AGB System Call Reference Manual

Version 1.0

Revision History

Date	Version	Description
12/01/2000	1.0	-Added description of sound driver -Changes to system call -Support for TS2 system ROM -Support for final system ROM -Deletions of description along with inability to use Music Player -Changed format of Index -Changed use of “Cassette” to “Cartridge” -Changed name of document from “AGB System Call Reference” to “AGB System Call Reference Manual”.

Notes regarding the use of system calls

Please refer to the C language headers for system calls when using this Reference.

ArcTan

Type

s16ArcTan(s16 Tan)

Function

Calculates the arctangent.

The return value returns -pi/2< theta < pi/2 in a range of 0xc000-0x4000.

However, there is a problem in accuracy with, theta < -pi/4, pi/4 < theta.

Arguments

s16	Tan	Sign: 1bit
		Integer: 1bit
		Decimal: 14bit

ArcTan2

Type

u16ArcTan2(s16 X, s16 Y)

Function

Calculates the arctangent after correction processing.

Use this in normal situations.

The return value returns 0 < or = theta < 2pi in a range of 0-0xffff.

Arguments

s16	X,Y	Sign: 1bit
		Integer: 1bit
		Decimal: 14bit

BgAffineSet

Type

void BgAffineSet(BgAffineSrcData* Srcp, BgAffineDestData* Destp, s32 Num)

Function

Calculates and sets the BG affine parameters from center coordinates of the data and the display, as well as the scaling ratio and the angle of rotation.

Based on the parameters set in Srcp, the affine parameters are calculated and set in Destp.

When Srcp is an array, the calculation can be performed repeatedly by specifying Num.

Using the calculated data, a rotated/scaled BG can be translated, rotated and scaled.

Arguments

BgAffineSrcData*	Srcp	Source address
BgAffineDestData*	Destp	Destination address
s32	Num	Number of calculations

BgAffineSrcData Structure

s32	SrcCenterX	Original data's center X coordinate (8bit fractional portion)
s32	SrcCenterY	Original data's center Y coordinate (8bit fractional portion)
s16	DispCenterX	Display's center X coordinate
s16	DispCenterY	Display's center Y coordinate
s16	RatioX	Scaling ratio in the X direction (8bit fractional portion)
s16	RatioY	Scaling ratio in the Y direction (8bit fractional portion)
u16	Theta	Angle of rotation (8bit fractional portion), Effective Range 0 – 0xffff

BgAffineDestData Structure		
s16	H_DiffX	Difference in X coordinate along same line
s16	V_DiffX	Difference in X coordinate along next line
s16	H_DiffY	Difference in Y coordinate along same line
s16	V_DiffY	Difference in Y coordinate along next line
s32	StartX	Start X coordinate
s32	StartY	Start Y coordinate

BitUnPack

Type

void BitUnPack(void* Srcp, void* Destp, BitUnPackParam* BitUnPackParamp)

Function

Expands data packed with 0 fixed bit.

Align the destination address to a 4Byte boundary.

Arguments

void*	Srcp	Source address
void*	Destp	Destination address
BitUnPackParam*	Paramp	BitUnPackParam structure data address

BitUnPackParam structure		
u16	SrcNum	Source Data Byte Size
u8	SrcBitNum	1 Source Data Bit Number
u8	DestBitNum	1 Destination Data Bit Number
u32	DestOffset:31	Offset value to add to source data
	DestOffset0_On:1	Flag for whether or not to add offset to 0 data

CpuFastSet

Type

void CpuFastSet(void* Srcp, void* Destp, u32 DmaCntData)

Function

Use CPU to transmit data quickly between memory addresses

It is a 32 bit transfer in 32 byte (8 word) units.

If the argument is set outside of a 4 byte boundary, access is done forcibly with a 4 byte boundary.

