1. INTRODUCCIÓN:

El objetivo de este documento es introducir al lector en el formato empleado por el macroensamblador mcm, el cual genera ficheros que pueden ser usados como entradas por el driver de sonido de Micro Cabbin. Los ficheros de entrada aceptados por mcm son ficheros de texto ASCII, y diferencia entre mayusculas y minusculas, por ejemplo channel es diferente de CHANNEL. Los identificadores comienza por una letra que puede ir seguida de letras, diguitos o el simbolo _, hasta un maximo de 30 caracteres. Los identificadores terminan con un espacio en blanco, un salto de linea o con el carácter '?'.

Ej: f1 -> Valido
f_ -> Valido
1f -> Invalido
_f -> Invalido

Un numero es una secuencia de digitos de tamaño maximo 3 digitos, ya que el maximo valor representado es el 255. Una sentencia finaliza cuando se encuentre el salto de linea o el carácter : (este ultimo es usado para incluir varias sentencias en una sola linea). Para introducir comentarios en el fuente se emplea ';' el cual hace que mcm ignore el resto de caracteres hasta el final de la linea. Todas las lineas que comiencen por '#' seran ignoradas y no se contaran a la hora de indicar los numero de linea en los informes de error (su principal mision es la de permitir el uso de preprocesadores como el de C, lo cual permite incluir ficheros con definiciones y el uso de macros).

A continuacion se pasa a describir el formato basico de un fichero de entrada empleando la siguiente notacion:

- [...] Para elementos opcionales.
- * Para elementos que se pueden repetir cero o mas veces.
- + Para elementos que se pueden repetir una o mas veces.
- | Para indicar una opcion

Todas aquellas palabras reservadas por mcm se colocaran en negrita.

2. FORMATO:

El formato basico de un fichero de entrada es el siguiente:

```
[[var name1 [,name2]*] | [name=expr]]*          ;Declaracion e inicializacion de variables

[channel NUMBER | name                          ;Definicion de los canales
 [name=expr]                                            ;Asignacion de variables
 [command]                                              ;comandos del canal
 endc]+
```

write

La declaracion de todas las variables debe colocarse al principio del fichero, antes de la presencia de ninguna sentencia channel. Una vez declaradas se puede modificar su valor fuera de las sentencias channel o dentro de las mismas antes de la primera instrucción del canal.

2.1 Sentencias de declaración de variables:

Mediante esta sentencia se definen nuevas variables para su uso posterior en aquellos comandos que lo requiera. El formato basico empleado es el siguiente:

var name

Mediante lo cual se declarara la nueva variable name. Se pueden declarar varias variables en una sola sentencia usando para ello la coma:

var name1,name2,name3

2.2 Expresiones:

En una expresion puede aparecer numeros, variables y signos aritmeticos. Para la evaluacion de las mismas se usan las reglas de la aritmetica matematica definiendose la prioridad de los operadores tal como sigue:

Alta prioridad	( )
	* /
Baja prioridad	- +

Entre dos operandos de igual prioridad se evalua de izquierda a derecha. Para cambiar el orden de evaluacion se pueden emplear los parentesis.

2.3 Sentencias de asignación:

Una sentencia de asignacion tiene la siguiente forma:

label = expr

si la variable con nombre label existe se le asignara el valor resultante de evaluar la expresion.

2.4 Sentencia write:

Mediante la sentencia write se genera el fichero de datos listo para ser reproducido mediante el driver de microcabbin. odas las instrucciones posteriores en el fichero

seran ignoradas.

2.5 Sentencias channel:

Mediante esta sentencia se inicia la descripcion de un canal de musica. Para indicar el canal se puede usar un numero o bien una variable. El compilador tiene definidas una serie de variables para facilitar la comprension:

psg1 -> 1
psg2 -> 2
psg3 -> 3
fm1 -> 4
fm2 -> 5
fm3 -> 6
fm4 -> 7
fm5 -> 8
fm6 -> 9
fm7 -> 10
fm8 -> 11
fm9 -> 12

Una vez que el compilador se encuentre con la primera sentencia channel ya no se admitiran mas declaraciones de variables,aunque si se podran cambiar los valores de aquellas variables definidas.

Dentro de la definicion de un canal se pueden incluir las siguientes instrucciones:

- **loop**

Mediante este comando se define un bucle en la cancion. El final del bucle lo define el comando endl. Formato:

loop expr

donde expr debe generar un valor entre 0 y 9. Si se coloca el valor 0 se toma como un bucle infinito. El driver de microcabbin necesita que se coloque una instruccion de loop antes de tocar la primera nota (dicha condicion no es considereda por el compilador, asi que el musico es el encargado de asegurar dicha condicion).

- **loopl**

Igual que el anterior pero admite un valor entre 0 y 255 (ocupa dos bytes).

- **endl**

Sirve para definir el final de un bucle.

