68000 Assembly programming for the Atari ST

Copied from https://www.chibiakumas.com/68000/atarist.php

Memory Map

Address	Purpose
000000	512k ram
07FFFF	
FA0000	128K Rom
FC0000	System Rom
FEFFF	
FF8000	Io Area
FF8200	Io Area
FF8400	Io Area
FF8600	Io Area
FF8800	Io Area SOUND AY-3-8910
FFFA00	Io Area MFP 68901
FFFC00	Io Area 2 ACIA:s 6580

Registers

Because the 68000 uses a 24 bit address bus, These can also be addressed at $FFFFxxxx - for example $FF8004 is the same as $FFFF8004

Address	Mode	Bits	Purpose
FF8004	RW	----xxxx	Memory Config
FF8201	RW	HHHHHHHH	Video Base H
FF8202	EW	LLLLLLLL	Video Base L
FF8205	R	HHHHHHHH	Video Counter H (current drawing line)
FF8207	R	MMMMMMMM	Video Counter M
FF8209	R	LLLLLLLL	Video Counter L
FF820A	RW	------SR	Sync Mode
FF8240	RW	-----RRR -GGG-BBB	Palette Color 0
FF8242	RW	-----RRR -GGG-BBB	Palette Color 1
FF8244	RW	-----RRR -GGG-BBB	Palette Color 2
FF8246	RW	-----RRR -GGG-BBB	Palette Color 3
FF8248	RW	-----RRR -GGG-BBB	Palette Color 4
FF824A	RW	-----RRR -GGG-BBB	Palette Color 5
FF824C	RW	-----RRR -GGG-BBB	Palette Color 6
FF824E	RW	-----RRR -GGG-BBB	Palette Color 7
FF8250	RW	-----RRR -GGG-BBB	Palette Color 8
FF8252	RW	-----RRR -GGG-BBB	Palette Color 9

FF8254	RW	-----RRR -GGG-BBB	Palette Color 10
FF8256	RW	-----RRR -GGG-BBB	Palette Color 11
FF8258	RW	-----RRR -GGG-BBB	Palette Color 12
FF825A	RW	-----RRR -GGG-BBB	Palette Color 13
FF825C	RW	-----RRR -GGG-BBB	Palette Color 14
FF825E	RW	-----RRR -GGG-BBB	Palette Color 15
FF8260	RW	------SS	Screen Mode (00=320x240x4bpp)
FF8400			Reserved
FF8600			Reserved
FF8602			Reserved
FF8604		-------- XXXXXXXX	Disk controler data access
FF8606	R	-------- -----DSE	DMA Status Mode control
FF8606	W	-------W FD0RGAB	DMA Mode Control
FF8609	RW	HHHHHHHH	Dna Base & Counter H
FF860B	RW	MMMMMMMM	Dna Base & Counter M
FF860D	RW	LLLLLLLL	Dna Base & Counter L
FF8800	R	DDDDDDDD	IO Port B Data
FF8800	W	RRRRRRRR	PSG Register Select
FF8802	W	RGCDRFGH	PSG Write Data IO Port A
FF8802	RW	DDDDDDDD	IO Port B Data
FFFA01		M-IW---P	MFP GP IO
FFFA03		DDDDDDDD	MFP Active Edge
FFFA05		DDDDDDDD	MFP Data Direction
FFFA07		DDDDDDDD	MFP Interrupt Enable A
FFFA09		DDDDDDDD	MFP Interrupt Enable B
FFFA0B		DDDDDDDD	MFP Interrupt Pending A
FFFA0D		DDDDDDDD	MFP Interrupt Pending B
FFFA0F		DDDDDDDD	MFP Interrupt Interrupt In Service A
FFFA11		DDDDDDDD	MFP Interrupt Interrupt In Service B
FFFA13		DDDDDDDD	MFP Interrupt Mask A
FFFA15		DDDDDDDD	MFP Interrupt Mask B
FFFA17		DDDDDDDD	MFP Vector Base
FFFA19		DDDDDDDD	MFP Timer A Control
FFFA1B		DDDDDDDD	MFP Timer B Control
FFFA1D		DDDDDDDD	MFP Timers C & D Control
FFFA1F		DDDDDDDD	MFP Timer A data
FFFA21		DDDDDDDD	MFP Timer B data
FFFA23		DDDDDDDD	MFP Timer C data
FFFA25		DDDDDDDD	MFP Timer D data
FFFA27		DDDDDDDD	MFP Sync Character
FFFA29		DDDDDDDD	MFP USART control reg
FFFA2B		DDDDDDDD	MFP receiver status
FFFA2D		DDDDDDDD	MFP Transmitter status

FFFA2F	DDDDDDDD	MFP USART Data
FFFC00	DDDDDDDD	Keyboard ACIA Control
FFFC02	DDDDDDDD	Keyboard Data
FFFC04	DDDDDDDD	Midi ACIA Control
FFFC06	DDDDDDDD	Midi Data

Aline commands

Aline commands are defined by DW $A0xx - these are handled by the vectors of the 68000 like an RST command on the Z80... see Anatomy of the Atari St for more details