Arguments

void*	Srcp	Source address
void*	Destp	Destination address
u32	DmaCntData	Only DMA_SRC_FIX/DMA_COUNT_MASK is effective. DMA_SRC_FIX(0,1) = (Source address/ increment, source address fixed) DMA_COUNT_MASK & DmaCntData= number of transfers

Upper Macro

CpuFastClear, CpuFastArrayClear, CpuFastCopy, CpuFastArrayCopy

CpuSet

Type

void CpuSet(void* Srcp, void* Destp, u32 DmaCntData)

Function

Use CPU to transmit data between memory addresses

With a 32bit transfer, access is done forcibly with a 4 byte boundary, however with 16 bit transfer, you need to use a 2 byte boundary with the argument.

Arguments

void*	Srcp	Source address
void*	Destp	Destination address
u32	DmaCntData	Only DMA_SRC_FIX / DMA_32BIT_BUS / DMA_COUNT_MASK are effective. DMA_SRC_FIX(0,1) = (Source address increment, source address fixed) DMA_32BIT_BUS(0,1)=(16bit transfer, 32bit transfer) DMA_COUNT_MASK & DmaCntData = number of transfers

Upper Macro

CpuClear, CpuArrayClear, CpuCopy, CpuArrayCopy

Diff16BitUnFilter

Type

void Diff16BitUnFilter(void* Srcp, void* Destp)

Function

Expand 16bit-difference filtered data and write in units of 16bits.

Align the source address to a 4Byte boundary.

Arguments

void*	Srcp	Source address
void*	Destp	Destination address

Subject data format

Data header

u32	ByteSize:4	1 data: byte size (=2)
	FilterType:4	Filter type (=8)
	DestSize:24	Data size after expansion

Data format

u8	Origin	Original data
u8	Diff	Difference data
	:	

Diff8BitUnFilterVram

Type

void Diff8BitUnFilterVram(void* Srcp, void* Destp)

Function

Expand 8bit-difference filtered data and write in units of 16bits.

Can also be expanded in WorkRAM, but that is slower than Diff8BitUnFilterWram().

Align the source address to a 4Byte boundary.

Arguments

void*	Srcp	Source address
void*	Destp	Destination address

Subject data format

Data header

u32	ByteSize:4	1 data: byte size (=1)
	FilterType:4	Filter type (=8)
	DestSize:24	Data size after expansion

Data format

u8	Origin	Original data
u8	Diff	Difference data
	:	

Diff8BitUnFilterWram

Type

void Diff8BitUnFilterWram(void* Srcp, void* Destp)

Function

Expand 8bit-difference filtered data and write in units of 8bits.

Cannot be expanded in VRAM

Align the source address to a 4Byte boundary.

Arguments

void*	Srcp	Source address
void*	Destp	Destination address

Subject data format

Data header		
u32	ByteSize:4	1 data byte size (=1)
	FilterType:4	Filter type (=8)
	DestSize:24	Data size after expansion
Data format		
u8	Origin	Original data
u8	Diff	Difference data
	:	

Div/DivArm

Type

s32 Div(s32 Number,s32 Denom)	Supports Red Hat’s(formerly Cygnus) library
s32 DivArm(s32 Denom, s32 Number)	Supports Arm’s library

Function

Computes the quotient of a signed division calculation.

Returns the calculation result of Number ÷ Denom.

The register values are reset to: r0=Number/Denom, r1=Number%Denom, r3=|Number/Denom|.

Arguments

s32	Number	Numerator
s32	Denom	Denominator

DivRem/DivRemArm

Type

s32 DivRem(s32 Number,s32 Denom)   Supports Red Hat's(formerly Cygnus) library

s32 DivRemArm(s32 Denom, s32 Number)   Supports Arm's library

Function

Computes the remainder of a signed division calculation.

Returns the calculation result of Number % Denom

The register values are reset to: r0=Number%Denom, r1=Number%Denom, r3=|Number/Denom|

Arguments

s32	Number	Numerator
s32	Denom	Denominator

Halt

Macro

Halt()

Function

Stops only the CPU.

