notare che i programmi sviluppati con la tecnica descritta possono essere usati anche nel caso si voglia gestire il sistema delegando all'Host Computer il controllo degli inviluppi (sfruttando ad esempio la sorgente di sincronizzazione IRQ3). Basta, a questo scopo, che il mondo esterno lasci sempre FALSE la variabile GATE, la routine di gestione degli inviluppi non verrà mai eseguita e si potranno controllare direttamente gli inviluppi dei parametri.

Si noti a questo proposito che la distinzione delle variabili in globali e locali è una distinzione "logica", fatta per chiarezza e per comodità, non una distinzione "fisica". Il mondo esterno può tranquillamente modificare le variabili che abbiamo definito locali all'algoritmo di sintesi nel caso lo ritenga necessario.

5 - Problemi connessi con l'implementazione dell'algoritmo "FM" in Assembler TMS32010.

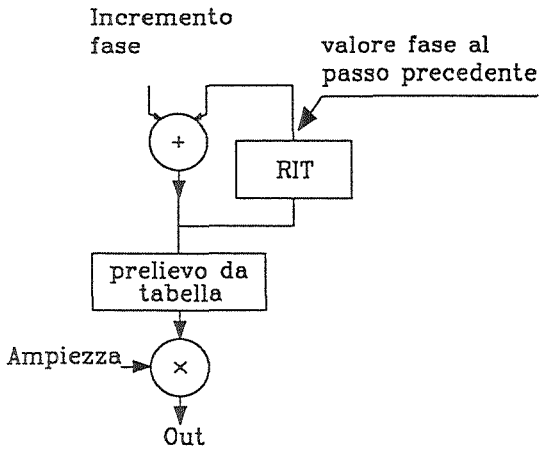
È opportuno premettere alle problematiche di questa tecnica, una breve descrizione della implementazione, con il microprocessore per DSP della Texas, di oscillatori digitali di precisione mediante la nota tecnica del "Table Look Up" {8,9}.
Sostanzialmente essa consiste nella lettura, ad intervalli di tempo costanti (periodo di campionamento), di un certo numero di valori da una tabella contenente i campioni della forma d' onda da generare. Per generare segnali di frequenza diversa si modifica il passo di scansione della tabella. Per ottenere una adeguata risoluzione in frequenza il passo di scansione deve essere rappresentato con un numero frazionario. Questa tecnica comporta che i segnali a frequenza più alta vengono prodotti con un numero sempre minore di campioni fino al minimo consentito dalla condizione di Nyquist che ci impone di utilizzare almeno due campioni per periodo. In fig. 7 è riportato lo schema funzionale di un oscillatore realizzato con la tecnica della scansione tabellare, ed uno schema più compatto che semplifica la rappresentazione di algoritmi più complessi. Supponiamo di voler generare il segnale: A(t) sen 2¶ft.

Nella nostra realizzazione abbiamo utilizzato tabelle di 512 valori (T_L=512) su 16 bit, ed un passo di scansione formato da un numero senza segno con 9 bit di parte intera e 7 bit di parte frazionaria (Q7).

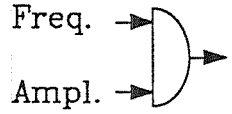
Durante il periodo di campionamento dobbiamo prelevare dalla tabella un nuovo campione. Per individuarlo, sommiamo alla fase precedente (cioè alla precedente posizione in tabella, rappresentata nella figura dalla quantità che esce dall'elemento di ritardo) un incremento che è funzione della frequenza "f" che vogliamo generare:

$$i = f\ T_L/f_c = f\ T_L\ T_c \qquad (f_c = \text{frequenza di campionamento}) \qquad [1].$$

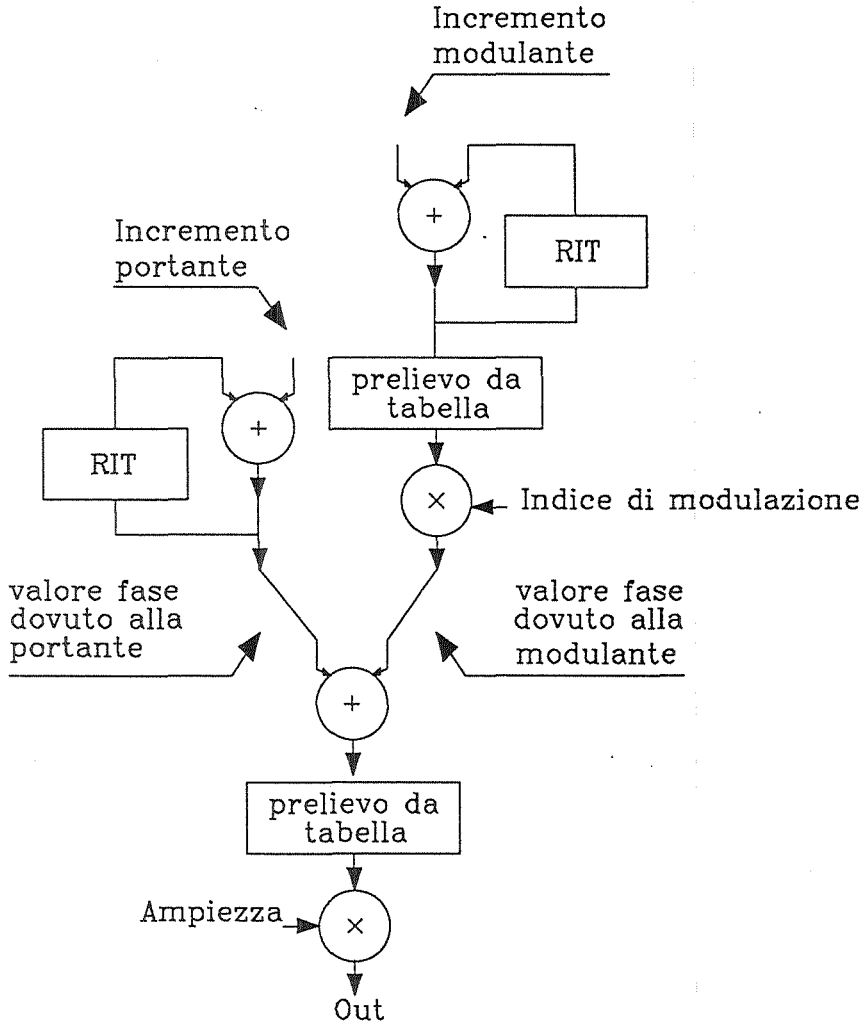
Nel sistema di cui stiamo discutendo f_c è selezionabile, via software, programmando l'orologio che genera gli interrupt verso il TMS32010; nelle prove effettuate è stata mantenuta f_c=32030 Hz. La somma viene effettuata ignorando un eventuale riporto, compiendo così l'operazione di "modulo lunghezza della tabella" (ovvero di "modulo 2¶"). Fatto questo si passa



OSCILLATORE VIRTUALE



SCHEMA SEMPLIFICATO



ALGORITMO PER LA SINTESI FM

Fig.7 Schema a blocchi di un oscillatore virtuale e dell'algoritmo di sintesi FM.

dall'indirizzo "logico" (un intero senza segno su 9 bit che fa da puntatore nell'array tabella) all'indirizzo fisico (indirizzo nella memoria del processore) e si esegue la lettura. Infine si moltiplica il valore letto per l'ampiezza desiderata "A" che in generale è funzione del tempo poichè controllata come già detto da un apposito generatore di inviluppo.

Combinando più generatori sinusoidali in modo che l'uscita di uno (modulatore) faccia variare la frequenza dell'altro (portante) si riescono ad ottenere segnali dallo spettro piuttosto ricco di armoniche e la larghezza di banda di questo spettro può essere modificata agendo su di un solo parametro: l'ampiezza della modulante. Attualmente questa tecnica di sintesi si è largamente diffusa nel settore commerciale in seguito all'introduzione di appositi integrati *VLSI* prodotti dalla Yamaha (Japan) per i sintetizzatori della serie "X". Per arricchire maggiormente lo spettro del segnale da generare si possono utilizzare i modulatori connessi in cascata o in parallelo fra di loro a piacere. Si noti infine che se la frequenza di un modulatore è piuttosto bassa (al massimo qualche decina di Hz) ed indipendente da quella portante si ottiene un effetto di vibrato.

L'algoritmo illustrato nel programma esemplificativo allegato a questa nota sfrutta due modulatori in cascata (il primo dei quali, inoltre, si "automodula") con in parallelo un generatore di vibrato per modulare l'oscillatore portante. La fig.8 illustra tale algoritmo evidenziando le differenti elaborazioni in parallelo e le variabili globali che l'Host controlla per ottenere il desiderato segnale musicale in uscita.

Ci occupiamo ora dettagliatamente dell'implementazione dell'algoritmo di sintesi FM in Assembler TMS32010.

Iniziamo con il fare una considerazione. Supponiamo di voler generare, mediante scansione tabellare, una forma d'onda di frequenza f_S. Dalla [1] si vede con facilità che, per la linearità dell'espressione che dà l'incremento, se si volesse generare una forma d'onda a frequenza $f_S = f_1 + f_2$, si può calcolare l'incremento i come $i_1 + i_2$ dove i_1 e i_2 sono gli incrementi necessari per generare le frequenze f_1 e f_2.

L'espressione matematica che rappresenta un segnale modulato in frequenza è del tipo:

$$s(t)= \sin(2\P f_p t+I \sin 2\P f_m t) \qquad [2]$$

dove: f_p è la frequenza della portante, f_m quella della modulante, ed **I** l'indice di modulazione; con $f_m/f_p = r$, rapporto di modulazione.

La frequenza, che è una funzione del tempo, risulta:

$$f(t)= 1/2¶\cdot(2¶f_p+2¶f_mI \cos 2¶f_mt)= f_p+f_mI \cos (2¶f_mt) \quad [3]$$

ed il suo valore massimo è:

$$f_p + f_mI$$

Quindi f_mI rappresenta il valore massimo della deviazione in frequenza rispetto alla portante.

Per realizzare una modulazione di frequenza occorre mantenere 2 oscillatori: uno genera la frequenza portante, l'altro la frequenza modulante. Il valore istantaneo dell'ampiezza dell'oscillazione prodotta dal secondo oscillatore viene moltiplicato per **I**. A questo valore si somma la fase istantanea dell'oscillatore portante. La somma delle due frequenze si esegue semplicemente sommando gli incrementi corrispondenti, sfruttando la proprietà osservata in precedenza.