- **play**

Mediante esta instrucción se tocara una nota musical con con la fraccion de tiempo octava y volumen previamente definido en el canal (vease mas adelante los comandos oct,incoc,decoc,decv,incv,vol,time,times,timel). El formato del coamndo es el siguiente:

play expr1[# | &][,expr2[# | &]]*

Mediante la coma se pueden definir varias notas con los mismos parametros. Si despues de la expresion se coloca el caracter # se aumentara la nota en medio tono, y de igual manera si se coloca el simbolo & se decrementara medio tono. Se pueden colocar tantos # y & como se desee, pero teniendo en cuenta que siempre se estara dentro de la misma octava, por lo que se puede pasar del tono mas bajo al mas alto si se coloca un & (o al reves si se coloca un #). La instrcción admite un valor entre 0 y 33 (sin tener en cuenta los & y #) teniendo en cuenta que medio tono es correspondiente a un incremento o decrementoo de 3.

Para facilitar la edicion el compilador incluye las giguientes variables internas:

do,re,mi,fa,sol,la,si,c,d,e,f,g,h,a,b,C,D,E,F,G,A,B.

- **rest**

Mantiene apagado el canal el tiempo determinado por el anterior comando de temporizacion (time,times,timel).

- **oct**

Coloca la octava del canal. Formato:

oct expr

donde el valor final de expr debe estar entre 0 y 7

- **decoc**

Decrementa el valor la octava actual

- **incoc**

Incrementa el valor actual de la octava

- **vol**

Coloca el volumen del canal. Formato:

vol expr

donde el valor final de expr debe estar entre 0 y 15

- **decv**

Decrementa el volumen actual del canal.

- **incv**

Incrementa el volumen actual del canal.

- **times**

Coloca el numero de interrupciones que va a durar la siguiente nota. El formato del comando es el siguiente:

times expr

donde expr debe generar un valor entre 0 y 90.

- **timel**

Igual que el anterior pero admite un rango entre 0 y 255 (ocupa 2 bytes).

- **time**

Mediante este comando se coloca la duracion de la siguiente nota, pero indicandolo en tiempo real en funcion de las variables definidas por el compilador tempo y Hz. La primera de ellas define el numero de negras por segundo (en basic L4) y Hz indica la frecuencia de reproduccion (en un principio los valores validos son 50 o 60, aunque como se vera mas adelante tambien se pueden usar otros valores). Por defecto la variable tempo tiene el valor 120 y la variable Hz 60, aunque ambos parametros pueden ser modificados mediante una asignacion, incluso tomando valores diferentes para cada canal (aunque no se recomienda hacer esto ultimo, es mas aconsejable que todos los canales tengan el mismo valor de la variable tempo, pudiendo asi modificar la velocidad de reproduccion de una manera facil)

El compilador tiene definidas las siguientes variables para usar con este comando:

d1 = whole = redonda	(equivalente a L1 en BASIC)
d2 = half = blanca	(equivalente a L2 en BASIC)
d4 = quarter = negra	(equivalente a L4 en BASIC)
d8 = eigth = corchea	(equivalente a L8 en BASIC)
d16 = sixteenth = semicorchea	(equivalente a L16 en BASIC)
d32 = thirtysecond = fusa	(equivalente a L32 en BASIC)
d64 = sixtyfourth = semifusa	(equivalente a L64 en BASIC)

El formato del comando es el siguiente:

time expr[.]*

El caractar punto añade la mitad de la duracion del punto anterior anterior siendo el valor del primero la mitad de la duracion de la nota simple. Por ejemplo si se añade a una duracion .. se le sumara a la duracion inicial 3/4 de su valor. El compilador admite cualquier numero de puntos tras una nota, pero tras el sexto no añadiran tiempo alguna a la duracion.

El compilador decide en funcion del resultado si debe emplear el comando del driver **times** o **timel** por lo que el musico se tiene que despreocupar de los limites.

El poder usar una expresion para definir la duracion del tiempo añade una gran potencia al compilador, ya que permite usar facilmente los recursos musicales empleados en las partituras. Ejemplos:

- Ligaduras:

Mediante las ligaduras se suma la duracion de dos notas. Por ejemplo si se quiere conseguir la duracion de una negra mas una fusa seria tan facil como:

time d4+d32 (o bien time negra+fusa)

- Grupos irregulares de notas:

Mediante este recurso se quieren colocar un numero de notas en un tiempo determinado sin que haya una definicion exacta para el tiempo exacto de cada nota. Un ejemplo de grupo irregular puede ser un trisillo que seria un grupo irregular de 3 notas. Si se quiere que la duracion total del grupo sea equivalente al tiempo de una negra bastaria con escribir algo como:

time d4/3 (o bien time negra/3)

- **noise**

Mediante este comando se generara una señal de ruido en un canal PSG (si el canal es un canal FM se producira un error). El canal generara la señal de ruido a la frecucuencia que se le pase como parametro durante indicado por la anterior instrucción de temporizacion.