Command	Command	Detail
$A000	Initialize	Determine address of required variable range
$A001	Put Pixel	Set point on the screen
$A002	Get Pixel	Determine color of a screen point
$A003	Line	Draw a line on the screen
$A004	Horizontal Line	Draw a horizontal line (fast)
$A005	Filled Rectangle	Fill a rectangele wit color
$A006	Filled Polygon	Fill a polygon line by line
$A007	BitBlt	Bit block transfer
$A008	TextBlt	Text block transfer
$A009	Show Mouse	enable mouse cursor
$A00A	Hide Cursor	Disable mouse cursor
$A00B	Transform Mouse	Change mouse cursor form
$A00C	Undraw Sprite	Clear sprite
$A00D	Draw Sprite	Enable sprite
$A00E	Copy Raster From	Copy raster form

Vblank!

It's possible to 'Wait for Vblank' by using the line counter at $FF8205/7/9.

First we calculate the address of the last line of our 32k screen, in this case by adding 320*200/2 to 'ScreenBase'

Next we wait for the line counter to point to this line - at this point the bottom of the screen is being drawn,

Here is an example of a possible solution - this assumes memory address 'ScreenBase' has a long pointer of the address of the first line of the screen.

```
waitVBlank:
   moveM.l d0-d1,-(sp)
      move.l ScreenBase,d1     ;Get
screen pointer
      add.l #(320*200/2),d1    ;Add
one full screen

WaitVblankAgain:
      jsr VblankGetLine
      cmp.l d1,d0
      blt WaitVblankAgain      ;Wait
for last line to be drawn

WaitVblankAgain2:
      jsr VblankGetLine
      cmp.l d0,d1
      beq WaitVblankAgain2     ;Wait
for line to end
   moveM.l (sp)+,d0-d1
   rts
```

```
VblankGetLine:
   clr.l d0
   or.b $FF8205,d0        ;Get current drawing line
   lsl.l #8,d0            ;from hardware regs
   or.b $FF8207,d0
   lsl.l #8,d0
   or.b $FF8209,d0
   rts
```

Joystick reading

I'm not a fan of using the firmware for doing things, but I can't find any other way to read the joysick on the Atari ST

The Atari ST joystick is connected as part of the keyboard... to read it we have to install a Joystick event interrupt handler...

Shown to the right is the simplest joystick handler I have working...
we store the two joystick bytes to a temporary variable called "Joydata"

```
    move.w    #$14,-(sp)    ;IKBD command $14 - set joystick event reporting
   move.w    #4,-(sp)    ;Device no 4 (keyboard - Joystick is part of keyboard)
   move.w    #3,-(sp)    ;Bconout (send cmd to keyboard)
   trap    #13           ;BIOS Trap
   addq.l    #6,sp        ;Fix the stack

   move.w  #34,-(sp)    ;return IKBD vector table (KBDVBASE)
   trap      #14          ;XBIOS trap
   addq.l  #2,sp         ;Fix the stack

   move.l  d0,ikbd_vec     ; store IKBD vectors address for later
   move.l  d0,a0             ; A0 points to IKBD vectors
   move.l  24(a0),old_joy     ; backup old joystick vector so we can restore it
   move.l  #JoystickHandler,24(a0); Set our Joystick Handler
   rts

JoystickHandler:
   move.b  (1,a0),joydata          ; store joy 0 data
   move.b  (2,a0),joydata+1     ; store joy 1 data
   rts

ikbd_vec:    dc.l 0            ; old IKBD vector storage
old_joy:    dc.l 0           ; old joy vector storage
joydata:    ds.b 2           ;Joypad bytes F----RLDU
```

AY Sound Chip:

The AY uses a series of 8 bit registers to control the 3 sound channels.

To set a register, we must first select the register number by writing a byte to **$FF8800**, then send the byte of data to **$FF8802**

Register	Meaning	Bit Meaning	Details
0	Tone Pitch L - Channel A	LLLLLLLL	Lower value = Higher pitch
1	Tone Pitch H - Channel A	----HHHH	Lower value = Higher pitch
2	Tone Pitch L - Channel B	LLLLLLLL	Lower value = Higher pitch
3	Tone Pitch H - Channel B	----HHHH	Lower value = Higher pitch
4	Tone Pitch L - Channel C	LLLLLLLL	Lower value = Higher pitch
5	Tone Pitch H - Channel C	----HHHH	Lower value = Higher pitch
6	Noise Generator	---NNNNN	Higer = Faster noise
7	Mixer	--NNNTTT	N=Noise T=Tone (Channel --CBACBA 1=mute 0=normal)
8	Amplitude - Channel A	---EVVVV	E=Envelope (1=Enabled) VVVV=Volume
9	Amplitude - Channel B	---EVVVV	E=Envelope (1=Enabled) VVVV=Volume
10	Amplitude - Channel C	---EVVVV	E=Envelope (1=Enabled) VVVV=Volume
11	Envelope L (Volume over time)	LLLLLLLL	Lower=Faster Envelope
12	Envelope H (Volume over time)	HHHHHHHH	Lower=Faster Envelope
13	Envelope Selection	----EEEE	Envelope number (See PDF)

