1.7. Программирование звуковых эффектов

Fix Me!

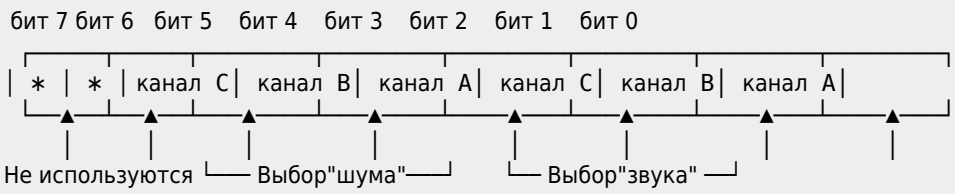
Мы уже говорили о возможностях работы с Программируемым Звуковым Генератором — PSG. Здесь мы расскажем Вам о работе с PSG при программировании в машинных кодах.

Вначале кратко *повторим* основные сведения.

Назначение регистров PSG Вы легко вспомните, если взгляните на таблицу:

Регистр	***Назначение***
0	Младший байт частоты канала A
1	Старший байт частоты канала A
2	Младший байт частоты канала B
3	Старший байт частоты канала B
4	Младший байт частоты канала C
5	Старший байт частоты канала C
6	Частота шума
7	Выбор звучания каналов
8	Громкость канала A
9	Громкость канала B
10	Громкость канала C
11	Младший байт периода волны
12	Старший байт периода волны
13	Указатель формы волны

Звуки бывают двух типов, которые мы назовем «звук» и «шум». За сочетание звучания «звука» и «шума» отвечает 7–й регистр PSG:



Частота звучания «звука» из некоторого канала (A,B или C) определяется по следующей формуле:

$$\frac{1789772.5\ \text{Гц}}{16 \cdot \text{частота}} = \text{значение в регистрах для канала } \texttt{A, B или C}$$

По той же формуле вычисляется частота звучания «шума»(значение берется из регистра с номером 6).

Период волны определяется по следующей формуле:

$$\frac{1789772.5\ \text{Гц}}{256 \cdot \text{период}} = \text{значение в регистрах 11 и 12.}$$

Работа звукового генератора поддерживается процессором через порты ввода–вывода со следующими адресами:

Порт	***Назначение***
A0h	Номер регистра PSG для записи числа

Порт	*Назначение*
A1h	Число для записи в отмеченный регистр
A2h	Последнее число, записанное в PSG

Существует два способа записи чисел в PSG средствами языка MSX BASIC:

- α)

```
SOUND reg, data
```

- β)

```
OUT &HA0,reg:OUT &HA1,data
```

Пример 1. Программа в машинных кодах, записывающая число 254 в звукогенератор (в регистр 0).
1070-01.bas
1070-01.bas

```
10 CLEAR 200,&HF000:DEFUSR=&HF000
20 DATA 3E,00 :' LD  A,0
30 DATA D3,A0 :' OUT (A0h),A
40 DATA 3E,FE :' LD  A,FEh
50 DATA D3,A1 :' OUT (A1h),A
60 DATA C9    :' RET
70 DATA RET
80 READ Z$
90 IF Z$="RET" THEN 110
100 POKE &HF000+T,VAL("&h"+Z$):T=T+1:GOTO 80
110 A=USR(A)
```

Следующая таблица поможет Вам при моделировании оператора PLAY в машинных кодах (данные приведены для канала A):

Октава	Нота	Рег.0	Рег.1
O1 (контр-октава)	C	93	13
	C#	156	12
	D	231	11
	D#	60	11
	E	155	10
	F	2	10
	F#	115	9
	G	235	8
	G#	107	8
	A	242	7
	A#	128	7
	B	20	7

Октава	Нота	Рег.0	Рег.1
O2 (большая)	C	175	6
	C#	78	6
	D	244	5
	D#	158	5
	E	78	5
	F	1	5
	F#	186	4
	G	118	4
	G#	54	4
	A	249	3
	A#	192	3
	B	138	3
O3 (малая)	C	87	3
	C#	39	3
	D	250	2
	D#	207	2
	E	167	2
	F	129	2
	F#	93	2
	G	59	2
	G#	27	2
	A	253	1
	A#	224	1
	B	197	1
O4 (первая)	C	172	1
	C#	148	1
	D	125	1
	D#	104	1
	E	83	1
	F	64	1
	F#	46	1
	G	29	1
	G#	13	1
	A	254	0
	A#	240	0
	B	227	0