- **inst**

Mediante este comando se coloca el tipo de instrumento FM en el canal (si el canal es un canal PSG se producira un error). El formato es el siguiente:

inst expr

donde expr debe generar un valor comprendido entre 0 y 40

El chip OPLL tiene definidos 15 instrumentos internamente, permitiendo definir un instrumento adicional. Para seleccionar el instrumento definido por software se debe indicar instrumento 0 (o bien emplear la variable original). En este comando los instrumentos del 1 al 15 son los definidos internamente por el OPLL mientras que del 16 al 51 son instrumentos software. Del 15 al 51 estan definidos en el propio driver, mientras que del 52 al 63 en principio estan en blanco a menos que se redifinan mediante las llamadas adecuadas de configuracion del driver. Hay que resaltar que debido a que solo se puede usar un instrumento software si se cambia el instrumento software en un canal entonces todos los canales que esten reproduciendo ese instrumento cambiaran el sonido producido.

El compilador internamente define las siguientes variables para los instrumentos internos

violin
guitar
piano
flute
clarinet
oboe
trumpet
organ
horn
synth
harp
vibra
synthbass
acustic
electric

Para los instrumentos software definidos en el propio driver se pueden emplear las siguiente variables:

- **instl**

Igual que el anterior pero ocupa dos bytes (Se añade por compatibilidad con el driver, pero no debe usarse).

- **adsr**

Define para un canal PSG una variacion en la envolvente de amplitud. El driver lleva incorporadas 16 variaciones distintas, que se pueden combinar entre ellas para conseguir la curva adsr deseada. Cada variacion viene definida por 3 parametros:

- Tiempo entre modificacion del valor de amplitud.
- Magnitud del cambio en cada modificacion de la envolvente.

- Valor de estabilizacion.

El tramo de envolvente comenzara en el valor de volumen definido para el canal y continuara variando hasta alcanzar el valor definido por el tipo de variacion, momento que se queda estable (a menos que se defina un tiempo de mantenimiento mediante la instrucción **divsus**). A continuacion se dara una descripcion de cada uno de los tramos disponibles en el driver (los valores iniciales se han puesto de manera arbitraria o aquellos que generasen una grafica mas clara):

Tramo 0:
Tiempo de actualizacion: 10 interrupciones
Valor del escalon: -1
Valor de mantenimiento: 7

Tramo 1:
Tiempo de actualizacion: 2 interrupciones
Valor del escalon: -1
Valor de mantenimiento: 9

Tramo 2:
Tiempo de actualizacion: 2 interrupciones
Valor del escalon: -1
Valor de mantenimiento: 6

Tramo 3:
Tiempo de actualizacion: 1 interrupcion
Valor del escalon: 1
Valor de mantenimiento: 8

Tramo 4:
Tiempo de actualizacion: 2 interrupciones
Valor del escalon:-1
Valor de mantenimiento:10

Tramo 5:
Tiempo de actualizacion: 1 interrupcion
Valor del escalon: -1
Valor de mantenimiento: 0

Tramo 6:
Tiempo de actualizacion: 2 interrupciones
Valor del escalon: 1
Valor de mantenimiento: 9

Tramo 7:
Tiempo de actualizacion: 10 interrupciones

Valor del escalon: -1
Valor de mantenimiento: 0

Tramo 8:
Tiempo de actualizacion: 4 interrupciones
Valor del escalon: -1
Valor de mantenimiento: 0

Tramo 9:
Tiempo de actualizacion: 2 interrupciones
Valor del escalon: -1
Valor de mantenimiento: 2

Tramo 10:
Tiempo de actualizacion: 15 interrupciones
Valor del escalon: -1
Valor de mantenimiento: 0

Tramo 11:
Tiempo de actualizacion: 3 interrupciones
Valor del escalon: -1
Valor de mantenimiento: 11

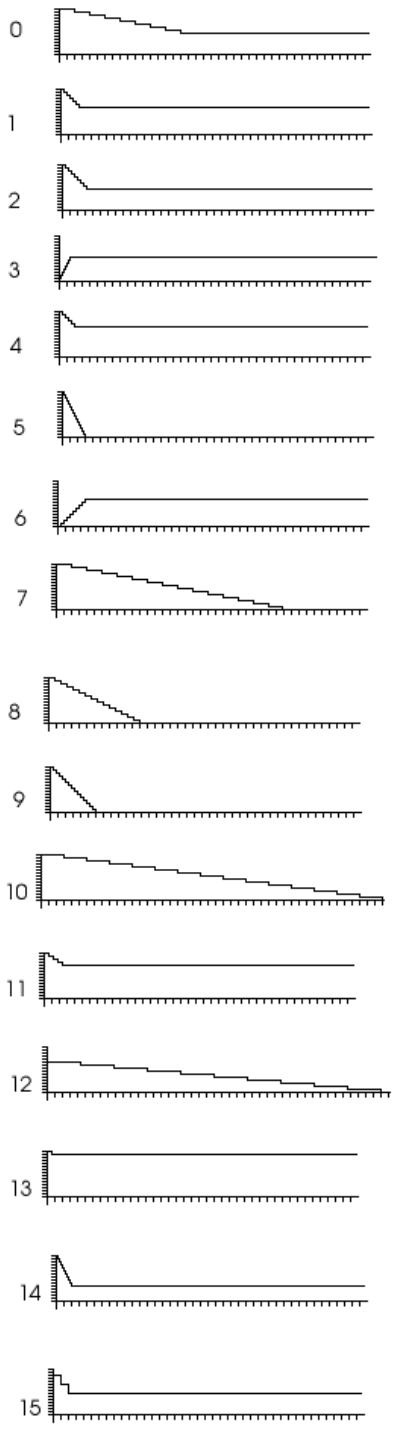
Tramo 12:
Tiempo de actualizacion: 22 interrupciones
Valor del escalon: -1
Valor de mantenimiento: 0

