uMAPI (MicroMachine Application Program Interface)
Version 1 (BASE Edition)


МикроМашина (далее в тексте мМ) состоит из 4-х под-систем:

| | |
|---|---|
| Видео под-система Video SubSystem | (VIDEO) |
| Звуковая под-система Sound SubSystem | (SOUND) |
| Под-система ввода Key SubSystem | (KEY) |
| Микро-яркий светодиод LED SubSystem | (LED) |

1) VIDEO

Video SubSystem - самая навороченная под-система мМ

Аппаратно состоит из:

| | |
|---|---|
| ПЗУ спрайтов Sprite ROM | (SROM) |
| ОЗУ ВидеоБуфера Video RAM | (VRAM) |
| ЖК-дисплея | (LCD) |

Алгоритм работы Video SubSystem:

1) VRAM заливается константой
2) Спрайты из SROM копируются в VRAM
3) Содержимое VRAM отображается на LCD

Далее процесс повторяется...

Некоторые характеристики:

| | |
|---|---|
| Доступное разрешение LCD | 128x128* |
| Доступная глубина цвета | 8bpp (256 color)** |
| Объём VRAM | 16384 байт (16kB) |
| Диапазон X-координаты спрайтов | -128..127 |
| Диапазон Y-координаты спрайтов | -128..127 |
| Отсечение спрайтов по X | X<0, X>127 |
| Отсечение спрайтов по Y | Y<0, Y>127 |

*- на самом деле физическое(!) разрешение LCD 132x132, просто для упрощения и ускорения алгоритмов видеовывода логическое(!) разрешение LCD в uMAPI версии 1 взято 128x128
Это облегчает реализацию отсечения спрайтов, координаты которых могут быть меньше 0

**- аппаратно LCD мМ кроме режима 256 цветов, поддерживает ещё режим 4096 цветов
В uMAPI версии 1 этот режим не используется

uMAPI-функции, описывающие алгоритм работы Video:
- ClearVRAM - заливка VRAM константой
- OutSprite - спрайт из SROM копируется в VRAM
- OutLCD - содержимое VRAM передаётся на LCD

Всего 256 цветов, которые определены в Palette.bmp*
Из них выбираются любые два цвета:
- Invisible Color - невидимый цвет (a.k.a. цветовой ключ)
- Background Color - фоновый цвет (a.k.a. константа для заполнения VRAM)

А также есть ещё два служебных цвета, которые не обрабатываются uMAPI:
- EOS - End Of Sprite Color - цвет конца спрайта
- Ignore Color - игнорируемый цвет

Данные цвета нужны для программ-конвертеров, которые преобразуют BMP-файл спрайтов в удобоваримые данные для uMAPI

*- аппаратно можно определить любые(!) 256 цветов из 4096 возможных
Просто в данной версии uMAPI используется уже готовая палитра (Palette) цветов, которая на взгляд автора оптимальна
Такой подход облегчает инициализацию контроллера LCD и избавляет программиста от лишних хлопот

В рамках uMAPI, спрайт - наибольший объект, с которым работает Video SubSystem
Спрайты имеют свой индекс (0..65535), X-координату (-128..127), Y-координату (-128..127)
Располагаются спрайты в SROM (младшие 64kB флеш-памяти MCU)*
Для каждого спрайта ведётся контроль отсечения по X- и Y- координате
Участки спрайта, закрашенные Invisible Color не передаются в VRAM и LCD (в этих местах сохраняется Background Color)
В SROM спрайты располагаются линейно, т.е. слева-направо, сверху-вниз, один за другим без свободных промежутков

Схема организации данных в SROM:
- Sprite Data - данные спрайтов (их видеообразы)
- Sprite Offset (0..65535) - относительные смещения от начала SROM (для считывания спрайтов)**
- Sprite Width (0..255) - ширины спрайтов
- Sprite Height (0..255) - высоты спрайтов

Необходимо указать адреса данных SROM (структура с "плавающими" адресами)