Октава	Нота	Рег.0	Рег.1
O5 (вторая)	C	214	0
	C#	202	0
	D	190	0
	D#	180	0
	E	170	0
	F	160	0
	F#	151	0
	G	143	0
	G#	135	0
	A	127	0
	A#	120	0
	B	113	0
O6 (третья)	C	107	0
	C#	101	0
	D	95	0
	D#	90	0
	E	85	0
	F	80	0
	F#	76	0
	G	71	0
	G#	67	0
	A	64	0
	A#	60	0
	B	57	0
O7 (четвёртая)	C	53	0
	C#	50	0
	D	48	0
	D#	45	0
	E	42	0
	F	40	0
	F#	38	0
	G	36	0
	G#	34	0
	A	32	0
	A#	30	0
	B	28	0

Октава	Нота	Рег.0	Рег.1
O8 (пятая)	C	27	0
	C#	25	0
	D	24	0
	D#	22	0
	E	21	0
	F	20	0
	F#	19	0
	G	18	0
	G#	17	0
	A	16	0
	A#	15	0
	B	14	0
(шестая)	—	13	0

Пример 2. Программа, позволяющая прочесть данные из звукогенератора и проверить приведённую таблицу.
1070-02.bas
1070-02.bas

```
10 CLEAR 200,&HF000:DEFUSR=&HF000
20 DATA CD,1F,52       :'CALL 521F ;
30 DATA CD,96,00       :'CALL 0096 ; Чтение числа из PSG
40 DATA 26,00          :'LD   H,0  ;
50 DATA 6F             :'LD   L,A  ;
60 DATA C3,99,2F,"RET" :'JP   2F99 ;
70 READ Z$
80 IF Z$="RET" THEN 100
90 POKE &HF000+T,VAL("&h"+Z$):T=T+1:GOTO 70
100 PLAY "O4 C#"
110 PRINT USR(0);USR(1) 'Читаем содержимое 0-го и 1-го регистров
```

Фоновое музыкальное сопровождение

(Данный раздел написан А.Н.Никитиным)

Существует несколько вариантов реализации фонового сопровождения программы. Вспомним один из рутинных вариантов воплощения фонового музыкального сопровождения. Зная, что длительность нот намного превышает время выполнения процессором арифметических операций, можно установить соответствие между музыкальной длительностью и определённым количеством арифметических операций, выполняемых процессором. Далее остаётся последовательно вставлять музыкальные фрагменты (их длина, разумеется, должна быть ограничена настолько,чтобы программа не «зависала»!) между операторами основной программы. Ясно, что такой способ вызывает массу неудобств: значительно усложняется редактирование программы, программирование требует огромных затрат времени и труда, об универсальности программы не может быть и речи.

Предлагаемый нами алгоритм реализации фонового музыкального сопровождения полностью *исключает* перечисленные неудобства!

Расскажем Вам идею алгоритма. Во время работы процессор совершает маскируемые прерывания частотой 5O Гц и опрашивает область ловушек (hooks),а точнее происходят обращения (CALL ...) по адресам FD9Ah, FD9Fh. Напомним Вам, что эти прерывания разрешаются командой ассемблера EI, а запрещаются командой DI. Начальный адрес Таблицы ловушек — &hFD9A. По этому адресу системной области хранится подпрограмма перехода на программу сетевого обмена у компьютеров серии MSX-2 и число &HC9(машинный код команды RET) у компьютеров серии MSX-1, поэтому при обращении к данной ловушке либо из подпрограммы обработки прерываний (которая расположена по адресу 0038h), либо из Вашей программы означает мгновенный возврат в основную программу (если MSX-2 не ведёт «сетевой диалог»).

Если по адресу FD9Ah «положить» подпрограмму:

```
RST    30h                                    ,
DEFB   N_slot
DEFW   Adres
RET
```

где N_slot — это номер слота, в котором будет расположена Ваша программа, а Adres — это адрес Вашей подпрограммы обработки прерываний, то 50 раз в секунду управление будет передаваться Вашей подпрограмме, в которой можно управлять музыкой, печатью, графикой или опрашивать клавиатуру.

Вообще говоря,число 50 не очень удобно для точного отсчёта музыкальной длительности (так как 50 не делится на 8, 16, 32; в этом смысле идеальным было бы число 64), но при сравнении длительностей между собой (а не с метрономом) искажение звука практически незаметно.

Структура музыкальной подпрограммы довольно проста: она должна обрабатывать данные и управлять музыкальными очередями. Вся сложность состоит в том, чтобы построить такую структуру музыкальных данных, которая удовлетворяла бы следующим требованиям: объем данных должен быть минимальным, возможности управления музыкальным генератором должны быть большими, должны присутствовать элементы программирования (циклы,переходы и т.д.), должно выполняться большинство музыкальных выражений (Legato, Staccato и т.д.).

В предложенном ниже описании структуры музыкальных данных, мы постарались удовлетворить лишь части названных требований.Структура данных очень напоминает стандарт MIDI–интерфейса(Musical Instrument Digital Interface).

Порядок размещения данных в очереди следующий:

```
<Команда>,{Данное},{Данное},<Команда>,{Данное},{Данное},...
```

Следует отметить, что {Данное} может быть опущено.