Tramo 13:
Tiempo de actualizacion: 3
Valor del escalon: -1
Valor de mantenimiento: 14

Tramo 14:
Tiempo de actualizacion: 1 interrupcion
Valor del escalon: -1
Valor de mantenimiento: 5

Tramo 15:
Tiempo de actualizacion: 5 interrupciones

Valor del escalon: -3
Valor de mantenimiento: 7



Estos tramos se pueden combinar entre si para generar la envolvente deseada. Ejemplo:

```
channel psg2

    loop 0

    vol 0           ;Atack
    adsr 3
    times 8
    noise 0

    vol 8
    times 12
    adsr 8
    noise 0         ;Decay

    vol 5
    adsr 5
    times 5
    noise 0         ;release

    endc

write
```

- **divsus**

Marca el tiempo del timbre del sonido (es decir durante cuanto tiempo se oye el instrumento, que no tiene porque ser igual a la duracion de la nota. El formato empleado es el siguiente:

divsus expr

- **linef**

Mediante este comando se consigue una variacion lineal de la frecuencia de una nota. Es importante destacar que es solo aplicable al comando play (no se puede usar con noise en canales psg) y que afecta solo a la duracion de una nota. Si nos encontramos en un canal fm y despues del **linef** colocamos un silencio (**rest**) el canal tocara durante la durecion del mismo la ultima frecuencia colocada por el **linef**, por lo que se recomienda colocar siempre un **play** (en caso de querer hacer un silencio se puede emplear un play con vol 0).

El formato empleado es el siguiente:

linef expr

donde expr debe generar un valor entre 0 y 15. Dicho valor correspondera a una de las entradas de la tabla sguiente:

	PSG	***FM***
0	-16/1	-266/1
1	+42/1	+42/1
2	+5/1	+5/1
3	-5/1	-261/1
4	-2/1	-258/1
5	-1/1	-257/2
6	+3/2	+3/2
7	+2/3	+2/3
8	-2/1	-258/3
9	+50/1	+50/1
10	-73/1	-329/1
11	-68/1	-324/1
10	+20/1	+20/1
13	+3/1	+3/1
14	-3/1	-259/1
15	0	0

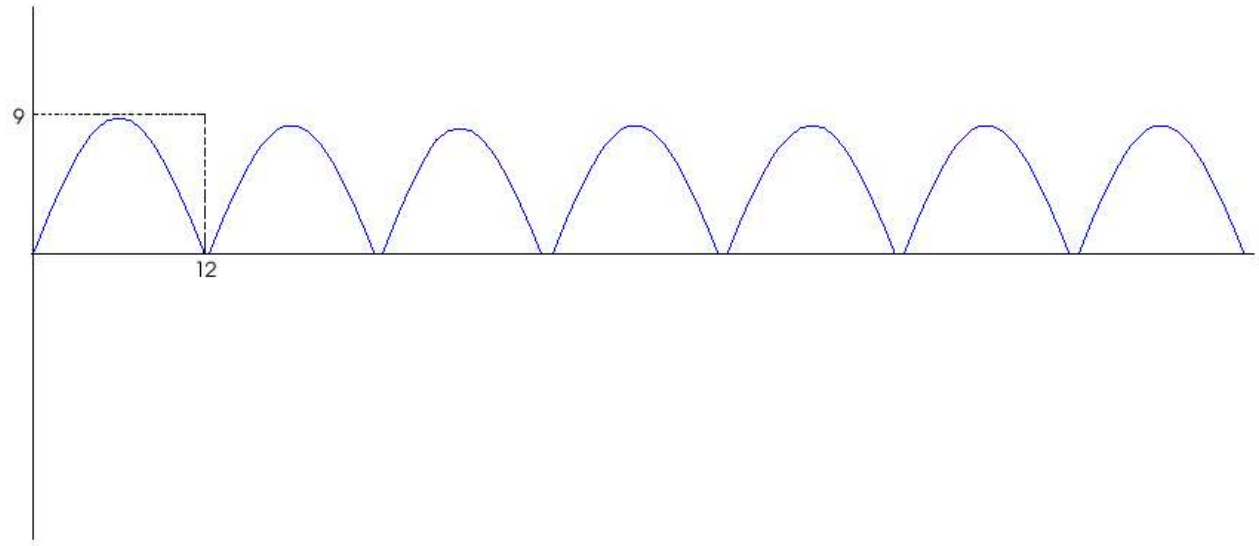
El valor recogido es la pendiente con la que se modifica la frecuencia. Este comando es idoneo para hacer fades entre notas.

