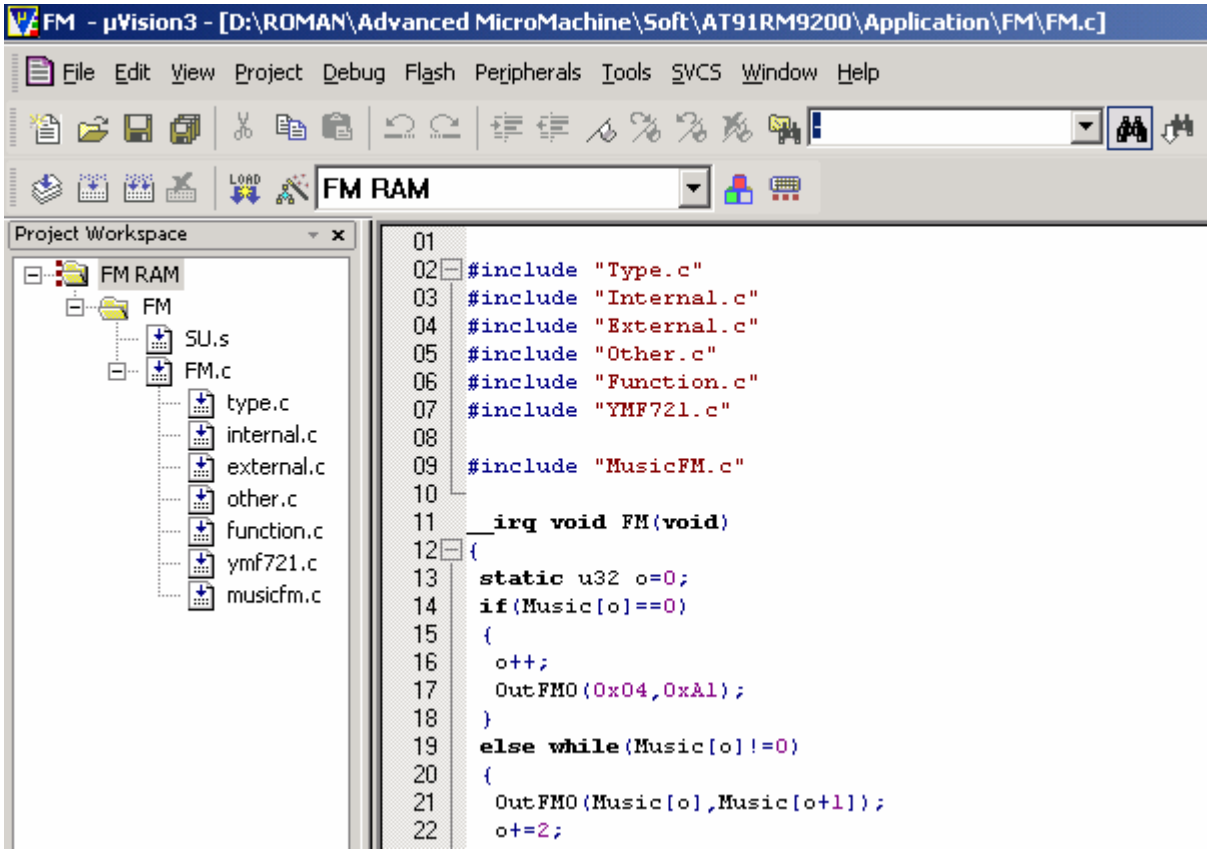
Advanced MicroMachine

Version 1



Руководство программиста

вводный курс

E-Mail: dre1983@mail.ru

1. Введение

Данное руководство вкратце ознакомит читателя с возможностями Advanced MicroMachine (далее AuMv1) с программной точки зрения. А также будет рассмотрено введение в AuMAPI (Advanced MicroMachine Application Program Interface) с целью ознакомления с библиотеками, их структурами и прототипами функций.

Следует отметить, что данное руководство не претендует на полноценно законченную статью по программированию AuMv1, так как AuMAPI полностью ещё не написан и не отлажен.

Также не решён вопрос передачи кода в широкие массы (сделать код в виде открытых .C, .H – файлов или закрытые .LIB – файлы с описанием прототипов функций в .H – файлах).

