

[Main Page](#)

[Modules](#)

[Data Structures](#)

[Files](#)

[Related Pages](#)

---

# Tonclib 1.4 (20080825)

Tonclib is the library accompanying the set of GBA tutorials known as [Tonc](#). Initially, it was just a handful of macros and functions for dealing with the GBA hardware: the memory map and its bits, affine transformation code and things like that. More recently, more general items have been added like [tonccpy\(\)](#) and [toncset\(\)](#), the TSurface system and TTE. All these items should provide a firm basis on which to build GBA software.

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

1

# libtonc Modules

Here is a list of all modules:

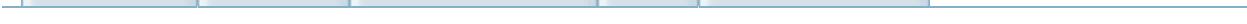
- **Bios Calls**
  - **BIOS informalities**
  - **BIOS functions**
  - **More BIOS functions**
- **Core**
  - **Bit(field) macros**
  - **Data routines**
  - **Miscellaneous routines**
  - **no\$gba debugging**
- **DMA**
- **Input**
- **Interrupt**
- **Math**
  - **Base math**
  - **Fixed point math**
  - **Look-up tables**
  - **Point functions**
  - **Vector functions**
  - **Rect functions**
- **Memory Map**
  - **Memory map bit(fields)**
    - **Display Control Flags**
    - **Display Status Flags**
    - **Background Control Flags**
    - **Graphic effects**
    - **Blend Flags**
    - **Tone Generator, Sweep Flags**
    - **Tone Generator, Square Flags**

- Tone Generator, Frequency Flags
- Tone Generator, Control Flags
- Direct Sound Flags
- Sound Status Flags
- DMA Control Flags
- Timer Control Flags
- Serial I/O Control
- Comm control.
- Key Flags
- Key Control Flags
- Interrupt Flags
- Waitstate Control Flags
- Screen-entry Flags
- Object Attribute 0 Flags
- Object Attribute 1 Flags
- Object Attribute 2 Flags
- Memory mapped arrays
- IO Registers
- IO Alternates
- Sound
- Tonic Text Engine
  - Operations
  - Attributes
  - Console IO
  - Tilemap text
  - Character text, column-major
  - Character text, row-major
  - Bitmap text
  - Object text
- Old Text
  - Tilemap text
  - Bitmap text
  - Object text

- **Timer**
- **Video**
  - **Surface functions**
    - **16bpp bitmap surfaces**
    - **8bpp bitmap surfaces**
    - **4bpp tiled surfaces, column major**
    - **4bpp tiled surfaces, row major**
  - **Colors**
  - **Tiled Backgrounds**
  - **Bitmaps**
  - **Objects**
  - **Affine functions**
- **Types and attributes**
  - **Type attributes**
  - **Primary types**
  - **Secondary types**
  - **Tertiary types**

---

Generated on Mon Aug 25 17:03:57 2008 for libtonc by  doxygen 1.5.3

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# Bios Calls

Interfaces and constants for the GBA BIOS routines. [More...](#)

# Modules

[BIOS informalities](#)

[BIOS functions](#)

[More BIOS functions](#)

---

## Detailed Description

Interfaces and constants for the GBA BIOS routines.

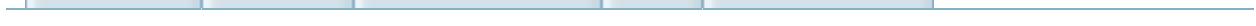
For details, see [tonc:keys](#) and especially [gbatek:bios](#).

### Note:

While the speeds of the routines are fair, there is a large overhead in calling the functions.

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **BIOS informalities**

## **[Bios Calls]**

# Data Structures

struct	<b>BUP</b> <i>BitUpPack ( for swi 10h).</i> <a href="#">More...</a>
struct	<b>MultiBootParam</b> <i>Multiboot struct.</i> <a href="#">More...</a>

## SoftReset flags

#define	<b>ROM_RESTART</b> 0x00
<i>Restart from ROM entry point.</i>	
#define	<b>RAM_RESTART</b> 0x01
<i>Restart from RAM entry point.</i>	

## RegisterRamReset flags

#define	<b>RESET_EWRAM</b> 0x0001
	<i>Clear 256K on-board WRAM.</i>
#define	<b>RESET_IWRAM</b> 0x0002
	<i>Clear 32K in-chip WRAM.</i>
#define	<b>RESET_PALETTE</b> 0x0004
	<i>Clear Palette.</i>
#define	<b>RESET_VRAM</b> 0x0008
	<i>Clear VRAM.</i>
#define	<b>RESET_OAM</b> 0x0010
	<i>Clear OAM. does NOT disable OBJS!</i>
#define	<b>RESET_REG_SIO</b> 0x0020
	<i>Switches to general purpose mode.</i>
#define	<b>RESET_REG_SOUND</b> 0x0040
	<i>Reset Sound registers.</i>
#define	<b>RESET_REG</b> 0x0080
	<i>All other registers.</i>
#define	<b>RESET_MEM_MASK</b> 0x001F
	<i>Clear 256K on-board WRAM.</i>
#define	<b>RESET_REG_MASK</b> 0x00E0
	<i>Clear 256K on-board WRAM.</i>
#define	<b>RESET_GFX</b> 0x001C
	<i>Clear all gfx-related memory.</i>

## Cpu(Fast)Set flags

#define	<b>CS_CPY</b> 0
	<i>Copy mode.</i>
#define	<b>CS_FILL</b> (1<<24)
	<i>Fill mode.</i>
#define	<b>CS_CPY16</b> 0
	<i>Copy in halfwords.</i>
#define	<b>CS_CPY32</b> (1<<26)
	<i>Copy words.</i>
#define	<b>CS_FILL32</b> (5<<24)
	<i>Fill words.</i>
#define	<b>CFS_CPY</b> CS_CPY
	<i>Copy words.</i>
#define	<b>CFS_FILL</b> CS_FILL
	<i>Fill words.</i>

## ObjAffineSet P-element offsets

#define	<b>BG_AFF_OFS</b> 2 <i>BgAffineDest</i> offsets.
#define	<b>OBJ_AFF_OFS</b> 8 <i>ObjAffineDest</i> offsets.

## Decompression routines

```
#define BUP_ALL_OFS (1<<31)
#define LZ_TYPE 0x00000010
#define LZ_SIZE_MASK 0xFFFFFFF00
#define LZ_SIZE_SHIFT 8
#define HUF_BPP_MASK 0x0000000F
#define HUF_TYPE 0x00000020
#define HUF_SIZE_MASK 0xFFFFFFF00
#define HUF_SIZE_SHIFT 8
#define RL_TYPE 0x00000030
#define RL_SIZE_MASK 0xFFFFFFF00
#define RL_SIZE_SHIFT 8
#define DIF_8 0x00000001
#define DIF_16 0x00000002
#define DIF_TYPE 0x00000080
#define DIF_SIZE_MASK 0xFFFFFFF00
#define DIF_SIZE_SHIFT 8
```

## Multiboot modes

```
#define MBOOT_NORMAL 0x00
#define MBOOT_MULTI 0x01
#define MBOOT_FAST 0x02
```

## Defines

```
#define swi_call(x) asm volatile("swi\t#x"<<16\" ::: "r0", "r1", "r2", "r3")  
BIOS calls from C.
```

---

# Define Documentation

```
#define swi_call ( x )    asm volatile("swi\t"#x"<<16" ::: "r0", "r1", "
```

BIOS calls from C.

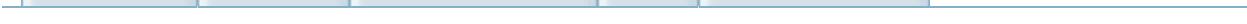
You can use this macro in a C BIOS-call wrapper. The wrapper should declare the flags, then this call will do the rest.

## Parameters:

x Number of swi call (THUMB number)

## Note:

It checks the \_\_thumb\_\_ #define to see whether we're in ARM or THUMB mode and fixes the swi number accordingly. Huzzah for the C proprocessor!

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **BIOS functions**

## **[Bios Calls]**

## Reset functions

void **SoftReset** (void)

void **RegisterRamReset** (u32 flags)

## Halt functions

void	<b>Halt</b> (void)
void	<b>Stop</b> (void)
void	<b>IntrWait</b> (u32 flagClear, u32 irq)
void	<b>VBlankIntrWait</b> (void) <i>Wait for the next VBlank (swi 05h).</i>

## Math functions

s32	<b>Div</b> (s32 num, s32 den) <i>Basic integer division (swi 06h).</i>
s32	<b>DivArm</b> (s32 den, s32 num) <i>Basic integer division, but with switched arguments (swi 07h).</i>
u32	<b>Sqrt</b> (u32 num) <i>Integer Square root (swi 08h).</i>
s16	<b>ArcTan</b> (s16 dydx) <i>Arctangent of dydx (swi 08h).</i>
s16	<b>ArcTan2</b> (s16 x, s16 y) <i>Arctangent of a coordinate pair (swi 09h).</i>

## Memory copiers/fillers

void	<b>CpuSet</b> (const void *src, void *dst, u32 mode) <i>Transfer via CPU in (half)word chunks.</i>
void	<b>CpuFastSet</b> (const void *src, void *dst, u32 mode) <i>A fast transfer via CPU in 32 byte chunks.</i>

## Rot/scale functions

void	<b>ObjAffineSet</b> (const <b>ObjAffineSource</b> *src, void *dst, s32 num, s32 offset)
	<i>Sets up a simple scale-then-rotate affine transformation (swi 0Eh).</i>
void	<b>BgAffineSet</b> (const <b>BgAffineSource</b> *src, <b>BgAffineDest</b> *dst, s32 num)
	<i>Sets up a simple scale-then-rotate affine transformation (swi 0Eh).</i>

## Decompression (see GBATek for format details)

void	<b>BitUnPack</b> (const void *src, void *dst, const <b>BUP</b> *bup)
void	<b>LZ77UnCompWram</b> (const void *src, void *dst)
void	<b>LZ77UnCompVram</b> (const void *src, void *dst)
void	<b>HuffUnComp</b> (const void *src, void *dst)
void	<b>RLUnCompWram</b> (const void *src, void *dst)
void	<b>RLUnCompVram</b> (const void *src, void *dst)
void	<b>Diff8bitUnFilterWram</b> (const void *src, void *dst)
void	<b>Diff8bitUnFilterVram</b> (const void *src, void *dst)
void	<b>Diff16bitUnFilter</b> (const void *src, void *dst)

## Sound Functions

void	<b>SoundBias</b> (u32 bias)
void	<b>SoundDriverInit</b> (void *src)
void	<b>SoundDriverMode</b> (u32 mode)
void	<b>SoundDriverMain</b> (void)
void	<b>SoundDriverVSync</b> (void)
void	<b>SoundChannelClear</b> (void)
u32	<b>MidiKey2Freq</b> (void *wa, u8 mk, u8 fp)
void	<b>SoundDriverVSyncOff</b> (void)
void	<b>SoundDriverVSyncOn</b> (void)

## Multiboot handshake

```
int MultiBoot (MultiBootParam *mb, u32 mode)
```

# Functions

u32 **BiosCheckSum** (void)

---

# Function Documentation

## **s16 ArcTan ( s16 *dydx* )**

Arctangent of *dydx* (swi 08h).

### **Parameters:**

*dydx* Slope to get the arctangent of.

### **Returns:**

Arctangent of *dydx* in the range <-4000h, 4000h>, corresponding to < $-\frac{1}{2}\pi$ ,  $\frac{1}{2}\pi$ > .

### **Note:**

Said to be inaccurate near the range's limits.

## **s16 ArcTan2 ( s16 *x*,                   s16 *y*                   )**

Arctangent of a coordinate pair (swi 09h).

This is the full-circle arctan, with an angle range of [0,FFFFh].

## **void BgAffineSet ( const BgAffineSource \* *src*,                       BgAffineDest \*      *dst*,                       s32                   *num*                       )**

Sets up a simple scale-then-rotate affine transformation (swi

0Eh).

Uses a single **ObjAffineSource** struct to set up an array of affine matrices (either BG or Object) with a certain transformation. The matrix created is

$$s_x \cdot \cos(\alpha) \ -s_x \cdot \sin(\alpha)$$

$$s_y \cdot \sin(\alpha) \ s_y \cdot \cos(\alpha)$$

### Parameters:

*src* Array with scale and angle information.

*dst* Array of affine matrices, starting at a *pa* element.

*num* Number of matrices to set.

*offset* Offset between affine elements. Use 2 for BG and 8 for object matrices.

### Note:

Each element in *src* needs to be word aligned, which devkitPro doesn't do anymore by itself.

```
void CpuFastSet( const void * src,  
                  void *      dst,  
                  u32          mode  
                )
```

A fast transfer via CPU in 32 byte chunks.

This uses ARM's ldmia/stmia instructions to copy 8 words at a time, making it rival DMA transfers in speed. With bit 26 set it will keep the source address constant, effectively performing fills instead of copies.

### Parameters:

```
src    Source address.  
dst    Destination address.  
mode   Number of words to transfer, and mode bits.
```

**Note:**

Both source and destination must be word aligned; the number of copies must be a multiple of 8.

In fill-mode (bit 26), the source is *still* an address, not a value.

memcpy32/16 and memset32/16 basically do the same things, but safer. Use those instead.

```
void CpuSet ( const void * src,  
              void *          dst,  
              u32            mode  
            )
```

Transfer via CPU in (half)word chunks.

The default mode is 16bit copies. With bit 24 set, it copies words; with bit 26 set it will keep the source address constant, effectively performing fills instead of copies.

**Parameters:**

```
src    Source address.  
dst    Destination address.  
mode   Number of transfers, and mode bits.
```

**Note:**

This basically does a straightforward loop-copy, and is not

particularly fast.

In fill-mode (bit 26), the source is *still* an address, not a value.

```
s32 Div( s32 num,  
          s32 den  
      )
```

Basic integer division (swi 06h).

**Parameters:**

*num* Numerator.

*den* Denominator.

**Returns:**

*num / den*

**Note:**

div/0 results in an infinite loop. Try `DivSafe` instead

```
s32 DivArm( s32 den,  
            s32 num  
        )
```

Basic integer division, but with switched arguments (swi 07h).

**Parameters:**

*num* Numerator.

*den* Denominator.

**Returns:**

*num / den*

**Note:**

div/0 results in an infinite loop.

```
void ObjAffineSet ( const ObjAffineSource * src,  
                     void *                 dst,  
                     s32                  num,  
                     s32                  offset  
)
```

Sets up a simple scale-then-rotate affine transformation (swi 0Eh).

Uses a single **ObjAffineSource** struct to set up an array of affine matrices (either BG or Object) with a certain transformation. The matrix created is

$$s_x \cdot \cos(\alpha) - s_x \cdot \sin(\alpha)$$

$$s_y \cdot \sin(\alpha) \quad s_y \cdot \cos(\alpha)$$

**Parameters:**

*src* Array with scale and angle information.

*dst* Array of affine matrices, starting at a *pa* element.

*num* Number of matrices to set.

*offset* Offset between affine elements. Use 2 for BG and 8 for object matrices.

**Note:**

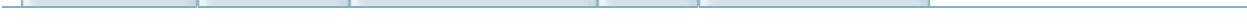
Each element in *src* needs to be word aligned, which devkitPro doesn't do anymore by itself.

```
void VBlankIntrWait( void )
```

Wait for the next VBlank (swi 05h).

**Note:**

Requires clearing of REG\_IFBIOS bit 0 at the interrupt  
tonc's master interrupt handler does this for you.

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# More BIOS functions

## [Bios Calls]

## Defines

```
#define DivMod Mod
```

## Functions

void	<b>VBlankIntrDelay</b> (u32 count) <i>Wait for count frames.</i>
int	<b>DivSafe</b> (int num, int den) <i>Div/0-safe division.</i>
int	<b>Mod</b> (int num, int den) <i>Modulo: num % den.</i>
u32	<b>DivAbs</b> (int num, int den) <i>Absolute value of num / den.</i>
int	<b>DivArmMod</b> (int den, int num) <i>Modulo: num % den.</i>
u32	<b>DivArmAbs</b> (int den, int num) <i>Absolute value of num / den.</i>
void	<b>CpuFastFill</b> (u32 wd, void *dst, u32 mode) <i>A fast word fill.</i>

---

# Function Documentation

```
void CpuFastFill ( u32    wd,
                    void * dst,
                    u32    mode
                )
```

A fast word fill.

While you can perform fills with [CpuFastSet\(\)](#), the fact that swi 12 requires a source address makes it awkward to use. This function is more like the traditional memset formulation.

## Parameters:

*wd* Fill word.  
*dst* Destination address.  
*mode* Number of words to transfer

```
int DivSafe ( int num,
              int den
            )
```

Div/0-safe division.

The standard Div hangs if *den* = 0. This version will return INT\_MAX/MIN in that case, depending on the sign of *num*, or just *num / den* if *den* is not 0.

## Parameters:

*num* Numerator.  
*den* Denominator.

---

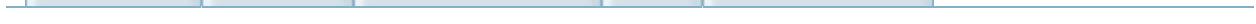
**void VBlankIntrDelay ( u32 *count* )**

---

Wait for *count* frames.

---

Generated on Mon Aug 25 17:03:57 2008 for libtonc by  doxygen 1.5.3

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **Core**

# Modules

- [\*\*Bit\(field\) macros\*\*](#)
  - [\*\*Data routines\*\*](#)
  - [\*\*Miscellaneous routines\*\*](#)
  - [\*\*no\\$gba debugging\*\*](#)
-

## Detailed Description

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  **doxygen** 1.5.3

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **Bit(field) macros**

## [Core]

## Simple bit macros

#define	<b>BIT</b> (n) ( 1<<(n) )	
		<i>Create value with bit n set.</i>
#define	<b>BIT_SHIFT</b> (a, n) ( (a)<<(n) )	
		<i>Shift a by n.</i>
#define	<b>BIT_MASK</b> (len) ( BIT(len)-1 )	
		<i>Create a bitmask len bits long.</i>
#define	<b>BIT_SET</b> (y, flag) ( y  = (flag) )	
		<i>Set the flag bits in word.</i>
#define	<b>BIT_CLEAR</b> (y, flag) ( y &= ~(flag) )	
		<i>Clear the flag bits in word.</i>
#define	<b>BIT_FLIP</b> (y, flag) ( y ^= (flag) )	
		<i>Flip the flag bits in word.</i>
#define	<b>BIT_EQ</b> (y, flag) ( ((y)&(flag)) == (flag) )	
		<i>Test whether all the flag bits in word are set.</i>
#define	<b>BF_MASK</b> (shift, len) ( BIT_MASK(len)<<(shift) )	
		<i>Create a bitmask of length len starting at bit shift.</i>
#define	<b>_BF_GET</b> (y, shift, len) ( ((y)>>(len))&(shift) )	
		<i>Retrieve a bitfield mask of length starting at bit shift from y.</i>
#define	<b>_BF_PREP</b> (x, shift, len) ( ((x)&BIT_MASK(len))<<(shift) )	
		<i>Prepare a bitmask for insertion or combining.</i>
#define	<b>_BF_SET</b> (y, x, shift, len) ( y= ((y) &~ BF_MASK(shift, len))   _BF_PREP(x, shift, len) )	
		<i>Insert a new bitfield value x into y.</i>

## some EVIL bit-field operations, >:)

These allow you to mimic bitfields with macros. Most of the bitfields in the registers have *foo\_SHIFT* and *foo\_MASK* macros indicating the mask and shift values of the bitfield named *foo* in a variable. These macros let you prepare, get and set the bitfields.

#define	<b>BFN_PREP</b> (x, name) ( ((x)<<name##_SHIFT) & name##_MASK )	<i>Prepare a named bit-field for for insterion or combination.</i>
#define	<b>BFN_GET</b> (y, name) ( ((y) & name##_MASK)>>name##_SHIFT )	<i>Get the value of a named bitfield from y. Equivalent to (var=) y.name.</i>
#define	<b>BFN_SET</b> (y, x, name) ( y = ((y)&~name##_MASK)   BFN_PREP(x,name) )	<i>Set a named bitfield in y to x. Equivalent to y.name= x.</i>
#define	<b>BFN_CMP</b> (y, x, name) ( ((y)&name##_MASK) == (x) )	<i>Compare a named bitfield to named literal x.</i>
#define	<b>BFN_PREP2</b> (x, name) ( (x) & name##_MASK )	<i>Massage x for use in bitfield name with pre-shifted x.</i>
#define	<b>BFN_GET2</b> (y, name) ( (y) & name##_MASK )	<i>Get the value of bitfield name from y, but don't down-shift.</i>
#define	<b>BFN_SET2</b> (y, x, name) ( y = ((y)&~name##_MASK)   BFN_PREP2(x,name) )	<i>Set bitfield name from y to x with pre-shifted x.</i>

## Functions

INLINE u32	<b>bf_get</b> (u32 y, uint shift, uint len) <i>Get len long bitfield from y, starting at shift.</i>
INLINE u32	<b>bf_merge</b> (u32 y, u32 x, uint shift, uint len) <i>Merge x into an len long bitfield from y, starting at shift.</i>
INLINE u32	<b>bf_clamp</b> (int x, uint len) <i>Clamp to within the range allowed by len bits.</i>
INLINE int	<b>bit_tribool</b> (u32 flags, uint plus, uint minus) <i>Gives a tribool (-1, 0, or +1) depending on the state of some bits.</i>
INLINE u32	<b>ROR</b> (u32 x, uint ror) <i>Rotate bits right. Yes, this does lead to a ror instruction.</i>

# Function Documentation

```
INLINE u32 bf_get( u32 y,
                    uint shift,
                    uint len
)
```

Get *len* long bitfield from *y*, starting at *shift*.

## Parameters:

*y* Value containing bitfield.  
*shift* Bitfield Start;  
*len* Length of bitfield.

## Returns:

Bitfield between bits *shift* and *shift + length*.

```
INLINE u32 bf_merge( u32 y,
                      u32 x,
                      uint shift,
                      uint len
)
```

Merge *x* into an *len* long bitfield from *y*, starting at *shift*.

## Parameters:

*y* Value containing bitfield.  
*x* Value to merge (will be masked to fit).  
*shift* Bitfield Start;  
*len* Length of bitfield.

**Returns:**

Result of merger:  $(y \& \sim M) | (x << s \& M)$

**Note:**

Does *not* write the result back into  $y$  (Because pure C doesn't have references, that's why)

```
INLINE int bit_tribool ( u32 flags,
                         uint plus,
                         uint minus
                       )
```

Gives a tribool (-1, 0, or +1) depending on the state of some bits.

Looks at the *plus* and *minus* bits of *flags*, and subtracts their status to give a +1, -1 or 0 result. Useful for direction flags.

**Parameters:**

*flags* Value with bit-flags.

*plus* Bit number for positive result.

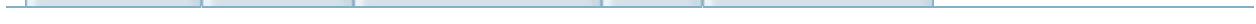
*minus* Bit number for negative result.

**Returns:**

**+1** if *plus* bit is set but *minus* bit isn't

**-1** if *minus* bit is set and *plus* bit isn't

**0** if neither or both are set.

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# Data routines

## [Core]

## Copying and filling routines

void *	<b>tonccpy</b> (void *dst, const void *src, uint size) VRAM-safe <i>cpy</i> .
void *	<b>_toncset</b> (void *dst, u32 fill, uint size) VRAM-safe <i>memset</i> , <i>internal routine</i> .
INLINE void *	<b>toncset</b> (void *dst, u8 src, uint count) VRAM-safe <i>memset</i> , <i>byte version</i> . <i>Size in bytes</i> .
INLINE void *	<b>toncset16</b> (void *dst, u16 src, uint count) VRAM-safe <i>memset</i> , <i>halfword version</i> . <i>Size in hwords</i> .
INLINE void *	<b>toncset32</b> (void *dst, u32 src, uint count) VRAM-safe <i>memset</i> , <i>word version</i> . <i>Size in words</i> .
void	<b>memset16</b> (void *dst, u16 hw, uint hwcount) <i>Fastfill for halfwords, analogous to memset()</i> .
IWRAM_CODE void	<b>memset32</b> (void *dst, u32 wd, uint wdcount) <i>Fast-fill by words, analogous to memset()</i> .
IWRAM_CODE void	<b>memcpy32</b> (void *dst, const void *src, uint wdcount)
#define	<b>GRIT_CPY</b> (dst, name) memcpy16(dst, name, name##Len/2) <i>Simplified copier for GRIT-exported data.</i>

## Repeated-value creators

These functions take a hex-value and duplicate it to all fields, like 0x88 -> 0x88888888.

INLINE u16	<b>dup8</b> (u8 x) <i>Duplicate a byte to form a halfword: 0x12 -&gt; 0x1212.</i>
INLINE u32	<b>dup16</b> (u16 x) <i>Duplicate a halfword to form a word: 0x1234 -&gt; 0x12341234.</i>
INLINE u32	<b>quad8</b> (u8 x) <i>Quadruple a byte to form a word: 0x12 -&gt; 0x12121212.</i>
INLINE u32	<b>octup</b> (u8 x) <i>Octuple a nybble to form a word: 0x1 -&gt; 0x11111111.</i>

## Packing routines.

INLINE u16	<b>bytes2hword</b> (u8 b0, u8 b1) <i>Pack 2 bytes into a word. Little-endian order.</i>
INLINE u32	<b>bytes2word</b> (u8 b0, u8 b1, u8 b2, u8 b3) <i>Pack 4 bytes into a word. Little-endian order.</i>
INLINE u32	<b>hword2word</b> (u16 h0, u16 h1) <i>Pack 2 bytes into a word. Little-endian order.</i>

## Defines

```
#define countof(_array) ( sizeof(_array)/sizeof(_array[0]) )
```

*Get the number of elements in an array.*

# Functions

INLINE uint **align** (uint x, uint width)

*Align x to the next multiple of width.*

---

# Function Documentation

```
void* __toncset( void * dst,
                  u32    fill,
                  uint   size
                )
```

VRAM-safe memset, internal routine.

This version mimics memset in functionality, with the benefit of working for VRAM as well. It is also slightly faster than the original memset.

## Parameters:

*dst* Destination pointer.  
*fill* Word to fill with.  
*size* Fill-length in bytes.

## Returns:

*dst*.

## Note:

The *dst* pointer and *size* need not be word-aligned. In the case of unaligned fills, *fill* will be masked off to match the situation.

```
void memcpy16( void *      dst,
               const void * src,
               uint        hwcount
             )
```

Copy for halfwords.

Uses [memcpy32\(\)](#) if  $hwn > 6$  and  $src$  and  $dst$  are aligned equally.

#### Parameters:

$dst$       Destination address.  
 $src$       Source address.  
 $hwcount$  Number of halfwords to fill.

#### Note:

$dst$  and  $src$  **must** be halfword aligned.

$r0$  and  $r1$  return as  $dst + hwcount * 2$  and  $src + hwcount * 2$ .

```
IWRAM_CODE void memcpy32 ( void *      dst,
                           const void * src,
                           uint        wdcount
                         )
```

Fast-copy by words.

Like [CpuFastFill\(\)](#), only without the requirement of 32byte chunks

#### Parameters:

$dst$       Destination address.  
 $src$       Source address.  
 $wdcount$  Number of words.

#### Note:

$src$  and  $dst$  **must** be word aligned.

$r0$  and  $r1$  return as  $dst + wdcount * 4$  and  $src + wdcount * 4$ .

```
void memset16 ( void * dst,  
                u16    hw,  
                uint   hwcount  
            )
```

Fastfill for halfwords, analogous to `memset()`.

Uses [memset32\(\)](#) if *hwcount*>5

### Parameters:

*dst* Destination address.  
*hw* Source halfword (not address).  
*hwcount* Number of halfwords to fill.

### Note:

*dst* must be halfword aligned.

*r0* returns as *dst* + *hwcount*\*2.

```
IWRAM_CODE void memset32 ( void * dst,  
                           u32    wd,  
                           uint   wdcount  
                     )
```

Fast-fill by words, analogous to `memset()`.

Like [CpuFastSet\(\)](#), only without the requirement of 32byte chunks and no awkward store-value-in-memory-first issue.

### Parameters:

*dst* Destination address.

*wd* Fill word (not address).  
*wdcount* Number of words to fill.

**Note:**

*dst* must be word aligned.

*r0* returns as *dst* + *wdcount*\*4.

```
void* tonccpy ( void *      dst,  
                 const void * src,  
                 uint          size  
               )
```

VRAM-safe cpy.

This version mimics memcpy in functionality, with the benefit of working for VRAM as well. It is also slightly faster than the original memcpy, but faster implementations can be made.

**Parameters:**

*dst* Destination pointer.  
*src* Source pointer.  
*size* Fill-length in bytes.

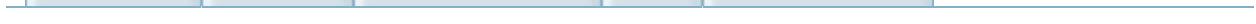
**Returns:**

*dst*.

**Note:**

The pointers and size need not be word-aligned.

*Generated on Mon Aug 25 17:03:57 2008 for libtongc by doxygen 1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# Miscellaneous routines

## [Core]

## Sector checking

u32	<b>octant</b> (int x, int y) <i>Get the octant that (x, y) is in.</i>
u32	<b>octant_rot</b> (int x0, int y0) <i>Get the rotated octant that (x, y) is in.</i>

## Random numbers

int	<b>sqran</b> (int seed)
INLINE int	<b>qran</b> (void) <i>Quick (and very dirty) pseudo-random number generator.</i>
INLINE int	<b>qran_range</b> (int min, int max) <i>Ranged random number.</i>
#define	<b>QRAN_SHIFT</b> 15
#define	<b>QRAN_MASK</b> ((1<<QRAN_SHIFT)-1)
#define	<b>QRAN_MAX</b> QRAN_MASK

## Inline assembly

#define	<b>ASM_CMT</b> (str) asm volatile("@# " str)
	<i>Assembly comment.</i>
#define	<b>ASM_BREAK</b> () asm volatile("\tmov\t\tr11, r11")
	<i>No\$gba breakpoint.</i>
#define	<b>ASM_NOP</b> () asm volatile("\tnop")
	<i>No-op; wait a bit.</i>

## Defines

#define	<b>STR(x)</b>	#x
#define	<b>XSTR(x)</b>	STR(x)
<i>Create text string from a literal.</i>		

---

## Define Documentation

```
#define STR( x ) #x
```

---

## Function Documentation

```
u32 octant( int x,  
             int y  
           )
```

Get the octant that  $(x, y)$  is in.

This function divides the circle in 8 parts. The angle starts at the  $y=0$  line and then moves in the direction of the  $x=0$  line. On the screen, this would be like starting at the 3 o'clock position and moving clockwise

```
u32 octant_rot( int x0,  
                  int y0  
                )
```

Get the rotated octant that  $(x, y)$  is in.

Like [octant\(\)](#) but with a twist. The 0-octant starts 22.5° earlier so that 3 o'clock falls in the middle of octant 0, instead of at its start. This can be useful for 8 directional pointing.

```
INLINE int qran( void )
```

Quick (and very dirty) pseudo-random number generator.

**Returns:**

random in range [0,8000h>

```
INLINE int qran_range ( int min,  
                      int max  
)
```

Ranged random number.

**Returns:**

random in range [*min*, *max*]

**Note:**

(max-min) must be lower than 8000h

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **no\$gba debugging**

## **[Core]**

## Functions

	int <b>nocash_puts</b> (const char *str) <i>Output a string to no\$gba debugger.</i>
EWRAM_CODE void	<b>nocash_message</b> (void) <i>Print the current nocash_buffer to the no\$gba debugger.</i>

## Variables

```
EWRAM_DATA char nocash_buffer [80]
```

---

## **Detailed Description**

The non-freeware versions of no\$gba have window to which you can output messages for debugging purposes. These functions allow you to work with that.

---

# Function Documentation

```
int nocash_puts ( const char * str )
```

Output a string to no\$gba debugger.

**Parameters:**

*str* Text to print.

**Returns:**

Number of characters printed.

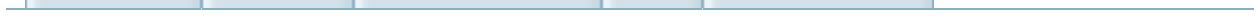
---

# Variable Documentation

**EWRAM\_DATA char nocash\_buffer[80]**

---

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by  1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# DMA

## Defines

```
#define DMA_TRANSFER(_dst, _src, count, ch, mode)  
    General purpose DMA transfer macro.
```

## Functions

INLINE void	<b>dma_cpy</b> (void *dst, const void *src, uint count, uint ch, u32 mode) <i>Generic DMA copy routine.</i>
INLINE void	<b>dma_fill</b> (void *dst, volatile u32 src, uint count, uint ch, u32 mode) <i>Generic DMA fill routine.</i>
INLINE void	<b>dma3_cpy</b> (void *dst, const void *src, uint size) <i>Specific DMA copier, using channel 3, word transfers.</i>
INLINE void	<b>dma3_fill</b> (void *dst, volatile u32 src, uint size) <i>Specific DMA filler, using channel 3, word transfers.</i>

## **Detailed Description**

---

# Define Documentation

```
#define DMA_TRANSFER( _dst,  
                     _src,  
                     count,  
                     ch,  
                     mode )
```

## Value:

```
do {  
    REG_DMA[ch].cnt= 0;  
    REG_DMA[ch].src= (const void*)(_src);  
    REG_DMA[ch].dst= (void*)(_dst);  
    REG_DMA[ch].cnt= (count) | (mode);  
} while(0)
```

General purpose DMA transfer macro.

## Parameters:

- \_dst* Destination address.
- \_src* Source address.
- count* Number of transfers.
- ch* DMA channel.
- mode* DMA mode.

---

# Function Documentation

```
INLINE void dma3_cpy ( void *      dst,
                      const void * src,
                      uint        size
)
```

Specific DMA copier, using channel 3, word transfers.

## Parameters:

*dst* Destination address.

*src* Source address.

*size* Number of bytes to copy

## Note:

*size* is the number of bytes

```
INLINE void dma3_fill ( void *      dst,
                       volatile u32 src,
                       uint        size
)
```

Specific DMA filler, using channel 3, word transfers.

## Parameters:

*dst* Destination address.

*src* Source value.

*size* Number of bytes to copy

## Note:

*size* is the number of bytes

```
INLINE void dma_cpy ( void *      dst,
                      const void * src,
                      uint        count,
                      uint        ch,
                      u32         mode
)
```

Generic DMA copy routine.

#### Parameters:

*dst* Destination address.  
*src* Source address.  
*count* Number of copies to perform.  
*ch* DMA channel.  
*mode* DMA transfer mode.

#### Note:

*count* is the number of copies, not the size in bytes.

```
INLINE void dma_fill ( void *      dst,
                      volatile u32 src,
                      uint        count,
                      uint        ch,
                      u32         mode
)
```

Generic DMA fill routine.

#### Parameters:

*dst* Destination address.

*src* Source value.  
*count* Number of copies to perform.  
*ch* DMA channel.  
*mode* DMA transfer mode.

**Note:**

*count* is the number of copies, not the size in bytes.

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **Input**

Routines for synchronous and asynchronous button states.

[More...](#)

## Basic synchronous keystates

void	<b>key_poll ()</b> <i>Poll for keystates and repeated keys.</i>
INLINE u32	<b>key_curr_state (void)</b> <i>Get current keystate.</i>
INLINE u32	<b>key_prev_state (void)</b> <i>Get previous key state.</i>
INLINE u32	<b>key_is_down (u32 key)</b> <i>Gives the keys of key that are currently down.</i>
INLINE u32	<b>key_is_up (u32 key)</b> <i>Gives the keys of key that are currently up.</i>
INLINE u32	<b>key_was_down (u32 key)</b> <i>Gives the keys of key that were previously down.</i>
INLINE u32	<b>key_was_up (u32 key)</b> <i>Gives the keys of key that were previously up.</i>

## Transitional keystates

INLINE u32	<b>key_transit</b> (u32 key) <i>Gives the keys of key that are different from before.</i>
INLINE u32	<b>key_held</b> (u32 key) <i>Gives the keys of key that are being held down.</i>
INLINE u32	<b>key_hit</b> (u32 key) <i>Gives the keys of key that are pressed (down now but not before).</i>
INLINE u32	<b>key_released</b> (u32 key) <i>Gives the keys of key that are being released.</i>

## Tribools

INLINE int	<b>key_tri_horz</b> (void) <i>Horizontal tribool (right,left)=(+,-).</i>
INLINE int	<b>key_tri_vert</b> (void) <i>Vertical tribool (down,up)=(+,-).</i>
INLINE int	<b>key_tri_shoulder</b> (void) <i>Shoulder-button tribool (R,L)=(+,-).</i>
INLINE int	<b>key_tri_fire</b> (void) <i>Fire-button tribool (A,B)=(+,-).</i>

## Key repeats

u32	<b>key_repeat</b> (u32 keys) <i>Get status of repeated keys.</i>
void	<b>key_repeat_mask</b> (u32 mask) <i>Set repeat mask. Only these keys will be considered for repeats.</i>
void	<b>key_repeat_limits</b> (uint delay, uint repeat) <i>Set the delay and repeat limits for repeated keys.</i>

## Defines

#define	<b>KEY_FULL</b> 0xFFFFFFFF
	<i>Define for checking all keys.</i>
#define	<b>KEY_DOWN_NOW(key)</b> (~(REG_KEYINPUT) & key)
#define	<b>KEY_UP_NOW(key)</b> ( (REG_KEYINPUT) & key)
#define	<b>KEY_EQ(key_fun, keys)</b> ( key_fun(keys) == (keys) )
#define	<b>KEY_TRIBOOL(fnKey, plus, minus)</b> ( bit_tribool(fnKey(KEY_FULL), plus, minus) )

## Enumerations

```
enum eKeyIndex {  
    KI_A = 0, KI_B, KI_SELECT, KI_START,  
    KI_RIGHT, KI_LEFT, KI_UP, KI_DOWN,  
    KI_R, KI_L, KI_MAX  
}
```

## Functions

void	<b>key_wait_for_clear</b> (u32 key)
------	-------------------------------------

void	<b>key_wait_till_hit</b> (u16 key)
------	------------------------------------

	<i>Wait until key is hit.</i>
--	-------------------------------

## Variables

u16	<b>__key_curr</b>
u16	<b>__key_prev</b>

---

## Detailed Description

Routines for synchronous and asynchronous button states.

For details, see [tonc:keys](#).

---

# Enumeration Type Documentation

**enum eKeyIndex**

---

# Function Documentation

```
void key_repeat_limits ( uint delay,  
                        uint repeat  
                      )
```

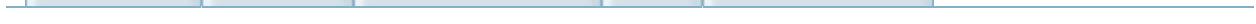
Set the delay and repeat limits for repeated keys.

## Parameters:

*delay* Set first repeat limit. If 0, repeats are off.  
*repeat* Sets later repeat limit.

## Note:

Both limits have a range of [0, 255]. If either argument is <0, the old value will be kept.

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **Interrupt**

Hardware interrupt management. [More...](#)

## Data Structures

struct	<b>IRQ_REC</b>
	<i>Struct for prioritized irq table.</i> <a href="#">More...</a>

## Options for `irq_set`

#define	<b>ISR_LAST</b> 0x0040
	<i>Last isr in line (Lowest priority).</i>
#define	<b>ISR_REPLACE</b> 0x0080
	<i>Replace old isr if existing (prio ignored).</i>
#define	<b>ISR_PRIO_MASK</b> 0x003F
	<i>Last isr in line (Lowest priority).</i>
#define	<b>ISR_PRIO_SHIFT</b> 0
	<i>Last isr in line (Lowest priority).</i>
#define	<b>ISR_PRIO(n)</b> ((n)<<ISR_PRIO_SHIFT)
	<i>Last isr in line (Lowest priority).</i>
#define	<b>ISR_DEF</b> (ISR_LAST ISR_REPLACE)
	<i>Last isr in line (Lowest priority).</i>

## Defines

#define	<b>IRQ_INIT()</b> irq_init(NULL)
	<i>Default <b>irq_init()</b> call: use <b>irq_master_nest()</b> for switchboard.</i>
#define	<b>IRQ_SET(irq_id)</b> irq_set(lI_##irq_id, NULL, ISR_DEF)
	<i>Default <b>irq_set()</b> call: no isr, add to back of priority stack.</i>
#define	<b>IRQ_ADD(irq_id)</b> irq_add(lI_##irq_id, NULL)

## Enumerations

enum	<pre>eIrqIndex {     II_VBLANK = 0, II_HBLANK, II_VCOUNT, II_TIMER0,     II_TIMER1, II_TIMER2, II_TIMER3, II_SERIAL,     II_DMA0, II_DMA1, II_DMA2, II_DMA3,     II_KEYPAD, II_GAMEPAK, II_MAX }</pre>
------	--

*IRQ indices, to be used in most functions.*

## Functions

IWRAM_CODE void	<b>isr_master</b> (void)
IWRAM_CODE void	<b>isr_master_nest</b> (void)
void	<b>irq_init</b> ( <b>fnptra</b> isr) <i>Initialize irq business.</i>
<b>fnptra</b>	<b>irq_set_master</b> ( <b>fnptra</b> isr) <i>Set a master ISR.</i>
<b>fnptra</b>	<b>irq_add</b> (enum <b>eIRQIndex</b> irq_id, <b>fnptra</b> isr) <i>Add a specific ISR.</i>
<b>fnptra</b>	<b>irq_delete</b> (enum <b>eIRQIndex</b> irq_id) <i>Remove an ISR.</i>
<b>fnptra</b>	<b>irq_set</b> (enum <b>eIRQIndex</b> irq_id, <b>fnptra</b> isr, u32 opts) <i>General IRQ manager.</i>
void	<b>irq_enable</b> (enum <b>eIRQIndex</b> irq_id)
void	<b>irq_disable</b> (enum <b>eIRQIndex</b> irq_id)

## Variables

**IRQ\_REC**    **\_\_isr\_table [II\_MAX+1]**

---

## **Detailed Description**

Hardware interrupt management.

For details, see [tonc:irq](#)

---

# Function Documentation

```
fnptr irq_add ( enum elrqIndex irq_id,  
                 fnptr           isr  
               )
```

Add a specific ISR.

Special case of `irq_set`. If the interrupt has an ISR already it'll be replaced; if not it will add it in the back.

**Parameters:**

*irq\_id* Index of irq.

*isr* Interrupt service routine for this irq; can be NULL

**Returns:**

Previous ISR

**Note:**

*irq\_id* is *NOT* a bit-mask, it is an index!

```
fnptr irq_delete ( enum elrqIndex irq_id )
```

Remove an ISR.

it'll be replaced; if not it will add it in the back.

**Parameters:**

*irq\_id* Index of irq.

**Returns:**

## Previous ISR

### Note:

*irq\_id* is NOT a bit-mask, it is an index!

```
void irq_init (fnptr isr )
```

Initialize irq business.

Clears ISR table and sets up a master isr.

### Parameters:

*isr* Master ISR. If NULL, *isr\_master\_nest* is used

```
fnptr irq_set ( enum elrqIndex irq_id,  
                 fnptr          isr,  
                 u32            opts  
               )
```

General IRQ manager.

This routine manages the ISRs of interrupts and their priorities.

### Parameters:

*irq\_id* Index of irq.

*isr* Interrupt service routine for this irq; can be NULL

*opts* ISR options

### Returns:

Previous specific ISR

**Note:**

*irq\_id* is *NOT* a bit-mask, it is an index!

## **fnptr irq\_set\_master ( fnptr *isr* )**

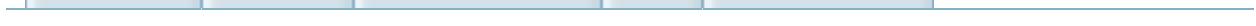
Set a master ISR.

**Parameters:**

*isr* Master ISR. If NULL, *isr\_master\_multi* is used

**Returns:**

Previous master ISR

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **Math**

# Modules

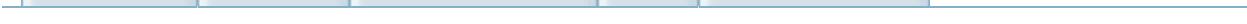
<b>Base math</b>
<i>Basic math macros and functions like MIN, MAX.</i>
<b>Fixed point math</b>
<b>Look-up tables</b>
<i>Tonc's internal look-up tables and related routines.</i>
<b>Point functions</b>
<b>Vector functions</b>
<b>Rect functions</b>

---

## Detailed Description

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  **doxygen** 1.5.3

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

## Base math

### [[Math](#)]

Basic math macros and functions like MIN, MAX. [More...](#)

## core math macros

INLINE int	<b>sgn</b> (int x) <i>Get the sign of x.</i>
INLINE int	<b>sgn3</b> (int x) <i>Tri-state sign of x: -1 for negative, 0 for 0, +1 for positive.</i>
INLINE int	<b>max</b> (int a, int b) <i>Get the maximum of a and b.</i>
INLINE int	<b>min</b> (int a, int b) <i>Get the minimum of a and b.</i>
#define	<b>ABS</b> (x) ( (x)>=0 ? (x) : -(x) ) <i>Get the absolute value of x.</i>
#define	<b>SGN</b> (x) ( (x)>=0 ? 1 : -1 ) <i>Get the sign of x.</i>
#define	<b>SGN2</b> SGN <i>Get the absolute value of x.</i>
#define	<b>SGN3</b> (x) ( (x)>0 ? 1 : ( (x)<0 ? -1 : 0 ) ) <i>Tri-state sign: -1 for negative, 0 for 0, +1 for positive.</i>
#define	<b>MAX</b> (a, b) ( ((a) > (b)) ? (a) : (b) ) <i>Get the maximum of a and b.</i>
#define	<b>MIN</b> (a, b) ( ((a) < (b)) ? (a) : (b) ) <i>Get the minimum of a and b.</i>
#define	<b>SWAP2</b> (a, b) do { a=(a)-(b); b=(a)+(b); a=(b)-(a); } while(0) <i>In-place swap.</i>
#define	<b>SWAP</b> SWAP2 <i>Get the absolute value of x.</i>
#define	<b>SWAP3</b> (a, b, tmp) do { (tmp)=(a); (a)=(b); (b)=(tmp); } while(0) <i>Swaps a and b, using tmp as a temporary.</i>

## Boundary response macros

INLINE BOOL	<b>in_range</b> (int x, int min, int max) <i>Range check.</i>
INLINE int	<b>clamp</b> (int x, int min, int max) <i>Truncates x to stay in range [min, max].</i>
INLINE int	<b>reflect</b> (int x, int min, int max) <i>Reflects x at boundaries min and max.</i>
INLINE int	<b>wrap</b> (int x, int min, int max) <i>Wraps x to stay in range [min, max].</i>
#define	<b>IN_RANGE</b> (x, min, max) ( ((x)>=(min)) && ((x)<(max)) ) <i>Range check.</i>
#define	<b>CLAMP</b> (x, min, max) ( (x)>=(max) ? ((max)-1) : ( ((x)<(min)) ? (min) : (x) ) ) <i>Truncates x to stay in range [min, max].</i>
#define	<b>REFLECT</b> (x, min, max) ( (x)>=(max) ? 2*((max)-1)-(x) : ( ((x)<(min)) ? 2*(min)-(x) : (x) ) ) <i>Reflects x at boundaries min and max.</i>
#define	<b>WRAP</b> (x, min, max) ( (x)>=(max) ? (x)+(min)-(max) : ( ((x)<(min)) ? (x)+(max)-(min) : (x) ) ) <i>Wraps x to stay in range [min, max].</i>

## **Detailed Description**

Basic math macros and functions like MIN, MAX.

---

# Define Documentation

```
#define CLAMP ( x,  
              min,  
              max )  ( (x)>=(max) ? ((max)-1) : ( ((x)<(min)) ? (r
```

Truncates  $x$  to stay in range  $[min, max]$ .

## Returns:

Truncated value of  $x$ .

## Note:

$max$  is exclusive!

```
#define REFLECT ( x,  
                 min,  
                 max )  ( (x)>=(max) ? 2*((max)-1)-(x) : ( ((x)<(m
```

Reflects  $x$  at boundaries  $min$  and  $max$ .

If  $x$  is outside the range  $[min, max]$ , it'll be placed inside again with the same distance to the 'wall', but on the other side. Example for lower border:  $y = min - (x - min) = 2*min + x$ .

## Returns:

Reflected value of  $x$ .

## Note:

$max$  is exclusive!

# Function Documentation

```
INLINE int clamp ( int x,  
                  int min,  
                  int max  
                )
```

Truncates *x* to stay in range  $[min, max]$ .

**Returns:**

Truncated value of *x*.

**Note:**

*max* is exclusive!

```
INLINE int reflect ( int x,  
                     int min,  
                     int max  
                   )
```

Reflects *x* at boundaries *min* and *max*.

If *x* is outside the range  $[min, max]$ , it'll be placed inside again with the same distance to the 'wall', but on the other side. Example for lower border:  $y = min - (x - min) = 2*min + x$ .

**Returns:**

Reflected value of *x*.

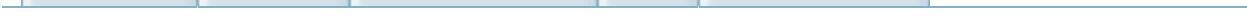
**Note:**

*max* is exclusive!

---

---

Generated on Mon Aug 25 17:03:57 2008 for libtonc by  doxygen 1.5.3

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# Fixed point math

## [Math]

## Defines

#define	<b>FIX_SHIFT</b> 8
#define	<b>FIX_SCALE</b> ( 1<<FIX_SHIFT )
#define	<b>FIX_MASK</b> ( FIX_SCALE-1 )
#define	<b>FIX_SCALEF</b> ( (float)FIX_SCALE )
#define	<b>FIX_SCALEF_INV</b> ( 1.0/FIX_SCALEF )
#define	<b>FIX_ONE</b> FIX_SCALE
#define	<b>FX_RECIPROCAL</b> (a, fp) ( ((1<<(fp))+(a)-1)/(a) ) <i>Get the fixed point reciprocal of a, in fp fractional bits.</i>
#define	<b>FX_RECIMUL</b> (x, a, fp) ( ((x)*((1<<(fp))+(a)-1)/(a))>>(fp) ) <i>Perform the division x/ a by reciprocal multiplication.</i>

# Functions

INLINE <b>FIXED</b>	<b>int2fx</b> (int d) <i>Convert an integer to fixed-point.</i>
INLINE <b>FIXED</b>	<b>float2fx</b> (float f) <i>Convert a float to fixed-point.</i>
INLINE u32	<b>fx2uint</b> ( <b>FIXED</b> fx) <i>Convert a FIXED point value to an unsigned integer (orly?).</i>
INLINE u32	<b>fx2ufrac</b> ( <b>FIXED</b> fx) <i>Get the unsigned fractional part of a fixed point value (orly?).</i>
INLINE int	<b>fx2int</b> ( <b>FIXED</b> fx) <i>Convert a FIXED point value to an signed integer.</i>
INLINE float	<b>fx2float</b> ( <b>FIXED</b> fx) <i>Convert a fixed point value to floating point.</i>
INLINE <b>FIXED</b>	<b>fxadd</b> ( <b>FIXED</b> fa, <b>FIXED</b> fb) <i>Add two fixed point values.</i>
INLINE <b>FIXED</b>	<b>fxsub</b> ( <b>FIXED</b> fa, <b>FIXED</b> fb) <i>Subtract two fixed point values.</i>
INLINE <b>FIXED</b>	<b>fxmul</b> ( <b>FIXED</b> fa, <b>FIXED</b> fb) <i>Multiply two fixed point values.</i>
INLINE <b>FIXED</b>	<b>fxdiv</b> ( <b>FIXED</b> fa, <b>FIXED</b> fb) <i>Divide two fixed point values.</i>
INLINE <b>FIXED</b>	<b>fxmul64</b> ( <b>FIXED</b> fa, <b>FIXED</b> fb) <i>Multiply two fixed point values using 64bit math.</i>
INLINE <b>FIXED</b>	<b>fxdiv64</b> ( <b>FIXED</b> fa, <b>FIXED</b> fb) <i>Divide two fixed point values using 64bit math.</i>

## **Detailed Description**

---

# Define Documentation

```
#define FIX_SHIFT 8
```

```
#define FX_RECIMUL ( x,  
                    a,  
                    fp )  ( ((x)*((1<<(fp))+(a)-1)/(a))>>(fp) )
```

Perform the division  $x/a$  by reciprocal multiplication.

Division is slow, but you can approximate division by a constant by multiplying with its reciprocal:  $x/a$  vs  $x*(1/a)$ . This routine gives the reciprocal of  $a$  as a fixed point number with  $fp$  fractional bits.

## Parameters:

- $a$  Value to take the reciprocal of.
- $fp$  Number of fixed point bits

## Note:

The routine does do a division, but the compiler will optimize it to a single constant ... *if* both  $a$  and  $fp$  are constants!

Rules for safe reciprocal division, using  $n = 2^{fp}$  and  $m = (n+a-1)/a$  (i.e., rounding up)

- Maximum safe numerator  $x$ :  $x < n/(m*a-n)$
- Minimum  $n$  for known  $x$ :  $n > x*(a-1)$

```
#define FX_RECIPROCAL ( a,  
                           fp )   ( ((1<<(fp))+(a)-1)/(a) )
```

Get the fixed point reciprocal of *a*, in *fp* fractional bits.

**Parameters:**

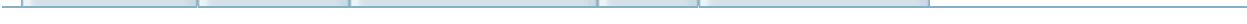
- a* Value to take the reciprocal of.
- fp* Number of fixed point bits

**Note:**

The routine does do a division, but the compiler will optimize it to a single constant ... *if* both *a* and *fp* are constants!

**See also:**

[\*\*FX\\_RECIMUL\*\*](#)

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# Look-up tables

## [[Math](#)]

Tonc's internal look-up tables and related routines. [More...](#)

## Defines

```
#define SIN_LUT_SIZE 514
#define DIV_LUT_SIZE 257
```

# Functions

INLINE s32	<b>lu_sin</b> (uint theta) <i>Look-up a sine value (<math>2\pi = 0x10000</math>).</i>
INLINE s32	<b>lu_cos</b> (uint theta) <i>Look-up a cosine value (<math>2\pi = 0x10000</math>).</i>
INLINE uint	<b>lu_div</b> (uint x) <i>Look-up a division value between 0 and 255.</i>
INLINE int	<b>lu_lerp32</b> (const s32 lut[], uint x, const uint shift) <i>Linear interpolator for 32bit LUTs.</i>
INLINE int	<b>lu_lerp16</b> (const s16 lut[], uint x, const uint shift) <i>As lu_lerp32, but for 16bit LUTs.</i>

## Variables

s32	<b>div_lut</b> [257]
s16	<b>sin_lut</b> [514]

---

## **Detailed Description**

Tonc's internal look-up tables and related routines.

---

## Define Documentation

```
#define SIN_LUT_SIZE 514
```

---

# Function Documentation

**INLINE s32 lu\_cos ( uint *theta* )**

Look-up a cosine value ( $2\pi = 0x10000$ ).

**Parameters:**

*theta* Angle in [0,FFFFh] range

**Returns:**

.12f cosine value

**INLINE uint lu\_div ( uint *x* )**

Look-up a division value between 0 and 255.

**Parameters:**

*x* reciprocal to look up.

**Returns:**

1/x (.16f)

**INLINE int lu\_lerp32 ( const s32 *lut*[],  
                  uint       *x*,  
                  const uint *shift*  
                )**

Linear interpolator for 32bit LUTs.

A lut is essentially the discrete form of a function,  $f(x)$ . You

can get values for non-integer  $x$  via (linear) interpolation between  $f(x)$  and  $f(x+1)$ .

### Parameters:

- lut* The LUT to interpolate from.
- x* Fixed point number to interpolate at.
- shift* Number of fixed-point bits of *x*.

## INLINE s32 lu\_sin ( uint *theta* )

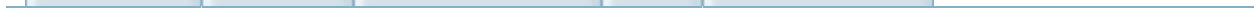
Look-up a sine value ( $2\pi = 0x10000$ ).

### Parameters:

- theta* Angle in [0,FFFFh] range

### Returns:

.12f sine value

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# Point functions

## [Math]

# Data Structures

struct	<b>POINT32</b>
	<i>2D Point struct</i> <a href="#">More...</a>

# Functions

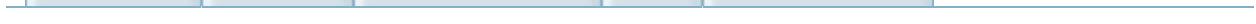
INLINE POINT *	<b>pt_set</b> (POINT *pd, int x, int y) <i>Initialize pd to (x, y).</i>
INLINE POINT *	<b>pt_add</b> (POINT *pd, const POINT *pa, const POINT *pb) <i>Point addition: pd = pa + pb.</i>
INLINE POINT *	<b>pt_sub</b> (POINT *pd, const POINT *pa, const POINT *pb) <i>Point subtraction: pd = pa - pb.</i>
INLINE POINT *	<b>pt_scale</b> (POINT *pd, const POINT *pa, int c) <i>Point scale: pd = c * pa.</i>
INLINE POINT *	<b>pt_add_eq</b> (POINT *pd, const POINT *pb) <i>Point increment: pd += pb.</i>
INLINE POINT *	<b>pt_sub_eq</b> (POINT *pd, const POINT *pb) <i>Point decrement: pd -= pb.</i>
INLINE POINT *	<b>pt_scale_eq</b> (POINT *pd, int c) <i>Point scale: pd *= c.</i>
INLINE int	<b>pt_cross</b> (const POINT *pa, const POINT *pb) <i>Point 'cross'-product: pa × pb.</i>
INLINE int	<b>pt_dot</b> (const POINT *pa, const POINT *pb) <i>Point 'dot'-product: pa · pb.</i>
int	<b>pt_in_rect</b> (const POINT *pt, const struct RECT *rc)

## Function Documentation

```
INLINE int pt_cross ( const POINT * pa,  
                      const POINT * pb  
)
```

Point 'cross'-product:  $pa \times pb$ .

Actually, there's no such thing as a 2D cross-product, but you could extend it to 3D and get the value of its z-component, which can be used for a test for parallelism.

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# Vector functions

[[Math](#)]

# Data Structures

struct	<b>VECTOR</b>
	<i>Vector struct.</i> <a href="#">More...</a>

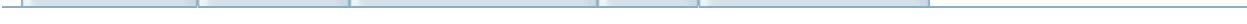
# Functions

INLINE <b>VECTOR</b> *	<b>vec_set</b> ( <b>VECTOR</b> *vd, <b>FIXED</b> x, <b>FIXED</b> y, <b>FIXED</b> z) <i>Initialize a vector.</i>
INLINE <b>VECTOR</b> *	<b>vec_add</b> ( <b>VECTOR</b> *vd, const <b>VECTOR</b> *va, const <b>VECTOR</b> *vb) <i>Add vectors: <math>d = a + b;</math></i>
INLINE <b>VECTOR</b> *	<b>vec_sub</b> ( <b>VECTOR</b> *vd, const <b>VECTOR</b> *va, const <b>VECTOR</b> *vb) <i>Subtract vectors: <math>d = a - b;</math></i>
INLINE <b>VECTOR</b> *	<b>vec_mul</b> ( <b>VECTOR</b> *vd, const <b>VECTOR</b> *va, const <b>VECTOR</b> *vb) <i>Multiply vectors elements: <math>d = S(ax, ay, az) \diamond b.</math></i>
INLINE <b>VECTOR</b> *	<b>vec_scale</b> ( <b>VECTOR</b> *vd, const <b>VECTOR</b> *va, <b>FIXED</b> c) <i>Scale vector: <math>d = c*a.</math></i>
INLINE <b>FIXED</b>	<b>vec_dot</b> (const <b>VECTOR</b> *va, const <b>VECTOR</b> *vb) <i>Dot-product: <math>d = a \diamond b.</math></i>
INLINE <b>VECTOR</b> *	<b>vec_add_eq</b> ( <b>VECTOR</b> *vd, const <b>VECTOR</b> *vb) <i>Increment vector: <math>d += b;</math></i>
INLINE <b>VECTOR</b> *	<b>vec_sub_eq</b> ( <b>VECTOR</b> *vd, const <b>VECTOR</b> *vb) <i>Decrease vector: <math>d -= b;</math></i>
INLINE <b>VECTOR</b> *	<b>vec_mul_eq</b> ( <b>VECTOR</b> *vd, const <b>VECTOR</b> *vb) <i>Multiply vectors elements: <math>d = S(dx, dy, dz) \diamond b.</math></i>
INLINE <b>VECTOR</b> *	<b>vec_scale_eq</b> ( <b>VECTOR</b> *vd, <b>FIXED</b> c) <i>Scale vector: <math>d = c*d.</math></i>
<b>VECTOR</b> *	<b>vec_cross</b> ( <b>VECTOR</b> *vd, const <b>VECTOR</b> *va, const <b>VECTOR</b> *vb)

## Detailed Description

---

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  **doxygen** 1.5.3

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# Rect functions

## [Math]

# Data Structures

struct	<b>RECT32</b>
--------	---------------

*Rectangle struct.* [More...](#)

# Functions

INLINE RECT *	<b>rc_set</b> (RECT *rc, int l, int t, int r, int b) <i>Initialize a rectangle.</i>
INLINE RECT *	<b>rc_set2</b> (RECT *rc, int x, int y, int w, int h) <i>Initialize a rectangle, with sizes inside of max boundaries.</i>
INLINE int	<b>rc_width</b> (const RECT *rc) <i>Get rectangle width.</i>
INLINE int	<b>rc_height</b> (const RECT *rc) <i>Get rectangle height.</i>
INLINE RECT *	<b>rc_set_pos</b> (RECT *rc, int x, int y) <i>Move rectangle to (x, y) position.</i>
INLINE RECT *	<b>rc_set_size</b> (RECT *rc, int w, int h) <i>Reside rectangle.</i>
INLINE RECT *	<b>rc_move</b> (RECT *rc, int dx, int dy) <i>Move rectangle by (dx, dy).</i>
INLINE RECT *	<b>rc_inflate</b> (RECT *rc, int dw, int dh) <i>Increase size by dw horizontally and dh vertically.</i>
INLINE RECT *	<b>rc_inflate2</b> (RECT *rc, const RECT *dr) <i>Increase sizes on all sides by values of rectangle dr.</i>
RECT *	<b>rc_normalize</b> (RECT *rc)

## **Detailed Description**

---

# Function Documentation

```
INLINE RECT * rc_set ( RECT * rc,
                      int      l,
                      int      t,
                      int      r,
                      int      b
)
```

Initialize a rectangle.

## Parameters:

- l* Left side.
- t* Top side.
- r* Right side.
- b* Bottom side.

```
INLINE RECT * rc_set2 ( RECT * rc,
                       int      x,
                       int      y,
                       int      w,
                       int      h
)
```

Initialize a rectangle, with sizes inside of max boundaries.