Dai valori delle frequenze si ricavano, mediante la formula vista sopra, quelli degli 'incrementì da fornire ai puntatori alla tabella sinusoidale coi quali si realizzano gli oscillatori. Per quel che riguarda **I** si possono fare le seguenti osservazioni:

- negli istanti $\underline{t}$ l'incremento con cui viene aggiornata la fase (cioè il passo di scansione della tabella) deve essere pari a quello necessario per generare un' oscillazione a frequenza pari a $f_{MAX}=f_p+f_mI$;
- ad **I** deve corrispondere nel programma che realizza l'algoritmo di FM un incremento: i_I. Infatti il programma, una volta letto dalla tabella il valore di "sen $2¶f_mt$", che è un numero puro, lo moltiplica per i_I e lo somma all'incremento dell'oscillatore portante al quale i_I deve essere omogeneo;
- i_I rappresenta la massima deviazione di frequenza della portante. Infatti i_I viene moltiplicato per "sen $2¶f_mt$" che al massimo vale uno.

La regola pratica per determinare i_I è quindi la seguente: si moltiplica I per la frequenza modulante, si guarda a quale incremento corrisponde la frequenza così ottenuta e lo si rappresenta in Q7 come gli altri incrementi.

Occupiamoci dei problemi di aritmetica che si incontrano nell'implementare l'algoritmo illustrato, non perchè questi presentino qualche particolarità rispetto a problemi simili che si incontrano nella programmazione in qualsiasi linguaggio Assembler, ma perchè costituiscono un aspetto trascurato da chi è abituato a programmare in linguaggi ad alto livello senza doversi preoccupare di tali dettagli.

Un primo problema nasce quando si effettuano delle moltiplicazioni. In generale, la moltiplicazione di due valori su di un certo numero di bit non è più rappresentabile sullo stesso numero di bit, occorre passare a lavorare su di un numero doppio di bit, a meno di non rinunciare ad un pò di precisione. Nel nostro caso lavoriamo con numeri con segno su 16 bit in virgola fissa e per ragioni di velocità abbiamo escluso di portare avanti dei calcoli in doppia precisione.

Supponiamo di interpretare i due fattori di un prodotto come due numeri rappresentati in Qi e Qj (cioè rispettivamente con i e con j cifre frazionarie). Escludiamo a priori il caso in cui entrambi i fattori hanno il valore minimo consentito (ovvero sono rappresentati da >8000), staremo attenti a non metterci mai in questa condizione. Il risultato ha 15-i + 15-j cifre intere, i+j cifre frazionarie. Se 15-i+15-j è minore di 16, salvando la parte alta dell'accumulatore non perdiamo cifre intere significative. Può però succedere che le cifre più significative (MSB) siano degli zeri. Allora avremmo perso inutilmente parte dell'informazione contenuta nelle cifre frazionarie. Ci interessa salvare il risultato nel formato che consente di conservare il massimo numero di cifre significative. Vediamo di chiarirci le idee con un esempio. Moltiplichiamo un intero rappresentato in Q7 per uno in Q12. Supponiamo di volere il risultato rappresentato in Q7 perchè la natura del problema fisico che stiamo risolvendo ci garantisce la sua rappresentabilità in tale formato. È ovvio, da questo, che *non* possiamo moltiplicare *qualsiasi* valore, al contrario dovremo stare ben attenti che i due valori da moltiplicare siano tali che il loro prodotto sia effettivamente rappresentabile in Q7 (abbia cioè una parte intera che non necessità di più di 9 bit, compreso il segno, per essere rappresentata). Eseguiamo il prodotto e salviamo il risultato nell'accumulatore a 32 bit. Il

risultato è rappresentato su 32 bit in Q19 (cioè 7 + 12). A noi interessano solo 16 bit, allora eseguiamo lo shift dell'accumulatore di 4 posizioni verso sinistra (Attenzione! se il risultato è effettivamente rappresentabile le quattro cifre che si perdono sono tutte a zero). Ora il risultato è rappresentato in Q23 su 32 bit (anche se in realtà i quattro LSB non sono significativi: sono stati aggiunti ponendoli arbitrariamente a zero). Tronchiamo la rappresentazione "buttando via" i 16 LSB (teniamo solo la parte alta dell'accumulatore). Rimangono a questo punto solo 7 bit di parte frazionaria (23-16). Il risultato è rappresentato su 16 bit in Q7.

Vediamo come questo si applica all'algoritmo di modulazione in frequenza illustrato prima. Supponiamo, come avviene in pratica, di conoscere la frequenza portante ed il suo rapporto con la modulante e di voler ricavare quest'ultima.

Lavoreremo sugli "incrementi", cioè sui passi di scansione delle tabelle, poichè sono questi che vengono forniti dal mondo esterno al sistema di sintesi. Alla frequenza portante f_p corrisponde l'incremento i_p. Esso è rappresentato su 16 bit in Q7, quindi:

$$-256 <= i_p <= 255.992 \quad \text{(accuratezza 0.008)}$$

Il rapporto, per ragioni di comodo che chiariremo dopo sarà rappresentato su 16 bit in Q12, quindi:

$$-8 <= r <= 7.999 \quad \text{(accuratezza 0.001)}$$

affinchè il prodotto $r\ i_p = i_m$ (incremento corrispondente alla frequenza di modulazione) sia rappresentabile su 16 bit in Q7 deve essere:

$$-256 <= r\ i_p <= 255.992$$

Sia, ad esempio, i_p=64.0 (f_p = 4003 Hz), rappresentato su 16 bit dalla stringa 0010 0000 0000 0000. Se r=1.25 quindi a rappresentato su 16 bit dalla stringa 0001 0100 0000 0000 il loro prodotto su 32 bit è rappresentato da

0000 0010 1000 0000 0000 0000 0000 0000

eseguiamo lo shift di 4 posizioni verso sinistra:

0010 1000 0000 0000 0000 0000 0000 0000

scartiamo i 16 LSB ottenendo:

$r\ i_p$ = 0010 1000 0000 0000

che rappresenta 80 in Q7 (64 x 1.25 = 80). A questo incremento corrisponde la frequenza f_m = 5004 Hz.

Se avessimo, invece, avuto: i_p=64.0, r=7.0 il prodotto avrebbe dato 448 che *non è* rappresentabile su 16 bit in Q7. In questo caso, procedendo come abbiamo visto prima, si otterrebbe un risultato non corretto: lo shift di 4 posizioni infatti eliminerebbe delle cifre significative (anche uno zero è significativo come MSB quando il secondo MSB è un uno). Il TMS32010 consente di eseguire in un unico colpo solo gli shift di 0, 1 o 4 posizioni: questo è il motivo per cui è stata scelta la rappresentazione in Q12 per il rapporto r.

Ogni volta che viene moltiplicato un valore per un fattore di scala (in genere interpretabile come un numero rappresentato in Q15, compreso fra -1 e 1), si ha il risultato del prodotto in un registro a 32 bit ma i due MSB sono identici eccetto che nel caso che abbiamo escluso (basta mantenere le ampiezze rigorosamente maggiori di >8000). Quindi, il salvataggio della parte alta del registro che contiene il risultato della moltiplicazione avviene sempre contemporaneamente ad uno shift di una posizione verso sinistra, che elimina un MSB superfluo.

Concludiamo questo paragrafo facendo notare che nel programma esemplificativo è presente anche una routine denominata "eco". Questa salva in un "buffer" circolare i campioni che vengono inviati ai DAC e, dopo un breve intervallo di tempo (pari al prodotto della lunghezza dell'area di memoria utilizzata per il periodo di campionamento) li rilegge, moltiplicandoli per un fattore di ampiezza e sommandoli al campione calcolato nell'intervallo attuale. Il risultato viene infine spedito ai DAC e salvato al posto del valore di "eco" letto. La somma viene eseguita non appena il risultato del prodotto dell'eco per il volume è stato trasferito nell'accumulatore. Per quanto detto sopra, dopo il prodotto sono significativi i bit fra A15 e A30 (chiamando A0 il LSB dell'accumulatore ed A31 il MSB). Quando viene eseguita la somma, ovviamente eseguendo un opportuno shift dell'altro addendo prima di presentarlo al sommatore, si può avere un riporto che rende significativo anche A31. Ecco perchè in quella routine il salvataggio del risultato avviene senza il solito shift.

6 - Il programma di sintesi.

In figura è riportato lo schema funzionale dell'algoritmo implementato. I semicerchi rappresentano degli oscillatori, realizzati nel nostro programma mediante scansione tabellare, con i due ingressi: ampiezza e frequenza. Moltiplicatori e sommatori sono rappresentati dai simboli usuali per quelle operazioni racchiusi in un cerchio.

Sulla sinistra sono mostrati i parametri in ingresso all'algoritmo. In generale i rapporti di frequenza vengono forniti come caratteristiche del timbro e non vengono più mutati fino a che non si cambia timbro. Le ampiezze dei due oscillatori modulanti e l'ampiezza complessiva del suono ottenuto vengono inviluppate dal programma principale in esecuzione sul TMS32010, il mondo esterno controlla, in fase di impostazione del timbro i fattori di scala per i quali moltiplicare i valori letti dalle tabelle d'inviluppo così come le velocità di scansione delle tabelle e le posizioni dei "nodi". L'ampiezza del "feed-back" di frequenza del primo modulatore potrebbe essere inviluppata ma nel programma riportato è lasciata costante (viene anch'essa fissata dall'esterno in fase di impostazione del timbro). Ampiezza e frequenza del vibrato verranno impostati dall'esterno per quelle note che hanno bisogno di quest'effetto così come il volume dell'eco che, pur nei limiti dovuti al fatto che il ritardo è costante e pari a circa 63 millisecondi (potrebbe al limite essere diminuito ma non aumentato a causa della scarsa disponibilità di spazio in memoria), dà un minimo effetto di "ambienza" che rende più piacevole il timbro.

7 - Ambiente di sviluppo SW.

Per poter utilizzare il sistema, è stato necessario sviluppare in ambiente MS-DOS, ove sono disponibili gli strumenti SW della Texas Instruments (MacroAssembler, Linker, Simulator) sul PC-IBM, un certo numero di programmi di corredo. Tra di questi figurano:

- un loader che consente di caricare gli oggetti prodotti dal cross-assemblatore nella memoria del TMS32010 sulla scheda;

- un editore di tabelle, che consente di creare le tabelle nello stesso formato utilizzato dall'assemblatore, oppure come file di interi (2 byte per ogni valore);
- un loader che consente di caricare le tabelle scritte nel secondo formato;
- una routine per lanciare l'esecuzione del programma sulla scheda;
- un programma di set-up per l'orologio che genera le interruzioni verso la scheda **MP3A** e verso il PC.
- alcuni programmi che consentono di interagire con il sistema di sintesi osservando in tempo reale l'efficacia, sul risultato sonoro, delle modifiche effettuate sui vari parametri; mediante questi programmi è possibile studiare timbri diversi, "suonarli", salvarli su disco e ricaricarli per l'esecuzione di un brano.