Данного руководства будет вполне достаточно, чтобы понять базовые механизмы программирования AuMv1, которые включают в себя: программирование регистров, управление памятью, управление периферией, настройкой прерываний. В принципе, вся эта рутинная работа спрятана в модулях, управление осуществляется с помощью параметров AuMAPI-функций, тоесть, если можно так сказать – программист абстрагирован от низкоуровневого программирования железа AuMv1. Но это нисколько не мешает опуститься на нижний уровень, если что-то не реализовано или что-то не устраивает в стандартной AuMAPI-библиотеке (модулях).

2. Аппаратура МикроМашины с точки зрения программиста

Рассмотрим каждую “железку” AuMv1 с точки зрения программиста (программный доступ к аппаратным ресурсам).

CPU:

PLLA – первый синтезатор частоты. Не работоспособен! Возможно из-за Errata микроконтроллера. В AuMv1 не используется.

PLLB – второй синтезатор частоты. Содержит умножитель (MUL) и делитель (DIV). Частота на выходе PLLB находится следующим образом:

PLLB=18.432*(MUL+1)/DIV, кварц на 18.432 MHz (возможны другие варианты)

PCK – тактовая частота ядра процессора (Processor Clock). Управляется прескалером (PRES, делителем степени 2). Находится следующим образом:

PCK=PLLB/PRES

MCK – тактовая частота шины (Master Clock). Управляется делителем (MDIV). Данная частота является отправной для получения тактовых частот на различные узлы как внутренней, так и внешней периферии. Находится следующим образом:

MCK=PCK/MDIV

Модуль StartUp.c позволяет задать параметры MUL, DIV, PRES, MDIV и проинициализировать все вышеуказанные тактовые частоты. По умолчанию в AuMv1 задано:

PLLB = 179.98848 MHz (<180 MHz)
PCK = PLLB/1 (<180 MHz)
MCK = PCK/3 = 59.99616 MHz (<80 MHz)

В скобках даны ограничения от ATMEL, которые, как видно, выполняются.

После установки основных частот, разрешается тактирование следующих узлов периферии (и источников прерывания соответственно):

SYSIRQ – вся внутренняя периферия (пока прерывание System Timer 32768 Hz)
PIOA – порт A (прерывание свободно)
PIOB – порт B (прерывание свободно)
PIOC – порт C (прерывание свободно)
SPI – последовательный интерфейс (прерывание свободно)
TC0 – таймер-счётчик 0 (прерывание опроса джойстика)
TC1 – таймер-счётчик 1 (прерывание свободно)
TC2 – таймер-счётчик 2 (прерывание свободно)
TC3 – таймер-счётчик 3 (прерывание свободно)
TC4 – таймер-счётчик 4 (прерывание свободно)
TC5 – таймер-счётчик 5 (прерывание свободно)
IRQ0 – внешнее прерывание (прерывание таймеров YMF721)
IRQ6 – внешнее прерывание (прерывание DREQ чипа VS1003)

Как говорилось ранее в хардварной спецификации на AuMv1, для увеличения производительности при обмене данными, введён виртуальный режим (включен MMU), адреса сегментов равны физическим, контроль доступа отключен.

Следует также отметить, что ядро работает в режиме SuperVisor, что позволяет делать “вкусные вещи” (например, на лету включать/отключать кеши). В этом режиме разрешены IRQ и FIQ. В режимах IRQ и FIQ прерывания запрещены.

Таблица векторов прерывания: все Exception вызовут зацикливание в их Handler. Нормально описаны только: Reset, IRQ, FIQ.

External Memory:

Объём памяти 2 МБ, чтение/запись по 8/16/32 бит. Шина 16-битная. Это означает, что в режиме ARM обращение к памяти идёт за два цикла. В Thumb режиме – за один цикл. Разумеется, если Кеши выключены. Если задействованы Instruction и Data Cache, то обращение к внешней памяти происходит непостоянно (из-за этого производительность системы не падает резко, когда временные циклы устанавливаются длиннее или шина данных укорачивается).