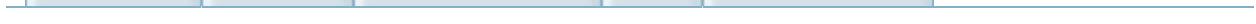
## Parameters:

- x* Left side.
- y* Top side.
- w* Width.

*h* Height.

---

Generated on Mon Aug 25 17:03:57 2008 for libtongc by  doxygen 1.5.3

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **Memory Map**

Basic memory map. [More...](#)

# Modules

<b>Memory map bit(fields)</b> <i>List of all bit(field) definitions of memory mapped items.</i>
<b>Memory mapped arrays</b> <i>These are some macros for easier access of various memory sections. They're all arrays or matrices, using the types that would be the most natural for that concept.</i>
<b>IO Registers</b>
<b>IO Alternates</b> <i>Alternate names for some of the registers.</i>

## Main sections

#define	<b>MEM_EWRAM</b> 0x02000000 <i>External work RAM.</i>
#define	<b>MEM_IWRAM</b> 0x03000000 <i>Internal work RAM.</i>
#define	<b>MEM_IO</b> 0x04000000 <i>I/O registers.</i>
#define	<b>MEM_PAL</b> 0x05000000 <i>Palette. Note: no 8bit write !!</i>
#define	<b>MEM_VRAM</b> 0x06000000 <i>Video RAM. Note: no 8bit write !!</i>
#define	<b>MEM_OAM</b> 0x07000000 <i>Object Attribute Memory (OAM) Note: no 8bit write !!</i>
#define	<b>MEM_ROM</b> 0x08000000 <i>ROM. No write at all (duh).</i>
#define	<b>MEM_SRAM</b> 0x0E000000 <i>Static RAM. 8bit write only.</i>

## Main section sizes

```
#define EWRAM_SIZE 0x40000
#define IWRAM_SIZE 0x08000
#define PAL_SIZE 0x00400
#define VRAM_SIZE 0x18000
#define OAM_SIZE 0x00400
#define SRAM_SIZE 0x10000
```

## Sub section sizes

#define	<b>PAL_BG_SIZE</b>	0x00200
		<i>BG palette size.</i>
#define	<b>PAL_OBJ_SIZE</b>	0x00200
		<i>Object palette size.</i>
#define	<b>CBB_SIZE</b>	0x04000
		<i>Charblock size.</i>
#define	<b>SBB_SIZE</b>	0x00800
		<i>Screenblock size.</i>
#define	<b>VRAM_BG_SIZE</b>	0x10000
		<i>BG VRAM size.</i>
#define	<b>VRAM_OBJ_SIZE</b>	0x08000
		<i>Object VRAM size.</i>
#define	<b>M3_SIZE</b>	0x12C00
		<i>Mode 3 buffer size.</i>
#define	<b>M4_SIZE</b>	0x09600
		<i>Mode 4 buffer size.</i>
#define	<b>M5_SIZE</b>	0x0A000
		<i>Mode 5 buffer size.</i>
#define	<b>VRAM_PAGE_SIZE</b>	0x0A000
		<i>Bitmap page size.</i>

## Sub sections

#define	<b>REG_BASE</b> MEM_IO
#define	<b>MEM_PAL_BG</b> (MEM_PAL) <i>Background palette address.</i>
#define	<b>MEM_PAL_OBJ</b> (MEM_PAL + PAL_BG_SIZE) <i>Object palette address.</i>
#define	<b>MEM_VRAM_FRONT</b> (MEM_VRAM) <i>Front page address.</i>
#define	<b>MEM_VRAM_BACK</b> (MEM_VRAM + VRAM_PAGE_SIZE) <i>Back page address.</i>
#define	<b>MEM_VRAM_OBJ</b> (MEM_VRAM + VRAM_BG_SIZE) <i>Object VRAM address.</i>

---

## Detailed Description

Basic memory map.

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **Memory map bit(fields)**

## **[[Memory Map](#)]**

List of all bit(field) definitions of memory mapped items. [More...](#)

# Modules

<b>Display Control Flags</b> <i>Bits for REG_DISPcnt.</i>
<b>Display Status Flags</b> <i>Bits for REG_DISPstat.</i>
<b>Background Control Flags</b> <i>Bits for REG_BGxCnt.</i>
<b>Graphic effects</b>
<b>Blend Flags</b> <i>Macros for REG_BLDCnt, REG_BLDY and REG_BLDALPHA.</i>
<b>Tone Generator, Sweep Flags</b> <i>Bits for REG_SND1SWEEP (aka REG_SOUND1Cnt_L).</i>
<b>Tone Generator, Square Flags</b> <i>Bits for REG_SND{1,2,4}CNT (aka REG_SOUND1Cnt_H, REG_SOUND2Cnt_L, REG_SOUND4Cnt_L, respectively).</i>
<b>Tone Generator, Frequency Flags</b> <i>Bits for REG_SND{1-3}FREQ (aka REG_SOUND1Cnt_X, REG_SOUND2Cnt_H, REG_SOUND3Cnt_X).</i>
<b>Tone Generator, Control Flags</b> <i>Bits for REG_SNDDMGCNT (aka REG_SOUNDcnt_L).</i>
<b>Direct Sound Flags</b> <i>Bits for REG_SNDDSCNT (aka REG_SOUNDcnt_H).</i>
<b>Sound Status Flags</b> <i>Bits for REG SNDSTAT (and REG_SOUNDcnt_X).</i>
<b>DMA Control Flags</b> <i>Bits for REG_DMAXCnt.</i>
<b>Timer Control Flags</b> <i>Bits for REG_TMXCnt.</i>
<b>Serial I/O Control</b> <i>Bits for REG_TMXCnt.</i>
<b>Comm control.</b> <i>Communication mode select and general purpose I/O (REG_RCNT).</i>

**Key Flags**

*Bits for REG\_KEYINPUT and REG\_KEYCNT.*

**Key Control Flags**

*Bits for REG\_KEYCNT.*

**Interrupt Flags**

*Bits for REG\_IE, REG\_IF and REG\_IFBIOS.*

**Waitstate Control Flags**

*Bits for REG\_WAITCNT.*

**Screen-entry Flags****Object Attribute 0 Flags****Object Attribute 1 Flags****Object Attribute 2 Flags**

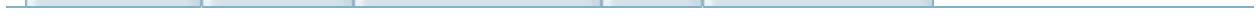
---

## Detailed Description

List of all bit(field) definitions of memory mapped items.

---

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **Display Control Flags**

## [**Memory map bit(fields)**]

Bits for REG\_DISPCNT. [More...](#)

## Defines

#define	<b>DCNT_MODE0</b> 0
	<i>Mode 0; bg 0-4: reg.</i>
#define	<b>DCNT_MODE1</b> 0x0001
	<i>Mode 1; bg 0-1: reg; bg 2: affine.</i>
#define	<b>DCNT_MODE2</b> 0x0002
	<i>Mode 2; bg 2-3: affine.</i>
#define	<b>DCNT_MODE3</b> 0x0003
	<i>Mode 3; bg2: 240x160@16 bitmap.</i>
#define	<b>DCNT_MODE4</b> 0x0004
	<i>Mode 4; bg2: 240x160@8 bitmap.</i>
#define	<b>DCNT_MODE5</b> 0x0005
	<i>Mode 5; bg2: 160x128@16 bitmap.</i>
#define	<b>DCNT_GB</b> 0x0008
	<i>(R) GBC indicator</i>
#define	<b>DCNT_PAGE</b> 0x0010
	<i>Page indicator.</i>
#define	<b>DCNT_OAM_HBL</b> 0x0020
	<i>Allow OAM updates in HBlank.</i>
#define	<b>DCNT_OBJ_2D</b> 0
	<i>OBJ-VRAM as matrix.</i>
#define	<b>DCNT_OBJ_1D</b> 0x0040
	<i>OBJ-VRAM as array.</i>
#define	<b>DCNT_BLANK</b> 0x0080
	<i>Force screen blank.</i>
#define	<b>DCNT_BG0</b> 0x0100
	<i>Enable bg 0.</i>
#define	<b>DCNT_BG1</b> 0x0200
	<i>Enable bg 1.</i>
#define	<b>DCNT_BG2</b> 0x0400
	<i>Enable bg 2.</i>
#define	<b>DCNT_BG3</b> 0x0800

	<i>Enable bg 3.</i>
#define	<b>DCNT_OBJ</b> 0x1000 <i>Enable objects.</i>
#define	<b>DCNT_WIN0</b> 0x2000 <i>Enable window 0.</i>
#define	<b>DCNT_WIN1</b> 0x4000 <i>Enable window 1.</i>
#define	<b>DCNT_WINOBJ</b> 0x8000 <i>Enable object window.</i>
#define	<b>DCNT_MODE_MASK</b> 0x0007
#define	<b>DCNT_MODE_SHIFT</b> 0
#define	<b>DCNT_MODE(n)</b> ((n)<<DCNT_MODE_SHIFT)
#define	<b>DCNT_LAYER_MASK</b> 0x1F00
#define	<b>DCNT_LAYER_SHIFT</b> 8
#define	<b>DCNT_LAYER(n)</b> ((n)<<DCNT_LAYER_SHIFT)
#define	<b>DCNT_WIN_MASK</b> 0xE000
#define	<b>DCNT_WIN_SHIFT</b> 13
#define	<b>DCNT_WIN(n)</b> ((n)<<DCNT_WIN_SHIFT)
#define	<b>DCNT_BUILD(mode, layer, win, obj1d, objhbl)</b>

---

## **Detailed Description**

Bits for REG\_DISPCNT.

---

# Define Documentation

```
#define DCNT_BUILD ( mode,  
                    layer,  
                    win,  
                    obj1d,  
                    objhbl )
```

**Value:**

```
(  
    (((win)&7)<<13) | (((layer)&31)<<8) |  
    | (((objhbl)&1)<<5) | ((mode)&7)  
)
```

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# Display Status Flags

## [Memory map bit(fields)]

Bits for REG\_DISPSTAT. [More...](#)

## Defines

#define	<b>DSTAT_IN_VBL</b>	0x0001
		<i>Now in VBlank.</i>
#define	<b>DSTAT_IN_HBL</b>	0x0002
		<i>Now in HBlank.</i>
#define	<b>DSTAT_IN_VCT</b>	0x0004
		<i>Now in set VCount.</i>
#define	<b>DSTAT_VBL_IRQ</b>	0x0008
		<i>Enable VBlank irq.</i>
#define	<b>DSTAT_HBL_IRQ</b>	0x0010
		<i>Enable HBlank irq.</i>
#define	<b>DSTAT_VCT_IRQ</b>	0x0020
		<i>Enable VCount irq.</i>
#define	<b>DSTAT_VCT_MASK</b>	0xFF00
#define	<b>DSTAT_VCT_SHIFT</b>	8
#define	<b>DSTAT_VCT(n)</b>	((n)<<DSTAT_VCT_SHIFT)

---

## Detailed Description

Bits for REG\_DISPSTAT.

---

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  **1.5.3**

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **Background Control Flags**

## [Memory map bit(fields)]

Bits for REG\_BGxCNT. [More...](#)

## Defines

#define	<b>BG_MOSAIC</b> 0x0040
	<i>Enable Mosaic.</i>
#define	<b>BG_4BPP</b> 0
	<i>4bpp (16 color) bg (no effect on affine bg)</i>
#define	<b>BG_8BPP</b> 0x0080
	<i>8bpp (256 color) bg (no effect on affine bg)</i>
#define	<b>BG_WRAP</b> 0x2000
	<i>Wrap around edges of affine bgs.</i>
#define	<b>BG_SIZE0</b> 0
#define	<b>BG_SIZE1</b> 0x4000
#define	<b>BG_SIZE2</b> 0x8000
#define	<b>BG_SIZE3</b> 0xC000
#define	<b>BG_REG_32x32</b> 0
	<i>reg bg, 32x32 (256x256 px)</i>
#define	<b>BG_REG_64x32</b> 0x4000
	<i>reg bg, 64x32 (512x256 px)</i>
#define	<b>BG_REG_32x64</b> 0x8000
	<i>reg bg, 32x64 (256x512 px)</i>
#define	<b>BG_REG_64x64</b> 0xC000
	<i>reg bg, 64x64 (512x512 px)</i>
#define	<b>BG_AFF_16x16</b> 0
	<i>affine bg, 16x16 (128x128 px)</i>
#define	<b>BG_AFF_32x32</b> 0x4000
	<i>affine bg, 32x32 (256x256 px)</i>
#define	<b>BG_AFF_64x64</b> 0x8000
	<i>affine bg, 64x64 (512x512 px)</i>
#define	<b>BG_AFF_128x128</b> 0xC000
	<i>affine bg, 128x128 (1024x1024 px)</i>
#define	<b>BG_PRIO_MASK</b> 0x0003
#define	<b>BG_PRIO_SHIFT</b> 0
#define	<b>BG_PRIO(n)</b> ((n)<<BG_PRIO_SHIFT)
#define	<b>BG_CBB_MASK</b> 0x000C

```
#define BG_CBB_SHIFT 2
#define BG_CBB(n) ((n)<<BG_CBB_SHIFT)
#define BG_SBB_MASK 0x1F00
#define BG_SBB_SHIFT 8
#define BG_SBB(n) ((n)<<BG_SBB_SHIFT)
#define BG_SIZE_MASK 0xC000
#define BG_SIZE_SHIFT 14
#define BG_SIZE(n) ((n)<<BG_SIZE_SHIFT)
#define BG_BUILD(ccb, sbb, size, bpp, prio, mos, wrap)
```

---

## **Detailed Description**

Bits for REG\_BGxCNT.

---

# Define Documentation

```
#define BG_BUILD ( cbb,
                 sbb,
                 size,
                 bpp,
                 prio,
                 mos,
                 wrap )
```

## Value:

```
(  
    ((size)<<14) | (((wrap)&1)<<13) | (((  
    | (((bpp)&8)<<4) | (((mos)&1)<<6) | (((  
    | ((prio)&3)  
)
```

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **Graphic effects**

## [Memory map bit(fields)]

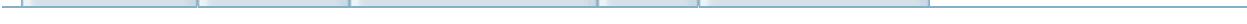
## Window macros

#define	<b>WIN_BG0</b> 0x0001 <i>Windowed bg 0.</i>
#define	<b>WIN_BG1</b> 0x0002 <i>Windowed bg 1.</i>
#define	<b>WIN_BG2</b> 0x0004 <i>Windowed bg 2.</i>
#define	<b>WIN_BG3</b> 0x0008 <i>Windowed bg 3.</i>
#define	<b>WIN_OBJ</b> 0x0010 <i>Windowed objects.</i>
#define	<b>WIN_ALL</b> 0x001F <i>All layers in window.</i>
#define	<b>WIN_BLD</b> 0x0020 <i>Windowed blending.</i>
#define	<b>WIN_LAYER_MASK</b> 0x003F <i>Windowed bg 0.</i>
#define	<b>WIN_LAYER_SHIFT</b> 0 <i>Windowed bg 0.</i>
#define	<b>WIN_LAYER(n)</b> ((n)<<WIN_LAYER_SHIFT) <i>Windowed bg 0.</i>
#define	<b>WIN_BUILD</b> (low, high) ( ((high)<<8)   (low) ) <i>Windowed bg 0.</i>
#define	<b>WININ_BUILD</b> (win0, win1) WIN_BUILD(win0, win1) <i>Windowed bg 0.</i>
#define	<b>WINOUT_BUILD</b> (out, obj) WIN_BUILD(out, obj) <i>Windowed bg 0.</i>

## Mosaic macros

```
#define MOS_BH_MASK 0x000F
#define MOS_BH_SHIFT 0
#define MOS_BH(n) ((n)<<MOS_BH_SHIFT)
#define MOS_BV_MASK 0x00F0
#define MOS_BV_SHIFT 4
#define MOS_BV(n) ((n)<<MOS_BV_SHIFT)
#define MOS_OH_MASK 0x0F00
#define MOS_OH_SHIFT 8
#define MOS_OH(n) ((n)<<MOS_OH_SHIFT)
#define MOS_OV_MASK 0xF000
#define MOS_OV_SHIFT 12
#define MOS_OV(n) ((n)<<MOS_OV_SHIFT)
#define MOS_BUILD(bh, bv, oh, ov) ( (((ov)&15)<<12) | (((oh)&15)<<8) |
((bv)&15)<<4)| ((bh)&15) )
```

Generated on Mon Aug 25 17:03:57 2008 for libtonc by  1.5.3

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

## **Blend Flags**

### **[Memory map bit(fields)]**

Macros for REG\_BLDCNT, REG\_BLDY and REG\_BLDALPHA.

[More...](#)

## Blend weights

```
#define BLD_EVA_MASK 0x001F
#define BLD_EVA_SHIFT 0
#define BLD_EVA(n) ((n)<<BLD_EVA_SHIFT)
#define BLD_EVB_MASK 0x1F00
#define BLD_EVB_SHIFT 8
#define BLD_EVB(n) ((n)<<BLD_EVB_SHIFT)
#define BLDA_BUILD(eva, evb) ( ((eva)&31) | (((evb)&31)<<8) )
```

## Fade levels

```
#define BLDY_MASK 0x001F
#define BLDY_SHIFT 0
#define BLDY(n) ((n)<<BLD_EY_SHIFT)
#define BLDY_BUILD(ey) ((ey)&31)
```

## Defines

#define	<b>BLD_BG0</b> 0x0001	\ name Blend control
#define	<b>BLD_BG1</b> 0x0002	Blend bg 1.
#define	<b>BLD_BG2</b> 0x0004	Blend bg 2.
#define	<b>BLD_BG3</b> 0x0008	Blend bg 3.
#define	<b>BLD_OBJ</b> 0x0010	Blend objects.
#define	<b>BLD_ALL</b> 0x001F	All layers (except backdrop).
#define	<b>BLD_BACKDROP</b> 0x0020	Blend backdrop.
#define	<b>BLD_OFF</b> 0	Blend mode is off.
#define	<b>BLD_STD</b> 0x0040	Normal alpha blend (with REG_EV).
#define	<b>BLD_WHITE</b> 0x0080	Fade to white (with REG_Y).
#define	<b>BLD_BLACK</b> 0x00C0	Fade to black (with REG_Y).
#define	<b>BLD_TOP_MASK</b> 0x003F	
#define	<b>BLD_TOP_SHIFT</b> 0	
#define	<b>BLD_TOP(n)</b> ((n)<<BLD_TOP_SHIFT)	
#define	<b>BLD_MODE_MASK</b> 0x00C0	
#define	<b>BLD_MODE_SHIFT</b> 6	
#define	<b>BLD_MODE(n)</b> ((n)<<BLD_MODE_SHIFT)	
#define	<b>BLD_BOT_MASK</b> 0x3F00	
#define	<b>BLD_BOT_SHIFT</b> 8	
#define	<b>BLD_BOT(n)</b> ((n)<<BLD_BOT_SHIFT)	
#define	<b>BLD_BUILD</b> (top, bot, mode)	((bot)&63)<<8)   (((mode)&3)<<6)

((top)&63) )

---

## **Detailed Description**

Macros for REG\_BLDCNT, REG\_BLDY and REG\_BLDALPHA.

---

# Define Documentation

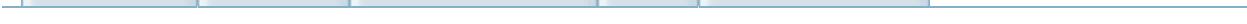
```
#define BLD_BG0 0x0001
```

```
\ name Blend control
```

```
Blend bg 0
```

---

*Generated on Mon Aug 25 17:03:57 2008 for libtong by  doxygen 1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

## **Tone Generator, Sweep Flags**

### **[Memory map bit(fields)]**

Bits for REG\_SND1SWEEP (aka REG\_SOUND1CNT\_L).

[More...](#)

## Defines

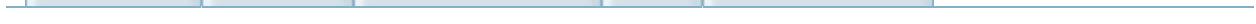
#define	<b>SSW_INC</b> 0
	<i>Increasing sweep rate.</i>
#define	<b>SSW_DEC</b> 0x0008
	<i>Decreasing sweep rate.</i>
#define	<b>SSW_OFF</b> 0x0008
	<i>Disable sweep altogether.</i>
#define	<b>SSW_SHIFT_MASK</b> 0x0007
#define	<b>SSW_SHIFT_SHIFT</b> 0
#define	<b>SSW_SHIFT(n)</b> ((n)<<SSW_SHIFT_SHIFT)
#define	<b>SSW_TIME_MASK</b> 0x0070
#define	<b>SSW_TIME_SHIFT</b> 4
#define	<b>SSW_TIME(n)</b> ((n)<<SSW_TIME_SHIFT)
#define	<b>SSW_BUILD(shift, dir, time)</b> ( (((time)&7)<<4)   ((dir)<<3)   ((shift)&7) )

## Detailed Description

Bits for REG\_SND1SWEEP (aka REG\_SOUND1CNT\_L).

---

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

## **Tone Generator, Square Flags**

### **[Memory map bit(fields)]**

Bits for REG\_SND{1,2,4}CNT (aka REG\_SOUND1CNT\_H, REG\_SOUND2CNT\_L, REG\_SOUND4CNT\_L, respectively).

[More...](#)

## Defines

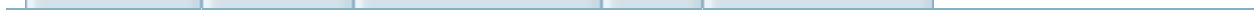
#define	<b>SSQR_DUTY1_8</b> 0 <i>12.5% duty cycle (#-----)</i>
#define	<b>SSQR_DUTY1_4</b> 0x0040 <i>25% duty cycle (##-----)</i>
#define	<b>SSQR_DUTY1_2</b> 0x0080 <i>50% duty cycle (#####---)</i>
#define	<b>SSQR_DUTY3_4</b> 0x00C0 <i>75% duty cycle (#####---) Equivalent to 25%</i>
#define	<b>SSQR_INC</b> 0 <i>Increasing volume.</i>
#define	<b>SSQR_DEC</b> 0x0800 <i>Decreasing volume.</i>
#define	<b>SSQR_LEN_MASK</b> 0x003F
#define	<b>SSQR_LEN_SHIFT</b> 0
#define	<b>SSQR_LEN(n)</b> ((n)<<SSQR_LEN_SHIFT)
#define	<b>SSQR_DUTY_MASK</b> 0x00C0
#define	<b>SSQR_DUTY_SHIFT</b> 6
#define	<b>SSQR_DUTY(n)</b> ((n)<<SSQR_DUTY_SHIFT)
#define	<b>SSQR_TIME_MASK</b> 0x0700
#define	<b>SSQR_TIME_SHIFT</b> 8
#define	<b>SSQR_TIME(n)</b> ((n)<<SSQR_TIME_SHIFT)
#define	<b>SSQR_IVOL_MASK</b> 0xF000
#define	<b>SSQR_IVOL_SHIFT</b> 12
#define	<b>SSQR_IVOL(n)</b> ((n)<<SSQR_IVOL_SHIFT)
#define	<b>SSQR_ENV_BUILD(ivol, dir, time)</b> ( ((ivol)<<12)   ((dir)<<11)   (((time)&7)<<8) )
#define	<b>SSQR_BUILD(_ivol, dir, step, duty, len)</b> ( SSQR_ENV_BUILD(ivol,dir,step)   (((duty)&3)<<6)   ((len)&63) )

## Detailed Description

Bits for REG\_SND{1,2,4}CNT (aka REG\_SOUND1CNT\_H, REG\_SOUND2CNT\_L, REG\_SOUND4CNT\_L, respectively).

---

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

## **Tone Generator, Frequency Flags**

### [Memory map bit(fields)]

Bits for REG\_SND{1-3}FREQ (aka REG\_SOUND1CNT\_X, REG\_SOUND2CNT\_H, REG\_SOUND3CNT\_X). [More...](#)

## Defines

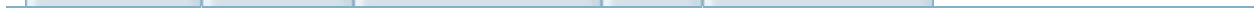
#define	<b>SFREQ_HOLD</b> 0
	<i>Continuous play.</i>
#define	<b>SFREQ_TIMED</b> 0x4000
	<i>Timed play.</i>
#define	<b>SFREQ_RESET</b> 0x8000
	<i>Reset sound.</i>
#define	<b>SFREQ_RATE_MASK</b> 0x07FF
#define	<b>SFREQ_RATE_SHIFT</b> 0
#define	<b>SFREQ_RATE(n)</b> ((n)<<SFREQ_RATE_SHIFT)
#define	<b>SFREQ_BUILD</b> (rate, timed, reset) ( ((rate)&0x7FF)   ((timed)<<14)   ((reset)<<15) )

## Detailed Description

Bits for REG\_SND{1-3}FREQ (aka REG\_SOUND1CNT\_X,  
REG\_SOUND2CNT\_H, REG\_SOUND3CNT\_X).

---

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

## **Tone Generator, Control Flags**

### **[Memory map bit(fields)]**

Bits for REG\_SNDDMGCNT (aka REG\_SOUNDcnt\_L).

[More...](#)

## Defines

#define	<b>SDMG_LSQR1</b> 0x0100
	<i>Enable channel 1 on left.</i>
#define	<b>SDMG_LSQR2</b> 0x0200
	<i>Enable channel 2 on left.</i>
#define	<b>SDMG_LWAVE</b> 0x0400
	<i>Enable channel 3 on left.</i>
#define	<b>SDMG_LNOISE</b> 0x0800
	<i>Enable channel 4 on left.</i>
#define	<b>SDMG_RSQR1</b> 0x1000
	<i>Enable channel 1 on right.</i>
#define	<b>SDMG_RSQR2</b> 0x2000
	<i>Enable channel 2 on right.</i>
#define	<b>SDMG_RWAVE</b> 0x4000
	<i>Enable channel 3 on right.</i>
#define	<b>SDMG_RNOISE</b> 0x8000
	<i>Enable channel 4 on right.</i>
#define	<b>SDMG_LVOL_MASK</b> 0x0007
#define	<b>SDMG_LVOL_SHIFT</b> 0
#define	<b>SDMG_LVOL(n)</b> ((n)<<SDMG_LVOL_SHIFT)
#define	<b>SDMG_RVOL_MASK</b> 0x0070
#define	<b>SDMG_RVOL_SHIFT</b> 4
#define	<b>SDMG_RVOL(n)</b> ((n)<<SDMG_RVOL_SHIFT)
#define	<b>SDMG_SQR1</b> 0x01
#define	<b>SDMG_SQR2</b> 0x02
#define	<b>SDMG_WAVE</b> 0x04
#define	<b>SDMG_NOISE</b> 0x08
#define	<b>SDMG_BUILD(_lmode, _rmode, _lvol, _rvol)</b> ( ((_rmode)<<12)   (_lmode)<<8)   (((_rvol)&7)<<4)   (((_lvol)&7) ) )
#define	<b>SDMG_BUILD_LR(_mode, _vol)</b> SDMG_BUILD(_mode, _mode, _vol, _vol)

## Detailed Description

Bits for REG\_SNDDMGCNT (aka REG\_SOUND\_CNT\_L).

---

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

## **Direct Sound Flags** **[Memory map bit(fields)]**

Bits for REG\_SNDDSCNT (aka REG\_SOUND\_CNT\_H). [More...](#)

## Defines

#define	<b>SDS_DMG25</b> 0
	<i>Tone generators at 25% volume.</i>
#define	<b>SDS_DMG50</b> 0x0001
	<i>Tone generators at 50% volume.</i>
#define	<b>SDS_DMG100</b> 0x0002
	<i>Tone generators at 100% volume.</i>
#define	<b>SDS_A50</b> 0
	<i>Direct Sound A at 50% volume.</i>
#define	<b>SDS_A100</b> 0x0004
	<i>Direct Sound A at 100% volume.</i>
#define	<b>SDS_B50</b> 0
	<i>Direct Sound B at 50% volume.</i>
#define	<b>SDS_B100</b> 0x0008
	<i>Direct Sound B at 100% volume.</i>
#define	<b>SDS_AR</b> 0x0100
	<i>Enable Direct Sound A on right.</i>
#define	<b>SDS_AL</b> 0x0200
	<i>Enable Direct Sound A on left.</i>
#define	<b>SDS_ATMR0</b> 0
	<i>Direct Sound A to use timer 0.</i>
#define	<b>SDS_ATMR1</b> 0x0400
	<i>Direct Sound A to use timer 1.</i>
#define	<b>SDS_ARESET</b> 0x0800
	<i>Reset FIFO of Direct Sound A.</i>
#define	<b>SDS_BR</b> 0x1000
	<i>Enable Direct Sound B on right.</i>
#define	<b>SDS_BL</b> 0x2000
	<i>Enable Direct Sound B on left.</i>
#define	<b>SDS_BTMR0</b> 0
	<i>Direct Sound B to use timer 0.</i>
#define	<b>SDS_BTMR1</b> 0x4000

*Direct Sound B to use timer 1.*

#define **SDS\_BRESET** 0x8000

*Reset FIFO of Direct Sound B.*

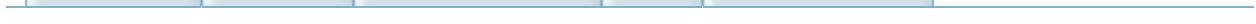
---

## Detailed Description

Bits for REG\_SNDDSCNT (aka REG\_SOUND\_CNT\_H).

---

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

## **Sound Status Flags** **[Memory map bit(fields)]**

Bits for REG\_SNDSTAT (and REG\_SOUNDcnt\_X). [More...](#)

## Defines

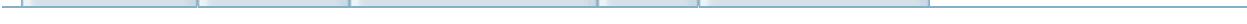
#define	<b>SSTAT_SQR1</b> 0x0001 <i>(R) Channel 1 status</i>
#define	<b>SSTAT_SQR2</b> 0x0002 <i>(R) Channel 2 status</i>
#define	<b>SSTAT_WAVE</b> 0x0004 <i>(R) Channel 3 status</i>
#define	<b>SSTAT_NOISE</b> 0x0008 <i>(R) Channel 4 status</i>
#define	<b>SSTAT_DISABLE</b> 0 <i>Disable sound.</i>
#define	<b>SSTAT_ENABLE</b> 0x0080 <i>Enable sound. NOTE: enable before using any other sound regs.</i>

## Detailed Description

Bits for REG\_SNDSTAT (and REG\_SOUND\_CNT\_X).

---

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# DMA Control Flags

## [Memory map bit(fields)]

Bits for REG\_DMAXCNT. [More...](#)

## Defines

#define	<b>DMA_DST_INC</b> 0
	<i>Incrementing destination address.</i>
#define	<b>DMA_DST_DEC</b> 0x00200000
	<i>Decrementing destination.</i>
#define	<b>DMA_DST_FIXED</b> 0x00400000
	<i>Fixed destination.</i>
#define	<b>DMA_DST_RELOAD</b> 0x00600000
	<i>Increment destination, reset after full run.</i>
#define	<b>DMA_SRC_INC</b> 0
	<i>Incrementing source address.</i>
#define	<b>DMA_SRC_DEC</b> 0x00800000
	<i>Decrementing source address.</i>
#define	<b>DMA_SRC_FIXED</b> 0x01000000
	<i>Fixed source address.</i>
#define	<b>DMA_REPEAT</b> 0x02000000
	<i>Repeat transfer at next start condition.</i>
#define	<b>DMA_16</b> 0
	<i>Transfer by halfword.</i>
#define	<b>DMA_32</b> 0x04000000
	<i>Transfer by word.</i>
#define	<b>DMA_AT_NOW</b> 0
	<i>Start transfer now.</i>
#define	<b>DMA_GAMEPAK</b> 0x08000000
	<i>Gamepak DRQ.</i>
#define	<b>DMA_AT_VBLANK</b> 0x10000000
	<i>Start transfer at VBlank.</i>
#define	<b>DMA_AT_HBLANK</b> 0x20000000
	<i>Start transfer at HBlank.</i>
#define	<b>DMA_AT_SPECIAL</b> 0x30000000
	<i>Start copy at 'special' condition. Channel dependent.</i>
#define	<b>DMA_AT_FIFO</b> 0x30000000

	<i>Start at FIFO empty (DMA0/DMA1).</i>
#define	<b>DMA_AT_REFRESH</b> 0x30000000 <i>VRAM special; start at VCount=2 (DMA3).</i>
#define	<b>DMA_IRQ</b> 0x40000000 <i>Enable DMA irq.</i>
#define	<b>DMA_ENABLE</b> 0x80000000 <i>Enable DMA.</i>
#define	<b>DMA_COUNT_MASK</b> 0x0000FFFF
#define	<b>DMA_COUNT_SHIFT</b> 0
#define	<b>DMA_COUNT(n)</b> ((n)<<DMA_COUNT_SHIFT)
#define	<b>DMA_NOW</b> (DMA_ENABLE   DMA_AT_NOW)
#define	<b>DMA_16NOW</b> (DMA_NOW   DMA_16)
#define	<b>DMA_32NOW</b> (DMA_NOW   DMA_32)
#define	<b>DMA_CPY16</b> (DMA_NOW   DMA_16)
#define	<b>DMA_CPY32</b> (DMA_NOW   DMA_32)
#define	<b>DMA_FILL16</b> (DMA_NOW   DMA_SRC_FIXED   DMA_16)
#define	<b>DMA_FILL32</b> (DMA_NOW   DMA_SRC_FIXED   DMA_32)
#define	<b>DMA_HDMA</b> (DMA_ENABLE   DMA_REPEAT   DMA_AT_HBLANK   DMA_DST_RELOAD)

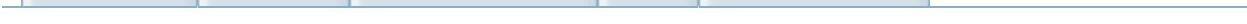
---

## Detailed Description

Bits for REG\_DMAMASK.

---

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  **1.5.3**

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **Timer Control Flags**

## [Memory map bit(fields)]

Bits for REG\_TMxCNT. [More...](#)

## Defines

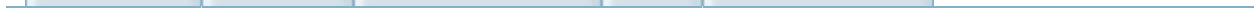
#define	<b>TM_FREQ_SYS</b> 0
	<i>System clock timer (16.7 Mhz).</i>
#define	<b>TM_FREQ_1</b> 0
	<i>1 cycle/tick (16.7 Mhz)</i>
#define	<b>TM_FREQ_64</b> 0x0001
	<i>64 cycles/tick (262 kHz)</i>
#define	<b>TM_FREQ_256</b> 0x0002
	<i>256 cycles/tick (66 kHz)</i>
#define	<b>TM_FREQ_1024</b> 0x0003
	<i>1024 cycles/tick (16 kHz)</i>
#define	<b>TM CASCADE</b> 0x0004
	<i>Increment when preceding timer overflows.</i>
#define	<b>TM_IRQ</b> 0x0040
	<i>Enable timer irq.</i>
#define	<b>TM_ENABLE</b> 0x0080
	<i>Enable timer.</i>
#define	<b>TM_FREQ_MASK</b> 0x0003
#define	<b>TM_FREQ_SHIFT</b> 0
#define	<b>TM_FREQ(n)</b> ((n)<<TM_FREQ_SHIFT)

## Detailed Description

Bits for REG\_TMxCNT.

---

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **Serial I/O Control**

## **[Memory map bit(fields)]**

Bits for REG\_TMxCNT. [More...](#)

## General SIO bits.

#define	<b>SIO_MODE_8BIT</b> 0x0000
	<i>Normal comm mode, 8-bit.</i>
#define	<b>SIO_MODE_32BIT</b> 0x1000
	<i>Normal comm mode, 32-bit.</i>
#define	<b>SIO_MODE_MULTI</b> 0x2000
	<i>Multi-play comm mode.</i>
#define	<b>SIO_MODE_UART</b> 0x3000
	<i>UART comm mode.</i>
#define	<b>SIO_SI_HIGH</b> 0x0004
	<i>Normal comm mode, 8-bit.</i>
#define	<b>SIO_IRQ</b> 0x4000
	<i>Enable serial irq.</i>
#define	<b>SIO_MODE_MASK</b> 0x3000
	<i>Normal comm mode, 8-bit.</i>
#define	<b>SIO_MODE_SHIFT</b> 12
	<i>Normal comm mode, 8-bit.</i>
#define	<b>SIO_MODE(n)</b> ((n)<<SIO_MODE_SHIFT)
	<i>Normal comm mode, 8-bit.</i>

## Normal mode bits. UNTESTED.

#define	<b>SION_CLK_EXT</b> 0x0000
	<i>Slave unit; use external clock (default).</i>
#define	<b>SION_CLK_INT</b> 0x0001
	<i>Master unit; use internal clock.</i>
#define	<b>SION_256KHZ</b> 0x0000
	<i>256 kHz clockspeed (default).</i>
#define	<b>SION_2MHZ</b> 0x0002
	<i>2 MHz clockspeed.</i>
#define	<b>SION_RECV_HIGH</b> 0x0004
	<i>SI high; opponent ready to receive (R).</i>
#define	<b>SION_SEND_HIGH</b> 0x0008
	<i>SO high; ready to transfer.</i>
#define	<b>SION_ENABLE</b> 0x0080
	<i>Start transfer/transfer enabled.</i>

## Multiplayer mode bits. UNTESTED.

#define	<b>SIOM_9600</b> 0x0000
	<i>Baud rate, 9.6 kbps.</i>
#define	<b>SIOM_38400</b> 0x0001
	<i>Baud rate, 38.4 kbps.</i>
#define	<b>SIOM_57600</b> 0x0002
	<i>Baud rate, 57.6 kbps.</i>
#define	<b>SIOM_115200</b> 0x0003
	<i>Baud rate, 115.2 kbps.</i>
#define	<b>SIOM_SI</b> 0x0004
	<i>SI port (R).</i>
#define	<b>SIOM_SLAVE</b> 0x0004
	<i>Not the master (R).</i>
#define	<b>SIOM_SD</b> 0x0008
	<i>SD port (R).</i>
#define	<b>SIOM_CONNECTED</b> 0x0008
	<i>All GBAs connected (R).</i>
#define	<b>SIOM_ERROR</b> 0x0040
	<i>Error in transfer (R).</i>
#define	<b>SIOM_ENABLE</b> 0x0080
	<i>Start transfer/transfer enabled.</i>
#define	<b>SIOM_BAUD_MASK</b> 0x0003
	<i>Baud rate, 9.6 kbps.</i>
#define	<b>SIOM_BAUD_SHIFT</b> 0
	<i>Baud rate, 9.6 kbps.</i>
#define	<b>SIOM_BAUD(n)</b> ((n)<<SIOM_BAUD_SHIFT)
	<i>Baud rate, 9.6 kbps.</i>
#define	<b>SIOM_ID_MASK</b> 0x0030
	<i>Multi-player ID mask (R).</i>
#define	<b>SIOM_ID_SHIFT</b> 4
	<i>Baud rate, 9.6 kbps.</i>
#define	<b>SIOM_ID(n)</b> ((n)<<SIOM_ID_SHIFT)

*Baud rate, 9.6 kbps.*

## UART mode bits. UNTESTED.

#define	<b>SIOU_9600</b> 0x0000 <i>Baud rate, 9.6 kbps.</i>
#define	<b>SIOU_38400</b> 0x0001 <i>Baud rate, 38.4 kbps.</i>
#define	<b>SIOU_57600</b> 0x0002 <i>Baud rate, 57.6 kbps.</i>
#define	<b>SIOU_115200</b> 0x0003 <i>Baud rate, 115.2 kbps.</i>
#define	<b>SIOU_CTS</b> 0x0004 <i>CTS enable.</i>
#define	<b>SIOU_PARITY_EVEN</b> 0x0000 <i>Use even parity.</i>
#define	<b>SIOU_PARITY_ODD</b> 0x0008 <i>Use odd parity.</i>
#define	<b>SIOU_SEND_FULL</b> 0x0010 <i>Send data is full (R).</i>
#define	<b>SIOU_RECV_EMPTY</b> 0x0020 <i>Receive data is empty (R).</i>
#define	<b>SIOU_ERROR</b> 0x0040 <i>Error in transfer (R).</i>
#define	<b>SIOU_7BIT</b> 0x0000 <i>Data is 7bits long.</i>
#define	<b>SIOU_8BIT</b> 0x0080 <i>Data is 8bits long.</i>
#define	<b>SIOU_SEND</b> 0x0100 <i>Start sending data.</i>
#define	<b>SIOU_RECV</b> 0x0200 <i>Start receiving data.</i>
#define	<b>SIOU_BAUD_MASK</b> 0x0003 <i>Baud rate, 9.6 kbps.</i>
#define	<b>SIOU_BAUD_SHIFT</b> 0

*Baud rate, 9.6 kbps.*

```
#define SIOU_BAUD(n) ((n)<<SIOU_BAUD_SHIFT)  
Baud rate, 9.6 kbps.
```

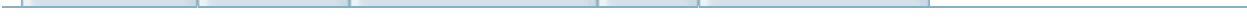
---

## Detailed Description

Bits for REG\_TMxCNT.

---

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  **1.5.3**

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

## **Comm control.**

### **[Memory map bit(fields)]**

Communication mode select and general purpose I/O  
(REG\_RCNT). [More...](#)

## Communication mode select.

#define	<b>R_MODE_NORMAL</b> 0x0000
	<i>Normal mode.</i>
#define	<b>R_MODE_MULTI</b> 0x0000
	<i>Multiplayer mode.</i>
#define	<b>R_MODE_UART</b> 0x0000
	<i>UART mode.</i>
#define	<b>R_MODE_GPIO</b> 0x8000
	<i>General purpose mode.</i>
#define	<b>R_MODE_JOYBUS</b> 0xC000
	<i>JOY mode.</i>
#define	<b>R_MODE_MASK</b> 0xC000
	<i>Normal mode.</i>
#define	<b>R_MODE_SHIFT</b> 14
	<i>Normal mode.</i>
#define	<b>R_MODE(n)</b> ((n)<<R_MODE_SHIFT)
	<i>Normal mode.</i>

## General purpose I/O data

```
#define GPIO_SC 0x0001
#define GPIO_SD 0x0002
#define GPIO_SI 0x0004
#define GPIO_SO 0x0008
#define GPIO_SC_IO 0x0010
#define GPIO_SD_IO 0x0020
#define GPIO_SI_IO 0x0040
#define GPIO_SO_IO 0x0080
#define GPIO_SC_INPUT 0x0000
#define GPIO_SD_INPUT 0x0000
#define GPIO_SI_INPUT 0x0000
#define GPIO_SO_INPUT 0x0000
#define GPIO_SC_OUTPUT 0x0010
#define GPIO_SD_OUTPUT 0x0020
#define GPIO_SI_OUTPUT 0x0040
#define GPIO_SO_OUTPUT 0x0080
#define GPIO_IRQ 0x0100
```

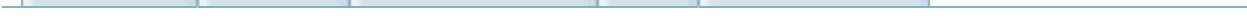
---

## Detailed Description

Communication mode select and general purpose I/O (REG\_RCNT).

---

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  **doxygen** 1.5.3

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

## Key Flags

### [Memory map bit(fields)]

Bits for REG\_KEYINPUT and REG\_KEYCNT. [More...](#)

## Defines

#define	<b>KEY_A</b> 0x0001
	<i>Button A.</i>
#define	<b>KEY_B</b> 0x0002
	<i>Button B.</i>
#define	<b>KEY_SELECT</b> 0x0004
	<i>Select button.</i>
#define	<b>KEY_START</b> 0x0008
	<i>Start button.</i>
#define	<b>KEY_RIGHT</b> 0x0010
	<i>Right D-pad.</i>
#define	<b>KEY_LEFT</b> 0x0020
	<i>Left D-pad.</i>
#define	<b>KEY_UP</b> 0x0040
	<i>Up D-pad.</i>
#define	<b>KEY_DOWN</b> 0x0080
	<i>Down D-pad.</i>
#define	<b>KEY_R</b> 0x0100
	<i>Shoulder R.</i>
#define	<b>KEY_L</b> 0x0200
	<i>Shoulder L.</i>
#define	<b>KEY_ACCEPT</b> 0x0009
	<i>Accept buttons: A or start.</i>
#define	<b>KEY_CANCEL</b> 0x0002
	<i>Cancel button: B (well, it usually is).</i>
#define	<b>KEY_RESET</b> 0x030C
	<i>St+Se+L+R.</i>
#define	<b>KEY_FIRE</b> 0x0003
	<i>Fire buttons: A or B.</i>
#define	<b>KEY_SPECIAL</b> 0x000C
	<i>Special buttons: Select or Start.</i>
#define	<b>KEY_DIR</b> 0x00F0

*Directions: left, right, up down.*

#define **KEY\_SHOULDER** 0x0300

*L or R.*

#define **KEY\_ANY** 0x03FF

*Here's the Any key :).*

#define **KEY\_MASK** 0x03FF

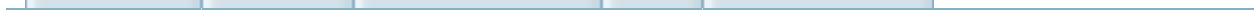
---

## Detailed Description

Bits for REG\_KEYINPUT and REG\_KEYCNT.

---

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **Key Control Flags**

## [Memory map bit(fields)]

Bits for REG\_KEYCNT. [More...](#)

## Defines

#define	<b>KCNT_IRQ</b> 0x4000
	<i>Enable key irq.</i>
#define	<b>KCNT_OR</b> 0
	<i>Interrupt on any of selected keys.</i>
#define	<b>KCNT_AND</b> 0x8000
	<i>Interrupt on all of selected keys.</i>

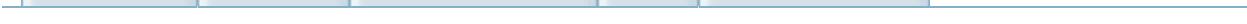
---

## Detailed Description

Bits for REG\_KEYCNT.

---

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  **1.5.3**

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# Interrupt Flags

## [Memory map bit(fields)]

Bits for REG\_IE, REG\_IF and REG\_IFBIOS. [More...](#)

## Defines

#define	<b>IRQ_VBLANK</b> 0x0001
	<i>Catch VBlank irq.</i>
#define	<b>IRQ_HBLANK</b> 0x0002
	<i>Catch HBlank irq.</i>
#define	<b>IRQ_VCOUNT</b> 0x0004
	<i>Catch VCount irq.</i>
#define	<b>IRQ_TIMER0</b> 0x0008
	<i>Catch timer 0 irq.</i>
#define	<b>IRQ_TIMER1</b> 0x0010
	<i>Catch timer 1 irq.</i>
#define	<b>IRQ_TIMER2</b> 0x0020
	<i>Catch timer 2 irq.</i>
#define	<b>IRQ_TIMER3</b> 0x0040
	<i>Catch timer 3 irq.</i>
#define	<b>IRQ_SERIAL</b> 0x0080
	<i>Catch serial comm irq.</i>
#define	<b>IRQ_DMA0</b> 0x0100
	<i>Catch DMA 0 irq.</i>
#define	<b>IRQ_DMA1</b> 0x0200
	<i>Catch DMA 1 irq.</i>
#define	<b>IRQ_DMA2</b> 0x0400
	<i>Catch DMA 2 irq.</i>
#define	<b>IRQ_DMA3</b> 0x0800
	<i>Catch DMA 3 irq.</i>
#define	<b>IRQ_KEYPAD</b> 0x1000
	<i>Catch key irq.</i>
#define	<b>IRQ_GAMEPAK</b> 0x2000
	<i>Catch cart irq.</i>

---

## Detailed Description

Bits for REG\_IE, REG\_IF and REG\_IFBIOS.

---

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  *1.5.3*

---

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

## **Waitstate Control Flags** **[Memory map bit(fields)]**

Bits for REG\_WAITCNT. [More...](#)

## Defines

#define	<b>WS_SRAM_4</b>	0
#define	<b>WS_SRAM_3</b>	0x0001
#define	<b>WS_SRAM_2</b>	0x0002
#define	<b>WS_SRAM_8</b>	0x0003
#define	<b>WS_ROM0_N4</b>	0
#define	<b>WS_ROM0_N3</b>	0x0004
#define	<b>WS_ROM0_N2</b>	0x0008
#define	<b>WS_ROM0_N8</b>	0x000C
#define	<b>WS_ROM0_S2</b>	0
#define	<b>WS_ROM0_S1</b>	0x0010
#define	<b>WS_ROM1_N4</b>	0
#define	<b>WS_ROM1_N3</b>	0x0020
#define	<b>WS_ROM1_N2</b>	0x0040
#define	<b>WS_ROM1_N8</b>	0x0060
#define	<b>WS_ROM1_S4</b>	0
#define	<b>WS_ROM1_S1</b>	0x0080
#define	<b>WS_ROM2_N4</b>	0
#define	<b>WS_ROM2_N3</b>	0x0100
#define	<b>WS_ROM2_N2</b>	0x0200
#define	<b>WS_ROM2_N8</b>	0x0300
#define	<b>WS_ROM2_S8</b>	0
#define	<b>WS_ROM2_S1</b>	0x0400
#define	<b>WS_PHI_OFF</b>	0
#define	<b>WS_PHI_4</b>	0x0800
#define	<b>WS_PHI_2</b>	0x1000
#define	<b>WS_PHI_1</b>	0x1800
#define	<b>WS_PREFETCH</b>	0x4000
#define	<b>WS_GBA</b>	0
#define	<b>WS_CGB</b>	0x8000
#define	<b>WS_STANDARD</b>	0x4317

---

## **Detailed Description**

Bits for REG\_WAITCNT.

---

# Define Documentation

```
#define WS_SRAM_4 0
```

---

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by  doxygen 1.5.3*

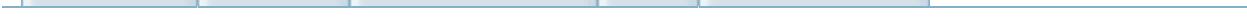
[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **Screen-entry Flags**

## [Memory map bit(fields)]

## Defines

```
#define SE_HFLIP 0x0400
    Horizontal flip.
#define SE_VFLIP 0x0800
    Vertical flip.
#define SE_ID_MASK 0x03FF
#define SE_ID_SHIFT 0
#define SE_ID(n) ((n)<<SE_ID_SHIFT)
#define SE_FLIP_MASK 0x0C00
#define SE_FLIP_SHIFT 10
#define SE_FLIP(n) ((n)<<SE_FLIP_SHIFT)
#define SE_PALBANK_MASK 0xF000
#define SE_PALBANK_SHIFT 12
#define SE_PALBANK(n) ((n)<<SE_PALBANK_SHIFT)
#define SE_BUILD(id, PALBANK, hflip, vflip) ( ((id)&0x03FF) | (((hflip)&1)
    <<10) | (((vflip)&1)<<11) | ((PALBANK)<<12) )
```

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **Object Attribute 0 Flags**

## [Memory map bit(fields)]

## Defines

#define	<b>ATTR0_REG</b> 0	
		<i>Regular object.</i>
#define	<b>ATTR0_AFF</b> 0x0100	
		<i>Affine object.</i>
#define	<b>ATTR0_HIDE</b> 0x0200	
		<i>Inactive object.</i>
#define	<b>ATTR0_AFF_DBL</b> 0x0300	
		<i>Double-size affine object.</i>
#define	<b>ATTR0_AFF_DBL_BIT</b> 0x0200	
#define	<b>ATTR0_BLEND</b> 0x0400	
		<i>Enable blend.</i>
#define	<b>ATTR0_WINDOW</b> 0x0800	
		<i>Use for object window.</i>
#define	<b>ATTR0_MOSAIC</b> 0x1000	
		<i>Enable mosaic.</i>
#define	<b>ATTR0_4BPP</b> 0	
		<i>Use 4bpp (16 color) tiles.</i>
#define	<b>ATTR0_8BPP</b> 0x2000	
		<i>Use 8bpp (256 color) tiles.</i>
#define	<b>ATTR0_SQUARE</b> 0	
		<i>Square shape.</i>
#define	<b>ATTR0_WIDE</b> 0x4000	
		<i>Tall shape (<i>height &gt; width</i>).</i>
#define	<b>ATTR0_TALL</b> 0x8000	
		<i>Wide shape (<i>height &lt; width</i>).</i>
#define	<b>ATTR0_Y_MASK</b> 0x00FF	
#define	<b>ATTR0_Y_SHIFT</b> 0	
#define	<b>ATTR0_Y(n)</b> ((n)<< <b>ATTR0_Y_SHIFT</b> )	
#define	<b>ATTR0_MODE_MASK</b> 0x0300	
#define	<b>ATTR0_MODE_SHIFT</b> 8	
#define	<b>ATTR0_MODE(n)</b> ((n)<< <b>ATTR0_MODE_SHIFT</b> )	
#define	<b>ATTR0_SHAPE_MASK</b> 0xC000	

```
#define ATTR0_SHAPE_SHIFT 14
#define ATTR0_SHAPE(n) ((n)<<ATTR0_SHAPE_SHIFT)
#define ATTR0_BUILD(y, shape, bpp, mode, mos, bld, win)
```

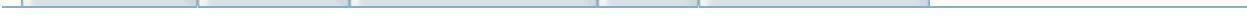
---

# Define Documentation

```
#define ATTR0_BUILD ( y,  
                      shape,  
                      bpp,  
                      mode,  
                      mos,  
                      bld,  
                      win      )
```

## Value:

```
(  
    ((y)&255) | (((mode)&3)<<8) | (((bld)&1)<<16)  
    | (((mos)&1)<<12) | (((bpp)&8)<<10) | (((shape)&1)<<24)  
)
```

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **Object Attribute 1 Flags**

## [Memory map bit(fields)]

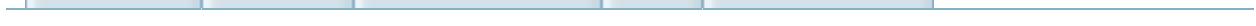
## Defines

#define	<b>ATTR1_HFLIP</b> 0x1000	
		<i>Horizontal flip (reg obj only).</i>
#define	<b>ATTR1_VFLIP</b> 0x2000	
		<i>Vertical flip (reg obj only).</i>
#define	<b>ATTR1_SIZE_8</b> 0	
#define	<b>ATTR1_SIZE_16</b> 0x4000	
#define	<b>ATTR1_SIZE_32</b> 0x8000	
#define	<b>ATTR1_SIZE_64</b> 0xC000	
#define	<b>ATTR1_SIZE_8x8</b> 0	
		<i>Size flag for 8x8 px object.</i>
#define	<b>ATTR1_SIZE_16x16</b> 0x4000	
		<i>Size flag for 16x16 px object.</i>
#define	<b>ATTR1_SIZE_32x32</b> 0x8000	
		<i>Size flag for 32x32 px object.</i>
#define	<b>ATTR1_SIZE_64x64</b> 0xC000	
		<i>Size flag for 64x64 px object.</i>
#define	<b>ATTR1_SIZE_8x16</b> 0	
		<i>Size flag for 8x16 px object.</i>
#define	<b>ATTR1_SIZE_8x32</b> 0x4000	
		<i>Size flag for 8x32 px object.</i>
#define	<b>ATTR1_SIZE_16x32</b> 0x8000	
		<i>Size flag for 16x32 px object.</i>
#define	<b>ATTR1_SIZE_32x64</b> 0xC000	
		<i>Size flag for 32x64 px object.</i>
#define	<b>ATTR1_SIZE_16x8</b> 0	
		<i>Size flag for 16x8 px object.</i>
#define	<b>ATTR1_SIZE_32x8</b> 0x4000	
		<i>Size flag for 32x8 px object.</i>
#define	<b>ATTR1_SIZE_32x16</b> 0x8000	
		<i>Size flag for 32x16 px object.</i>
#define	<b>ATTR1_SIZE_64x32</b> 0xC000	
		<i>Size flag for 64x64 px object.</i>

```
#define ATTR1_X_MASK 0x01FF
#define ATTR1_X_SHIFT 0
#define ATTR1_X(n) ((n)<<ATTR1_X_SHIFT)
#define ATTR1_AFF_ID_MASK 0x3E00
#define ATTR1_AFF_ID_SHIFT 9
#define ATTR1_AFF_ID(n) ((n)<<ATTR1_AFF_ID_SHIFT)
#define ATTR1_FLIP_MASK 0x3000
#define ATTR1_FLIP_SHIFT 12
#define ATTR1_FLIP(n) ((n)<<ATTR1_FLIP_SHIFT)
#define ATTR1_SIZE_MASK 0xC000
#define ATTR1_SIZE_SHIFT 14
#define ATTR1_SIZE(n) ((n)<<ATTR1_SIZE_SHIFT)
#define ATTR1_BUILDR(x, size, hflip, vflip) ( ((x)&511) | (((hflip)&1)<<12) |
((vflip)&1)<<13) | (((size)&3)<<14) )
#define ATTR1_BUILD(x, size, affid) ( ((x)&511) | (((affid)&31)<<9) |
((size)&3)<<14) )
```

---

Generated on Mon Aug 25 17:03:57 2008 for libtonc by  1.5.3

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

## **Object Attribute 2 Flags**

### **[Memory map bit(fields)]**

## Defines

```
#define ATTR2_ID_MASK 0x03FF
#define ATTR2_ID_SHIFT 0
#define ATTR2_ID(n) ((n)<<ATTR2_ID_SHIFT)
#define ATTR2_PRIO_MASK 0x0C00
#define ATTR2_PRIO_SHIFT 10
#define ATTR2_PRIO(n) ((n)<<ATTR2_PRIO_SHIFT)
#define ATTR2_PALBANK_MASK 0xF000
#define ATTR2_PALBANK_SHIFT 12
#define ATTR2_PALBANK(n) ((n)<<ATTR2_PALBANK_SHIFT)
#define ATTR2_BUILD(id, pb, prio) ( ((id)&0x3FF) | (((pb)&15)<<12) |
    (((prio)&3)<<10) )
```

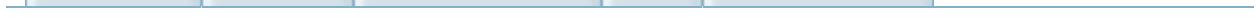
---

# Define Documentation

```
#define ATTR2_ID_MASK 0x03FF
```

---

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by  doxygen 1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

## **Memory mapped arrays**

### **[[Memory Map](#)]**

These are some macros for easier access of various memory sections. They're all arrays or matrices, using the types that would be the most natural for that concept. [More...](#)

## Palette

#define	<b>pal_bg_mem</b> (( <b>COLOR</b> *)MEM_PAL) <i>Background palette.</i>
#define	<b>pal_obj_mem</b> (( <b>COLOR</b> *)MEM_PAL_OBJ) <i>Object palette.</i>
#define	<b>pal_bg_bank</b> (( <b>PALBANK</b> *)MEM_PAL) <i>Background palette matrix.</i>
#define	<b>pal_obj_bank</b> (( <b>PALBANK</b> *)MEM_PAL_OBJ) <i>Object palette matrix.</i>

# VRAM

#define	<b>tile_mem</b> ((CHARBLOCK*)MEM_VRAM)
	<i>Charblocks, 4bpp tiles.</i>
#define	<b>tile8_mem</b> ((CHARBLOCK8*)MEM_VRAM)
	<i>Charblocks, 8bpp tiles.</i>
#define	<b>tile_mem_obj</b> ((CHARBLOCK*)MEM_VRAM_OBJ)
	<i>Object charblocks, 4bpp tiles.</i>
#define	<b>tile8_mem_obj</b> ((CHARBLOCK8*)MEM_VRAM_OBJ)
	<i>Object charblocks, 4bpp tiles.</i>
#define	<b>se_mem</b> ((SCREENBLOCK*)MEM_VRAM)
	<i>Screenblocks as arrays.</i>
#define	<b>se_mat</b> ((SCREENMAT*)MEM_VRAM)
	<i>Screenblock as matrices.</i>
#define	<b>vid_mem</b> ((COLOR*)MEM_VRAM)
	<i>Main mode 3/5 frame as an array.</i>
#define	<b>m3_mem</b> ((M3LINE*)MEM_VRAM)
	<i>Mode 3 frame as a matrix.</i>
#define	<b>m4_mem</b> ((M4LINE*)MEM_VRAM)
	<i>Mode 4 first page as a matrix.</i>
#define	<b>m5_mem</b> ((M5LINE*)MEM_VRAM)
	<i>Mode 5 first page as a matrix.</i>
#define	<b>vid_mem_front</b> ((COLOR*)MEM_VRAM)
	<i>First page array.</i>
#define	<b>vid_mem_back</b> ((COLOR*)MEM_VRAM_BACK)
	<i>Second page array.</i>
#define	<b>m4_mem_back</b> ((M4LINE*)MEM_VRAM_BACK)
	<i>Mode 4 second page as a matrix.</i>
#define	<b>m5_mem_back</b> ((M5LINE*)MEM_VRAM_BACK)
	<i>Mode 5 second page as a matrix.</i>

## OAM

#define	<b>oam_mem</b> ((OBJ_ATTR*)MEM_OAM)
	<i>Object attribute memory.</i>
#define	<b>obj_mem</b> ((OBJ_ATTR*)MEM_OAM)
	<i>Object attribute memory.</i>
#define	<b>obj_aff_mem</b> ((OBJ_AFFINE*)MEM_OAM)
	<i>Object affine memory.</i>

## ROM

```
#define rom_mem ((u16*)MEM_ROM)  
ROM pointer.
```

## SRAM

```
#define sram_mem ((u8*)MEM_SRAM)  
SRAM pointer.
```

---

## **Detailed Description**

These are some macros for easier access of various memory sections. They're all arrays or matrices, using the types that would be the most natural for that concept.