Will resume when the interrupt request that is set in the IE register is set in the IF register.

HuffUnComp

Type

void HuffUnComp(void* Srcp, void* Destp)

Function

Expands Huffman-compressed data and writes in units of 32bits.

If the size of the compressed data is not a multiple of 4, please adjust it as much as possible by padding with 0.

Align the source address to a 4Byte boundary.

Arguments

void*	Srcp	Source address
void*	Destp	Destination address

Subject data format

Data header		
u32	BitSize:4	1 data: bit size (normally 4|8)
	CompType:4	Compressed type (=2)
	DestSize:24	Data size after expansion
Tree table		
u8	TreeSize	Tree table size / 2-1
TreeNodeData	RootNode	Root node
TreeNodeData	LeftNode	Root left node
TreeNodeData	RightNode	Root right node
TreeNodeData	LeftLeftNode	Left left node
TreeNodeData	LeftRightNode	Left right node
TreeNodeData	RightLeftNode	Right left node
TreeNodeData	RightRightNode	Right right node
The compressed data, after data header + tree table		

TreeNodeData structure		
u8	NextNodeOffset:6	Offset to next node data -1 (2byte units)
	RightEndFlag:1	Right node end flag (If end flag is set, data is in next node)
	LeftEndFlag:1	Left node end flag

IntrWait

Macro

IntrWait(u8 InitCheckClear, u16 IntrFlags)

Function

Continues to wait in Halt status until the interrupt specified by IntrFlags occurs.

Set a flag with the interrupt routine that corresponds to INTR_CHECK_BUF(0x3007ff8).

When using multiple interrupts at the same time, the overhead for calling system calls can be decreased when compared to repeatedly calling Halt().

Arguments

u8	InitCheckClear	Specification of whether or not to clear if an appropriate flag has been set.
u16	IntrFlags	Specification of interrupt wait (See AgbDefine.h)

LZ77UnCompVram

Type

void LZ77UnCompWram(void* Srcp, void* Destp)

Function

Expands LZ77-compressed data and writes in units of 16bits.

The data can also be expanded in work RAM, but that is slower than with LZ77UnCompVram().

Search the compressed data for matching character strings of less than 2 Bytes.

If the size of the compressed data is not a multiple of 4, please adjust it as much as possible by padding with 0.

Align the source address to a 4-Byte boundary.

Arguments

void*	Srcp	Source address
void*	Destp	Destination address

Subject data format

Data header		
u32	:4	Reserved
	CompType:4	Compressed type (=1)
	DestSize:24	Data size after expansion
Flag data		
u8	Flags	Compressed / uncompressed flag
		0: Uncompressed data
		1: Compressed data
LZ77 compressed code data	(Big Endian)	
u16	Length:4	Length of expanded data - 3
		(Compress if matching length over 3Bytes)
	Offset:12	Matching data offset (>=2) - 1

LZ77UnCompWram

Type

void LZ77UnCompWram(void* Srcp, void* Destp)

Function

Expands LZ77-compressed data and writes in units of 8bits.

The data can not be expanded in VRAM.

If the size of the compressed data is not a multiple of 4, please adjust it as much as possible by padding with 0.

Align the source address to a 4-Byte boundary.

Arguments

void*	Srcp	Source address
void*	Destp	Destination address

Subject data format

Data header		
u32	:4	Reserved
	CompType:4	Compressed type (=1)
	DestSize:24	Data size after expansion
Flag data		
u8	Flags	Compressed / uncompressed flag
		0: Uncompressed data
		1: Compressed data
LZ77 compressed code data	(Big Endian)	
u16	Length:4	Length of expanded data - 3
		(Compress if matching length over 3Bytes)
	Offset:12	Matching data offset - 1

MidiKey2Freq

Type

u32 MidiKey2Freq(WaveData* wa, u8 mk, u8 fp)

Function

Calculates the value of the assignment to ((SoundArea)sa).vchn[x].fr when playing the wave data, wa, with the interval (MIDI KEY) mk and the fine adjustment value (halftones=256) fp.

