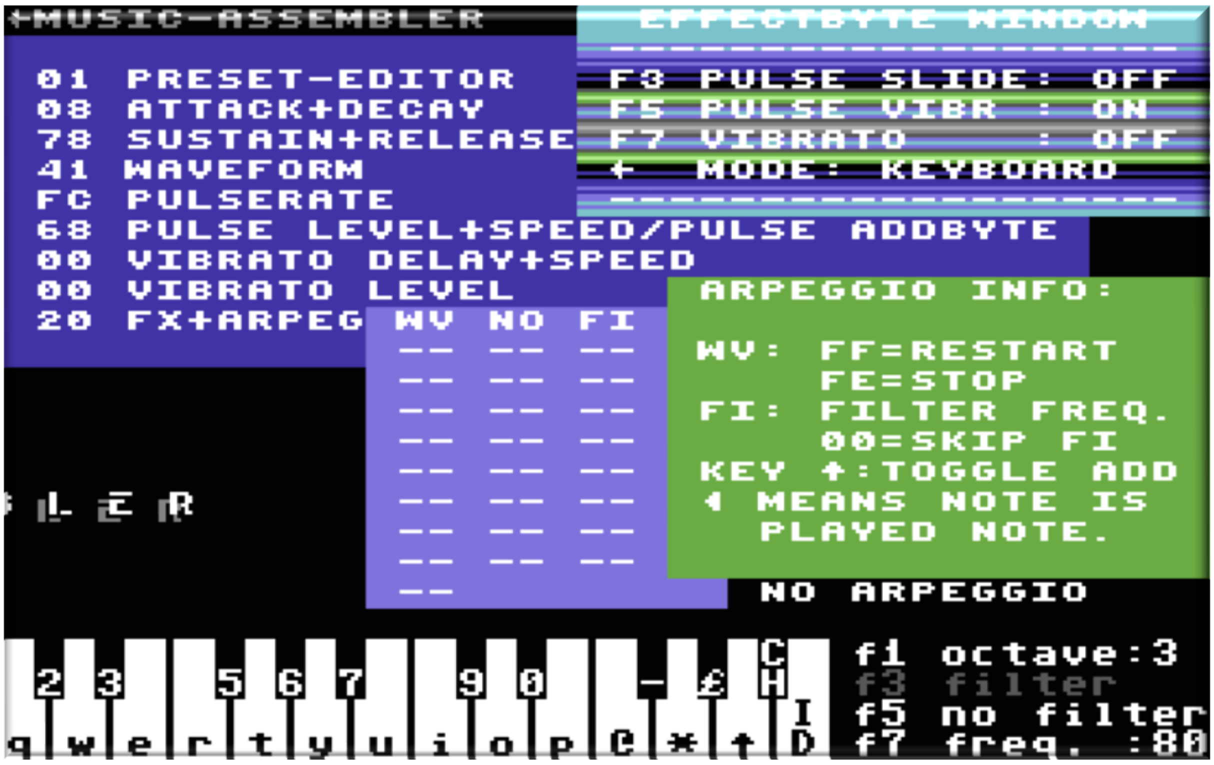
Music Assembler 1.0 User Manual



```
←MUSIC-ASSEMBLER          EFFECTBYTE WINDOW
                          -------------------
01 PRESET-EDITOR          F3 PULSE SLIDE: OFF
08 ATTACK+DECAY           F5 PULSE VIBR : ON
78 SUSTAIN+RELEASE        F7 VIBRATO    : OFF
41 WAVEFORM               ←  MODE: KEYBOARD
FC PULSERATE              -------------------
68 PULSE LEVEL+SPEED/PULSE ADDBYTE
00 VIBRATO DELAY+SPEED
00 VIBRATO LEVEL          ARPEGGIO INFO:
20 FX+ARPEG WV NO FI
            -- -- --      WV: FF=RESTART
            -- -- --          FE=STOP
            -- -- --      FI: FILTER FREQ.
            -- -- --          00=SKIP FI
            -- -- --      KEY ↑:TOGGLE ADD
            -- -- --      ◄ MEANS NOTE IS
            -- -- --        PLAYED NOTE.
            -- -- --
            --              NO ARPEGGIO
```

Version 0.01b

Written by Marco Swagerman

Table of contents

Prologue

I'd like to take the opportunity to thank Oscar Giesen for creating the finest C64 SID music editor of the 80's together with me… Did you know you can download (and pay for) ringtones with MC and OPM music these days? OMG there's people trying to make money off our asses!