---

## Define Documentation

```
#define m3_mem ((M3LINE*)MEM_VRAM)
```

Mode 3 frame as a matrix.

m3\_mem[y][x] = pixel (x, y) ( COLOR )

```
#define m4_mem ((M4LINE*)MEM_VRAM)
```

Mode 4 first page as a matrix.

m4\_mem[y][x] = pixel (x, y) ( u8 )

**Note:**

This is a byte-buffer. Not to be used for writing.

```
#define m4_mem_back ((M4LINE*)MEM_VRAM_BACK)
```

Mode 4 second page as a matrix.

m4\_mem[y][x] = pixel (x, y) ( u8 )

**Note:**

This is a byte-buffer. Not to be used for writing.

```
#define m5_mem ((M5LINE*)MEM_VRAM)
```

Mode 5 first page as a matrix.

```
m5_mem[y][x] = pixel (x, y) ( COLOR )
```

```
#define m5_mem_back ((M5LINE*)MEM_VRAM_BACK)
```

Mode 5 second page as a matrix.

```
m5_mem[y][x] = pixel (x, y) ( COLOR )
```

```
#define oam_mem ((OBJ_ATTR*)MEM_OAM)
```

Object attribute memory.

```
oam_mem[i] = object i ( OBJ_ATTR )
```

```
#define obj_aff_mem ((OBJ_AFFINE*)MEM_OAM)
```

Object affine memory.

```
obj_aff_mem[i] = object matrix i ( OBJ_AFFINE )
```

```
#define obj_mem ((OBJ_ATTR*)MEM_OAM)
```

Object attribute memory.

```
oam_mem[i] = object i ( OBJ_ATTR )
```

```
#define pal_bg_bank ((PALBANK*)MEM_PAL)
```

Background palette matrix.

```
pal_bg_bank[y] = bank y ( COLOR[ ] )
pal_bg_bank[y][x] = color color y*16+x ( COLOR )
```

```
#define pal_bg_mem ((COLOR*)MEM_PAL)
```

Background palette.

```
pal_bg_mem[i] = color i ( COLOR )
```

```
#define pal_obj_bank ((PALBANK*)MEM_PAL_OBJ)
```

Object palette matrix.

```
pal_obj_bank[y] = bank y ( COLOR[ ] )
pal_obj_bank[y][x] = color y*16+x ( COLOR )
```

```
#define pal_obj_mem ((COLOR*)MEM_PAL_OBJ)
```

Object palette.

```
pal_obj_mem[i] = color i ( COLOR )
```

```
#define se_mat ((SCREENMAT*)MEM_VRAM)
```

Screenblock as matrices.

```
se_mat[s] = screenblock s ( SCR_ENTRY[ ][ ] )
```

```
se_mat[s][y][x] = screenblock s, entry (x,y) ( SCR_ENTRY )
```

```
#define se_mem ((SCREENBLOCK*)MEM_VRAM)
```

Screenblocks as arrays.

```
se_mem[y] = screenblock y ( SCR_ENTRY[ ] )  
se_mem[y][x] = screenblock y, entry x ( SCR_ENTRY )
```

```
#define tile8_mem ((CHARBLOCK8*)MEM_VRAM)
```

Charblocks, 8bpp tiles.

```
tile_mem[y] = charblock y ( TILE[ ] )  
tile_mem[y][x] = block y, tile x ( TILE )
```

```
#define tile8_mem_obj ((CHARBLOCK8*)MEM_VRAM_OBJ)
```

Object charblocks, 4bpp tiles.

```
tile_mem[y] = charblock y ( TILE[ ] )  
tile_mem[y][x] = block y, tile x ( TILE )
```

```
#define tile_mem ((CHARBLOCK*)MEM_VRAM)
```

Charblocks, 4bpp tiles.

```
tile_mem[y] = charblock y ( TILE[ ] )  
tile_mem[y][x] = block y, tile x ( TILE )
```

```
#define tile_mem_obj ( (CHARBLOCK*)MEM_VRAM_OBJ)
```

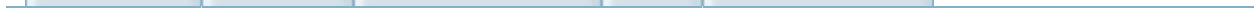
Object charblocks, 4bpp tiles.

```
tile_mem[y] = charblock y ( TILE[ ] )
tile_mem[y][x] = block y, tile x ( TILE )
```

```
#define vid_mem ((COLOR*)MEM_VRAM)
```

Main mode 3/5 frame as an array.

```
vid_mem[i] = pixel i ( COLOR )
```

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# IO Registers

## [Memory Map]

## IWRAM 'registers'

#define	<b>REG_IFBIOS</b> *(vu16*)(REG_BASE-0x0008)
	<i>IRQ ack for IntrWait functions.</i>
#define	<b>REG_RESET_DST</b> *(vu16*)(REG_BASE-0x0006)
	<i>Destination for after SoftReset.</i>
#define	<b>REG_ISR_MAIN</b> *(fnptr*)(REG_BASE-0x0004)
	<i>IRQ handler address.</i>

## Display registers

#define	<b>REG_DISPCTL</b> *(vu32*)(REG_BASE+0x0000)
	<i>Display control.</i>
#define	<b>REG_DISPSTAT</b> *(vu16*)(REG_BASE+0x0004)
	<i>Display status.</i>
#define	<b>REG_VCOUNT</b> *(vu16*)(REG_BASE+0x0006)
	<i>Scanline count.</i>

## Background control registers

#define	<b>REG_BGCNT</b> ((vu16*)(REG_BASE+0x0008)) <i>Bg control array.</i>
#define	<b>REG_BG0CNT</b> *(vu16*)(REG_BASE+0x0008) <i>Bg0 control.</i>
#define	<b>REG_BG1CNT</b> *(vu16*)(REG_BASE+0x000A) <i>Bg1 control.</i>
#define	<b>REG_BG2CNT</b> *(vu16*)(REG_BASE+0x000C) <i>Bg2 control.</i>
#define	<b>REG_BG3CNT</b> *(vu16*)(REG_BASE+0x000E) <i>Bg3 control.</i>

## Regular background scroll registers. (write only!)

#define	<b>REG_BG_OFS</b> (( <b>BG_POINT</b> *)((REG_BASE+0x0010))) <i>Bg scroll array.</i>
#define	<b>REG_BG0HOFS</b> *(vu16*)(REG_BASE+0x0010) <i>Bg0 horizontal scroll.</i>
#define	<b>REG_BG0VOFS</b> *(vu16*)(REG_BASE+0x0012) <i>Bg0 vertical scroll.</i>
#define	<b>REG_BG1HOFS</b> *(vu16*)(REG_BASE+0x0014) <i>Bg1 horizontal scroll.</i>
#define	<b>REG_BG1VOFS</b> *(vu16*)(REG_BASE+0x0016) <i>Bg1 vertical scroll.</i>
#define	<b>REG_BG2HOFS</b> *(vu16*)(REG_BASE+0x0018) <i>Bg2 horizontal scroll.</i>
#define	<b>REG_BG2VOFS</b> *(vu16*)(REG_BASE+0x001A) <i>Bg2 vertical scroll.</i>
#define	<b>REG_BG3HOFS</b> *(vu16*)(REG_BASE+0x001C) <i>Bg3 horizontal scroll.</i>
#define	<b>REG_BG3VOFS</b> *(vu16*)(REG_BASE+0x001E) <i>Bg3 vertical scroll.</i>

## Affine background parameters. (write only!)

#define	<b>REG_BG_AFFINE</b> (( <b>BG_AFFINE</b> *)((REG_BASE+0x0000))) <i>Bg affine array.</i>
#define	<b>REG_BG2PA</b> *(vs16*)(REG_BASE+0x0020) <i>Bg2 matrix.pa.</i>
#define	<b>REG_BG2PB</b> *(vs16*)(REG_BASE+0x0022) <i>Bg2 matrix.pb.</i>
#define	<b>REG_BG2PC</b> *(vs16*)(REG_BASE+0x0024) <i>Bg2 matrix.pc.</i>
#define	<b>REG_BG2PD</b> *(vs16*)(REG_BASE+0x0026) <i>Bg2 matrix.pd.</i>
#define	<b>REG_BG2X</b> *(vs32*)(REG_BASE+0x0028) <i>Bg2 x scroll.</i>
#define	<b>REG_BG2Y</b> *(vs32*)(REG_BASE+0x002C) <i>Bg2 y scroll.</i>
#define	<b>REG_BG3PA</b> *(vs16*)(REG_BASE+0x0030) <i>Bg3 matrix.pa.</i>
#define	<b>REG_BG3PB</b> *(vs16*)(REG_BASE+0x0032) <i>Bg3 matrix.pb.</i>
#define	<b>REG_BG3PC</b> *(vs16*)(REG_BASE+0x0034) <i>Bg3 matrix.pc.</i>
#define	<b>REG_BG3PD</b> *(vs16*)(REG_BASE+0x0036) <i>Bg3 matrix.pd.</i>
#define	<b>REG_BG3X</b> *(vs32*)(REG_BASE+0x0038) <i>Bg3 x scroll.</i>
#define	<b>REG_BG3Y</b> *(vs32*)(REG_BASE+0x003C) <i>Bg3 y scroll.</i>

## Windowing registers

#define	<b>REG_WIN0H</b> *(vu16*)(REG_BASE+0x0040) <i>win0 right, left (0xLLRR)</i>
#define	<b>REG_WIN1H</b> *(vu16*)(REG_BASE+0x0042) <i>win1 right, left (0xLLRR)</i>
#define	<b>REG_WIN0V</b> *(vu16*)(REG_BASE+0x0044) <i>win0 bottom, top (0xTTBB)</i>
#define	<b>REG_WIN1V</b> *(vu16*)(REG_BASE+0x0046) <i>win1 bottom, top (0xTTBB)</i>
#define	<b>REG_WININ</b> *(vu16*)(REG_BASE+0x0048) <i>win0, win1 control</i>
#define	<b>REG_WINOUT</b> *(vu16*)(REG_BASE+0x004A) <i>winOut, winObj control</i>

## Alternate Windowing registers

#define	<b>REG_WIN0R</b> *(vu8*)(REG_BASE+0x0040) <i>Win 0 right.</i>
#define	<b>REG_WIN0L</b> *(vu8*)(REG_BASE+0x0041) <i>Win 0 left.</i>
#define	<b>REG_WIN1R</b> *(vu8*)(REG_BASE+0x0042) <i>Win 1 right.</i>
#define	<b>REG_WIN1L</b> *(vu8*)(REG_BASE+0x0043) <i>Win 1 left.</i>
#define	<b>REG_WIN0B</b> *(vu8*)(REG_BASE+0x0044) <i>Win 0 bottom.</i>
#define	<b>REG_WIN0T</b> *(vu8*)(REG_BASE+0x0045) <i>Win 0 top.</i>
#define	<b>REG_WIN1B</b> *(vu8*)(REG_BASE+0x0046) <i>Win 1 bottom.</i>
#define	<b>REG_WIN1T</b> *(vu8*)(REG_BASE+0x0047) <i>Win 1 top.</i>
#define	<b>REG_WIN0CNT</b> *(vu8*)(REG_BASE+0x0048) <i>window 0 control</i>
#define	<b>REG_WIN1CNT</b> *(vu8*)(REG_BASE+0x0049) <i>window 1 control</i>
#define	<b>REG_WINOUTCNT</b> *(vu8*)(REG_BASE+0x004A) <i>Out window control.</i>
#define	<b>REG_WINOBJCNT</b> *(vu8*)(REG_BASE+0x004B) <i>Obj window control.</i>

## Graphic effects

#define	<b>REG_MOSAIC</b> *(vu32*)(REG_BASE+0x004C)	
		<i>Mosaic control.</i>
#define	<b>REG_BLDCNT</b> *(vu16*)(REG_BASE+0x0050)	
		<i>Alpha control.</i>
#define	<b>REG_BLDALPHA</b> *(vu16*)(REG_BASE+0x0052)	
		<i>Fade level.</i>
#define	<b>REG_BLDY</b> *(vu16*)(REG_BASE+0x0054)	
		<i>Blend levels.</i>

## Channel 1: Square wave with sweep

#define	<b>REG_SND1SWEEP</b> *(vu16*)(REG_BASE+0x0060) <i>Channel 1 Sweep.</i>
#define	<b>REG_SND1CNT</b> *(vu16*)(REG_BASE+0x0062) <i>Channel 1 Control.</i>
#define	<b>REG_SND1FREQ</b> *(vu16*)(REG_BASE+0x0064) <i>Channel 1 frequency.</i>

## Channel 2: Simple square wave

```
#define REG_SND2CNT *(vu16*)(REG_BASE+0x0068)  
    Channel 2 control.  
#define REG_SND2FREQ *(vu16*)(REG_BASE+0x006C)  
    Channel 2 frequency.
```

## Channel 3: Wave player

#define	<b>REG_SND3SEL</b> *(vu16*)(REG_BASE+0x0070)
	<i>Channel 3 wave select.</i>
#define	<b>REG_SND3CNT</b> *(vu16*)(REG_BASE+0x0072)
	<i>Channel 3 control.</i>
#define	<b>REG_SND3FREQ</b> *(vu16*)(REG_BASE+0x0074)
	<i>Channel 3 frequency.</i>

## Channel 4: Noise generator

#define	<b>REG_SND4CNT</b> *(vu16*)(REG_BASE+0x0078) <i>Channel 4 control.</i>
#define	<b>REG_SND4FREQ</b> *(vu16*)(REG_BASE+0x007C) <i>Channel 4 frequency.</i>

## Sound control

#define	<b>REG_SNDCNT</b> *(vu32*)(REG_BASE+0x0080)
	<i>Main sound control.</i>
#define	<b>REG_SNDDMGCNT</b> *(vu16*)(REG_BASE+0x0080)
	<i>DMG channel control.</i>
#define	<b>REG_SNDDSCNT</b> *(vu16*)(REG_BASE+0x0082)
	<i>Direct Sound control.</i>
#define	<b>REG SNDSTAT</b> *(vu16*)(REG_BASE+0x0084)
	<i>Sound status.</i>
#define	<b>REG_SNDBIAS</b> *(vu16*)(REG_BASE+0x0088)
	<i>Sound bias.</i>

## Sound buffers

#define	<b>REG_WAVE_RAM</b> (vu32*)(REG_BASE+0x0090) <i>Channel 3 wave buffer.</i>
#define	<b>REG_WAVE_RAM0</b> *(vu32*)(REG_BASE+0x0090) <i>Channel 3 wave buffer.</i>
#define	<b>REG_WAVE_RAM1</b> *(vu32*)(REG_BASE+0x0094) <i>Channel 3 wave buffer.</i>
#define	<b>REG_WAVE_RAM2</b> *(vu32*)(REG_BASE+0x0098) <i>Channel 3 wave buffer.</i>
#define	<b>REG_WAVE_RAM3</b> *(vu32*)(REG_BASE+0x009C) <i>Channel 3 wave buffer.</i>
#define	<b>REG_FIFO_A</b> *(vu32*)(REG_BASE+0x00A0) <i>DSound A FIFO.</i>
#define	<b>REG_FIFO_B</b> *(vu32*)(REG_BASE+0x00A4) <i>DSound B FIFO.</i>

## DMA registers

#define	<b>REG_DMA</b> ((volatile <b>DMA_REC</b> *)((REG_BASE+0x00B0)) <i>DMA as DMA_REC array.</i>
#define	<b>REG_DMA0SAD</b> *(vu32*)(REG_BASE+0x00B0) <i>DMA 0 Source address.</i>
#define	<b>REG_DMA0DAD</b> *(vu32*)(REG_BASE+0x00B4) <i>DMA 0 Destination address.</i>
#define	<b>REG_DMA0CNT</b> *(vu32*)(REG_BASE+0x00B8) <i>DMA 0 Control.</i>
#define	<b>REG_DMA1SAD</b> *(vu32*)(REG_BASE+0x00BC) <i>DMA 1 Source address.</i>
#define	<b>REG_DMA1DAD</b> *(vu32*)(REG_BASE+0x00C0) <i>DMA 1 Destination address.</i>
#define	<b>REG_DMA1CNT</b> *(vu32*)(REG_BASE+0x00C4) <i>DMA 1 Control.</i>
#define	<b>REG_DMA2SAD</b> *(vu32*)(REG_BASE+0x00C8) <i>DMA 2 Source address.</i>
#define	<b>REG_DMA2DAD</b> *(vu32*)(REG_BASE+0x00CC) <i>DMA 2 Destination address.</i>
#define	<b>REG_DMA2CNT</b> *(vu32*)(REG_BASE+0x00D0) <i>DMA 2 Control.</i>
#define	<b>REG_DMA3SAD</b> *(vu32*)(REG_BASE+0x00D4) <i>DMA 3 Source address.</i>
#define	<b>REG_DMA3DAD</b> *(vu32*)(REG_BASE+0x00D8) <i>DMA 3 Destination address.</i>
#define	<b>REG_DMA3CNT</b> *(vu32*)(REG_BASE+0x00DC) <i>DMA 3 Control.</i>

## Timer registers

#define	<b>REG_TM</b> ((volatile <b>TMR_REC</b> *)((REG_BASE+0x0100))) <i>Timers as <b>TMR_REC</b> array.</i>
#define	<b>REG_TM0D</b> *(vu16*)((REG_BASE+0x0100)) <i>Timer 0 data.</i>
#define	<b>REG_TM0CNT</b> *(vu16*)((REG_BASE+0x0102)) <i>Timer 0 control.</i>
#define	<b>REG_TM1D</b> *(vu16*)((REG_BASE+0x0104)) <i>Timer 1 data.</i>
#define	<b>REG_TM1CNT</b> *(vu16*)((REG_BASE+0x0106)) <i>Timer 1 control.</i>
#define	<b>REG_TM2D</b> *(vu16*)((REG_BASE+0x0108)) <i>Timer 2 data.</i>
#define	<b>REG_TM2CNT</b> *(vu16*)((REG_BASE+0x010A)) <i>Timer 2 control.</i>
#define	<b>REG_TM3D</b> *(vu16*)((REG_BASE+0x010C)) <i>Timer 3 data.</i>
#define	<b>REG_TM3CNT</b> *(vu16*)((REG_BASE+0x010E)) <i>Timer 3 control.</i>

## Serial communication

#define	<b>REG_SIOCNT</b> *(vu16*)(REG_BASE+0x0128) <i>Serial IO control (Normal/MP/UART).</i>
#define	<b>REG_SIODATA</b> ((vu32*)(REG_BASE+0x0120)) <i>Serial IO control (Normal/MP/UART).</i>
#define	<b>REG_SIODATA32</b> *(vu32*)(REG_BASE+0x0120) <i>Normal/UART 32bit data.</i>
#define	<b>REG_SIODATA8</b> *(vu16*)(REG_BASE+0x012A) <i>Normal/UART 8bit data.</i>
#define	<b>REG_SIOMULTI</b> ((vu16*)(REG_BASE+0x0120)) <i>Multiplayer data array.</i>
#define	<b>REG_SIOMULTIO</b> *(vu16*)(REG_BASE+0x0120) <i>MP master data.</i>
#define	<b>REG_SIOMULTI1</b> *(vu16*)(REG_BASE+0x0122) <i>MP Slave 1 data.</i>
#define	<b>REG_SIOMULTI2</b> *(vu16*)(REG_BASE+0x0124) <i>MP Slave 2 data.</i>
#define	<b>REG_SIOMULTI3</b> *(vu16*)(REG_BASE+0x0126) <i>MP Slave 3 data.</i>
#define	<b>REG_SIOMLT_RECV</b> *(vu16*)(REG_BASE+0x0120) <i>MP data receiver.</i>
#define	<b>REG_SIOMLT_SEND</b> *(vu16*)(REG_BASE+0x012A) <i>MP data sender.</i>

## Keypad registers

#define	<b>REG_KEYINPUT</b> *(vu16*)(REG_BASE+0x0130)
	<i>Key status (read only??).</i>
#define	<b>REG_KEYCNT</b> *(vu16*)(REG_BASE+0x0132)
	<i>Key IRQ control.</i>

## Joybus communication

#define	<b>REG_RCNT</b> *(vu16*)(REG_BASE+0x0134)
	<i>SIO Mode Select/General Purpose Data.</i>
#define	<b>REG_JOYCNT</b> *(vu16*)(REG_BASE+0x0140)
	<i>JOY bus control.</i>
#define	<b>REG_JOY_RECV</b> *(vu32*)(REG_BASE+0x0150)
	<i>JOY bus receiver.</i>
#define	<b>REG_JOY_TRANS</b> *(vu32*)(REG_BASE+0x0154)
	<i>JOY bus transmitter.</i>
#define	<b>REG_JOYSTAT</b> *(vu16*)(REG_BASE+0x0158)
	<i>JOY bus status.</i>

## Interrupt / System registers

#define	<b>REG_IE</b> *(vu16*)(REG_BASE+0x0200) <i>IRQ enable.</i>
#define	<b>REG_IF</b> *(vu16*)(REG_BASE+0x0202) <i>IRQ status/acknowledge.</i>
#define	<b>REG_WAITCNT</b> *(vu16*)(REG_BASE+0x0204) <i>Waitstate control.</i>
#define	<b>REG_IME</b> *(vu16*)(REG_BASE+0x0208) <i>IRQ master enable.</i>
#define	<b>REG_PAUSE</b> *(vu16*)(REG_BASE+0x0300) <i>Pause system (?).</i>

---

## Detailed Description

---

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  **doxygen** 1.5.3

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

## **IO Alternates** **[Memory Map]**

Alternate names for some of the registers. [More...](#)

## Defines

```
#define REG_BLDMOD *(vu16*)(REG_BASE+0x0050)
#define REG_COLEV *(vu16*)(REG_BASE+0x0052)
#define REG_COLEY *(vu16*)(REG_BASE+0x0054)
#define REG_SOUND1CNT *(vu32*)(REG_BASE+0x0060)
#define REG_SOUND1CNT_L *(vu16*)(REG_BASE+0x0060)
#define REG_SOUND1CNT_H *(vu16*)(REG_BASE+0x0062)
#define REG_SOUND1CNT_X *(vu16*)(REG_BASE+0x0064)
#define REG_SOUND2CNT_L *(vu16*)(REG_BASE+0x0068)
#define REG_SOUND2CNT_H *(vu16*)(REG_BASE+0x006C)
#define REG_SOUND3CNT *(vu32*)(REG_BASE+0x0070)
#define REG_SOUND3CNT_L *(vu16*)(REG_BASE+0x0070)
#define REG_SOUND3CNT_H *(vu16*)(REG_BASE+0x0072)
#define REG_SOUND3CNT_X *(vu16*)(REG_BASE+0x0074)
#define REG_SOUND4CNT_L *(vu16*)(REG_BASE+0x0078)
#define REG_SOUND4CNT_H *(vu16*)(REG_BASE+0x007C)
#define REG_SOUNDCNT *(vu32*)(REG_BASE+0x0080)
#define REG_SOUNDCNT_L *(vu16*)(REG_BASE+0x0080)
#define REG_SOUNDCNT_H *(vu16*)(REG_BASE+0x0082)
#define REG_SOUNDCNT_X *(vu16*)(REG_BASE+0x0084)
#define REG_SOUNDBIAS *(vu16*)(REG_BASE+0x0088)
#define REG_WAVE (vu32*)(REG_BASE+0x0090)
#define REG_FIFOA *(vu32*)(REG_BASE+0x00A0)
#define REG_FIFOB *(vu32*)(REG_BASE+0x00A4)
#define REG_DMA0CNT_L *(vu16*)(REG_BASE+0x00B8)
#define REG_DMA0CNT_H *(vu16*)(REG_BASE+0x00BA)
#define REG_DMA1CNT_L *(vu16*)(REG_BASE+0x00C4)
#define REG_DMA1CNT_H *(vu16*)(REG_BASE+0x00C6)
#define REG_DMA2CNT_L *(vu16*)(REG_BASE+0x00D0)
#define REG_DMA2CNT_H *(vu16*)(REG_BASE+0x00D2)
#define REG_DMA3CNT_L *(vu16*)(REG_BASE+0x00DC)
#define REG_DMA3CNT_H *(vu16*)(REG_BASE+0x00DE)
#define REG_TM0CNT_L *(vu16*)(REG_BASE+0x0100)
```

```
#define REG_TM0CNT_H *(vu16*)(REG_BASE+0x0102)
#define REG_TM1CNT_L *(vu16*)(REG_BASE+0x0104)
#define REG_TM1CNT_H *(vu16*)(REG_BASE+0x0106)
#define REG_TM2CNT_L *(vu16*)(REG_BASE+0x0108)
#define REG_TM2CNT_H *(vu16*)(REG_BASE+0x010a)
#define REG_TM3CNT_L *(vu16*)(REG_BASE+0x010c)
#define REG_TM3CNT_H *(vu16*)(REG_BASE+0x010e)
#define REG_KEYS *(vu16*)(REG_BASE+0x0130)
#define REG_P1 *(vu16*)(REG_BASE+0x0130)
#define REG_P1CNT *(vu16*)(REG_BASE+0x0132)
#define REG_SCD0 *(vu16*)(REG_BASE+0x0120)
#define REG_SCD1 *(vu16*)(REG_BASE+0x0122)
#define REG_SCD2 *(vu16*)(REG_BASE+0x0124)
#define REG_SCD3 *(vu16*)(REG_BASE+0x0126)
#define REG_SCCNT *(vu32*)(REG_BASE+0x0128)
#define REG_SCCNT_L *(vu16*)(REG_BASE+0x0128)
#define REG_SCCNT_H *(vu16*)(REG_BASE+0x012A)
#define REG_R *(vu16*)(REG_BASE+0x0134)
#define REG_HS_CTRL *(vu16*)(REG_BASE+0x0140)
#define REG_JOYRE *(vu32*)(REG_BASE+0x0150)
#define REG_JOYRE_L *(vu16*)(REG_BASE+0x0150)
#define REG_JOYRE_H *(vu16*)(REG_BASE+0x0152)
#define REG_JOYTR *(vu32*)(REG_BASE+0x0154)
#define REG_JOYTR_L *(vu16*)(REG_BASE+0x0154)
#define REG_JOYTR_H *(vu16*)(REG_BASE+0x0156)
#define REG_JSTAT *(vu16*)(REG_BASE+0x0158)
#define REG_WSCNT *(vu16*)(REG_BASE+0x0204)
```

---

## **Detailed Description**

Alternate names for some of the registers.

---

# Define Documentation

```
#define REG_BLDMOD *(vu16*)(REG_BASE+0x0050)
```

---

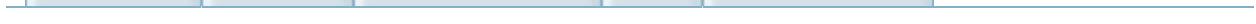
*Generated on Mon Aug 25 17:03:57 2008 for libtonc by doxygen 1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# Sound

---

Generated on Mon Aug 25 17:03:56 2008 for libtonc by  doxygen 1.5.3

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# Tonc Text Engine

A generalized raster text system. [More...](#)

# Modules

<b>Operations</b>
<i>Basic operations.</i>
<b>Attributes</b>
<i>Basic getters and setters.</i>
<b>Console IO</b>
<i>Stdio functionality.</i>
<b>Tilemap text</b>
<i>Text for regular and affine tilemaps.</i>
<b>Character text, column-major</b>
<i>Text on surface composed of 4bpp tiles, mapped in column-major order.</i>
<b>Character text, row-major</b>
<i>Text on surface composed of 4bpp tiles, mapped in row-major order.</i>
<b>Bitmap text</b>
<i>Text for 16bpp and 8bpp bitmap surfaces: modes 3, 4 and 5.</i>
<b>Object text</b>
<i>Text using objects.</i>

# Data Structures

struct	<b>TFont</b> <i>Font description struct.</i> <a href="#">More...</a>
struct	<b>TTC</b> <i>TTE context struct.</i> <a href="#">More...</a>

## Internal fonts

const <b>TFont</b>	<b>sys8Font</b> <i>System font '-127. FWF 8x 8@1.</i>
const <b>TFont</b>	<b>verdana9Font</b> <i>Verdana 9 '-'-'?'. VWF 8x12@1.</i>
const <b>TFont</b>	<b>verdana9bFont</b> <i>Verdana 9 bold '-'-'?'. VWF 8x12@1.</i>
const <b>TFont</b>	<b>verdana9iFont</b> <i>Verdana 9 italic '-'-'?'. VWF 8x12@1.</i>
const <b>TFont</b>	<b>verdana10Font</b> <i>Verdana 10 '-'-'?'. VWF 16x14@1.</i>
const <b>TFont</b>	<b>verdana9_b4Font</b> <i>Verdana 9 '-'-'?'. VWF 8x12@4.</i>
const unsigned int	<b>sys8Glyphs</b> [192] <i>System font '-127. FWF 8x 8@1.</i>
const unsigned int	<b>verdana9Glyphs</b> [896] <i>System font '-127. FWF 8x 8@1.</i>
const unsigned char	<b>verdana9Widths</b> [224] <i>System font '-127. FWF 8x 8@1.</i>
const unsigned int	<b>verdana9bGlyphs</b> [896] <i>System font '-127. FWF 8x 8@1.</i>
const unsigned char	<b>verdana9bWidths</b> [224] <i>System font '-127. FWF 8x 8@1.</i>
const unsigned int	<b>verdana9iGlyphs</b> [896] <i>System font '-127. FWF 8x 8@1.</i>
const unsigned char	<b>verdana9iWidths</b> [224] <i>System font '-127. FWF 8x 8@1.</i>
const unsigned int	<b>verdana10Glyphs</b> [1792] <i>System font '-127. FWF 8x 8@1.</i>
const unsigned char	<b>verdana10Widths</b> [224] <i>System font '-127. FWF 8x 8@1.</i>
const unsigned int	<b>verdana9_b4Glyphs</b> [3584]

	<i>System font '-127. FWF 8x 8@1.</i>
const unsigned char	<b>verdana9_b4Widths</b> [224] <i>System font '-127. FWF 8x 8@1.</i>

## Color lut indices

```
#define TTE_INK 0
#define TTE_SHADOW 1
#define TTE_PAPER 2
#define TTE_SPECIAL 3
```

## drawg helper macros

#define	<b>TTE_BASE_VARS</b> (_tc, _font)
<i>Declare and define base drawg variables.</i>	
#define	<b>TTE_CHAR_VARS</b> (font, gid, src_t, _sD, _sL, _chW, _chH)
<i>Declare and define basic source drawg variables.</i>	
#define	<b>TTE_DST_VARS</b> (tc, dst_t, _dD, _dL, _dP, _x, _y)
<i>Declare and define basic destination drawg variables.</i>	

## Default fonts

#define	<b>fwf_default sys8Font</b>
	<i>Default fixed-width font.</i>
#define	<b>vwf_default verdana9Font</b>
	<i>Default variable-width font.</i>

## Default glyph renderers

```
#define ase_drawg_default ((fnDrawg)ase_drawg_s)
#define bmp8_drawg_default ((fnDrawg)bmp8_drawg_b1cts)
#define bmp16_drawg_default ((fnDrawg)bmp16_drawg_b1cts)
#define chr4c_drawg_default ((fnDrawg)chr4c_drawg_b1cts)
#define chr4r_drawg_default ((fnDrawg)chr4r_drawg_b1cts)
#define obj_drawg_default ((fnDrawg)obj_drawg)
#define se_drawg_default ((fnDrawg)se_drawg_s)
```

## Default initializers

```
#define tte_init_se_default(bgnr, bgcnt) tte_init_se( bgnr, bgcnt, 0xF000,  
CLR_YELLOW, 0, &fwf_default, NULL)  
#define tte_init_ase_default(bgnr, bgcnt) tte_init_ase(bgnr, bgcnt, 0x0000,  
CLR_YELLOW, 0, &fwf_default, NULL)  
#define tte_init_chr4c_default(bgnr, bgcnt)  
#define tte_init_chr4r_default(bgnr, bgcnt)  
#define tte_init_chr4c_b4_default(bgnr, bgcnt)  
#define tte_init bmp _default(mode) tte_init bmp(mode, &v wf _ default,  
NULL)  
#define tte_init_obj _ default(pObj) tte_init_obj(pObj, 0, 0, 0xF000,  
CLR_YELLOW, 0, &fwf_default, NULL)
```

## Defines

```
#define TTE_TAB_WIDTH 24
```

## Typedefs

typedef void(*	<b>fnDrawg</b> )(uint gid) <i>Glyph render function format.</i>
typedef void(*	<b>fnErase</b> )(int left, int top, int right, int bottom) <i>Erase rectangle function format.</i>

# Variables

TTC \* gp\_tte\_context

---

## Detailed Description

A generalized raster text system.

As of v1.3, Tonc has a completely new way of handling text. It can handle (practically) all modes, VRAM types and font sizes and brings them together under a unified interface. It uses function pointers to store *drawg* and *erase* functions of each rendering family. The families currently supported are:

- **ase**: Affine screen entries (Affine tiled BG)
- **bmp8**: 8bpp bitmaps (Mode 4)
- **bmp16** 16bpp bitmaps (Mode 3/5)
- **chr4c** 4bpp characters, column-major (Regular tiled BG)
- **chr4r** 4bpp characters, row-major (Regular tiled BG)
- **obj** Objects
- **se** Regular screen entries (Regular tiled BG)

Each of these consists of an initializer, `tte_init_foo`, and one or more glyph rendering functions, `foo_puts_bar`. The `bar` part of the renderer denotes the style of the particular renderer, which can indicate:

- Expected bitdepth of font data (`b1` for 1bpp, etc)
- Expected sizes of the character (`w8` and `h8`, for example).
- Application of system colors (`c`).
- Transparent or opaque background pixels (`t` or `o` ).
- Whether the font-data is in 'strip' layout (`s` )

The included renderers here are usually transparent, recolored, using 1bpp strip glyphs (`_b1cts` ). The initializer takes a bunch of options specific to each family, as well as font and renderer pointers. You can provide your own font and renderers,

provided they're formatted correctly. For the default font/renderers, use `NULL`.

After calling the initializer, you can write utf-8 encoded text with `tte_write()` or `tte_write_ex()`. You can also enable stdio-related functions by calling `tte_init_con()`.

The system also supports rudimentary scripting for positions, colors, margins and erases. See `tte_cmd_default()` and `con_cmd_parse()` for details.

**See also:**

[Surface functions](#)

---

# Define Documentation

```
#define TTE_BASE_VARS (_tc,  
                      _font )
```

**Value:**

```
TTC *_tc= tte_get_context();           \  
TFont *_font= tc->font;             \  
                                \
```

Declare and define base drawg variables.

Each drawg renderer usually starts with the same thing:

- Get the system and font pointers.
- Translate from ascii-character to glyph offset.
- Get the glyph (and glyph-cell) dimensions.
- Get the source and destination pointers and positions.  
These macros will make declarint and defining that easier.

```
#define TTE_CHAR_VARS ( font,  
                      gid,  
                      src_t,  
                      _sD,  
                      _sL,  
                      _chW,  
                      _chH )
```

**Value:**

```
src_t * _sD= (src_t*)(font->data+gid*font->cellWidth);
    uint _chW= font->widths ? font->widths[gid];
    uint _chH= font->charH;
```

Declare and define basic source drawg variables.

```
#define TTE_DST_VARS ( tc,
                      dst_t,
                      _dD,
                      _dL,
                      _dP,
                      _x,
                      _y )
```

**Value:**

```
uint _x= tc->cursorX, _y= tc->cursorY, _dP= tc->destPitch;
dst_t *_dD= (dst_t*)(tc->dst.data+_y*_dP),
```

Declare and define basic destination drawg variables.

```
#define tte_init_chr4c_b4_default ( bgnr,
                                    bgcnt )
```

**Value:**

```
tte_init_chr4c(bgnr, bgcnt, 0xF000, 0x0201, CI
&verdana9_b4Font, chr4c_drawg_b4cts)
```

```
#define tte_init_chr4c_default ( bgnr,
```

```
    bgcnt )
```

**Value:**

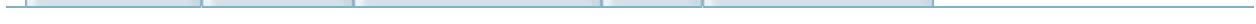
```
tte_init_chr4c(bgnr, bgcnt, 0xF000, 0x0201, CI  
    &vwf_default, NULL)
```

```
#define tte_init_chr4r_default( bgnr,  
                                bgcnt )
```

**Value:**

```
tte_init_chr4r(bgnr, bgcnt, 0xF000, 0x0201, CI  
    &vwf_default, NULL)
```

```
#define TTE_TAB_WIDTH 24
```

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# Operations

## [Tonic Text Engine]

Basic operations. [More...](#)

# Functions

void	<b><a href="#">tte_set_context</a></b> (TTC *tc) <i>Set the master context pointer.</i>
INLINE TTC *	<b><a href="#">tte_get_context</a></b> (void) <i>Get the master text-system.</i>
INLINE uint	<b><a href="#">tte_get_glyph_id</a></b> (int ch) <i>Get the glyph index of character ch.</i>
INLINE int	<b><a href="#">tte_get_glyph_width</a></b> (uint gid) <i>Get the width of glyph id.</i>
INLINE int	<b><a href="#">tte_get_glyph_height</a></b> (uint gid) <i>Get the height of glyph id.</i>
INLINE const void *	<b><a href="#">tte_get_glyph_data</a></b> (uint gid) <i>Get the glyph data of glyph id.</i>
void	<b><a href="#">tte_set_color</a></b> (eint type, u16 clr) <i>Set color attribute of type to cattr.</i>
void	<b><a href="#">tte_set_colors</a></b> (const u16 colors[]) <i>Load important color data.</i>
void	<b><a href="#">tte_set_color_attr</a></b> (eint type, u16 cattr) <i>Set color attribute of type to cattr.</i>
void	<b><a href="#">tte_set_color_attrs</a></b> (const u16 cattrs[]) <i>Load important color attribute data.</i>
char *	<b><a href="#">tte_cmd_default</a></b> (const char *str) <i>Text command handler.</i>
int	<b><a href="#">tte_putc</a></b> (int ch) <i>Plot a single character; does wrapping too.</i>
int	<b><a href="#">tte_write</a></b> (const char *text) <i>Render a string.</i>
int	<b><a href="#">tte_write_ex</a></b> (int x, int y, const char *text, const u16 *clrlut) <i>Extended string writer, with positional and color info.</i>
void	<b><a href="#">tte_erase_rect</a></b> (int left, int top, int right, int bottom) <i>Erase a portion of the screen (ignores margins).</i>

	void	<b>tte_erase_screen</b> (void) <i>Erase the screen (within the margins).</i>
	void	<b>tte_erase_line</b> (void) <i>Erase the whole line (within the margins).</i>
POINT16		<b>tte_get_text_size</b> (const char *str) <i>Get the size taken up by a string.</i>
	void	<b>tte_init_base</b> (const <b>TFont</b> *font, <b>fnDrawg</b> drawProc, <b>fnErase</b> eraseProc) <i>Base initializer of a <b>TTC</b>.</i>

---

## **Detailed Description**

Basic operations.

This covers most of the things you can actually use TTE for, like writing the text, getting information about a glyph and setting color attributes.

---

# Function Documentation

```
char* tte_cmd_default ( const char * str )
```

Text command handler.

Takes commands formatted as "#{[cmd]:[opt];[[cmd]:[opt]];...}" and deals with them.

Command list:

- **P** Set cursor to margin top-left.
- **Ps** Save cursor position
- **Pr** Restore cursor position.
- **P:#x,#y** Set cursorX/Y to x, y.
- **X** Set cursorX to margin left.
- **X:#x** Set cursorX to x.
- **Y** Set cursorY to margin top.
- **Y:#y** Set cursorX to y.
- **c[ispX]:#val** Set ink/shadow/paper/special color to val.
- **e[slbfr]** Erase screen/line/backward/forward/rect
- **m:#l,#t,#r,#b** Set all margins
- **m[ltrb]:#val** Set margin to val.
- **p:#x,#y** Move cursorX/Y by x, y.
- **w:#val** Wait val frame.
- **x:#x** Move cursorX by x.
- **y:#y** Move cursorX by y.

Examples:

- **#{X:32}** Move to x = 32;
- **#{ci:0x7FFF}** Set ink color to white.

- **#{w:120;es;P}** Wait 120 frames, clear screen, return to top of screen.

**Parameters:**

*str* Start of command. Assumes the initial "\{" is lobbed off already.

**Returns:**

pointer to after the parsed command.

**Note:**

Routine does text wrapping. Make sure margins are set.

This function involves heavy (yet necessary) switching. Leave your sanity at the door before viewing.

**Todo:**

Scrolling and variables ?

**Todo:**

Restructure for safety checks.

### **void tte\_erase\_line( void )**

Erase the whole line (within the margins).

**Note:**

Ponder: set paper color?

### **void tte\_erase\_screen( void )**

Erase the screen (within the margins).

**Note:**

Ponder: set paper color?

**POINT16 tte\_get\_text\_size ( const char \* *str* )**

Get the size taken up by a string.

**Parameters:**

*str* String to check.

**Returns:**

width and height, packed into a POINT16.

**Note:**

This function *ignores* tte commands, so don't use on strings that use commands.

**int tte\_putc ( int *ch* )**

Plot a single character; does wrapping too.

**Parameters:**

*ch* Character to plot (not glyph-id).

**Returns:**

Character width.

**Note:**

Overhead: ~70 cycles.

```
void tte_set_context ( TTC * tc )
```

Set the master context pointer.

```
int tte_write ( const char * text )
```

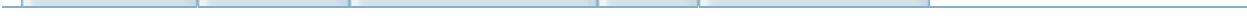
Render a string.

**Parameters:**

*text* String to parse and write.

**Returns:**

Number of parsed characters.

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# Attributes

## [Tonic Text Engine]

Basic getters and setters. [More...](#)

## Functions

INLINE void	<b>tte_get_pos</b> (int *x, int *y) <i>Get cursor position.</i>
INLINE u16	<b>tte_get_ink</b> (void) <i>Get ink color attribute.</i>
INLINE u16	<b>tte_get_shadow</b> (void) <i>Get shadow color attribute.</i>
INLINE u16	<b>tte_get_paper</b> (void) <i>Get paper color attribute.</i>
INLINE u16	<b>tte_get_special</b> (void) <i>Get special color attribute.</i>
INLINE TSurface *	<b>tte_get_surface</b> () <i>Get a pointer to the text surface.</i>
INLINE TFont *	<b>tte_get_font</b> (void) <i>Get the active font.</i>
INLINE fnDrawg	<b>tte_get_drawg</b> (void) <i>Get the active character plotter.</i>
INLINE fnErase	<b>tte_get_erase</b> (void) <i>Get the character plotter.</i>
INLINE char **	<b>tte_get_string_table</b> (void) <i>Get string table.</i>
INLINE TFont **	<b>tte_get_font_table</b> (void) <i>Get font table.</i>
INLINE void	<b>tte_set_pos</b> (int x, int y) <i>Set cursor position.</i>
INLINE void	<b>tte_set_ink</b> (u16 cattr) <i>Set ink color attribute.</i>
INLINE void	<b>tte_set_shadow</b> (u16 cattr) <i>Set shadow color attribute.</i>
INLINE void	<b>tte_set_paper</b> (u16 cattr) <i>Set paper color attribute.</i>
INLINE void	<b>tte_set_special</b> (u16 cattr)

	<i>Set special color attribute.</i>
INLINE void	<b>tte_set_surface</b> (const TSurface *srf) <i>Set the text surface.</i>
INLINE void	<b>tte_set_font</b> (const <b>TFont</b> *font) <i>Set the font.</i>
INLINE void	<b>tte_set_drawg</b> (fnDrawg proc) <i>Set the character plotter.</i>
INLINE void	<b>tte_set_erase</b> (fnErase proc) <i>Set the character plotter.</i>
INLINE void	<b>tte_set_string_table</b> (const char *table[]) <i>Set string table.</i>
INLINE void	<b>tte_set_font_table</b> (const <b>TFont</b> *table[]) <i>Set font table.</i>
void	<b>tte_set_margins</b> (int left, int top, int right, int bottom)

---

## **Detailed Description**

Basic getters and setters.

---

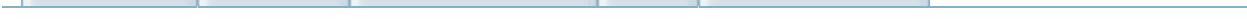
# Function Documentation

```
INLINE void tte_get_pos( int * x,  
                         int * y  
                       )
```

Get cursor position.

---

*Generated on Mon Aug 25 17:03:57 2008 for libtong by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# Console IO

## [Tonic Text Engine]

Stdio functionality. [More...](#)

## Defines

```
#define tte_printf iprintf
```

## Functions

void	<b>tte_init_con</b> (void)
<i>Init stdio capabilities.</i>	
int	<b>tte_cmd_vt100</b> (const char *text)
<i>Parse for VT100-sequences.</i>	
int	<b>tte_con_write</b> (struct _reent *r, int fd, const char *text, int len)
<i>Internal routine for stdio functionality.</i>	
int	<b>tte_con_nocash</b> (struct _reent *r, int fd, const char *text, int len)

---

## Detailed Description

Stdio functionality.

These functions allow you to use stdio routines for writing, like printf, puts and such. Note that tte\_printf is just iprintf ... at least for now.

---

## Define Documentation

```
#define tte_printf iprintf
```

Wrapper 'function' to hide that we're making iprintf do things it doesn't usually do.

---

# Function Documentation

```
int tte_cmd_vt100 ( const char * text )
```

Parse for VT100-sequences.

Taken liberally from libgba.

See [here](#) for a full overview.

## Parameters:

*text* Sequence string, starting at the '['.

## Todo:

: check for buffer overflow.

```
int tte_con_write ( struct _reent * r,
                     int           fd,
                     const char *   text,
                     int           len
)
```

Internal routine for stdio functionality.

## Note:

While this function 'works', I am not 100% sure I'm handling everything correctly.

```
void tte_init_con ( void )
```

Init stdio capabilities.

---

---

Generated on Mon Aug 25 17:03:57 2008 for libtonc by  doxygen 1.5.3

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **Tilemap text**

## **[Tonic Text Engine]**

Text for regular and affine tilemaps. [More...](#)

## Regular tilemaps

void	<b>tte_init_se</b> (int bgnr, u16 bgcnt, SCR_ENTRY se0, u32 clrs, u32 bupofs, const <b>TFont</b> *font, <b>fnDrawg</b> proc)	<i>Initialize text system for screen-entry fonts.</i>
void	<b>se_erase</b> (int left, int top, int right, int bottom)	<i>Erase part of the regular tilemap canvas.</i>
void	<b>se_drawg_w8h8</b> (uint gid)	<i>Character-plot for reg BGs using an 8x8 font.</i>
void	<b>se_drawg_w8h16</b> (uint gid)	<i>Character-plot for reg BGs using an 8x16 font.</i>
void	<b>se_drawg</b> (uint gid)	<i>Character-plot for reg BGs, any sized font.</i>
void	<b>se_drawg_s</b> (uint gid)	<i>Character-plot for reg BGs, any sized, vertically tiled font.</i>

## Affine tilemaps

void	<b>tte_init_ase</b> (int bgnr, u16 bgcnt, u8 ase0, u32 clrs, u32 bupofs, const <b>TFont</b> *font, <b>fnDrawg</b> proc)	<i>Initialize text system for affine screen-entry fonts.</i>
void	<b>ase_erase</b> (int left, int top, int right, int bottom)	<i>Erase part of the affine tilemap canvas.</i>
void	<b>ase_drawg_w8h8</b> (uint gid)	<i>Character-plot for affine BGs using an 8x8 font.</i>
void	<b>ase_drawg_w8h16</b> (uint gid)	<i>Character-plot for affine BGs using an 8x16 font.</i>
void	<b>ase_drawg</b> (uint gid)	<i>Character-plot for affine Bgs, any size.</i>
void	<b>ase_drawg_s</b> (uint gid)	<i>Character-plot for affine BGs, any sized, vertically oriented font.</i>

## Detailed Description

Text for regular and affine tilemaps.

The tilemap sub-system loads the tiles into memory first, then writes to the map to show the letters. For this to work properly, the glyph sizes should be 8-pixel aligned.

**Note:**

At present, the regular tilemap text ignores screenblock boundaries, so 512px wide maps may not work properly.

---

# Function Documentation

```
void tte_init_ase( int          bgnr,
                    u16          bgcnt,
                    u8           ase0,
                    u32          clrs,
                    u32          bupofs,
                    const TFont * font,
                    fnDrawg      proc
)
```

Initialize text system for affine screen-entry fonts.

## Parameters:

- bgnr* Number of background to be used for text.
- bgcnt* Background control flags.
- ase0* Base screen entry. This allows a greater range in capabilities, like offset tile-starts.
- clrs* colors to use for the text. The palette entries used depends on *ase0* and *bupofs*.
- bupofs* Flags for font bit-unpacking. Basically indicates pixel values (and hence palette use).
- font* Font to initialize with.
- proc* Character plotting procedure.

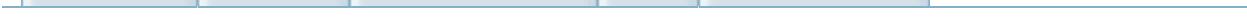
```
void tte_init_se( int          bgnr,
                   u16          bgcnt,
                   SCR_ENTRY   se0,
                   u32          clrs,
                   u32          bupofs,
```

```
    const TFont * font,  
    fnDrawg      proc  
)
```

Initialize text system for screen-entry fonts.

### Parameters:

- bgnr* Number of background to be used for text.
- bgcnt* Background control flags.
- se0* Base screen entry. This allows a greater range in capabilities, like offset tile-starts and palettes.
- clrs* colors to use for the text. The palette entries used depends on *se0* and *bupofs*.
- bupofs* Flags for font bit-unpacking. Basically indicates pixel values (and hence palette use).
- font* Font to initialize with.
- proc* Glyph renderer.

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

## **Character text, column-major**

### **[Tonic Text Engine]**

Text on surface composed of 4bpp tiles, mapped in column-major order. [More...](#)

## 4bpp tiles

	void	<b>tte_init_chr4c</b> (int bgnr, u16 bgcnt, u16 se0, u32 cattrs, u32 clrs, const <b>TFont</b> *font, <b>fnDrawg</b> proc) <i>Initialize text system for 4bpp tiled, column-major surfaces.</i>
	void	<b>chr4c_erase</b> (int left, int top, int right, int bottom) <i>Erase part of the 4bpp text canvas.</i>
	void	<b>chr4c_drawg_b1cts</b> (uint gid) <i>Render 1bpp fonts to 4bpp tiles.</i>
IWRAM_CODE	void	<b>chr4c_drawg_b1cts_fast</b> (uint gid) <i>Initialize text system for 4bpp tiled, column-major surfaces.</i>
	void	<b>chr4c_drawg_b4cts</b> (uint gid) <i>Initialize text system for 4bpp tiled, column-major surfaces.</i>
IWRAM_CODE	void	<b>chr4c_drawg_b4cts_fast</b> (uint gid) <i>Initialize text system for 4bpp tiled, column-major surfaces.</i>

## Detailed Description

Text on surface composed of 4bpp tiles, mapped in column-major order.

There are actually two *chr4* systems. The difference between the two is the ordering of the tiles: column-major versus row-major. Since column-major is 'better', this is considered the primary sub-system for tiled text.

### See also:

[4bpp tiled surfaces, column major](#)

---

# Function Documentation

```
IWRAM_CODE void chr4c_drawg_b1cts_fast ( uint gid )
```

Initialize text system for 4bpp tiled, column-major surfaces.

## Parameters:

*bgnr* Background number.  
*bgcnt* Background control flags.  
*se0* Base offset for screen-entries.  
*cattrs* Color attributes; one byte per attr.  
*clrs* ink(/shadow) colors.  
*font* Font to initialize with.  
*proc* Glyph renderer

```
void chr4c_drawg_b4cts ( uint gid )
```

Initialize text system for 4bpp tiled, column-major surfaces.

## Parameters:

*bgnr* Background number.  
*bgcnt* Background control flags.  
*se0* Base offset for screen-entries.  
*cattrs* Color attributes; one byte per attr.  
*clrs* ink(/shadow) colors.  
*font* Font to initialize with.  
*proc* Glyph renderer

```
IWRAM_CODE void chr4c_drawg_b4cts_fast ( uint gid )
```

Initialize text system for 4bpp tiled, column-major surfaces.

### Parameters:

*bgnr* Background number.  
*bgcnt* Background control flags.  
*se0* Base offset for screen-entries.  
*cattrs* Color attributes; one byte per attr.  
*clrs* ink(/shadow) colors.  
*font* Font to initialize with.  
*proc* Glyph renderer

```
void tte_init_chr4c ( int          bgnr,  
                      u16           bgcnt,  
                      u16           se0,  
                      u32           cattrs,  
                      u32           clrs,  
                      const TFont * font,  
                      fnDrawg       proc  
)
```

Initialize text system for 4bpp tiled, column-major surfaces.

### Parameters:

*bgnr* Background number.  
*bgcnt* Background control flags.  
*se0* Base offset for screen-entries.  
*cattrs* Color attributes; one byte per attr.  
*clrs* ink(/shadow) colors.  
*font* Font to initialize with.  
*proc* Glyph renderer

---

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  **1.5.3**

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

## **Character text, row-major**

### **[Tonic Text Engine]**

Text on surface composed of 4bpp tiles, mapped in row-major order. [More...](#)

## 4bpp tiles

	void	<b>tte_init_chr4r</b> (int bgnr, u16 bgcnt, u16 se0, u32 cattrs, u32 clrs, const <b>TFont</b> *font, <b>fnDrawg</b> proc) <i>Initialize text system for 4bpp tiled, column-major surfaces.</i>
	void	<b>chr4r_erase</b> (int left, int top, int right, int bottom) <i>Erase part of the 4bpp text canvas.</i>
	void	<b>chr4r_drawg_b1cts</b> (uint gid) <i>Render 1bpp fonts to 4bpp tiles.</i>
IWRAM_CODE	void	<b>chr4r_drawg_b1cts_fast</b> (uint gid) <i>Initialize text system for 4bpp tiled, column-major surfaces.</i>

## Detailed Description

Text on surface composed of 4bpp tiles, mapped in row-major order.

There are actually two *chr4* systems, with row-major and column-major tile indexing. The column-major version is more advanced, so use that when possible.

**See also:**

[4bpp tiled surfaces, row major](#)

---

# Function Documentation

```
IWRAM_CODE void chr4r_drawg_b1cts_fast( uint gid )
```

Initialize text system for 4bpp tiled, column-major surfaces.

## Parameters:

*bgnr* Background number.  
*bgcnt* Background control flags.  
*se0* Base offset for screen-entries.  
*cattrs* Color attributes; one byte per attr.  
*clrs* ink(/shadow) colors.  
*font* Font to initialize with.  
*proc* Glyph renderer

```
void tte_init_chr4r( int bgnr,
                      u16   bgcnt,
                      u16   se0,
                      u32   cattrs,
                      u32   clrs,
                      const TFont * font,
                      fnDrawg     proc
                    )
```

Initialize text system for 4bpp tiled, column-major surfaces.

## Parameters:

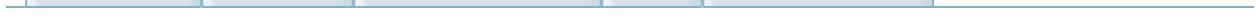
*bgnr* Background number.  
*bgcnt* Background control flags.  
*se0* Base offset for screen-entries.

*cattrs* Color attributes; one byte per attr.

*clrs* ink(/shadow) colors.

*font* Font to initialize with.

*proc* Glyph renderer

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

## **Bitmap text**

### **[Tonic Text Engine]**

Text for 16bpp and 8bpp bitmap surfaces: modes 3, 4 and 5.

[More...](#)

## 8bpp bitmaps

void	<b>bmp8_erase</b> (int left, int top, int right, int bottom) <i>Erase part of the 8bpp text canvas.</i>
void	<b>bmp8_drawg</b> (uint gid) <i>Linear 8 bpp bitmap glyph renderer, opaque.</i>
void	<b>bmp8_drawg_t</b> (uint gid) <i>Linear 8 bpp bitmap glyph renderer, transparent.</i>
void	<b>bmp8_drawg_b1cts</b> (uint gid) <i>Erase part of the 8bpp text canvas.</i>
IWRAM_CODE void	<b>bmp8_drawg_b1cts_fast</b> (uint gid) <i>Erase part of the 8bpp text canvas.</i>
void	<b>bmp8_drawg_b1cos</b> (uint gid) <i>Erase part of the 8bpp text canvas.</i>

## 16bpp bitmaps

void	<b>bmp16_erase</b> (int left, int top, int right, int bottom) <i>Erase part of the 16bpp text canvas.</i>
void	<b>bmp16_drawg</b> (uint gid) <i>Linear 16bpp bitmap glyph renderer, opaque.</i>
void	<b>bmp16_drawg_t</b> (uint gid) <i>Linear 16bpp bitmap glyph renderer, transparent.</i>
void	<b>bmp16_drawg_b1cts</b> (uint gid) <i>Linear bitmap, 16bpp transparent character plotter.</i>
void	<b>bmp16_drawg_b1cos</b> (uint gid) <i>Linear bitmap, 16bpp opaque character plotter.</i>

## Functions

void	<b>tte_init bmp</b> (int vmode, const <b>TFont</b> *font, <b>fnDrawg</b> proc) <i>Initialize text system for bitmap fonts.</i>
------	---

---

## **Detailed Description**

Text for 16bpp and 8bpp bitmap surfaces: modes 3, 4 and 5.

Note that TTE does not update the pointer of the surface for page-flipping. You'll have to do that yourself.

---

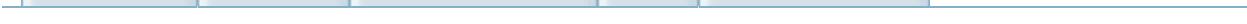
# Function Documentation

```
void tte_init_bmp ( int           vmode,
                     const TFont * font,
                     fnDrawg      proc
)
```

Initialize text system for bitmap fonts.

## Parameters:

- vmode* Video mode (3,4 or 5).
- font* Font to initialize with.
- proc* Glyph renderer.

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **Object text**

## [Tonic Text Engine]

Text using objects. [More...](#)

## Functions

void	<b>tte_init_obj</b> ( <b>OBJ_ATTR</b> *dst, u32 attr0, u32 attr1, u32 attr2, u32 clrs, u32 bupofs, const <b>TFont</b> *font, <b>fnDrawg</b> proc)
	<i>Initialize text system for screen-entry fonts.</i>
void	<b>obj_erase</b> (int left, int top, int right, int bottom)
	<i>Unwind the object text-buffer.</i>
void	<b>obj_drawg</b> (uint gid)
	<i>Character-plot for objects.</i>

## Detailed Description

Text using objects.

This is similar to tilemaps, in that the glyphs are loaded into object VRAM first and pointed to by the objects. Unlike tilemaps, though, variable-width fonts are possible here. The members of the surface member are used a little differently here, though. The `pitch` is used as an index to the current object, and `width` is the number of objects allowed to be used for text.

---

# Function Documentation

```
void tte_init_obj( OBJ_ATTR * obj,
                    u32           attr0,
                    u32           attr1,
                    u32           attr2,
                    u32           clrs,
                    u32           bupofs,
                    const TFont * font,
                    fnDrawg       proc
)
```

Initialize text system for screen-entry fonts.

## Parameters:

- obj* Destination object.
- attr0* Base obj.attr0.
- attr1* Base obj.attr1.
- attr2* Base obj.attr2.
- clrs* colors to use for the text. The palette entries used depends on *attr2* and *bupofs*.
- bupofs* Flags for font bit-unpacking. Basically indicates pixel values (and hence palette use).
- font* Font to initialize with.
- proc* Character plotting procedure.

## Note:

The TTE-obj system uses the surface differently than the rest. Be careful when modifying the surface data.

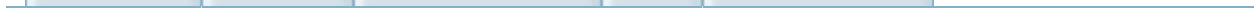
## Todo:

Multi-bpp.

---

---

Generated on Mon Aug 25 17:03:57 2008 for libtonc by  doxygen 1.5.3

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

## **Old Text**

Text writers for all modes and objects. [More...](#)

# Modules

- Tilemap text**
- Bitmap text**
- Object text**

## Functions

---

void	<a href="#">txt_init_std</a> ()
void	<a href="#">txt_bup_1toX</a> (void *dstv, const void *srcv, u32 len, int bpp, u32 base)

---

## Detailed Description

Text writers for all modes and objects.

### Deprecated:

While potentially still useful, TTE is considerably more advanced. Use that instead.

There are three types of text writers here:

- Tilemap (`se_routines`)
- Bitmap (`bm_and_mx_routines`)
- Object (`obj_routines`)

Each of these has an initializer, a char writer, and string writer and a string clearer. The general interface for all of these is `foo(x, y, string/char, special)`, Where x and y are the positions **in pixels**, and special depends on the mode-type: it can be a color, base screenentry or whatever.

The clearing routines also use a string parameter, which is used to indicate the exact area to clear. You're free to clear the whole buffer if you like.

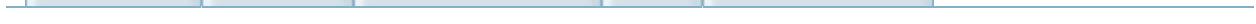
---

# Function Documentation

```
void txt_init_std( )
```

---

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by  doxygen 1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

**Tilemap text**  
**[Old Text]**

## Functions

void	<b>txt_init_se</b> (int bgnr, u16 bgcnt, SCR_ENTRY se0, u32 clrs, u32 base)
void	<b>se_putc</b> (int x, int y, int c, SCR_ENTRY se0)
void	<b>se_puts</b> (int x, int y, const char *str, SCR_ENTRY se0)
void	<b>se_clrs</b> (int x, int y, const char *str, SCR_ENTRY se0)

---

# Function Documentation

```
void txt_init_se ( int          bgnr,  
                  u16           bgcnt,  
                  SCR_ENTRY    se0,  
                  u32           clrs,  
                  u32           base  
            )
```

---

Generated on Mon Aug 25 17:03:57 2008 for libtonc by  doxygen 1.5.3

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

**Bitmap text**  
**[Old Text]**

## Mode-independent functions

```
void bm_putc (int x, int y, int c, COLOR clr)
void bm_puts (int x, int y, const char *str, COLOR clr)
void bm_clrs (int x, int y, const char *str, COLOR clr)
```

## Mode 3 functions

INLINE void	<b>m3_putc</b> (int x, int y, int c, <b>COLOR</b> clr) <i>Write character c to (x, y) in color clr in mode 3.</i>
INLINE void	<b>m3_puts</b> (int x, int y, const char *str, <b>COLOR</b> clr) <i>Write string str to (x, y) in color clr in mode 3.</i>
INLINE void	<b>m3_clrs</b> (int x, int y, const char *str, <b>COLOR</b> clr) <i>Clear the space used by string str at (x, y) in color clr in mode 3.</i>

## Mode 4 functions

INLINE void	<b>m4_putc</b> (int x, int y, int c, u8 clrid) <i>Write character c to (x, y) in color-index clrid in mode 4.</i>
INLINE void	<b>m4_puts</b> (int x, int y, const char *str, u8 clrid) <i>Write string str to (x, y) in color-index clrid in mode 4.</i>
INLINE void	<b>m4_clrs</b> (int x, int y, const char *str, u8 clrid) <i>Clear the space used by string str at (x, y) in color-index clrid in mode 4.</i>

## Mode 5 functions

INLINE void	<b>m5_putc</b> (int x, int y, int c, <b>COLOR</b> clr) <i>Write character c to (x, y) in color clr in mode 5.</i>
INLINE void	<b>m5_puts</b> (int x, int y, const char *str, <b>COLOR</b> clr) <i>Write string str to (x, y) in color clr in mode 5.</i>
INLINE void	<b>m5_clrs</b> (int x, int y, const char *str, <b>COLOR</b> clr) <i>Clear the space used by string str at (x, y) in color clr in mode 5.</i>

## Mode 5 functions

---

```
void bm16_putc (u16 *dst, int c, COLOR clr, int pitch)
void bm16_puts (u16 *dst, const char *str, COLOR clr, int pitch)
void bm16_clrs (u16 *dst, const char *str, COLOR clr, int pitch)
void bm8_putc (u16 *dst, int c, u8 clrid)
void bm8_puts (u16 *dst, const char *str, u8 clrid)
```

---

Generated on Mon Aug 25 17:03:57 2008 for libtong by  doxygen 1.5.3

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

**Object text**  
**[Old Text]**

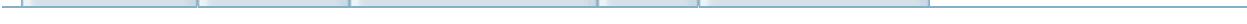
## Functions

INLINE void	<b>obj_putc2</b> (int x, int y, int c, u16 attr2, <b>OBJ_ATTR</b> *obj0) <i>Write character c to (x, y) in color clr using objects obj0 and on.</i>
INLINE void	<b>obj_puts2</b> (int x, int y, const char *str, u16 attr2, <b>OBJ_ATTR</b> *obj0) <i>Write string str to (x, y) in color clr using objects obj0 and on.</i>
void	<b>txt_init_obj</b> ( <b>OBJ_ATTR</b> *obj0, u16 attr2, u32 clrs, u32 base)
void	<b>obj_putc</b> (int x, int y, int c, u16 attr2)
void	<b>obj_puts</b> (int x, int y, const char *str, u16 attr2)
void	<b>obj_clrs</b> (int x, int y, const char *str)

# Function Documentation

```
INLINE void obj_putc2( int           x,
                      int           y,
                      int           c,
                      u16          attr2,
                      OBJ_ATTR * obj0
)
```

Write character *c* to (*x*, *y*) in color *clr* using objects *obj0* and on.