MultiBoot

Type

Int MultiBoot(MultiBootParam* mp)

Function

Main processing for multi-play boot server.

The standard recognition procedures must be done between all of the connected client AGB units in advance.

ObjAffineSet

Type

void ObjAffineSet(ObjAffineSrcData* Srcp, void* Destp, s32 Num, s32 Offset)

Function

Calculates and sets the OBJ's affine parameters from the scaling ratio and angle of rotation.

The affine parameters are calculated from the parameters set in Srcp. The four affine parameters are set every Offset bytes, starting from the Destp address.

If the Offset value is 2, the parameters are stored contiguously. If the value is 8, they match the structure of OAM.

When Srcp is arrayed, the calculation can be performed continuously by specifying Num.

Arguments

ObjAffineSrcData*	Srcp	Source address
void*	Destp	Destination address
s32	Num	Number of calculations
s32	Offset	Offset of parameter address Number of bytes (Normally 2 | 8) Specify 8 when setting directly in OAM

ObjAffineSrcData Structure

s16	RatioX	Scaling ratio in the X direction (8bit fractional portion)
s16	RatioY	Scaling ratio in the Y direction (8bit fractional portion)
U16	Theta	Angle of rotation (8bit fractional portion), Effective Range 0 – 0xffff

ObjAffineDestData Structure

s16	H_DiffX	Difference in X coordinate along same line
s16	V_DiffX	Difference in X coordinate along next line
s16	H_DiffY	Difference in Y coordinate along same line
s16	V_DiffY	Difference in Y coordinate along next line

RegisterRamReset

Type

void RegisterRamReset(u32 ResetFlags)

Function

Resets the registers and RAM specified with ResetFlags.

However, it does not clear the CPU internal RAM area from 0x3007e00-0x3007fff.

Arguments

u32	ResetFlags	Specification of register and RAM to reset. (See AgbDefine.h)

RLUnCompVram

Type

void RLUnCompVram(void* Srcp, void* Destp)

Function

Expands run-length compressed data and writes it in units of 16bits.

The data can be expanded in Work RAM, but that is slower than with RLUnCompWram().

If the size of the compressed data is not a multiple of 4, please adjust it as much as possible by padding with 0.

Align the source address to a 4Byte boundary.

Arguments

void*	Srcp	Source address
void*	Destp	Destination address

Subject data format

Data header		
u32	:4	Reserved
	CompType:4	Compressed type (=3)
	DestSize:24	Data size after expansion
Flag data		
u8	Length:7	Expanded data length - 1 (when uncompressed)

		Expanded data length - 3 (when compressed to a concatenated length longer than 3 Bytes)
	Flag:1	0: Uncompressed data
		1: Compressed data

RLUnCompWram

Type

void RLUnCompWram(void* Srcp, void* Destp)

Function

Expands run-length compressed data and writes it in units of 8bits.

The data cannot be expanded in VRAM.

If the size of the compressed data is not a multiple of 4, please adjust it as much as possible by padding with 0.

Align the source address to a 4Byte boundary.

Arguments

void*	Srcp	Source address
void*	Destp	Destination address

Subject data format

Data header		
u32	:4	Reserved
	CompType:4	Compressed data (=3)
	DestSize:24	Data size after expansion
Flag data		
u8	Length:7	Expanded data length - 1 (when uncompressed)
		Expanded data length - 3 (when compressed to a concatenated length longer than 3 Bytes)
	Flag:1	0: Uncompressed data
		1: Compressed data

SoftReset

Type

void SoftReset(u32 ResetFlags)

Function

Resets a register and RAM specified by ResetFlags and returns to the head address of a cartridge or CPU external RAM with the value for: SOFT_RESET_DIRECT_BUF(0x03007ffa).

Do not specify RESET_EX_WRAM_FLAG when returning to the CPU external RAM.

Do not specify RESET_REG_SIO_FLAG when returning to a cartridge from a download program.