I think it must have been somewhere in early 1986 when I started attempts to create music on the Commodore 64 and its marvelous SID sound chip. In those days a lot of the games and demos on this computer started having really good music and I found myself wanting to write such stuff too… At the time there weren't any soundtrackers or alike, just some experimental and very limited music programs. Hearing tunes by my heroes in those days (Rob Hubbard, Martin Galway, Ben Daglish to name a few) Inspired me to check out what they were doing and how they did it. Being a moderately skilled assembly programmer I wrote a couple of simple music player routines through a "monitor" (debugger) program. My attempts were ridiculously poor and I wasn't happy with the results at all…

After later writing a couple of tunes in Chris Huelsbeck's "Soundmonitor" and still not being happy with the result me and my long time buddy Oscar Giesen (OPM) decided to really go for it and attempt to write the be-all end-all music editor… Sure enough, when Music Assembler appeared it did cause quite a stir and I know of at least a bunch of people who switched to it as their main tool for music creation. You can probably imagine me and Oscar were well pleased with the result at the time. Music Assembler was kicking some serious butt!

Looking back at the program it seems crude. You edit music data in notes and lots of hexadecimal numbers. For today's computer users it will seem an impossible beast to deal with but back in its day Music Assembler was a top notch and quite user friendly editor with its windowed interface and easy to work with file management. I am still very proud of the fact two boys from a small town somewhere in the Netherlands managed to create such a magnificent product.

Music Assembler was at the time being published by Markt+Technik. who sold quite a large number of copies, mainly in Germany. From the money earned me (MC) and my buddy OPM got our Amiga and Atari ST computers amongst other things. Creating Music Assembler and the experience of our creation being such an influence on the C64 'scene' shaped part of our youths and I am still thrilled when I see the program somewhere every single time… Hearing a C64 really play music still is and will always remain to be a blast.

As with most applications in those days, a clear user manual was never created. I hope some people will appreciate this (21 years) late attempt to clarify the many possibilities Music Assembler has to offer in creating some of your best classic and modern day SID tunes.

Marco Swagerman
September 2010

Music Assembler mechanics

An odd little editor.

Music Assembler is not like your everyday soundtracker program. The layout and mechanics are a lot different from this line of music editors. You're advised to not compare Music Assembler with any of the "trackers" but look at it from a different perspective.

On the C64 there's a couple things which are really important about the music files you create:

1. The memory footprint must be small as there's very limited memory available on a C64.
2. Playing the music must use the least amount of resources possible.

With this in mind the mechanics and music player routine were created.

Sound presets, effects and arpeggios.

Before anything else a composer needs sounds and presets. Music Assembler's presets were designed to take very little memory. Each preset holds just 8 bytes of information. Within each preset information about ADSR envelope, waveform, vibrato and pulse waveform modulation is stored. A preset can also be linked to one of the arpeggios you create. There can be 32 presets and 16 arpeggios in total within one Music Assembler file.

Editing an arpeggio is done in the same screen as editing the preset but bear in mind they are not one and the same. If an arpeggio is being shared by multiple presets music assembler will tell you it is a "used arpeggio".

Tracks and Sequences.

This is where Music Assembler differs most from tracker style editors. As the SID chip has three channels/Oscillators there's also three separate tracks playing music through these channels. Instead of having just one track and using polyphonic sequences which play through all three of the C64's voices we chose the more flexible approach of having three separate tracks which hold information on which sequences to play on which channel, how many times to repeat them and if needed transpose them up to over an octave higher.

Of course this means Music Assembler's sequences are monophonic thus easy and fast to work with. Within sequences you also manage the slide (portamento) and low pass filtering effects.

Tracks in Music Assembler are edited fully in hexadecimal number format, sequences are edited through music instructions with all extra parameters being edited also in hexadecimal numbers. Better get your HEX codes straight before editing music…

Why the term “Assembler”?

Music Assembler does not simply store the music data in the format you edit it. Instead it assembles a complete executable music file which incorporates the player routine and code to have it play automatically. This means Music-Assembler tunes can be played even through BASIC. For instance when you saved your song with a starting address of $C000 you can play the music by loading it and typing “SYS 49152” from the Commodore Basic command line. Sadly we didn’t incorporate any code to stop the music from Basic at the time. ☺

Besides incorporating the player routine, Music Assembler organizes and stores music data in a compressed format which is not used by the editor. The instructions typed within the sequence editor is assembled into intricate, to many people unreadable data which is disassembled by the player routine while playing the music. Small memory footprint was the key concern when creating the data format.