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **Timer**

## Functions

INLINE void	<b>profile_start</b> (void) <i>Start a profiling run.</i>
INLINE uint	<b>profile_stop</b> (void) <i>Stop a profiling run and return the time since its start.</i>

---

## **Detailed Description**

---

# Function Documentation

**INLINE void profile\_start ( void )**

Start a profiling run.

**Note:**

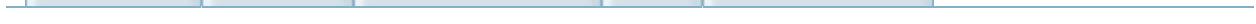
Routine uses timers 3 and 3; if you're already using these somewhere, chaos is going to ensue.

**INLINE uint profile\_stop ( void )**

Stop a profiling run and return the time since its start.

**Returns:**

32bit cycle count

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **Video**

# Modules

## [Surface functions](#)

*Basic video surface API. The TSurface struct and the various functions working on it provide a basic API for working with different types of graphic surfaces, like 16bpp bitmaps, 8bpp bitmaps, but also tiled surfaces.*

## [Colors](#)

## [Tiled Backgrounds](#)

## [Bitmaps](#)

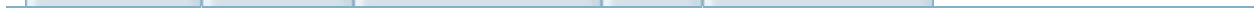
## [Objects](#)

## [Affine functions](#)

## Detailed Description

---

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  **doxygen** 1.5.3

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# Surface functions

## [[Video](#)]

Basic video surface API. The TSurface struct and the various functions working on it provide a basic API for working with different types of graphic surfaces, like 16bpp bitmaps, 8bpp bitmaps, but also tiled surfaces.

. [More...](#)

# Modules

[\*\*16bpp bitmap surfaces\*\*](#)

[\*\*8bpp bitmap surfaces\*\*](#)

[\*\*4bpp tiled surfaces, column major\*\*](#)

[\*\*4bpp tiled surfaces, row major\*\*](#)

## Enumerations

enum	<pre>ESurfaceType {     SRF_NONE = 0, SRF_BMP16 = 1, SRF_BMP8 = 2, SRF_CHR4R = 4,     SRF_CHR4C = 5, SRF_CHR8 = 6, SRF_ALLOCATED = 0x80 }</pre>
	<i>Surface types.</i> <a href="#">More...</a>

## Functions

void	<b>srf_init</b> (TSurface *srf, enum <b>ESurfaceType</b> type, const void *data, uint width, uint height, uint bpp, u16 *pal) <i>Initialize a surface for type formatted graphics.</i>
void	<b>srf_pal_copy</b> (const TSurface *dst, const TSurface *src, uint count) <i>Copy count colors from src's palette to dst's palette.</i>
void *	<b>srf_get_ptr</b> (const TSurface *srf, uint x, uint y) <i>Get the byte address of coordinates (x, y) on the surface.</i>
INLINE uint	<b>srf_align</b> (uint width, uint bpp) <i>Get the word-aligned number of bytes for a scanline.</i>
INLINE void	<b>srf_set_ptr</b> (TSurface *srf, const void *ptr) <i>Set Data-pointer surface for srf.</i>
INLINE void	<b>srf_set_pal</b> (TSurface *srf, const u16 *pal, uint size) <i>Set the palette pointer and its size.</i>
INLINE void *	<b>_srf_get_ptr</b> (const TSurface *srf, uint x, uint y, uint stride) <i>Inline and semi-safe version of <b>srf_get_ptr()</b>. Use with caution.</i>

## Detailed Description

Basic video surface API. The `TSurface` struct and the various functions working on it provide a basic API for working with different types of graphic surfaces, like 16bpp bitmaps, 8bpp bitmaps, but also tiled surfaces.

Tonclib's Surface system provides the basic functionality for drawing onto graphic surfaces of different types. This includes

- **bmp16**: 16bpp bitmap surfaces
  - **bmp8**: 8bpp bitmap surfaces.
  - **chr4(c/r)**: 4bpp tiled surfaces. This covers almost all of the GBA graphic modes.
- 
- **SRF\_BMP8**: 8bpp linear (Mode 4 / affine BGs)
  - **SRF\_BMP16** 16bpp bitmaps (Mode 3/5 / regular BGs to some extent)
  - **SRF\_CHR4C** 4bpp tiles, column-major (Regular tiled BG)
  - **SRF\_CHR4R** 4bpp tiles, row-major (Regular tiled BG, OBJs)

For each of these functions exist for the most important drawing options: plotting, lines and rectangles. For BMP8/BMP16 and to some extent CHR4C, there are blitters as well.

---

# Enumeration Type Documentation

## enum **ESurfaceType**

Surface types.

### Enumerator:

<i>SRF_NONE</i>	No specific type.
<i>SRF_BMP16</i>	16bpp linear (bitmap/tilemap).
<i>SRF_BMP8</i>	8bpp linear (bitmap/tilemap).
<i>SRF_CHR4R</i>	4bpp tiles, row-major.
<i>SRF_CHR4C</i>	4bpp tiles, column-major.
<i>SRF_CHR8</i>	8bpp tiles, row-major.
<i>SRF_ALLOCATED</i>	Pointers have been allocated.

---

# Function Documentation

```
INLINE uint srf_align ( uint width,  
                      uint bpp  
)
```

Get the word-aligned number of bytes for a scanline.

## Parameters:

*width* Number of pixels.

*bpp* Bits per pixel.

```
void srf_init ( TSurface * srf,  
                enum ESurfaceType type,  
                const void * data,  
                uint width,  
                uint height,  
                uint bpp,  
                u16 * pal  
)
```

Initialize a surface for *type* formatted graphics.

## Parameters:

*srf* Surface to initialize.

*type* Surface type. See

## See also:

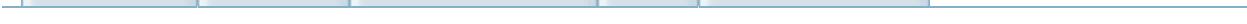
[ESurfaceType](#) for details.

## Parameters:

*data* Pointer to the surface memory.  
*width* Width of surface.  
*height* Height of surface.  
*bpp* Bitdepth. If *type* is not 0, this value will be ignored.  
*pal* Pointer to the surface's palette.

---

Generated on Mon Aug 25 17:03:57 2008 for *libtong* by  1.5.3

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **16bpp bitmap surfaces**

## **[Surface functions]**

## Functions

u32	<b>sbmp16_get_pixel</b> (const TSurface *src, int x, int y) <i>Get the pixel value of src at (x, y).</i>
void	<b>sbmp16_plot</b> (const TSurface *dst, int x, int y, u32 clr) <i>Plot a single pixel on a 16-bit buffer.</i>
void	<b>sbmp16_hline</b> (const TSurface *dst, int x1, int y, int x2, u32 clr) <i>Draw a horizontal line on an 16bit buffer.</i>
void	<b>sbmp16_vline</b> (const TSurface *dst, int x, int y1, int y2, u32 clr) <i>Draw a vertical line on an 16bit buffer.</i>
void	<b>sbmp16_line</b> (const TSurface *dst, int x1, int y1, int x2, int y2, u32 clr) <i>Draw a line on an 16bit buffer.</i>
void	<b>sbmp16_rect</b> (const TSurface *dst, int left, int top, int right, int bottom, u32 clr) <i>Draw a rectangle in 16bit mode.</i>
void	<b>sbmp16_frame</b> (const TSurface *dst, int left, int top, int right, int bottom, u32 clr) <i>Draw a rectangle in 16bit mode.</i>
void	<b>sbmp16.blit</b> (const TSurface *dst, int dstX, int dstY, uint width, uint height, const TSurface *src, int srcX, int srcY) <i>16bpp blitter. Copies a rectangle from one surface to another.</i>
void	<b>sbmp16_floodfill</b> (const TSurface *dst, int x, int y, u32 clr) <i>Floodfill an area of the same color with new color clr.</i>
INLINE void	<b>_sbmp16_plot</b> (const TSurface *dst, int x, int y, u32 clr) <i>Plot a single pixel on a 16-bit buffer; inline version.</i>
INLINE u32	<b>_sbmp16_get_pixel</b> (const TSurface *src, int x, int y) <i>Get the pixel value of src at (x, y); inline version.</i>

## **Detailed Description**

Routines for 16bpp linear surfaces. For use in modes 3 and 5.  
Can also be used for regular tilemaps to a point.

---

# Function Documentation

```
void sbmp16.blit ( const TSurface * dst,  
                    int          dstX,  
                    int          dstY,  
                    uint         width,  
                    uint         height,  
                    const TSurface * src,  
                    int          srcX,  
                    int          srcY  
                )
```

16bpp blitter. Copies a rectangle from one surface to another.

## Parameters:

*dst* Destination surface.  
*dstX* Left coord of rectangle on *dst*.  
*dstY* Top coord of rectangle on *dst*.  
*width* Width of rectangle to blit.  
*height* Height of rectangle to blit.  
*src* Source surface.  
*srcX* Left coord of rectangle on *src*.  
*srcY* Top coord of rectangle on *src*.

## Note:

The rectangle will be clipped to both *src* and *dst*.

```
void sbmp16.floodfill ( const TSurface * dst,  
                        int          x,  
                        int          y,
```

```
    u32           clr  
    )
```

Floodfill an area of the same color with new color *clr*.

#### Parameters:

*dst* Destination surface.  
*x* X-coordinate.  
*y* Y-coordinate;  
*clr* Color.

```
void sbmp16_frame ( const TSurface * dst,  
                     int             left,  
                     int             top,  
                     int             right,  
                     int            bottom,  
                     u32            clr  
                     )
```

Draw a rectangle in 16bit mode.

#### Parameters:

*dst* Destination surface.  
*left* Left side of rectangle;  
*top* Top side of rectangle.  
*right* Right side of rectangle.  
*bottom* Bottom side of rectangle.  
*clr* Color.

#### Note:

Does normalization, but not bounds checks.

## PONDER: RB in- or exclusive?

```
u32 sbmp16_get_pixel ( const TSurface * src,
                        int           x,
                        int           y
                      )
```

Get the pixel value of *src* at (*x*, *y*).

```
void sbmp16_hline ( const TSurface * dst,
                     int           x1,
                     int           y,
                     int           x2,
                     u32          clr
                   )
```

Draw a horizontal line on an 16bit buffer.

### Parameters:

*dst* Destination surface.

*x1* First X-coord.

*y* Y-coord.

*x2* Second X-coord.

*clr* Color.

### Note:

Does normalization, but not bounds checks.

```
void sbmp16_line ( const TSurface * dst,
```

```
    int          x1,  
    int          y1,  
    int          x2,  
    int          y2,  
    u32         clr  
)
```

Draw a line on an 16bit buffer.

### Parameters:

*dst* Destination surface.  
*x1* First X-coord.  
*y1* First Y-coord.  
*x2* Second X-coord.  
*y2* Second Y-coord.  
*clr* Color.

### Note:

Does normalization, but not bounds checks.

```
void sbmp16_plot ( const TSurface * dst,  
                   int          x,  
                   int          y,  
                   u32         clr  
)
```

Plot a single pixel on a 16-bit buffer.

### Parameters:

*dst* Destination surface.  
*x* X-coord.  
*y* Y-coord.

*clr* Color.

**Note:**

Slow as fuck. Inline plotting functionality if possible.

```
void sbmp16_rect( const TSurface * dst,  
                  int           left,  
                  int           top,  
                  int           right,  
                  int           bottom,  
                  u32          clr  
                )
```

Draw a rectangle in 16bit mode.

**Parameters:**

*dst* Destination surface.  
*left* Left side of rectangle;  
*top* Top side of rectangle.  
*right* Right side of rectangle.  
*bottom* Bottom side of rectangle.  
*clr* Color.

**Note:**

Does normalization, but not bounds checks.

```
void sbmp16_vline( const TSurface * dst,  
                   int           x,  
                   int           y1,  
                   int           y2,  
                   u32          clr
```

)

Draw a vertical line on an 16bit buffer.

**Parameters:**

*dst* Destination surface.

*x* X-coord.

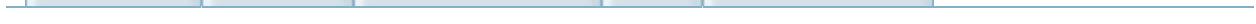
*y1* First Y-coord.

*y2* Second Y-coord.

*clr* Color.

**Note:**

Does normalization, but not bounds checks.

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **8bpp bitmap surfaces**

## **[Surface functions]**

## Functions

u32	<b>sbmp8_get_pixel</b> (const TSurface *src, int x, int y) <i>Get the pixel value of src at (x, y).</i>
void	<b>sbmp8_plot</b> (const TSurface *dst, int x, int y, u32 clr) <i>Plot a single pixel on a 8-bit buffer.</i>
void	<b>sbmp8_hline</b> (const TSurface *dst, int x1, int y, int x2, u32 clr) <i>Draw a horizontal line on an 8-bit buffer.</i>
void	<b>sbmp8_vline</b> (const TSurface *dst, int x, int y1, int y2, u32 clr) <i>Draw a vertical line on an 8-bit buffer.</i>
void	<b>sbmp8_line</b> (const TSurface *dst, int x1, int y1, int x2, int y2, u32 clr) <i>Draw a line on an 8-bit buffer.</i>
void	<b>sbmp8_rect</b> (const TSurface *dst, int left, int top, int right, int bottom, u32 clr) <i>Draw a rectangle in 8-bit mode.</i>
void	<b>sbmp8_frame</b> (const TSurface *dst, int left, int top, int right, int bottom, u32 clr) <i>Draw a rectangle in 8-bit mode.</i>
void	<b>sbmp8_blit</b> (const TSurface *dst, int dstX, int dstY, uint width, uint height, const TSurface *src, int srcX, int srcY) <i>16bpp blitter. Copies a rectangle from one surface to another.</i>
void	<b>sbmp8_floodfill</b> (const TSurface *dst, int x, int y, u32 clr) <i>Floodfill an area of the same color with new color clr.</i>
INLINE void	<b>_sbmp8_plot</b> (const TSurface *dst, int x, int y, u32 clr) <i>Plot a single pixel on a 8-bit surface; inline version.</i>
INLINE u32	<b>_sbmp8_get_pixel</b> (const TSurface *src, int x, int y) <i>Get the pixel value of src at (x, y); inline version.</i>

## **Detailed Description**

Routines for 8bpp linear surfaces. For use in mode 4 and affine tilemaps.

---

# Function Documentation

```
void sbmp8.blit ( const TSurface * dst,
                  int           dstX,
                  int           dstY,
                  uint          width,
                  uint          height,
                  const TSurface * src,
                  int           srcX,
                  int           srcY
                )
```

16bpp blitter. Copies a rectangle from one surface to another.

## Parameters:

- dst* Destination surface.
- dstX* Left coord of rectangle on *dst*.
- dstY* Top coord of rectangle on *dst*.
- width* Width of rectangle to blit.
- height* Height of rectangle to blit.
- src* Source surface.
- srcX* Left coord of rectangle on *src*.
- srcY* Top coord of rectangle on *src*.

## Note:

The rectangle will be clipped to both *src* and *dst*.

```
void sbmp8.floodfill ( const TSurface * dst,
                      int           x,
                      int           y,
```

```
    u32           clr  
)
```

Floodfill an area of the same color with new color *clr*.

#### Parameters:

*dst* Destination surface.  
*x* X-coordinate.  
*y* Y-coordinate;  
*clr* Color.

```
void sbmp8_frame ( const TSurface * dst,  
                    int           left,  
                    int           top,  
                    int           right,  
                    int           bottom,  
                    u32           clr  
)
```

Draw a rectangle in 8-bit mode.

#### Parameters:

*dst* Destination surface.  
*left* Left side of rectangle;  
*top* Top side of rectangle.  
*right* Right side of rectangle.  
*bottom* Bottom side of rectangle.  
*clr* Color.

#### Note:

Does normalization, but not bounds checks.

## PONDER: RB in- or exclusive?

```
u32 sbmp8_get_pixel ( const TSurface * src,  
                      int           x,  
                      int           y  
)
```

Get the pixel value of *src* at (*x*, *y*).

```
void sbmp8_hline ( const TSurface * dst,  
                   int           x1,  
                   int           y,  
                   int           x2,  
                   u32          clr  
)
```

Draw a horizontal line on an 8-bit buffer.

### Parameters:

*dst* Destination surface.

*x1* First X-coord.

*y* Y-coord.

*x2* Second X-coord.

*clr* Color.

### Note:

Does normalization, but not bounds checks.

```
void sbmp8_line ( const TSurface * dst,
```

```
    int          x1,  
    int          y1,  
    int          x2,  
    int          y2,  
    u32         clr  
)
```

Draw a line on an 8-bit buffer.

### Parameters:

*dst* Destination surface.  
*x1* First X-coord.  
*y1* First Y-coord.  
*x2* Second X-coord.  
*y2* Second Y-coord.  
*clr* Color.

### Note:

Does normalization, but not bounds checks.

```
void sbmp8_plot ( const TSurface * dst,  
                  int          x,  
                  int          y,  
                  u32         clr  
)
```

Plot a single pixel on a 8-bit buffer.

### Parameters:

*dst* Destination surface.  
*x* X-coord.  
*y* Y-coord.

*clr* Color.

**Note:**

Slow as fuck. Inline plotting functionality if possible.

```
void sbmp8_rect ( const TSurface * dst,  
                  int           left,  
                  int           top,  
                  int           right,  
                  int           bottom,  
                  u32          clr  
                )
```

Draw a rectangle in 8-bit mode.

**Parameters:**

*dst* Destination surface.  
*left* Left side of rectangle;  
*top* Top side of rectangle.  
*right* Right side of rectangle.  
*bottom* Bottom side of rectangle.  
*clr* Color.

**Note:**

Does normalization, but not bounds checks.

```
void sbmp8_vline ( const TSurface * dst,  
                   int           x,  
                   int           y1,  
                   int           y2,  
                   u32          clr
```

)

Draw a vertical line on an 8-bit buffer.

**Parameters:**

*dst* Destination surface.

*x* X-coord.

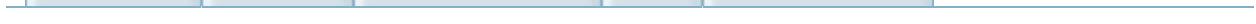
*y1* First Y-coord.

*y2* Second Y-coord.

*clr* Color.

**Note:**

Does normalization, but not bounds checks.

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **4bpp tiled surfaces, column major**

## **[Surface functions]**

## Functions

u32	<b>schr4c_get_pixel</b> (const TSurface *src, int x, int y) <i>Get the pixel value of src at (x, y).</i>
void	<b>schr4c_plot</b> (const TSurface *dst, int x, int y, u32 clr) <i>Plot a single pixel on a 4bpp tiled surface.</i>
void	<b>schr4c_hline</b> (const TSurface *dst, int x1, int y, int x2, u32 clr) <i>Draw a horizontal line on a 4bpp tiled surface.</i>
void	<b>schr4c_vline</b> (const TSurface *dst, int x, int y1, int y2, u32 clr) <i>Draw a vertical line on a 4bpp tiled surface.</i>
void	<b>schr4c_line</b> (const TSurface *dst, int x1, int y1, int x2, int y2, u32 clr) <i>Draw a line on a 4bpp tiled surface.</i>
void	<b>schr4c_rect</b> (const TSurface *dst, int left, int top, int right, int bottom, u32 clr) <i>Render a rectangle on a 4bpp tiled canvas.</i>
void	<b>schr4c_frame</b> (const TSurface *dst, int left, int top, int right, int bottom, u32 clr) <i>Draw a rectangle on a 4bpp tiled surface.</i>
void	<b>schr4c_blt</b> (const TSurface *dst, int dstX, int dstY, uint width, uint height, const TSurface *src, int srcX, int srcY) <i>Blitter for 4bpp tiled surfaces. Copies a rectangle from one surface to another.</i>
void	<b>schr4c_floodfill</b> (const TSurface *dst, int x, int y, u32 clr) <i>Floodfill an area of the same color with new color clr.</i>
void	<b>schr4c_prep_map</b> (const TSurface *srf, u16 *map, u16 se0) <i>Prepare a screen-entry map for use with chr4.</i>
u32 *	<b>schr4c_get_ptr</b> (const TSurface *srf, int x, int y) <i>Special pointer getter for chr4: start of in-tile line.</i>
INLINE void	<b>_schr4c_plot</b> (const TSurface *dst, int x, int y, u32 clr) <i>Plot a single pixel on a 4bpp tiled,col-jamor surface; inline version.</i>
INLINE u32	<b>_schr4c_get_pixel</b> (const TSurface *src, int x, int y) <i>Get the pixel value of src at (x, y); inline version.</i>

## Detailed Description

A (4bpp) tiled surface is formed when each tilemap entry references a unique tile (this is done by [schr4c\\_prep\\_map\(\)](#)). The pixels on the tiles will then uniquely map onto pixels on the screen.

There are two ways of map-layout here: row-major indexing and column-major indexing. The difference is that tile 1 is to the right of tile 0 in the former, but under it in the latter.

30x20t screen:

Row-major:

0	1	2	3	...
30	31	32	33	...
60	61	62	63	...

Column-major:

0	20	40	60	...
1	21	41	61	...
2	22	42	62	...

With 4bpp tiles, the column-major version makes the y coordinate match up nicely with successive words. For this reason, column-major is preferred over row-major.

---

# Function Documentation

```
void schr4c.blit ( const TSurface * dst,
                   int           dstX,
                   int           dstY,
                   uint          width,
                   uint          height,
                   const TSurface * src,
                   int           srcX,
                   int           srcY
)
```

Blitter for 4bpp tiled surfaces. Copies a rectangle from one surface to another.

## Parameters:

*dst* Destination surface.  
*dstX* Left coord of rectangle on *dst*.  
*dstY* Top coord of rectangle on *dst*.  
*width* Width of rectangle to blit.  
*height* Height of rectangle to blit.  
*src* Source surface.  
*srcX* Left coord of rectangle on *src*.  
*srcY* Top coord of rectangle on *src*.

## Note:

The rectangle will be clipped to both *src* and *dst*.

```
void schr4c_floodfill ( const TSurface * dst,
                        int           x,
```

```
    int          y,  
    u32         clr  
)
```

Floodfill an area of the same color with new color *clr*.

### Parameters:

*dst* Destination surface.  
*x* X-coordinate.  
*y* Y-coordinate;  
*clr* Color.

### Note:

This routines is probably very, very slow.

```
void schr4c_frame ( const TSurface * dst,  
                     int           left,  
                     int           top,  
                     int           right,  
                     int           bottom,  
                     u32          clr  
)
```

Draw a rectangle on a 4bpp tiled surface.

### Parameters:

*dst* Destination surface.  
*left* Left side of rectangle;  
*top* Top side of rectangle.  
*right* Right side of rectangle.  
*bottom* Bottom side of rectangle.  
*clr* Color.

**Note:**

Does normalization, but not bounds checks.

PONDER: RB in- or exclusive?

```
u32 schr4c_get_pixel ( const TSurface * src,
                      int           x,
                      int           y
                    )
```

Get the pixel value of *src* at (*x*, *y*).

```
void schr4c_hline ( const TSurface * dst,
                     int           x1,
                     int           y,
                     int           x2,
                     u32          clr
                   )
```

Draw a horizontal line on a 4bpp tiled surface.

**Parameters:**

*dst* Destination surface.

*x1* First X-coord.

*y* Y-coord.

*x2* Second X-coord.

*clr* Color.

**Note:**

Does normalization, but not bounds checks.

```
void schr4c_line ( const TSurface * dst,
                    int           x1,
                    int           y1,
                    int           x2,
                    int           y2,
                    u32          clr
)
```

Draw a line on a 4bpp tiled surface.

#### Parameters:

*dst* Destination surface.

*x1* First X-coord.

*y1* First Y-coord.

*x2* Second X-coord.

*y2* Second Y-coord.

*clr* Color.

#### Note:

Does normalization, but not bounds checks.

```
void schr4c_plot ( const TSurface * dst,
                    int           x,
                    int           y,
                    u32          clr
)
```

Plot a single pixel on a 4bpp tiled surface.

#### Parameters:

*dst* Destination surface.

*x* X-coord.  
*y* Y-coord.  
*clr* Color.

**Note:**

Fairly slow. Inline plotting functionality if possible.

```
void schr4c_prep_map ( const TSurface * srf,  
                      u16 * map,  
                      u16 se0  
)
```

Prepare a screen-entry map for use with chr4.

**Parameters:**

*srf* Surface with size information.  
*map* Screen-blocked map to initialize.  
*se0* Additive base screen-entry.

```
void schr4c_rect ( const TSurface * dst,  
                   int left,  
                   int top,  
                   int right,  
                   int bottom,  
                   u32 clr  
)
```

Render a rectangle on a 4bpp tiled canvas.

**Parameters:**

*dst* Destination surface.

*left* Left side of rectangle;  
*top* Top side of rectangle.  
*right* Right side of rectangle.  
*bottom* Bottom side of rectangle.  
*clr* Color-index.

```
void schr4c_vline ( const TSurface * dst,
                     int           x,
                     int           y1,
                     int           y2,
                     u32          clr
                   )
```

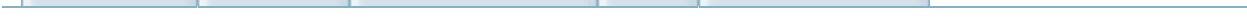
Draw a vertical line on a 4bpp tiled surface.

### Parameters:

*dst* Destination surface.  
*x* X-coord.  
*y1* First Y-coord.  
*y2* Second Y-coord.  
*clr* Color.

### Note:

Does normalization, but not bounds checks.

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **4bpp tiled surfaces, row major**

## **[Surface functions]**

## Functions

u32	<b>schr4r_get_pixel</b> (const TSurface *src, int x, int y) <i>Get the pixel value of src at (x, y).</i>
void	<b>schr4r_plot</b> (const TSurface *dst, int x, int y, u32 clr) <i>Plot a single pixel on a 4bpp tiled surface.</i>
void	<b>schr4r_hline</b> (const TSurface *dst, int x1, int y, int x2, u32 clr) <i>Draw a horizontal line on a 4bpp tiled surface.</i>
void	<b>schr4r_vline</b> (const TSurface *dst, int x, int y1, int y2, u32 clr) <i>Draw a vertical line on a 4bpp tiled surface.</i>
void	<b>schr4r_line</b> (const TSurface *dst, int x1, int y1, int x2, int y2, u32 clr) <i>Draw a line on a 4bpp tiled surface.</i>
void	<b>schr4r_rect</b> (const TSurface *dst, int left, int top, int right, int bottom, u32 clr) <i>Render a rectangle on a tiled canvas.</i>
void	<b>schr4r_frame</b> (const TSurface *dst, int left, int top, int right, int bottom, u32 clr) <i>Draw a rectangle on a 4bpp tiled surface.</i>
void	<b>schr4r_prep_map</b> (const TSurface *srf, u16 *map, u16 se0) <i>Prepare a screen-entry map for use with chr4.</i>
u32 *	<b>schr4r_get_ptr</b> (const TSurface *srf, int x, int y) <i>Special pointer getter for chr4: start of in-tile line.</i>
INLINE void	<b>_schr4r_plot</b> (const TSurface *dst, int x, int y, u32 clr) <i>Plot a single pixel on a 4bpp tiled, row-major surface; inline version.</i>
INLINE u32	<b>_schr4r_get_pixel</b> (const TSurface *src, int x, int y) <i>Get the pixel value of src at (x, y); inline version.</i>

## Detailed Description

A (4bpp) tiled surface is formed when each tilemap entry references a unique tile (this is done by [schr4r\\_prep\\_map\(\)](#)). The pixels on the tiles will then uniquely map onto pixels on the screen.

There are two ways of map-layout here: row-major indexing and column-major indexing. The difference is that tile 1 is to the right of tile 0 in the former, but under it in the latter.

30x20t screen:

Row-major:

0	1	2	3	...
30	31	32	33	...
60	61	62	63	...

Column-major:

0	20	40	60	...
1	21	41	61	...
2	22	42	62	...

With 4bpp tiles, the column-major version makes the y coordinate match up nicely with successive words. For this reason, column-major is preferred over row-major.

---

# Function Documentation

```
void schr4r_frame ( const TSurface * dst,
                     int           left,
                     int           top,
                     int           right,
                     int           bottom,
                     u32          clr
                   )
```

Draw a rectangle on a 4bpp tiled surface.

## Parameters:

*dst*      Destination surface.  
*left*     Left side of rectangle;  
*top*     Top side of rectangle.  
*right*    Right side of rectangle.  
*bottom*   Bottom side of rectangle.  
*clr*      Color.

## Note:

Does normalization, but not bounds checks.

PONDER: RB in- or exclusive?

```
u32 schr4r_get_pixel ( const TSurface * src,
                      int           x,
                      int           y
                    )
```

Get the pixel value of *src* at (*x*, *y*).

```
void schr4r_hline ( const TSurface * dst,
                     int           x1,
                     int           y,
                     int           x2,
                     u32          clr
)
```

Draw a horizontal line on a 4bpp tiled surface.

### Parameters:

*dst* Destination surface.  
*x1* First X-coord.  
*y* Y-coord.  
*x2* Second X-coord.  
*clr* Color.

### Note:

Does normalization, but not bounds checks.

```
void schr4r_line ( const TSurface * dst,
                    int           x1,
                    int           y1,
                    int           x2,
                    int           y2,
                    u32          clr
)
```

Draw a line on a 4bpp tiled surface.

### Parameters:

*dst* Destination surface.  
*x1* First X-coord.  
*y1* First Y-coord.  
*x2* Second X-coord.  
*y2* Second Y-coord.  
*clr* Color.

### Note:

Does normalization, but not bounds checks.

```
void schr4r_plot ( const TSurface * dst,
                    int x,
                    int y,
                    u32 clr
                )
```

Plot a single pixel on a 4bpp tiled surface.

### Parameters:

*dst* Destination surface.  
*x* X-coord.  
*y* Y-coord.  
*clr* Color.

### Note:

Slow as fuck. Inline plotting functionality if possible.

```
void schr4r_prep_map ( const TSurface * srf,
                       u16 * map,
```

```
    u16      se0  
    )
```

Prepare a screen-entry map for use with chr4.

#### Parameters:

*srf* Surface with size information.  
*map* Screen-blocked map to initialize.  
*se0* Additive base screen-entry.

```
void schr4r_rect( const TSurface * dst,  
                  int           left,  
                  int           top,  
                  int           right,  
                  int           bottom,  
                  u32          clr  
)
```

Render a rectangle on a tiled canvas.

#### Parameters:

*dst* Destination surface.  
*left* Left side of rectangle;  
*top* Top side of rectangle.  
*right* Right side of rectangle.  
*bottom* Bottom side of rectangle.  
*clr* Color-index. Octupled if > 16.

#### Note:

For a routine like this you can strive for programmer sanity or speed. This is for speed. Except for very small rects, this is between 5x and 300x faster than the trivial

version. Here's how it works: | c | +---+ a | d | b +---+ | e |

Boundaries are tile-boundaries;

- If unaligned left : draw A [left,8), update dstD/width
- If unaligned right: draw B [right&~7,right), Adjust dstD/width
- If width>0
  - if unaligned top : draw C in ix/iy loop. Adjust dstD/height
  - If height>8 : draw D in memset32 blocks, adjust height
  - Final sets : draw E in ix/iy loop

```
void schr4r_vline ( const TSurface * dst,
                     int           x,
                     int           y1,
                     int           y2,
                     u32          clr
                   )
```

Draw a vertical line on a 4bpp tiled surface.

### Parameters:

*dst* Destination surface.  
*x* X-coord.  
*y1* First Y-coord.  
*y2* Second Y-coord.  
*clr* Color.

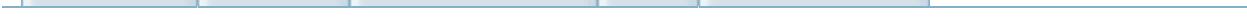
### Note:

Does normalization, but not bounds checks.

---

---

Generated on Mon Aug 25 17:03:57 2008 for libtonc by  doxygen 1.5.3

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **Colors**

## [[Video](#)]

## Base Color constants

```
#define CLR_BLACK 0x0000
#define CLR_RED 0x001F
#define CLR_LIME 0x03E0
#define CLR_YELLOW 0x03FF
#define CLR_BLUE 0x7C00
#define CLR_MAG 0x7C1F
#define CLR_CYAN 0x7FE0
#define CLR_WHITE 0xFFFF
```

## Additional colors

```
#define CLR_DEAD 0xDEAD
#define CLR_MAROON 0x0010
#define CLR_GREEN 0x0200
#define CLR_OLIVE 0x0210
#define CLR_ORANGE 0x021F
#define CLR_NAVY 0x4000
#define CLR_PURPLE 0x4010
#define CLR_TEAL 0x4200
#define CLR_GRAY 0x4210
#define CLR_MEDGRAY 0x5294
#define CLR_SILVER 0x6318
#define CLR_MONEYGREEN 0x6378
#define CLR_FUCHSIA 0x7C1F
#define CLR_SKYBLUE 0x7B34
#define CLR_CREAM 0x7BFF
```

## Defines

```
#define CLR_MASK 0x001F
#define RED_MASK 0x001F
#define RED_SHIFT 0
#define GREEN_MASK 0x03E0
#define GREEN_SHIFT 5
#define BLUE_MASK 0x7C00
#define BLUE_SHIFT 10
```

# Functions

void	<b>clr_rotate</b> ( <b>COLOR</b> *clrs, uint nclrs, int ror) <i>Rotate nclrs colors at clrs to the right by ror.</i>
void	<b>clr_blend</b> (const <b>COLOR</b> *srca, const <b>COLOR</b> *srcb, <b>COLOR</b> *dst, u32 nclrs, u32 alpha) <i>Blends color arrays srca and srcb into dst.</i>
void	<b>clr_fade</b> (const <b>COLOR</b> *src, <b>COLOR</b> clr, <b>COLOR</b> *dst, u32 nclrs, u32 alpha) <i>Fades color arrays srca to clr into dst.</i>
void	<b>clr_grayscale</b> ( <b>COLOR</b> *dst, const <b>COLOR</b> *src, uint nclrs) <i>Transform colors to grayscale.</i>
void	<b>clr_rgbscale</b> ( <b>COLOR</b> *dst, const <b>COLOR</b> *src, uint nclrs, <b>COLOR</b> clr) <i>Transform colors to an rgb-scale.</i>
void	<b>clr_adj_brightness</b> ( <b>COLOR</b> *dst, const <b>COLOR</b> *src, uint nclrs, <b>FIXED</b> bright) <i>Adjust brightness by bright.</i>
void	<b>clr_adj_contrast</b> ( <b>COLOR</b> *dst, const <b>COLOR</b> *src, uint nclrs, <b>FIXED</b> contrast) <i>Adjust contrast by contrast.</i>
void	<b>clr_adj_intensity</b> ( <b>COLOR</b> *dst, const <b>COLOR</b> *src, uint nclrs, <b>FIXED</b> intensity) <i>Adjust intensity by intensity.</i>
void	<b>pal_gradient</b> ( <b>COLOR</b> *pal, int first, int last) <i>Create a gradient between pal[first] and pal[last].</i>
void	<b>pal_gradient_ex</b> ( <b>COLOR</b> *pal, int first, int last, <b>COLOR</b> clr_first, <b>COLOR</b> clr_last) <i>Create a gradient between pal[first] and pal[last].</i>
IWRAM_CODE void	<b>clr_blend_fast</b> ( <b>COLOR</b> *srca, <b>COLOR</b> *srcb, <b>COLOR</b> *dst, uint nclrs, u32 alpha) <i>Blends color arrays srca and srcb into dst.</i>
IWRAM_CODE void	<b>clr_fade_fast</b> ( <b>COLOR</b> *src, <b>COLOR</b> clr, <b>COLOR</b> *dst, uint nclrs, u32 alpha)

	<i>Fades color arrays srca to clr into dst.</i>
INLINE COLOR	<b>RGB15</b> (int red, int green, int blue) <i>Create a 15bit BGR color.</i>
INLINE COLOR	<b>RGB15_SAFE</b> (int red, int green, int blue) <i>Create a 15bit BGR color, with proper masking of R,G,B components.</i>
INLINE COLOR	<b>RGB8</b> (u8 red, u8 green, u8 blue) <i>Create a 15bit BGR color, using 8bit components.</i>

# Function Documentation

```
void clr_adj_brightness ( COLOR *      dst,  
                         const COLOR * src,  
                         uint          nclrs,  
                         FIXED         bright  
)
```

Adjust brightness by *bright*.

Operation: color= color+dB;

## Parameters:

*dst* Destination color array  
*src* Source color array.  
*nclrs* Number of colors.  
*bright* Brightness difference, dB (in .8f)

## Note:

Might be faster if preformed via lut.

```
void clr_adj_contrast ( COLOR *      dst,  
                        const COLOR * src,  
                        uint          nclrs,  
                        FIXED         contrast  
)
```

Adjust contrast by *contrast*.

Operation: color = color\*(1+dC) - MAX\*dC/2

### Parameters:

*dst* Destination color array  
*src* Source color array.  
*nclrs* Number of colors.  
*contrast* Contrast difference in .8f

### Note:

Might be faster if preformed via lut.

```
void clr_adj_intensity ( COLOR *      dst,
                        const COLOR *  src,
                        uint          nclrs,
                        FIXED         intensity
)
```

Adjust intensity by *intensity*.

Operation: color = (1+di)\*color.

### Parameters:

*dst* Destination color array  
*src* Source color array.  
*nclrs* Number of colors.  
*intensity* Intensity difference, di (in .8f)

### Note:

Might be faster if preformed via lut.

```
void clr_blend ( const COLOR * srca,
                  const COLOR * srcb,
                  COLOR *       dst,
```

```
    uint      nclrs,  
    u32      alpha  
)
```

Blends color arrays *srca* and *srcb* into *dst*.

Specific transitional blending effects can be created by making a 'target' color array with other routines, then using *alpha* to morph into it.

### Parameters:

*srca* Source array A.  
*srcb* Source array B  
*dst* Destination array.  
*nclrs* Number of colors.  
*alpha* Blend weight (range: 0-32). 0 Means full *srca*

```
IWRAM_CODE void clr_blend_fast ( COLOR * srca,  
                                COLOR * srcb,  
                                COLOR * dst,  
                                uint      nclrs,  
                                u32      alpha  
)
```

Blends color arrays *srca* and *srcb* into *dst*.

### Parameters:

*srca* Source array A.  
*srcb* Source array B  
*dst* Destination array.  
*nclrs* Number of colors.  
*alpha* Blend weight (range: 0-32).

**Note:**

Handles 2 colors per loop. Very fast.

```
void clr_fade ( const COLOR * src,  
                 COLOR      clr,  
                 COLOR *    dst,  
                 uint       nclrs,  
                 u32        alpha  
             )
```

Fades color arrays *src* to *clr* into *dst*.

**Parameters:**

*src* Source array.  
*clr* Final color (at alpha=32).  
*dst* Destination array.  
*nclrs* Number of colors.  
*alpha* Blend weight (range: 0-32). 0 Means full *src*

```
IWRAM_CODE void clr_fade_fast ( COLOR * src,  
                               COLOR   clr,  
                               COLOR * dst,  
                               uint    nclrs,  
                               u32     alpha  
                         )
```

Fades color arrays *src* to *clr* into *dst*.

**Parameters:**

*src* Source array.

*clr* Final color (at alpha=32).  
*dst* Destination array.  
*nclrs* Number of colors.  
*alpha* Blend weight (range: 0-32).

**Note:**

Handles 2 colors per loop. Very fast.

```
void clr_grayscale( COLOR *      dst,
                     const COLOR * src,
                     uint          nclrs
)
```

Transform colors to grayscale.

**Parameters:**

*dst* Destination color array  
*src* Source color array.  
*nclrs* Number of colors.

```
void clr_rgbscale( COLOR *      dst,
                    const COLOR * src,
                    uint          nclrs,
                    COLOR        clr
)
```

Transform colors to an rgb-scale.

*clr* indicates a color vector in RGB-space. Each source color is converted to a brightness value (i.e., grayscale) and then mapped onto that color vector. A grayscale is a special case

of this, using a color with R=G=B.

### Parameters:

*dst* Destination color array  
*src* Source color array.  
*ncls* Number of colors.  
*clr* Destination color vector.

```
void clr_rotate ( COLOR * clrs,
                  uint      ncls,
                  int       ror
                )
```

Rotate *ncls* colors at *clrs* to the right by *ror*.

### Note:

I can't help but think there's a faster way ... I just can't see it atm.

```
void pal_gradient ( COLOR * pal,
                     int      first,
                     int      last
                   )
```

Create a gradient between *pal[first]* and *pal[last]*.

### Parameters:

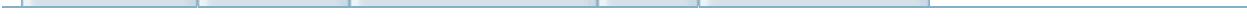
*pal* Palette to work on.  
*first* First index of gradient.  
*last* Last index of gradient.

```
void pal_gradient_ex( COLOR * pal,  
                      int      first,  
                      int      last,  
                      COLOR   clr_first,  
                      COLOR   clr_last  
)
```

Create a gradient between *pal[first]* and *pal[last]*.

### Parameters:

- pal*      Palette to work on.
- first*     First index of gradient.
- last*     Last index of gradient.
- clr\_first* Color of first index.
- clr\_last* Color of last index.

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# Tiled Backgrounds

[[Video](#)]

## Defines

```
#define CBB_CLEAR(cbb) memset32(&tile_mem[cbb], 0, CBB_SIZE/4)
#define SBB_CLEAR(sbb) memset32(&se_mem[sbb], 0, SBB_SIZE/4)
#define SBB_CLEAR_ROW(sbb, row) memset32(&se_mem[sbb][(row)*32], 0, 32/2)
#define __BG_TYPES ((0x0C7F<<16)|(0x0C40))
#define BG_IS_AFFINE(n) ( (__BG_TYPES>>(4*(REG_DISPCNT&7)+(n)))&1 )
#define BG_IS_AVAIL(n) ( (__BG_TYPES>>(4*(REG_DISPCNT&7)+(n)+16))&1 )
```

## Functions

INLINE void	<b>se_fill</b> (SCR_ENTRY *sbb, SCR_ENTRY se) <i>Fill screenblock sbb with se.</i>
INLINE void	<b>se_plot</b> (SCR_ENTRY *sbb, int x, int y, SCR_ENTRY se) <i>Plot a screen entry at (x,y) of screenblock sbb.</i>
INLINE void	<b>se_rect</b> (SCR_ENTRY *sbb, int left, int top, int right, int bottom, SCR_ENTRY se) <i>Fill a rectangle on sbb with se.</i>
INLINE void	<b>se_frame</b> (SCR_ENTRY *sbb, int left, int top, int right, int bottom, SCR_ENTRY se) <i>Create a border on sbb with se.</i>
void	<b>se_window</b> (SCR_ENTRY *sbb, int left, int top, int right, int bottom, SCR_ENTRY se0) <i>Create a framed rectangle.</i>
void	<b>se_hline</b> (SCR_ENTRY *sbb, int x0, int x1, int y, SCR_ENTRY se)
void	<b>se_vline</b> (SCR_ENTRY *sbb, int x, int y0, int y1, SCR_ENTRY se)
INLINE void	<b>bg_aff_set</b> (BG_AFFINE *bgaff, FIXED pa, FIXED pb, FIXED pc, FIXED pd) <i>Set the elements of an bg affine matrix.</i>
INLINE void	<b>bg_aff_identity</b> (BG_AFFINE *bgaff) <i>Set an bg affine matrix to the identity matrix.</i>
INLINE void	<b>bg_aff_scale</b> (BG_AFFINE *bgaff, FIXED sx, FIXED sy) <i>Set an bg affine matrix for scaling.</i>
INLINE void	<b>bg_aff_shearx</b> (BG_AFFINE *bgaff, FIXED hx)
INLINE void	<b>bg_aff_sheary</b> (BG_AFFINE *bgaff, FIXED hy)
void	<b>bg_aff_rotate</b> (BG_AFFINE *bgaff, u16 alpha) <i>Set bg matrix to counter-clockwise rotation.</i>
void	<b>bg_aff_rotscale</b> (BG_AFFINE *bgaff, int sx, int sy, u16 alpha) <i>Set bg matrix to 2d scaling, then counter-clockwise rotation.</i>
void	<b>bg_aff_premul</b> (BG_AFFINE *dst, const BG_AFFINE *src) <i>Pre-multiply dst by src: D = S*D.</i>
void	<b>bg_aff_postmul</b> (BG_AFFINE *dst, const BG_AFFINE *src)

	<i>Post-multiply dst by src: D= D*S.</i>
void	<b>bg_aff_rotscale2</b> ( <b>BG_AFFINE</b> *bgaff, const <b>AFF_SRC</b> *as) <i>Set bg matrix to 2d scaling, then counter-clockwise rotation.</i>
void	<b>bg_rotscale_ex</b> ( <b>BG_AFFINE</b> *bgaff, const <b>AFF_SRC_EX</b> *asx) <i>Set bg affine matrix to a rot/scale around an arbitrary point.</i>

---

## Define Documentation

```
#define CBB_CLEAR( cbb )    memset32( &tile_mem[ cbb ], 0, CBB_
```

---

# Function Documentation

```
void bg_aff_rotate ( BG_AFFINE * bgaff,  
                      u16           alpha  
                    )
```

Set bg matrix to counter-clockwise rotation.

## Parameters:

*bgaff* Object affine struct to set.  
*alpha* CCW angle. full-circle is 10000h.

```
void bg_aff_rotscale ( BG_AFFINE * bgaff,  
                       int            sx,  
                       int            sy,  
                       u16           alpha  
                     )
```

Set bg matrix to 2d scaling, then counter-clockwise rotation.

## Parameters:

*bgaff* Object affine struct to set.  
*sx* Horizontal scale (zoom). .8 fixed point.  
*sy* Vertical scale (zoom). .8 fixed point.  
*alpha* CCW angle. full-circle is 10000h.

```
void bg_aff_rotscale2 ( BG_AFFINE *      bgaff,  
                        const AFF_SRC * as  
                      )
```

Set bg matrix to 2d scaling, then counter-clockwise rotation.

### Parameters:

*bgaff* Object affine struct to set.  
*as* Struct with scales and angle.

```
void bg_rotscale_ex( BG_AFFINE *          bgaff,  
                      const AFF_SRC_EX * asx  
                    )
```

Set bg affine matrix to a rot/scale around an arbitrary point.

Rotate and scale round an arbitrary point using the asx data.

### Parameters:

*bgaff* BG affine data to set.  
*asx* Affine source data: screen and texture origins, scales and angle.

```
void se_window( SCR_ENTRY * sbb,  
                int           left,  
                int           top,  
                int           right,  
                int           bottom,  
                SCR_ENTRY    se0  
              )
```

Create a framed rectangle.

In contrast to [se\\_frame\(\)](#), se\_frame\_ex() uses nine tiles starting at *se0* for the frame, which indicate the borders and

center for the window.

**Note:**

Rectangle is nor normalized.

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# Bitmaps

## [Video]

## Generic 8bpp bitmaps

void	<b>bmp8_plot</b> (int x, int y, u32 clr, void *dstBase, uint dstP) <i>Plot a single pixel on a 8-bit buffer.</i>
void	<b>bmp8_hline</b> (int x1, int y, int x2, u32 clr, void *dstBase, uint dstP) <i>Draw a horizontal line on an 8bit buffer.</i>
void	<b>bmp8_vline</b> (int x, int y1, int y2, u32 clr, void *dstBase, uint dstP) <i>Draw a vertical line on an 8bit buffer.</i>
void	<b>bmp8_line</b> (int x1, int y1, int x2, int y2, u32 clr, void *dstBase, uint dstP) <i>Draw a line on an 8bit buffer.</i>
void	<b>bmp8_rect</b> (int left, int top, int right, int bottom, u32 clr, void *dstBase, uint dstP) <i>Draw a rectangle in 8bit mode; internal routine.</i>
void	<b>bmp8_frame</b> (int left, int top, int right, int bottom, u32 clr, void *dstBase, uint dstP) <i>Draw a rectangle in 8bit mode; internal routine.</i>

## Generic 16bpp bitmaps

void	<b>bmp16_plot</b> (int x, int y, u32 clr, void *dstBase, uint dstP) <i>Plot a single pixel on a 16-bit buffer.</i>
void	<b>bmp16_hline</b> (int x1, int y, int x2, u32 clr, void *dstBase, uint dstP) <i>Draw a horizontal line on an 16bit buffer.</i>
void	<b>bmp16_vline</b> (int x, int y1, int y2, u32 clr, void *dstBase, uint dstP) <i>Draw a vertical line on an 16bit buffer.</i>
void	<b>bmp16_line</b> (int x1, int y1, int x2, int y2, u32 clr, void *dstBase, uint dstP) <i>Draw a line on an 16bit buffer.</i>
void	<b>bmp16_rect</b> (int left, int top, int right, int bottom, u32 clr, void *dstBase, uint dstP) <i>Draw a rectangle in 16bit mode; internal routine.</i>
void	<b>bmp16_frame</b> (int left, int top, int right, int bottom, u32 clr, void *dstBase, uint dstP) <i>Draw a rectangle in 16bit mode; internal routine.</i>

## mode 3

INLINE void	<b>m3_fill</b> ( <b>COLOR</b> clr) <i>Fill the mode 3 background with color clr.</i>
INLINE void	<b>m3_plot</b> (int x, int y, <b>COLOR</b> clr) <i>Plot a single clr colored pixel in mode 3 at (x, y).</i>
INLINE void	<b>m3_hline</b> (int x1, int y, int x2, <b>COLOR</b> clr) <i>Draw a clr colored horizontal line in mode 3.</i>
INLINE void	<b>m3_vline</b> (int x, int y1, int y2, <b>COLOR</b> clr) <i>Draw a clr colored vertical line in mode 3.</i>
INLINE void	<b>m3_line</b> (int x1, int y1, int x2, int y2, <b>COLOR</b> clr) <i>Draw a clr colored line in mode 3.</i>
INLINE void	<b>m3_rect</b> (int left, int top, int right, int bottom, <b>COLOR</b> clr) <i>Draw a clr colored rectangle in mode 3.</i>
INLINE void	<b>m3_frame</b> (int left, int top, int right, int bottom, <b>COLOR</b> clr) <i>Draw a clr colored frame in mode 3.</i>
#define	<b>M3_CLEAR()</b> <code>memset32(vid_mem, 0, M3_SIZE/4)</code> <i>Fill the mode 3 background with color clr.</i>

## mode 4

INLINE void	<b>m4_fill</b> (u8 clrid) <i>Fill the current mode 4 backbuffer with clrid.</i>
INLINE void	<b>m4_plot</b> (int x, int y, u8 clrid) <i>Plot a clrid pixel on the current mode 4 backbuffer.</i>
INLINE void	<b>m4_hline</b> (int x1, int y, int x2, u8 clrid) <i>Draw a clrid colored horizontal line in mode 4.</i>
INLINE void	<b>m4_vline</b> (int x, int y1, int y2, u8 clrid) <i>Draw a clrid colored vertical line in mode 4.</i>
INLINE void	<b>m4_line</b> (int x1, int y1, int x2, int y2, u8 clrid) <i>Draw a clrid colored line in mode 4.</i>
INLINE void	<b>m4_rect</b> (int left, int top, int right, int bottom, u8 clrid) <i>Draw a clrid colored rectangle in mode 4.</i>
INLINE void	<b>m4_frame</b> (int left, int top, int right, int bottom, u8 clrid) <i>Draw a clrid colored frame in mode 4.</i>
#define	<b>M4_CLEAR()</b> <code>memset32(vid_page, 0, M4_SIZE/4)</code> <i>Fill the current mode 4 backbuffer with clrid.</i>

## mode 5

INLINE void	<b>m5_fill</b> ( <b>COLOR</b> clr) <i>Fill the current mode 5 backbuffer with clr.</i>
INLINE void	<b>m5_plot</b> (int x, int y, <b>COLOR</b> clr) <i>Plot a clr'd pixel on the current mode 5 backbuffer.</i>
INLINE void	<b>m5_hline</b> (int x1, int y, int x2, <b>COLOR</b> clr) <i>Draw a clr colored horizontal line in mode 5.</i>
INLINE void	<b>m5_vline</b> (int x, int y1, int y2, <b>COLOR</b> clr) <i>Draw a clr colored vertical line in mode 5.</i>
INLINE void	<b>m5_line</b> (int x1, int y1, int x2, int y2, <b>COLOR</b> clr) <i>Draw a clr colored line in mode 5.</i>
INLINE void	<b>m5_rect</b> (int left, int top, int right, int bottom, <b>COLOR</b> clr) <i>Draw a clr colored rectangle in mode 5.</i>
INLINE void	<b>m5_frame</b> (int left, int top, int right, int bottom, <b>COLOR</b> clr) <i>Draw a clr colored frame in mode 5.</i>
#define	<b>M5_CLEAR()</b> <code>memset32(vid_page, 0, M5_SIZE/4)</code> <i>Fill the current mode 5 backbuffer with clr.</i>

## Detailed Description

Basic functions for dealing with bitmapped graphics.

### Deprecated:

The bmp8/bmp16 functions have been superceded by the surface functions (sbmp8/sbmp16) for the most part. The former group has been kept mostly for reference purposes.

---

# Function Documentation

```
void bmp16_frame( int    left,
                  int    top,
                  int    right,
                  int    bottom,
                  u32   clr,
                  void * dstBase,
                  uint   dstP
                )
```

Draw a rectangle in 16bit mode; internal routine.

## Parameters:

*left* Left side of rectangle;  
*top* Top side of rectangle.  
*right* Right side of rectangle.  
*bottom* Bottom side of rectangle.  
*clr* Color.  
*dstBase* Canvas pointer.  
*dstP* Canvas pitch in bytes

## Note:

Does normalization, but not bounds checks.

PONDER: RB in- or exclusive?

```
void bmp16_hline( int    x1,
                  int    y,
                  int    x2,
```

```
    u32    clr,  
    void * dstBase,  
    uint   dstP  
)
```

Draw a horizontal line on an 16bit buffer.

#### Parameters:

*x1* First X-coord.  
*y* Y-coord.  
*x2* Second X-coord.  
*clr* Color.  
*dstBase* Canvas pointer (halfword-aligned plz).  
*dstP* Canvas pitch in bytes.

#### Note:

Does normalization, but not bounds checks.

```
void bmp16_line ( int    x1,  
                  int    y1,  
                  int    x2,  
                  int    y2,  
                  u32    clr,  
                  void * dstBase,  
                  uint   dstP  
)
```

Draw a line on an 16bit buffer.

#### Parameters:

*x1* First X-coord.  
*y1* First Y-coord.

*x2*      Second X-coord.  
*y2*      Second Y-coord.  
*clr*      Color.  
*dstBase* Canvas pointer (halfword-aligned plz).  
*dstP*      Canvas pitch in bytes.

**Note:**

Does normalization, but not bounds checks.

```
void bmp16_plot( int    x,
                  int    y,
                  u32    clr,
                  void * dstBase,
                  uint   dstP
                )
```

Plot a single pixel on a 16-bit buffer.

**Parameters:**

*x*      X-coord.  
*y*      Y-coord.  
*clr*      Color.  
*dstBase* Canvas pointer (halfword-aligned plz).  
*dstP*      Canvas pitch in bytes.

**Note:**

Slow as fuck. Inline plotting functionality if possible.

```
void bmp16_rect( int    left,
                  int    top,
                  int    right,
```

```
    int    bottom,  
    u32    clr,  
    void * dstBase,  
    uint   dstP  
)
```

Draw a rectangle in 16bit mode; internal routine.

### Parameters:

*left* Left side of rectangle;  
*top* Top side of rectangle.  
*right* Right side of rectangle.  
*bottom* Bottom side of rectangle.  
*clr* Color.  
*dstBase* Canvas pointer.  
*dstP* Canvas pitch in bytes

### Note:

Does normalization, but not bounds checks.

```
void bmp16_vline ( int    x,  
                   int    y1,  
                   int    y2,  
                   u32    clr,  
                   void * dstBase,  
                   uint   dstP  
)
```

Draw a vertical line on an 16bit buffer.

### Parameters:

*x* X-coord.

*y1* First Y-coord.  
*y2* Second Y-coord.  
*clr* Color.  
*dstBase* Canvas pointer (halfword-aligned plz).  
*dstP* Canvas pitch in bytes.

**Note:**

Does normalization, but not bounds checks.

```
void bmp8_frame ( int    left,
                  int    top,
                  int    right,
                  int    bottom,
                  u32    clr,
                  void * dstBase,
                  uint   dstP
                )
```

Draw a rectangle in 8bit mode; internal routine.

**Parameters:**

*left* Left side of rectangle;  
*top* Top side of rectangle.  
*right* Right side of rectangle.  
*bottom* Bottom side of rectangle.  
*clr* Color-index.  
*dstBase* Canvas pointer.  
*dstP* Canvas pitch in bytes

**Note:**

Does normalization, but not bounds checks.

## PONDER: RB in- or exclusive?

```
void bmp8_hline ( int    x1,
                  int    y,
                  int    x2,
                  u32   clr,
                  void * dstBase,
                  uint   dstP
                )
```

Draw a horizontal line on an 8bit buffer.

### Parameters:

*x1* First X-coord.  
*y* Y-coord.  
*x2* Second X-coord.  
*clr* Color index.  
*dstBase* Canvas pointer (halfword-aligned plz).  
*dstP* canvas pitch in bytes.

### Note:

Does normalization, but not bounds checks.

```
void bmp8_line ( int    x1,
                  int    y1,
                  int    x2,
                  int    y2,
                  u32   clr,
                  void * dstBase,
                  uint   dstP
```

)

Draw a line on an 8bit buffer.

### Parameters:

*x1* First X-coord.  
*y1* First Y-coord.  
*x2* Second X-coord.  
*y2* Second Y-coord.  
*clr* Color index.  
*dstBase* Canvas pointer (halfword-aligned plz).  
*dstP* Canvas pitch in bytes.

### Note:

Does normalization, but not bounds checks.

```
void bmp8_plot( int    x,  
                int    y,  
                u32    clr,  
                void * dstBase,  
                uint   dstP  
            )
```

Plot a single pixel on a 8-bit buffer.

### Parameters:

*x* X-coord.  
*y* Y-coord.  
*clr* Color.  
*dstBase* Canvas pointer (halfword-aligned plz).  
*dstP* Canvas pitch in bytes.

**Note:**

Slow as fuck. Inline plotting functionality if possible.

```
void bmp8_rect( int    left,
                int    top,
                int    right,
                int    bottom,
                u32    clr,
                void * dstBase,
                uint   dstP
            )
```

Draw a rectangle in 8bit mode; internal routine.

**Parameters:**

*left* Left side of rectangle;  
*top* Top side of rectangle.  
*right* Right side of rectangle.  
*bottom* Bottom side of rectangle.  
*clr* Color-index.  
*dstBase* Canvas pointer.  
*dstP* Canvas pitch in bytes

**Note:**

Does normalization, but not bounds checks.

```
void bmp8_vline( int    x,
                  int    y1,
                  int    y2,
                  u32    clr,
```

```
    void * dstBase,  
    uint   dstP  
)
```

Draw a vertical line on an 8bit buffer.

### Parameters:

*x* X-coord.  
*y1* First Y-coord.  
*y2* Second Y-coord.  
*clr* Color index.  
*dstBase* Canvas pointer (halfword-aligned plz).  
*dstP* canvas pitch in bytes.

### Note:

Does normalization, but not bounds checks.

```
INLINE void m3_frame ( int      left,  
                      int      top,  
                      int      right,  
                      int      bottom,  
                      COLOR   clr  
)
```

Draw a *clr* colored frame in mode 3.

### Parameters:

*left* Left side, inclusive.  
*top* Top size, inclusive.  
*right* Right size, exclusive.  
*bottom* Bottom size, exclusive.  
*clr* Color.

**Note:**

Normalized, but not clipped.

```
INLINE void m3_rect ( int      left,  
                      int      top,  
                      int      right,  
                      int      bottom,  
                      COLOR   clr  
                    )
```

Draw a *clr* colored rectangle in mode 3.

**Parameters:**

*left* Left side, inclusive.  
*top* Top size, inclusive.  
*right* Right size, exclusive.  
*bottom* Bottom size, exclusive.  
*clr* Color.

**Note:**

Normalized, but not clipped.

```
INLINE void m4_frame ( int left,  
                      int top,  
                      int right,  
                      int bottom,  
                      u8   clrid  
                    )
```

Draw a *clrid* colored frame in mode 4.

### Parameters:

*left*      Left side, inclusive.  
*top*      Top size, inclusive.  
*right*      Right size, exclusive.  
*bottom*      Bottom size, exclusive.  
*clrid*      color index.

### Note:

Normalized, but not clipped.

```
INLINE void m4_rect ( int left,  
                      int top,  
                      int right,  
                      int bottom,  
                      u8 clrid  
                    )
```

Draw a *clrid* colored rectangle in mode 4.