*- аппаратно MCU мМ может обращаться только к нижним 65536-ти байтам SROM
Объясняется это тем, что команда "lpm" способна загрузить байт с флеш-памяти MCU только по 16-битному адресу,
несмотря на то, что MCU имеет 128kB флеш-памяти

**- в uMAPI все данные, которые шире 8-ми бит, организованы по принципу "младший байт - по младшему адресу"

Из карты памяти MCU следует, что External SRAM (внешнее статическое ОЗУ) располагается по адресам 0x1100..0x90FF (32kB)
Internal SRAM используется для регистров, портов ввода/вывода MCU, а также для хранения небольших данных и
располагается по адресам 0x0000..0x10FF
Учитывая то, что в конце External SRAM обычно находится стек, VideoBuffer логично расположить
в диапазоне адресов 0x1100..0x50FF (16 kB)

При написании программы для мМ следует очень внимательно следить за расположением переменных, данных, и.т.п.
Пересечение адресов приводит к неправильной работе программы

uMAPI-функция, которая готовит LCD к работе:
PrepareLCD

Коротко о том, что эта функция делает:
1) Сбрасывает контроллер LCD
2) Инициализирует контроллер LCD
3) Переводит LCD в режим 256 цветов
4) Заполняет палитру
5) Включает LCD
6) Очищает LCD нулевым цветом

После вызова этой функции, X-координата растёт слева-направо, Y-координата растёт сверху-вниз

Такой подход, конечно, скрывает детальную инициализацию и управление контроллером LCD,
но в данной версии uMAPI это оправдано

За кадром остались такие вещи как:
1) Управление яркостью/контрастностью LCD
2) Установка режима 4096 цветов
3) Прокрутка изображения
4) Инверсия изображения
5) Вертикальная/горизонтальная зеркализация изображения
6) Установка частоты обновления LCD
И многое другое...

При написании игр под мМ требуется определять наложение спрайтов друг на друга
Поэтому необходимо считывать данные ВидеоБуфера, в частности - читать области VRAM

В uMAPI версии 1 есть функция, определяющая цвет пикселя в определённых координатах:

| InPixel | |
|---|---|
| X-координата | (0..127) |
| Y-координата | (0..127) |
| Возращает цвет пикселя | (0..255) |

При выходе за диапазон X- и/или Y- координаты, функция возвращает Background Color
Тоесть в uMAPI принято, что за экраном - цвет фона

В целом для определения столкновения спрайта с другим, достаточно определить цвет его пикселей, находящихся по краям
Если хоть один крайний пиксель поменял цвет, значит до спарйта дотронулся другой спрайт

На всякий случай, в uMAPI оставлена функция вывода одного пикселя в VideoBuffer:

| OutPixel | |
|---|---|
| X-координата | (0..127) |
| Y-координата | (0..127) |
| Цвет пикселя | (0..255) |

Если цвет пикселя равен Invisible Color (т.е. цветовому ключу), то он не выводится
При выходе за диапазон X- и/или Y- координаты, пиксель тоже не выводится

2) SOUND

Sound SubSystem - тоже довольно навороченная под-система мМ

Весь звук в мМ генерируется с помощью FM-синтезатора (тип OPN2)*
Полифония - 6 каналов (инструментов, голосов)
Каждый канал состоит из 4-х операторов
Звук Stereo - каждый канал может панорамироваться в Левый, Правый или в оба спикера

*- аппаратно звуковая под-система мМ может, кроме FM-синтеза, воспроизводить 8-битные монофонические оцифровки
Но в силу того, что ресурсы Памяти и Скорости МикроМашины сведены к минимуму, в uMAPI данной версии это не реализовано

В связи с тем, что управлять современным FM-синтезатором довольно сложно с программной точки зрения, составлена
таблица из 256 сэмплов (патчей) для FM-синтезатора