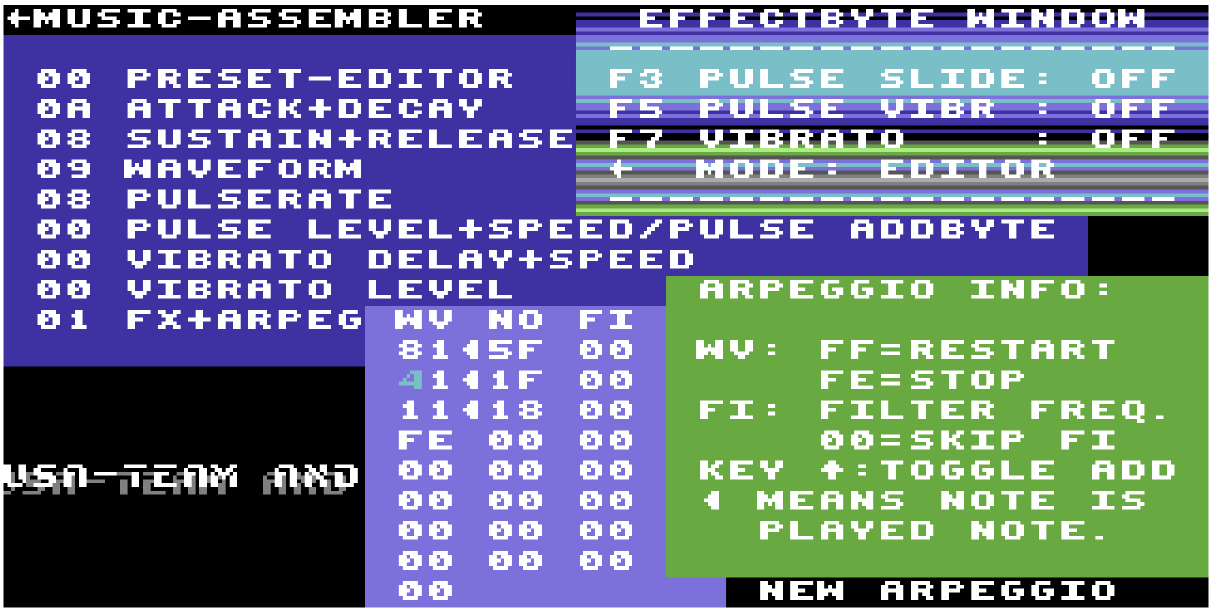
Song and Presets files.

When creating tunes (or songs if you like) you tend to develop a certain style of using the sounds the SID chip is capable of. You will find a lot of the presets you use will remain largely the same in multiple projects. For this reason Music Assembler allows you to save just the presets of a song into a “p.filename” file. This also allows you to make a backup of your presets before editing them within your song. You will be able to play back your music using different preset banks and so tweak the overall mix of presets until you’re completely happy.

A complete song can be saved as an “s.filename” file and be relocated to anywhere in memory (between $0400 and $FF00 hex) during the save process. This gives you an unprecedented amount of flexibility to incorporate your music in games or demos.

Editing.

Preset Editor.



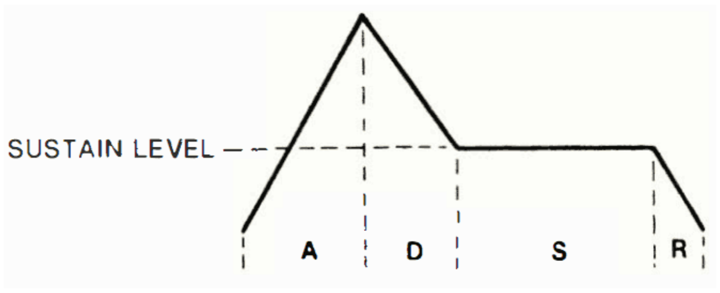
```
←MUSIC-ASSEMBLER      EFFECTBYTE WINDOW

 00 PRESET-EDITOR     F3 PULSE SLIDE: OFF
 0A ATTACK+DECAY      F5 PULSE VIBR : OFF
 08 SUSTAIN+RELEASE   F7 VIBRATO    : OFF
 09 WAVEFORM           ← MODE: EDITOR
 08 PULSERATE
 00 PULSE LEVEL+SPEED/PULSE ADDBYTE
 00 VIBRATO DELAY+SPEED
 00 VIBRATO LEVEL        ARPEGGIO INFO:
 01 FX+ARPEG WV NO FI
             81◂5F 00   WV: FF=RESTART
             41◂1F 00       FE=STOP
             11◂18 00   FI: FILTER FREQ.
             FE 00 00       00=SKIP FI
USA-TEAM AND 00 00 00   KEY ↑:TOGGLE ADD
             00 00 00   ◂ MEANS NOTE IS
             00 00 00     PLAYED NOTE.
             00 00 00
             00           NEW ARPEGGIO
```

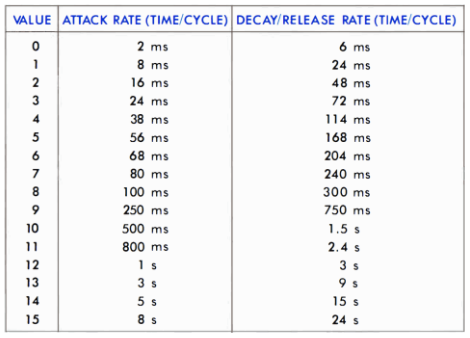
After loading and starting ,Music Assembler starts in the preset editor. This is where you create the voices that will sing your song for years to come.