### Parameters:

*left*      Left side, inclusive.  
*top*      Top size, inclusive.  
*right*      Right size, exclusive.  
*bottom*      Bottom size, exclusive.  
*clrid*      color index.

### Note:

Normalized, but not clipped.

```
INLINE void m5_frame ( int left,
```

```
        int      top,  
        int      right,  
        int      bottom,  
        COLOR   clr  
    )
```

Draw a *clr* colored frame in mode 5.

### Parameters:

*left* Left side, inclusive.  
*top* Top size, inclusive.  
*right* Right size, exclusive.  
*bottom* Bottom size, exclusive.  
*clr* Color.

### Note:

Normalized, but not clipped.

```
INLINE void m5_rect ( int      left,  
                      int      top,  
                      int      right,  
                      int      bottom,  
                      COLOR   clr  
    )
```

Draw a *clr* colored rectangle in mode 5.

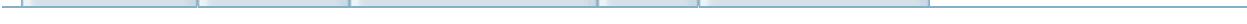
### Parameters:

*left* Left side, inclusive.  
*top* Top size, inclusive.  
*right* Right size, exclusive.  
*bottom* Bottom size, exclusive.

*clr*      Color.

**Note:**

Normalized, but not clipped.

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# Objects

## [[Video](#)]

## Defines

```
#define OAM_CLEAR() memset32(oam_mem, 0, OAM_SIZE/4)
```

# Functions

void	<b>oam_init</b> ( <b>OBJ_ATTR</b> *obj, uint count) <i>Initialize an array of count OBJ_ATTRs with safe values.</i>
INLINE void	<b>oam_copy</b> ( <b>OBJ_ATTR</b> *dst, const <b>OBJ_ATTR</b> *src, uint count) <i>Copies count OAM entries from src to dst.</i>
INLINE <b>OBJ_ATTR</b> *	<b>obj_set_attr</b> ( <b>OBJ_ATTR</b> *obj, u16 a0, u16 a1, u16 a2) <i>Set the attributes of an object.</i>
INLINE void	<b>obj_set_pos</b> ( <b>OBJ_ATTR</b> *obj, int x, int y) <i>Set the position of obj.</i>
INLINE void	<b>obj_hide</b> ( <b>OBJ_ATTR</b> *obj) <i>Hide an object.</i>
INLINE void	<b>obj_unhide</b> ( <b>OBJ_ATTR</b> *obj, u16 mode) <i>Unhide an object.</i>
INLINE const u8 *	<b>obj_get_size</b> (const <b>OBJ_ATTR</b> *obj) <i>Get object's sizes as a byte array.</i>
INLINE int	<b>obj_get_width</b> (const <b>OBJ_ATTR</b> *obj) <i>Get object's width.</i>
INLINE int	<b>obj_get_height</b> (const <b>OBJ_ATTR</b> *obj) <i>Gets object's height.</i>
void	<b>obj_copy</b> ( <b>OBJ_ATTR</b> *dst, const <b>OBJ_ATTR</b> *src, uint count) <i>Copy attributes 0-2 in count OBJ_ATTRs.</i>
void	<b>obj_hide_multi</b> ( <b>OBJ_ATTR</b> *obj, u32 count) <i>Hide an array of OBJ_ATTRs.</i>
void	<b>obj_unhide_multi</b> ( <b>OBJ_ATTR</b> *obj, u16 mode, uint count)
void	<b>obj_aff_copy</b> ( <b>OBJ_AFFINE</b> *dst, const <b>OBJ_AFFINE</b> *src, uint count)
INLINE void	<b>obj_aff_set</b> ( <b>OBJ_AFFINE</b> *oaff, <b>FIXED</b> pa, <b>FIXED</b> pb, <b>FIXED</b> pc, <b>FIXED</b> pd) <i>Set the elements of an object affine matrix.</i>

INLINE void	<b>obj_aff_identity (OBJ_AFFINE *oaff)</b> Set an object affine matrix to the identity matrix.
INLINE void	<b>obj_aff_scale (OBJ_AFFINE *oaff, FIXED sx, FIXED sy)</b> Set an object affine matrix for scaling.
INLINE void	<b>obj_aff_shearx (OBJ_AFFINE *oaff, FIXED hx)</b>
INLINE void	<b>obj_aff_sheary (OBJ_AFFINE *oaff, FIXED hy)</b>
void	<b>obj_aff_rotate (OBJ_AFFINE *oaff, u16 alpha)</b> Set obj matrix to counter-clockwise rotation.
void	<b>obj_aff_rotscale (OBJ_AFFINE *oaff, FIXED sx, FIXED sy, u16 alpha)</b> Set obj matrix to 2d scaling, then counter-clockwise rotation.
void	<b>obj_aff_premul (OBJ_AFFINE *dst, const OBJ_AFFINE *src)</b> Pre-multiply dst by src: $D = S*D$ .
void	<b>obj_aff_postmul (OBJ_AFFINE *dst, const OBJ_AFFINE *src)</b> Post-multiply dst by src: $D= D*S$ .
void	<b>obj_aff_rotscale2 (OBJ_AFFINE *oaff, const AFF_SRC *as)</b> Set obj matrix to 2d scaling, then counter-clockwise rotation.
void	<b>obj_rotscale_ex (OBJ_ATTR *obj, OBJ_AFFINE *oaff, const AFF_SRC_EX *asx)</b> Rot/scale an object around an arbitrary point.
INLINE void	<b>obj_aff_scale_inv (OBJ_AFFINE *oa, FIXED wx, FIXED wy)</b>
INLINE void	<b>obj_aff_rotate_inv (OBJ_AFFINE *oa, u16 theta)</b>
INLINE void	<b>obj_aff_shearx_inv (OBJ_AFFINE *oa, FIXED hx)</b>
INLINE void	<b>obj_aff_sheary_inv (OBJ_AFFINE *oa, FIXED hy)</b>

## **Detailed Description**

---

## Define Documentation

```
#define OAM_CLEAR( )    memset32(oam_mem, 0, OAM_SIZE/4)
```

---

# Function Documentation

```
void obj_aff_rotate( OBJ_AFFINE * oaff,  
                      u16                alpha  
                    )
```

Set obj matrix to counter-clockwise rotation.

## Parameters:

*oaff* Object affine struct to set.  
*alpha* CCW angle. full-circle is 10000h.

```
void obj_aff_rotscale( OBJ_AFFINE * oaff,  
                       FIXED             sx,  
                       FIXED             sy,  
                       u16               alpha  
                     )
```

Set obj matrix to 2d scaling, then counter-clockwise rotation.

## Parameters:

*oaff* Object affine struct to set.  
*sx* Horizontal scale (zoom). .8 fixed point.  
*sy* Vertical scale (zoom). .8 fixed point.  
*alpha* CCW angle. full-circle is 10000h.

```
void obj_aff_rotscale2( OBJ_AFFINE * oaff,  
                        const AFF_SRC * as  
                          )
```

Set obj matrix to 2d scaling, then counter-clockwise rotation.

### Parameters:

*oaff* Object affine struct to set.  
*as* Struct with scales and angle.

```
void obj_rotscale_ex( OBJ_ATTR *          obj,  
                      OBJ_AFFINE *        oaff,  
                      const AFF_SRC_EX * asx  
)
```

Rot/scale an object around an arbitrary point.

Sets up *obj* and *oaff* for rot/scale transformation around an arbitrary point using the *asx* data.

### Parameters:

*obj* Object to set.  
*oaff* Object affine data to set.  
*asx* Affine source data: screen and texture origins, scales and angle.

```
INLINE void obj_unhide( OBJ_ATTR * obj,  
                      u16            mode  
)
```

Unhide an object.

### Parameters:

*obj* Object to unhide.  
*mode* Object mode to unhide to. Necessary because this affects

the affine-ness of the object.

---

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  *1.5.3*

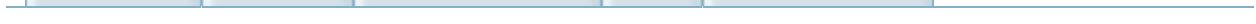
[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# Affine functions

## [[Video](#)]

---

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by  doxygen 1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **Types and attributes**

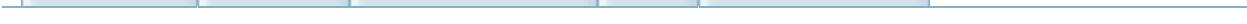
# Modules

- [Type attributes](#)
  - [Primary types](#)
  - [Secondary types](#)
  - [Tertiary types](#)
-

## Detailed Description

---

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  **doxygen** 1.5.3

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# Type attributes

## [Types and attributes]

## Defines

#define	<b>IWRAM_DATA</b> __attribute__((section(".iwrام")))) <i>Put variable in IWRAM (default).</i>
#define	<b>EWRAM_DATA</b> __attribute__((section(".ewram")))) <i>Put variable in EWRAM.</i>
#define	<b>EWRAM_BSS</b> __attribute__((section(".sbss")))) <i>Put non-initialized variable in EWRAM.</i>
#define	<b>IWRAM_CODE</b> __attribute__((section(".iwrام"), long_call)) <i>Put function in IWRAM.</i>
#define	<b>EWRAM_CODE</b> __attribute__((section(".ewram"), long_call)) <i>Put function in EWRAM.</i>
#define	<b>ALIGN(n)</b> __attribute__((aligned(n))) <i>Force a variable to an n-byte boundary.</i>
#define	<b>ALIGN4</b> __attribute__((aligned(4))) <i>Force word alignment.</i>
#define	<b>PACKED</b> __attribute__((packed)) <i>Pack aggregate members.</i>
#define	<b>DEPRECATED</b> __attribute__((deprecated)) <i>Deprecated notice.</i>
#define	<b>INLINE</b> static inline <i>Inline function declarator.</i>

---

# Define Documentation

```
#define ALIGN4 __attribute__((aligned(4)))
```

Force word alignment.

## Note:

In the old days, GCC aggregates were always word aligned. In the EABI environment (devkitPro r19 and higher), they are aligned to their widest member. While technically a good thing, it may cause problems for struct-copies. If you have aggregates that can multiples of 4 in size but don't have word members, consider using this attribute to make struct-copies possible again.

```
#define DEPRECATED __attribute__((deprecated))
```

Deprecated notice.

Indicates that this function/type/variable should not be used anymore. Replacements are (usually) present somewhere as well.

```
#define INLINE static inline
```

Inline function declarator.

'inline' inlines the function when  $-O > 0$  when called, but also creates a body for the function itself 'static' removes the body as well

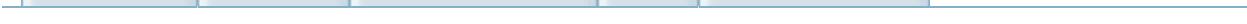
```
#define PACKED __attribute__((packed))
```

Pack aggregate members.

By default, members in aggregates are aligned to their native boundaries. Adding this prevents that. It will slow access though.

---

Generated on Mon Aug 25 17:03:57 2008 for libtonc by  doxygen 1.5.3

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **Primary types**

## [Types and attributes]

# Data Structures

struct	<b>BLOCK</b>
	<i>8-word type for fast struct-copies</i> <a href="#">More...</a>

## Base types

Basic signed and unsigned types for 8, 16, 32 and 64-bit integers.

- s# : signed #-bit integer.
- u#/u{type} : unsigned #-bit integer.
- e{type} : enum'ed #-bit integer.

typedef unsigned char	<b>u8</b>
typedef unsigned char	<b>byte</b>
typedef unsigned char	<b>uchar</b>
typedef unsigned char	<b>echar</b>
typedef unsigned short	<b>u16</b>
typedef unsigned short	<b>hword</b>
typedef unsigned short	<b>ushort</b>
typedef unsigned short	<b>eshort</b>
typedef unsigned int	<b>u32</b>
typedef unsigned int	<b>word</b>
typedef unsigned int	<b>uint</b>
typedef unsigned int	<b>eint</b>
typedef unsigned long long	<b>u64</b>
typedef signed char	<b>s8</b>
typedef signed short	<b>s16</b>
typedef signed int	<b>s32</b>
typedef signed long long	<b>s64</b>

## Volatile types

Volatile types for registers

typedef volatile u8	<b>vu8</b>
typedef volatile u16	<b>vu16</b>
typedef volatile u32	<b>vu32</b>
typedef volatile u64	<b>vu64</b>
typedef volatile s8	<b>vs8</b>
typedef volatile s16	<b>vs16</b>
typedef volatile s32	<b>vs32</b>
typedef volatile s64	<b>vs64</b>

## Const types

Const types for const function parameters

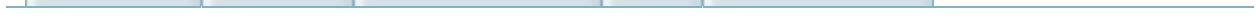
typedef const u8	<b>cu8</b>
typedef const u16	<b>cu16</b>
typedef const u32	<b>cu32</b>
typedef const u64	<b>cu64</b>
typedef const s8	<b>cs8</b>
typedef const s16	<b>cs16</b>
typedef const s32	<b>cs32</b>
typedef const s64	<b>cs64</b>

## Typedefs

typedef const char *const	<b>CSTR</b> <i>Type for consting a string as well as the pointer than points to it.</i>
---------------------------	--

---

Generated on Mon Aug 25 17:03:57 2008 for libtonc by  doxygen 1.5.3

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **Secondary types**

## [Types and attributes]

# Data Structures

struct	<b>TILE</b> <i>4bpp tile type, for easy indexing and copying of 4-bit tiles</i> <a href="#">More...</a>
struct	<b>TILE8</b> <i>8bpp tile type, for easy indexing and 8-bit tiles</i> <a href="#">More...</a>
struct	<b>ObjAffineSource</b> <i>Simple scale-rotation source struct.</i> <a href="#">More...</a>
struct	<b>ObjAffineSource</b> <i>Simple scale-rotation source struct.</i> <a href="#">More...</a>
struct	<b>BgAffineSource</b> <i>Extended scale-rotate source struct.</i> <a href="#">More...</a>
struct	<b>ObjAffineDest</b> <i>Simple scale-rotation destination struct, BG version.</i> <a href="#">More...</a>
struct	<b>BgAffineDest</b> <i>Extended scale-rotate destination struct.</i> <a href="#">More...</a>

## Defines

```
#define TRUE 1  
#define FALSE 0
```

## Typedefs

typedef s32	<b>FIXED</b> <i>Fixed point type.</i>
typedef u16	<b>COLOR</b> <i>Type for colors.</i>
typedef u16	<b>SCR_ENTRY</b>
typedef u16	<b>SE</b> <i>Type for screen entries.</i>
typedef u8	<b>SCR_AFF_ENTRY</b>
typedef u8	<b>SAE</b> <i>Type for affine screen entries.</i>
typedef struct <b>TILE</b>	<b>TILE4</b>
typedef u8	<b>BOOL</b>
typedef void(*)	<b>fnptr</b> )(void) <i>void foo() function pointer</i>
typedef void(*)	<b>fn_v_i</b> )(int) <i>void foo(int x) function pointer</i>
typedef int(*)	<b>fn_i_i</b> )(int) <i>int foo(int x) function pointer</i>

## Enumerations

enum	<b>bool</b> { <b>false</b> , <b>true</b> }
	<i>Boolean type.</i>

---

Generated on Mon Aug 25 17:03:57 2008 for libtonc by  doxygen 1.5.3

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **Tertiary types**

## [Types and attributes]

# Data Structures

struct	<b>BG_POINT</b> <i>Regular bg points; range: :0010 - :001F.</i> <a href="#">More...</a>
struct	<b>BG_POINT</b> <i>Regular bg points; range: :0010 - :001F.</i> <a href="#">More...</a>
struct	<b>DMA_REC</b> <i>DMA struct; range: 0400:00B0 - 0400:00DF.</i> <a href="#">More...</a>
struct	<b>TMR_REC</b> <i>Timer struct, range: 0400:0100 - 0400:010F.</i> <a href="#">More...</a>
struct	<b>OBJ_ATTR</b> <i>Object attributes.</i> <a href="#">More...</a>
struct	<b>OBJ_ATTR</b> <i>Object attributes.</i> <a href="#">More...</a>
struct	<b>OBJ_AFFINE</b> <i>Object affine parameters.</i> <a href="#">More...</a>

## IO register types

typedef struct AFF_DST_EX	<b>BG_AFFINE</b> <i>Affine parameters for backgrounds; range : 0400:0020 - 0400:003F.</i>
---------------------------	--

## PAL types

typedef COLOR	<b>PALBANK</b> [16] <i>Palette bank type, for 16-color palette banks.</i>
---------------	--

## VRAM array types

These types allow VRAM access as arrays or matrices in their most natural types.

typedef SCR_ENTRY	<b>SCREENLINE</b> [32]
typedef SCR_ENTRY	<b>SCREENMAT</b> [32][32]
typedef SCR_ENTRY	<b>SCREENBLOCK</b> [1024]
typedef <b>COLOR</b>	<b>M3LINE</b> [240]
typedef u8	<b>M4LINE</b> [240]
typedef <b>COLOR</b>	<b>M5LINE</b> [160]
typedef <b>TILE</b>	<b>CHARBLOCK</b> [512]
typedef <b>TILE8</b>	<b>CHARBLOCK8</b> [256]

## Detailed Description

These types are used for memory mapping of VRAM, affine registers and other areas that would benefit from logical memory mapping.

---

Generated on Mon Aug 25 17:03:57 2008 for *libtonc* by  1.5.3

[Main Page](#) [Modules](#) [Data Structures](#) [Files](#) [Related Pages](#)

---

[Alphabetical List](#) [Data Structures](#) [Data Fields](#)

# libtcon Data Structures

Here are the data structures with brief descriptions:

<b>BG_POINT</b>	<i>Regular bg points; range: :0010 - :001F</i>
<b>BgAffineDest</b>	<i>Extended scale-rotate destination struct</i>
<b>BgAffineSource</b>	<i>Extended scale-rotate source struct</i>
<b>BLOCK</b>	<i>8-word type for fast struct-copies</i>
<b>BUP</b>	<i>BitUpPack ( for swi 10h)</i>
<b>DMA_REC</b>	<i>DMA struct; range: 0400:00B0 - 0400:00DF</i>
<b>IRQ_REC</b>	<i>Struct for prioritized irq table</i>
<b>IRQ_SENDER</b>	<i>IRQ Sender information</i>
<b>MultiBootParam</b>	<i>Multiboot struct</i>
<b>OBJ_AFFINE</b>	<i>Object affine parameters</i>
<b>OBJ_ATTR</b>	<i>Object attributes</i>
<b>ObjAffineDest</b>	<i>Simple scale-rotation destination struct, BG version</i>
<b>ObjAffineSource</b>	<i>Simple scale-rotation source struct</i>
<b>POINT32</b>	<i>2D Point struct</i>
<b>RECT32</b>	<i>Rectangle struct</i>
<b>REPEAT_REC</b>	<i>Repeated keys struct</i>
<b>TFont</b>	<i>Font description struct</i>
<b>TILE</b>	<i>4bpp tile type, for easy indexing and copying of 4-bit tiles</i>
<b>TILE8</b>	<i>8bpp tile type, for easy indexing and 8-bit tiles</i>
<b>TMR_REC</b>	<i>Timer struct, range: 0400:0100 - 0400:010F</i>
<b>TTC</b>	<i>TTE context struct</i>
<b>VECTOR</b>	<i>Vector struct</i>

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  **doxygen** 1.5.3

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)[Alphabetical List](#)[Data Structures](#)[Data Fields](#)

# **BG\_POINT Struct Reference**

## [[Tertiary types](#), [Tertiary types](#)]

Regular bg points; range: :0010 - :001F. [More...](#)

```
#include <tonc_types.h>
```

## Data Fields

s16	x
s16	y

---

## Detailed Description

Regular bg points; range: :0010 - :001F.

---

The documentation for this struct was generated from the following file:

- [tonc\\_types.h](#)
- 

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)[Alphabetical List](#)[Data Structures](#)[Data Fields](#)

# BgAffineDest Struct Reference

## [Secondary types]

Extended scale-rotate destination struct. [More...](#)

```
#include <tonc_types.h>
```

## Data Fields

s16	<b>pa</b>
s16	<b>pb</b>
s16	<b>pc</b>
s16	<b>pd</b>
s32	<b>dx</b>
s32	<b>dy</b>

---

## Detailed Description

Extended scale-rotate destination struct.

This contains the P-matrix and a fixed-point offset , the combination can be used to rotate around an arbitrary point. Mainly intended for BgAffineSet, but the struct can be used for object transforms too.

---

The documentation for this struct was generated from the following file:

- [tonc\\_types.h](#)
- 

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)[Alphabetical List](#)[Data Structures](#)[Data Fields](#)

# BgAffineSource Struct Reference

## [Secondary types]

Extended scale-rotate source struct. [More...](#)

```
#include <tonc_types.h>
```

## Data Fields

s32	<b>tex_x</b> <i>Texture-space anchor, x coordinate (.8f).</i>
s32	<b>tex_y</b> <i>Texture-space anchor, y coordinate (.8f).</i>
s16	<b>scr_x</b> <i>Screen-space anchor, x coordinate (.0f).</i>
s16	<b>scr_y</b> <i>Screen-space anchor, y coordinate (.0f).</i>
s16	<b>sx</b> <i>Horizontal zoom (8.8f).</i>
s16	<b>sy</b> <i>Vertical zoom (8.8f).</i>
u16	<b>alpha</b> <i>Counter-clockwise angle ( range [0, 0xFFFF] ).</i>

---

## Detailed Description

Extended scale-rotate source struct.

This is used to scale/rotate around an arbitrary point. See tonc's main text for all the details.

---

The documentation for this struct was generated from the following file:

- [tonc\\_types.h](#)
- 

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by  1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)[Alphabetical List](#)[Data Structures](#)[Data Fields](#)

# **BLOCK Struct Reference**

## **[Primary types]**

8-word type for fast struct-copies [More...](#)

```
#include <tonc_types.h>
```

## Data Fields

u32 **data** [8]

---

## Detailed Description

8-word type for fast struct-copies

---

The documentation for this struct was generated from the following file:

- [tonc\\_types.h](#)
- 

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)[Alphabetical List](#)[Data Structures](#)[Data Fields](#)

# BUP Struct Reference

## [[BIOS informalities](#)]

BitUpPack ( for swi 10h). [More...](#)

```
#include <tonc_bios.h>
```

## Data Fields

u16	<b>src_len</b>
	<i>source length (bytes)</i>
u8	<b>src_bpp</b>
	<i>source bitdepth (1,2,4,8)</i>
u8	<b>dst_bpp</b>
	<i>destination bitdepth (1,2,4,8,16,32)</i>
u32	<b>dst_ofs</b>
	<i>{0-30}: added offset {31}: zero-data offset flag</i>

---

## Detailed Description

BitUpPack ( for swi 10h).

---

The documentation for this struct was generated from the following file:

- [tonc\\_bios.h](#)
- 

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by  doxygen 1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)[Alphabetical List](#)[Data Structures](#)[Data Fields](#)

# **DMA\_REC Struct Reference**

## **[Tertiary types]**

DMA struct; range: 0400:00B0 - 0400:00DF. [More...](#)

```
#include <tonc_types.h>
```

## Data Fields

const void *	<b>src</b>
void *	<b>dst</b>
u32	<b>cnt</b>

---

## Detailed Description

DMA struct; range: 0400:00B0 - 0400:00DF.

---

The documentation for this struct was generated from the following file:

- [tonc\\_types.h](#)
- 

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)[Alphabetical List](#)[Data Structures](#)[Data Fields](#)

# **IRQ\_REC Struct Reference**

## **[Interrupt]**

Struct for prioritized irq table. [More...](#)

```
#include <tonc_irq.h>
```

## Data Fields

u32	<b>flag</b>
<i>Flag for interrupt in REG_IF, etc.</i>	
fnptr	<b>isr</b>
<i>Pointer to interrupt routine.</i>	

---

## Detailed Description

Struct for prioritized irq table.

---

The documentation for this struct was generated from the following file:

- [tonc\\_irq.h](#)
- 

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)[Alphabetical List](#)[Data Structures](#)[Data Fields](#)

# **IRQ\_SENDER Struct Reference**

IRQ Sender information. [More...](#)

## Data Fields

u16	<b>reg_ofs</b>
	<i>sender reg - REG_BASE</i>
u16	<b>flag</b>
	<i>irq-bit in sender reg</i>

---

## Detailed Description

IRQ Sender information.

---

The documentation for this struct was generated from the following file:

- [tonc\\_irq.c](#)
- 

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)[Alphabetical List](#)[Data Structures](#)[Data Fields](#)

# **MultiBootParam Struct Reference**

## [**BIOS informalities**]

Multiboot struct. [More...](#)

```
#include <tonc_bios.h>
```

## Data Fields

u32	<b>reserved1</b> [5]
u8	<b>handshake_data</b>
u8	<b>padding</b>
u16	<b>handshake_timeout</b>
u8	<b>probe_count</b>
u8	<b>client_data</b> [3]
u8	<b>palette_data</b>
u8	<b>response_bit</b>
u8	<b>client_bit</b>
u8	<b>reserved2</b>
u8 *	<b>boot_srcp</b>
u8 *	<b>boot_endp</b>
u8 *	<b>masterp</b>
u8 *	<b>reserved3</b> [3]
u32	<b>system_work2</b> [4]
u8	<b>sendflag</b>
u8	<b>probe_target_bit</b>
u8	<b>check_wait</b>
u8	<b>server_type</b>

---

## Detailed Description

Multiboot struct.

---

The documentation for this struct was generated from the following file:

- [tonc\\_bios.h](#)
- 

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)[Alphabetical List](#)[Data Structures](#)[Data Fields](#)

# **OBJ\_AFFINE** Struct Reference

## [**Tertiary types**]

Object affine parameters. [More...](#)

```
#include <tonc_types.h>
```

## Data Fields

u16	<b>fill0</b> [3]
s16	<b>pa</b>
u16	<b>fill1</b> [3]
s16	<b>pb</b>
u16	<b>fill2</b> [3]
s16	<b>pc</b>
u16	<b>fill3</b> [3]
s16	<b>pd</b>

---

## Detailed Description

Object affine parameters.

### Note:

most fields are padding for the interlace with **OBJ\_ATTR**.

---

The documentation for this struct was generated from the following file:

- [tonc\\_types.h](#)
- 

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by  doxygen 1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)[Alphabetical List](#)[Data Structures](#)[Data Fields](#)

# **OBJ\_ATTR Struct Reference**

## [**Tertiary types, Tertiary types**]

Object attributes. [More...](#)

```
#include <tonc_types.h>
```

## Data Fields

u16	<b>attr0</b>
u16	<b>attr1</b>
u16	<b>attr2</b>
s16	<b>fill</b>

---

## Detailed Description

Object attributes.

### Note:

attribute 3 is padding for the interlace with **OBJ\_AFFINE**. If not using affine objects, it can be used as a free field

---

The documentation for this struct was generated from the following file:

- [tonc\\_types.h](#)
- 

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  **doxygen** 1.5.3

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)[Alphabetical List](#)[Data Structures](#)[Data Fields](#)

# **ObjAffineDest Struct Reference**

## **[Secondary types]**

Simple scale-rotation destination struct, BG version. [More...](#)

```
#include <tonc_types.h>
```

## Data Fields

s16	<b>pa</b>
s16	<b>pb</b>
s16	<b>pc</b>
s16	<b>pd</b>

---

## Detailed Description

Simple scale-rotation destination struct, BG version.

This is a P-matrix with continuous elements, like the BG matrix.  
It can be used with ObjAffineSet.

---

The documentation for this struct was generated from the  
following file:

- [tonc\\_types.h](#)
- 

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)[Alphabetical List](#)[Data Structures](#)[Data Fields](#)

# **ObjAffineSource Struct Reference**

## **[Secondary types, Secondary types]**

Simple scale-rotation source struct. [More...](#)

```
#include <tonc_types.h>
```

## Data Fields

s16	<b>sx</b>
<i>Horizontal zoom (8.8f).</i>	
s16	<b>sy</b>
<i>Vertical zoom (8.8f).</i>	
u16	<b>alpha</b>
<i>Counter-clockwise angle ( range [0, 0xFFFF] ).</i>	

---

## Detailed Description

Simple scale-rotation source struct.

This can be used with ObjAffineSet, and several of tonc's affine functions

---

The documentation for this struct was generated from the following file:

- [tonc\\_types.h](#)
- 

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by  doxygen 1.5.3*

[Main Page](#) [Modules](#) [Data Structures](#) [Files](#) [Related Pages](#)

---

[Alphabetical List](#) [Data Structures](#) [Data Fields](#)

# **POINT32 Struct Reference**

## **[Point functions]**

2D Point struct [More...](#)

```
#include <tonc_math.h>
```

## Data Fields

int	x
int	y

---

## Detailed Description

### 2D Point struct

---

The documentation for this struct was generated from the following file:

- [tonc\\_math.h](#)
- 

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)[Alphabetical List](#)[Data Structures](#)[Data Fields](#)

# **RECT32 Struct Reference**

## **[Rect functions]**

Rectangle struct. [More...](#)

```
#include <tonc_math.h>
```

## Data Fields

int	<b>left</b>
int	<b>top</b>
int	<b>right</b>
int	<b>bottom</b>

---

## Detailed Description

Rectangle struct.

---

The documentation for this struct was generated from the following file:

- [tonc\\_math.h](#)
- 

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)[Alphabetical List](#)[Data Structures](#)[Data Fields](#)

# **REPEAT\_REC Struct Reference**

Repeated keys struct. [More...](#)

## Data Fields

u16	<b>keys</b>
	<i>Repeated keys.</i>
u16	<b>mask</b>
	<i>Only check repeats for these keys.</i>
u8	<b>count</b>
	<i>Repeat counter.</i>
u8	<b>delay</b>
	<i>Limit for first repeat.</i>
u8	<b>repeat</b>
	<i>Limit for successive repeats.</i>

---

## Detailed Description

Repeated keys struct.

---

The documentation for this struct was generated from the following file:

- [tonc\\_input.c](#)
- 

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)[Alphabetical List](#)[Data Structures](#)[Data Fields](#)

# **TFont Struct Reference**

## [**Tonc Text Engine**]

Font description struct. [More...](#)

```
#include <tonc_tte.h>
```

## Data Fields

const void *	<b>data</b> <i>Character data.</i>
const u8 *	<b>widths</b> <i>Width table for variable width font.</i>
const u8 *	<b>heights</b> <i>Height table for variable height font.</i>
u16	<b>charOffset</b> <i>Character offset.</i>
u16	<b>charCount</b> <i>Number of characters in font.</i>
u8	<b>charW</b> <i>Character width (fwf).</i>
u8	<b>charH</b> <i>Character height.</i>
u8	<b>cellW</b> <i>Glyph cell width.</i>
u8	<b>cellH</b> <i>Glyph cell height.</i>
u16	<b>cellSize</b> <i>Cell-size (bytes).</i>
u8	<b>bpp</b> <i>Font bitdepth;.</i>
u8	<b>extra</b> <i>Padding. Free to use.</i>

---

## Detailed Description

Font description struct.

The [TFont](#) contains a description of the font, including pointers to the glyph data and width data (for VWF fonts), an ascii-offset for when the first glyph isn't for ascii-null (which is likely. Usually it starts at ' ' (32)).

The font-bitmap is a stack of cells, each containing one glyph each. The cells and characters need not be the same size, but the character glyph must fit within the cell.

The formatting of the glyphs themselves should fit the rendering procedure. The default renderers use 1bpp 8x8 tiled graphics, where for multi-tiled cells the tiles are in a **vertical** 'strip' format. In an 16x16 cell, the 4 tiles would be arranged as:

```
">


|   |   |
|---|---|
| 0 | 2 |
| 1 | 3 |


```

---

The documentation for this struct was generated from the following file:

- [tonc\\_tte.h](#)
-

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)[Alphabetical List](#)[Data Structures](#)[Data Fields](#)

## **TILE Struct Reference**

### **[Secondary types]**

4bpp tile type, for easy indexing and copying of 4-bit tiles

[More...](#)

```
#include <tonc_types.h>
```

## Data Fields

u32 **data [8]**

---

## Detailed Description

4bpp tile type, for easy indexing and copying of 4-bit tiles

---

The documentation for this struct was generated from the following file:

- [tonc\\_types.h](#)
- 

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)[Alphabetical List](#)[Data Structures](#)[Data Fields](#)

# **TILE8 Struct Reference**

## **[Secondary types]**

8bpp tile type, for easy indexing and 8-bit tiles [More...](#)

```
#include <tonc_types.h>
```

## Data Fields

u32 **data** [16]

---

## Detailed Description

8bpp tile type, for easy indexing and 8-bit tiles

---

The documentation for this struct was generated from the following file:

- [tonc\\_types.h](#)
- 

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)[Alphabetical List](#)[Data Structures](#)[Data Fields](#)

# **TMR\_REC Struct Reference**

## [**Tertiary types**]

Timer struct, range: 0400:0100 - 0400:010F. [More...](#)

```
#include <tonc_types.h>
```

## Data Fields

union {	
u16 start	
u16 count	
}	PACKED
u16 cnt	

---

## Detailed Description

Timer struct, range: 0400:0100 - 0400:010F.

### Note:

The attribute is required, because union's counted as u32 otherwise.

---

The documentation for this struct was generated from the following file:

- [tonc\\_types.h](#)
- 

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)[Alphabetical List](#)[Data Structures](#)[Data Fields](#)

# TTC Struct Reference

## [[Tonc Text Engine](#)]

TTE context struct. [More...](#)

```
#include <tonc_tte.h>
```

## Data Fields

TSurface	<b>dst</b> <i>Destination surface.</i>
s16	<b>cursorX</b> <i>Cursor X-coord.</i>
s16	<b>cursorY</b> <i>Cursor Y-coord.</i>
<b>TFont *</b>	<b>font</b> <i>Current font.</i>
u8 *	<b>charLut</b> <i>Character mapping lut. (if any).</i>
u16	<b>cattr</b> [4] <i>ink, shadow, paper and special color attributes.</i>
u16	<b>flags0</b>
u16	<b>ctrl</b> <i>BG control flags. (PONDER: remove?).</i>
u16	<b>marginLeft</b>
u16	<b>marginTop</b>
u16	<b>marginRight</b>
u16	<b>marginBottom</b>
s16	<b>savedX</b>
s16	<b>savedY</b>
<b>fnDrawg</b>	<b>drawgProc</b> <i>Glyph render procedure.</i>
<b>fnErase</b>	<b>eraseProc</b> <i>Text eraser procedure.</i>
const <b>TFont **</b>	<b>fontTable</b> <i>Pointer to font table for f}.</i>
const char **	<b>stringTable</b> <i>Pointer to string table for s}.</i>

---

## Detailed Description

TTE context struct.

---

The documentation for this struct was generated from the following file:

- [tonc\\_tte.h](#)
- 

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)[Alphabetical List](#)[Data Structures](#)[Data Fields](#)

# **VECTOR Struct Reference**

## **[Vector functions]**

Vector struct. [More...](#)

```
#include <tonc_math.h>
```

## Data Fields

<b>FIXED</b>	x
<b>FIXED</b>	y
<b>FIXED</b>	z

---

## Detailed Description

Vector struct.

---

The documentation for this struct was generated from the following file:

- [tonc\\_math.h](#)
- 

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  *1.5.3*

Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- a -

- alpha : [ObjAffineSource](#) , [BgAffineSource](#)

- b -

- bpp : [TFont](#)

- c -

- cattr : [TTC](#)
- cellH : [TFont](#)
- cellSize : [TFont](#)
- cellW : [TFont](#)
- charCount : [TFont](#)
- charH : [TFont](#)
- charLut : [TTC](#)
- charOffset : [TFont](#)
- charW : [TFont](#)
- count : [REPEAT\\_REC](#)
- ctrl : [TTC](#)
- cursorX : [TTC](#)
- cursorY : [TTC](#)

- d -

- data : **TFont**
- delay : **REPEAT\_REC**
- drawgProc : **TTC**
- dst : **TTC**
- dst\_bpp : **BUP**
- dst\_ofs : **BUP**

- e -

- eraseProc : **TTC**
- extra : **TFont**

- f -

- flag : **IRQ\_REC** , **IRQ\_SENDER**
- font : **TTC**
- fontTable : **TTC**

- h -

- heights : **TFont**

- i -

- isr : **IRQ\_REC**

- k -

- keys : **REPEAT\_REC**

- m -

- mask : **REPEAT\_REC**

- r -

- reg\_ofs : **IRQ\_SENDER**
- repeat : **REPEAT\_REC**

- s -

- scr\_x : **BgAffineSource**
- scr\_y : **BgAffineSource**
- src\_bpp : **BUP**
- src\_len : **BUP**
- stringTable : **TTC**
- sx : **BgAffineSource , ObjAffineSource**
- sy : **BgAffineSource , ObjAffineSource**

- t -

- tex\_x : **BgAffineSource**
- tex\_y : **BgAffineSource**

- w -

- widths : **TFont**

[Main Page](#) [Modules](#) [Data Structures](#) [Files](#) [Related Pages](#)

---

[File List](#) [Globals](#)

# libtonc File List

Here is a list of all documented files with brief descriptions:

<a href="#">ase_drawg.c</a>	
<a href="#">bmp16_drawg.c</a>	
<a href="#">bmp16_drawg_b1cs.c</a>	
<a href="#">bmp8_drawg.c</a>	
<a href="#">bmp8_drawg_b1cs.c</a>	
<a href="#">chr4c_drawg_b1cts.c</a>	
<a href="#">chr4c_drawg_b4cts.c</a>	
<a href="#">chr4r_drawg_b1cts.c</a>	
<a href="#">obj_drawg.c</a>	
<a href="#">se_drawg.c</a>	
<a href="#">tonc.h</a>	
<a href="#">tonc_bg.c</a>	
<a href="#">tonc_bg_affine.c</a>	
<a href="#">tonc_bios.h</a>	
<a href="#">tonc bmp16.c</a>	
<a href="#">tonc bmp8.c</a>	
<a href="#">tonc_color.c</a>	
<a href="#">tonc_core.c</a>	
<a href="#">tonc_core.h</a>	
<a href="#">tonc_input.c</a>	
<a href="#">tonc_input.h</a>	
<a href="#">tonc_irq.c</a>	
<a href="#">tonc_irq.h</a>	
<a href="#">tonc_legacy.h</a>	

<a href="#">tonc_libgba.h</a>	
<a href="#">tonc_math.c</a>	
<a href="#">tonc_math.h</a>	
<a href="#">tonc_memdef.h</a>	
<a href="#">tonc_memmap.h</a>	
<a href="#">tonc_nocash.h</a>	
<a href="#">tonc_oam.c</a>	
<a href="#">tonc_oam.h</a>	
<a href="#">tonc_obj_affine.c</a>	
<a href="#">tonc_sbmp16.c</a>	
<a href="#">tonc_sbmp8.c</a>	
<a href="#">tonc_schr4c.c</a>	
<a href="#">tonc_schr4r.c</a>	
<a href="#">tonc_surface.c</a>	
<a href="#">tonc_surface.h</a>	
<a href="#">tonc_text.h</a>	
<a href="#">tonc_tte.h</a>	
<a href="#">tonc_types.h</a>	
<a href="#">tonc_video.c</a>	
<a href="#">tonc_video.h</a>	
<a href="#">tte_init_ase.c</a>	
<a href="#">tte_init_bmp.c</a>	
<a href="#">tte_init_chr4c.c</a>	
<a href="#">tte_init_chr4r.c</a>	
<a href="#">tte_init_obj.c</a>	
<a href="#">tte_init_se.c</a>	
<a href="#">tte_iohook.c</a>	
<a href="#">tte_main.c</a>	

---

*Generated on Mon Aug 25 17:03:57 2008 for libtonc by*  **1.5.3**

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# ase\_drawg.c File Reference

```
#include "tonc_types.h" #include "tonc_surface.h"  
#include "tonc_tte.h"
```

## Functions

void	<b>ase_erase</b> (int left, int top, int right, int bottom)
------	---

*Erase part of the affine tilemap canvas.*

void	<b>ase_drawg_w8h8</b> (uint gid)
------	----------------------------------

*Character-plot for affine BGs using an 8x8 font.*

void	<b>ase_drawg_w8h16</b> (uint gid)
------	-----------------------------------

*Character-plot for affine BGs using an 8x16 font.*

void	<b>ase_drawg</b> (uint gid)
------	-----------------------------

*Character-plot for affine Bgs, any size.*

void	<b>ase_drawg_s</b> (uint gid)
------	-------------------------------

*Character-plot for affine BGs, any sized, vertically oriented font.*

---

# Detailed Description

**Author:**

J Vijn

**Date:**

20070701 - 20080516

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **bmp16\_drawg.c File Reference**

```
#include "tonc_memdef.h" #include "tonc_tte.h"
```

## Functions

void	<b>bmp16_drawg</b> (uint gid)
<i>Linear 16bpp bitmap glyph renderer, opaque.</i>	
void	<b>bmp16_drawg_t</b> (uint gid)
<i>Linear 16bpp bitmap glyph renderer, transparent.</i>	

# Detailed Description

**Author:**

J Vijn

**Date:**

20080311 - 20080311

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **bmp16\_drawg\_b1cs.c File Reference**

```
#include "tonc_memdef.h" #include "tonc_tte.h"
```

## Functions

void	<b>bmp16_drawg_b1cts</b> (uint gid)
<i>Linear bitmap, 16bpp transparent character plotter.</i>	
void	<b>bmp16_drawg_b1cos</b> (uint gid)
<i>Linear bitmap, 16bpp opaque character plotter.</i>	

## Detailed Description

**Author:**

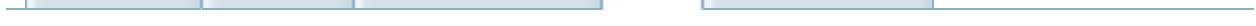
J Vijn

**Date:**

20070605 - 20070704

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **bmp8\_drawg.c File Reference**

```
#include "tonc_memdef.h" #include "tonc_tte.h"
```

## Functions

void	<b>bmp8_drawg</b> (uint gid)
<i>Linear 8 bpp bitmap glyph renderer, opaque.</i>	
void	<b>bmp8_drawg_t</b> (uint gid)
<i>Linear 8 bpp bitmap glyph renderer, transparent.</i>	

---

# Detailed Description

**Author:**

J Vijn

**Date:**

20080311 - 20080311

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **bmp8\_drawg\_b1cs.c File Reference**

```
#include "tonc_memdef.h" #include "tonc_surface.h"  
#include "tonc_tte.h"
```

## Functions

```
void bmp8_drawg_b1cts (uint gid)  
void bmp8_drawg_b1cos (uint gid)
```

---

## Detailed Description

**Author:**

J Vijn

**Date:**

20070613 - 20070613

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **chr4c\_drawg\_b1cts.c File Reference**

```
#include "tonc_memdef.h" #include "tonc_tte.h"
```

## Functions

void	<b>chr4c_drawg_b1cts</b> (uint gid)
<i>Render 1bpp fonts to 4bpp tiles.</i>	

---

# Detailed Description

**Author:**

J Vijn

**Date:**

20070621 - 20080427

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **chr4c\_drawg\_b4cts.c File Reference**

```
#include "tonc_memdef.h" #include "tonc_tte.h"
```

## Functions

```
void chr4c_drawg_b4cts (uint gid)
```

---

# Detailed Description

**Author:**

J Vijn

**Date:**

20080427 - 20080427

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **chr4r\_drawg\_b1cts.c File Reference**

```
#include "tonc_memdef.h" #include "tonc_tte.h"
```

## Functions

void	<b>chr4r_drawg_b1cts</b> (uint gid)
<i>Render 1bpp fonts to 4bpp tiles.</i>	

---

# Detailed Description

**Author:**

J Vijn

**Date:**

20070621 - 20070725

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **obj\_drawg.c File Reference**

```
#include "tonc_types.h" #include "tonc_memdef.h"  
#include "tonc_core.h"  
#include "tonc_oam.h"  
#include "tonc_tte.h"
```

## Functions

void	<b>obj_erase</b> (int left, int top, int right, int bottom)
------	---

*Unwind the object text-buffer.*

void	<b>obj_drawg</b> (uint gid)
------	-----------------------------

*Character-plot for objects.*

---

# Detailed Description

**Author:**

J Vijn

**Date:**

20070715 - 20070822

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **se\_drawg.c File Reference**

```
#include "tonc_types.h" #include "tonc_tte.h"  
#include "tonc_surface.h"
```

## Defines

```
#define PXSIZE sizeof(pixel_t)
#define PXPTR(psrf, x, y) ((pixel_t*)((psrf)->data + (y)*(psrf)->pitch +
(x)*sizeof(pixel_t) ))
```

## **Typedefs**

```
typedef u16 pixel_t
```

# Functions

void	<b>se_erase</b> (int left, int top, int right, int bottom)	
		<i>Erase part of the regular tilemap canvas.</i>
void	<b>se_drawg_w8h8</b> (uint gid)	
		<i>Character-plot for reg BGs using an 8x8 font.</i>
void	<b>se_drawg_w8h16</b> (uint gid)	
		<i>Character-plot for reg BGs using an 8x16 font.</i>
void	<b>se_drawg</b> (uint gid)	
		<i>Character-plot for reg BGs, any sized font.</i>
void	<b>se_drawg_s</b> (uint gid)	
		<i>Character-plot for reg BGs, any sized, vertically tiled font.</i>

## Detailed Description

**Author:**

J Vijn

**Date:**

20070628 - 20070628

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# tonc.h File Reference

```
#include "tonc_types.h" #include "tonc_memmap.h"
#include "tonc_memdef.h"
#include "tonc_bios.h"
#include "tonc_core.h"
#include "tonc_input.h"
#include "tonc_irq.h"
#include "tonc_math.h"
#include "tonc_oam.h"
#include "tonc_tte.h"
#include "tonc_video.h"
#include "tonc_surface.h"
#include "tonc_nocash.h"
#include "tonc_text.h"
```

---

# Detailed Description

**Author:**

J Vijn

**Date:**

20060508 - 20080825

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tonc\_bg.c File Reference**

```
#include "tonc_memmap.h" #include "tonc_video.h"
```

## Functions

void	<b>se_window</b> (SCR_ENTRY *sbb, int left, int top, int right, int bottom, SCR_ENTRY se0)
	<i>Create a framed rectangle.</i>

---

# Detailed Description

**Author:**

J Vijn

**Date:**

20061112 - 20061117

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tonc\_bg\_affine.c File Reference**

```
#include "tonc_memmap.h" #include "tonc_core.h"
#include "tonc_video.h"
#include "tonc_math.h"
#include "tonc_bios.h"
```

## Functions

void	<b>bg_aff_rotate</b> ( <b>BG_AFFINE</b> *bgaff, u16 alpha)	<i>Set bg matrix to counter-clockwise rotation.</i>
void	<b>bg_aff_rotscale</b> ( <b>BG_AFFINE</b> *bgaff, int sx, int sy, u16 alpha)	<i>Set bg matrix to 2d scaling, then counter-clockwise rotation.</i>
void	<b>bg_aff_premul</b> ( <b>BG_AFFINE</b> *dst, const <b>BG_AFFINE</b> *src)	<i>Pre-multiply dst by src: <math>D = S*D</math>.</i>
void	<b>bg_aff_postmul</b> ( <b>BG_AFFINE</b> *dst, const <b>BG_AFFINE</b> *src)	<i>Post-multiply dst by src: <math>D= D*S</math>.</i>
void	<b>bg_aff_rotscale2</b> ( <b>BG_AFFINE</b> *bgaff, const AFF_SRC *as)	<i>Set bg matrix to 2d scaling, then counter-clockwise rotation.</i>
void	<b>bg_rotscale_ex</b> ( <b>BG_AFFINE</b> *bgaff, const AFF_SRC_EX *asx)	<i>Set bg affine matrix to a rot/scale around an arbitrary point.</i>

# Detailed Description

**Author:**

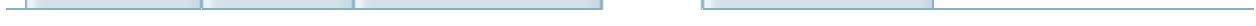
J Vijn

**Date:**

20060916 - 20060916

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tonc\_bios.h File Reference**

```
#include "tonc_types.h"
```

# Data Structures

struct	<b>BUP</b> <i>BitUpPack ( for swi 10h).</i> <a href="#">More...</a>
struct	<b>MultiBootParam</b> <i>Multiboot struct.</i> <a href="#">More...</a>

## Defines

#define	<b>swi_call</b> (x) asm volatile("swi\t#x"<<16" :: "r0", "r1", "r2", "r3") <i>BIOS calls from C.</i>
#define	<b>DivMod</b> Mod

### SoftReset flags

#define	<b>ROM_RESTART</b> 0x00 <i>Restart from ROM entry point.</i>
#define	<b>RAM_RESTART</b> 0x01 <i>Restart from RAM entry point.</i>

### RegisterRamReset flags

#define	<b>RESET_EWRAM</b> 0x0001 <i>Clear 256K on-board WRAM.</i>
#define	<b>RESET_IWRAM</b> 0x0002 <i>Clear 32K in-chip WRAM.</i>
#define	<b>RESET_PALETTE</b> 0x0004 <i>Clear Palette.</i>
#define	<b>RESET_VRAM</b> 0x0008 <i>Clear VRAM.</i>
#define	<b>RESET_OAM</b> 0x0010 <i>Clear OAM. does NOT disable OBJS!</i>
#define	<b>RESET_REG_SIO</b> 0x0020 <i>Switches to general purpose mode.</i>
#define	<b>RESET_REG_SOUND</b> 0x0040 <i>Reset Sound registers.</i>
#define	<b>RESET_REG</b> 0x0080 <i>All other registers.</i>
#define	<b>RESET_MEM_MASK</b> 0x001F <i>Clear 256K on-board WRAM.</i>
#define	<b>RESET_REG_MASK</b> 0x00E0 <i>Clear 256K on-board WRAM.</i>
#define	<b>RESET_GFX</b> 0x001C

*Clear all gfx-related memory.*

## Cpu(Fast)Set flags

#define	<b>CS_CPY</b> 0 <i>Copy mode.</i>
#define	<b>CS_FILL</b> (1<<24) <i>Fill mode.</i>
#define	<b>CS_CPY16</b> 0 <i>Copy in halfwords.</i>
#define	<b>CS_CPY32</b> (1<<26) <i>Copy words.</i>
#define	<b>CS_FILL32</b> (5<<24) <i>Fill words.</i>
#define	<b>CFS_CPY</b> CS_CPY <i>Copy words.</i>
#define	<b>CFS_FILL</b> CS_FILL <i>Fill words.</i>

## ObjAffineSet P-element offsets

#define	<b>BG_AFF_OFS</b> 2 <i>BgAffineDest offsets.</i>
#define	<b>OBJ_AFF_OFS</b> 8 <i>ObjAffineDest offsets.</i>

## Decompression routines

#define	<b>BUP_ALL_OFS</b> (1<<31)
#define	<b>LZ_TYPE</b> 0x00000010
#define	<b>LZ_SIZE_MASK</b> 0xFFFFFFF00
#define	<b>LZ_SIZE_SHIFT</b> 8
#define	<b>HUF_BPP_MASK</b> 0x0000000F
#define	<b>HUF_TYPE</b> 0x00000020
#define	<b>HUF_SIZE_MASK</b> 0xFFFFFFF00
#define	<b>HUF_SIZE_SHIFT</b> 8
#define	<b>RL_TYPE</b> 0x00000030
#define	<b>RL_SIZE_MASK</b> 0xFFFFFFF00

```
#define RL_SIZE_SHIFT 8
#define DIF_8 0x00000001
#define DIF_16 0x00000002
#define DIF_TYPE 0x00000080
#define DIF_SIZE_MASK 0xFFFFFFF00
#define DIF_SIZE_SHIFT 8
```

## Multiboot modes

```
#define MBOOT_NORMAL 0x00
#define MBOOT_MULTI 0x01
#define MBOOT_FAST 0x02
```

# Functions

u32	<b>BiosCheckSum</b> (void)
void	<b>VBlankIntrDelay</b> (u32 count) <i>Wait for count frames.</i>
int	<b>DivSafe</b> (int num, int den) <i>Div/0-safe division.</i>
int	<b>Mod</b> (int num, int den) <i>Modulo: num % den.</i>
u32	<b>DivAbs</b> (int num, int den) <i>Absolute value of num / den.</i>
int	<b>DivArmMod</b> (int den, int num) <i>Modulo: num % den.</i>
u32	<b>DivArmAbs</b> (int den, int num) <i>Absolute value of num / den.</i>
void	<b>CpuFastFill</b> (u32 wd, void *dst, u32 mode) <i>A fast word fill.</i>

## Reset functions

void	<b>SoftReset</b> (void)
void	<b>RegisterRamReset</b> (u32 flags)

## Halt functions

void	<b>Halt</b> (void)
void	<b>Stop</b> (void)
void	<b>IntrWait</b> (u32 flagClear, u32 irq)
void	<b>VBlankIntrWait</b> (void) <i>Wait for the next VBlank (swi 05h).</i>

## Math functions

s32	<b>Div</b> (s32 num, s32 den) <i>Basic integer division (swi 06h).</i>
s32	<b>DivArm</b> (s32 den, s32 num) <i>Basic integer division, but with switched arguments (swi 07h).</i>

u32	<b>Sqrt</b> (u32 num) <i>Integer Square root (swi 08h).</i>
s16	<b>ArcTan</b> (s16 dydx) <i>Arctangent of dydx (swi 08h).</i>
s16	<b>ArcTan2</b> (s16 x, s16 y) <i>Arctangent of a coordinate pair (swi 09h).</i>

## Memory copiers/fillers

void	<b>CpuSet</b> (const void *src, void *dst, u32 mode) <i>Transfer via CPU in (half)word chunks.</i>
void	<b>CpuFastSet</b> (const void *src, void *dst, u32 mode) <i>A fast transfer via CPU in 32 byte chunks.</i>

## Rot/scale functions

void	<b>ObjAffineSet</b> (const <b>ObjAffineSource</b> *src, void *dst, s32 num, s32 offset) <i>Sets up a simple scale-then-rotate affine transformation (swi 0Eh).</i>
void	<b>BgAffineSet</b> (const <b>BgAffineSource</b> *src, <b>BgAffineDest</b> *dst, s32 num) <i>Sets up a simple scale-then-rotate affine transformation (swi 0Eh).</i>

## Decompression (see GBATek for format details)

void	<b>BitUnPack</b> (const void *src, void *dst, const <b>BUP</b> *bup)
void	<b>LZ77UnCompWram</b> (const void *src, void *dst)
void	<b>LZ77UnCompVram</b> (const void *src, void *dst)
void	<b>HuffUnComp</b> (const void *src, void *dst)
void	<b>RLUnCompWram</b> (const void *src, void *dst)
void	<b>RLUnCompVram</b> (const void *src, void *dst)
void	<b>Diff8bitUnFilterWram</b> (const void *src, void *dst)
void	<b>Diff8bitUnFilterVram</b> (const void *src, void *dst)
void	<b>Diff16bitUnFilter</b> (const void *src, void *dst)

## Sound Functions

void	<b>SoundBias</b> (u32 bias)
void	<b>SoundDriverInit</b> (void *src)
void	<b>SoundDriverMode</b> (u32 mode)
void	<b>SoundDriverMain</b> (void)

void	<b>SoundDriverVSync</b> (void)
void	<b>SoundChannelClear</b> (void)
u32	<b>MidiKey2Freq</b> (void *wa, u8 mk, u8 fp)
void	<b>SoundDriverVSyncOff</b> (void)
void	<b>SoundDriverVSyncOn</b> (void)

## Multiboot handshake

int	<b>MultiBoot</b> ( <a href="#">MultiBootParam</a> *mb, u32 mode)
-----	--

---

## Detailed Description

**Author:**

J Vijn

**Date:**

20060508 - 20070208

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tonc\_bmp16.c File Reference**

```
#include "tonc_memmap.h" #include "tonc_core.h"  
#include "tonc_video.h"
```

## Functions

void	<b>bmp16_plot</b> (int x, int y, u32 clr, void *dstBase, uint dstP) <i>Plot a single pixel on a 16-bit buffer.</i>
void	<b>bmp16_hline</b> (int x1, int y, int x2, u32 clr, void *dstBase, uint dstP) <i>Draw a horizontal line on an 16bit buffer.</i>
void	<b>bmp16_vline</b> (int x, int y1, int y2, u32 clr, void *dstBase, uint dstP) <i>Draw a vertical line on an 16bit buffer.</i>
void	<b>bmp16_line</b> (int x1, int y1, int x2, int y2, u32 clr, void *dstBase, uint dstP) <i>Draw a line on an 16bit buffer.</i>
void	<b>bmp16_rect</b> (int left, int top, int right, int bottom, u32 clr, void *dstBase, uint dstP) <i>Draw a rectangle in 16bit mode; internal routine.</i>
void	<b>bmp16_frame</b> (int left, int top, int right, int bottom, u32 clr, void *dstBase, uint dstP) <i>Draw a rectangle in 16bit mode; internal routine.</i>

# Detailed Description

**Author:**

J Vijn

**Date:**

20060604 - 20070703

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tonc\_bmp8.c File Reference**

```
#include "tonc_memmap.h" #include "tonc_core.h"  
#include "tonc_video.h"
```

## Functions

void	<b>bmp8_plot</b> (int x, int y, u32 clr, void *dstBase, uint dstP) <i>Plot a single pixel on a 8-bit buffer.</i>
void	<b>bmp8_hline</b> (int x1, int y, int x2, u32 clr, void *dstBase, uint dstP) <i>Draw a horizontal line on an 8bit buffer.</i>
void	<b>bmp8_vline</b> (int x, int y1, int y2, u32 clr, void *dstBase, uint dstP) <i>Draw a vertical line on an 8bit buffer.</i>
void	<b>bmp8_line</b> (int x1, int y1, int x2, int y2, u32 clr, void *dstBase, uint dstP) <i>Draw a line on an 8bit buffer.</i>
void	<b>bmp8_rect</b> (int left, int top, int right, int bottom, u32 clr, void *dstBase, uint dstP) <i>Draw a rectangle in 8bit mode; internal routine.</i>
void	<b>bmp8_frame</b> (int left, int top, int right, int bottom, u32 clr, void *dstBase, uint dstP) <i>Draw a rectangle in 8bit mode; internal routine.</i>

# Detailed Description

**Author:**

J Vijn

**Date:**

20060604 - 20080516

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tonc\_color.c File Reference**

```
#include "tonc_memmap.h" #include "tonc_video.h"  
#include "tonc_bios.h"  
#include "tonc_math.h"
```

## Functions

void	<b>clr_rotate</b> ( <b>COLOR</b> *clrs, uint nclrs, int ror) <i>Rotate nclrs colors at clrs to the right by ror.</i>
void	<b>clr_blend</b> (const <b>COLOR</b> *srca, const <b>COLOR</b> *srcb, <b>COLOR</b> *dst, uint nclrs, u32 alpha) <i>Blends color arrays srca and srcb into dst.</i>
void	<b>clr_fade</b> (const <b>COLOR</b> *src, <b>COLOR</b> clr, <b>COLOR</b> *dst, uint nclrs, u32 alpha) <i>Fades color arrays srca to clr into dst.</i>
void	<b>clr_grayscale</b> ( <b>COLOR</b> *dst, const <b>COLOR</b> *src, uint nclrs) <i>Transform colors to grayscale.</i>
void	<b>clr_rgbscale</b> ( <b>COLOR</b> *dst, const <b>COLOR</b> *src, uint nclrs, <b>COLOR</b> clr) <i>Transform colors to an rgbscale.</i>
void	<b>pal_gradient</b> ( <b>COLOR</b> *pal, int first, int last) <i>Create a gradient between pal[first] and pal[last].</i>
void	<b>pal_gradient_ex</b> ( <b>COLOR</b> *pal, int first, int last, <b>COLOR</b> clr_first, <b>COLOR</b> clr_last) <i>Create a gradient between pal[first] and pal[last].</i>
void	<b>clr_adj_brightness</b> ( <b>COLOR</b> *dst, const <b>COLOR</b> *src, uint nclrs, <b>FIXED</b> bright) <i>Adjust brightness by bright.</i>
void	<b>clr_adj_contrast</b> ( <b>COLOR</b> *dst, const <b>COLOR</b> *src, uint nclrs, <b>FIXED</b> contrast) <i>Adjust contrast by contrast.</i>
void	<b>clr_adj_intensity</b> ( <b>COLOR</b> *dst, const <b>COLOR</b> *src, uint nclrs, <b>FIXED</b> intensity) <i>Adjust intensity by intensity.</i>

## Detailed Description

**Author:**

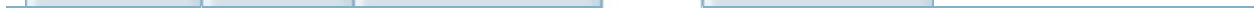
J Vijn

**Date:**

20070823 - 20070907

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tonc\_core.c File Reference**

```
#include "tonc_memmap.h" #include "tonc_core.h"
```

# Functions

void *	<b>tonccpy</b> (void *dst, const void *src, uint size) VRAM-safe cpy.
void *	<b>_toncset</b> (void *dst, u32 fill, uint size) VRAM-safe memset, internal routine.
int	<b>sqrans</b> (int seed)
u32	<b>octant</b> (int x, int y) Get the octant that (x, y) is in.
u32	<b>octant_rot</b> (int x0, int y0) Get the rotated octant that (x, y) is in.