The CPU core register and the area of 0x3007e00 ~ 0x3007fff of CPU internal RAM are forcibly cleared.

Arguments

u32	ResetFlags	Specify register and RAM to be reset. (See AgbDefine.h)
(u8)	SOFT_RESET_DIRECT_BUF	Specify where to return 0 : 0x08000000 address Not 0 : 0x02000000 address

SoftResetExram

Type

void SoftResetExram(u32 ResetFlags)

Function

Resets register and RAM specified by ResetFlags and returns to the address, 0x02000000 (head of CPU external RAM).

RESET_EX_WRAM_FLAG is cleared to return to the CPU external RAM.

The CPU core register and the area of 0x3007e00 ~ 0x3007fff of CPU internal RAM are forcibly cleared.

Arguments

u32	ResetFlags	Specify register and RAM to be reset. (See AgbDefine.h)

SoftResetRom

Type

void SoftResetRom(u32 ResetFlags)

Function

Resets register and RAM specified by ResetFlags and returns to the address, 0x08000000 (head of cartridge).

Do not specify RESET_REG_SIO_FLAG so that a cartridge can distinguish from a normal start-up when returning from a download program.

The CPU core register and the area of 0x3007e00 ~ 0x3007fff of CPU internal RAM are forcibly cleared.

Arguments

u32	ResetFlags	Specify register and RAM to be reset. (See AgbDefine.h)

SoundBiasReset

Type

void SoundBiasReset(void)

Function

Changes the sound BIAS level from its mid-value (0x200) to 0.

SoundBiasSet

Type

void SoundBiasSet(void)

Function

Changes the sound BIAS level from 0 to its mid-value (0x200).

SoundChannelClear

Type

void SoundChannelClear (void)

Function

Clears all direct sound channels and stops the sound.

This function may not operate properly when the library which expands the sound driver feature is combined afterwards. In this case, do not use it.

SoundDriverInit

Type

void SoundDriverInit(SoundArea* sa)

Function

Initializes the sound driver.

Call this only once when the game starts up.

It is essential that the work area sa already be secured at the time this function is called.

You cannot execute this driver multiple times, even if separate work areas have been prepared.

Arguments

SoundArea*	sa	Work area for sound driver

SoundArea Structure

u32	ident	Flag the system checks to see whether the work area has been initialized and whether it is currently being accessed.
vu8	DmaCount	User access prohibited
u8	reverb	Variable for applying reverb effects to direct sound
u16	d1	User access prohibited
void	(*func)()	User access prohibited
int	intp	User access prohibited
void*	NoUse	User access prohibited

SoundChannel	vchn[MAX_SOUND_CHANNEL]	The structure array for controlling the direct sound channels (currently 8 channels are available). The term "channel" here does not refer to hardware channels, but rather to virtual constructs inside the sound driver.
s8	pcmbuf[PCM_BF*2]	

SoundChannel Structure

u8	sf	The flag indicating the status of this channel. When 0 sound is stopped. To start sound, set other parameters and then write 0x80 to here. To stop sound, logical OR 0x40 for a release-attached off (key-off), or write zero for a pause. The use of other bits is prohibited.
u8	r1	User access prohibited
u8	rv	Sound volume output to right side
u8	lv	Sound volume output to left side
u8	at	The attack value of the envelope. When the sound starts, the volume begins at zero and increases every 1/60 second. When it reaches 255, the process moves on to the next decay value.
u8	de	The decay value of the envelope. It is multiplied by "this value/256" every 1/60 sec. and when sustain value is reached, the process moves to the sustain condition.
u8	su	The sustain value of the envelope. The sound is sustained by this amount. (Actually, multiplied by rv/256, lv/256 and output left and right.)
u8	re	The release value of the envelope. Key-off (logical OR 0x40 in sf) to enter this state. The value is multiplied by "this value/256" every 1/60 sec. and when it reaches zero, this channel is completely stopped.
u8	r2[4]	User access prohibited
u32	fr	The frequency of the produced sound. Write the value obtained with the MidiKey2Freq function here.