В последних 16 KB внешней памяти находится Таблица трансляции адресов первого уровня (сегменты 1 MB), поэтому в настройках среды программирования необходимо позаботиться о том, чтобы случайно “не наехать” на этот регион. В противном случае Таблица будет разрушена, программа повиснет.

Доступные адреса внешней памяти: 0x10000000..0x101FBFFF. Деление на код и данные – на усмотрение программиста.

Внимание!!! Ничего не пишите случайно в следующий диапазон адресов: 0x101FC000…0x101FFFFF

SRAM_Address 0x10000000 //External SRAM Base Address

Sound Synthesizer:

Frequency Modulation OPL2/3 (Adlib)

```
FM_Address0     *(volatile u8*) 0x20000000 //Address 0 Write, Status Read
FM_Address1     *(volatile u8*) 0x20000002 //Address 1 Write
FM_Data         *(volatile u8*) 0x20000001 //Data Write, Data Read
```

Wave Table OPL4 (MoonSound)

```
WT_Address      *(volatile u8*) 0x20000004 //Address Write, Status Read
WT_Data         *(volatile u8*) 0x20000005 //Data Write, Data Read
```

MPU401 (MIDI Interpreter)

```
MPU401_Data         *(volatile u8*) 0x30000000 //MIDI Data Write, Acknowledge
MPU401_Command  *(volatile u8*) 0x30000001 /Command Write, Status Read
```

Данный звуковой чип содержит два типа синтезаторов: Частотный и Табличный.

Первый является OPL 2/3 9-, 18- канальным синтезатором. По сути, точно такой же, как в Adlib звуковых картах. Это значит, что 100% совместимость с ДОСовскими плеерами и трекерами, которых бесчисленное множество!!! ☺ Это упрощает создание софта для написания и воспроизведения Adlib-музыки. Уже не говоря о том, что в наследство от ДОСа достались тысячи музыкальных композиций с трекеров, демок и игр.

Второй OPL4, 24- канальный синтезатор. Семплы встроены в чип (1 MB), их количество более 300. Аля MoonSound (или GUS), но только нет возможности загружать свои семплы (для этого в вышеупомянутых звуковых картах была внешняя микросхема RAM, см. инфу на чип YMF278, YRW801). Вероятно, возможно воспроизведение музыкальных композиций от OPL4- трекеров, которые тоже являются наследием старого доброго ДОСа ☺

И наконец, надстройка над WaveTable синтезатором – MPU401 16- канальный MIDI интерпретатор. Попросту говоря – хардварный декодер MIDI- сообщений (events). Позволяет “играть” MID- файлы (SMF0 или SMF1). MID- файлов также бесчисленное множество!!!

Звуковой чип YMF721 содержит таймеры-счётчики, которые могут вызвать прерывание. Это позволяет сделать проигрывание музыки в фоне (в обработчике прерывания), разгрузив тем самым основной цикл программы (псевдо-распараллеливание процессов). Управление таймерами весьма гибкое, что позволяет

реализовать широкую вариацию частоты вызова прерывания (подкачка данных в регистры синтезатора).

Audio Decoder:

Если микросхема YMF721 является синтезатором звука, то VS1003 служит для воспроизведения оцифровок в форматах: MP3, WMA, WAV и MID (SMF0). Последний формат можно “сыграть” и YMF721.

VS1003 – это ещё один процессор в AuMv1 – мощный DSP, декодирующий вышеназванные форматы. Есть эквалайзеры (Treble, Bass), возможность исполнения небольшой пользовательской программы (ревербератор, фильтр, и т.п.).

Подключен к SPI, в AuMv1 работает на частоте 4 MHz (поток данных, команды). Для синхронизации с CPU имеет вывод DREQ (=0, когда первичный буфер чипа заполнен), состояние которого можно определить через порт вода/вывода (GPIO) или что ещё лучше – прерывание по занятости чипа. Тоесть как только чип свободен, происходит вызов прерывания, обработчик которого прокачивает новые данные (допускается без опроса “пропихать” максимум 32 байта). Чип снова взводит DREQ в 0, происходит выход из прерывания (снова в основной цикл программы, который таким образом разгружается).