## Variables

const u8	<b>oam_sizes</b> [3][4][2]
const <b>BG_AFFINE</b>	<b>bg_aff_default</b> = { 256, 0, 0, 256, 0, 0 }
const u32	<b>__snd_rates</b> [12]
int	<b>__qran_seed</b> = 42
<b>COLOR</b> *	<b>vid_page</b> = vid_mem_back

---

## **Detailed Description**

**Author:**

J Vijn

**Date:**

20060508 - 20060508

---

# Variable Documentation

```
const u32 __snd_rates[12]
```

**Initial value:**

```
{  
    8013, 7566, 7144, 6742,  
    6362, 6005, 5666, 5346,  
    5048, 4766, 4499, 4246  
}
```

```
const u8 oam_sizes[3][4][2]
```

**Initial value:**

```
{  
    { { 8, 8}, {16,16}, {32,32}, {64,64} },  
    { {16, 8}, {32, 8}, {32,16}, {64,32} },  
    { { 8,16}, { 8,32}, {16,32}, {32,64} },  
}
```

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tonc\_core.h File Reference**

```
#include "tonc_memmap.h" #include "tonc_memdef.h"
```

## Copying and filling routines

#define	<b>GRIT_CPY</b> (dst, name) <code>memcpy16(dst, name, name##Len/2)</code> <i>Simplified copier for GRIT-exported data.</i>
void *	<b>tonccpy</b> (void *dst, const void *src, uint size) VRAM-safe <i>cpy</i> .
void *	<b>_toncset</b> (void *dst, u32 fill, uint size) VRAM-safe <i>memset</i> , <i>internal routine</i> .
INLINE void *	<b>toncset</b> (void *dst, u8 src, uint count) VRAM-safe <i>memset</i> , <i>byte version</i> . <i>Size in bytes</i> .
INLINE void *	<b>toncset16</b> (void *dst, u16 src, uint count) VRAM-safe <i>memset</i> , <i>halfword version</i> . <i>Size in hwords</i> .
INLINE void *	<b>toncset32</b> (void *dst, u32 src, uint count) VRAM-safe <i>memset</i> , <i>word version</i> . <i>Size in words</i> .
void	<b>memset16</b> (void *dst, u16 hw, uint hwcount) <i>Fastfill for halfwords, analogous to memset()</i> .
IWRAM_CODE void	<b>memset32</b> (void *dst, u32 wd, uint wdcount) <i>Fast-fill by words, analogous to memset()</i> .
IWRAM_CODE void	<b>memcpy32</b> (void *dst, const void *src, uint wdcount)

## Random numbers

#define	<b>QRAN_SHIFT</b> 15
#define	<b>QRAN_MASK</b> ((1<<QRAN_SHIFT)-1)
#define	<b>QRAN_MAX</b> QRAN_MASK
int	<b>qrان</b> (int seed)
INLINE int	<b>qrان</b> (void) <i>Quick (and very dirty) pseudo-random number generator.</i>
INLINE int	<b>qrان_range</b> (int min, int max) <i>Ranged random number.</i>

## Defines

#define	<b>countof</b> ( <u>_array</u> ) ( <u>sizeof(_array)/sizeof(_array[0])</u> ) <i>Get the number of elements in an array.</i>
#define	<b>DMA_TRANSFER</b> ( <u>_dst</u> , <u>_src</u> , <u>count</u> , <u>ch</u> , <u>mode</u> ) <i>General purpose DMA transfer macro.</i>
#define	<b>SND_RATE</b> ( <u>note</u> , <u>oct</u> ) ( <u>2048-(__snd_rates[note]&gt;&gt;(4+(oct)))</u> ) <i>Gives the period of a note for the tone-gen registers.</i>
#define	<b>STR</b> ( <u>x</u> ) # <u>x</u>
#define	<b>XSTR</b> ( <u>x</u> ) <b>STR</b> ( <u>x</u> ) <i>Create text string from a literal.</i>

## Simple bit macros

#define	<b>BIT</b> ( <u>n</u> ) ( <u>1&lt;&lt;(n)</u> ) <i>Create value with bit n set.</i>
#define	<b>BIT_SHIFT</b> ( <u>a</u> , <u>n</u> ) ( <u>(a)&lt;&lt;(n)</u> ) <i>Shift a by n.</i>
#define	<b>BIT_MASK</b> ( <u>len</u> ) ( <u>BIT(len)-1</u> ) <i>Create a bitmask len bits long.</i>
#define	<b>BIT_SET</b> ( <u>y</u> , <u>flag</u> ) ( <u>y  = (flag)</u> ) <i>Set the flag bits in word.</i>
#define	<b>BIT_CLEAR</b> ( <u>y</u> , <u>flag</u> ) ( <u>y &amp;= ~flag</u> ) <i>Clear the flag bits in word.</i>
#define	<b>BIT_FLIP</b> ( <u>y</u> , <u>flag</u> ) ( <u>y ^= (flag)</u> ) <i>Flip the flag bits in word.</i>
#define	<b>BIT_EQ</b> ( <u>y</u> , <u>flag</u> ) ( <u>((y)&amp;(flag)) == (flag)</u> ) <i>Test whether all the flag bits in word are set.</i>
#define	<b>BF_MASK</b> ( <u>shift</u> , <u>len</u> ) ( <u>BIT_MASK(len)&lt;&lt;(shift)</u> ) <i>Create a bitmask of length len starting at bit shift.</i>
#define	<b>_BF_GET</b> ( <u>y</u> , <u>shift</u> , <u>len</u> ) ( <u>((y)&gt;&gt;(len))&amp;(shift)</u> ) <i>Retrieve a bitfield mask of length starting at bit shift from y.</i>
#define	<b>_BF_PREP</b> ( <u>x</u> , <u>shift</u> , <u>len</u> ) ( <u>((x)&amp;BIT_MASK(len))&lt;&lt;(shift)</u> ) <i>Prepare a bitmask for insertion or combining.</i>
#define	<b>_BF_SET</b> ( <u>y</u> , <u>x</u> , <u>shift</u> , <u>len</u> ) ( <u>y = ((y) &amp;~ BF_MASK(shift, len))  </u> <b>_BF_SET</b> ( <u>x</u> , <u>shift</u> , <u>len</u> ) ) <i>Set a bitfield in a word.</i>

**\_BF\_PREP(x, shift, len) )**

*Insert a new bitfield value x into y.*

## some EVIL bit-field operations, >:)

*These allow you to mimic bitfields with macros. Most of the bitfields in the registers have foo\_SHIFT and foo\_SHIFT macros indicating the mask and shift values of the bitfield named foo in a variable. These macros let you prepare, get and set the bitfields.*

#define	<b>BNF_PREP</b> (x, name) ( ((x)<<name##_SHIFT) & name##_MASK ) <i>Prepare a named bit-field for for insterion or combination.</i>
#define	<b>BNF_GET</b> (y, name) ( ((y) & name##_MASK)>>name##_SHIFT ) <i>Get the value of a named bitfield from y. Equivalent to (var=) y.name.</i>
#define	<b>BNF_SET</b> (y, x, name) ( y = ((y)&~name##_MASK)   BNF_PREP(x,name) ) <i>Set a named bitfield in y to x. Equivalent to y.name= x.</i>
#define	<b>BNF_CMP</b> (y, x, name) ( ((y)&name##_MASK) == (x) ) <i>Compare a named bitfield to named literal x.</i>
#define	<b>BNF_PREP2</b> (x, name) ( (x) & name##_MASK ) <i>Massage x for use in bitfield name with pre-shifted x.</i>
#define	<b>BNF_GET2</b> (y, name) ( (y) & name##_MASK ) <i>Get the value of bitfield name from y, but don't down-shift.</i>
#define	<b>BNF_SET2</b> (y, x, name) ( y = ((y)&~name##_MASK)   BNF_PREP2(x,name) ) <i>Set bitfield name from y to x with pre-shifted x.</i>

## Inline assembly

#define	<b>ASM_CMT</b> (str) asm volatile("@# " str) <i>Assembly comment.</i>
#define	<b>ASM_BREAK()</b> asm volatile("\tmov\t\tr11, r11") <i>No\$gba breakpoint.</i>
#define	<b>ASM_NOP()</b> asm volatile("\tnop") <i>No-op; wait a bit.</i>

## Enumerations

```
enum eSndNotelId {  
    NOTE_C = 0, NOTE_CIS, NOTE_D, NOTE_DIS,  
    NOTE_E, NOTE_F, NOTE_FIS, NOTE_G,  
    NOTE_GIS, NOTE_A, NOTE_BES, NOTE_B  
}
```

# Functions

INLINE u32	<b>bf_get</b> (u32 y, uint shift, uint len) <i>Get len long bitfield from y, starting at shift.</i>
INLINE u32	<b>bf_merge</b> (u32 y, u32 x, uint shift, uint len) <i>Merge x into an len long bitfield from y, starting at shift.</i>
INLINE u32	<b>bf_clamp</b> (int x, uint len) <i>Clamp to within the range allowed by len bits.</i>
INLINE int	<b>bit_tribool</b> (u32 flags, uint plus, uint minus) <i>Gives a tribool (-1, 0, or +1) depending on the state of some bits.</i>
INLINE u32	<b>ROR</b> (u32 x, uint ror) <i>Rotate bits right. Yes, this does lead to a ror instruction.</i>
INLINE uint	<b>align</b> (uint x, uint width) <i>Align x to the next multiple of width.</i>
INLINE void	<b>dma_cpy</b> (void *dst, const void *src, uint count, uint ch, u32 mode) <i>Generic DMA copy routine.</i>
INLINE void	<b>dma_fill</b> (void *dst, volatile u32 src, uint count, uint ch, u32 mode) <i>Generic DMA fill routine.</i>
INLINE void	<b>dma3_cpy</b> (void *dst, const void *src, uint size) <i>Specific DMA copier, using channel 3, word transfers.</i>
INLINE void	<b>dma3_fill</b> (void *dst, volatile u32 src, uint size) <i>Specific DMA filler, using channel 3, word transfers.</i>
INLINE void	<b>profile_start</b> (void) <i>Start a profiling run.</i>
INLINE uint	<b>profile_stop</b> (void) <i>Stop a profiling run and return the time since its start.</i>

## Repeated-value creators

*These function take a hex-value and duplicate it to all fields, like 0x88 -> 0x88888888.*

INLINE u16	<b>dup8</b> (u8 x)
------------	--------------------

	<i>Duplicate a byte to form a halfword: 0x12 -&gt; 0x1212.</i>
INLINE u32	<b>dup16</b> (u16 x) <i>Duplicate a halfword to form a word: 0x1234 -&gt; 0x12341234.</i>
INLINE u32	<b>quad8</b> (u8 x) <i>Quadruple a byte to form a word: 0x12 -&gt; 0x12121212.</i>
INLINE u32	<b>octup</b> (u8 x) <i>Octuple a nybble to form a word: 0x1 -&gt; 0x11111111.</i>

## Packing routines.

INLINE u16	<b>bytes2hword</b> (u8 b0, u8 b1) <i>Pack 2 bytes into a word. Little-endian order.</i>
INLINE u32	<b>bytes2word</b> (u8 b0, u8 b1, u8 b2, u8 b3) <i>Pack 4 bytes into a word. Little-endian order.</i>
INLINE u32	<b>hword2word</b> (u16 h0, u16 h1) <i>Pack 2 bytes into a word. Little-endian order.</i>

## Sector checking

u32	<b>octant</b> (int x, int y) <i>Get the octant that (x, y) is in.</i>
u32	<b>octant_rot</b> (int x0, int y0) <i>Get the rotated octant that (x, y) is in.</i>

## Variables

const uint	<b>__snd_rates</b> [12]
const u8	<b>oam_sizes</b> [3][4][2]
const <b>BG_AFFINE</b>	<b>bg_aff_default</b>
<b>COLOR</b> *	<b>vid_page</b>
int	<b>__qran_seed</b>

---

## **Detailed Description**

**Author:**

J Vijn

**Date:**

20060508 - 20080128

---

# Define Documentation

```
#define SND_RATE( note,  
                oct ) ( 2048-(__snd_rates[note]>>(4+(oct)))
```

Gives the period of a note for the tone-gen registers.

GBA sound range: 8 octaves: [-2, 5];  $8 \times 12 = 96$  notes (kinda).

## Parameters:

*note* ID (range: [0,11>). See eSndNoteld.

*oct* octave (range [-2,4>).

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tonc\_input.c File Reference**

```
#include "tonc_input.h" #include "tonc_bios.h"
```

# Data Structures

struct **REPEAT\_REC**

*Repeated keys struct.* [More...](#)

## Functions

void	<b>key_poll</b> (void)	<i>Poll for keystates and repeated keys.</i>
void	<b>key_wait_till_hit</b> (u16 key)	<i>Wait until key is hit.</i>
u32	<b>key_repeat</b> (u32 keys)	<i>Get status of repeated keys.</i>
void	<b>key_repeat_mask</b> (u32 mask)	<i>Set repeat mask. Only these keys will be considered for repeats.</i>
void	<b>key_repeat_limits</b> (uint delay, uint repeat)	<i>Set the delay and repeat limits for repeated keys.</i>

## Variables

u16	<b>__key_curr</b> = 0
u16	<b>__key_prev</b> = 0
<b>REPEAT_REC</b>	<b>__key_rpt</b> = { 0, KEY_MASK, 60, 60, 30 }

---

## Detailed Description

**Author:**

J Vijn

**Date:**

20070406 - 20070406

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tonc\_input.h** File Reference

```
#include "tonc_memmap.h" #include "tonc_memdef.h"  
#include "tonc_core.h"
```

## Defines

#define	<b>KEY_FULL</b> 0xFFFFFFFF
	<i>Define for checking all keys.</i>
#define	<b>KEY_DOWN_NOW(key)</b> (~(REG_KEYINPUT) & key)
#define	<b>KEY_UP_NOW(key)</b> ( (REG_KEYINPUT) & key)
#define	<b>KEY_EQ(key_fun, keys)</b> ( key_fun(keys) == (keys) )
#define	<b>KEY_TRIBOOL(fnKey, plus, minus)</b> ( bit_tribool(fnKey(KEY_FULL), plus, minus) )

## Enumerations

```
enum eKeyIndex {  
    KI_A = 0, KI_B, KI_SELECT, KI_START,  
    KI_RIGHT, KI_LEFT, KI_UP, KI_DOWN,  
    KI_R, KI_L, KI_MAX  
}
```

# Functions

void	<b>key_wait_for_clear</b> (u32 key)
void	<b>key_wait_till_hit</b> (u16 key) <i>Wait until key is hit.</i>

## Basic synchonous keystates

void	<b>key_poll</b> () <i>Poll for keystates and repeated keys.</i>
INLINE u32	<b>key_curr_state</b> (void) <i>Get current keystate.</i>
INLINE u32	<b>key_prev_state</b> (void) <i>Get previous key state.</i>
INLINE u32	<b>key_is_down</b> (u32 key) <i>Gives the keys of key that are currently down.</i>
INLINE u32	<b>key_is_up</b> (u32 key) <i>Gives the keys of key that are currently up.</i>
INLINE u32	<b>key_was_down</b> (u32 key) <i>Gives the keys of key that were previously down.</i>
INLINE u32	<b>key_was_up</b> (u32 key) <i>Gives the keys of key that were previously up.</i>

## Transitional keystates

INLINE u32	<b>key_transit</b> (u32 key) <i>Gives the keys of key that are different from before.</i>
INLINE u32	<b>key_held</b> (u32 key) <i>Gives the keys of key that are being held down.</i>
INLINE u32	<b>key_hit</b> (u32 key) <i>Gives the keys of key that are pressed (down now but not before).</i>
INLINE u32	<b>key_released</b> (u32 key) <i>Gives the keys of key that are being released.</i>

## Tribools

INLINE int	<b>key_tri_horz</b> (void) <i>Horizontal tribool (right,left)=(+,-).</i>
INLINE int	<b>key_tri_vert</b> (void) <i>Vertical tribool (down,up)=(+,-).</i>
INLINE int	<b>key_tri_shoulder</b> (void) <i>Shoulder-button tribool (R,L)=(+,-).</i>
INLINE int	<b>key_tri_fire</b> (void) <i>Fire-button tribool (A,B)=(+,-).</i>

## Key repeats

u32	<b>key_repeat</b> (u32 keys) <i>Get status of repeated keys.</i>
void	<b>key_repeat_mask</b> (u32 mask) <i>Set repeat mask. Only these keys will be considered for repeats.</i>
void	<b>key_repeat_limits</b> (uint delay, uint repeat) <i>Set the delay and repeat limits for repeated keys.</i>

## Variables

u16	<b>__key_curr</b>
u16	<b>__key_prev</b>

---

## Detailed Description

**Author:**

J Vijn

**Date:**

20060508 - 20070406

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tonc\_irq.c File Reference**

```
#include "tonc_memmap.h" #include "tonc_memdef.h"
#include "tonc_core.h"
#include "tonc_irq.h"
```

# Data Structures

struct	<b>IRQ_SENDER</b>
	<i>IRQ Sender information.</i> <a href="#">More...</a>

# Functions

void	<b>irq_init</b> ( <b>fnptra</b> isr) <i>Initialize irq business.</i>
<b>fnptra</b>	<b>irq_set_master</b> ( <b>fnptra</b> isr) <i>Set a master ISR.</i>
<b>fnptra</b>	<b>irq_set</b> (enum <b>eIrqIndex</b> irq_id, <b>fnptra</b> isr, u32 opts) <i>General IRQ manager.</i>
<b>fnptra</b>	<b>irq_add</b> (enum <b>eIrqIndex</b> irq_id, <b>fnptra</b> isr) <i>Add a specific ISR.</i>
<b>fnptra</b>	<b>irq_delete</b> (enum <b>eIrqIndex</b> irq_id) <i>Remove an ISR.</i>
void	<b>irq_enable</b> (enum <b>eIrqIndex</b> irq_id)
void	<b>irq_disable</b> (enum <b>eIrqIndex</b> irq_id)

## Variables

**IRQ\_REC**    **\_\_isr\_table [II\_MAX+1]**

---

# Detailed Description

**Author:**

J Vijn

**Date:**

20060908 - 20060928

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tonc\_irq.h File Reference**

```
#include "tonc_memmap.h" #include "tonc_memdef.h"
```

# Data Structures

struct	<b>IRQ_REC</b>
	<i>Struct for prioritized irq table.</i> <a href="#">More...</a>

## Defines

#define	<b>IRQ_INIT()</b> irq_init(NULL)
	<i>Default <b>irq_init()</b> call: use <b>irq_master_nest()</b> for switchboard.</i>
#define	<b>IRQ_SET(irq_id)</b> irq_set(lI_##irq_id, NULL, ISR_DEF)
	<i>Default <b>irq_set()</b> call: no isr, add to back of priority stack.</i>
#define	<b>IRQ_ADD(irq_id)</b> irq_add(lI_##irq_id, NULL)

### Options for **irq\_set**

#define	<b>ISR_LAST</b> 0x0040
	<i>Last isr in line (Lowest priority).</i>
#define	<b>ISR_REPLACE</b> 0x0080
	<i>Replace old isr if existing (prio ignored).</i>
#define	<b>ISR_PRIO_MASK</b> 0x003F
	<i>Last isr in line (Lowest priority).</i>
#define	<b>ISR_PRIO_SHIFT</b> 0
	<i>Last isr in line (Lowest priority).</i>
#define	<b>ISR_PRIO(n)</b> ((n)<<ISR_PRIO_SHIFT)
	<i>Last isr in line (Lowest priority).</i>
#define	<b>ISR_DEF</b> (ISR_LAST ISR_REPLACE)
	<i>Last isr in line (Lowest priority).</i>

## Enumerations

enum	<pre>eIrqIndex {     II_VBLANK = 0, II_HBLANK, II_VCOUNT, II_TIMER0,     II_TIMER1, II_TIMER2, II_TIMER3, II_SERIAL,     II_DMA0, II_DMA1, II_DMA2, II_DMA3,     II_KEYPAD, II_GAMEPAK, II_MAX }</pre>
------	--

*IRQ indices, to be used in most functions.*

## Functions

IWRAM_CODE void	<b>isr_master</b> (void)
IWRAM_CODE void	<b>isr_master_nest</b> (void)
void	<b>irq_init</b> ( <b>fnptra</b> isr) <i>Initialize irq business.</i>
<b>fnptra</b>	<b>irq_set_master</b> ( <b>fnptra</b> isr) <i>Set a master ISR.</i>
<b>fnptra</b>	<b>irq_add</b> (enum <b>eIRQIndex</b> irq_id, <b>fnptra</b> isr) <i>Add a specific ISR.</i>
<b>fnptra</b>	<b>irq_delete</b> (enum <b>eIRQIndex</b> irq_id) <i>Remove an ISR.</i>
<b>fnptra</b>	<b>irq_set</b> (enum <b>eIRQIndex</b> irq_id, <b>fnptra</b> isr, u32 opts) <i>General IRQ manager.</i>
void	<b>irq_enable</b> (enum <b>eIRQIndex</b> irq_id)
void	<b>irq_disable</b> (enum <b>eIRQIndex</b> irq_id)

## Variables

**IRQ\_REC**    **\_\_isr\_table [II\_MAX+1]**

---

# Detailed Description

**Author:**

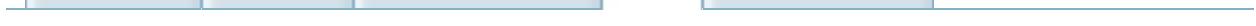
J Vijn

**Date:**

20060508 - 20080326

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tonc\_legacy.h File Reference**

---

## Detailed Description

**Author:**

J Vijn

**Date:**

20070131 - 20070131

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tonc\_libgba.h File Reference**

```
#include "tonc_memmap.h"
```

## Defines

```
#define VRAM MEM_VRAM
#define IWRAM MEM_IWRAM
#define EWRAM MEM_EWRAM
#define EWRAM_END (MEM_EWRAM+EWRAM_SIZE)
#define SRAM MEM_SRAM
#define SystemCall swi_call
#define FILL CS_FILL
#define COPY16 CS_CPY16
#define COPY32 CS_CPY32
#define DMA16 DMA_16
#define DMA32 DMA_32
#define GAMEPAK_DRQ DMA_GAMEPAK
#define DMA_IMMEDIATE DMA_AT_NOW
#define DMA_VBLANK DMA_AT_VBLANK
#define DMA_HBLANK DMA_AT_HBLANK
#define DMA_Copy(channel, source, dest, mode) DMA_TRANSFER(dest, source, mode, channel, DMA_ON)
#define DMA0COPY(source, dest, mode) DMA_Copy(0,(source),(dest), (mode))
#define DMA1COPY(source, dest, mode) DMA_Copy(1,(source),(dest), (mode))
#define DMA2COPY(source, dest, mode) DMA_Copy(2,(source),(dest), (mode))
#define DMA3COPY(source, dest, mode) DMA_Copy(3,(source),(dest), (mode))
#define DPAD KEY_DIR
#define KEYIRQ_ENABLE KCNT_IRQ
#define KEYIRQ_OR KCNT_OR
#define KEYIRQ_AND KCNT_AND
#define scanKeys key_poll
#define keysDown() key_hit(KEY_FULL)
#define keysDownRepeat() key_repeat(KEY_FULL)
#define keysUp() key_released(KEY_FULL)
```

```
#define keysHeld() key_is_down(KEY_FULL)
#define setRepeat key_repeat_limits
#define INT_VECTOR REG_ISR_MAIN
#define MODE32_NORMAL MBOOT_NORMAL
#define MODE16_MULTI MBOOT_MULTI
#define MODE32_2MHZ MBOOT_FAST
#define PCM_DMA_BUF 1584
#define MAX_DIRECTSOUND_CHANNELS 12
#define SND1_L_ENABLE SDMG_LSQR1
#define SND2_L_ENABLE SDMG_LSQR2
#define SND3_L_ENABLE SDMG_LWAVE
#define SND4_L_ENABLE SDMG_LNOISE
#define SND1_R_ENABLE SDMG_RSQR1
#define SND2_R_ENABLE SDMG_RSQR2
#define SND3_R_ENABLE SDMG_RWAVE
#define SND4_R_ENABLE SDMG_RNOISE
#define SNDA_VOL_50 SDS_A50
#define SNDA_VOL_100 SDS_A100
#define SNDB_VOL_50 SDS_B50
#define SNDB_VOL_100 SDS_B100
#define SNDA_R_ENABLE SDS_AR
#define SNDA_L_ENABLE SDS_AL
#define SNDA_RESET_FIFO SDS_ARESET
#define SNDB_R_ENABLE SDS_BR
#define SNDB_L_ENABLE SDS_BL
#define SNDB_RESET_FIFO SDS_BRESET
#define WAVE_RAM ((vu16*)(REG_BASE+0x0090))
#define OBJATTR OBJ_ATTR
#define OBJAFFINE OBJ_AFFINE
#define OAM oam_mem
#define OBJ_BASE_ADDR ((void*)(tile_mem[4]))
#define SPRITE_GFX ((u16*)(tile_mem[4]))
#define BITMAP_OBJ_BASE_ADDR ((void*)(tile_mem[5]))
#define SQUARE (ATTR0_SQUARE>>14)
#define WIDE (ATTR0_WIDE>>14)
#define TALL (ATTR0_TALL>>14)
```

```
#define ATTR0_NORMAL ATTR0_REG

#define ATTR0_ROTSCALE ATTR0_AFF
#define ATTR0_DISABLED ATTR0_HIDE
#define ATTR0_ROTSCALE_DOUBLE ATTR0_AFF_DBL_BIT
#define ATTR0_COLOR_16 ATTR0_4BPP
#define ATTR0_COLOR_256 ATTR0_8BPP
#define ATTR1_FLIP_X ATTR1_HFLIP
#define ATTR1_FLIP_Y ATTR1_VFLIP
#define ATTR1_ROTDATA(n) ATTR1_AFF_ID(m)
#define ATTR2_PRIORITY(n) ATTR2_PRIO(m)
#define ATTR2_PALETTE(n) ATTR2_PALBANK(m)
#define OBJ_ROT_SCALE_ON ATTR0_AFF
#define OBJ_DISABLE ATTR0_HIDE
#define OBJ_DOUBLE ATTR0_AFF_DBL_BIT
#define OBJ_TRANSLUCENT ATTR0_BLEND
#define OBJ_OBJWINDOW ATTR0_WINDOW
#define OBJ_MOSAIC ATTR0_MOSAIC
#define OBJ_16_COLOR ATTR0_4BPP
#define OBJ_256_COLOR ATTR0_8BPP
#define OBJ_SQUARE ATTR0_SQUARE
#define OBJ_WIDE ATTR0_WIDE
#define OBJ_TALL ATTR0_TALL
#define OBJ_Y(m) ATTR0_Y(m)
#define OBJ_MODE(m) ((m)<<10)
#define OBJ_SHAPE(m) ATTR0_SHAPE(m)
#define OBJ_HFLIP ATTR1_HFLIP
#define OBJ_VFLIP ATTR1_VFLIP
#define OBJ_X(m) ATTR1_X(m)
#define OBJ_ROT_SCALE(m) ATTR1_AFF_ID(m)
#define OBJ_SIZE(m) ATTR1_SIZE(m)
#define OBJ_CHAR(m) ATTR2_ID(m)
#define OBJ_PRIORITY(m) ATTR2_PRIO(m)
#define OBJ_PALETTE(m) ATTR2_PALBANK(m)
#define TIMER_COUNT TM CASCADE
#define TIMER_IRQ TM IRQ
```

#define	<b>TIMER_START</b> TM_ENABLE
#define	<b>BG_COLORS</b> pal_bg_mem
#define	<b>BG_PALETTE</b> pal_bg_mem
#define	<b>OBJ_COLORS</b> pal_obj_mem
#define	<b>SPRITE_PALETTE</b> pal_obj_mem
#define	<b>MODE_0</b> DCNT_MODE0
#define	<b>MODE_1</b> DCNT_MODE1
#define	<b>MODE_2</b> DCNT_MODE2
#define	<b>MODE_3</b> DCNT_MODE3
#define	<b>MODE_4</b> DCNT_MODE4
#define	<b>MODE_5</b> DCNT_MODE5
#define	<b>BACKBUFFER</b> DCNT_PAGE
#define	<b>OBJ_1D_MAP</b> DCNT_OBJ_1D
#define	<b>LCDC_OFF</b> DCNT_BLANK
#define	<b>BG0_ENABLE</b> DCNT_BG0
#define	<b>BG1_ENABLE</b> DCNT_BG1
#define	<b>BG2_ENABLE</b> DCNT_BG2
#define	<b>BG3_ENABLE</b> DCNT_BG3
#define	<b>OBJ_ENABLE</b> DCNT_OBJ
#define	<b>WIN0_ENABLE</b> DCNT_WIN0
#define	<b>WIN1_ENABLE</b> DCNT_WIN1
#define	<b>OBJ_WIN_ENABLE</b> DCNT_WINOBJ
#define	<b>BG_ALL_ENABLE</b> (0x0F00)
#define	<b>LCDC_VBL_FLAG</b> DSTAT_IN_VBL
#define	<b>LCDC_HBL_FLAG</b> DSTAT_IN_HBL
#define	<b>LCDC_VCNT_FLAG</b> DSTAT_IN_VCT
#define	<b>LCDC_VBL</b> DSTAT_VBL_IRQ
#define	<b>LCDC_HBL</b> DSTAT_HBL_IRQ
#define	<b>LCDC_VCNT</b> DSTAT_VCT_IRQ
#define	<b>VCOUNT</b> DSTAT_VCT
#define	<b>bg_scroll</b> <b>BG_POINT</b>
#define	<b>BGCTRL</b> REG_BGCNT
#define	<b>BG_OFFSET</b> REG_BG_OFS
#define	<b>BG_16_COLOR</b> BG_4BPP
#define	<b>BG_256_COLOR</b> BG_8BPP

#define	<b>BG_SIZE_0</b>	BG_SIZE0
#define	<b>BG_SIZE_1</b>	BG_SIZE1
#define	<b>BG_SIZE_2</b>	BG_SIZE2
#define	<b>BG_SIZE_3</b>	BG_SIZE3
#define	<b>BG_WID_32</b>	BG_SIZE_0
#define	<b>BG_WID_64</b>	BG_SIZE_1
#define	<b>BG_HT_32</b>	BG_SIZE_0
#define	<b>BG_HT_64</b>	BG_SIZE_2
#define	<b>TEXTBG_SIZE_256x256</b>	BG_REG_32x32
#define	<b>TEXTBG_SIZE_512x256</b>	BG_REG_64x32
#define	<b>TEXTBG_SIZE_256x512</b>	BG_REG_32x64
#define	<b>TEXTBG_SIZE_512x512</b>	BG_REG_64x64
#define	<b>ROTBG_SIZE_128x128</b>	BG_AFF_16x16
#define	<b>ROTBG_SIZE_256x256</b>	BG_AFF_32x32
#define	<b>ROTBG_SIZE_512x512</b>	BG_AFF_64x64
#define	<b>ROTBG_SIZE_1024x1024</b>	BG_AFF_128x128
#define	<b>BG_PRIORITY(m)</b>	BG_PRIO(m)
#define	<b>BG_TILE_BASE(m)</b>	BG_CBB(m)
#define	<b>BG_MAP_BASE(m)</b>	BG_SBB(m)
#define	<b>TILE_BASE(m)</b>	BG_CBB(m)
#define	<b>MAP_BASE(m)</b>	BG_SBB(m)
#define	<b>CHAR_BASE_ADR(m)</b>	(void*)(tile_mem[m])
#define	<b>CHAR_BASE_BLOCK(m)</b>	(void*)(tile_mem[m])
#define	<b>TILE_BASE_ADR(m)</b>	(void*)(tile_mem[m])
#define	<b>MAP_BASE_ADR(m)</b>	(void*)(se_mem[m])
#define	<b>SCREEN_BASE_BLOCK(m)</b>	(void*)(se_mem[m])
#define	<b>MODE3_LINE</b>	M3LINE
#define	<b>MODE5_LINE</b>	M5LINE
#define	<b>MODE3_FB</b>	m3_mem
#define	<b>MODE5_FB</b>	m5_mem
#define	<b>MODE5_BB</b>	m5_mem_back
#define	<b>RGB5</b>	RGB15
#define	<b>RGB8(r, g, b)</b>	((r)>>3)   (((g)>>3)<<5)   (((b)>>3)<<10) )

## SIOCNT bits

#define	<b>SIO_8BIT</b>	0x0000
<i>Normal 8-bit communication mode.</i>		
#define	<b>SIO_32BIT</b>	0x1000
<i>Normal 32-bit communication mode.</i>		
#define	<b>SIO_MULTI</b>	0x2000
<i>Multi-play communication mode.</i>		
#define	<b>SIO_UART</b>	0x3000
<i>UART communication mode.</i>		
#define	<b>SIO_IRQ</b>	0x4000
<i>Enable serial irq.</i>		

## Baud rate settings

#define	<b>SIO_9600</b>	0x0000
#define	<b>SIO_38400</b>	0x0001
#define	<b>SIO_57600</b>	0x0002
#define	<b>SIO_115200</b>	0x0003
#define	<b>SIO_CLK_INT</b>	(1<<0)
<i>Select internal clock.</i>		
#define	<b>SIO_2MHZ_CLK</b>	(1<<1)
<i>Select 2MHz clock.</i>		
#define	<b>SIO_RDY</b>	(1<<2)
<i>Opponent SO state.</i>		
#define	<b>SIO_SO_HIGH</b>	(1<<3)
<i>Our SO state.</i>		
#define	<b>SIO_START</b>	(1<<7)

## SIO modes set with REG\_RCNT

#define	<b>R_NORMAL</b>	0x0000
#define	<b>R_MULTI</b>	0x0000
#define	<b>R_UART</b>	0x0000
#define	<b>R_GPIO</b>	0x8000
#define	<b>R_JOYBUS</b>	0xC000

## General purpose mode control bits used with REG\_RCNT

#define	<b>GPIO_SC</b>	0x0001
---------	----------------	--------

```
#define GPIO_SD 0x0002
#define GPIO_SI 0x0004
#define GPIO_SO 0x0008
#define GPIO_SC_IO 0x0010
#define GPIO_SD_IO 0x0020

#define GPIO_SI_IO 0x0040
#define GPIO_SO_IO 0x0080
#define GPIO_SC_INPUT 0x0000
#define GPIO_SD_INPUT 0x0000
#define GPIO_SI_INPUT 0x0000
#define GPIO_SO_INPUT 0x0000
#define GPIO_SC_OUTPUT 0x0010
#define GPIO_SD_OUTPUT 0x0020
#define GPIO_SI_OUTPUT 0x0040
#define GPIO_SO_OUTPUT 0x0080
```

### Sound 3 control bits

```
#define SOUND3_STEP32 (0<<5)
#define SOUND3_STEP64 (1<<5)
#define SOUND3_SETBANK(n) (n<<6)
#define SOUND3_PLAY (1<<7)
#define SOUND3_STOP (0<<7)
```

## Typedefs

```
typedef void(* IntFn )(void)
```

## Enumerations

```
enum SPRITE_SIZECODE {
    Sprite_8x8 = 0, Sprite_16x16, Sprite_32x32, Sprite_64x64,
    Sprite_16x8 = 0, Sprite_32x8, Sprite_32x16, Sprite_64x32,
    Sprite_8x16 = 0, Sprite_8x32, Sprite_16x32, Sprite_32x64
}
```

---

## Detailed Description

**Author:**

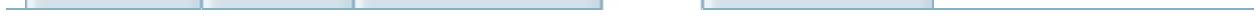
J Vijn

**Date:**

20070921 - 20070921

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tonc\_math.c File Reference**

```
#include "tonc_memmap.h" #include "tonc_core.h"  
#include "tonc_math.h"
```

## Functions

int	<b>pt_in_rect</b> (const POINT *pt, const RECT *rc) <i>Is a point inside a rectangle.</i>
<b>VECTOR</b> *	<b>vec_cross</b> ( <b>VECTOR</b> *vd, const <b>VECTOR</b> *va, const <b>VECTOR</b> *vb)
RECT *	<b>rc_normalize</b> (RECT *rc)

---

# Detailed Description

**Author:**

J Vijn

**Date:**

20060508 - 20060508

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tonc\_math.h File Reference**

```
#include "tonc_types.h"
```

# Data Structures

struct	<b>POINT32</b>
	<i>2D Point struct</i> <a href="#">More...</a>
struct	<b>RECT32</b>
	<i>Rectangle struct.</i> <a href="#">More...</a>
struct	<b>VECTOR</b>
	<i>Vector struct.</i> <a href="#">More...</a>

## core math macros

#define	<b>ABS</b> (x) ( (x)>=0 ? (x) : -(x) ) <i>Get the absolute value of x.</i>
#define	<b>SGN</b> (x) ( (x)>=0 ? 1 : -1 ) <i>Get the sign of x.</i>
#define	<b>SGN2</b> SGN <i>Get the absolute value of x.</i>
#define	<b>SGN3</b> (x) ( (x)>0 ? 1 : ( (x)<0 ? -1 : 0 ) ) <i>Tri-state sign: -1 for negative, 0 for 0, +1 for positive.</i>
#define	<b>MAX</b> (a, b) ( ((a) > (b)) ? (a) : (b) ) <i>Get the maximum of a and b.</i>
#define	<b>MIN</b> (a, b) ( ((a) < (b)) ? (a) : (b) ) <i>Get the minimum of a and b.</i>
#define	<b>SWAP2</b> (a, b) do { a=(a)-(b); b=(a)+(b); a=(b)-(a); } while(0) <i>In-place swap.</i>
#define	<b>SWAP</b> SWAP2 <i>Get the absolute value of x.</i>
#define	<b>SWAP3</b> (a, b, tmp) do { (tmp)=(a); (a)=(b); (b)=(tmp); } while(0) <i>Swaps a and b, using tmp as a temporary.</i>
INLINE int	<b>sgn</b> (int x) <i>Get the sign of x.</i>
INLINE int	<b>sgn3</b> (int x) <i>Tri-state sign of x: -1 for negative, 0 for 0, +1 for positive.</i>
INLINE int	<b>max</b> (int a, int b) <i>Get the maximum of a and b.</i>
INLINE int	<b>min</b> (int a, int b) <i>Get the minimum of a and b.</i>

## Boundary response macros

#define	<b>IN_RANGE</b> (x, min, max) ( ((x)>=(min)) && ((x)<(max)) ) <i>Range check.</i>
#define	<b>CLAMP</b> (x, min, max) ( (x)>=(max) ? ((max)-1) : ( ((x)<(min)) ? (min) : (x) ) ) <i>Truncates x to stay in range [min, max].</i>
#define	<b>REFLECT</b> (x, min, max) ( (x)>=(max) ? 2*((max)-1)-(x) : ( ((x)<(min)) ? 2*(min)-(x) : (x) ) ) <i>Reflects x at boundaries min and max.</i>
#define	<b>WRAP</b> (x, min, max) ( (x)>=(max) ? (x)+(min)-(max) : ( ((x)<(min)) ? (x)+(max)-(min) : (x) ) ) <i>Wraps x to stay in range [min, max].</i>
INLINE BOOL	<b>in_range</b> (int x, int min, int max) <i>Range check.</i>
INLINE int	<b>clamp</b> (int x, int min, int max) <i>Truncates x to stay in range [min, max].</i>
INLINE int	<b>reflect</b> (int x, int min, int max) <i>Reflects x at boundaries min and max.</i>
INLINE int	<b>wrap</b> (int x, int min, int max) <i>Wraps x to stay in range [min, max].</i>

## Defines

#define	<b>FIX_SHIFT</b> 8
#define	<b>FIX_SCALE</b> ( 1<<FIX_SHIFT )
#define	<b>FIX_MASK</b> ( FIX_SCALE-1 )
#define	<b>FIX_SCALEF</b> ( (float)FIX_SCALE )
#define	<b>FIX_SCALEF_INV</b> ( 1.0/FIX_SCALEF )
#define	<b>FIX_ONE</b> FIX_SCALE
#define	<b>FX_RECIPROCAL</b> (a, fp) ( ((1<<(fp))+(a)-1)/(a) ) <i>Get the fixed point reciprocal of a, in fp fractional bits.</i>
#define	<b>FX_RECIMUL</b> (x, a, fp) ( ((x)*((1<<(fp))+(a)-1)/(a))>>(fp) ) <i>Perform the division x/ a by reciprocal multiplication.</i>
#define	<b>SIN_LUT_SIZE</b> 514
#define	<b>DIV_LUT_SIZE</b> 257

# Functions

INLINE <b>FIXED</b>	<b>int2fx</b> (int d) <i>Convert an integer to fixed-point.</i>
INLINE <b>FIXED</b>	<b>float2fx</b> (float f) <i>Convert a float to fixed-point.</i>
INLINE u32	<b>fx2uint (FIXED fx)</b> <i>Convert a FIXED point value to an unsigned integer (orly?).</i>
INLINE u32	<b>fx2ufrac (FIXED fx)</b> <i>Get the unsigned fractional part of a fixed point value (orly?).</i>
INLINE int	<b>fx2int (FIXED fx)</b> <i>Convert a FIXED point value to an signed integer.</i>
INLINE float	<b>fx2float (FIXED fx)</b> <i>Convert a fixed point value to floating point.</i>
INLINE <b>FIXED</b>	<b>fxadd (FIXED fa, FIXED fb)</b> <i>Add two fixed point values.</i>
INLINE <b>FIXED</b>	<b>fxsub (FIXED fa, FIXED fb)</b> <i>Subtract two fixed point values.</i>
INLINE <b>FIXED</b>	<b>fxmul (FIXED fa, FIXED fb)</b> <i>Multiply two fixed point values.</i>
INLINE <b>FIXED</b>	<b>fxdiv (FIXED fa, FIXED fb)</b> <i>Divide two fixed point values.</i>
INLINE <b>FIXED</b>	<b>fxmul64 (FIXED fa, FIXED fb)</b> <i>Multiply two fixed point values using 64bit math.</i>
INLINE <b>FIXED</b>	<b>fxdiv64 (FIXED fa, FIXED fb)</b> <i>Divide two fixed point values using 64bit math.</i>
INLINE s32	<b>lu_sin</b> (uint theta) <i>Look-up a sine value (<math>2\pi = 0x10000</math>).</i>
INLINE s32	<b>lu_cos</b> (uint theta) <i>Look-up a cosine value (<math>2\pi = 0x10000</math>).</i>
INLINE uint	<b>lu_div</b> (uint x) <i>Look-up a division value between 0 and 255.</i>

INLINE int	<b>lu_lerp32</b> (const s32 lut[], uint x, const uint shift) <i>Linear interpolator for 32bit LUTs.</i>
INLINE int	<b>lu_lerp16</b> (const s16 lut[], uint x, const uint shift) <i>As lu_lerp32, but for 16bit LUTs.</i>
INLINE POINT *	<b>pt_set</b> (POINT *pd, int x, int y) <i>Initialize pd to (x, y).</i>
INLINE POINT *	<b>pt_add</b> (POINT *pd, const POINT *pa, const POINT *pb) <i>Point addition: pd = pa + pb.</i>
INLINE POINT *	<b>pt_sub</b> (POINT *pd, const POINT *pa, const POINT *pb) <i>Point subtraction: pd = pa - pb.</i>
INLINE POINT *	<b>pt_scale</b> (POINT *pd, const POINT *pa, int c) <i>Point scale: pd = c * pa.</i>
INLINE POINT *	<b>pt_add_eq</b> (POINT *pd, const POINT *pb) <i>Point increment: pd += pb.</i>
INLINE POINT *	<b>pt_sub_eq</b> (POINT *pd, const POINT *pb) <i>Point decrement: pd -= pb.</i>
INLINE POINT *	<b>pt_scale_eq</b> (POINT *pd, int c) <i>Point scale: pd *= c.</i>
INLINE int	<b>pt_cross</b> (const POINT *pa, const POINT *pb) <i>Point 'cross'-product: pa × pb.</i>
INLINE int	<b>pt_dot</b> (const POINT *pa, const POINT *pb) <i>Point 'dot'-product: pa · pb.</i>
int	<b>pt_in_rect</b> (const POINT *pt, const struct RECT *rc)
INLINE RECT *	<b>rc_set</b> (RECT *rc, int l, int t, int r, int b) <i>Initialize a rectangle.</i>
INLINE RECT *	<b>rc_set2</b> (RECT *rc, int x, int y, int w, int h) <i>Initialize a rectangle, with sizes inside of max boundaries.</i>
INLINE int	<b>rc_width</b> (const RECT *rc) <i>Get rectangle width.</i>
INLINE int	<b>rc_height</b> (const RECT *rc) <i>Get rectangle height.</i>
INLINE RECT *	<b>rc_set_pos</b> (RECT *rc, int x, int y) <i>Move rectangle to (x, y) position.</i>
INLINE RECT *	<b>rc_set_size</b> (RECT *rc, int w, int h)

	<i>Reside rectangle.</i>
INLINE RECT *	<b>rc_move</b> (RECT *rc, int dx, int dy) <i>Move rectangle by (dx, dy).</i>
INLINE RECT *	<b>rc_inflate</b> (RECT *rc, int dw, int dh) <i>Increase size by dw horizontally and dh vertically.</i>
INLINE RECT *	<b>rc_inflate2</b> (RECT *rc, const RECT *dr) <i>Increase sizes on all sides by values of rectangle dr.</i>
RECT *	<b>rc_normalize</b> (RECT *rc)
INLINE VECTOR *	<b>vec_set</b> (VECTOR *vd, FIXED x, FIXED y, FIXED z) <i>Initialize a vector.</i>
INLINE VECTOR *	<b>vec_add</b> (VECTOR *vd, const VECTOR *va, const VECTOR *vb) <i>Add vectors: <math>d = a + b;</math></i>
INLINE VECTOR *	<b>vec_sub</b> (VECTOR *vd, const VECTOR *va, const VECTOR *vb) <i>Subtract vectors: <math>d = a - b;</math></i>
INLINE VECTOR *	<b>vec_mul</b> (VECTOR *vd, const VECTOR *va, const VECTOR *vb) <i>Multiply vectors elements: <math>d = S(ax, ay, az) \diamond b.</math></i>
INLINE VECTOR *	<b>vec_scale</b> (VECTOR *vd, const VECTOR *va, FIXED c) <i>Scale vector: <math>d = c*a.</math></i>
INLINE FIXED	<b>vec_dot</b> (const VECTOR *va, const VECTOR *vb) <i>Dot-product: <math>d = a \diamond b.</math></i>
INLINE VECTOR *	<b>vec_add_eq</b> (VECTOR *vd, const VECTOR *vb) <i>Increment vector: <math>d += b;</math></i>
INLINE VECTOR *	<b>vec_sub_eq</b> (VECTOR *vd, const VECTOR *vb) <i>Decrease vector: <math>d -= b;</math></i>
INLINE VECTOR *	<b>vec_mul_eq</b> (VECTOR *vd, const VECTOR *vb) <i>Multiply vectors elements: <math>d = S(dx, dy, dz) \diamond b.</math></i>
INLINE VECTOR *	<b>vec_scale_eq</b> (VECTOR *vd, FIXED c) <i>Scale vector: <math>d = c*d.</math></i>
VECTOR *	<b>vec_cross</b> (VECTOR *vd, const VECTOR *va, const VECTOR *vb)

## Variables

s32	<b>div_lut</b> [257]
s16	<b>sin_lut</b> [514]

---

# Detailed Description

**Author:**

J Vijn

**Date:**

20060508 - 20060908

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tonc\_memdef.h File Reference**

## Defines

#define	<b>DCNT_MODE0</b> 0
	<i>Mode 0; bg 0-4: reg.</i>
#define	<b>DCNT_MODE1</b> 0x0001
	<i>Mode 1; bg 0-1: reg; bg 2: affine.</i>
#define	<b>DCNT_MODE2</b> 0x0002
	<i>Mode 2; bg 2-3: affine.</i>
#define	<b>DCNT_MODE3</b> 0x0003
	<i>Mode 3; bg2: 240x160@16 bitmap.</i>
#define	<b>DCNT_MODE4</b> 0x0004
	<i>Mode 4; bg2: 240x160@8 bitmap.</i>
#define	<b>DCNT_MODE5</b> 0x0005
	<i>Mode 5; bg2: 160x128@16 bitmap.</i>
#define	<b>DCNT_GB</b> 0x0008
	<i>(R) GBC indicator</i>
#define	<b>DCNT_PAGE</b> 0x0010
	<i>Page indicator.</i>
#define	<b>DCNT_OAM_HBL</b> 0x0020
	<i>Allow OAM updates in HBlank.</i>
#define	<b>DCNT_OBJ_2D</b> 0
	<i>OBJ-VRAM as matrix.</i>
#define	<b>DCNT_OBJ_1D</b> 0x0040
	<i>OBJ-VRAM as array.</i>
#define	<b>DCNT_BLANK</b> 0x0080
	<i>Force screen blank.</i>
#define	<b>DCNT_BG0</b> 0x0100
	<i>Enable bg 0.</i>
#define	<b>DCNT_BG1</b> 0x0200
	<i>Enable bg 1.</i>
#define	<b>DCNT_BG2</b> 0x0400
	<i>Enable bg 2.</i>
#define	<b>DCNT_BG3</b> 0x0800

	<i>Enable bg 3.</i>
#define	<b>DCNT_OBJ</b> 0x1000 <i>Enable objects.</i>
#define	<b>DCNT_WIN0</b> 0x2000 <i>Enable window 0.</i>
#define	<b>DCNT_WIN1</b> 0x4000 <i>Enable window 1.</i>
#define	<b>DCNT_WINOBJ</b> 0x8000 <i>Enable object window.</i>
#define	<b>DCNT_MODE_MASK</b> 0x0007
#define	<b>DCNT_MODE_SHIFT</b> 0
#define	<b>DCNT_MODE(n)</b> ((n)<<DCNT_MODE_SHIFT)
#define	<b>DCNT_LAYER_MASK</b> 0x1F00
#define	<b>DCNT_LAYER_SHIFT</b> 8
#define	<b>DCNT_LAYER(n)</b> ((n)<<DCNT_LAYER_SHIFT)
#define	<b>DCNT_WIN_MASK</b> 0xE000
#define	<b>DCNT_WIN_SHIFT</b> 13
#define	<b>DCNT_WIN(n)</b> ((n)<<DCNT_WIN_SHIFT)
#define	<b>DCNT_BUILD(mode, layer, win, obj1d, objhbl)</b>
#define	<b>DSTAT_IN_VBL</b> 0x0001 <i>Now in VBlank.</i>
#define	<b>DSTAT_IN_HBL</b> 0x0002 <i>Now in HBlank.</i>
#define	<b>DSTAT_IN_VCT</b> 0x0004 <i>Now in set VCount.</i>
#define	<b>DSTAT_VBL_IRQ</b> 0x0008 <i>Enable VBlank irq.</i>
#define	<b>DSTAT_HBL_IRQ</b> 0x0010 <i>Enable HBlank irq.</i>
#define	<b>DSTAT_VCT_IRQ</b> 0x0020 <i>Enable VCount irq.</i>
#define	<b>DSTAT_VCT_MASK</b> 0xFF00
#define	<b>DSTAT_VCT_SHIFT</b> 8
#define	<b>DSTAT_VCT(n)</b> ((n)<<DSTAT_VCT_SHIFT)
#define	<b>BG_MOSAIC</b> 0x0040

	<i>Enable Mosaic.</i>
#define	<b>BG_4BPP</b> 0 <i>4bpp (16 color) bg (no effect on affine bg)</i>
#define	<b>BG_8BPP</b> 0x0080 <i>8bpp (256 color) bg (no effect on affine bg)</i>
#define	<b>BG_WRAP</b> 0x2000 <i>Wrap around edges of affine bgs.</i>
#define	<b>BG_SIZE0</b> 0
#define	<b>BG_SIZE1</b> 0x4000
#define	<b>BG_SIZE2</b> 0x8000
#define	<b>BG_SIZE3</b> 0xC000
#define	<b>BG_REG_32x32</b> 0 <i>reg bg, 32x32 (256x256 px)</i>
#define	<b>BG_REG_64x32</b> 0x4000 <i>reg bg, 64x32 (512x256 px)</i>
#define	<b>BG_REG_32x64</b> 0x8000 <i>reg bg, 32x64 (256x512 px)</i>
#define	<b>BG_REG_64x64</b> 0xC000 <i>reg bg, 64x64 (512x512 px)</i>
#define	<b>BG_AFF_16x16</b> 0 <i>affine bg, 16x16 (128x128 px)</i>
#define	<b>BG_AFF_32x32</b> 0x4000 <i>affine bg, 32x32 (256x256 px)</i>
#define	<b>BG_AFF_64x64</b> 0x8000 <i>affine bg, 64x64 (512x512 px)</i>
#define	<b>BG_AFF_128x128</b> 0xC000 <i>affine bg, 128x128 (1024x1024 px)</i>
#define	<b>BG_PRIO_MASK</b> 0x0003
#define	<b>BG_PRIO_SHIFT</b> 0
#define	<b>BG_PRIO(n)</b> ((n)<<BG_PRIO_SHIFT)
#define	<b>BG_CBB_MASK</b> 0x000C
#define	<b>BG_CBB_SHIFT</b> 2
#define	<b>BG_CBB(n)</b> ((n)<<BG_CBB_SHIFT)
#define	<b>BG_SBB_MASK</b> 0x1F00

<code>#define</code>	<b>BG_SBB_SHIFT</b> 8
<code>#define</code>	<b>BG_SBB(n)</b> ((n)<<BG_SBB_SHIFT)
<code>#define</code>	<b>BG_SIZE_MASK</b> 0xC000
<code>#define</code>	<b>BG_SIZE_SHIFT</b> 14
<code>#define</code>	<b>BG_SIZE(n)</b> ((n)<<BG_SIZE_SHIFT)
<code>#define</code>	<b>BG_BUILD</b> (ccb, sbb, size, bpp, prio, mos, wrap)
<code>#define</code>	<b>BLD_BG0</b> 0x0001 \ <i>name Blend control</i>
<code>#define</code>	<b>BLD_BG1</b> 0x0002 <i>Blend bg 1.</i>
<code>#define</code>	<b>BLD_BG2</b> 0x0004 <i>Blend bg 2.</i>
<code>#define</code>	<b>BLD_BG3</b> 0x0008 <i>Blend bg 3.</i>
<code>#define</code>	<b>BLD_OBJ</b> 0x0010 <i>Blend objects.</i>
<code>#define</code>	<b>BLD_ALL</b> 0x001F <i>All layers (except backdrop).</i>
<code>#define</code>	<b>BLD_BACKDROP</b> 0x0020 <i>Blend backdrop.</i>
<code>#define</code>	<b>BLD_OFF</b> 0 <i>Blend mode is off.</i>
<code>#define</code>	<b>BLD_STD</b> 0x0040 <i>Normal alpha blend (with REG_EV).</i>
<code>#define</code>	<b>BLD_WHITE</b> 0x0080 <i>Fade to white (with REG_Y).</i>
<code>#define</code>	<b>BLD_BLACK</b> 0x00C0 <i>Fade to black (with REG_Y).</i>
<code>#define</code>	<b>BLD_TOP_MASK</b> 0x003F
<code>#define</code>	<b>BLD_TOP_SHIFT</b> 0
<code>#define</code>	<b>BLD_TOP(n)</b> ((n)<<BLD_TOP_SHIFT)
<code>#define</code>	<b>BLD_MODE_MASK</b> 0x00C0
<code>#define</code>	<b>BLD_MODE_SHIFT</b> 6
<code>#define</code>	<b>BLD_MODE(n)</b> ((n)<<BLD_MODE_SHIFT)

#define	<b>BLD_BOT_MASK</b>	0x3F00
#define	<b>BLD_BOT_SHIFT</b>	8
#define	<b>BLD_BOT(n)</b>	((n)<<BLD_BOT_SHIFT)
#define	<b>BLD_BUILD(top, bot, mode)</b>	((((bot)&63)<<8)   (((mode)&3)<<6)   ((top)&63))
#define	<b>SSW_INC</b>	0
	<i>Increasing sweep rate.</i>	
#define	<b>SSW_DEC</b>	0x0008
	<i>Decreasing sweep rate.</i>	
#define	<b>SSW_OFF</b>	0x0008
	<i>Disable sweep altogether.</i>	
#define	<b>SSW_SHIFT_MASK</b>	0x0007
#define	<b>SSW_SHIFT_SHIFT</b>	0
#define	<b>SSW_SHIFT(n)</b>	((n)<<SSW_SHIFT_SHIFT)
#define	<b>SSW_TIME_MASK</b>	0x0070
#define	<b>SSW_TIME_SHIFT</b>	4
#define	<b>SSW_TIME(n)</b>	((n)<<SSW_TIME_SHIFT)
#define	<b>SSW_BUILD(shift, dir, time)</b>	((((time)&7)<<4)   ((dir)<<3)   ((shift)&7))
#define	<b>SSQR_DUTY1_8</b>	0
	<i>12.5% duty cycle (#-----)</i>	
#define	<b>SSQR_DUTY1_4</b>	0x0040
	<i>25% duty cycle (##-----)</i>	
#define	<b>SSQR_DUTY1_2</b>	0x0080
	<i>50% duty cycle (#####---)</i>	
#define	<b>SSQR_DUTY3_4</b>	0x00C0
	<i>75% duty cycle (#####---) Equivalent to 25%</i>	
#define	<b>SSQR_INC</b>	0
	<i>Increasing volume.</i>	
#define	<b>SSQR_DEC</b>	0x0800
	<i>Decreasing volume.</i>	
#define	<b>SSQR_LEN_MASK</b>	0x003F
#define	<b>SSQR_LEN_SHIFT</b>	0
#define	<b>SSQR_LEN(n)</b>	((n)<<SSQR_LEN_SHIFT)
#define	<b>SSQR_DUTY_MASK</b>	0x00C0

#define	<b>SSQR_DUTY_SHIFT</b> 6
#define	<b>SSQR_DUTY(n)</b> ((n)<<SSQR_DUTY_SHIFT)
#define	<b>SSQR_TIME_MASK</b> 0x0700
#define	<b>SSQR_TIME_SHIFT</b> 8
#define	<b>SSQR_TIME(n)</b> ((n)<<SSQR_TIME_SHIFT)
#define	<b>SSQR_IVOL_MASK</b> 0xF000
#define	<b>SSQR_IVOL_SHIFT</b> 12
#define	<b>SSQR_IVOL(n)</b> ((n)<<SSQR_IVOL_SHIFT)
#define	<b>SSQR_ENV_BUILD(ivol, dir, time)</b> ( ((ivol)<<12)   ((dir)<<11)   (((time)&7)<<8) )
#define	<b>SSQR_BUILD(_ivol, dir, step, duty, len)</b> ( SSQR_ENV_BUILD(ivol,dir,step)   (((duty)&3)<<6)   ((len)&63) )
#define	<b>SFREQ_HOLD</b> 0 <i>Continuous play.</i>
#define	<b>SFREQ_TIMED</b> 0x4000 <i>Timed play.</i>
#define	<b>SFREQ_RESET</b> 0x8000 <i>Reset sound.</i>
#define	<b>SFREQ_RATE_MASK</b> 0x07FF
#define	<b>SFREQ_RATE_SHIFT</b> 0
#define	<b>SFREQ_RATE(n)</b> ((n)<<SFREQ_RATE_SHIFT)
#define	<b>SFREQ_BUILD(rate, timed, reset)</b> ( ((rate)&0x7FF)   ((timed)<<14)   ((reset)<<15) )
#define	<b>SDMG_LSQR1</b> 0x0100 <i>Enable channel 1 on left.</i>
#define	<b>SDMG_LSQR2</b> 0x0200 <i>Enable channel 2 on left.</i>
#define	<b>SDMG_LWAVE</b> 0x0400 <i>Enable channel 3 on left.</i>
#define	<b>SDMG_LNOISE</b> 0x0800 <i>Enable channel 4 on left.</i>
#define	<b>SDMG_RSQR1</b> 0x1000 <i>Enable channel 1 on right.</i>
#define	<b>SDMG_RSQR2</b> 0x2000 <i>Enable channel 2 on right.</i>

#define	<b>SDMG_RWAVE</b> 0x4000 <i>Enable channel 3 on right.</i>
#define	<b>SDMG_RNOISE</b> 0x8000 <i>Enable channel 4 on right.</i>
#define	<b>SDMG_LVOL_MASK</b> 0x0007
#define	<b>SDMG_LVOL_SHIFT</b> 0
#define	<b>SDMG_LVOL(n)</b> ((n)<<SDMG_LVOL_SHIFT)
#define	<b>SDMG_RVOL_MASK</b> 0x0070
#define	<b>SDMG_RVOL_SHIFT</b> 4
#define	<b>SDMG_RVOL(n)</b> ((n)<<SDMG_RVOL_SHIFT)
#define	<b>SDMG_SQR1</b> 0x01
#define	<b>SDMG_SQR2</b> 0x02
#define	<b>SDMG_WAVE</b> 0x04
#define	<b>SDMG_NOISE</b> 0x08
#define	<b>SDMG_BUILD(_lmode, _rmode, _lvol, _rvol)</b> ( (_rmode)<<12)   (_lmode)<<8)   (((_rvol)&7)<<4)   (((_lvol)&7) )
#define	<b>SDMG_BUILD_LR(_mode, _vol)</b> SDMG_BUILD(_mode, _mode, _vol, _vol)
#define	<b>SDS_DMG25</b> 0 <i>Tone generators at 25% volume.</i>
#define	<b>SDS_DMG50</b> 0x0001 <i>Tone generators at 50% volume.</i>
#define	<b>SDS_DMG100</b> 0x0002 <i>Tone generators at 100% volume.</i>
#define	<b>SDS_A50</b> 0 <i>Direct Sound A at 50% volume.</i>
#define	<b>SDS_A100</b> 0x0004 <i>Direct Sound A at 100% volume.</i>
#define	<b>SDS_B50</b> 0 <i>Direct Sound B at 50% volume.</i>
#define	<b>SDS_B100</b> 0x0008 <i>Direct Sound B at 100% volume.</i>
#define	<b>SDS_AR</b> 0x0100 <i>Enable Direct Sound A on right.</i>
#define	<b>SDS_AL</b> 0x0200

	<i>Enable Direct Sound A on left.</i>
#define	<b>SDS_ATMR0</b> 0 <i>Direct Sound A to use timer 0.</i>
#define	<b>SDS_ATMR1</b> 0x0400 <i>Direct Sound A to use timer 1.</i>
#define	<b>SDS_ARESET</b> 0x0800 <i>Reset FIFO of Direct Sound A.</i>
#define	<b>SDS_BR</b> 0x1000 <i>Enable Direct Sound B on right.</i>
#define	<b>SDS_BL</b> 0x2000 <i>Enable Direct Sound B on left.</i>
#define	<b>SDS_BTMR0</b> 0 <i>Direct Sound B to use timer 0.</i>
#define	<b>SDS_BTMR1</b> 0x4000 <i>Direct Sound B to use timer 1.</i>
#define	<b>SDS_BRESET</b> 0x8000 <i>Reset FIFO of Direct Sound B.</i>
#define	<b>SSTAT_SQR1</b> 0x0001 <i>(R) Channel 1 status</i>
#define	<b>SSTAT_SQR2</b> 0x0002 <i>(R) Channel 2 status</i>
#define	<b>SSTAT_WAVE</b> 0x0004 <i>(R) Channel 3 status</i>
#define	<b>SSTAT_NOISE</b> 0x0008 <i>(R) Channel 4 status</i>
#define	<b>SSTAT_DISABLE</b> 0 <i>Disable sound.</i>
#define	<b>SSTAT_ENABLE</b> 0x0080 <i>Enable sound. NOTE: enable before using any other sound regs.</i>
#define	<b>DMA_DST_INC</b> 0 <i>Incrementing destination address.</i>
#define	<b>DMA_DST_DEC</b> 0x00200000 <i>Decrementing destination.</i>
#define	<b>DMA_DST_FIXED</b> 0x00400000