Приведём описание команд и данных.

Команда	**Описание**
0	Установка частоты (высоты) звучания: 1 полубайт — команда 00 ; 2 полубайт — номер ноты 1÷12.

1	Микширование (регистр 7 PSG,по умолчанию — &hb10111000):
	1 полубайт — команда 01 , 2 полубайт — формальный , следующий байт — байт состояния (структура аналогична структуре данных, находящихся в регистре 7):
	`&b10 *** ***`
	`▲▲▲ ▲▲▲`
	`||| |||`
	`cba cba – номера каналов`
	`Шум Сигнал`

2	Установка темпа воспроизведения (по умолчанию — 1): 1 полубайт — команда 02 ; 2 полубайт — данное (1÷2).
3	Установка амплитуды звучания (по умолчанию — 8): 1 полубайт — команда 03, 2 полубайт — данное (0÷15) (амплитуда = данное + 1).
4	Установка частоты шума (по умолчанию — 20): 1 полубайт — команда 04, 2 полубайт — данное (1÷15, частота = данное·2).
5	Установка формы волны (пакета): 1 полубайт — команда 05 , 2 полубайт — данное (1÷15).
6	Установка значения для 11 регистра PSG(по умолчанию — 100) 1 полубайт — команда 06, 2 полубайт — данное (0÷15, значение = данное · 10).
7	Установка значения для 12 регистра PSG(по умолчанию — 10): 1 полубайт — команда 07, 2 полубайт — данное (0÷15, значение=данное·2).

Команда	**Описание**
8	Установка текущей длительности звучания: 1 полубайт — команда 08, 2 полубайт — данное (1÷15), где • 1 — целая с точкой, • 2 — целая, • 3 — половинная с точкой, • 4 — половинная, • 5 — четвертная с точкой, • 6 — четвертная, • 7 — восьмая с точкой, • 8 — восьмая, • 9 — шестнадцатая с точкой, • 10 — шестнадцатая, • 11 — тридцатьторая с точкой, • 12 — тридцатьторая.
9	Пауза: 1 полубайт — команда 09, 2 полубайт — формальный.
10	Переход (CALL или JUMP) по адресу данных 1 полубайт — команда 10 , 2 полубайт — формальный, следующие два байта — относительный адрес данных от расположения всех музыкальных данных.
11	Возврат из подпрограммы (адреса хранятся «у себя») Возвращаться можно сколь угодно раз (ибо это не стек)! 1 полубайт — команда 11 , 2 полубайт — формальный.
12	Стоп, выключить канал: 1 полубайт — команда 12, 2 полубайт — формальный.
13	Установка текущей октавы: 1 полубайт — команда 13, 2 полубайт — номер октавы (1,2,3,...,8).
14	Приём игры (Staccato, Legato): 1 полубайт — команда 14, 2 полубайт — 1, если Staccato и 0, если Legato. Заметим, что числа от 2 до 15 могут выражать и другие приёмы игры!
15	Резервная: 1 полубайт — команда 15, 2 полубайт — ... (можно использовать как передачу других команд, т.е. 15–я команда становится *префиксной*).

Таким образом, Ваша подпрограмма должна распознавать номер команды, исполнять её, исходя из параметров, следующих за командой и помещать указатель данных на следующую команду. Если Вы желаете увеличить музыкальные возможности программы,а команд (в полубайт можно поместить число от 0 до 15) не хватает, то необходимо выделить префиксную команду (например, 15–ю, так как она резервная)и теперь порядок размещения данных в очереди станет следующим:

```
<Команда>,<Команда>,{данное},{данное},...
<Команда>,<Команда>,{данное},{данное},...
```

Кроме аппаратной реализации фонового музыкального сопровождения,существует и программная. Вся прелесть аппаратной реализации в том, что уменьшение длительности нот происходит в режиме реального времени.Программно можно запараллелить многие процессы (в том числе и музыку):

```
Process:  CALL   ...                    ,
          CALL   Music
          CALL   ...
          JR     Process
```

но дело в том, что тактовая частота процессора гораздо выше, чем частота маскируемых прерываний и без задержки (которая очень часто просто не нужна) казалось бы не обойтись. *Но выход есть*!

Маскируемые прерывания оставляют след в области системных переменных, и этим можно воспользоваться. Ячейка с адресом 0FC9Eh (Jiffy) хранит этот след. Содержимое Jiffy (2 байта) при каждом новом прерывании увеличивается на единицу, а значит, момент нового «тика» (это выражение произошло из принципа работы подпрограммы часов

компьютера)всегда можно регистрировать.

В Приложении приведена программа обработки музыкальных очередей, написанная на ассемблере Z80 для компьютера MSX-2.

---

https://sysadminmosaic.ru/msx/basic_dialogue_programming_language/107

2023-02-18 18:12