Возможно одновременная работа обеих звуковых чипов по прерываниям! Это делает возможным распределить звуковые ресурсы в программах (играх) так, чтобы например фоновая музыка игралась синтезатором, а спец-звуки (выстрелы, помощь,…) декодером. Можно и наоборот – всё зависит от фантазии и умения…

OLED Display:

Высококонтрастный, цветастый, яркий дисплей с углом обзора 180 градусов! Графическое разрешение 128 x 128 (может 132 x 132). Доступны следующие цветовые разрешения:

256 Color – RGB=3:3:2
65536 Color – RGB=5:6:5
262144 Color – RGB=6:6:6

Все режимы – Direct Color (без палитры).

Дисплей содержит встроенный видеоконтроллер SSD1339 с графическим ускорителем первичной поверхности. Ускоритель поддерживает следующие команды:

Рисование линий
Рисование окружностей
Рисование прямоугольников
Очистка экрана
Копирование областей экрана, к сожалению, без цвета прозрачности ☹
Плавная прокрутка

Обмен данными CPU => Display не требует синхронизации – ни по арбитражу памяти, ни по синхронизации (отсутствуют эффекты мерцания, искривления, как в IBM PC- совместимых компьютерах). Поэтому возможна максимальная скорость

обмена без всяких Wait HSync, VSync!!! Стоит отметить, что даже в регистре состояния OLED- дисплея нет таких битов…

Частота обновления экрана регулируется и может быть до 180 Hz включительно. Гибкое управление контрастностью, времянками развёртки и т.п.

В играх используется программная буферизация. Часть внешней памяти отводится под вторичный видеобуфер (поверхность). Там и происходит рендеринг спрайтов. Отрендерённый кадр “выплёвывается” в память дисплея (первичная поверхность).

Буферизация позволяет гибко управлять рендерингом всей видеосистемы – отсутствие мерцания при наложении спрайтов, возможно сделать цвет прозрачности, полупрозрачности и много чего ещё другого…

```
OLED_Command    *(volatile u16*) 0x40000000 //Command Register
OLED_Data       *(volatile u16*) 0x40000002 //Data Register
```

Регистры дисплея сделаны 16 битным, так как в режиме 262144 цветов необходимо передать 18 бит (два раза по 9 бит). Фактически играют роль только младшие 9 бит шины данных.

JoyStick:

В AuMv1 применён джойстик игровой приставки SEGA Genesis (MegaDrive). Джойстик содержит мультиплексированные параллельные линии вывода. Мультиплексирование однобитовое. Опрос клавиш джойстика происходит в обработчике прерывания таймера-счётчика (канал 0). Обработчик предусматривает возможность вставки пользовательской функции, если необходимо.

До этого была предпринята попытка “посадить“ джойстик на прерывание PIOB. Ни к чему хорошему идея не привела, так как джойстик мультиплексирован.

MMC:

Сразу оговорюсь, что не проверял работу на SD карточке, поэтому далее гарантировано справедливо для MMC карточки.

MMC подключена по SPI. Можно было по MCI, но он оттолкнул Errat’ой и малой распространённостью, по сравнению с SPI режимом (Native mode для всех мультимедиа- карт).

MMC инициализируется на частоте SPI 400 kHz. После инициализации тактовая частота 15 MHz.

Проверена функция чтения секторов MMC. Функция записи есть, но она не проверялась в составе всего комплекса AuMAPI.

DMA (PDC) пока не задействован. Да я думаю надо ли ?

COM:

Последовательный порт (RS232). Используется для программирования, отладки или передачи данных. Модули пока не написаны.

USB:

Используется для программирования и питания схемы. Непаханое поле…

DataFlash:

Используется для хранения загрузчика (Loader). Модули пока не написаны. Да и вообще, туда лучше ничего не писать (для чего и предназначен джампер Write Protect красного цвета). Прошивался мной редко. Можно дополнительно использовать для хранения установок (типа CMOS Setup как в IBM PC).