	<i>Fixed destination.</i>
#define	<b>DMA_DST_RELOAD</b> 0x00600000 <i>Increment destination, reset after full run.</i>
#define	<b>DMA_SRC_INC</b> 0 <i>Incrementing source address.</i>
#define	<b>DMA_SRC_DEC</b> 0x00800000 <i>Decrementing source address.</i>
#define	<b>DMA_SRC_FIXED</b> 0x01000000 <i>Fixed source address.</i>
#define	<b>DMA_REPEAT</b> 0x02000000 <i>Repeat transfer at next start condition.</i>
#define	<b>DMA_16</b> 0 <i>Transfer by halfword.</i>
#define	<b>DMA_32</b> 0x04000000 <i>Transfer by word.</i>
#define	<b>DMA_AT_NOW</b> 0 <i>Start transfer now.</i>
#define	<b>DMA_GAMEPAK</b> 0x08000000 <i>Gamepak DRQ.</i>
#define	<b>DMA_AT_VBLANK</b> 0x10000000 <i>Start transfer at VBlank.</i>
#define	<b>DMA_AT_HBLANK</b> 0x20000000 <i>Start transfer at HBlank.</i>
#define	<b>DMA_AT_SPECIAL</b> 0x30000000 <i>Start copy at 'special' condition. Channel dependent.</i>
#define	<b>DMA_AT_FIFO</b> 0x30000000 <i>Start at FIFO empty (DMA0/DMA1).</i>
#define	<b>DMA_AT_REFRESH</b> 0x30000000 <i>VRAM special; start at VCount=2 (DMA3).</i>
#define	<b>DMA_IRQ</b> 0x40000000 <i>Enable DMA irq.</i>
#define	<b>DMA_ENABLE</b> 0x80000000 <i>Enable DMA.</i>
#define	<b>DMA_COUNT_MASK</b> 0x0000FFFF

#define	<b>DMA_COUNT_SHIFT</b> 0
#define	<b>DMA_COUNT(n)</b> ((n)<<DMA_COUNT_SHIFT)
#define	<b>DMA_NOW</b> (DMA_ENABLE   DMA_AT_NOW)
#define	<b>DMA_16NOW</b> (DMA_NOW   DMA_16)
#define	<b>DMA_32NOW</b> (DMA_NOW   DMA_32)
#define	<b>DMA_CPY16</b> (DMA_NOW   DMA_16)
#define	<b>DMA_CPY32</b> (DMA_NOW   DMA_32)
#define	<b>DMA_FILL16</b> (DMA_NOW   DMA_SRC_FIXED   DMA_16)
#define	<b>DMA_FILL32</b> (DMA_NOW   DMA_SRC_FIXED   DMA_32)
#define	<b>DMA_HDMA</b> (DMA_ENABLE   DMA_REPEAT   DMA_AT_HBLANK   DMA_DST_RELOAD)
#define	<b>TM_FREQ_SYS</b> 0 <i>System clock timer (16.7 Mhz).</i>
#define	<b>TM_FREQ_1</b> 0 <i>1 cycle/tick (16.7 Mhz)</i>
#define	<b>TM_FREQ_64</b> 0x0001 <i>64 cycles/tick (262 kHz)</i>
#define	<b>TM_FREQ_256</b> 0x0002 <i>256 cycles/tick (66 kHz)</i>
#define	<b>TM_FREQ_1024</b> 0x0003 <i>1024 cycles/tick (16 kHz)</i>
#define	<b>TM.Cascade</b> 0x0004 <i>Increment when preceding timer overflows.</i>
#define	<b>TM_IRQ</b> 0x0040 <i>Enable timer irq.</i>
#define	<b>TM_ENABLE</b> 0x0080 <i>Enable timer.</i>
#define	<b>TM_FREQ_MASK</b> 0x0003
#define	<b>TM_FREQ_SHIFT</b> 0
#define	<b>TM_FREQ(n)</b> ((n)<<TM_FREQ_SHIFT)
#define	<b>KEY_A</b> 0x0001 <i>Button A.</i>
#define	<b>KEY_B</b> 0x0002 <i>Button B.</i>
#define	<b>KEY_SELECT</b> 0x0004

	<i>Select button.</i>
#define	<b>KEY_START</b> 0x0008 <i>Start button.</i>
#define	<b>KEY_RIGHT</b> 0x0010 <i>Right D-pad.</i>
#define	<b>KEY_LEFT</b> 0x0020 <i>Left D-pad.</i>
#define	<b>KEY_UP</b> 0x0040 <i>Up D-pad.</i>
#define	<b>KEY_DOWN</b> 0x0080 <i>Down D-pad.</i>
#define	<b>KEY_R</b> 0x0100 <i>Shoulder R.</i>
#define	<b>KEY_L</b> 0x0200 <i>Shoulder L.</i>
#define	<b>KEY_ACCEPT</b> 0x0009 <i>Accept buttons: A or start.</i>
#define	<b>KEY_CANCEL</b> 0x0002 <i>Cancel button: B (well, it usually is).</i>
#define	<b>KEY_RESET</b> 0x030C <i>St+Se+L+R.</i>
#define	<b>KEY_FIRE</b> 0x0003 <i>Fire buttons: A or B.</i>
#define	<b>KEY_SPECIAL</b> 0x000C <i>Special buttons: Select or Start.</i>
#define	<b>KEY_DIR</b> 0x00F0 <i>Directions: left, right, up down.</i>
#define	<b>KEY_SHOULDER</b> 0x0300 <i>L or R.</i>
#define	<b>KEY_ANY</b> 0x03FF <i>Here's the Any key :).</i>
#define	<b>KEY_MASK</b> 0x03FF
#define	<b>KCNT_IRQ</b> 0x4000 <i>Enable key irq.</i>

#define	<b>KCNT_OR</b> 0	<i>Interrupt on any of selected keys.</i>
#define	<b>KCNT_AND</b> 0x8000	<i>Interrupt on all of selected keys.</i>
#define	<b>IRQ_VBLANK</b> 0x0001	<i>Catch VBlank irq.</i>
#define	<b>IRQ_HBLANK</b> 0x0002	<i>Catch HBlank irq.</i>
#define	<b>IRQ_VCOUNTER</b> 0x0004	<i>Catch VCount irq.</i>
#define	<b>IRQ_TIMER0</b> 0x0008	<i>Catch timer 0 irq.</i>
#define	<b>IRQ_TIMER1</b> 0x0010	<i>Catch timer 1 irq.</i>
#define	<b>IRQ_TIMER2</b> 0x0020	<i>Catch timer 2 irq.</i>
#define	<b>IRQ_TIMER3</b> 0x0040	<i>Catch timer 3 irq.</i>
#define	<b>IRQ_SERIAL</b> 0x0080	<i>Catch serial comm irq.</i>
#define	<b>IRQ_DMA0</b> 0x0100	<i>Catch DMA 0 irq.</i>
#define	<b>IRQ_DMA1</b> 0x0200	<i>Catch DMA 1 irq.</i>
#define	<b>IRQ_DMA2</b> 0x0400	<i>Catch DMA 2 irq.</i>
#define	<b>IRQ_DMA3</b> 0x0800	<i>Catch DMA 3 irq.</i>
#define	<b>IRQ_KEYPAD</b> 0x1000	<i>Catch key irq.</i>
#define	<b>IRQ_GAMEPAK</b> 0x2000	<i>Catch cart irq.</i>
#define	<b>WS_SRAM_4</b> 0	
#define	<b>WS_SRAM_3</b> 0x0001	

#define	<b>WS_SRAM_2</b>	0x0002
#define	<b>WS_SRAM_8</b>	0x0003
#define	<b>WS_ROM0_N4</b>	0
#define	<b>WS_ROM0_N3</b>	0x0004
#define	<b>WS_ROM0_N2</b>	0x0008
#define	<b>WS_ROM0_N8</b>	0x000C
#define	<b>WS_ROM0_S2</b>	0
#define	<b>WS_ROM0_S1</b>	0x0010
#define	<b>WS_ROM1_N4</b>	0
#define	<b>WS_ROM1_N3</b>	0x0020
#define	<b>WS_ROM1_N2</b>	0x0040
#define	<b>WS_ROM1_N8</b>	0x0060
#define	<b>WS_ROM1_S4</b>	0
#define	<b>WS_ROM1_S1</b>	0x0080
#define	<b>WS_ROM2_N4</b>	0
#define	<b>WS_ROM2_N3</b>	0x0100
#define	<b>WS_ROM2_N2</b>	0x0200
#define	<b>WS_ROM2_N8</b>	0x0300
#define	<b>WS_ROM2_S8</b>	0
#define	<b>WS_ROM2_S1</b>	0x0400
#define	<b>WS_PHI_OFF</b>	0
#define	<b>WS_PHI_4</b>	0x0800
#define	<b>WS_PHI_2</b>	0x1000
#define	<b>WS_PHI_1</b>	0x1800
#define	<b>WS_PREFETCH</b>	0x4000
#define	<b>WS_GBA</b>	0
#define	<b>WS_CGB</b>	0x8000
#define	<b>WS_STANDARD</b>	0x4317
#define	<b>SE_HFLIP</b>	0x0400 <i>Horizontal flip.</i>
#define	<b>SE_VFLIP</b>	0x0800 <i>Vertical flip.</i>
#define	<b>SE_ID_MASK</b>	0x03FF
#define	<b>SE_ID_SHIFT</b>	0
#define	<b>SE_ID(n)</b>	((n)<<SE_ID_SHIFT)

<code>#define SE_FLIP_MASK 0x0C00</code>	
<code>#define SE_FLIP_SHIFT 10</code>	
<code>#define SE_FLIP(n) ((n)&lt;&lt;SE_FLIP_SHIFT)</code>	
<code>#define SE_PALBANK_MASK 0xF000</code>	
<code>#define SE_PALBANK_SHIFT 12</code>	
<code>#define SE_PALBANK(n) ((n)&lt;&lt;SE_PALBANK_SHIFT)</code>	
<code>#define SE_BUILD(id, PALBANK, hflip, vflip) ( ((id)&amp;0x03FF)   (((hflip)&amp;1) &lt;&lt;10)   (((vflip)&amp;1)&lt;&lt;11)   ((PALBANK)&lt;&lt;12) )</code>	
<code>#define ATTR0_REG 0</code>	
	<i>Regular object.</i>
<code>#define ATTR0_AFF 0x0100</code>	
	<i>Affine object.</i>
<code>#define ATTR0_HIDE 0x0200</code>	
	<i>Inactive object.</i>
<code>#define ATTR0_AFF_DBL 0x0300</code>	
	<i>Double-size affine object.</i>
<code>#define ATTR0_AFF_DBL_BIT 0x0200</code>	
<code>#define ATTR0_BLEND 0x0400</code>	
	<i>Enable blend.</i>
<code>#define ATTR0_WINDOW 0x0800</code>	
	<i>Use for object window.</i>
<code>#define ATTR0_MOSAIC 0x1000</code>	
	<i>Enable mosaic.</i>
<code>#define ATTR0_4BPP 0</code>	
	<i>Use 4bpp (16 color) tiles.</i>
<code>#define ATTR0_8BPP 0x2000</code>	
	<i>Use 8bpp (256 color) tiles.</i>
<code>#define ATTR0_SQUARE 0</code>	
	<i>Square shape.</i>
<code>#define ATTR0_WIDE 0x4000</code>	
	<i>Tall shape (<math>height &gt; width</math>).</i>
<code>#define ATTR0_TALL 0x8000</code>	
	<i>Wide shape (<math>height &lt; width</math>).</i>
<code>#define ATTR0_Y_MASK 0x00FF</code>	

#define	<b>ATTR0_Y_SHIFT</b> 0	
#define	<b>ATTR0_Y(n)</b> ((n)<<ATTR0_Y_SHIFT)	
#define	<b>ATTR0_MODE_MASK</b> 0x0300	
#define	<b>ATTR0_MODE_SHIFT</b> 8	
#define	<b>ATTR0_MODE(n)</b> ((n)<<ATTR0_MODE_SHIFT)	
#define	<b>ATTR0_SHAPE_MASK</b> 0xC000	
#define	<b>ATTR0_SHAPE_SHIFT</b> 14	
#define	<b>ATTR0_SHAPE(n)</b> ((n)<<ATTR0_SHAPE_SHIFT)	
#define	<b>ATTR0_BUILD(y, shape, bpp, mode, mos, bld, win)</b>	
#define	<b>ATTR1_HFLIP</b> 0x1000	
		<i>Horizontal flip (reg obj only).</i>
#define	<b>ATTR1_VFLIP</b> 0x2000	
		<i>Vertical flip (reg obj only).</i>
#define	<b>ATTR1_SIZE_8</b> 0	
#define	<b>ATTR1_SIZE_16</b> 0x4000	
#define	<b>ATTR1_SIZE_32</b> 0x8000	
#define	<b>ATTR1_SIZE_64</b> 0xC000	
#define	<b>ATTR1_SIZE_8x8</b> 0	
		<i>Size flag for 8x8 px object.</i>
#define	<b>ATTR1_SIZE_16x16</b> 0x4000	
		<i>Size flag for 16x16 px object.</i>
#define	<b>ATTR1_SIZE_32x32</b> 0x8000	
		<i>Size flag for 32x32 px object.</i>
#define	<b>ATTR1_SIZE_64x64</b> 0xC000	
		<i>Size flag for 64x64 px object.</i>
#define	<b>ATTR1_SIZE_8x16</b> 0	
		<i>Size flag for 8x16 px object.</i>
#define	<b>ATTR1_SIZE_8x32</b> 0x4000	
		<i>Size flag for 8x32 px object.</i>
#define	<b>ATTR1_SIZE_16x32</b> 0x8000	
		<i>Size flag for 16x32 px object.</i>
#define	<b>ATTR1_SIZE_32x64</b> 0xC000	
		<i>Size flag for 32x64 px object.</i>
#define	<b>ATTR1_SIZE_16x8</b> 0	

	<i>Size flag for 16x8 px object.</i>
#define	<b>ATTR1_SIZE_32x8</b> 0x4000 <i>Size flag for 32x8 px object.</i>
#define	<b>ATTR1_SIZE_32x16</b> 0x8000 <i>Size flag for 32x16 px object.</i>
#define	<b>ATTR1_SIZE_64x32</b> 0xC000 <i>Size flag for 64x64 px object.</i>
#define	<b>ATTR1_X_MASK</b> 0x01FF
#define	<b>ATTR1_X_SHIFT</b> 0
#define	<b>ATTR1_X(n)</b> ((n)<<ATTR1_X_SHIFT)
#define	<b>ATTR1_AFF_ID_MASK</b> 0x3E00
#define	<b>ATTR1_AFF_ID_SHIFT</b> 9
#define	<b>ATTR1_AFF_ID(n)</b> ((n)<<ATTR1_AFF_ID_SHIFT)
#define	<b>ATTR1_FLIP_MASK</b> 0x3000
#define	<b>ATTR1_FLIP_SHIFT</b> 12
#define	<b>ATTR1_FLIP(n)</b> ((n)<<ATTR1_FLIP_SHIFT)
#define	<b>ATTR1_SIZE_MASK</b> 0xC000
#define	<b>ATTR1_SIZE_SHIFT</b> 14
#define	<b>ATTR1_SIZE(n)</b> ((n)<<ATTR1_SIZE_SHIFT)
#define	<b>ATTR1_BUILDR(x, size, hflip, vflip)</b> ( ((x)&511)   (((hflip)&1)<<12)   (((vflip)&1)<<13)   (((size)&3)<<14) )
#define	<b>ATTR1_BUILDA(x, size, affid)</b> ( ((x)&511)   (((affid)&31)<<9)   (((size)&3)<<14) )
#define	<b>ATTR2_ID_MASK</b> 0x03FF
#define	<b>ATTR2_ID_SHIFT</b> 0
#define	<b>ATTR2_ID(n)</b> ((n)<<ATTR2_ID_SHIFT)
#define	<b>ATTR2_PRIO_MASK</b> 0x0C00
#define	<b>ATTR2_PRIO_SHIFT</b> 10
#define	<b>ATTR2_PRIO(n)</b> ((n)<<ATTR2_PRIO_SHIFT)
#define	<b>ATTR2_PALBANK_MASK</b> 0xF000
#define	<b>ATTR2_PALBANK_SHIFT</b> 12
#define	<b>ATTR2_PALBANK(n)</b> ((n)<<ATTR2_PALBANK_SHIFT)
#define	<b>ATTR2_BUILD(id, pb, prio)</b> ( ((id)&0x3FF)   (((pb)&15)<<12)   (((prio)&3)<<10) )

## Window macros

#define	<b>WIN_BG0</b> 0x0001 <i>Windowed bg 0.</i>
#define	<b>WIN_BG1</b> 0x0002 <i>Windowed bg 1.</i>
#define	<b>WIN_BG2</b> 0x0004 <i>Windowed bg 2.</i>
#define	<b>WIN_BG3</b> 0x0008 <i>Windowed bg 3.</i>
#define	<b>WIN_OBJ</b> 0x0010 <i>Windowed objects.</i>
#define	<b>WIN_ALL</b> 0x001F <i>All layers in window.</i>
#define	<b>WIN_BLD</b> 0x0020 <i>Windowed blending.</i>
#define	<b>WIN_LAYER_MASK</b> 0x003F <i>Windowed bg 0.</i>
#define	<b>WIN_LAYER_SHIFT</b> 0 <i>Windowed bg 0.</i>
#define	<b>WIN_LAYER(n)</b> ((n)<<WIN_LAYER_SHIFT) <i>Windowed bg 0.</i>
#define	<b>WIN_BUILD</b> (low, high) ( ((high)<<8)   (low) ) <i>Windowed bg 0.</i>
#define	<b>WININ_BUILD</b> (win0, win1) WIN_BUILD(win0, win1) <i>Windowed bg 0.</i>
#define	<b>WINOUT_BUILD</b> (out, obj) WIN_BUILD(out, obj) <i>Windowed bg 0.</i>

## Mosaic macros

#define	<b>MOS_BH_MASK</b> 0x000F
#define	<b>MOS_BH_SHIFT</b> 0
#define	<b>MOS_BH(n)</b> ((n)<<MOS_BH_SHIFT)
#define	<b>MOS_BV_MASK</b> 0x00F0
#define	<b>MOS_BV_SHIFT</b> 4
#define	<b>MOS_BV(n)</b> ((n)<<MOS_BV_SHIFT)

#define	<b>MOS_OH_MASK</b>	0x0F00
#define	<b>MOS_OH_SHIFT</b>	8
#define	<b>MOS_OH(n)</b>	((n)<<MOS_OH_SHIFT)
#define	<b>MOS_OV_MASK</b>	0xF000
#define	<b>MOS_OV_SHIFT</b>	12
#define	<b>MOS_OV(n)</b>	((n)<<MOS_OV_SHIFT)
#define	<b>MOS_BUILD(bh, bv, oh, ov)</b>	((((ov)&15)<<12)   (((oh)&15)<<8)   (((bv)&15)<<4)   ((bh)&15))

## Blend weights

#define	<b>BLD_EVA_MASK</b>	0x001F
#define	<b>BLD_EVA_SHIFT</b>	0
#define	<b>BLD_EVA(n)</b>	((n)<<BLD_EVA_SHIFT)
#define	<b>BLD_EVB_MASK</b>	0x1F00
#define	<b>BLD_EVB_SHIFT</b>	8
#define	<b>BLD_EVB(n)</b>	((n)<<BLD_EVB_SHIFT)
#define	<b>BLDA_BUILD(eva, evb)</b>	((eva)&31)   (((evb)&31)<<8))

## Fade levels

#define	<b>BLDY_MASK</b>	0x001F
#define	<b>BLDY_SHIFT</b>	0
#define	<b>BLDY(n)</b>	((n)<<BLD_EY_SHIFT)
#define	<b>BLDY_BUILD(ey)</b>	((ey)&31))

## General SIO bits.

#define	<b>SIO_MODE_8BIT</b>	0x0000 <i>Normal comm mode, 8-bit.</i>
#define	<b>SIO_MODE_32BIT</b>	0x1000 <i>Normal comm mode, 32-bit.</i>
#define	<b>SIO_MODE_MULTI</b>	0x2000 <i>Multi-play comm mode.</i>
#define	<b>SIO_MODE_UART</b>	0x3000 <i>UART comm mode.</i>
#define	<b>SIO_SI_HIGH</b>	0x0004 <i>Normal comm mode, 8-bit.</i>

#define	<b>SIO_IRQ</b> 0x4000 <i>Enable serial irq.</i>
#define	<b>SIO_MODE_MASK</b> 0x3000 <i>Normal comm mode, 8-bit.</i>
#define	<b>SIO_MODE_SHIFT</b> 12 <i>Normal comm mode, 8-bit.</i>
#define	<b>SIO_MODE(n)</b> ((n)<<SIO_MODE_SHIFT) <i>Normal comm mode, 8-bit.</i>

### Normal mode bits. UNTESTED.

#define	<b>SION_CLK_EXT</b> 0x0000 <i>Slave unit; use external clock (default).</i>
#define	<b>SION_CLK_INT</b> 0x0001 <i>Master unit; use internal clock.</i>
#define	<b>SION_256KHZ</b> 0x0000 <i>256 kHz clockspeed (default).</i>
#define	<b>SION_2MHZ</b> 0x0002 <i>2 MHz clockspeed.</i>
#define	<b>SION_RECV_HIGH</b> 0x0004 <i>SI high; opponent ready to receive (R).</i>
#define	<b>SION_SEND_HIGH</b> 0x0008 <i>SO high; ready to transfer.</i>
#define	<b>SION_ENABLE</b> 0x0080 <i>Start transfer/transfer enabled.</i>

### Multiplayer mode bits. UNTESTED.

#define	<b>SIOM_9600</b> 0x0000 <i>Baud rate, 9.6 kbps.</i>
#define	<b>SIOM_38400</b> 0x0001 <i>Baud rate, 38.4 kbps.</i>
#define	<b>SIOM_57600</b> 0x0002 <i>Baud rate, 57.6 kbps.</i>
#define	<b>SIOM_115200</b> 0x0003 <i>Baud rate, 115.2 kbps.</i>

#define	<b>SIOM_SI</b> 0x0004
	<i>SI port (R).</i>
#define	<b>SIOM_SLAVE</b> 0x0004
	<i>Not the master (R).</i>
#define	<b>SIOM_SD</b> 0x0008
	<i>SD port (R).</i>
#define	<b>SIOM_CONNECTED</b> 0x0008
	<i>All GBAs connected (R).</i>
#define	<b>SIOM_ERROR</b> 0x0040
	<i>Error in transfer (R).</i>
#define	<b>SIOM_ENABLE</b> 0x0080
	<i>Start transfer/transfer enabled.</i>
#define	<b>SIOM_BAUD_MASK</b> 0x0003
	<i>Baud rate, 9.6 kbps.</i>
#define	<b>SIOM_BAUD_SHIFT</b> 0
	<i>Baud rate, 9.6 kbps.</i>
#define	<b>SIOM_BAUD(n)</b> ((n)<<SIOM_BAUD_SHIFT)
	<i>Baud rate, 9.6 kbps.</i>
#define	<b>SIOM_ID_MASK</b> 0x0030
	<i>Multi-player ID mask (R).</i>
#define	<b>SIOM_ID_SHIFT</b> 4
	<i>Baud rate, 9.6 kbps.</i>
#define	<b>SIOM_ID(n)</b> ((n)<<SIOM_ID_SHIFT)
	<i>Baud rate, 9.6 kbps.</i>

### UART mode bits. UNTESTED.

#define	<b>SIOU_9600</b> 0x0000
	<i>Baud rate, 9.6 kbps.</i>
#define	<b>SIOU_38400</b> 0x0001
	<i>Baud rate, 38.4 kbps.</i>
#define	<b>SIOU_57600</b> 0x0002
	<i>Baud rate, 57.6 kbps.</i>
#define	<b>SIOU_115200</b> 0x0003
	<i>Baud rate, 115.2 kbps.</i>

#define	<b>SIOU_CTS</b>	0x0004
		<i>CTS enable.</i>
#define	<b>SIOU_PARITY EVEN</b>	0x0000
		<i>Use even parity.</i>
#define	<b>SIOU_PARITY ODD</b>	0x0008
		<i>Use odd parity.</i>
#define	<b>SIOU_SEND FULL</b>	0x0010
		<i>Send data is full (R).</i>
#define	<b>SIOU_RECV EMPTY</b>	0x0020
		<i>Receive data is empty (R).</i>
#define	<b>SIOU_ERROR</b>	0x0040
		<i>Error in transfer (R).</i>
#define	<b>SIOU_7BIT</b>	0x0000
		<i>Data is 7bits long.</i>
#define	<b>SIOU_8BIT</b>	0x0080
		<i>Data is 8bits long.</i>
#define	<b>SIOU_SEND</b>	0x0100
		<i>Start sending data.</i>
#define	<b>SIOU_RECV</b>	0x0200
		<i>Start receiving data.</i>
#define	<b>SIOU_BAUD_MASK</b>	0x0003
		<i>Baud rate, 9.6 kbps.</i>
#define	<b>SIOU_BAUD_SHIFT</b>	0
		<i>Baud rate, 9.6 kbps.</i>
#define	<b>SIOU_BAUD(n)</b>	((n)<<SIOU_BAUD_SHIFT)
		<i>Baud rate, 9.6 kbps.</i>

## Communication mode select.

#define	<b>R_MODE_NORMAL</b>	0x0000
		<i>Normal mode.</i>
#define	<b>R_MODE_MULTI</b>	0x0000
		<i>Multiplayer mode.</i>
#define	<b>R_MODE_UART</b>	0x0000
		<i>UART mode.</i>

#define	<b>R_MODE_GPIO</b> 0x8000 <i>General purpose mode.</i>
#define	<b>R_MODE_JOYBUS</b> 0xC000 <i>JOY mode.</i>
#define	<b>R_MODE_MASK</b> 0xC000 <i>Normal mode.</i>
#define	<b>R_MODE_SHIFT</b> 14 <i>Normal mode.</i>
#define	<b>R_MODE(n)</b> ((n)<<R_MODE_SHIFT) <i>Normal mode.</i>

## General purpose I/O data

#define	<b>GPIO_SC</b> 0x0001
#define	<b>GPIO_SD</b> 0x0002
#define	<b>GPIO_SI</b> 0x0004
#define	<b>GPIO_SO</b> 0x0008
#define	<b>GPIO_SC_IO</b> 0x0010
#define	<b>GPIO_SD_IO</b> 0x0020
#define	<b>GPIO_SI_IO</b> 0x0040
#define	<b>GPIO_SO_IO</b> 0x0080
#define	<b>GPIO_SC_INPUT</b> 0x0000
#define	<b>GPIO_SD_INPUT</b> 0x0000
#define	<b>GPIO_SI_INPUT</b> 0x0000
#define	<b>GPIO_SO_INPUT</b> 0x0000
#define	<b>GPIO_SC_OUTPUT</b> 0x0010
#define	<b>GPIO_SD_OUTPUT</b> 0x0020
#define	<b>GPIO_SI_OUTPUT</b> 0x0040
#define	<b>GPIO_SO_OUTPUT</b> 0x0080
#define	<b>GPIO_IRQ</b> 0x0100

---

# Detailed Description

**Author:**

J Vijn

**Date:**

20060508 - 20080521

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tonc\_memmap.h File Reference**

```
#include "tonc_types.h"
```

## Defines

```
#define REG_BLDMOD *(vu16*)(REG_BASE+0x0050)
#define REG_COLEV *(vu16*)(REG_BASE+0x0052)
#define REG_COLEY *(vu16*)(REG_BASE+0x0054)
#define REG_SOUND1CNT *(vu32*)(REG_BASE+0x0060)
#define REG_SOUND1CNT_L *(vu16*)(REG_BASE+0x0060)
#define REG_SOUND1CNT_H *(vu16*)(REG_BASE+0x0062)
#define REG_SOUND1CNT_X *(vu16*)(REG_BASE+0x0064)
#define REG_SOUND2CNT_L *(vu16*)(REG_BASE+0x0068)
#define REG_SOUND2CNT_H *(vu16*)(REG_BASE+0x006C)
#define REG_SOUND3CNT *(vu32*)(REG_BASE+0x0070)
#define REG_SOUND3CNT_L *(vu16*)(REG_BASE+0x0070)
#define REG_SOUND3CNT_H *(vu16*)(REG_BASE+0x0072)
#define REG_SOUND3CNT_X *(vu16*)(REG_BASE+0x0074)
#define REG_SOUND4CNT_L *(vu16*)(REG_BASE+0x0078)
#define REG_SOUND4CNT_H *(vu16*)(REG_BASE+0x007C)
#define REG_SOUNDCNT *(vu32*)(REG_BASE+0x0080)
#define REG_SOUNDCNT_L *(vu16*)(REG_BASE+0x0080)
#define REG_SOUNDCNT_H *(vu16*)(REG_BASE+0x0082)
#define REG_SOUNDCNT_X *(vu16*)(REG_BASE+0x0084)
#define REG_SOUNDBIAS *(vu16*)(REG_BASE+0x0088)
#define REG_WAVE (vu32*)(REG_BASE+0x0090)
#define REG_FIFOA *(vu32*)(REG_BASE+0x00A0)
#define REG_FIFOB *(vu32*)(REG_BASE+0x00A4)
#define REG_DMA0CNT_L *(vu16*)(REG_BASE+0x00B8)
#define REG_DMA0CNT_H *(vu16*)(REG_BASE+0x00BA)
#define REG_DMA1CNT_L *(vu16*)(REG_BASE+0x00C4)
#define REG_DMA1CNT_H *(vu16*)(REG_BASE+0x00C6)
#define REG_DMA2CNT_L *(vu16*)(REG_BASE+0x00D0)
#define REG_DMA2CNT_H *(vu16*)(REG_BASE+0x00D2)
#define REG_DMA3CNT_L *(vu16*)(REG_BASE+0x00DC)
#define REG_DMA3CNT_H *(vu16*)(REG_BASE+0x00DE)
#define REG_TM0CNT_L *(vu16*)(REG_BASE+0x0100)
```

```

#define REG_TM0CNT_H *(vu16*)(REG_BASE+0x0102)
#define REG_TM1CNT_L *(vu16*)(REG_BASE+0x0104)
#define REG_TM1CNT_H *(vu16*)(REG_BASE+0x0106)
#define REG_TM2CNT_L *(vu16*)(REG_BASE+0x0108)
#define REG_TM2CNT_H *(vu16*)(REG_BASE+0x010a)
#define REG_TM3CNT_L *(vu16*)(REG_BASE+0x010c)
#define REG_TM3CNT_H *(vu16*)(REG_BASE+0x010e)
#define REG_KEYS *(vu16*)(REG_BASE+0x0130)
#define REG_P1 *(vu16*)(REG_BASE+0x0130)
#define REG_P1CNT *(vu16*)(REG_BASE+0x0132)
#define REG_SCD0 *(vu16*)(REG_BASE+0x0120)
#define REG_SCD1 *(vu16*)(REG_BASE+0x0122)
#define REG_SCD2 *(vu16*)(REG_BASE+0x0124)
#define REG_SCD3 *(vu16*)(REG_BASE+0x0126)
#define REG_SCCNT *(vu32*)(REG_BASE+0x0128)
#define REG_SCCNT_L *(vu16*)(REG_BASE+0x0128)
#define REG_SCCNT_H *(vu16*)(REG_BASE+0x012A)
#define REG_R *(vu16*)(REG_BASE+0x0134)
#define REG_HS_CTRL *(vu16*)(REG_BASE+0x0140)
#define REG_JOYRE *(vu32*)(REG_BASE+0x0150)
#define REG_JOYRE_L *(vu16*)(REG_BASE+0x0150)
#define REG_JOYRE_H *(vu16*)(REG_BASE+0x0152)
#define REG_JOYTR *(vu32*)(REG_BASE+0x0154)
#define REG_JOYTR_L *(vu16*)(REG_BASE+0x0154)
#define REG_JOYTR_H *(vu16*)(REG_BASE+0x0156)
#define REG_JSTAT *(vu16*)(REG_BASE+0x0158)
#define REG_WSCNT *(vu16*)(REG_BASE+0x0204)

```

## Main sections

#define	<b>MEM_EWRAM</b> 0x02000000
	<i>External work RAM.</i>
#define	<b>MEM_IWRAM</b> 0x03000000
	<i>Internal work RAM.</i>
#define	<b>MEM_IO</b> 0x04000000
	<i>I/O registers.</i>
#define	<b>MEM_PAL</b> 0x05000000

	<i>Palette. Note: no 8bit write !!</i>
#define	<b>MEM_VRAM</b> 0x06000000 <i>Video RAM. Note: no 8bit write !!</i>
#define	<b>MEM_OAM</b> 0x07000000 <i>Object Attribute Memory (OAM) Note: no 8bit write !!</i>
#define	<b>MEM_ROM</b> 0x08000000 <i>ROM. No write at all (duh).</i>
#define	<b>MEM_SRAM</b> 0x0E000000 <i>Static RAM. 8bit write only.</i>

## Main section sizes

#define	<b>EWRAM_SIZE</b> 0x40000
#define	<b>IWRAM_SIZE</b> 0x08000
#define	<b>PAL_SIZE</b> 0x00400
#define	<b>VRAM_SIZE</b> 0x18000
#define	<b>OAM_SIZE</b> 0x00400
#define	<b>SRAM_SIZE</b> 0x10000

## Sub section sizes

#define	<b>PAL_BG_SIZE</b> 0x00200 <i>BG palette size.</i>
#define	<b>PAL_OBJ_SIZE</b> 0x00200 <i>Object palette size.</i>
#define	<b>CBB_SIZE</b> 0x04000 <i>Charblock size.</i>
#define	<b>SBB_SIZE</b> 0x00800 <i>Screenblock size.</i>
#define	<b>VRAM_BG_SIZE</b> 0x10000 <i>BG VRAM size.</i>
#define	<b>VRAM_OBJ_SIZE</b> 0x08000 <i>Object VRAM size.</i>
#define	<b>M3_SIZE</b> 0x12C00 <i>Mode 3 buffer size.</i>
#define	<b>M4_SIZE</b> 0x09600 <i>Mode 4 buffer size.</i>

#define	<b>M5_SIZE</b> 0x0A000 <i>Mode 5 buffer size.</i>
#define	<b>VRAM_PAGE_SIZE</b> 0x0A000 <i>Bitmap page size.</i>

## Sub sections

#define	<b>REG_BASE</b> MEM_IO
#define	<b>MEM_PAL_BG</b> (MEM_PAL) <i>Background palette address.</i>
#define	<b>MEM_PAL_OBJ</b> (MEM_PAL + PAL_BG_SIZE) <i>Object palette address.</i>
#define	<b>MEM_VRAM_FRONT</b> (MEM_VRAM) <i>Front page address.</i>
#define	<b>MEM_VRAM_BACK</b> (MEM_VRAM + VRAM_PAGE_SIZE) <i>Back page address.</i>
#define	<b>MEM_VRAM_OBJ</b> (MEM_VRAM + VRAM_BG_SIZE) <i>Object VRAM address.</i>

## Palette

#define	<b>pal_bg_mem</b> ((COLOR*)MEM_PAL) <i>Background palette.</i>
#define	<b>pal_obj_mem</b> ((COLOR*)MEM_PAL_OBJ) <i>Object palette.</i>
#define	<b>pal_bg_bank</b> ((PALBANK*)MEM_PAL) <i>Background palette matrix.</i>
#define	<b>pal_obj_bank</b> ((PALBANK*)MEM_PAL_OBJ) <i>Object palette matrix.</i>

## VRAM

#define	<b>tile_mem</b> ( (CHARBLOCK*)MEM_VRAM) <i>Charblocks, 4bpp tiles.</i>
#define	<b>tile8_mem</b> ((CHARBLOCK8*)MEM_VRAM) <i>Charblocks, 8bpp tiles.</i>
#define	<b>tile_mem_obj</b> ( (CHARBLOCK*)MEM_VRAM_OBJ) <i>Object charblocks, 4bpp tiles.</i>

#define	<b>tile8_mem_obj</b> ((CHARBLOCK8*)MEM_VRAM_OBJ)
	<i>Object charblocks, 4bpp tiles.</i>
#define	<b>se_mem</b> ((SCREENBLOCK*)MEM_VRAM)
	<i>Screenblocks as arrays.</i>
#define	<b>se_mat</b> ((SCREENMAT*)MEM_VRAM)
	<i>Screenblock as matrices.</i>
#define	<b>vid_mem</b> ((COLOR*)MEM_VRAM)
	<i>Main mode 3/5 frame as an array.</i>
#define	<b>m3_mem</b> ((M3LINE*)MEM_VRAM)
	<i>Mode 3 frame as a matrix.</i>
#define	<b>m4_mem</b> ((M4LINE*)MEM_VRAM)
	<i>Mode 4 first page as a matrix.</i>
#define	<b>m5_mem</b> ((M5LINE*)MEM_VRAM)
	<i>Mode 5 first page as a matrix.</i>
#define	<b>vid_mem_front</b> ((COLOR*)MEM_VRAM)
	<i>First page array.</i>
#define	<b>vid_mem_back</b> ((COLOR*)MEM_VRAM_BACK)
	<i>Second page array.</i>
#define	<b>m4_mem_back</b> ((M4LINE*)MEM_VRAM_BACK)
	<i>Mode 4 second page as a matrix.</i>
#define	<b>m5_mem_back</b> ((M5LINE*)MEM_VRAM_BACK)
	<i>Mode 5 second page as a matrix.</i>

## OAM

#define	<b>oam_mem</b> ((OBJ_ATTR*)MEM_OAM)
	<i>Object attribute memory.</i>
#define	<b>obj_mem</b> ((OBJ_ATTR*)MEM_OAM)
	<i>Object attribute memory.</i>
#define	<b>obj_aff_mem</b> ((OBJ_AFFINE*)MEM_OAM)
	<i>Object affine memory.</i>

## ROM

#define	<b>rom_mem</b> ((u16*)MEM_ROM)
	<i>ROM pointer.</i>

## SRAM

#define	<b>sram_mem</b> ((u8*)MEM_SRAM) <i>SRAM pointer.</i>
---------	---

## IWRAM 'registers'

#define	<b>REG_IFBIOS</b> *(vu16*)(REG_BASE-0x0008) <i>IRQ ack for IntrWait functions.</i>
#define	<b>REG_RESET_DST</b> *(vu16*)(REG_BASE-0x0006) <i>Destination for after SoftReset.</i>
#define	<b>REG_ISR_MAIN</b> *(fnptr*)(REG_BASE-0x0004) <i>IRQ handler address.</i>

## Display registers

#define	<b>REG_DISPCNT</b> *(vu32*)(REG_BASE+0x0000) <i>Display control.</i>
#define	<b>REG_DISPSTAT</b> *(vu16*)(REG_BASE+0x0004) <i>Display status.</i>
#define	<b>REG_VCOUNT</b> *(vu16*)(REG_BASE+0x0006) <i>Scanline count.</i>

## Background control registers

#define	<b>REG_BGCNT</b> ((vu16*)(REG_BASE+0x0008)) <i>Bg control array.</i>
#define	<b>REG_BG0CNT</b> *(vu16*)(REG_BASE+0x0008) <i>Bg0 control.</i>
#define	<b>REG_BG1CNT</b> *(vu16*)(REG_BASE+0x000A) <i>Bg1 control.</i>
#define	<b>REG_BG2CNT</b> *(vu16*)(REG_BASE+0x000C) <i>Bg2 control.</i>
#define	<b>REG_BG3CNT</b> *(vu16*)(REG_BASE+0x000E) <i>Bg3 control.</i>

## Regular background scroll registers. (write only!)

#define	<b>REG_BG_OFS</b> ((BG_POINT*)(REG_BASE+0x0010)) <i>Bg scroll array.</i>
---------	---

#define	<b>REG_BG0HOFS</b> *(vu16*)(REG_BASE+0x0010)
	<i>Bg0 horizontal scroll.</i>
#define	<b>REG_BG0VOFS</b> *(vu16*)(REG_BASE+0x0012)
	<i>Bg0 vertical scroll.</i>
#define	<b>REG_BG1HOFS</b> *(vu16*)(REG_BASE+0x0014)
	<i>Bg1 horizontal scroll.</i>
#define	<b>REG_BG1VOFS</b> *(vu16*)(REG_BASE+0x0016)
	<i>Bg1 vertical scroll.</i>
#define	<b>REG_BG2HOFS</b> *(vu16*)(REG_BASE+0x0018)
	<i>Bg2 horizontal scroll.</i>
#define	<b>REG_BG2VOFS</b> *(vu16*)(REG_BASE+0x001A)
	<i>Bg2 vertical scroll.</i>
#define	<b>REG_BG3HOFS</b> *(vu16*)(REG_BASE+0x001C)
	<i>Bg3 horizontal scroll.</i>
#define	<b>REG_BG3VOFS</b> *(vu16*)(REG_BASE+0x001E)
	<i>Bg3 vertical scroll.</i>

## Affine background parameters. (write only!)

#define	<b>REG_BG_AFFINE</b> (( <b>BG_AFFINE</b> *)(REG_BASE+0x0000))
	<i>Bg affine array.</i>
#define	<b>REG_BG2PA</b> *(vs16*)(REG_BASE+0x0020)
	<i>Bg2 matrix.pa.</i>
#define	<b>REG_BG2PB</b> *(vs16*)(REG_BASE+0x0022)
	<i>Bg2 matrix.pb.</i>
#define	<b>REG_BG2PC</b> *(vs16*)(REG_BASE+0x0024)
	<i>Bg2 matrix.pc.</i>
#define	<b>REG_BG2PD</b> *(vs16*)(REG_BASE+0x0026)
	<i>Bg2 matrix.pd.</i>
#define	<b>REG_BG2X</b> *(vs32*)(REG_BASE+0x0028)
	<i>Bg2 x scroll.</i>
#define	<b>REG_BG2Y</b> *(vs32*)(REG_BASE+0x002C)
	<i>Bg2 y scroll.</i>
#define	<b>REG_BG3PA</b> *(vs16*)(REG_BASE+0x0030)
	<i>Bg3 matrix.pa.</i>
#define	<b>REG_BG3PB</b> *(vs16*)(REG_BASE+0x0032)

	<i>Bg3 matrix.pb.</i>
#define	<b>REG_BG3PC</b> *(vs16*)(REG_BASE+0x0034) <i>Bg3 matrix.pc.</i>
#define	<b>REG_BG3PD</b> *(vs16*)(REG_BASE+0x0036) <i>Bg3 matrix(pd).</i>
#define	<b>REG_BG3X</b> *(vs32*)(REG_BASE+0x0038) <i>Bg3 x scroll.</i>
#define	<b>REG_BG3Y</b> *(vs32*)(REG_BASE+0x003C) <i>Bg3 y scroll.</i>

## Windowing registers

#define	<b>REG_WIN0H</b> *(vu16*)(REG_BASE+0x0040) <i>win0 right, left (0xLLRR)</i>
#define	<b>REG_WIN1H</b> *(vu16*)(REG_BASE+0x0042) <i>win1 right, left (0xLLRR)</i>
#define	<b>REG_WIN0V</b> *(vu16*)(REG_BASE+0x0044) <i>win0 bottom, top (0xTTBB)</i>
#define	<b>REG_WIN1V</b> *(vu16*)(REG_BASE+0x0046) <i>win1 bottom, top (0xTTBB)</i>
#define	<b>REG_WININ</b> *(vu16*)(REG_BASE+0x0048) <i>win0, win1 control</i>
#define	<b>REG_WINOUT</b> *(vu16*)(REG_BASE+0x004A) <i>winOut, winObj control</i>

## Alternate Windowing registers

#define	<b>REG_WIN0R</b> *(vu8*)(REG_BASE+0x0040) <i>Win 0 right.</i>
#define	<b>REG_WIN0L</b> *(vu8*)(REG_BASE+0x0041) <i>Win 0 left.</i>
#define	<b>REG_WIN1R</b> *(vu8*)(REG_BASE+0x0042) <i>Win 1 right.</i>
#define	<b>REG_WIN1L</b> *(vu8*)(REG_BASE+0x0043) <i>Win 1 left.</i>
#define	<b>REG_WIN0B</b> *(vu8*)(REG_BASE+0x0044) <i>Win 0 bottom.</i>

#define	<b>REG_WIN0T</b> *(vu8*)(REG_BASE+0x0045) <i>Win 0 top.</i>
#define	<b>REG_WIN1B</b> *(vu8*)(REG_BASE+0x0046) <i>Win 1 bottom.</i>
#define	<b>REG_WIN1T</b> *(vu8*)(REG_BASE+0x0047) <i>Win 1 top.</i>
#define	<b>REG_WIN0CNT</b> *(vu8*)(REG_BASE+0x0048) <i>window 0 control</i>
#define	<b>REG_WIN1CNT</b> *(vu8*)(REG_BASE+0x0049) <i>window 1 control</i>
#define	<b>REG_WINOUTCNT</b> *(vu8*)(REG_BASE+0x004A) <i>Out window control.</i>
#define	<b>REG_WINOBJCNT</b> *(vu8*)(REG_BASE+0x004B) <i>Obj window control.</i>

## Graphic effects

#define	<b>REG_MOSAIC</b> *(vu32*)(REG_BASE+0x004C) <i>Mosaic control.</i>
#define	<b>REG_BLDCNT</b> *(vu16*)(REG_BASE+0x0050) <i>Alpha control.</i>
#define	<b>REG_BLDALPHA</b> *(vu16*)(REG_BASE+0x0052) <i>Fade level.</i>
#define	<b>REG_BLDY</b> *(vu16*)(REG_BASE+0x0054) <i>Blend levels.</i>

## Channel 1: Square wave with sweep

#define	<b>REG SND1SWEEP</b> *(vu16*)(REG_BASE+0x0060) <i>Channel 1 Sweep.</i>
#define	<b>REG SND1CNT</b> *(vu16*)(REG_BASE+0x0062) <i>Channel 1 Control.</i>
#define	<b>REG SND1FREQ</b> *(vu16*)(REG_BASE+0x0064) <i>Channel 1 frequency.</i>

## Channel 2: Simple square wave

#define	<b>REG SND2CNT</b> *(vu16*)(REG_BASE+0x0068)
---------	--

	<i>Channel 2 control.</i>
#define	<b>REG_SND2FREQ</b> *(vu16*)(REG_BASE+0x006C) <i>Channel 2 frequency.</i>

### Channel 3: Wave player

#define	<b>REG_SND3SEL</b> *(vu16*)(REG_BASE+0x0070) <i>Channel 3 wave select.</i>
#define	<b>REG_SND3CNT</b> *(vu16*)(REG_BASE+0x0072) <i>Channel 3 control.</i>
#define	<b>REG_SND3FREQ</b> *(vu16*)(REG_BASE+0x0074) <i>Channel 3 frequency.</i>

### Channel 4: Noise generator

#define	<b>REG_SND4CNT</b> *(vu16*)(REG_BASE+0x0078) <i>Channel 4 control.</i>
#define	<b>REG_SND4FREQ</b> *(vu16*)(REG_BASE+0x007C) <i>Channel 4 frequency.</i>

### Sound control

#define	<b>REG_SNDCNT</b> *(vu32*)(REG_BASE+0x0080) <i>Main sound control.</i>
#define	<b>REG_SNDDMGCNT</b> *(vu16*)(REG_BASE+0x0080) <i>DMG channel control.</i>
#define	<b>REG_SNDDSCNT</b> *(vu16*)(REG_BASE+0x0082) <i>Direct Sound control.</i>
#define	<b>REG SNDSTAT</b> *(vu16*)(REG_BASE+0x0084) <i>Sound status.</i>
#define	<b>REG_SNDBIAS</b> *(vu16*)(REG_BASE+0x0088) <i>Sound bias.</i>

### Sound buffers

#define	<b>REG_WAVE_RAM</b> (vu32*)(REG_BASE+0x0090) <i>Channel 3 wave buffer.</i>
#define	<b>REG_WAVE_RAM0</b> *(vu32*)(REG_BASE+0x0090) <i>Channel 3 wave buffer.</i>

#define	<b>REG_WAVE_RAM1</b> *(vu32*)(REG_BASE+0x0094) <i>Channel 3 wave buffer.</i>
#define	<b>REG_WAVE_RAM2</b> *(vu32*)(REG_BASE+0x0098) <i>Channel 3 wave buffer.</i>
#define	<b>REG_WAVE_RAM3</b> *(vu32*)(REG_BASE+0x009C) <i>Channel 3 wave buffer.</i>
#define	<b>REG_FIFO_A</b> *(vu32*)(REG_BASE+0x00A0) <i>DSound A FIFO.</i>
#define	<b>REG_FIFO_B</b> *(vu32*)(REG_BASE+0x00A4) <i>DSound B FIFO.</i>

## DMA registers

#define	<b>REG_DMA</b> ((volatile <b>DMA_REC</b> *) (REG_BASE+0x00B0)) <i>DMA as <b>DMA_REC</b> array.</i>
#define	<b>REG_DMA0SAD</b> *(vu32*)(REG_BASE+0x00B0) <i>DMA 0 Source address.</i>
#define	<b>REG_DMA0DAD</b> *(vu32*)(REG_BASE+0x00B4) <i>DMA 0 Destination address.</i>
#define	<b>REG_DMA0CNT</b> *(vu32*)(REG_BASE+0x00B8) <i>DMA 0 Control.</i>
#define	<b>REG_DMA1SAD</b> *(vu32*)(REG_BASE+0x00BC) <i>DMA 1 Source address.</i>
#define	<b>REG_DMA1DAD</b> *(vu32*)(REG_BASE+0x00C0) <i>DMA 1 Destination address.</i>
#define	<b>REG_DMA1CNT</b> *(vu32*)(REG_BASE+0x00C4) <i>DMA 1 Control.</i>
#define	<b>REG_DMA2SAD</b> *(vu32*)(REG_BASE+0x00C8) <i>DMA 2 Source address.</i>
#define	<b>REG_DMA2DAD</b> *(vu32*)(REG_BASE+0x00CC) <i>DMA 2 Destination address.</i>
#define	<b>REG_DMA2CNT</b> *(vu32*)(REG_BASE+0x00D0) <i>DMA 2 Control.</i>
#define	<b>REG_DMA3SAD</b> *(vu32*)(REG_BASE+0x00D4) <i>DMA 3 Source address.</i>
#define	<b>REG_DMA3DAD</b> *(vu32*)(REG_BASE+0x00D8)

	DMA 3 Destination address.
#define	<b>REG_DMA3CNT</b> *(vu32*)(REG_BASE+0x00DC) DMA 3 Control.

## Timer registers

#define	<b>REG_TM</b> ((volatile <b>TMR_REC</b> *)((REG_BASE+0x0100))) Timers as <b>TMR_REC</b> array.
#define	<b>REG_TM0D</b> *(vu16*)(REG_BASE+0x0100) Timer 0 data.
#define	<b>REG_TM0CNT</b> *(vu16*)(REG_BASE+0x0102) Timer 0 control.
#define	<b>REG_TM1D</b> *(vu16*)(REG_BASE+0x0104) Timer 1 data.
#define	<b>REG_TM1CNT</b> *(vu16*)(REG_BASE+0x0106) Timer 1 control.
#define	<b>REG_TM2D</b> *(vu16*)(REG_BASE+0x0108) Timer 2 data.
#define	<b>REG_TM2CNT</b> *(vu16*)(REG_BASE+0x010A) Timer 2 control.
#define	<b>REG_TM3D</b> *(vu16*)(REG_BASE+0x010C) Timer 3 data.
#define	<b>REG_TM3CNT</b> *(vu16*)(REG_BASE+0x010E) Timer 3 control.

## Serial communication

#define	<b>REG_SIOCNT</b> *(vu16*)(REG_BASE+0x0128) Serial IO control (Normal/MP/UART).
#define	<b>REG_SIODATA</b> ((vu32*)(REG_BASE+0x0120)) Serial IO control (Normal/MP/UART).
#define	<b>REG_SIODATA32</b> *(vu32*)(REG_BASE+0x0120) Normal/UART 32bit data.
#define	<b>REG_SIODATA8</b> *(vu16*)(REG_BASE+0x012A) Normal/UART 8bit data.
#define	<b>REG_SIOMULTI</b> ((vu16*)(REG_BASE+0x0120)) Multiplayer data array.

#define	<b>REG_SIOMULTI0</b> *(vu16*)(REG_BASE+0x0120) <i>MP master data.</i>
#define	<b>REG_SIOMULTI1</b> *(vu16*)(REG_BASE+0x0122) <i>MP Slave 1 data.</i>
#define	<b>REG_SIOMULTI2</b> *(vu16*)(REG_BASE+0x0124) <i>MP Slave 2 data.</i>
#define	<b>REG_SIOMULTI3</b> *(vu16*)(REG_BASE+0x0126) <i>MP Slave 3 data.</i>
#define	<b>REG_SIOMLT_RECV</b> *(vu16*)(REG_BASE+0x0120) <i>MP data receiver.</i>
#define	<b>REG_SIOMLT_SEND</b> *(vu16*)(REG_BASE+0x012A) <i>MP data sender.</i>

## Keypad registers

#define	<b>REG_KEYINPUT</b> *(vu16*)(REG_BASE+0x0130) <i>Key status (read only??).</i>
#define	<b>REG_KEYCNT</b> *(vu16*)(REG_BASE+0x0132) <i>Key IRQ control.</i>

## Joybus communication

#define	<b>REG_RCNT</b> *(vu16*)(REG_BASE+0x0134) <i>SIO Mode Select/General Purpose Data.</i>
#define	<b>REG_JOYCNT</b> *(vu16*)(REG_BASE+0x0140) <i>JOY bus control.</i>
#define	<b>REG_JOY_RECV</b> *(vu32*)(REG_BASE+0x0150) <i>JOY bus receiver.</i>
#define	<b>REG_JOY_TRANS</b> *(vu32*)(REG_BASE+0x0154) <i>JOY bus transmitter.</i>
#define	<b>REG_JOYSTAT</b> *(vu16*)(REG_BASE+0x0158) <i>JOY bus status.</i>

## Interrupt / System registers

#define	<b>REG_IE</b> *(vu16*)(REG_BASE+0x0200) <i>IRQ enable.</i>
#define	<b>REG_IF</b> *(vu16*)(REG_BASE+0x0202)

	<i>IRQ status/acknowledge.</i>
#define	<b>REG_WAITCNT</b> *(vu16*)(REG_BASE+0x0204) <i>Waitstate control.</i>
#define	<b>REG_IME</b> *(vu16*)(REG_BASE+0x0208) <i>IRQ master enable.</i>
#define	<b>REG_PAUSE</b> *(vu16*)(REG_BASE+0x0300) <i>Pause system (?).</i>

---

# Detailed Description

**Author:**

J Vijn

**Date:**

20060508 - 20060508

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tonc\_nocash.h File Reference**

```
#include "tonc_types.h"
```

## Functions

	int <b>nocash_puts</b> (const char *str) <i>Output a string to no\$gba debugger.</i>
EWRAM_CODE void	<b>nocash_message</b> (void) <i>Print the current nocash_buffer to the no\$gba debugger.</i>

## Variables

```
EWRAM_DATA char nocash_buffer [80]
```

---

# Detailed Description

**Author:**

J Vijn

**Date:**

20080422 - 20080422

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tonc\_oam.c File Reference**

```
#include "tonc_memmap.h" #include "tonc_memdef.h"
#include "tonc_oam.h"
```

## Functions

void	<b>oam_init</b> ( <b>OBJ_ATTR</b> *obj, u32 count)
<i>Initialize an array of count OBJ_ATTRs with safe values.</i>	
void	<b>obj_copy</b> ( <b>OBJ_ATTR</b> *dst, const <b>OBJ_ATTR</b> *src, u32 count)
<i>Copy attributes 0-2 in count OBJ_ATTRs.</i>	
void	<b>obj_hide_multi</b> ( <b>OBJ_ATTR</b> *obj, u32 count)
<i>Hide an array of OBJ_ATTRs.</i>	
void	<b>obj_unhide_multi</b> ( <b>OBJ_ATTR</b> *obj, u16 mode, u32 count)

# Detailed Description

**Author:**

J Vijn

**Date:**

20060604 - 20060604

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tonc\_oam.h File Reference**

```
#include "tonc_memmap.h" #include "tonc_memdef.h"  
#include "tonc_core.h"
```

## Defines

```
#define OAM_CLEAR() memset32(oam_mem, 0, OAM_SIZE/4)
```

# Functions

void	<b>oam_init</b> ( <b>OBJ_ATTR</b> *obj, uint count) <i>Initialize an array of count OBJ_ATTRs with safe values.</i>
INLINE void	<b>oam_copy</b> ( <b>OBJ_ATTR</b> *dst, const <b>OBJ_ATTR</b> *src, uint count) <i>Copies count OAM entries from src to dst.</i>
INLINE <b>OBJ_ATTR</b> *	<b>obj_set_attr</b> ( <b>OBJ_ATTR</b> *obj, u16 a0, u16 a1, u16 a2) <i>Set the attributes of an object.</i>
INLINE void	<b>obj_set_pos</b> ( <b>OBJ_ATTR</b> *obj, int x, int y) <i>Set the position of obj.</i>
INLINE void	<b>obj_hide</b> ( <b>OBJ_ATTR</b> *obj) <i>Hide an object.</i>
INLINE void	<b>obj_unhide</b> ( <b>OBJ_ATTR</b> *obj, u16 mode) <i>Unhide an object.</i>
INLINE const u8 *	<b>obj_get_size</b> (const <b>OBJ_ATTR</b> *obj) <i>Get object's sizes as a byte array.</i>
INLINE int	<b>obj_get_width</b> (const <b>OBJ_ATTR</b> *obj) <i>Get object's width.</i>
INLINE int	<b>obj_get_height</b> (const <b>OBJ_ATTR</b> *obj) <i>Gets object's height.</i>
void	<b>obj_copy</b> ( <b>OBJ_ATTR</b> *dst, const <b>OBJ_ATTR</b> *src, uint count) <i>Copy attributes 0-2 in count OBJ_ATTRs.</i>
void	<b>obj_hide_multi</b> ( <b>OBJ_ATTR</b> *obj, u32 count) <i>Hide an array of OBJ_ATTRs.</i>
void	<b>obj_unhide_multi</b> ( <b>OBJ_ATTR</b> *obj, u16 mode, uint count)
void	<b>obj_aff_copy</b> ( <b>OBJ_AFFINE</b> *dst, const <b>OBJ_AFFINE</b> *src, uint count)
INLINE void	<b>obj_aff_set</b> ( <b>OBJ_AFFINE</b> *oaff, <b>FIXED</b> pa, <b>FIXED</b> pb, <b>FIXED</b> pc, <b>FIXED</b> pd) <i>Set the elements of an object affine matrix.</i>

INLINE void	<b>obj_aff_identity (OBJ_AFFINE *oaff)</b> Set an object affine matrix to the identity matrix.
INLINE void	<b>obj_aff_scale (OBJ_AFFINE *oaff, FIXED sx, FIXED sy)</b> Set an object affine matrix for scaling.
INLINE void	<b>obj_aff_shearx (OBJ_AFFINE *oaff, FIXED hx)</b>
INLINE void	<b>obj_aff_sheary (OBJ_AFFINE *oaff, FIXED hy)</b>
void	<b>obj_aff_rotate (OBJ_AFFINE *oaff, u16 alpha)</b> Set obj matrix to counter-clockwise rotation.
void	<b>obj_aff_rotscale (OBJ_AFFINE *oaff, FIXED sx, FIXED sy, u16 alpha)</b> Set obj matrix to 2d scaling, then counter-clockwise rotation.
void	<b>obj_aff_premul (OBJ_AFFINE *dst, const OBJ_AFFINE *src)</b> Pre-multiply dst by src: $D = S*D$ .
void	<b>obj_aff_postmul (OBJ_AFFINE *dst, const OBJ_AFFINE *src)</b> Post-multiply dst by src: $D= D*S$ .
void	<b>obj_aff_rotscale2 (OBJ_AFFINE *oaff, const AFF_SRC *as)</b> Set obj matrix to 2d scaling, then counter-clockwise rotation.
void	<b>obj_rotscale_ex (OBJ_ATTR *obj, OBJ_AFFINE *oaff, const AFF_SRC_EX *asx)</b> Rot/scale an object around an arbitrary point.
INLINE void	<b>obj_aff_scale_inv (OBJ_AFFINE *oa, FIXED wx, FIXED wy)</b>
INLINE void	<b>obj_aff_rotate_inv (OBJ_AFFINE *oa, u16 theta)</b>
INLINE void	<b>obj_aff_shearx_inv (OBJ_AFFINE *oa, FIXED hx)</b>
INLINE void	<b>obj_aff_sheary_inv (OBJ_AFFINE *oa, FIXED hy)</b>

# Detailed Description

**Author:**

J Vijn

**Date:**

20060604 - 20060604

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tonc\_obj\_affine.c File Reference**

```
#include "tonc_memmap.h" #include "tonc_core.h"
#include "tonc_bios.h"
#include "tonc_math.h"
#include "tonc_oam.h"
```

## Functions

void	<b>obj_aff_copy</b> ( <b>OBJ_AFFINE</b> *dst, const <b>OBJ_AFFINE</b> *src, u32 count)
void	<b>obj_aff_rotate</b> ( <b>OBJ_AFFINE</b> *oaff, u16 alpha) Set obj matrix to counter-clockwise rotation.
void	<b>obj_aff_rotscale</b> ( <b>OBJ_AFFINE</b> *oaff, <b>FIXED</b> sx, <b>FIXED</b> sy, u16 alpha) Set obj matrix to 2d scaling, then counter-clockwise rotation.
void	<b>obj_aff_premul</b> ( <b>OBJ_AFFINE</b> *dst, const <b>OBJ_AFFINE</b> *src) Pre-multiply dst by src: $D = S*D$ .
void	<b>obj_aff_postmul</b> ( <b>OBJ_AFFINE</b> *dst, const <b>OBJ_AFFINE</b> *src) Post-multiply dst by src: $D= D*S$ .
void	<b>obj_aff_rotscale2</b> ( <b>OBJ_AFFINE</b> *oaff, const <b>AFF_SRC</b> *as) Set obj matrix to 2d scaling, then counter-clockwise rotation.
void	<b>obj_rotscale_ex</b> ( <b>OBJ_ATTR</b> *obj, <b>OBJ_AFFINE</b> *oaff, const <b>AFF_SRC_EX</b> *asx) Rot/scale an object around an arbitrary point.