WaveData*	wp	Pointer to the sound's waveform data. The waveform data can be generated automatically from the AIFF file using the tool (aif2agb.exe), so users normally do not need to create this themselves.
u32	r3[6]	User access prohibited
u8	r4[4]	User access prohibited

WaveData Structure

u16	type	Indicates the data type. This is currently not used.
u16	stat	At the present time, non-looped (1 shot) waveform is 0x0000 and forward loop is 0x4000.
u32	freq	This value is used to calculate the frequency. It is obtained using the following formula: sampling rate x 2^((180-original MIDI key)/12)
u32	loop	Loop pointer (start of loop)
u32	size	Number of samples (end position)
s8	data[ ]	The actual waveform data. Takes (number of samples + 1) bytes of 8bit signed linear uncompressed data. The last 1 byte is zero for a non-looped waveform, and the same value as the loop pointer data for a looped waveform.

SoundDriverMain

Type

void SoundDriverMain(void)

Function

Main of the sound driver.

Call every 1/60 of a second. The flow of the process is to call SoundDriverVSync(), which is explained later, immediately after the V-Blank interrupt.

After that, this routine is called after BG and OBJ processing is executed.

SoundDriverMode

Type

void SoundDriverMode(u32 mode)

Function

Sets the sound driver operation mode.

Arguments

u32	mode	Sound driver operation mode -Direct Sound Reverb (Default 0) mode= SOUND_MODE_REVERB_SET + (Reverb value 0-127); -Direct Sound Simultaneously-produced (Default 8) mode= (maximum simult. sounds 1-12) << SOUND_MODE_MAXCHN_SHIFT; -Direct Sound Master Volume (Default 15) mode= (Volume 1-15) << SOUND_MODE_MASVOL_SHIFT; -Direct Sound Playback Frequency (Default 13379Hz) mode= SOUND_MODE_FREQ_?????; (value defined in AgbSound.h (12 types)) -Final number of D/A converter bits (Default 8 bits) mode=SOUND_MODE_DA_BIT_?; 9-6 - You can set the preceding values at once using OR.

SoundDriverVSync

Type

void SoundDriverVSync(void)

Function

An extremely short system call that resets the sound DMA. The timing is extremely critical, so call this function immediately after the V-Blank interrupt every 1/60 second.

SoundDriverVSyncOff

Type

void SoundDriverVSyncOff(void)

Function

Due to problems with the main program if the V-Blank interrupts are stopped, and SoundDriverVSync() cannot be called every 1/60 a second, this function must be used to stop sound DMA.

Otherwise, even if you exceed the limit of the buffer the DMA will not stop and noise will result.

SoundDriverVSyncOn

Type

void SoundDriverVSyncOn(void)

Function

This function restarts the sound DMA stopped with the previously described SoundDriverVSyncOff().

After calling this function, have a V-Blank occur within 2/60 of a second and call SoundDriverVSync().

Sqrt

Type

u16 Sqrt(u32 X)

Function

Calculates the square root.

To increase the accuracy, left shift the argument X by a multiple of 2 only and pass the value. Also shift the return value and match the digits.

Stop

Type

Stop()

Function

Stops the system clock.

If the corresponding interrupt is permitted(set to IE), it returns based on the interrupt request conditions generating from the key, cartridge, or SIO.

The system clock is stopped so the IF flag is not set.

Always execute after setting the LCDC to OFF.

VBlankIntrWait

Macro

VblankIntrWait()

Function

Continues to wait in Halt status until V-Blank interrupt occurs.

Set the flag corresponding to INTR_CHECK_BUF(0x3007ff8) with interrupt processing.

When using multiple interrupts at the same time, the overhead for calling system calls can be decreased when compared to repeatedly calling Halt().

Equivalent to IntrWait(1, V_BLANK_INTR_FLAG).

Index