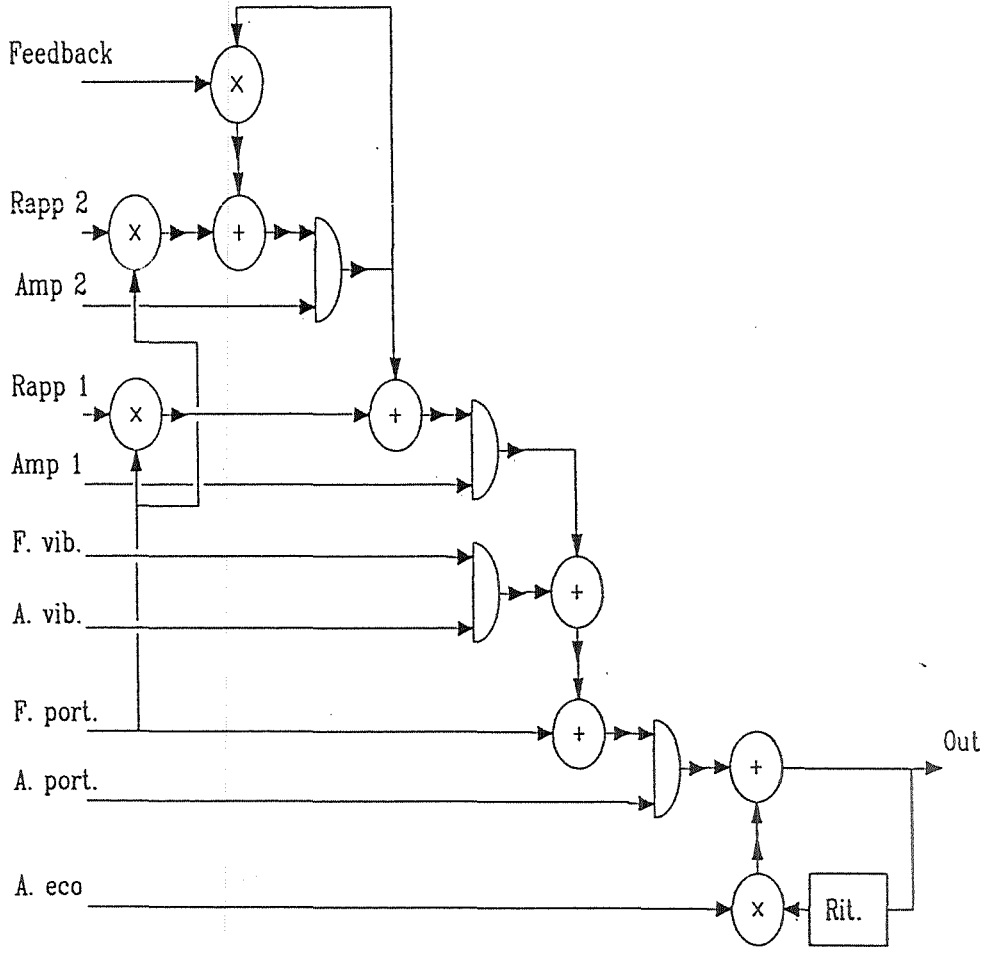


Fig.8 Schema completo dell'algoritmo di sintesi implementato.

APPENDICE A: Standard di programmazione sul TMS32010.

Vengono riportati qui di seguito alcuni utili modelli di impostazione del software a cui rifarsi per la stesura di programmi per la scheda di sintesi MP3A. Durante la fase di realizzazione e sperimentazione dei vari programmi per provare le tecniche di gestione illustrate e misurare le prestazioni del sistema, ci si è resi conto che l'utente ha talmente tanti gradi di libertà di fronte alle scelte, anche le più banali, che finisce per seguire ogni volta una strada diversa. Ciò può comportare un inutile sforzo per ricordare, ad esempio, l'indirizzo in cui il programma si aspetta di trovare una tabella di inviluppo, o quello in cui si è caricato la tabella contenente una certa forma d'onda, il numero assegnato ad un certo parametro, eccetera. Ecco perchè si è ritenuto utile stabilire alcune modalità di stesura dei programmi che possono essere considerate anche come riferimenti per una migliore comprensione dei listati. Si tenga comunque presente che, nel caso si voglia procedere in maniera diversa da quanto suggerito, *non è ovviamente necessario* rispettare questi standard.

Nota importante:

nel tipo di funzionamento "stand-alone" implementato, come noto è il TMS32010 stesso a gestire gli inviluppi all'interno del programma principale e ad effettuare la sintesi dei campioni nel driver di interruzione. Il driver si preoccupa solo di salvare l'accumulatore poichè si suppone che il programma usi variabili diverse dalle sue (ad esempio le varie variabili di appoggio). Il programma deve cautelarsi anche quando conta di trovare dati consistenti nei registri T e P (è bene pertanto eseguire sempre una PAC immediatamente dopo una moltiplicazione e togliere la possibilità di interrompere il microprocessore prima di caricare il registro T e ricordarsi infine di riabilitarla dopo averlo utilizzato).

Mappa della memoria "programma" dell'MP3A.

01F - FFF	(32 - 4095)	4 Kword RAM statica veloce
000 - 01F	(0 - 31)	32 word PROM di "bootstrap"

Per esigenze di tempo di esecuzione è impensabile che un programma occupi più di 1 Kword, allora sono state riservate 3 Kword (400 - FFF) alle tabelle. Le tabelle sono lunghe 256 o 512 word (inviluppi o forme d' onda). In fondo allo spazio di memoria (E00 - FFF) viene messa la tabella sinusoidale (o comunque quella "principale" usata per la generazione dei campioni), sotto di questa, andando verso indirizzi più bassi, se ne possono mettere altre, sempre per la generazione dei campioni (ad es.: in FM due tabelle diverse per portante e modulante). Infine le tabelle per gli inviluppi vengono caricate ad indirizzi ancora più bassi.

000 - 3FF	(0 - 1023)	area riservata a programma, driver di interruzione e costanti.
400 - FFF	(1024 - 4095)	spazio riservato a tabelle. Si riportano, per comodità dell'utente, gli indirizzi a cui caricare le tabelle in modo che non vi siano sovrapposizioni.

Forme d' onda		Inviluppi
400 - 5FF	(1024 - 1535)	400 - 4FF (1024 - 1279)
		500 - 5FF (1280 - 1535)
600 - 7FF	(1536 - 2047)	600 - 6FF (1536 - 1791)
		700 - 7FF (1792 - 2047)
800 - 9FF	(2048 - 2559)	800 - 8FF (2048 - 2303)
		900 - 9FF (2304 - 2559)
A00 - BFF	(2560 - 3071)	A00 - AFF (2560 - 2815)
		B00 - BFF(2816 - 3071)
C00 - DFF	(3072 - 3583)	C00 - CFF(3072 - 3327)
		D00 - DFF (3328 - 3583)
E00 - FFF	(3584 - 4095)	E00 - EFF (3584 - 3839)
		F00 - FFF (3840 - 4095)

Riservando 1 Kword a programma e driver, tenendo conto dello spazio occupato dalla ROM, si può stabilire la convenzione:

020 - 27F driver (uscita del campione, eventuale rinnovo parametri, elaborazione nuovo campione).

280 - 2FF area dati di inizializzazione per costanti e variabili. Si veda una tecnica, adottabile come standard, per leggere le costanti all'inizio del programma esemplificativo.

300 - 3FF programma (inizializzazione, eventuale rinnovo parametri, gestione inviluppo).

400 - FFF area tabelle.

Mappa della memoria dati dell'MP3A (Variabili dei programmi)

Le variabili dichiarate nei programmi si possono suddividere in due gruppi:

I - *Variabili di Interfaccia* (visibili anche all'Host che le può modificare inviando all'MP3A il relativo indirizzo ed il nuovo valore da apporvi);
II - *Variabili Locali* (del programma principale e del driver di interruzione).

A loro volta le variabili di interfaccia possono essere relative alla generazione di un campione (ad es.: frequenza di un oscillatore o rapporto fra frequenze) o alla gestione delle tabelle di inviluppo (ad es.: velocità di attack o release). L'ampiezza, anche se è il risultato della generazione di inviluppo, può essere compresa tra le variabili relative alla generazione di un campione perchè il programma sull'HOST può volerla gestire senza utilizzare la routine di controllo dell'inviluppo che gira sul TMS32010.

Si è suddivisa la DATA RAM interna al TMS32010, che è l'unica memoria dati dell'MP3A, nel seguente modo:

pagina 0 Data RAM

00 - 1F variabili di interfaccia relative al campione + ampiezza.

20 - 3F variabili di interfaccia relative all'inviluppo.
40 - 7F variabili locali

pagina 1 Data RAM

00 - 0F variabili locali.

È bene ricordare comunque che l'appellativo "locali" non significa che l'Host non può modificare tali variabili. Infatti ciò dipende dal tipo di colloquio che si implementa: se il programma che esegue l'ingresso dei parametri sull'MP3A è quello standard utilizzato nei vari programmi realizzati le variabili in pagina 1 non sono accessibili all'Host.