You can move around through the editable parameters by using the cursor keys. Typing in two digit hexadecimal numbers edits these parameters. For ease of use you can toggle effects on/off by pressing the stated function keys. These parameters are the ADSR envelope, waveform, pulse waveform rate, pulse effects, vibrato effects and the arpeggio to use.

ADSR envelope.

Editing the envelope is done through two pairs of hexadecimal digits. To further illustrate how this ADSR envelope works I have taken the liberty of borrowing the information from the original C64 programmer's manual. The stated values 0 to 15 translate to hexadecimal values $0-$F.

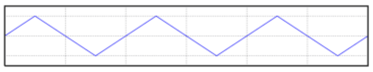


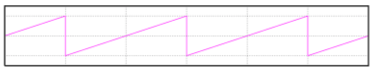


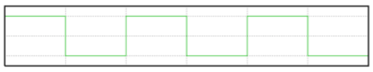
VALUE	ATTACK RATE (TIME/CYCLE)	DECAY/RELEASE RATE (TIME/CYCLE)
0	2 ms	6 ms
1	8 ms	24 ms
2	16 ms	48 ms
3	24 ms	72 ms
4	38 ms	114 ms
5	56 ms	168 ms
6	68 ms	204 ms
7	80 ms	240 ms
8	100 ms	300 ms
9	250 ms	750 ms
10	500 ms	1.5 s
11	800 ms	2.4 s
12	1 s	3 s
13	3 s	9 s
14	5 s	15 s
15	8 s	24 s

Waveform shapes, ring modulation and waveform synchronisation.

The SID chip is capable of producing any of three basic waveforms and also generates white (random) noise. Because of SID's brilliant layout it is capable of producing mixtures of these waveforms as well which can create some interesting sounds indeed. These are the basic waveforms:


Triangle waveform


Sawtooth waveform


Pulse waveform

Choosing a waveform is done by editing the two digit "waveform" hexadecimal number in the preset according to this table:

BIT	Value	Function	BIT	Value	Function
7	$80	Enable random noise	3	$08	Disable oscillator*
6	$40	Enable pulse waveform	2	$04	Enable ring modulation
5	$20	Enable sawtooth waveform	1	$02	Enable synchronization
4	$10	Enable triangle waveform	0	$01	Envelope gate
				(1 starts Attack/Decay/Sustain, 0 starts Release)	

Basically you select a waveform type and enable the gate to make the preset work. As an example a value of $21 will select the sawtooth waveform and enable ADSR triggering. In some rare cases you may not want to cause triggering but usually you do.

A waveform value of $51 will select both the pulse and triangle waveform which will result in something in between those two. Notice that in the "Waveform" value the left digit selects an actual waveform and the right digit is used for triggering the ADSR envelope, ring modulation and synchronization. The latter two make the channel this preset is used on perform ring modulation or synchronization with the left adjacent channel (3 with 2, 2 with 1 and 1 with 3).

Resetting Oscillators.

*Disabling the oscillator through a "shut up SID!" waveform value of $09 or $08 is performed by a lot of skilled C64 musicians to intentionally reset the oscillator. SID isn't flawless and in some cases (with the longest decay/release rates or really fast notes) performing a reset of the oscillator helps keeping stuff sound steady and neat.

This trick can be performed within Music Assembler by setting the Waveform to $09 or $08 (both will be converted to $08 by the player routine) and using an arpeggio to select the actual waveform(s) you want to use. This causes Music Assembler to switch off the oscillator instead of closing the gate and entering the release phase of the envelope at the end of a note's lifetime.

Some people refer to this technique as "hard resets".

Pulse rate and its effects.

The pulse waveform differs from the other waveforms through SID's ability to modulate the pulse waveform shape. The shape of the pulse waveform can be set by entering a two digit hexadecimal number in the preset. This value is divided in two digits, the left one being the least significant and the right one the most significant. A 100% pure pulse waveform is selected by entering the value $08.

Pulse effects can be switched on/off by using the F3 and F5 keys. The pulse effects triggered are named "Pulse Slide" and "Pulse Vibrate" within Music Assembler.

"Pulse slide" will make the pulse rate cycle by adding the value of "pulse byte" to it every frame (50 times a second). This is an oldskool effect which was used by some C64 music pioneers in the early days of C64 music. Using it creates a rough edge to the pulse waveform sound.

"Pulse Vibrate" will modulate the pulse rate by adding/subtracting the value of "pulse level" to it for the amount of times specified in "pulse speed" and then do the same the other way round. This effect has been widely used by SID music icons like Rob Hubbard and Martin Galway and creates a smoother sounding pulse effect then you will accomplish with the "Pulse Slide" effect.

Vibrato.

The vibrato effect has three parameters. Vibrato delay specifies the amount of frames to wait from triggering a note to kicking in the vibrato effect. Vibrato speed specifies the amount of frames it takes before the vibrato changes its direction (up or down from the note's actual frequency). Vibrato level specifies how much the vibrato effect will add to/subtract from this frequency.