3. Структура модулей AuMAPI

Type.c – описание всех типов (для того чтоб длинно не писать “unsigned long int”, а просто u32)

Internal.c – все внутренние регистры AT91RM9200
External.c – регистры внешней периферии
Other.c – остальные внутренние регистры, которые не вошли в Internal.c

Function.c – несколько функций

StartUp.c – C- часть стартапа
StartUp.s – ASM- часть стартапа
SU.s – мини-стартап для приложений

YMF721.c – базовые функции YMF721
VS1003.c – базовые функции VS1003

Joystick.c – всё про Джойстик
OLED.c – всё про OLED
MMC.c – всё про MMC

ADLIB.c – плеер Adlib
MPU401.c – плеер MIDI
CODEC.c – плеер Аудио-Декодера

Font8x8.c – шрифт 8x8 (сграблен с VideoBIOS’а видеокарты) и функции вывода текста

PuMAPI.c – Ported MicroMachine API (портированное API с AVR ATmega128 МикроМашины. Позволяет портировать за минуту все старые приложения, написанные под AVR МикроМашину!!!)

4. Прототипы некоторых AuMAPI- функций

```
u32 RandomSeed=3; //Начальное значение случайного числа
u32 Random(u32 m) //Случайное число 0..0xFFFFFFFF
void SimpleDelay(u32 d) //Простая задержка

void OutFM0(u8 Address0,u8 Data) //Запись в FM синтезатор (Register Array 0)
u8 InFM0(u8 Address0) //Чтение из FM синтезатора (Register Array 0)
void OutFM1(u8 Address1,u8 Data) //Запись в FM синтезатор (Register Array 1)
void OutWT(u8 Address,u8 Data) //Запись в WT синтезатор
void OutMPU401(u8 Data) //Запись в MPU401
void ClearFM0(void) //Очистка FM синтезатора (Register Array 0)
void ClearFM1(void) //Очистка FM синтезатора (Register Array 1)
void ClearMPU401(void) //Очистка MPU401

void OutVLSICommand(u8 Address,u16 Data) //Запись команды
void OutVLSIData(u8 Data) //Запись данных

#define JOYSTICK_U 0x01
#define JOYSTICK_D 0x02
#define JOYSTICK_L 0x04
#define JOYSTICK_R 0x08
#define JOYSTICK_A 0x10
#define JOYSTICK_B 0x20
#define JOYSTICK_C 0x40
#define JOYSTICK_S 0x80
volatile u8 Joystick=0xFF; //76543210 => SCBARLDU
void Joystick_Prepare(u32 Function, u8 Priority) //Готовит Joystick к работе

void PrepareOLED(u8 Mode)

s8 DetectMMC(void) //Наличие MMC
s8 PrepareMMC(void) //Подготовка MMC к работе
s8 InMMC(u32 Sector) //Чтение сектора MMC

#define TIMER_ST     0 //Источник прерываний - Системный таймер 32768 Hz
#define TIMER_YMF721 1 //Источник прерываний - Таймеры YMF721 12500 Hz, 3125 Hz
#define NOLOOP 0 //Одиночное воспроизведение
#define LOOP   1 //Циклическое воспроизведение
void ADLIB_Prepare(u8 Source,u8 Priority) //Готовит ADLIB к работе
void ADLIB_Load(const u8* Data) //Загрузка данных ADLIB
void ADLIB_Volume(u8 Left,u8 Right) //Установка громкости ADLIB
void ADLIB_Start(u8 Loop) //Начало воспроизведения ADLIB
void ADLIB_Stop(void) //Остановка воспроизведения ADLIB
void ADLIB_Continue(void) //Продолжение воспроизведения ADLIB

void MPU401_Prepare(u8 Source,u8 Priority) //Готовит MPU401 к работе
void MPU401_Load(const u8* Data) //Загрузка данных MIDI, вычисление и установка
частоты Таймера
void MPU401_Volume(u8 Volume) //Установка громкости MIDI
void MPU401_Start(u8 Loop) //Начало воспроизведения MIDI
void MPU401_Stop(void) //Остановка воспроизведения MIDI
```

```
void MPU401_Continue(void) //Продолжение воспроизведения MIDI

void CODEC_Prepare(u8 Priority) //Готовит CODEC к работе
void CODEC_Volume(u8 Left,u8 Right) //Установка громкости
void CODEC_Equalizer(u8 BassFrequency,u8 BassAmplification,u8 TrebleFrequency,s8
TrebleAmplification) //Эквалайзер
void CODEC_Start(u32 Start,u32 Number,u8 Loop) //Начало воспроизведения
void CODEC_Stop(void) //Остановка воспроизведения
void CODEC_Continue(void) //Продолжение воспроизведения

void OutChar(u8 x,u8 y,u8 c,u8 k)
void OutString(u8 x,u8 y,u8* s,u8 k)
void OutNumber(u8 x,u8 y,u16 n,u8 k)

#define KeyUP    JOYSTICK_U
#define KeyDOWN  JOYSTICK_D
#define KeyLEFT  JOYSTICK_L
#define KeyRIGHT JOYSTICK_R
#define KeyFIGHT JOYSTICK_A
#define KeyJUMP  JOYSTICK_B
#define KeySHIFT JOYSTICK_C
#define KeyPAUSE JOYSTICK_S
#define SpeakerNO    0x00
#define SpeakerLEFT  0x40
#define SpeakerRIGHT 0x80
#define SpeakerBOTH  0xC0
#define ChannelON  0xF0
#define ChannelOFF 0x00
#define Key Joystick
u8 UnUsed;
#define LED UnUsed
u8 InvisibleColor;
u8 BackgroundColor;
u16 Seed=0;
u16 Random(void)
void delay_ms(u32 d)
u8 InPixel(s8 X,s8 Y)
void OutPixel(s8 X,s8 Y,u8 C)
void OutSprite(u16 I,s8 X,s8 Y)
void ClearVRAM(void)
void OutLCD(void)
void uMStart(void)
void PrepareLCD(void)
void ResetFM(void)
void OutFM(u8 Bank,u8 Address,u8 Data)
void LoadPatch(u8 Patch,u8 Channel)
void ParameterChannel(u8 Channel,u8 Block,u16 Frequency)
void PanoramChannel(u8 Channel,u8 Mode)
void SoundFM(u8 Channel,u8 Mode)
```