Traps

Traps are used for firmware and OS functions... commands need to be pushed onto the stack before the call... see Anatomy of the Atari St for more details

Trap	Num	Command	Details
#1 (Gemdos)	**$00**	Term	
	$01	ConIn	
	$02	ConOut	
	$03	Auxilliary Input	
	$04	Auxilliary Output	
	$05	Printer Output	
	$06	RawConIO	
	$07	Direct ConIn Without Echo	
	$08	ConIn Without Echo	
	$09	Print Line	
	$0A	ReadLine	
	$0B	ConStat	
	$0E	SetDrv	
	$10	ConOut Stat	

	$11	PrtOut Stat	
	$12	AuxIn Stat	
	$13	AuxOut Stat	
	$19	Current Disk	
	$1A	Set Disk Transfer Access	
	$20	Super	
	$2A	Get Date	
	$2B	Set Date	
	$2C	Get Time	
	$2D	Set Time	
	$2F	Get DTA	
	$30	Get Version Number	
	$31	Keep Process	
	$36	Get Free Disk Space	
	$39	MKDir	
	$3A	RmDir	
	$3B	ChDir	
	$3C	Create	
	$3D	Open	
	$3E	Close	
	$3F	Read	
	$40	Write	
	$41	Unlink	
	$42	Lseek	
	$43	Change Mode (ChMod)	
	$45	Dup	
	$46	Force	
	$47	GetDir	
	$48	Malloc	
	$49	Mfree	
	$4A	SetBlock	
	$4B	Exec	
	$4C	Term	
	$4E	Sfirst	
	$4F	Snext	
	$56	Rename	
	$57	GSDTOF	
#13 (Bios)	**0**	getmpb	Get Memory Parameter Block
	1	bconstat	Return input device status
	2	conin	Read character from device
	3	bconout	Write character to device
	4	rwabs	Read and write disk sector
	5	setexec	set exception vectors

	6	tickcal	return millisecond per tick
	7	get bpb	get BIOS parameter block
	8	bcostat	return output device status
	9	mediach	inquire media change
	10	drvmap	inquire drive status
	11	kbshift	inquire/change keyboard status
#14 (xbios)	**0**	initmous	initialize mouse
	1	ssbrk	save memory space
	2	physbase	return screen RAM base address
	3	logbase	set logical screen base
	4	getrez	return screen resolution
	5	setscreen	Set Screen parameters
	6	setpalette	set color palette
	7	setcolor	set color palette
	8	floprd	read diskette sector
	9	flopwr	write disk sector
	10	flopfmt	format diskette
	12	midiws	write string to midi interface
	13	mfpint	initilize MFP format
	14	iorec	return record buffer
	15	rsconf	set rs232 configuration
	16	keytbl	set keyboard table
	17	random	return random number
	18	protobt	produce boot sector
	19	flopver	verify diskette sector
	20	scrdmp	output screen dump
	21	cursconf	set cursor configuration
	22	settime	set clock time and date
	23	gettime	return clock time and date
	24	bioskeys	restore keyboard table
	25	ikbdws	intelligent keyboard send
	26	jdisint	disable interrupts on MFP
	27	jenabint	enable interrupts on MFP
	28	giaaccess	access GI sound chip
	29	offgibit	reset Port A GI sound chip
	30	ongibit	cleart port A of GI sound chip
	31	xbtimer	start MFP timer
	32	dosound	Set sound parameters
	33	setprt	Set printer config
	34	kbdvbase	return keyboard vector table
	35	kbrate	set keyboard repeat rate
	36	prtblk	output block to printer
	37	wvbl	wait for video

38	supexec	set supervisor execution
39	puntaes	disable AES

KbdvBase

Xbios trap #34 returns the Keyboard vector table, this contains 7 Long addresses

Offset	Value (Long)	Purpose
0	MidiVec	Midi Input
4	VkbdErr	Keyboard Error
8	VmidErr	Midi Error
12	StatVec	IKBD Status
16	MouseVec	Mouse Routines
20	CockVec	Clock Time Routines
24	JoyVec	Joystick Routines

IKBD commands

Number	Purpose
$07	Returns the result of pressing one of the two mouse button
$08	Returns the relative mouse position from now on.
$09	Returns the absolute mouse position from now on
$0A	With this command it is possible to get the key numbers of the cursor keys instead of the coordinates.
$0B	This command sets the trigger threshold, above which movements will be announced.
$0C	Scale mouse.
$0D	Read absolute mouse position.
$0E	Set the internal coordinate counter
$0F	Set the origin for the Y-axis is down (next to the user)
$10	Set the origin for the Y-axis is up.
$11	The data transfer to the main processor is permitted again (see $13).
$12	Turn mouse off.
$13	Stop data transfer to main processor
$14	Every joystick movement is automatically returned.
$15	End the automatic-return mode for the joystick.
$16	Read joystick.
$17	Joystick duration message.
$18	Fire button duration message.
$19	Cursor key simulation mode for joystick 0 (!).
$1A	Turn off joysticks.
$1B	Set clock time.
$1C	Read clock time.
$20	Load memory.

$21 Read memoiy.
$22 Execute routine.