On a SID chip, the values for the lower octave note frequencies are closer to one another than the higher octave note frequencies are. Because of this the vibrato effect will have more impact on low notes then it has on the higher octave ones. Keep this in mind when creating presets – there will be need for separate bass and higher pitched presets.

Arpeggios.

An arpeggio is a tiny sequence of which each step is played only 1/50th of a second. Arpeggios can be used to emulate chords by quickly repeating 3 or 4 notes, or you can emulate drum and percussion sounds with them. Each arpeggio step consists of a waveform, note offset and filter frequency.

Each step of an arpeggio consists of a waveform, note offset and low pass filter frequency (read more on filtering in the Sequence editor chapter).

The arpeggio note offsets can be either an absolute value (Music Assembler will ignore the note triggered from a sequence and replace it with the values specified in the arpeggio) or the default "offset" which will make Music Assembler add up the sequenced and arpeggiated notes and allows you to play melodies with arpeggios. Rule of thumb is that for percussive sounds one uses absolute arpeggios and for emulating chords one uses the default arpeggios.

Absolute notes are denoted with a <.You can toggle between the normal and absolute notes by pressing the up arrow key while editing an arpeggio. Of course feel free to experiment with mixing absolute and offset notes and keep in mind that in hexadecimal numbers $0C is one octave (you will probably use this value a lot).

Ending the arpeggio waveform column with a hex value of $FF will make the arpeggio loop, ending with $FE will make the arpeggio stop at the end.

Please take notice that an arpeggio is not always attached to just one preset. Multiple presets can share the same arpeggio. If an arpeggio is only used by the current preset Music Assembler will show “new arpeggio”. An arpeggio which is shared between multiple presets will make Music Assembler show “used arpeggio”.

Using an arpeggio cancels any vibrato effect present in the preset utilizing the arpeggio.
Arpeggios are overruled by portamento (slide) effects originating from sequences. Be creative! ☺

Two example arpeggios:

chord emulation	**percussion triangle wave**
41 00 00	81 5F< 00
41 04 00	11 00 00
41 07 00	FE
FF	

The first example cycles through three notes (C, E, G if a C was triggered from a sequence). The second arpeggio adds a high pitched white noise click to every note played and continues the note triggered using a triangle waveform.

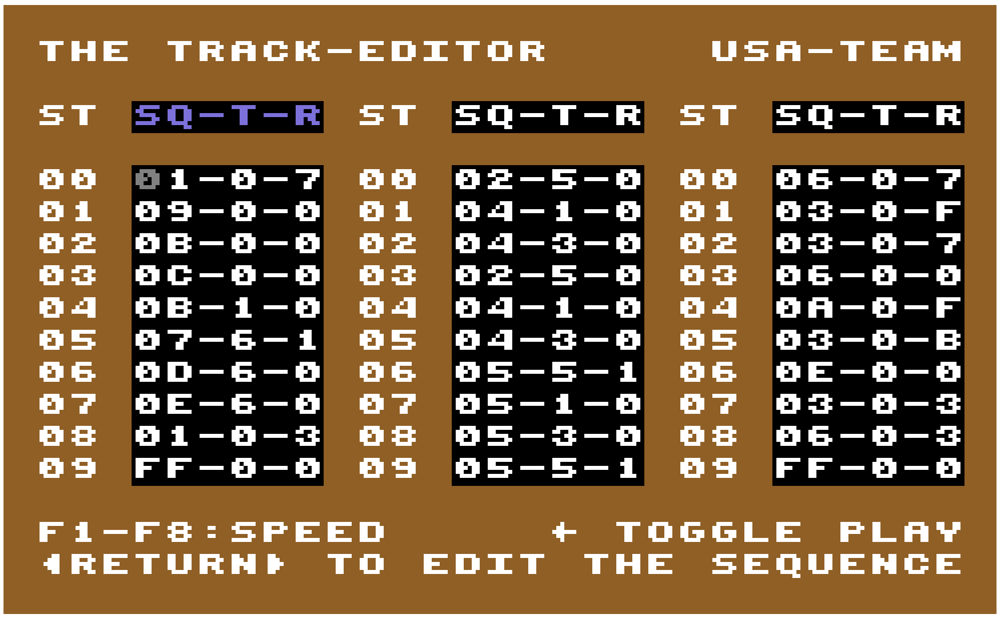
Keys used in the preset editor.

From the main preset editing screen you can use the following keys:

CLR	Clear preset or arp you’re editing
Cursor keys	move around the editable areas.
Left Arrow	Enter built-in keyboard (for testing presets)
RunStop	Play/Stop music.
Shift+D	Disk menu (file operations)
Spacebar	Open track editor

Pressing a key that has no function will warp you to the HELP section.