5. Пример AuMAPI- приложения (плеер MID- файлов)

```
#include "Type.c" //Type Definition
#include "Internal.c" //Internal Registers
#include "External.c" //External Registers
#include "Other.c" //Other Registers
#include "Function.c" //Some Functions
#include "StartUp.c" //C-part of StartUp
#include "Joystick.c" //Joystick Routine
#include "YMF721.c" //YMF721 Routine
#include "MPU401.c" //MPU401 Player
#include "MusicMIDI.c" //MID SMF0 Music Data

main(void)
{
 u8 V=127; //Maximum Volume
 Joystick_Prepare(0x00000000,6); //Set Joystick IRQ Handler - No User Function, IRQ Priority=6
 MPU401_Prepare(TIMER_YMF721,7); //Set YMF721 IRQ Handler - IRQ Source=YMF721Timer,
IRRQ Priority=7, Initialize MPU401
 MPU401_Load(MusicMIDI); //Pass Music Data Pointer
 MPU401_Volume(V); //Set Volume
 MPU401_Start(LOOP); //Start Play IRQ Routine with LOOP after End... now playing MIDI File ;)
 while(1) //Joystick Keys
 {
  if(!(Joystick&JOYSTICK_S)) //if pressed Start
  {
   SimpleDelay(8000000);
   MPU401_Start(LOOP); //Start again
  }
  if(!(Joystick&JOYSTICK_U)) //if pressed Up
  {
   SimpleDelay(800000);
   if(V<127) V++;
   MPU401_Volume(V); //Increase Output Volume
  }
  if(!(Joystick&JOYSTICK_D)) //if pressed Down
  {
   SimpleDelay(800000);
   if(V>0) V--;
   MPU401_Volume(V); //Decrease Output Volume
  }
  if(!(Joystick&JOYSTICK_A)) //if pressed A
  {
   SimpleDelay(8000000);
   MPU401_Stop(); //Stop Playing (Pause)
  }
  if(!(Joystick&JOYSTICK_B)) //if pressed B
  {
   SimpleDelay(8000000);
   MPU401_Continue(); //Continue (Resume) Playing
  }
 }
}
```

Romanich
24.04.08