APPENDICE B: Listato del programma di esempio.

```
*******************************************************************************
* MODULAZIONE DI FREQUENZA: ALGORITMO 1                    ver 1.0
*******************************************************************************
*
* CARATTERISTICHE
* 4 GENERATORI: UNA PORTANTE, DUE MODULANTI IN CASCATA, UN VIBRATO
*
* FEEDBACK DI FREQUENZA PER IL GENERATORE 2
*
* Sono necessari quattro oscillatori che vengono realizzati mediante
* table look-up utilizzando pero' la stessa tabella per evitare un' eccessiva
* occupazione di memoria. La tabella, lunga 512 word deve essere caricata in
* memoria a partire dall' indirizzo >E00. Se si desiderano utilizzare tabelle
* differenti e' sufficiente caricarle e modificare le variabili che contengono
* gli indirizzi delle tabelle (sono visibili all' Host Computer).
*
*     LA GESTIONE DEGLI INVILUPPI E' CONDOTTA DAL PROGRAMMA PRINCIPALE
*
* Vengono gestiti gli inviluppi dell' ampiezza globale e degli indici di
* modulazione, scandendo altre due tabelle, lunghe 256 word caricate agli
* indirizzi >C00 (inviluppo globale) e >D00 (per tutti gli indici di mod.).
* Gli inviluppi delle modulanti sono moltiplicati per un fattore di scala.
*
* Sono presenti anche dei controlli di volume.
*
*******************************************************************************
*   DEFINIZIONE MACRO
*******************************************************************************
*
* La macro INPUT legge il contenuto delle due porte di ingresso mettendolo in
* memoria e resetta il flip-flop che mantiene il livello basso sul pin BIO.
*
INPUT       $MACRO
            IN      PARNUM,TMS1
            IN      PARVAL,TMS2
            OUT     DUM,BIORES
            LAR     0,PARNUM
            LAC     PARVAL
            SACL    *
            $END
*
* La macro OSCILL riceve in ingresso i parametri FASE, FREQuenza e TABella
* ed incrementa il puntatore alla tabella, lo salva (aggiorna la FASE), legge
* dalla tabella lasciando il campione in TABVAL. Viene "sporcata" la variabile
* APPO.
*
OSCILL      $MACROFASE,FREQ,TAB
            ZALS    :FREQ.S:
            ADD     :FASE.S:
```

```
                SACL    :FASE.S:

                LAC     :FASE.S:,9
                SACH    APPO

                LAC     MSK1FF
                AND     APPO

                OR      :TAB.S:

                TBLR    TABVAL
                $END
*
* La macro OSCFM riceve in ingresso i parametri FASE e TABella
* e, TRAMITE L' ACCUMULATORE, l' ampiezza della modulante da sommare alla FASE
* per poter trovare l' offset nella tabella.
* Incrementa il puntatore alla tabella, NON lo salva (NON aggiorna la FASE),
* legge dalla tabella lasciando il campione in TABVAL. Viene "sporcata"
* la variabile APPO.
*
OSCFM           $MACROFASE,TAB
                ADD     :FASE.S:
                SACL    APPO

                LAC     APPO,9
                SACH    APPO

                LAC     MSK1FF
                AND     APPO

                OR      :TAB.S:

                TBLR    TABVAL
                $END
*
* La macro MOLTIP esegue la moltiplicazione del valore contenuto in VAL
* per il parametro ENV e lascia il risultato in RES
*
MOLTIP          $MACROENV,VAL,RES
                LT      :ENV.S:
                MPY     :VAL.S:
                PAC
                SACH    :RES.S:,1
                $END
*
*********************************************************************************
*
* Celle di memoria dati utilizzate
*
* Variabili di Interfaccia
*
                AORG    0
```

```
INCPOR      BSS 1       * frequenza portante
AMPPOR      BSS 1       * ampiezza              inviluppo complessivo
RAPP1       BSS 1       * rapporto fra prima freq. modulante e freq. portante
I1          BSS 1       * ampiezza inviluppo prima modulante
RAPP2       BSS 1       * rapporto fra seconda freq. modulante e freq. portante
I2          BSS 1       * ampiezza inviluppo seconda modulante
AMPFDB      BSS 1       * feedback
FREQVI      BSS 1       * freq. vibrato
IVI         BSS 1       * ampiezza vibrato
VOLECO      BSS 1       * volume eco
*
            AORG    >10

MXINV1      BSS 1       * max valore inviluppo prima modulante
MXINV2      BSS 1       * max valore inviluppo seconda modulante
*
            AORG    >20         * variabili globali per la gestione dell' inviluppo.

GATE        BSS 1
QUANTI      BSS 1
ATTSPE      BSS 1
RELSPE      BSS 1
NODO        BSS 1
VOLSN       BSS 1       * volume canale destro
VOLDX       BSS 1       * volume canale sinistro
*
* Variabili Locali
*
            AORG    >40

AMPMD1      BSS 1       * ampiezza istantanea prima modulante
AMPMD2      BSS 1       * ampiezza istantanea seconda modulante
AMPVI       BSS 1       * ampiezza istantanea vibrato
FASEP       BSS 1       * fase istantanea portante
FASEM1      BSS 1       * fase istantanea prima modulante
FASEM2      BSS 1       * fase istantanea seconda modulante
FASEVI      BSS 1       * fase istantanea vibrato
*
SAMTBP      BSS 1       * celle che contengono gli indirizzi delle tabelle
SAMTB1      BSS 1       * dei campioni
SAMTB2      BSS 1
SAMTVI      BSS 1
ENVTAB      BSS 1       * e degli inviluppi
ENVTB1      BSS 1
ENVTB2      BSS 1
*
MSK1FF      BSS 1       * celle che contengono le maschere: 0000 0001 1111 1111
MSK0FF      BSS 1       * 0000 0000 1111 1111
*
TABVAL      BSS 1
CAMP        BSS 1
CAMPL       BSS 1
CAMPR       BSS 1
```

```
*
ACCLO       BSS 1       * area di salvataggio per l' accumulatore
ACCHI       BSS 1
*
PARNUM      BSS 1       * celle per ingresso numero parametro
PARVAL      BSS 1       * e parametro
*
SUST        BSS 1       * variabili per gestione tabelle di inviluppo
RELEAS      BSS 1
PAUSA       BSS 1
FASENV      BSS 1
PUNTA       BSS 1
APPENV      BSS 1
*
ENVSEM      BSS 1
ENVCNT      BSS 1
UNO         BSS 1
*
APPO        BSS 1
DUM         BSS 1
*
PUNECO      BSS 1
ECO         BSS 1
LIMSUP      BSS 1
LIMINF      BSS 1
*
*********************************************************************************
* Porte di I/O
*
* porta in cui viene passato il numero del parametro
*
TMS1        EQU     0
*
* porta in cui viene passato il paramentro
*
TMS2        EQU     1
*
*
BIORES      EQU     4
INTRES      EQU     5
*
DACL        EQU     6           * convertitori
DACR        EQU     7
*
*********************************************************************************
*                          DRIVER DI INTERRUZIONE                               *
*********************************************************************************
* Driver di interruzione: calcola il nuovo campione e compie l' uscita del
* campione, calcolato durante l' interrupt precedente, verso i convertitori
*
            AORG    >20

DRIVER      SACL    ACCLO
```

```
                SACH    ACCHI
                OUT     CAMP,DACL
                OUT     CAMP,DACR
*
* eventuale rinnovo parametri (siamo sicuri che il pin BIO viene "testato"
* con una frequenza pari a quella di campionamento).
*
                BIOZ    ELAB
                INPUT
*
* Blocco che realizza l' ALGORITMO DI SINTESI
*
ELAB
*
* Blocco che realizza l' oscillatore 'modulante' 2 con feedback di frequenza
*
                LT      INCPOR      * si ricava la frequenza
                MPY     RAPP2
                PAC
                SACH    APPO,4

                ZALS    FASEM2      * aggiorna la fase come se non ci fosse modulaz
                ADD     APPO
                SACL    FASEM2

                LT      AMPMD2              * calcolo del feed back
                MPY     AMPFDB
                PAC
                SACH    APPO,1
                LAC     APPO                * lo lasciamo nell' accumulatore

                OSCFM   FASEM2,SAMTB2

                MOLTIP  I2,TABVAL,AMPMD2    * si moltiplica per 'I2' e  si salva
*
*********************************************************************************
*
* Blocco che realizza l'oscillatore 'modulante' 1 modulato dal gen 2
*
                LT      INCPOR              * si ricava la frequenza
                MPY     RAPP1
                PAC
                SACH    APPO,4

                ZALS    FASEM1
                ADD     APPO
                SACL    FASEM1

                LAC     AMPMD2
                OSCFM   FASEM1,SAMTB1

                MOLTIP  I1,TABVAL,AMPMD1    * si moltiplica per 'I1' e  si salva
*
```

```
*******************************************************************************
*
* Blocco che realizza Il vibrato
*
                OSCILL  FASEVI,FREQVI,SAMTVI

                MOLTIP  IVI,TABVAL,AMPVI        * si moltiplica per 'IVI' e  si salva
*
*******************************************************************************
*
*
* Blocco che realizza l' oscillatore portante'
*

ELAB1           ZALS    FASEP           * calcolo della fase dovuta alla freq portante
                ADD     INCPOR
                SACL    FASEP

                LAC     AMPMD1    * nell' ACC lasciamo la somma della modulante 1
                ADD     AMPVI                   * e del vibrato
                OSCFM   FASEP,SAMTBP

                MOLTIP  AMPPOR,TABVAL,CAMP      * si moltiplica per 'AMPPOR' e si salva
*
* aggiungiamo l' eco
*
INIZIO          ZALS    PUNECO                  * leggiamo l' eco dalla memoria
                TBLR    ECO
                LT      ECO                     * moltiplichiamo per il volume
                MPY     VOLECO
                PAC                             * il risultato della moltiplicazione e' ora
*                                       nell' ACC. I bit che ci interessano vanno da
*                                       A30 a A15.

                ADD     CAMP,15   * sommiamo ai bit A30 - A15 il CAMP
                SACH    CAMP                    * puo' esserci riporto -> diventa significativo
*                                       anche il bit A31
*
                ZALS    PUNECO                  * salviamo la somma CAMP+ECO
                TBLW    CAMP
*
                ADDS    UNO                     * aggiorniamo il puntatore alla zona ECO
                SACL    PUNECO
                XOR     LIMSUP
                BNZ     DENTRO
                ZALS    LIMINF
                SACL    PUNECO
*
DENTRO          NOP

                MOLTIP  CAMP,VOLSN,CAMPL  * moltiplica il campione per il volume
*                                       del canale sinistro
```

```
                MPY     VOLDX       * moltiplica il campione per il volume
                PAC                 * del canale destro
                SACH    CAMPR,1
*
* vediamo se e' il momento di far elaborare un nuovo valore di inviluppo
*
                ZALS    ENVCNT
                SUB     UNO         * decrementa il contatore
                SACL    ENVCNT      * se non e' zero salvalo ed esci
                BNZ     RIT

                ZALS    QUANTI      * altrimenti ricaricalo col valore iniziale
                SACL    ENVCNT      * e metti il semaforo "verde" --> quando il
                LACK    1           * programma principale trova il semaforo verde
                SACL    ENVSEM                  * inizia ad elaborare un nuovo valore per gli
*                                    inviluppi
*
* ripristiniamo l' accumulatore, resettiamo il flip flop che mantiene
* il livello basso sul pin di interrupt e torniamo
*
RIT             ZALH    ACCHI
                OR      ACCLO
                OUT     DUM,INTRES
                EINT
                RET
*******************************************************************************
*                       DICHIARAZIONE DI COSTANTI                             *
*******************************************************************************
*
TABLEN          EQU     >100
*
                AORG    >280
*
                DATA    >0E00       * Indirizzo della tabella dei campioni
                DATA    >0D00       * Indirizzo della prima tabella d' inviluppo (globale)
                DATA    >0C00       * Indirizzo della seconda tabella d' inviluppo (modulanti)
*
                DATA    >01FF       * Maschera per cancellare i 7 MSB
                DATA    >00FF       * Maschera per cancellare gli 8 MSB
*
                DATA    >07FE       * Frequenza portante iniziale
                DATA    >066C       * Rapporto fra frequenza portante e frequenze modulanti
*                                     iniziale (1/4: rappresentazione in Q12)
*
                DATA    >7FFF       * Valore iniziale ampiezza inviluppi e volumi
*
                DATA    >0100       * Attack speed
                DATA    >0100       * Release speed
                DATA    >007F       * Nodo
                DATA    >0020       * Numero di intervalli di campionamento nei quali
*                                     l' inviluppo rimane costante (con questo valore si ha un
*                                     rinnovo dell' inviluppo ogni millisecondo)
```

```
*
*   N.B.: DEVE ESSERE SUFFICIENTEMENTE ALTO AFFINCHE' IL
*   PROGRAMMA RIESCA A COMPLETARE IL CALCOLO DI UN VALORE DEGLI
*   INVILUPPI PRIMA CHE GLI VENGA RICHIESTO DI GENERARE IL
*   VALORE SUCCESSIVO. Il tempo utilizzabile dal programma per
*   la sua elaborazione e': Tc-Tempo esecuzione driver.
*   (Tc pari a circa 31 microsecondi).
*
                DATA    >400        * limite inferiore area riservata all' eco
                DATA    >BFF        * limite superiore area riservata all' eco
*******************************************************************************
*                               PROGRAMMA                                     *
*******************************************************************************
*
                AORG    >300
*
* Inizializzazione: vengono assegnati i valori iniziali alle maschere e alle
* variabili.
*
INIT            OUT     DUM,BIORES
                LARP    0
                LACK    >0028
                SACL    APPO
*
* lettura dall' area dati in program RAM
*
                ZAC
                ADD     APPO,4
                TBLR    SAMTBP
                TBLR    SAMTB1
                TBLR    SAMTB2
                TBLR    SAMTVI

                LACK    1
                ADD     APPO,4
                TBLR    ENVTAB

                LACK    2
                ADD     APPO,4
                TBLR    ENVTB1
                TBLR    ENVTB2

                LACK    3
                ADD     APPO,4
                TBLR    MSK1FF

                LACK    4
                ADD     APPO,4
                TBLR    MSK0FF

                LACK    5
                ADD     APPO,4
                TBLR    INCPOR
```

```
LACK    6
ADD     APPO,4
TBLR    RAPP1
TBLR    RAPP2

LACK    7
ADD     APPO,4
TBLR    AMPPOR
TBLR    I1
TBLR    I2
TBLR    MXINV1
TBLR    MXINV2
TBLR    VOLSN
TBLR    VOLDX

LACK    8
ADD     APPO,4
TBLR    ATTSPE

LACK    9
ADD     APPO,4
TBLR    RELSPE

LACK    >0A
ADD     APPO,4
TBLR    NODO

LACK    >0B
ADD     APPO,4
TBLR    QUANTI
TBLR    ENVCNT

LACK    >0C
ADD     APPO,4
TBLR    LIMINF
TBLR    PUNECO

LACK    >0D
ADD     APPO,4
TBLR    LIMSUP

*

LACK    1               * TRUE=1
SACL    PAUSA
SACL    RELEAS
SACL    ENVSEM
SACL    UNO

ZAC                     * FALSE=0
SACL    CAMP
SACL    FASEP
SACL    FASEM1
```

```
                SACL    FASEM2
                SACL    SUST
                SACL    GATE
                SACL    FASEVI
                SACL    FREQVI
                SACL    AMPVI
                SACL    IVI
                SACL    AMPMD2
                SACL    AMPFDB
                SACL    VOLECO
*
*******************************************************************************
*
* eventuale rinnovo parametri
*
NEWPAR          DINT
                BIOZ    CHKSEM
                INPUT
*
CHKSEM          EINT
                NOP
                ZALS    ENVSEM
                BZ      NEWPAR          * semaforo verde? no!-->torna indietro
*                                         si!-->prosegui
* Gestione dell' inviluppo
*
                ZALS    GATE
                BZ      RILASC
*
* Gate = true --> Attack Decay Sustain
*
                ZALS    RELEAS
                BZ      SOTTO
                ZAC             * veniamo da un release --> dobbiamo ripartire
*                                 dall' inizio della tabella
                SACL    FASENV  * fase:= inizio tabella
                SACL    RELEAS  * Release:= false
                B       AGGIOR

SOTTO           ZALS    SUST
                BNZ     FUORI   * se c'e' Sustain non dobbiamo far nulla

                SACL    PAUSA   * Pausa:= false
                ZALS    ATTSPE
                ADDS    FASENV
                SACL    FASENV  * Fase Inviluppo:= Fase Inviluppo+Attack speed

                LAC     FASENV,8 * Attenzione: Sign Extend!
                SACH    PUNTA   * bisogna resettare gli 8 MSB
                ZALS    PUNTA
                AND     MSK0FF
                SACL    PUNTA
```

```
                ZALS    NODO        * Abbiamo sorpassato il nodo?
                SUB     PUNTA
                BLZ     AVANTI      * Si! salta avanti

                ZALS    PUNTA       * No! Accumulatore:= Punta
                B       AGGIOR      * salta ad aggiornare

AVANTI          LACK    1           * Abbiamo sorpassato il nodo!
                SACL    SUST        * Sustain:= true

                ZALS    NODO        * Accumulatore:= Nodo
                B       AGGIOR      * salta ad aggiornare
*
* Gate = false --> Release
*
RILASC          LACK    1           * Release:= true
                SACL    RELEAS

                ZALS    PAUSA       * se siamo in pausa non far nulla --> esci
                BNZ     FUORI

                ZAC                 * altrimenti
                SACL    SUST        * Sustain:= false

                ZALS    RELSPE
                ADDS    FASENV
                SACL    FASENV      * Fase Inviluppo:= Fase Inviluppo+Release speed
                SACH    PAUSA       * Se la somma ha invaso la parte alta dell'
*                                     Accumulatore --> abbiamo finito la tabella

                LAC     FASENV,8            * Attenzione: Sign Extend!
                SACH    PUNTA               * bisogna resettare gli 8 MSB
                ZALS    PUNTA
                AND     MSK0FF
                SACL    PUNTA

                ZALS    PAUSA
                BZ      GIU                 * Se non siamo in pausa salta GIU

                LACK    TABLEN-1            * Altrimenti usa l' ultimo valore della
                B       AGGIOR              * tabella per aggiornare

GIU             ZALS    PUNTA

AGGIOR          SACL    APPENV
                OR      ENVTAB
                TBLR    AMPPOR

                AND     MSK0FF
                OR      ENVTB1
                DINT
                TBLR    I1
```

```
                LT      MXINV1          * moltiplico l' inviluppo della prima
                MPY     I1              * modulante per un fattore di scala
                PAC
                EINT
                SACH    I1,1

                ZALS    APPENV
                AND     MSK0FF
                OR      ENVTB2
                DINT
                TBLR    I2

                LT      MXINV2  * moltiplico l' inviluppo della seconda
                MPY     I2      * modulante per un fattore di scala
                PAC
                EINT
                SACH    I2,1

FUORI           ZAC
SACL            ENVSEM          * semaforo=rosso
                B       NEWPAR
*******************************************************************************

   END
```