Track Editor.



```
THE TRACK-EDITOR        USA-TEAM

ST SQ-T-R  ST SQ-T-R  ST SQ-T-R

00 01-0-7  00 02-5-0  00 06-0-7
01 09-0-0  01 04-1-0  01 03-0-F
02 0B-0-0  02 04-3-0  02 03-0-7
03 0C-0-0  03 02-5-0  03 06-0-0
04 0B-1-0  04 04-1-0  04 0A-0-F
05 07-6-1  05 04-3-0  05 03-0-B
06 0D-6-0  06 05-5-1  06 0E-0-0
07 0E-6-0  07 05-1-0  07 03-0-3
08 01-0-3  08 05-3-0  08 06-0-3
09 FF-0-0  09 05-5-1  09 FF-0-0

F1-F8:SPEED      ← TOGGLE PLAY
◀RETURN▶ TO EDIT THE SEQUENCE
```

Editing tracks.

Editing tracks is done by entering sequence numbers in the first columns of the three track channels. Sequence numbers must start from $00 up to a maximum of $FD. Values $FE and $FF are reserved for stopping and looping the track.

Each channel consists of three columns: Sequence number, Transpose offset and Repeat amount.

Just like in the preset editor, in the track editor you can move around the track using cursor keys. Switching to another track is done with the + and - keys.

Sequence number limitations.

Do pay attention to the fact there can't be any gap in between the sequence numbers. For instance you can't edit a sequence 03 if there isn't a sequence 02 already. When you enter an illegal sequence number (one that can't exist yet) you will not be able to edit that sequence number.

Transpose.

The transpose offset allows you to transpose any sequence to a higher (up to 15 notes) pitch somewhere down the track. This comes in handy when repeating stuff like basslines while following the chords/scales, you won't have to sequence the same bassline for every different step of the track.

Repeat.

Last column in the three tracks is the Repeat amount. If this value is left at its default 0 the sequence you entered in the track will be played only once. Increasing the repeat to 1 means it will repeat this sequence 1 time thus play it twice in total.

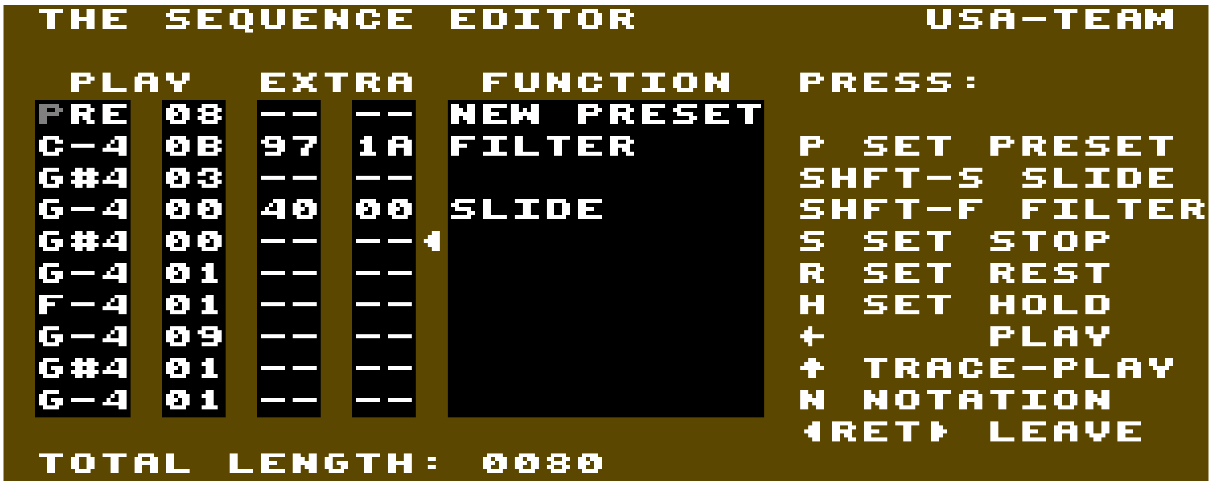
Swapping tracks.

By pressing Shift+S you can swap the track editor back and forth between the actual song you're writing and a secondary set of tracks in which you can test your combined sequences for a certain part of the song. This way you won't have to go through the entire song to check whether or not you're happy with the solo near the end of it. This feature will prove quite handy when writing lengthy pieces of music.

Keys used in the track editor.

Left arrow	Play / Stop
RunStop	Pause/Continue
Up arrow	Play and trace tracks
Return	Edit sequence
CLR	Clear track
Home	Back to step 00
Inst	Insert step
Del	Delete step
Shift+D	Disk menu (file operations)
Shift+S	Swap tracks
F1-F8	Set song speed

Sequence Editor.



```
 THE SEQUENCE EDITOR              USA-TEAM

  PLAY     EXTRA    FUNCTION     PRESS:
 PRE  08  --  --  NEW PRESET
 C-4  0B  97  1A  FILTER         P   SET PRESET
 G#4  03  --  --                 SHFT-S  SLIDE
 G-4  00  40  00  SLIDE          SHFT-F  FILTER
 G#4  00  --  --◄                S   SET STOP
 G-4  01  --  --                 R   SET REST
 F-4  01  --  --                 H   SET HOLD
 G-4  09  --  --                 ←       PLAY
 G#4  01  --  --                 ↑   TRACE-PLAY
 G-4  01  --  --                 N   NOTATION
                                 ◄RET►   LEAVE
 TOTAL LENGTH:  0080
```

Selecting presets.