Патч может быть загружен в один из шести каналов (0..5), возможна загрузка одного и того же патча в несколько каналов

uMAPI-функция, описывающая загрузку патча в канал FM-синтезатора:

LoadPatch
0..255 - номер патча
0..5 - номер канала, в который патч загружается

Таблица патчей, также как и таблица палитры цветов встроена в uMAPI с программной точки зрения
Физически же, эта таблица находится во флеш-памяти MCU мМ

Перед работой с Sound SubSystem, нужно сбросить FM-синтезатор
uMAPI-функция, которая это делает:

ResetFM

Сброс синтезатора приводит к обнулению его регистров

Но одного сброса и загрузки патчей в микросхему FM-синтезатора мало
Необходимо также указать параметры и панорамирование канала

uMAPI-функция, описывающая параметры канала:

ParameterChannel
0..5 - номер канала
0..7 - блок канала (октава)
0..2047- частота канала

Частота канала задаёт точную настройку тональности загруженного патча
Октава канала задаёт грубую настройку тональности загруженного патча

uMAPI-функция, позволяющая панорамировать канал в спикеры:

PanoramChannel
0..5 - номер канала
0..3 - режим панорамирования

Значения режимов панорамирования:

SpeakerNO =0x00 - канал не будет звучать ни в одном спикере*
SpeakerLEFT =0x40 - канал будет звучать в Левом спикере
SpeakerRIGHT =0x80 - канал будет звучать в Правом спикере
SpeakerBOTH =0xC0 - канал будет звучать в обоих спикерах

*- на самом деле будет звучать в обоих спикерах, только очень тихо :)
Это же касается и остальных режимов панорамирования

Включение/выключение каналов FM-синтезатора:

SoundFM
0..5 - номер канала
Режим канала (включить/выключить)

Для режимов канала определены следующие константы:

ChannelON =0xF0
ChannelOFF =0x00

Если нужно включать/выключать сразу несколько каналов, то функция SoundFM используется несколько раз

На всякий случай оставлена низкоуровневая uMAPI-функция, работающая напрямую с регистрами FM-синтезатора:

OutFM
0..1 - банк
0..255 - номер регистра
0..255 - значение регистра

Данная функция расширяет возможности использования FM-синтезатора МикроМашины

3) KEY

Назначение Key SubSystem - считать состояние всех кнопок, которые расположены на Клавиатуре мМ

uMAPI-переменная, возвращающая состояние всех(!) кнопок сразу:

Key
Биты расписываются следующим образом:

Бит 0 - UP
1 - DOWN
2 - LEFT
3 - RIGHT
4 - FIGHT
5 - JUMP
6 - SHIFT
7 - PAUSE

Причём когда бит=0, то кнопка нажата и наоборот - бит=1, когда кнопка отжата

В uMAPI определены константы-маски, позволяющие определить нажатие одной кнопки:

```
KeyUP     =0x01
KeyDOWN   =0x02
KeyLEFT   =0x04
KeyRIGHT  =0x08
KeyFIGHT  =0x10
KeyJUMP   =0x20
KeySHIFT  =0x40
KeyPAUSE  =0x80
```

Тоесть, например, надо проверить кнопку JUMP:

if Key and KeyJUMP=0 then <что-нибудь делаем если кнопка JUMP НАжата>;

if Key and KeyJUMP=KeyJUMP then <что-нибудь делаем если кнопка JUMP ОТжата>;

4) LED

Микро СветоДиод установлен в мМ для отладочных целей
При очень малых габаритах (диаметр 3мм) он светит очень ярко!

uMAPI-переменная, позволяющая управлять MicroLED:

```
LED
```

Если LED=1, то MicroLED загорится и будет гореть до тех пор, пока переменной LED
не присвоят значение 0 - MicroLED погаснет

5) Random generator

В uMAPI версии 1 есть встроенная функция, возвращающая случайное число (0..65535):

```
Random
```

А также есть начальное значение Seed (0..65535) для начальной инициализации
генератора псевдослучайных чисел