Bibliografia

{1} L. Tarabella, G. Bertini: "Il modulo di sintesi sonora MP3 con TMS32010: interfaccia con C-64 e modalità d'uso". Nota interna CNUCE C-87 n.16.

{2} Jon Bradley, Texas Instruments: "Interfacing to Asynchronous Inputs with the TMS32010". Digital Signal Processing Applications 1986.

{3} G. Chini, P. Chiodaroli: "Progetto e realizzazione di un sistema a microprocessori per la sintesi del suono". Tesi di Laurea, Relatori: G. Bertini, L. Tarabella, P. Corsini, F. Russo, Marzo 1987.

{4} J.M.Chowning: "The Synthesis of Complex Spectra by Mean of Frequency Modulation". Computer Music Journal, Aprile 1977.

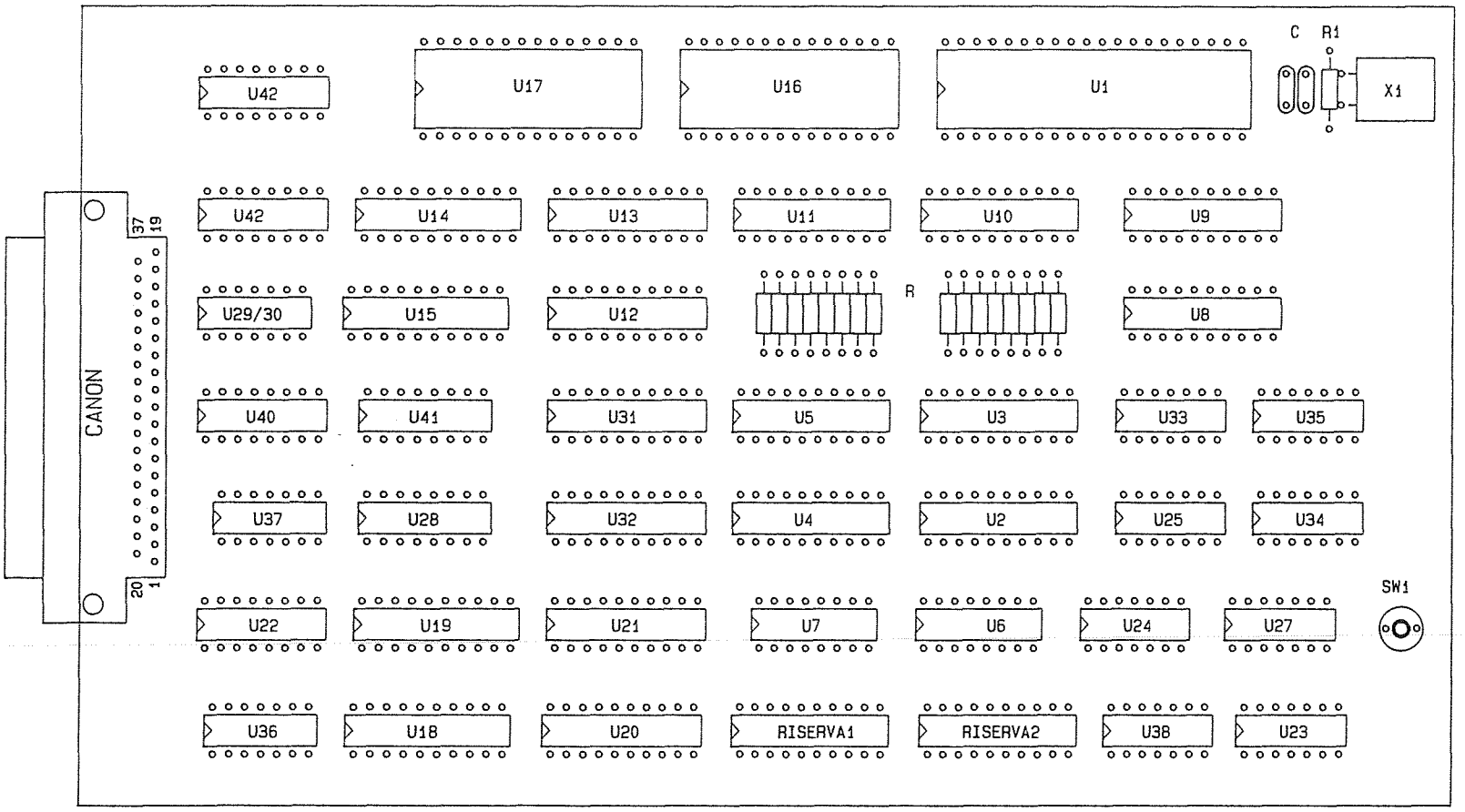
{5} G. Bertini, G. Chini, P. Chiodaroli, L. Tarabella: "Progetto di un sistema polifonico a sintesi digitale in tempo reale. Descrizione della realizzazione del prototipo". Nota interna IEI B4-6, gennaio 1987.