Before doing anything else Music Assembler must know which preset to use to play. Selecting a preset is done by simply hitting the P key. A “PRE” command will show up on the line you’re editing and you select the preset you want in the second column of the sequence editor.

Entering notes.

Entering notes as can be seen in the screenshot at the top of this page is quite simple. By default Music Assembler accepts standard (C - C# - D - D# - E - F - F# - G - G# - A - A# - B) notation but you can switch to German (C - C# - D - D# - E - F - F# - G - G# - A - A# - H) notation by pressing the N key.

In the second column you enter the duration of the entered note. This may seem a little confusing (and it is) but the duration times start counting from 00 instead of 01. What this means is when you for example enter 03 the note will actually have a length of 4 (a quarter note) and 07 will be 8 or a half note.

Don’t ask me why we ever took this approach and not have the Music Assembler user get exactly what he enters, it must have seemed logical at the time to a bunch of kids who got high on machine code… These are the values entered for typical note lengths:

1F	I’ll see you tomorrow
0F	Whole note
07	Half note
03	Quarter note
01	8th note
00	16th note

Canceling out ADSR triggering.

You can enter notes while holding shift to have Music Assembler play the note without causing a new trigger of the ADSR envelope. These "no trigger" notes are denoted with a < as can be seen in the second G#4 in the sequence editor screenshot at the top of the previous page. Typical usage of this feature is at the end of a portamento (slide) to make it seamlessly slide into the new note.

Rests and Holds.

Rests and holds can be very useful. A Hold simply holds the current note for a maximum of two whole notes. You can place several Hold commands after each other so there is no limit as to how long any note can last.

A Rest basically does the same thing a Hold does with one difference: A Rest kicks in the 'Release' phase of the ADSR envelope while a Hold does not.

Portamento (slide) effects.

Yes indeed. The ultimate tool of royal SID groovyness. This is what people know the SID for even if they never owned a C64… Slides allow you to create guitar like solo sequences that will catch anyone's attention! Grandmasters of this effect are (again) Rob Hubbard and Martin Galway although Yip and JT done some really ace stuff using it too.

Have the Music Assembler slide from one note to another by pressing Shift+S on the position of the note you want to slide from. The two extra columns are now activated for this sequence step. These two columns hold the LSB and MSB (read: fine and coarse) values for the slide effect

If you want to slide downwards instead of up, set the last column to FF or FE (hexadecimal for -1 and -2) and tweak the LSB/fine value until you're happy.

We incorporated a little trick that allows "rattling" slides instead of regular ones. Using even values in the LSB/fine column will result in regular slides, using odd values will engage the "rattling" slide effect. Music Assembler achieves this effect by making the effect audible only once in every two frames.

Triggering a slide effect will cancel out any arpeggio or vibrato effect being used by the current preset for the duration of the slide.

Low pass filter effects.

Another thrilling feature of the SID chip is its ability to filter waveforms. After experimenting a lot with filters and checking what other C64 musicians were doing we decided to stick to low pass filtering only (pretty much like everyone else) in Music Assembler. The band pass and high pass filtering types just don't produce enough musically usable effects.

Filtering is enabled by pressing Shift+F on the notelength column at the note you want the filter effect to start from. The two extra columns in the sequence editor are activated and off you go.

Filter effects use three parameters so the first extra column is split into two values. The first digit holds the filter starting cut-off frequency. 0 being the lowest – and F being the highest possible frequency.

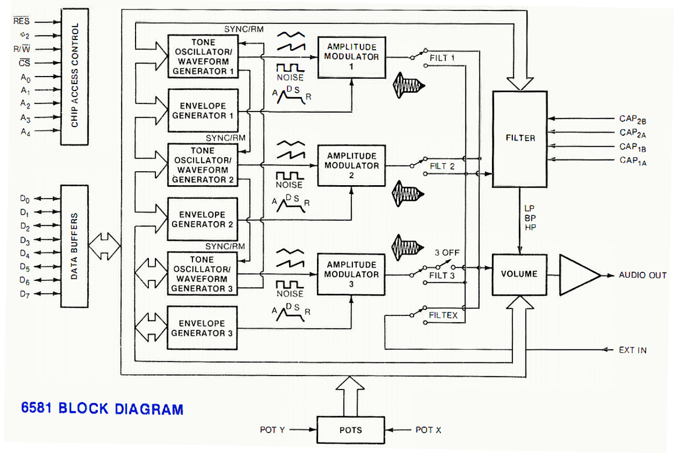
The second digit in this column will decide if and how fast the filter cut-off frequency will be raised or lowered. We managed to make this as intricate as possible so I'll simply write down a table to show which values you have to use:

Value	**Cut-off freq direction**	**Value**	**Cut-off freq direction**
7	down by 2 per frame	9	up by 2 per frame
6	down by 4 per frame	A	up by 4 per frame
5	down by 6 per frame	B	up by 6 per frame
4	down by 8 per frame	C	up by 8 per frame
3	down by 10 per frame	D	up by 10 per frame
2	down by 12 per frame	E	up by 12 per frame
1	down by 14 per frame	F	up by 14 per frame
0	down by 16 per frame	*8*	*hold steady*

The last extra column holds the amount of frames the filtering effect will be applied. This can be used to keep the cut-off frequency from falling below $00 when moving down or over $FF when moving up. Entering 00 in this last column switches off filtering.

Filtering and tracks.

The architecture of the SID chip does not allow separate filtering for each channel. There are three oscillators sharing just one filter. Before enabling filtering on multiple tracks it is vital to understand that only one of the three tracks will be triggering the filter effects. Because of this, in Music Assembler **filtering is applied to the track it is triggered from and all lower tracks.** Using filter effects in track 3 will also cause filtering to be applied to tracks 2 and 1. Filtering from track 2 will also be applied to track 1. Usually you will do your filtering in track 1 as this is the only track which can use the filter effects by itself.



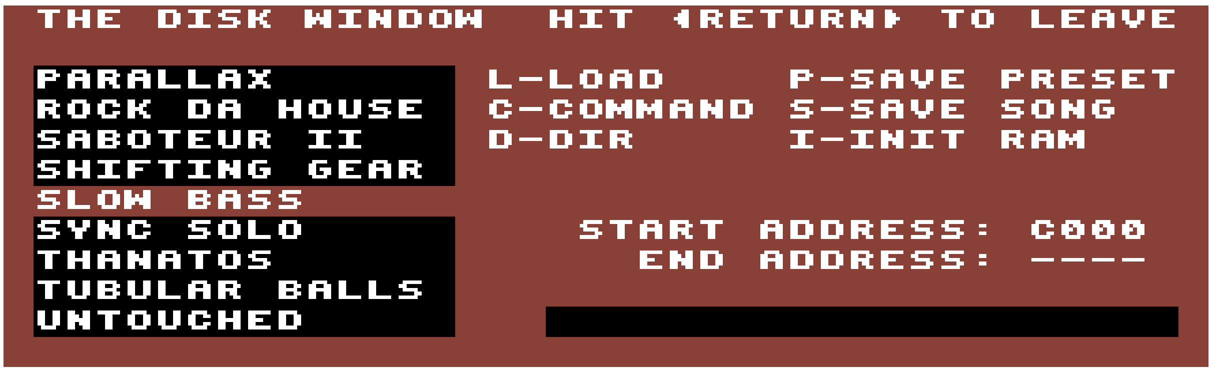
Copying sequences.

Wouldn't it be easy if you could copy a sequence and modify it instead of creating each sequence from scratch? YES WE CAN. Within the sequence editor press Shift+W to write the current sequence to a copying buffer. When you screw up while editing just hit Shift+R to read it back from the buffer. You can also exit the sequence editor, edit another sequence and hit Shift+R there. Now you have two copies of a sequence. How handy.

Keys used in the sequence editor.

Up arrow	Play and trace sequence
Left arrow	Play/stop sequence
RunStop	Play/stop sequence
CLR	Clear sequence
Home	Go back to start of sequence
Inst	Insert step
Del	Delete step
F1-F8	Set speed
Return	Store sequence and exit to track editor

Disk menu.



```
 THE DISK WINDOW  HIT ◀RETURN▶ TO LEAVE

PARALLAX       L-LOAD     P-SAVE PRESET
ROCK DA HOUSE  C-COMMAND  S-SAVE SONG
SABOTEUR II    D-DIR      I-INIT RAM
SHIFTING GEAR
SLOW BASS
SYNC SOLO         START ADDRESS: C000
THANATOS            END ADDRESS: ----
TUBULAR BALLS
UNTOUCHED
```

This part of Music Assembler is pretty self explanatory. Be aware of the fact that Music Assembler only shows its own file types when reading the directory from a disk. Other files are NOT shown in the disk menu.

You can relocate the song to another memory address during save operation. Formatting a disk is done by pressing C and typing **N:diskname,ID**. Scratching (deleting) a song is done by entering the command: **S:s.songname** (wildcards are allowed). BE CAREFUL with disk commands unless you know what you're doing.

P.S. Greetings to all who still know me from olden days. MC is still alive – I just took a little break for about 20 years… Peace out.