- **envl**

A continuacion se da una lista de las envolventes incluidas en el driver:

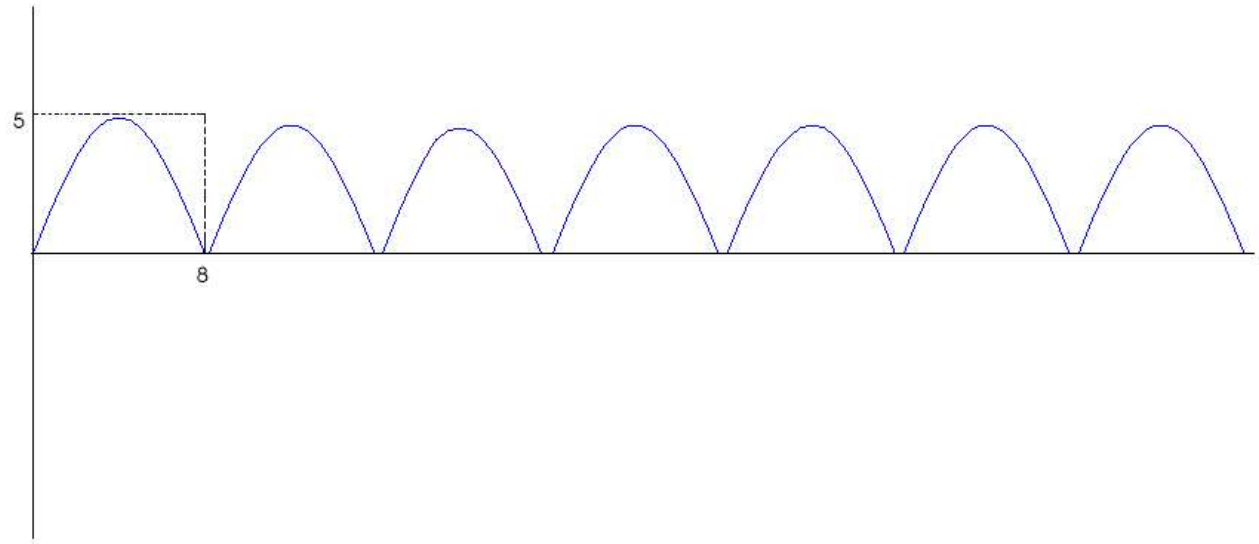
Envolvente 0:
Esta envolvente realmente tan solo sirve para eliminar otra envolvente previamente puesta.

Envolvente 1:



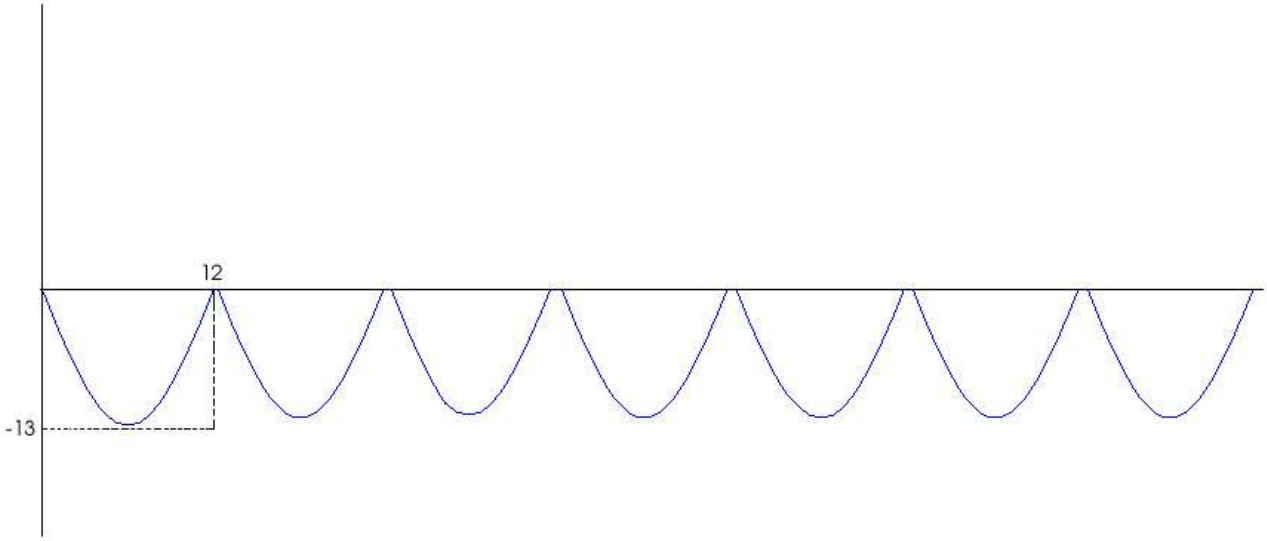
No es reiniciada en cada nota
Afecta a: Frecuencia

Envolvente 2:



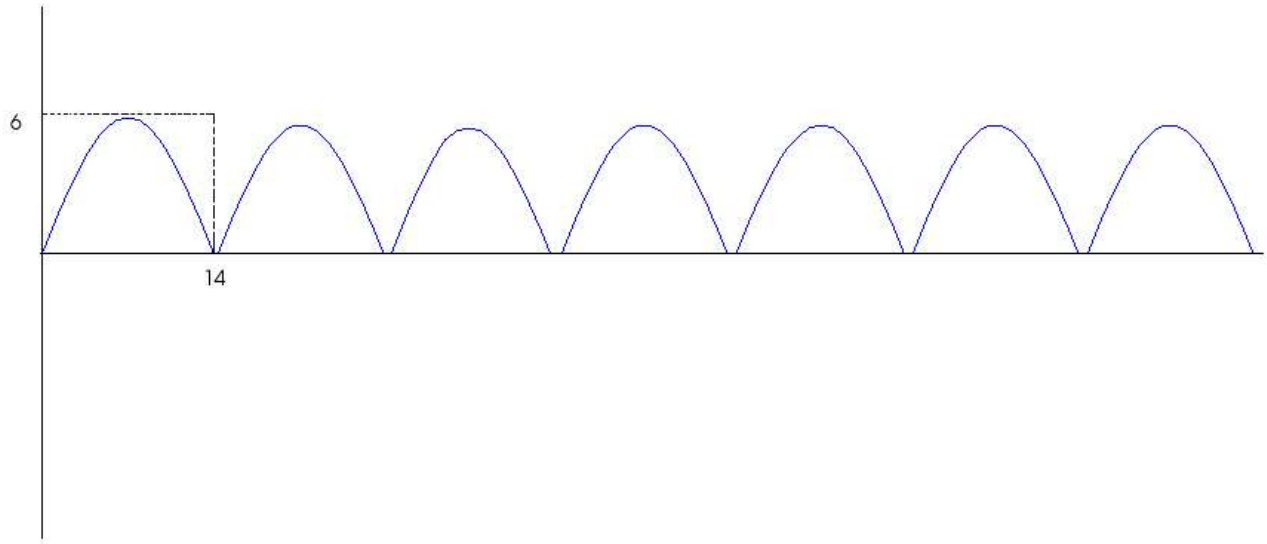
Es reiniciada en cada nota
Afecta a: Frecuencia

Envolvente 3:



Es reiniciada en cada nota
Afecta a: Frecuencia

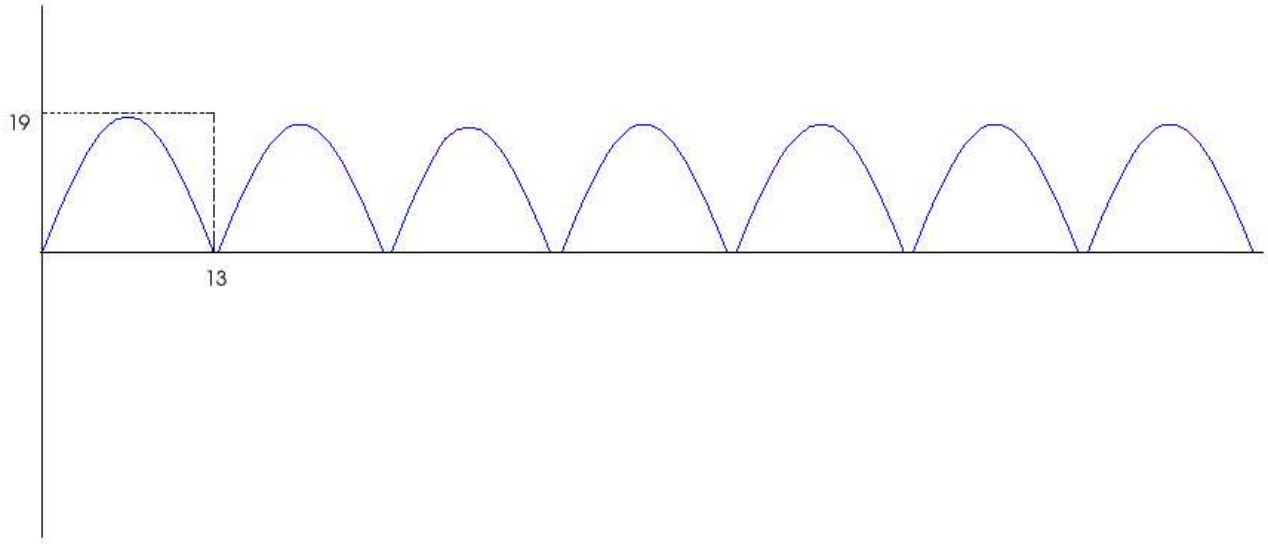
Envolvente 4:



No reinicia en cada nota

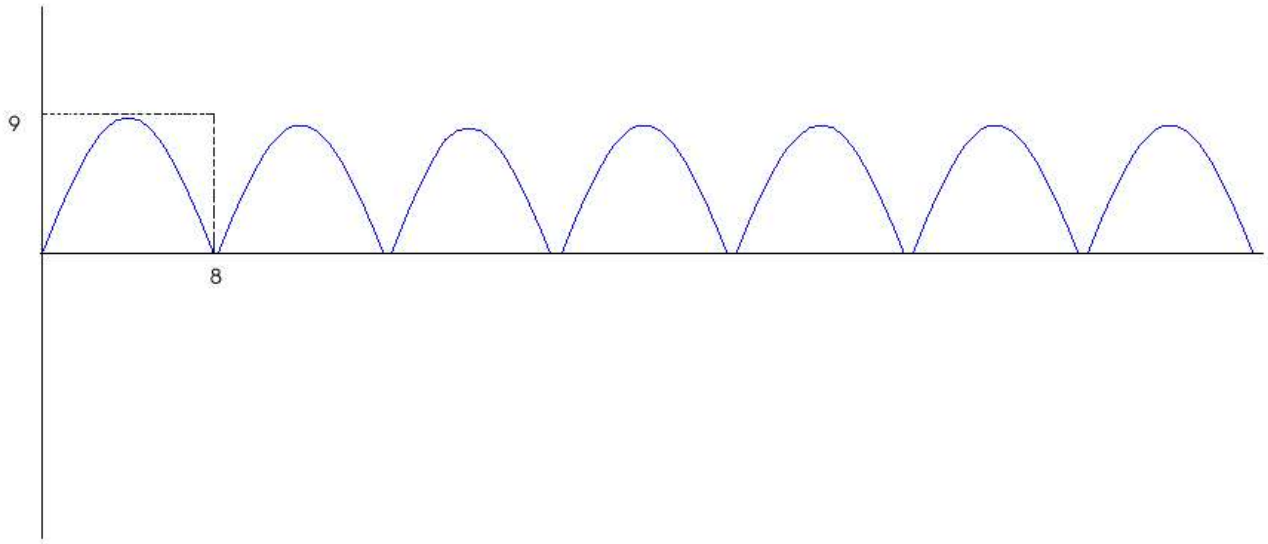
Afecta a: Volumen

Envolvente 5:



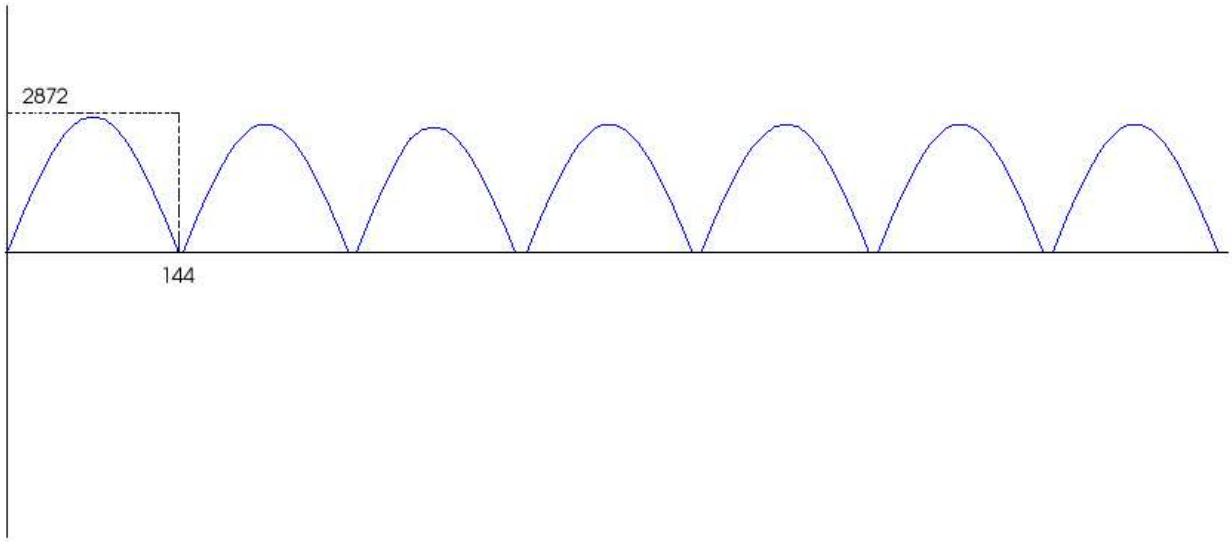
Es reiniciada en cada nota
Afecta a: Frecuencia

Envolvente 6:



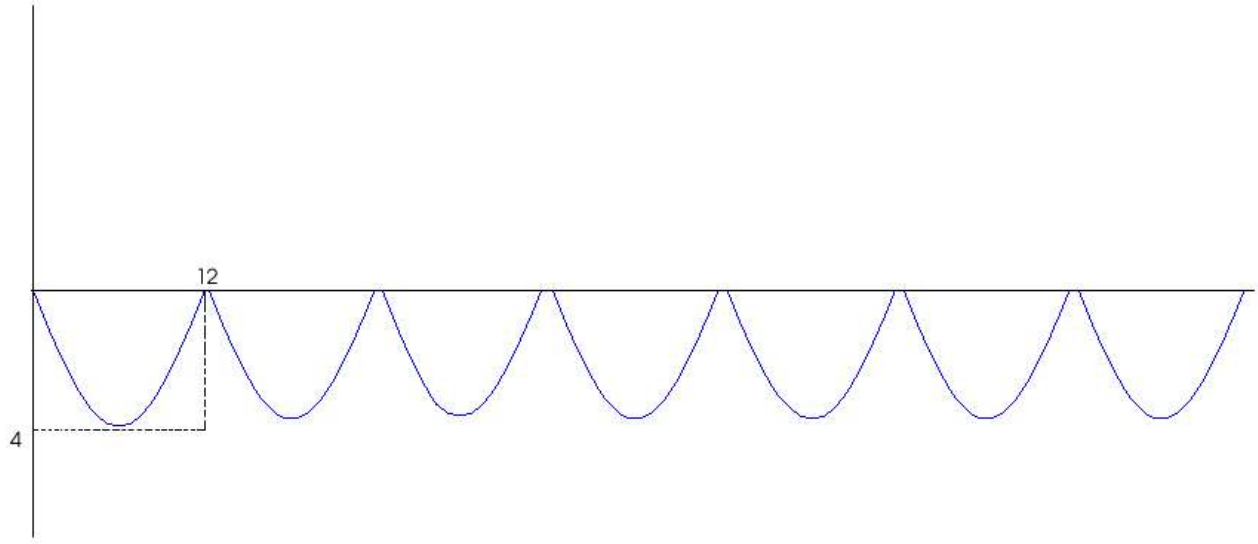
Es reiniciada en cada nota
Afecta a: Frecuencia

Envolvente 7:



Es reiniciada en cada nota
Afecta a: Frecuencia

Envolvente 8: Igual que la envolvente 1
Envolvente 9: Igual que la envolvente 1
Envolvente 10:



No reinicia en cada nota
Afecta a: Frecuencia

- **regfm**

Este comando se usa para escribir en un registro del OPLL. El formato empleado es el siguiente:

regfm expr1,expr2

Donde la primera expresion indica el registro y la segunda el valor que se quiere escribir.

- **par**

Mediante este comando se modifican algunos parametros internos del driver. Admite los siguientes subcomandos:

chnoff – Apaga el canal.
chnon – Enciende el canal.
suson – Activa la envolvente hardware en los canales FM. (FM)
susoff – Desactiva la envolvent hardware en los canales FM. (FM)
sd_hh – Coloca el canal en modo sd_hh (ver abajo). (FM)
tom-tcy – Coloca el canal en modo tom_tcy (ver abajo). (FM)
noise – Activa el ruido en el canal. (PSG)
noiseoff – Desactiva el ruido en el canal. (PSG)
deco – Mediante este parametro se puede modificar el numero de unidades de tiempo asignadas a cada interrupción. En principio a cada interrupción se le asignan 75 unidades de tiempo y cada paso de decodificación consume 75 unidades de tiempo. Por eso en un principio se puede decir que en cada interrupcion equivale a la minima unidad de temporización del driver. Con esta instrucción se pueden añadir unidades de tiempo a las interrupciones de tal manera que lo anterior no se cumple. Si por ejemplo añadimos 25 unidades de tiempo a cada interrupción tendremos que en 3 interrupciones se generaran 300 unidades de tiempo, que dividido entre 75 da 4, por lo que en 3 interrupciones se producirán 4 unidades de temporización en lugar de 3. Mediante este parametro se puede modificar la temporización, permitiendo que la variable Hz tome valores diferentes de 50 o 60 (aunque claro, tan solo sera una aproximación). Formato usado:

par deco expr

tamp – Tiempo antes de que empiece a aplicar la modulación colocada por la instrucción **envl (**durante el tiempo indicado sonara el sonido como si no hubiera modulación en el canal). Formato usado:
0
par tamp expr

ritmon – Activa el modo ritmico del FM. (FM)
melon – Activa el modo melódico del FM (FM)

Modo ritmico del FM

El FM tiene un modo ritmico en el cual se añaden 5 voces de instrumentos de percusión en lugar de los dos ultimos instrumentos. El driver para manejar este modo solo permite usar 3 instrumentos de percusión de los 5 que se podrian usar. Para elloel instrumento BD (bass drum) se maneja como un instrumento más en el canal 6 y los canales 7 y 8 se pueden colocar en uno de los siguientes modos:

	SD-HH	***TOM-TCY***
Canal 7	SD	TOM
Canal 8	HH	TCY

SD – Snare drumm.
TOM – Tom-ton.
TCY – Top Cymbal.
HH – High hat.

Este planteamiento facilita la gestión interna del driver pero hace que no se puedan usar los 5 instrumentos de percusión a la vez (ni tampoco las parejas SD-HH y TOM-TCY).

- **endc**

Mediante esta instrucción de deja bloqueado el canal, de tal menera que no continua decodificando el resto de comandos del canal. Una vez que el compilador se encuentre con este comando considerara que el canal ha concluido. Es necesario colocar el comando endc para terminar el canal.