{6} S.Toni: "Progetto e Realizzazione di un Sistema Multi-DSP per la Sintesi e l'Elaborazione di Segnali Audio Controllata da Personal Computer". Tesi di Laurea, Relatori: P.Corsini, G.Frosini, L.Tarabella, G.Bertini, Luglio 1988.

{7} L.Tarabella, G.Bertini: "Un Sistema di Sintesi ad Elevate Prestazioni Controllato da Personal Computer". Quaderni di Musica/Ricerca n.°14, Ed.Musicali 1987 (MI) pag.330-335

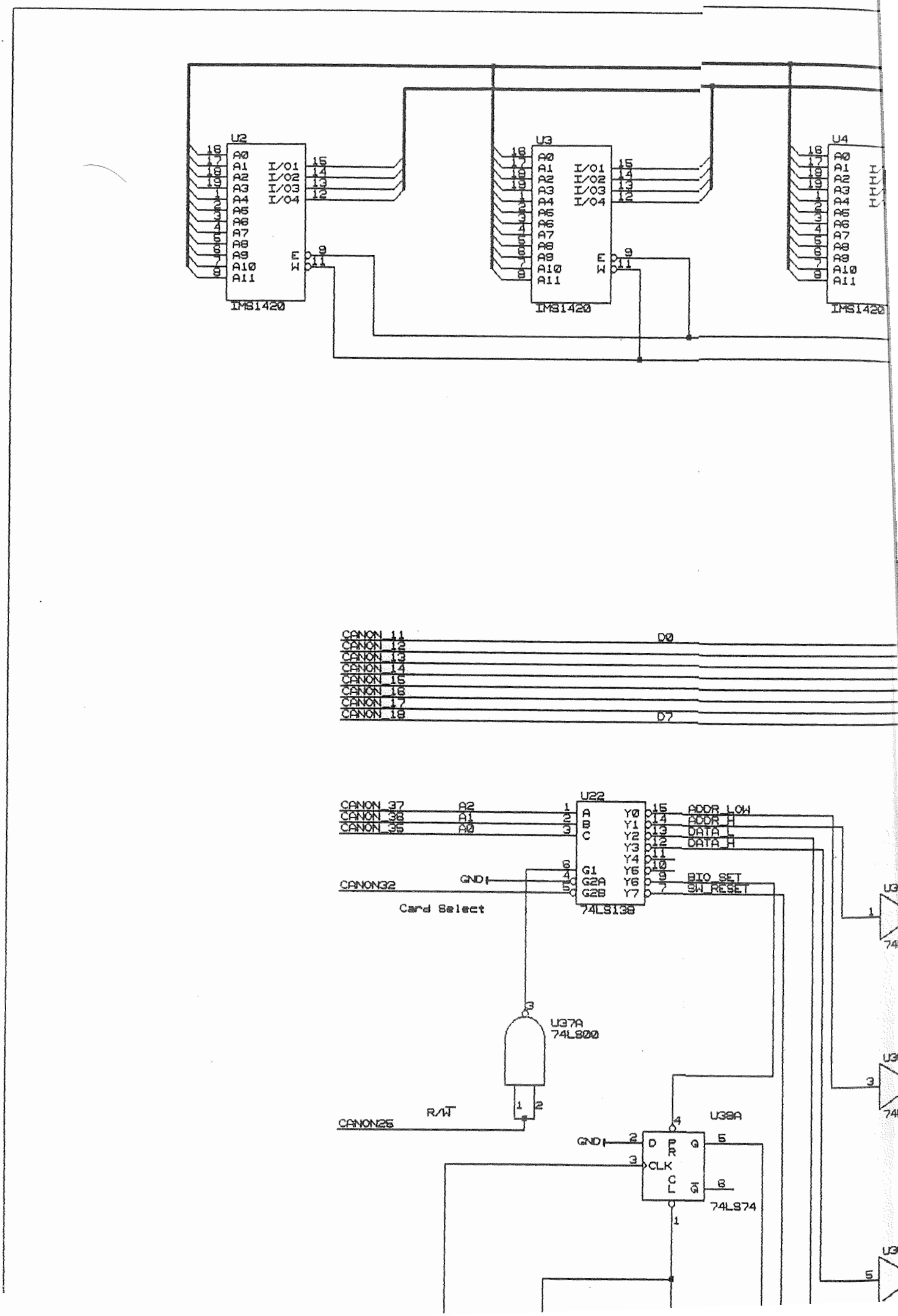
{8} L.M. Del Duca: "Musica Digitale, Sintesi, Analisi e Filtraggio Digitale". Franco Muzzio Editore, 1987.

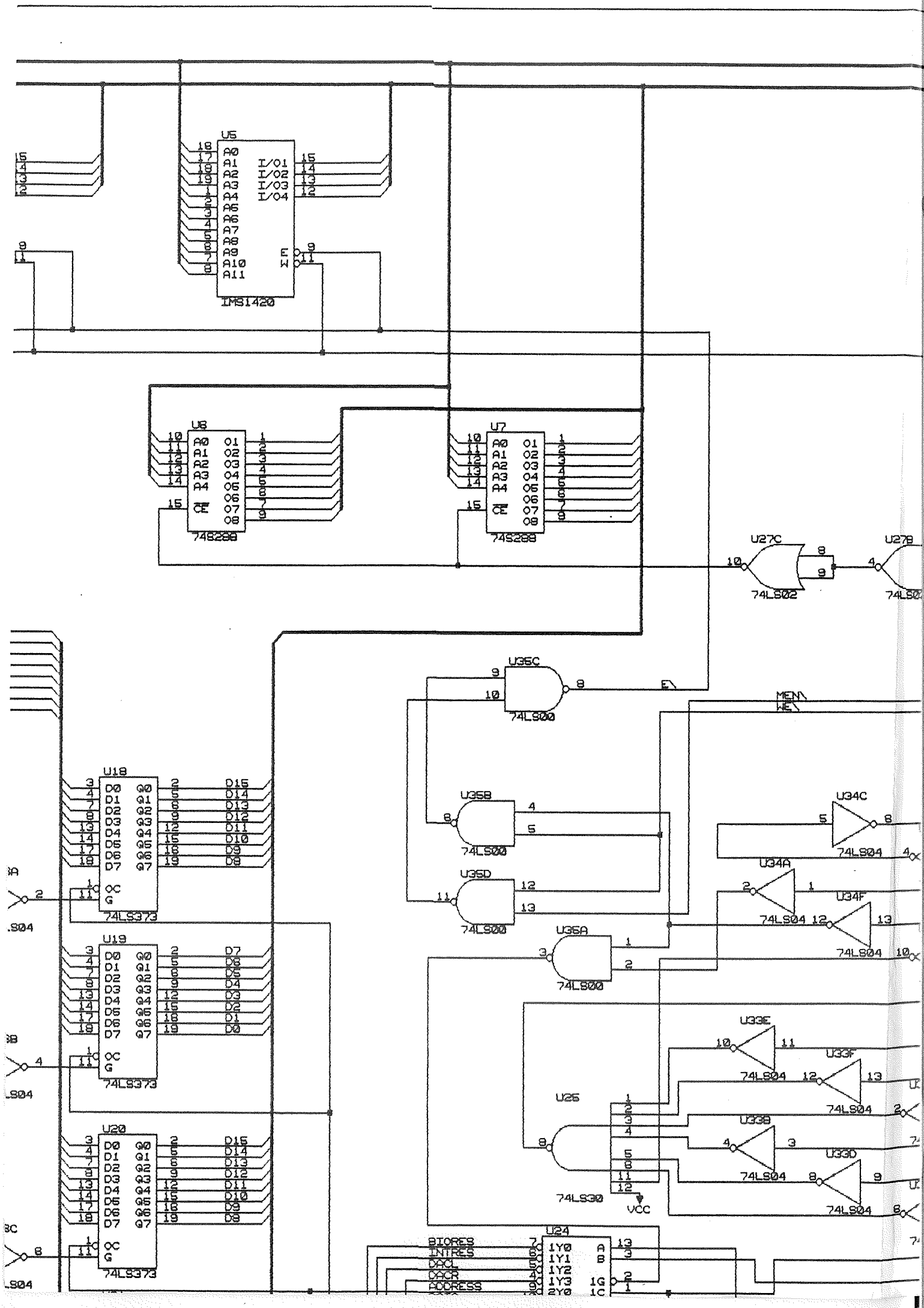
{9} Domingo Garcia: "Precision Digital Sine Wave Generation with the TMS32010". Digital Signal Processing Applications with the TMS32010 Family, Texas Instruments 1986, pag.269-290.

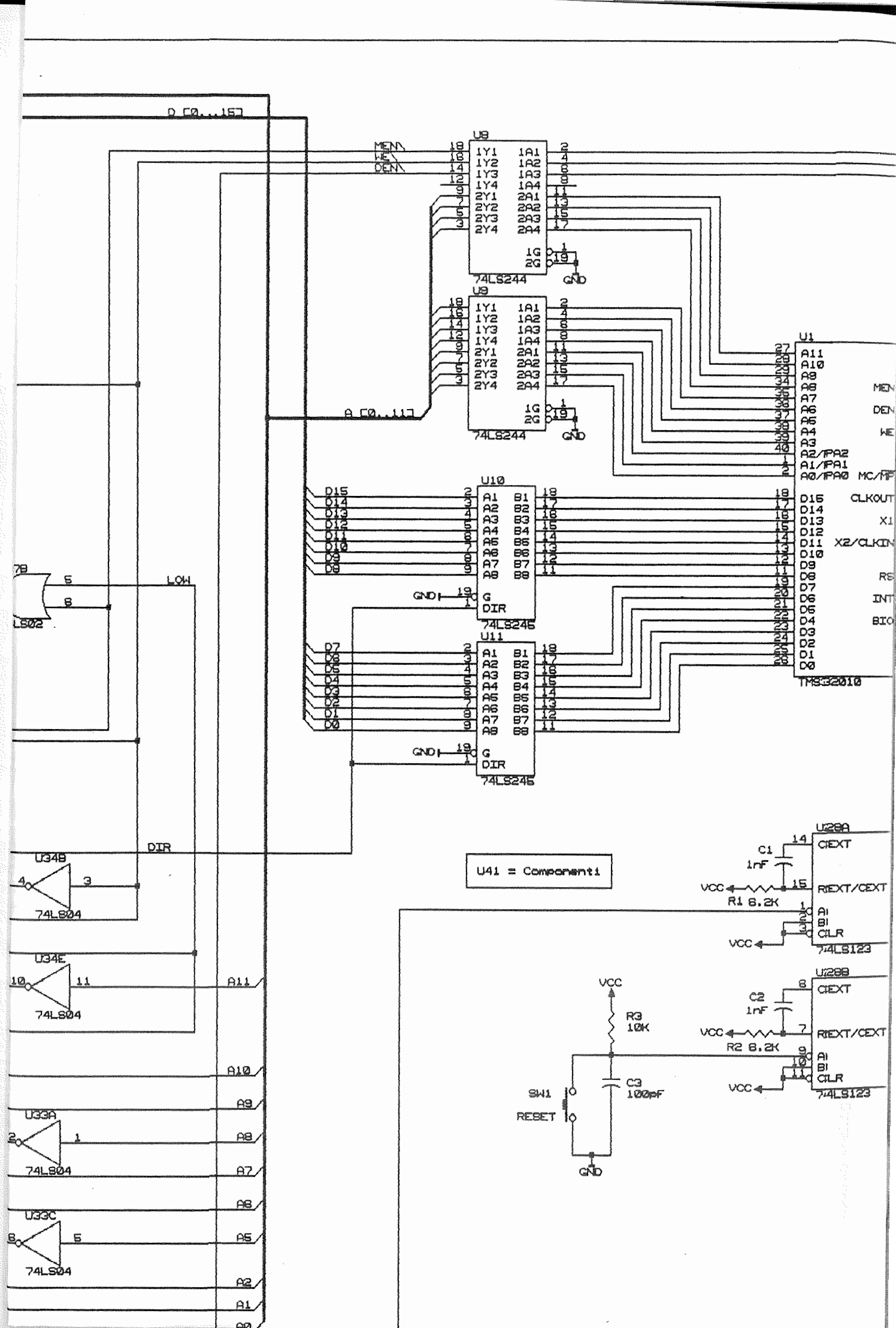


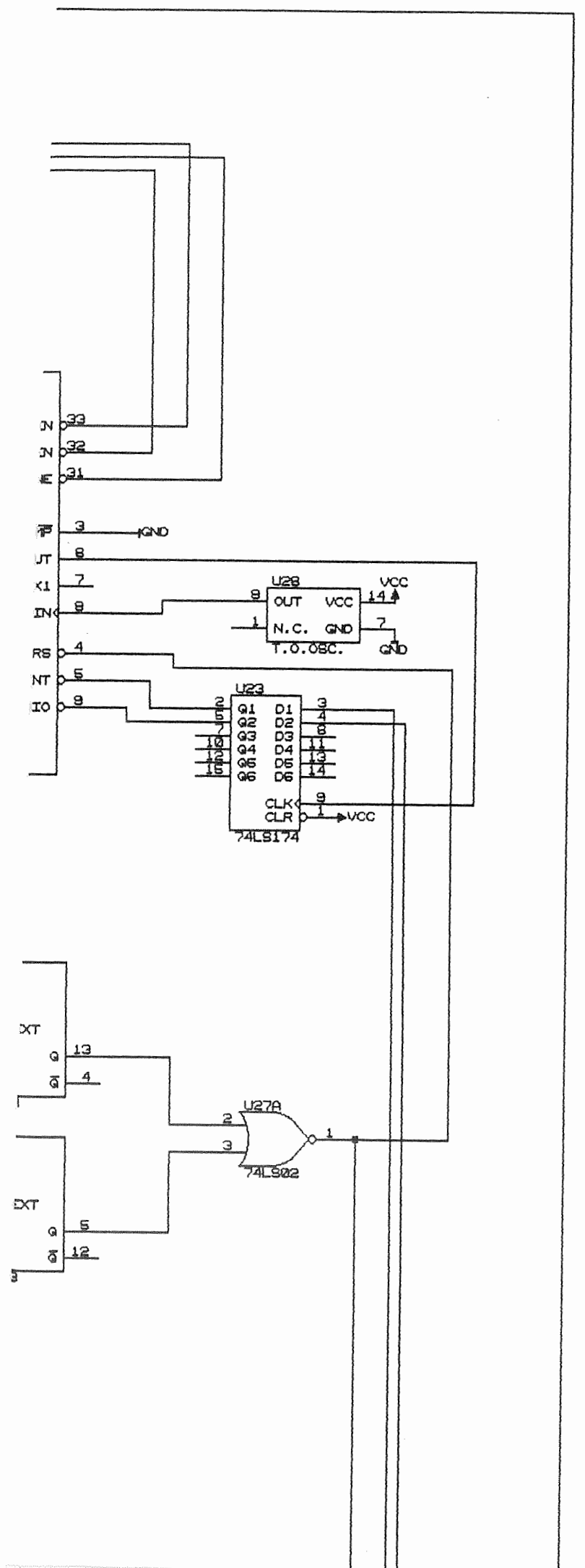
MP3A- MODULO PROCESSORE DIGITALE PER PC
Vista lato componenti
Piastra utilizzata 'Europa Media'
Aprile 1988

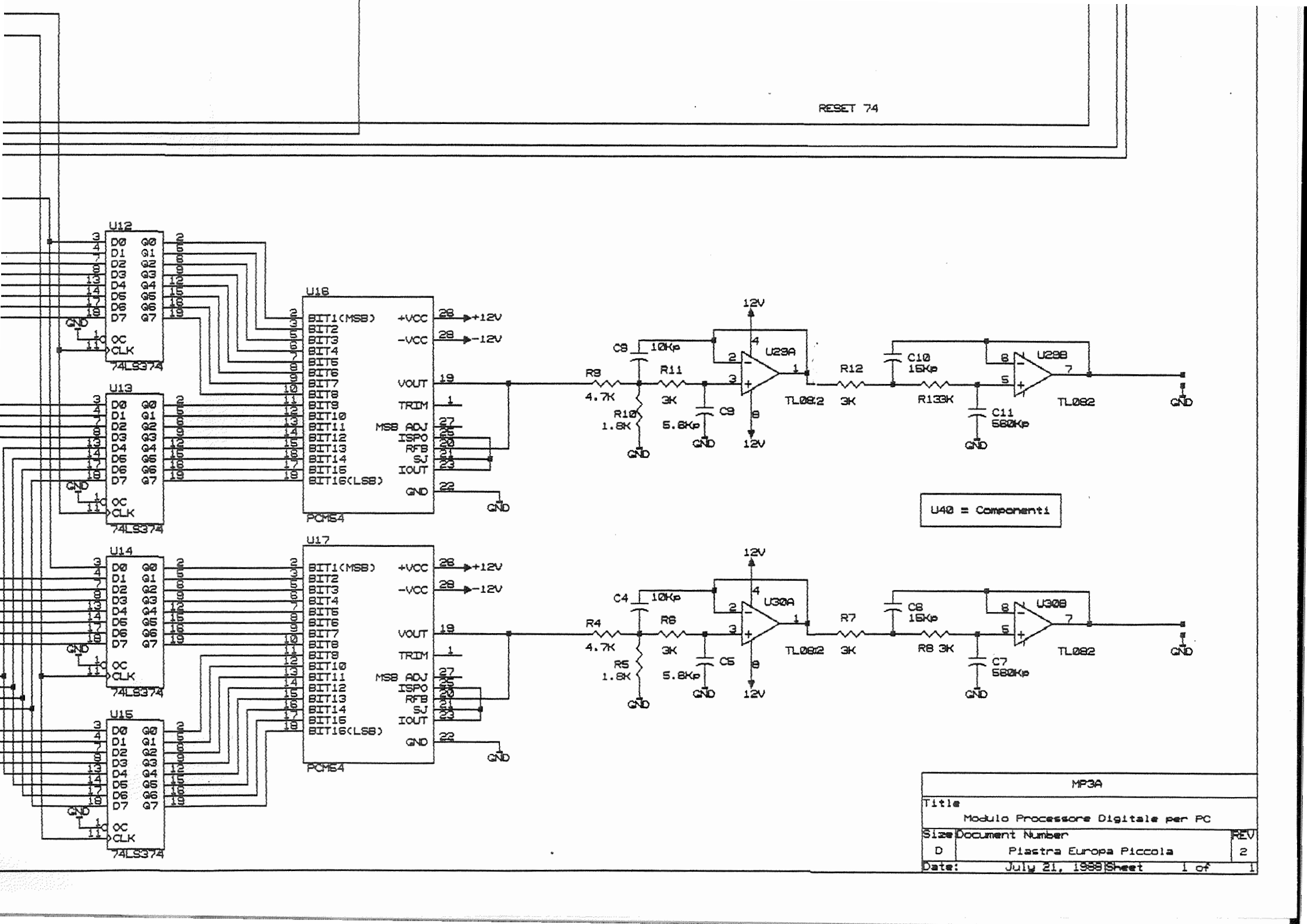
U1 =TMS32010
U2 =IMS1420
U3 =IMS1420
U4 =IMS1420
U5 =IMS1420
U6 =74LS288
U7 =74LS288
U8 =74LS244
U9 =74LS244
U10 =74LS245
U11 =74LS245
U12 =74LS374
U13 =74LS374
U14 =74LS374
U15 =74LS374
U16 =PCM54
U17 =PCM54
U18 =74LS373
U19 =74LS373
U20 =74LS373
U21 =74LS373
U22 =74LS138
U23 =74LS74
U24 =74LS155
U25 =74LS30
U27 =74LS02
U28 =74LS123
U29/30 =TL081
U31 =74LS240
U32 =74LS240
U33 =74LS04
U34 =74LS04
U35 =74LS00
U36 =74LS04
U37 =74LS02
U38 =74LLS74
U40, 41, 42 =Componenti dis
R =16 Res. da 27 ohm
R1 =3K
C =22Kp
X1 = Quarzo da 20Mhz

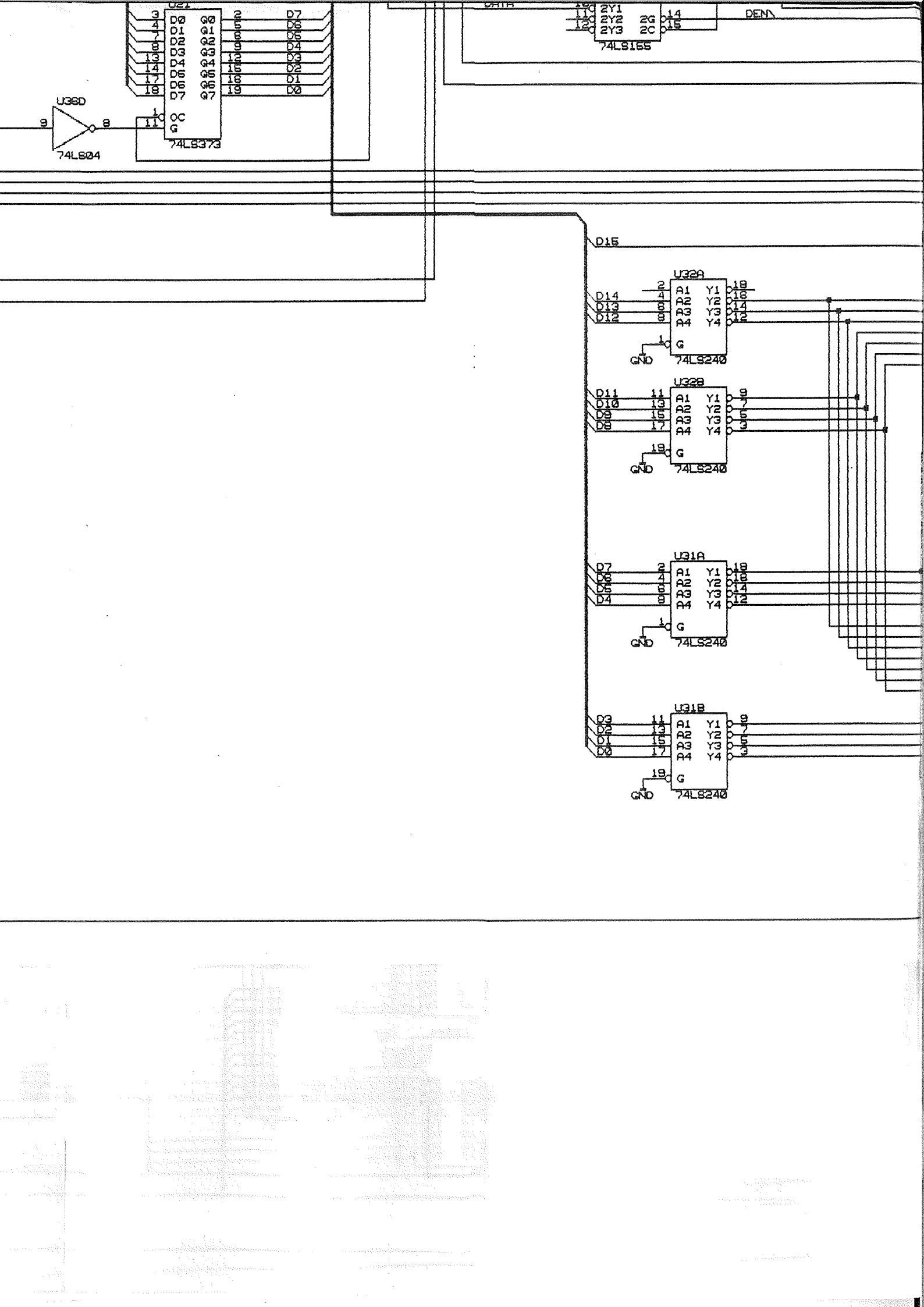

U2
U3
U4
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
I/O1
I/O2
I/O3
I/O4
E
W
IMS1420
CANON 11
CANON 12
CANON 13
CANON 14
CANON 15
CANON 16
CANON 17
CANON 18
D0
D7
U22
CANON 37
CANON 38
CANON 39
A2
A1
A0
A
B
C
G1
G2A
G2B
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7
GND
CANON32
Card Select
74LS138
ADDR_LOW
ADDR_H
DATA_L
DATA_H
BIO_SET
SW_RESET
U37A
74LS00
R/W
CANON25
U38A
CLK
PR
CL
Q
74LS74

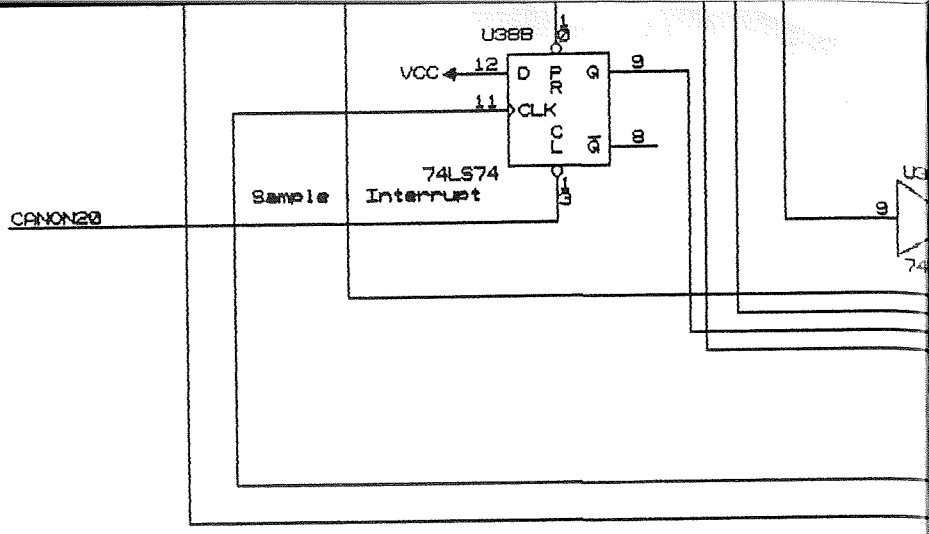

U5
IMS1420
U8
74S288
U7
74S288
U27C
74LS02
U27B
74LS02
U36C
74LS00
MEM WE
U35B
74LS00
U35D
74LS00
U35A
74LS00
U34C
74LS04
U34A
74LS04
U34F
74LS04
U18
74LS373
U19
74LS373
U20
74LS373
U33E
74LS04
U33F
74LS04
U33B
74LS04
U33D
74LS04
U25
74LS30
VCC
U24
74LS04
BIORES
INTRES
DACL
DACR
ADDRESS

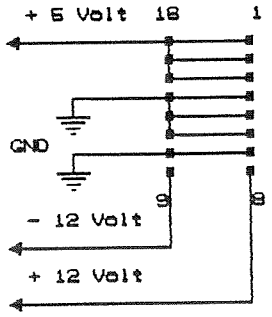

D [0..15]
A [0..11]
MEN
WE
DEN
U8
74LS244
U9
74LS244
GND
U1
TMS32010
MEN
DEN
WE
MC/MP
CLKOUT
X1
X2/CLKIN
RS
INT
BIO
U10
74LS245
U11
74LS245
DIR
LOW
U34B
74LS04
U34E
74LS04
U33A
74LS04
U33C
74LS04
A11
A10
A9
A8
A7
A6
A5
A2
A1
A0
U41 = Componenti
U28A
C1
1nF
R1 8.2K
VCC
CIEXT
REXT/CEXT
CLR
74LS123
U28B
C2
1nF
R2 8.2K
R3
10K
C3
100nF
SW1
RESET


IN 33
IN 32
IE 31
3 GND
UT 8
K1 7
IN 8
RS 4
NT 5
IO 9
U28
9 OUT VCC 14
1 N.C. GND 7
T.O.OSC.
VCC
GND
U23
2 Q1 D1 3
5 Q2 D2 4
7 Q3 D3 6
10 Q4 D4 11
12 Q5 D5 13
15 Q6 D6 14
CLK 9
CLR 1 VCC
74LS174
XT
Q 13
Q 4
U27A
2
3
1
74LS02
XT
Q 5
Q 12


RESET 74
U12
U13
U14
U15
74LS374
D0 Q0
D1 Q1
D2 Q2
D3 Q3
D4 Q4
D5 Q5
D6 Q6
D7 Q7
OC
CLK
GND
U16
U17
PCM54
BIT1(MSB)
BIT2
BIT3
BIT4
BIT5
BIT6
BIT7
BIT8
BIT9
BIT10
BIT11
BIT12
BIT13
BIT14
BIT15
BIT16(LSB)
+VCC
-VCC
VOUT
TRIM
MSB ADJ
ISPO
RFB
SJ
IOUT
+12V
-12V
12V
C8 10Kp
R8 4.7K
R11 3K
R10 1.8K
C9 5.8Kp
U29A
TL082
R12 3K
C10 15Kp
R133K
C11 582Kp
U29B
TL082
C4 10Kp
R4 4.7K
R6 3K
R5 1.8K
C5 5.8Kp
U30A
TL082
R7 3K
C6 15Kp
R8 3K
C7 582Kp
U30B
TL082
U40 = Componenti
MP3A
Title
Modulo Processore Digitale per PC
Size D
Document Number
Piastra Europa Piccola
REV 2
Date: July 21, 1988
Sheet 1 of 1


U21
D0 Q0
D1 Q1
D2 Q2
D3 Q3
D4 Q4
D5 Q5
D6 Q6
D7 Q7
OC
G
74LS373
D7
D6
D5
D4
D3
D2
D1
D0
U36D
74LS04
2Y1
2Y2
2Y3
2G
2C
74LS155
DEN
D15
U32A
A1 Y1
A2 Y2
A3 Y3
A4 Y4
G
GND
74LS240
D14
D13
D12
U32B
D11
D10
D9
D8
U31A
D7
D6
D5
D4
U31B
D3
D2
D1
D0





U42 = Zoccolo Tensioni