# Detailed Description

**Author:**

J Vijn

**Date:**

20060908 - 20060916

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tonc\_sbmp16.c File Reference**

```
#include "tonc_surface.h" #include "tonc_video.h"
```

## Defines

```
#define PXSIZE sizeof(pixel_t)
#define PXPTR(psrf, x, y) (pixel_t*)(psrf->data + (y)*psrf->pitch +
(x)*sizeof(pixel_t) )
#define BLIT_CLIP(_ax, _aw, _w, _bx)
```

## **Typedefs**

```
typedef u16 pixel_t
```

## Functions

void	<b>sbmp16_floodfill_internal</b> (const TSurface *dst, int x, int y, u16 clrNew, u16 clrOld)
u32	<b>sbmp16_get_pixel</b> (const TSurface *src, int x, int y) <i>Get the pixel value of src at (x, y).</i>
void	<b>sbmp16_plot</b> (const TSurface *dst, int x, int y, u32 clr) <i>Plot a single pixel on a 16-bit buffer.</i>
void	<b>sbmp16_hline</b> (const TSurface *dst, int x1, int y, int x2, u32 clr) <i>Draw a horizontal line on an 16bit buffer.</i>
void	<b>sbmp16_vline</b> (const TSurface *dst, int x, int y1, int y2, u32 clr) <i>Draw a vertical line on an 16bit buffer.</i>
void	<b>sbmp16_line</b> (const TSurface *dst, int x1, int y1, int x2, int y2, u32 clr) <i>Draw a line on an 16bit buffer.</i>
void	<b>sbmp16_rect</b> (const TSurface *dst, int left, int top, int right, int bottom, u32 clr) <i>Draw a rectangle in 16bit mode.</i>
void	<b>sbmp16_frame</b> (const TSurface *dst, int left, int top, int right, int bottom, u32 clr) <i>Draw a rectangle in 16bit mode.</i>
void	<b>sbmp16.blit</b> (const TSurface *dst, int dstX, int dstY, uint width, uint height, const TSurface *src, int srcX, int srcY) <i>16bpp blitter. Copies a rectangle from one surface to another.</i>
void	<b>sbmp16_floodfill</b> (const TSurface *dst, int x, int y, u32 clr) <i>Floodfill an area of the same color with new color clr.</i>
void	<b>sbmp16_floodfill_internal</b> (const TSurface *dst, int x, int y, pixel_t clrNew, pixel_t clrOld) <i>Internal routine for floodfill.</i>

## Variables

const TSurface	<b>m3_surface</b>
EWRAM_DATA TSurface	<b>m5_surface</b>
const TSurfaceProcTab	<b>bmp16_tab</b>

---

# **Detailed Description**

**Author:**

J Vijn

**Date:**

20080120 - 20080128

**Todo:**

Code consistency.

---

# Define Documentation

```
#define BLIT_CLIP (_ax,  
                 _aw,  
                 _w,  
                 _bx )
```

**Value:**

```
do {  
    if( (_ax) >= (_aw) || (_ax)+(_w) <= 0  
        return;  
    if( (_ax)<0 )  
    { _w += (_ax); _bx += (_ax); _ax= 0;  
        if( (_w) > (_aw)-(_ax) )  
            _w = (_aw)-(_ax);  
    } while(0)
```

---

## Variable Documentation

**const TSurfaceProcTab bmp16\_tab**

**Initial value:**

```
{  
    "bmp16",  
    sbmp16_get_pixel,  
    sbmp16_plot,  
    sbmp16_hline,  
    sbmp16_vline,  
    sbmp16_line,  
    sbmp16_rect,  
    sbmp16_frame,  
    sbmp16.blit,  
    sbmp16_floodfill,  
}
```

**const TSurface m3\_surface**

**Initial value:**

```
{  
    (u8*)m3_mem , M3_WIDTH*2, M3_WIDTH, M3_HEIGHT  
    0, NULL  
}
```

## EWRAM\_DATA TSurface m5\_surface

**Initial value:**

```
{  
    (u8*)m5_mem , M5_WIDTH*2, M5_WIDTH, M5_HEI  
    0, NULL  
}
```

---

Generated on Mon Aug 25 17:03:56 2008 for libtonc by [doxygen](#) 1.5.3

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tonc\_sbmp8.c File Reference**

```
#include "tonc_surface.h" #include "tonc_video.h"
```

## Defines

```
#define PXSIZE sizeof(pixel_t)
#define PXPTR(psrf, x, y) (pixel_t*)(psrf->data + (y)*psrf->pitch +
(x)*sizeof(pixel_t) )
#define BLIT_CLIP(_ax, _aw, _w, _bx)
```

## Typedefs

```
typedef u8 pixel_t
```

## Functions

void	<b>sbmp8_floodfill_internal</b> (const TSurface *dst, int x, int y, u8 clrNew, u8 clrOld)
u32	<b>sbmp8_get_pixel</b> (const TSurface *src, int x, int y) <i>Get the pixel value of src at (x, y).</i>
void	<b>sbmp8_plot</b> (const TSurface *dst, int x, int y, u32 clr) <i>Plot a single pixel on a 8-bit buffer.</i>
void	<b>sbmp8_hline</b> (const TSurface *dst, int x1, int y, int x2, u32 clr) <i>Draw a horizontal line on an 8-bit buffer.</i>
void	<b>sbmp8_vline</b> (const TSurface *dst, int x, int y1, int y2, u32 clr) <i>Draw a vertical line on an 8-bit buffer.</i>
void	<b>sbmp8_line</b> (const TSurface *dst, int x1, int y1, int x2, int y2, u32 clr) <i>Draw a line on an 8-bit buffer.</i>
void	<b>sbmp8_rect</b> (const TSurface *dst, int left, int top, int right, int bottom, u32 clr) <i>Draw a rectangle in 8-bit mode.</i>
void	<b>sbmp8_frame</b> (const TSurface *dst, int left, int top, int right, int bottom, u32 clr) <i>Draw a rectangle in 8-bit mode.</i>
void	<b>sbmp8_blt</b> (const TSurface *dst, int dstX, int dstY, uint width, uint height, const TSurface *src, int srcX, int srcY) <i>16bpp blitter. Copies a rectangle from one surface to another.</i>
void	<b>sbmp8_floodfill</b> (const TSurface *dst, int x, int y, u32 clr) <i>Floodfill an area of the same color with new color clr.</i>
void	<b>sbmp8_floodfill_internal</b> (const TSurface *dst, int x, int y, pixel_t clrNew, pixel_t clrOld) <i>Internal routine for floodfill.</i>

## Variables

EWRAM_DATA TSurface	<b>m4_surface</b>
const TSurfaceProcTab	<b>bmp8_tab</b>

---

# **Detailed Description**

**Author:**

J Vijn

**Date:**

20080127 - 20080128

**Todo:**

Code consistency.

---

# Define Documentation

```
#define BLIT_CLIP (_ax,  
                 _aw,  
                 _w,  
                 _bx )
```

**Value:**

```
do {  
    if( (_ax) >= (_aw) || (_ax)+(_w) <= 0  
        return;  
    if( (_ax)<0 )  
    { _w += (_ax); _bx += (_ax); _ax= 0;  
        if( (_w) > (_aw)-(_ax) )  
            _w = (_aw)-(_ax);  
    } while(0)
```

---

# Variable Documentation

```
const TSurfaceProcTab bmp8_tab
```

**Initial value:**

```
{  
    "bmp8",  
    sbmp8_get_pixel,  
    sbmp8_plot,  
    sbmp8_hline,  
    sbmp8_vline,  
    sbmp8_line,  
    sbmp8_rect,  
    sbmp8_frame,  
    sbmp8.blit,  
    sbmp8_floodfill,  
}
```

```
EWRAM_DATA TSurface m4_surface
```

**Initial value:**

```
{  
    (u8*)m4_mem, M4_WIDTH, M4_WIDTH, M4_HEIGHT  
    256, pal_bg_mem  
}
```

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  **1.5.3**

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tonc\_schr4c.c File Reference**

```
#include "tonc_surface.h" #include "tonc_video.h"  
#include "tonc_math.h"
```

## Defines

```
#define BLIT_CLIP(_ax, _aw, _w, _bx)
```

# Functions

INLINE u32	<b>chr4_lmask</b> (uint left) <i>Returns the clear-mask for the left side of a fill rect.</i>
INLINE u32	<b>chr4_rmask</b> (uint right) <i>Returns the clear-mask for the right side of a fill rect.</i>
INLINE void	<b>chr4c_plot</b> (int x, int y, u32 clr, void *dstBase, u32 dstP)
INLINE void	<b>chr4c_colset</b> (u32 *dstD, uint left, uint right, uint height, u32 clr) <i>Fill a rectangle inside a simple tile-column.</i>
void	<b>schr4c_floodfill_internal</b> (const TSurface *dst, int x, int y, u32 clrNew, u32 clrOld) <i>Internal routine for floodfill.</i>
void	<b>schr4c_prep_map</b> (const TSurface *srf, u16 *map, u16 se0) <i>Prepare a screen-entry map for use with chr4.</i>
u32 *	<b>schr4c_get_ptr</b> (const TSurface *srf, int x, int y) <i>Special pointer getter for chr4: start of in-tile line.</i>
u32	<b>schr4c_get_pixel</b> (const TSurface *src, int x, int y) <i>Get the pixel value of src at (x, y).</i>
void	<b>schr4c_plot</b> (const TSurface *dst, int x, int y, u32 clr) <i>Plot a single pixel on a 4bpp tiled surface.</i>
void	<b>schr4c_hline</b> (const TSurface *dst, int x1, int y, int x2, u32 clr) <i>Draw a horizontal line on a 4bpp tiled surface.</i>
void	<b>schr4c_vline</b> (const TSurface *dst, int x, int y1, int y2, u32 clr) <i>Draw a vertical line on a 4bpp tiled surface.</i>
void	<b>schr4c_line</b> (const TSurface *dst, int x1, int y1, int x2, int y2, u32 clr) <i>Draw a line on a 4bpp tiled surface.</i>
void	<b>schr4c_rect</b> (const TSurface *dst, int left, int top, int right, int bottom, u32 clr) <i>Render a rectangle on a 4bpp tiled canvas.</i>
void	<b>schr4c_frame</b> (const TSurface *dst, int left, int top, int right, int bottom, u32 clr) <i>Draw a rectangle on a 4bpp tiled surface.</i>
void	<b>schr4c_blit</b> (const TSurface *dst, int dstX, int dstY, uint width,

	<code>uint height, const TSurface *src, int srcX, int srcY)</code> <i>Blitter for 4bpp tiled surfaces. Copies a rectangle from one surface to another.</i>
<code>void</code>	<b>schr4c_floodfill</b> ( <code>const TSurface *dst, int x, int y, u32 clr)</code> <i>Floodfill an area of the same color with new color clr.</i>

## Variables

const TSurfaceProcTab **chr4c\_tab**

---

# **Detailed Description**

**Author:**

J Vijn

**Date:**

20080427 - 20080503

**Todo:**

Code consistency.

---

## Define Documentation

```
#define BLIT_CLIP( _ax,  
                 _aw,  
                 _w,  
                 _bx )
```

**Value:**

```
do {  
    if( (_ax) >= (_aw) || (_ax)+(_w) <= 0  
        return;  
    if( (_ax)<0 )  
    { _w += (_ax); _bx += (_ax); _ax= 0;  
        if( (_w) > (_aw)-(_ax) )  
            _w = (_aw)-(_ax);  
    } while(0)
```

---

# Function Documentation

```
INLINE void chr4c_colset( u32 * dstD,
                           uint left,
                           uint right,
                           uint height,
                           u32 clr
                         )
```

Fill a rectangle inside a simple tile-column.

**Note:**

*left* and *right* must already be between 0 and 8.

```
void schr4c_floodfill_internal( const TSurface * dst,
                                 int x,
                                 int y,
                                 u32 clrNew,
                                 u32 clrOld
                               )
```

Internal routine for floodfill.

**Note:**

This traverses the lines horizontally. Amazingly, this seems faster than vertically.

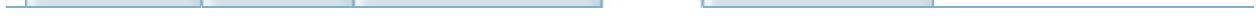
---

# Variable Documentation

```
const TSurfaceProcTab chr4c_tab
```

**Initial value:**

```
{  
    "chr4c",  
    schr4c_get_pixel,  
    schr4c_plot,  
    schr4c_hline,  
    schr4c_vline,  
    schr4c_line,  
    schr4c_rect,  
    schr4c_frame,  
    schr4c.blit,  
    schr4c_floodfill,  
}
```

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tonc\_schr4r.c File Reference**

```
#include "tonc_surface.h" #include "tonc_video.h"
```

# Functions

INLINE u32	<b>chr4_lmask</b> (uint left) <i>Returns the clear-mask for the left side of a fill rect.</i>
INLINE u32	<b>chr4_rmask</b> (uint right) <i>Returns the clear-mask for the right side of a fill rect.</i>
INLINE void	<b>chr4r_plot</b> (int x, int y, u32 clr, void *dstBase, u32 dstP)
INLINE void	<b>chr4r_colset</b> (u32 *dstD, uint dstP4, uint left, uint right, uint height, u32 clr) <i>Fill a rectangle inside a simple tile-column.</i>
void	<b>schr4r_prep_map</b> (const TSurface *srf, u16 *map, u16 se0) <i>Prepare a screen-entry map for use with chr4.</i>
u32 *	<b>schr4r_get_ptr</b> (const TSurface *srf, int x, int y) <i>Special pointer getter for chr4: start of in-tile line.</i>
u32	<b>schr4r_get_pixel</b> (const TSurface *src, int x, int y) <i>Get the pixel value of src at (x, y).</i>
void	<b>schr4r_plot</b> (const TSurface *dst, int x, int y, u32 clr) <i>Plot a single pixel on a 4bpp tiled surface.</i>
void	<b>schr4r_hline</b> (const TSurface *dst, int x1, int y, int x2, u32 clr) <i>Draw a horizontal line on a 4bpp tiled surface.</i>
void	<b>schr4r_vline</b> (const TSurface *dst, int x, int y1, int y2, u32 clr) <i>Draw a vertical line on a 4bpp tiled surface.</i>
void	<b>schr4r_line</b> (const TSurface *dst, int x1, int y1, int x2, int y2, u32 clr) <i>Draw a line on a 4bpp tiled surface.</i>
void	<b>schr4r_rect</b> (const TSurface *dst, int left, int top, int right, int bottom, u32 clr) <i>Render a rectangle on a tiled canvas.</i>
void	<b>schr4r_frame</b> (const TSurface *dst, int left, int top, int right, int bottom, u32 clr) <i>Draw a rectangle on a 4bpp tiled surface.</i>

## **Detailed Description**

**Author:**

J Vijn

**Date:**

20080409 - 20080409

---

# Function Documentation

```
INLINE void chr4r_colset( u32 * dstD,
                           uint dstP4,
                           uint left,
                           uint right,
                           uint height,
                           u32 clr
                         )
```

Fill a rectangle inside a simple tile-column.

**Note:**

*left* and *right* must already be between 0 and 8.

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tonc\_surface.c File Reference**

```
#include <string.h> #include "tonc_surface.h"  
#include "tonc_video.h"
```

## Functions

void	<b>srf_init</b> (TSurface *srf, enum <b>ESurfaceType</b> type, const void *data, uint width, uint height, uint bpp, u16 *pal) <i>Initialize a surface for type formatted graphics.</i>
void	<b>srf_pal_copy</b> (const TSurface *dst, const TSurface *src, uint count) <i>Copy count colors from src's palette to dst's palette.</i>
void *	<b>srf_get_ptr</b> (const TSurface *srf, uint x, uint y) <i>Get the byte address of coordinates (x, y) on the surface.</i>

# Detailed Description

**Author:**

J Vijn

**Date:**

20080409 - 20080409

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tonc\_surface.h** File Reference

```
#include "tonc_memmap.h" #include "tonc_core.h"
```

# Typedefs

## Rendering procedure types

typedef u32(*)	<b>fnGetPixel</b> )(const TSurface *src, int x, int y)
typedef void(*)	<b>fnPlot</b> )(const TSurface *dst, int x, int y, u32 clr)
typedef void(*)	<b>fnHLine</b> )(const TSurface *dst, int x1, int y, int x2, u32 clr)
typedef void(*)	<b>fnVLine</b> )(const TSurface *dst, int x, int y1, int y2, u32 clr)
typedef void(*)	<b>fnLine</b> )(const TSurface *dst, int x1, int y1, int x2, int y2, u32 clr)
typedef void(*)	<b>fnRect</b> )(const TSurface *dst, int left, int top, int right, int bottom, u32 clr)
typedef void(*)	<b>fnFrame</b> )(const TSurface *dst, int left, int top, int right, int bottom, u32 clr)
typedef void(*)	<b>fnBlit</b> )(const TSurface *dst, int dstX, int dstY, uint width, uint height, const TSurface *src, int srcX, int srcY)
typedef void(*)	<b>fnFlood</b> )(const TSurface *dst, int x, int y, u32 clr)

## Enumerations

enum	<pre>ESurfaceType {     SRF_NONE = 0, SRF_BMP16 = 1, SRF_BMP8 = 2, SRF_CHR4R = 4,     SRF_CHR4C = 5, SRF_CHR8 = 6, SRF_ALLOCATED = 0x80 }</pre>
	<i>Surface types.</i> <a href="#">More...</a>

# Functions

void	<b>srf_init</b> (TSurface *srf, enum <b>ESurfaceType</b> type, const void *data, uint width, uint height, uint bpp, u16 *pal) <i>Initialize a surface for type formatted graphics.</i>
void	<b>srf_pal_copy</b> (const TSurface *dst, const TSurface *src, uint count) <i>Copy count colors from src's palette to dst's palette.</i>
void *	<b>srf_get_ptr</b> (const TSurface *srf, uint x, uint y) <i>Get the byte address of coordinates (x, y) on the surface.</i>
INLINE uint	<b>srf_align</b> (uint width, uint bpp) <i>Get the word-aligned number of bytes for a scanline.</i>
INLINE void	<b>srf_set_ptr</b> (TSurface *srf, const void *ptr) <i>Set Data-pointer surface for srf.</i>
INLINE void	<b>srf_set_pal</b> (TSurface *srf, const u16 *pal, uint size) <i>Set the palette pointer and its size.</i>
INLINE void *	<b>_srf_get_ptr</b> (const TSurface *srf, uint x, uint y, uint stride) <i>Inline and semi-safe version of <b>srf_get_ptr()</b>. Use with caution.</i>
u32	<b>sbmp16_get_pixel</b> (const TSurface *src, int x, int y) <i>Get the pixel value of src at (x, y).</i>
void	<b>sbmp16_plot</b> (const TSurface *dst, int x, int y, u32 clr) <i>Plot a single pixel on a 16-bit buffer.</i>
void	<b>sbmp16_hline</b> (const TSurface *dst, int x1, int y, int x2, u32 clr) <i>Draw a horizontal line on an 16bit buffer.</i>
void	<b>sbmp16_vline</b> (const TSurface *dst, int x, int y1, int y2, u32 clr) <i>Draw a vertical line on an 16bit buffer.</i>
void	<b>sbmp16_line</b> (const TSurface *dst, int x1, int y1, int x2, int y2, u32 clr) <i>Draw a line on an 16bit buffer.</i>
void	<b>sbmp16_rect</b> (const TSurface *dst, int left, int top, int right, int bottom, u32 clr) <i>Draw a rectangle in 16bit mode.</i>
void	<b>sbmp16_frame</b> (const TSurface *dst, int left, int top, int right, int bottom, u32 clr)

		<i>Draw a rectangle in 16bit mode.</i>
void	<b>sbmp16.blit</b> (const TSurface *dst, int dstX, int dstY, uint width, uint height, const TSurface *src, int srcX, int srcY)	<i>16bpp blitter. Copies a rectangle from one surface to another.</i>
void	<b>sbmp16.floodfill</b> (const TSurface *dst, int x, int y, u32 clr)	<i>Floodfill an area of the same color with new color clr.</i>
INLINE void	<b>_sbmp16_plot</b> (const TSurface *dst, int x, int y, u32 clr)	<i>Plot a single pixel on a 16-bit buffer; inline version.</i>
INLINE u32	<b>_sbmp16_get_pixel</b> (const TSurface *src, int x, int y)	<i>Get the pixel value of src at (x, y); inline version.</i>
u32	<b>sbmp8.get_pixel</b> (const TSurface *src, int x, int y)	<i>Get the pixel value of src at (x, y).</i>
void	<b>sbmp8.plot</b> (const TSurface *dst, int x, int y, u32 clr)	<i>Plot a single pixel on a 8-bit buffer.</i>
void	<b>sbmp8.hline</b> (const TSurface *dst, int x1, int y, int x2, u32 clr)	<i>Draw a horizontal line on an 8-bit buffer.</i>
void	<b>sbmp8.vline</b> (const TSurface *dst, int x, int y1, int y2, u32 clr)	<i>Draw a vertical line on an 8-bit buffer.</i>
void	<b>sbmp8.line</b> (const TSurface *dst, int x1, int y1, int x2, int y2, u32 clr)	<i>Draw a line on an 8-bit buffer.</i>
void	<b>sbmp8.rect</b> (const TSurface *dst, int left, int top, int right, int bottom, u32 clr)	<i>Draw a rectangle in 8-bit mode.</i>
void	<b>sbmp8.frame</b> (const TSurface *dst, int left, int top, int right, int bottom, u32 clr)	<i>Draw a rectangle in 8-bit mode.</i>
void	<b>sbmp8.blit</b> (const TSurface *dst, int dstX, int dstY, uint width, uint height, const TSurface *src, int srcX, int srcY)	<i>16bpp blitter. Copies a rectangle from one surface to another.</i>
void	<b>sbmp8.floodfill</b> (const TSurface *dst, int x, int y, u32 clr)	<i>Floodfill an area of the same color with new color clr.</i>
INLINE void	<b>_sbmp8_plot</b> (const TSurface *dst, int x, int y, u32 clr)	<i>Plot a single pixel on a 8-bit surface; inline version.</i>
INLINE u32	<b>_sbmp8_get_pixel</b> (const TSurface *src, int x, int y)	

		<i>Get the pixel value of src at (x, y); inline version.</i>
u32	<b>schr4c_get_pixel</b> (const TSurface *src, int x, int y)	<i>Get the pixel value of src at (x, y).</i>
void	<b>schr4c_plot</b> (const TSurface *dst, int x, int y, u32 clr)	<i>Plot a single pixel on a 4bpp tiled surface.</i>
void	<b>schr4c_hline</b> (const TSurface *dst, int x1, int y, int x2, u32 clr)	<i>Draw a horizontal line on a 4bpp tiled surface.</i>
void	<b>schr4c_vline</b> (const TSurface *dst, int x, int y1, int y2, u32 clr)	<i>Draw a vertical line on a 4bpp tiled surface.</i>
void	<b>schr4c_line</b> (const TSurface *dst, int x1, int y1, int x2, int y2, u32 clr)	<i>Draw a line on a 4bpp tiled surface.</i>
void	<b>schr4c_rect</b> (const TSurface *dst, int left, int top, int right, int bottom, u32 clr)	<i>Render a rectangle on a 4bpp tiled canvas.</i>
void	<b>schr4c_frame</b> (const TSurface *dst, int left, int top, int right, int bottom, u32 clr)	<i>Draw a rectangle on a 4bpp tiled surface.</i>
void	<b>schr4c.blit</b> (const TSurface *dst, int dstX, int dstY, uint width, uint height, const TSurface *src, int srcX, int srcY)	<i>Blitter for 4bpp tiled surfaces. Copies a rectangle from one surface to another.</i>
void	<b>schr4c_floodfill</b> (const TSurface *dst, int x, int y, u32 clr)	<i>Floodfill an area of the same color with new color clr.</i>
void	<b>schr4c_prep_map</b> (const TSurface *srf, u16 *map, u16 se0)	<i>Prepare a screen-entry map for use with chr4.</i>
u32 *	<b>schr4c_get_ptr</b> (const TSurface *srf, int x, int y)	<i>Special pointer getter for chr4: start of in-tile line.</i>
INLINE void	<b>_schr4c_plot</b> (const TSurface *dst, int x, int y, u32 clr)	<i>Plot a single pixel on a 4bpp tiled,col-jamor surface; inline version.</i>
INLINE u32	<b>_schr4c_get_pixel</b> (const TSurface *src, int x, int y)	<i>Get the pixel value of src at (x, y); inline version.</i>
u32	<b>schr4r_get_pixel</b> (const TSurface *src, int x, int y)	<i>Get the pixel value of src at (x, y).</i>

void	<b>schr4r_plot</b> (const TSurface *dst, int x, int y, u32 clr) <i>Plot a single pixel on a 4bpp tiled surface.</i>
void	<b>schr4r_hline</b> (const TSurface *dst, int x1, int y, int x2, u32 clr) <i>Draw a horizontal line on a 4bpp tiled surface.</i>
void	<b>schr4r_vline</b> (const TSurface *dst, int x, int y1, int y2, u32 clr) <i>Draw a vertical line on a 4bpp tiled surface.</i>
void	<b>schr4r_line</b> (const TSurface *dst, int x1, int y1, int x2, int y2, u32 clr) <i>Draw a line on a 4bpp tiled surface.</i>
void	<b>schr4r_rect</b> (const TSurface *dst, int left, int top, int right, int bottom, u32 clr) <i>Render a rectangle on a tiled canvas.</i>
void	<b>schr4r_frame</b> (const TSurface *dst, int left, int top, int right, int bottom, u32 clr) <i>Draw a rectangle on a 4bpp tiled surface.</i>
void	<b>schr4r_prep_map</b> (const TSurface *srf, u16 *map, u16 se0) <i>Prepare a screen-entry map for use with chr4.</i>
u32 *	<b>schr4r_get_ptr</b> (const TSurface *srf, int x, int y) <i>Special pointer getter for chr4: start of in-tile line.</i>
INLINE void	<b>_schr4r_plot</b> (const TSurface *dst, int x, int y, u32 clr) <i>Plot a single pixel on a 4bpp tiled, row-major surface; inline version.</i>
INLINE u32	<b>_schr4r_get_pixel</b> (const TSurface *src, int x, int y) <i>Get the pixel value of src at (x, y); inline version.</i>

## Variables

const TSurface	<b>m3_surface</b>
EWRAM_DATA TSurface	<b>m4_surface</b>
EWRAM_DATA TSurface	<b>m5_surface</b>
const TSurfaceProcTab	<b>bmp16_tab</b>
const TSurfaceProcTab	<b>bmp8_tab</b>
const TSurfaceProcTab	<b>chr4c_tab</b>

---

# Detailed Description

**Author:**

J Vijn

**Date:**

20080119 - 20080514

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tonc\_text.h** File Reference

```
#include "tonc_memmap.h" #include "tonc_memdef.h"  
#include "tonc_core.h"
```

## Defines

```
#define toncfontTilesLen 768
```

# Functions

void	<b>txt_init_std</b> ()
void	<b>txt_bup_1toX</b> (void *dstv, const void *srcv, u32 len, int bpp, u32 base)
void	<b>txt_init_se</b> (int bgnr, u16 bgcnt, SCR_ENTRY se0, u32 clrs, u32 base)
void	<b>se_putc</b> (int x, int y, int c, SCR_ENTRY se0)
void	<b>se_puts</b> (int x, int y, const char *str, SCR_ENTRY se0)
void	<b>se_clrs</b> (int x, int y, const char *str, SCR_ENTRY se0)
INLINE void	<b>obj_putc2</b> (int x, int y, int c, u16 attr2, <b>OBJ_ATTR</b> *obj0) <i>Write character c to (x, y) in color clr using objects obj0 and on.</i>
INLINE void	<b>obj_puts2</b> (int x, int y, const char *str, u16 attr2, <b>OBJ_ATTR</b> *obj0) <i>Write string str to (x, y) in color clr using objects obj0 and on.</i>
void	<b>txt_init_obj</b> ( <b>OBJ_ATTR</b> *obj0, u16 attr2, u32 clrs, u32 base)
void	<b>obj_putc</b> (int x, int y, int c, u16 attr2)
void	<b>obj_puts</b> (int x, int y, const char *str, u16 attr2)
void	<b>obj_clrs</b> (int x, int y, const char *str)

## Mode-independent functions

void	<b>bm_putc</b> (int x, int y, int c, <b>COLOR</b> clr)
void	<b>bm_puts</b> (int x, int y, const char *str, <b>COLOR</b> clr)
void	<b>bm_clrs</b> (int x, int y, const char *str, <b>COLOR</b> clr)

## Mode 3 functions

INLINE void	<b>m3_putc</b> (int x, int y, int c, <b>COLOR</b> clr) <i>Write character c to (x, y) in color clr in mode 3.</i>
INLINE void	<b>m3_puts</b> (int x, int y, const char *str, <b>COLOR</b> clr) <i>Write string str to (x, y) in color clr in mode 3.</i>
INLINE void	<b>m3_clrs</b> (int x, int y, const char *str, <b>COLOR</b> clr) <i>Clear the space used by string str at (x, y) in color clr in mode 3.</i>

## Mode 4 functions

INLINE void	<b>m4_putc</b> (int x, int y, int c, u8 clrid)
-------------	--

	<i>Write character c to (x, y) in color-index clrid in mode 4.</i>
INLINE void	<b>m4_puts</b> (int x, int y, const char *str, u8 clrid) <i>Write string str to (x, y) in color-index clrid in mode 4.</i>
INLINE void	<b>m4_clrs</b> (int x, int y, const char *str, u8 clrid) <i>Clear the space used by string str at (x, y) in color-index clrid in mode 4.</i>

## Mode 5 functions

INLINE void	<b>m5_putc</b> (int x, int y, int c, <b>COLOR</b> clr) <i>Write character c to (x, y) in color clr in mode 5.</i>
INLINE void	<b>m5_puts</b> (int x, int y, const char *str, <b>COLOR</b> clr) <i>Write string str to (x, y) in color clr in mode 5.</i>
INLINE void	<b>m5_clrs</b> (int x, int y, const char *str, <b>COLOR</b> clr) <i>Clear the space used by string str at (x, y) in color clr in mode 5.</i>

## Mode 5 functions

void	<b>bm16_putc</b> (u16 *dst, int c, <b>COLOR</b> clr, int pitch)
void	<b>bm16_puts</b> (u16 *dst, const char *str, <b>COLOR</b> clr, int pitch)
void	<b>bm16_clrs</b> (u16 *dst, const char *str, <b>COLOR</b> clr, int pitch)
void	<b>bm8_putc</b> (u16 *dst, int c, u8 clrid)
void	<b>bm8_puts</b> (u16 *dst, const char *str, u8 clrid)

## Variables

const u32	<b>toncfontTiles</b> [192]
TXT_BASE	<u><b>_txt_base</b></u>
TXT_BASE *	<b>gptxt</b>
u8	<b>txt_lut</b> [256]
u16 *	<b>vid_page</b>

---

# Detailed Description

**Author:**

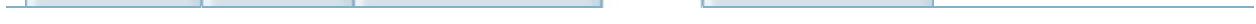
J Vijn

**Date:**

20060605 - 20060605

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tonc\_tte.h File Reference**

```
#include <stdio.h> #include "tonc_memmap.h"
#include "tonc_surface.h"
```

# Data Structures

struct	<b>TFont</b> <i>Font description struct.</i> <a href="#">More...</a>
struct	<b>TTC</b> <i>TTE context struct.</i> <a href="#">More...</a>

## Defines

```
#define TTE_TAB_WIDTH 24  
#define tte_printf iprintf
```

### Color lut indices

```
#define TTE_INK 0  
#define TTE_SHADOW 1  
#define TTE_PAPER 2  
#define TTE_SPECIAL 3
```

### drawg helper macros

```
#define TTE_BASE_VARS(_tc, _font)  
    Declare and define base drawg variables.  
#define TTE_CHAR_VARS(font, gid, src_t, _sD, _sL, _chW, _chH)  
    Declare and define basic source drawg variables.  
#define TTE_DST_VARS(tc, dst_t, _dD, _dL, _dP, _x, _y)  
    Declare and define basic destination drawg variables.
```

### Default fonts

```
#define fwf_default sys8Font  
    Default fixed-width font.  
#define vwf_default verdana9Font  
    Default variable-width font.
```

### Default glyph renderers

```
#define ase_drawg_default ((fnDrawg)ase_drawg_s)  
#define bmp8_drawg_default ((fnDrawg)bmp8_drawg_b1cts)  
#define bmp16_drawg_default ((fnDrawg)bmp16_drawg_b1cts)  
#define chr4c_drawg_default ((fnDrawg)chr4c_drawg_b1cts)  
#define chr4r_drawg_default ((fnDrawg)chr4r_drawg_b1cts)  
#define obj_drawg_default ((fnDrawg)obj_drawg)  
#define se_drawg_default ((fnDrawg)se_drawg_s)
```

### Default initializers

#define	<b>tte_init_se_default</b> (bgnr, bgcnt) tte_init_se( bgnr, bgcnt, 0xF000, CLR_YELLOW, 0, &fwf_default, NULL)
#define	<b>tte_init_ase_default</b> (bgnr, bgcnt) tte_init_ase(bgnr, bgcnt, 0x0000, CLR_YELLOW, 0, &fwf_default, NULL)
#define	<b>tte_init_chr4c_default</b> (bgnr, bgcnt)
#define	<b>tte_init_chr4r_default</b> (bgnr, bgcnt)
#define	<b>tte_init_chr4c_b4_default</b> (bgnr, bgcnt)
#define	<b>tte_init bmp default</b> (mode) tte_init bmp(mode, &vWF_default, NULL)
#define	<b>tte_init_obj_default</b> (pObj) tte_init_obj(pObj, 0, 0, 0xF000, CLR_YELLOW, 0, &fwf_default, NULL)

## Typedefs

typedef void(*	<b>fnDrawg</b> )(uint gid) <i>Glyph render function format.</i>
typedef void(*	<b>fnErase</b> )(int left, int top, int right, int bottom) <i>Erase rectangle function format.</i>

# Functions

void	<b>tte_set_context</b> (TTC *tc) <i>Set the master context pointer.</i>
INLINE TTC *	<b>tte_get_context</b> (void) <i>Get the master text-system.</i>
INLINE uint	<b>tte_get_glyph_id</b> (int ch) <i>Get the glyph index of character ch.</i>
INLINE int	<b>tte_get_glyph_width</b> (uint gid) <i>Get the width of glyph id.</i>
INLINE int	<b>tte_get_glyph_height</b> (uint gid) <i>Get the height of glyph id.</i>
INLINE const void *	<b>tte_get_glyph_data</b> (uint gid) <i>Get the glyph data of glyph id.</i>
void	<b>tte_set_color</b> (eint type, u16 clr) <i>Set color attribute of type to cattr.</i>
void	<b>tte_set_colors</b> (const u16 colors[]) <i>Load important color data.</i>
void	<b>tte_set_color_attr</b> (eint type, u16 cattr) <i>Set color attribute of type to cattr.</i>
void	<b>tte_set_color_attrs</b> (const u16 cattrs[]) <i>Load important color attribute data.</i>
char *	<b>tte_cmd_default</b> (const char *str) <i>Text command handler.</i>
int	<b>tte_putc</b> (int ch) <i>Plot a single character; does wrapping too.</i>
int	<b>tte_write</b> (const char *text) <i>Render a string.</i>
int	<b>tte_write_ex</b> (int x, int y, const char *text, const u16 *clrlut) <i>Extended string writer, with positional and color info.</i>
void	<b>tte_erase_rect</b> (int left, int top, int right, int bottom) <i>Erase a portion of the screen (ignores margins).</i>

	void	<b>tte_erase_screen</b> (void) <i>Erase the screen (within the margins).</i>
	void	<b>tte_erase_line</b> (void) <i>Erase the whole line (within the margins).</i>
POINT16		<b>tte_get_text_size</b> (const char *str) <i>Get the size taken up by a string.</i>
	void	<b>tte_init_base</b> (const <b>TFont</b> *font, <b>fnDrawg</b> drawProc, <b>fnErase</b> eraseProc) <i>Base initializer of a <b>TTC</b>.</i>
INLINE void		<b>tte_get_pos</b> (int *x, int *y) <i>Get cursor position.</i>
INLINE u16		<b>tte_get_ink</b> (void) <i>Get ink color attribute.</i>
INLINE u16		<b>tte_get_shadow</b> (void) <i>Get shadow color attribute.</i>
INLINE u16		<b>tte_get_paper</b> (void) <i>Get paper color attribute.</i>
INLINE u16		<b>tte_get_special</b> (void) <i>Get special color attribute.</i>
INLINE <b>TSurface</b> *		<b>tte_get_surface</b> () <i>Get a pointer to the text surface.</i>
INLINE <b>TFont</b> *		<b>tte_get_font</b> (void) <i>Get the active font.</i>
INLINE <b>fnDrawg</b>		<b>tte_get_drawg</b> (void) <i>Get the active character plotter.</i>
INLINE <b>fnErase</b>		<b>tte_get_erase</b> (void) <i>Get the character plotter.</i>
INLINE char **		<b>tte_get_string_table</b> (void) <i>Get string table.</i>
INLINE <b>TFont</b> **		<b>tte_get_font_table</b> (void) <i>Get font table.</i>
INLINE void		<b>tte_set_pos</b> (int x, int y) <i>Set cursor position.</i>
INLINE void		<b>tte_set_ink</b> (u16 cattr) <i>Set ink color attribute.</i>

INLINE void	<b>tte_set_shadow</b> (u16 cattr)  Set shadow color attribute.
INLINE void	<b>tte_set_paper</b> (u16 cattr) Set paper color attribute.
INLINE void	<b>tte_set_special</b> (u16 cattr) Set special color attribute.
INLINE void	<b>tte_set_surface</b> (const TSurface *srf) Set the text surface.
INLINE void	<b>tte_set_font</b> (const <b>TFont</b> *font) Set the font.
INLINE void	<b>tte_set_drawg</b> ( <b>fnDrawg</b> proc) Set the character plotter.
INLINE void	<b>tte_set_erase</b> ( <b>fnErase</b> proc) Set the character plotter.
INLINE void	<b>tte_set_string_table</b> (const char *table[]) Set string table.
INLINE void	<b>tte_set_font_table</b> (const <b>TFont</b> *table[]) Set font table.
void	<b>tte_set_margins</b> (int left, int top, int right, int bottom)
void	<b>tte_init_con</b> (void) Init stdio capabilities.
int	<b>tte_cmd_vt100</b> (const char *text) Parse for VT100-sequences.
int	<b>tte_con_write</b> (struct _reent *r, int fd, const char *text, int len) Internal routine for stdio functionality.
int	<b>tte_con_nocash</b> (struct _reent *r, int fd, const char *text, int len)
void	<b>tte_init_bmp</b> (int vmode, const <b>TFont</b> *font, <b>fnDrawg</b> proc) Initialize text system for bitmap fonts.
void	<b>tte_init_obj</b> ( <b>OBJ_ATTR</b> *dst, u32 attr0, u32 attr1, u32 attr2, u32 clrs, u32 bupofs, const <b>TFont</b> *font, <b>fnDrawg</b> proc) Initialize text system for screen-entry fonts.

	void	<b>obj_erase</b> (int left, int top, int right, int bottom) <i>Unwind the object text-buffer.</i>
	void	<b>obj_drawg</b> (uint gid) <i>Character-plot for objects.</i>

## Regular tilemaps

	void	<b>tte_init_se</b> (int bgnr, u16 bgcnt, SCR_ENTRY se0, u32 clrs, u32 bupofs, const <b>TFont</b> *font, <b>fnDrawg</b> proc) <i>Initialize text system for screen-entry fonts.</i>
	void	<b>se_erase</b> (int left, int top, int right, int bottom) <i>Erase part of the regular tilemap canvas.</i>
	void	<b>se_drawg_w8h8</b> (uint gid) <i>Character-plot for reg BGs using an 8x8 font.</i>
	void	<b>se_drawg_w8h16</b> (uint gid) <i>Character-plot for reg BGs using an 8x16 font.</i>
	void	<b>se_drawg</b> (uint gid) <i>Character-plot for reg BGs, any sized font.</i>
	void	<b>se_drawg_s</b> (uint gid) <i>Character-plot for reg BGs, any sized, vertically tiled font.</i>

## Affine tilemaps

	void	<b>tte_init_ase</b> (int bgnr, u16 bgcnt, u8 ase0, u32 clrs, u32 bupofs, const <b>TFont</b> *font, <b>fnDrawg</b> proc) <i>Initialize text system for affine screen-entry fonts.</i>
	void	<b>ase_erase</b> (int left, int top, int right, int bottom) <i>Erase part of the affine tilemap canvas.</i>
	void	<b>ase_drawg_w8h8</b> (uint gid) <i>Character-plot for affine BGs using an 8x8 font.</i>
	void	<b>ase_drawg_w8h16</b> (uint gid) <i>Character-plot for affine BGs using an 8x16 font.</i>
	void	<b>ase_drawg</b> (uint gid) <i>Character-plot for affine Bgs, any size.</i>
	void	<b>ase_drawg_s</b> (uint gid) <i>Character-plot for affine BGs, any sized, vertically</i>

*oriented font.*

## 4bpp tiles

	void	<b>tte_init_chr4c</b> (int bgnr, u16 bgcnt, u16 se0, u32 cattrs, u32 clrs, const <b>TFont</b> *font, <b>fnDrawg</b> proc)	<i>Initialize text system for 4bpp tiled, column-major surfaces.</i>
	void	<b>chr4c_erase</b> (int left, int top, int right, int bottom)	<i>Erase part of the 4bpp text canvas.</i>
	void	<b>chr4c_drawg_b1cts</b> (uint gid)	<i>Render 1bpp fonts to 4bpp tiles.</i>
IWRAM_CODE	void	<b>chr4c_drawg_b1cts_fast</b> (uint gid)	<i>Initialize text system for 4bpp tiled, column-major surfaces.</i>
	void	<b>chr4c_drawg_b4cts</b> (uint gid)	<i>Initialize text system for 4bpp tiled, column-major surfaces.</i>
IWRAM_CODE	void	<b>chr4c_drawg_b4cts_fast</b> (uint gid)	<i>Initialize text system for 4bpp tiled, column-major surfaces.</i>

## 4bpp tiles

	void	<b>tte_init_chr4r</b> (int bgnr, u16 bgcnt, u16 se0, u32 cattrs, u32 clrs, const <b>TFont</b> *font, <b>fnDrawg</b> proc)	<i>Initialize text system for 4bpp tiled, column-major surfaces.</i>
	void	<b>chr4r_erase</b> (int left, int top, int right, int bottom)	<i>Erase part of the 4bpp text canvas.</i>
	void	<b>chr4r_drawg_b1cts</b> (uint gid)	<i>Render 1bpp fonts to 4bpp tiles.</i>
IWRAM_CODE	void	<b>chr4r_drawg_b1cts_fast</b> (uint gid)	<i>Initialize text system for 4bpp tiled, column-major surfaces.</i>

## 8bpp bitmaps

	void	<b>bmp8_erase</b> (int left, int top, int right, int bottom)	<i>Erase part of the 8bpp text canvas.</i>
--	------	--	--

	void	<b>bmp8_drawg</b> (uint gid) <i>Linear 8 bpp bitmap glyph renderer, opaque.</i>
	void	<b>bmp8_drawg_t</b> (uint gid) <i>Linear 8 bpp bitmap glyph renderer, transparent.</i>
	void	<b>bmp8_drawg_b1cts</b> (uint gid) <i>Erase part of the 8bpp text canvas.</i>
IWRAM_CODE	void	<b>bmp8_drawg_b1cts_fast</b> (uint gid) <i>Erase part of the 8bpp text canvas.</i>
	void	<b>bmp8_drawg_b1cos</b> (uint gid) <i>Erase part of the 8bpp text canvas.</i>

## 16bpp bitmaps

	void	<b>bmp16_erase</b> (int left, int top, int right, int bottom) <i>Erase part of the 16bpp text canvas.</i>
	void	<b>bmp16_drawg</b> (uint gid) <i>Linear 16bpp bitmap glyph renderer, opaque.</i>
	void	<b>bmp16_drawg_t</b> (uint gid) <i>Linear 16bpp bitmap glyph renderer, transparent.</i>
	void	<b>bmp16_drawg_b1cts</b> (uint gid) <i>Linear bitmap, 16bpp transparent character plotter.</i>
	void	<b>bmp16_drawg_b1cos</b> (uint gid) <i>Linear bitmap, 16bpp opaque character plotter.</i>

# Variables

TTC *	gp_tte_context
<b>Internal fonts</b>	
const <b>TFont</b>	<b>sys8Font</b> <i>System font '-127. FWF 8x 8@1.</i>
const <b>TFont</b>	<b>verdana9Font</b> <i>Verdana 9 '-'-'?'. VWF 8x12@1.</i>
const <b>TFont</b>	<b>verdana9bFont</b> <i>Verdana 9 bold '-'-'?'. VWF 8x12@1.</i>
const <b>TFont</b>	<b>verdana9iFont</b> <i>Verdana 9 italic '-'-'?'. VWF 8x12@1.</i>
const <b>TFont</b>	<b>verdana10Font</b> <i>Verdana 10 '-'-'?'. VWF 16x14@1.</i>
const <b>TFont</b>	<b>verdana9_b4Font</b> <i>Verdana 9 '-'-'?'. VWF 8x12@4.</i>
const unsigned int	<b>sys8Glyphs</b> [192] <i>System font '-127. FWF 8x 8@1.</i>
const unsigned int	<b>verdana9Glyphs</b> [896] <i>System font '-127. FWF 8x 8@1.</i>
const unsigned char	<b>verdana9Widths</b> [224] <i>System font '-127. FWF 8x 8@1.</i>
const unsigned int	<b>verdana9bGlyphs</b> [896] <i>System font '-127. FWF 8x 8@1.</i>
const unsigned char	<b>verdana9bWidths</b> [224] <i>System font '-127. FWF 8x 8@1.</i>
const unsigned int	<b>verdana9iGlyphs</b> [896] <i>System font '-127. FWF 8x 8@1.</i>
const unsigned char	<b>verdana9iWidths</b> [224] <i>System font '-127. FWF 8x 8@1.</i>
const unsigned int	<b>verdana10Glyphs</b> [1792] <i>System font '-127. FWF 8x 8@1.</i>
const unsigned char	<b>verdana10Widths</b> [224]

	<i>System font '-127. FWF 8x 8@1.</i>
const unsigned int	<b>verdana9_b4Glyphs</b> [3584] <i>System font '-127. FWF 8x 8@1.</i>
const unsigned char	<b>verdana9_b4Widths</b> [224] <i>System font '-127. FWF 8x 8@1.</i>

---

# Detailed Description

**Author:**

J Vijn

**Date:**

20070517 - 20080503

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tonc\_types.h File Reference**

# Data Structures

struct	<b>BLOCK</b> <i>8-word type for fast struct-copies</i> <a href="#">More...</a>
struct	<b>TILE</b> <i>4bpp tile type, for easy indexing and copying of 4-bit tiles</i> <a href="#">More...</a>
struct	<b>TILE8</b> <i>8bpp tile type, for easy indexing and 8-bit tiles</i> <a href="#">More...</a>
struct	<b>ObjAffineSource</b> <i>Simple scale-rotation source struct.</i> <a href="#">More...</a>
struct	<b>BgAffineSource</b> <i>Extended scale-rotate source struct.</i> <a href="#">More...</a>
struct	<b>ObjAffineDest</b> <i>Simple scale-rotation destination struct, BG version.</i> <a href="#">More...</a>
struct	<b>BgAffineDest</b> <i>Extended scale-rotate destination struct.</i> <a href="#">More...</a>
struct	<b>BG_POINT</b> <i>Regular bg points; range: :0010 - :001F.</i> <a href="#">More...</a>
struct	<b>DMA_REC</b> <i>DMA struct; range: 0400:00B0 - 0400:00DF.</i> <a href="#">More...</a>
struct	<b>TMR_REC</b> <i>Timer struct, range: 0400:0100 - 0400:010F.</i> <a href="#">More...</a>
struct	<b>OBJ_ATTR</b> <i>Object attributes.</i> <a href="#">More...</a>
struct	<b>OBJ_AFFINE</b> <i>Object affine parameters.</i> <a href="#">More...</a>

## Defines

#define	<b>IWRAM_DATA</b> __attribute__((section(".iwrام")))) <i>Put variable in IWRAM (default).</i>
#define	<b>EWRAM_DATA</b> __attribute__((section(".ewram")))) <i>Put variable in EWRAM.</i>
#define	<b>EWRAM_BSS</b> __attribute__((section(".sbss")))) <i>Put non-initialized variable in EWRAM.</i>
#define	<b>IWRAM_CODE</b> __attribute__((section(".iwrام"), long_call)) <i>Put function in IWRAM.</i>
#define	<b>EWRAM_CODE</b> __attribute__((section(".ewram"), long_call)) <i>Put function in EWRAM.</i>
#define	<b>ALIGN(n)</b> __attribute__((aligned(n))) <i>Force a variable to an n-byte boundary.</i>
#define	<b>ALIGN4</b> __attribute__((aligned(4))) <i>Force word alignment.</i>
#define	<b>PACKED</b> __attribute__((packed)) <i>Pack aggregate members.</i>
#define	<b>DEPRECATED</b> __attribute__((deprecated)) <i>Deprecated notice.</i>
#define	<b>INLINE</b> static inline <i>Inline function declarator.</i>
#define	<b>TRUE</b> 1
#define	<b>FALSE</b> 0
#define	<b>NULL</b> (void*)0

## Typedefs

<code>typedef const char *const</code>	<b>CSTR</b> <i>Type for consting a string as well as the pointer than points to it.</i>
<code>typedef s32</code>	<b>FIXED</b> <i>Fixed point type.</i>
<code>typedef u16</code>	<b>COLOR</b> <i>Type for colors.</i>
<code>typedef u16</code>	<b>SCR_ENTRY</b>
<code>typedef u16</code>	<b>SE</b> <i>Type for screen entries.</i>
<code>typedef u8</code>	<b>SCR_AFF_ENTRY</b>
<code>typedef u8</code>	<b>SAE</b> <i>Type for affine screen entries.</i>
<code>typedef struct TILE</code>	<b>TILE4</b>
<code>typedef u8</code>	<b>BOOL</b>
<code>typedef void(*)</code>	<b>fnptr</b> <i>(void)</i> <i>void foo() function pointer</i>
<code>typedef void(*)</code>	<b>fn_v_i</b> <i>(int)</i> <i>void foo(int x) function pointer</i>
<code>typedef int(*)</code>	<b>fn_i_i</b> <i>(int)</i> <i>int foo(int x) function pointer</i>

## Base types

*Basic signed and unsigned types for 8, 16, 32 and 64-bit integers.*

- *s# : signed #-bit integer.*
- *u#/u{type} : unsigned #-bit integer.*
- *e{type} : enum'ed #-bit integer.*

<code>typedef unsigned char</code>	<b>u8</b>
<code>typedef unsigned char</code>	<b>byte</b>
<code>typedef unsigned char</code>	<b>uchar</b>

typedef unsigned char	<b>echar</b>
typedef unsigned short	<b>u16</b>
typedef unsigned short	<b>hword</b>
typedef unsigned short	<b>ushort</b>
typedef unsigned short	<b>eshort</b>
typedef unsigned int	<b>u32</b>
typedef unsigned int	<b>word</b>
typedef unsigned int	<b>uint</b>
typedef unsigned int	<b>eint</b>
typedef unsigned long long	<b>u64</b>
typedef signed char	<b>s8</b>
typedef signed short	<b>s16</b>
typedef signed int	<b>s32</b>
typedef signed long long	<b>s64</b>

## Volatile types

*Volatile types for registers*

typedef volatile u8	<b>vu8</b>
typedef volatile u16	<b>vu16</b>
typedef volatile u32	<b>vu32</b>
typedef volatile u64	<b>vu64</b>
typedef volatile s8	<b>vs8</b>
typedef volatile s16	<b>vs16</b>
typedef volatile s32	<b>vs32</b>
typedef volatile s64	<b>vs64</b>

## Const types

*Const types for const function parameters*

typedef const u8	<b>cu8</b>
typedef const u16	<b>cu16</b>
typedef const u32	<b>cu32</b>
typedef const u64	<b>cu64</b>
typedef const s8	<b>cs8</b>

typedef const s16	<b>cs16</b>
typedef const s32	<b>cs32</b>
typedef const s64	<b>cs64</b>

## IO register types

typedef struct AFF_DST_EX	<b>BG_AFFINE</b> <i>Affine parameters for backgrounds; range : 0400:0020 - 0400:003F.</i>
---------------------------	--

## PAL types

typedef <b>COLOR</b>	<b>PALBANK</b> [16] <i>Palette bank type, for 16-color palette banks.</i>
----------------------	--

## VRAM array types

*These types allow VRAM access as arrays or matrices in their most natural types.*

typedef SCR_ENTRY	<b>SCREENLINE</b> [32]
typedef SCR_ENTRY	<b>SCREENMAT</b> [32][32]
typedef SCR_ENTRY	<b>SCREENBLOCK</b> [1024]
typedef <b>COLOR</b>	<b>M3LINE</b> [240]
typedef u8	<b>M4LINE</b> [240]
typedef <b>COLOR</b>	<b>M5LINE</b> [160]
typedef <b>TILE</b>	<b>CHARBLOCK</b> [512]
typedef <b>TILE8</b>	<b>CHARBLOCK8</b> [256]

## Enumerations

enum	<b>bool</b> { <b>false</b> , <b>true</b> }
------	--

*Boolean type.*

---

# Detailed Description

**Author:**

J Vijn

**Date:**

20060508 - 20080111

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tonc\_video.c File Reference**

```
#include "tonc_memmap.h" #include "tonc_core.h"  
#include "tonc_video.h"
```

## Functions

<b>COLOR *</b>	<b>vid_flip</b> (void)
----------------	------------------------

*Flip the display page.*

---

## **Detailed Description**

**Author:**

J Vijn

**Date:**

20060604 - 20070805

---

# Function Documentation

**COLOR\* vid\_flip ( void )**

Flip the display page.

Toggles the display page in REG\_DISPCNT and sets *vid\_page* to point to the back buffer.

**Returns:**

Current back buffer pointer.

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tonc\_video.h File Reference**

```
#include "tonc_memmap.h" #include "tonc_memdef.h"
#include "tonc_core.h"
```

## mode 3

#define	<b>M3_CLEAR()</b> <code>memset32(vid_mem, 0, M3_SIZE/4)</code> <i>Fill the mode 3 background with color clr.</i>
INLINE void	<b>m3_fill (COLOR clr)</b> <i>Fill the mode 3 background with color clr.</i>
INLINE void	<b>m3_plot (int x, int y, COLOR clr)</b> <i>Plot a single clr colored pixel in mode 3 at (x, y).</i>
INLINE void	<b>m3_hline (int x1, int y, int x2, COLOR clr)</b> <i>Draw a clr colored horizontal line in mode 3.</i>
INLINE void	<b>m3_vline (int x, int y1, int y2, COLOR clr)</b> <i>Draw a clr colored vertical line in mode 3.</i>
INLINE void	<b>m3_line (int x1, int y1, int x2, int y2, COLOR clr)</b> <i>Draw a clr colored line in mode 3.</i>
INLINE void	<b>m3_rect (int left, int top, int right, int bottom, COLOR clr)</b> <i>Draw a clr colored rectangle in mode 3.</i>
INLINE void	<b>m3_frame (int left, int top, int right, int bottom, COLOR clr)</b> <i>Draw a clr colored frame in mode 3.</i>

## mode 4

#define	<b>M4_CLEAR()</b> <code>memset32(vid_page, 0, M4_SIZE/4)</code> <i>Fill the current mode 4 backbuffer with clrid.</i>
INLINE void	<b>m4_fill</b> (u8 clrid) <i>Fill the current mode 4 backbuffer with clrid.</i>
INLINE void	<b>m4_plot</b> (int x, int y, u8 clrid) <i>Plot a clrid pixel on the current mode 4 backbuffer.</i>
INLINE void	<b>m4_hline</b> (int x1, int y, int x2, u8 clrid) <i>Draw a clrid colored horizontal line in mode 4.</i>
INLINE void	<b>m4_vline</b> (int x, int y1, int y2, u8 clrid) <i>Draw a clrid colored vertical line in mode 4.</i>
INLINE void	<b>m4_line</b> (int x1, int y1, int x2, int y2, u8 clrid) <i>Draw a clrid colored line in mode 4.</i>
INLINE void	<b>m4_rect</b> (int left, int top, int right, int bottom, u8 clrid) <i>Draw a clrid colored rectangle in mode 4.</i>
INLINE void	<b>m4_frame</b> (int left, int top, int right, int bottom, u8 clrid) <i>Draw a clrid colored frame in mode 4.</i>

## mode 5

#define	<b>M5_CLEAR()</b> <code>memset32(vid_page, 0, M5_SIZE/4)</code> <i>Fill the current mode 5 backbuffer with clr.</i>
INLINE void	<b>m5_fill (COLOR clr)</b> <i>Fill the current mode 5 backbuffer with clr.</i>
INLINE void	<b>m5_plot (int x, int y, COLOR clr)</b> <i>Plot a clr'd pixel on the current mode 5 backbuffer.</i>
INLINE void	<b>m5_hline (int x1, int y, int x2, COLOR clr)</b> <i>Draw a clr colored horizontal line in mode 5.</i>
INLINE void	<b>m5_vline (int x, int y1, int y2, COLOR clr)</b> <i>Draw a clr colored vertical line in mode 5.</i>
INLINE void	<b>m5_line (int x1, int y1, int x2, int y2, COLOR clr)</b> <i>Draw a clr colored line in mode 5.</i>
INLINE void	<b>m5_rect (int left, int top, int right, int bottom, COLOR clr)</b> <i>Draw a clr colored rectangle in mode 5.</i>
INLINE void	<b>m5_frame (int left, int top, int right, int bottom, COLOR clr)</b> <i>Draw a clr colored frame in mode 5.</i>

## Defines

```
#define SCREEN_WIDTH 240
#define SCREEN_HEIGHT 160
#define M3_WIDTH SCREEN_WIDTH
#define M3_HEIGHT SCREEN_HEIGHT
#define M4_WIDTH SCREEN_WIDTH
#define M4_HEIGHT SCREEN_HEIGHT
#define M5_WIDTH 160
#define M5_HEIGHT 128
#define SCREEN_WIDTH_T (SCREEN_WIDTH/8)
#define SCREEN_HEIGHT_T (SCREEN_HEIGHT/8)
#define SCREEN_LINES 228
#define SCR_W SCREEN_WIDTH
#define SCR_H SCREEN_HEIGHT
#define SCR_WT SCREEN_WIDTH_T
#define SCR_HT SCREEN_HEIGHT_T
#define LAYER_BG0 0x0001
#define LAYER_BG1 0x0002
#define LAYER_BG2 0x0004
#define LAYER_BG3 0x0008
#define LAYER_OBJ 0x0010
#define LAYER_BD 0x0020
#define CLR_MASK 0x001F
#define RED_MASK 0x001F
#define RED_SHIFT 0
#define GREEN_MASK 0x03E0
#define GREEN_SHIFT 5
#define BLUE_MASK 0x7C00
#define BLUE_SHIFT 10
#define CBB_CLEAR(cbb) memset32(&tile_mem[cbb], 0, CBB_SIZE/4)
#define SBB_CLEAR(sbb) memset32(&se_mem[sbb], 0, SBB_SIZE/4)
#define SBB_CLEAR_ROW(sbb, row) memset32(&se_mem[sbb][(row)*32], 0, 32/2)
#define __BG_TYPES ((0x0C7F<<16)|(0x0C40))
```

```
#define BG_IS_AFFINE(n) (( __BG_TYPES>>(4*(REG_DISPCNT&7)+(n))&1 )
#define BG_IS_AVAIL(n) (( __BG_TYPES>>(4*(REG_DISPCNT&7)+(n)+16))&1 )
```

## Base Color constants

```
#define CLR_BLACK 0x0000
#define CLR_RED 0x001F
#define CLR_LIME 0x03E0
#define CLR_YELLOW 0x03FF
#define CLR_BLUE 0x7C00
#define CLR_MAG 0x7C1F
#define CLR_CYAN 0x7FE0
#define CLR_WHITE 0xFFFF
```

## Additional colors

```
#define CLR_DEAD 0xDEAD
#define CLR_MAROON 0x0010
#define CLR_GREEN 0x0200
#define CLR_OLIVE 0x0210
#define CLR_ORANGE 0x021F
#define CLR_NAVY 0x4000
#define CLR_PURPLE 0x4010
#define CLR_TEAL 0x4200
#define CLR_GRAY 0x4210
#define CLR_MEDGRAY 0x5294
#define CLR_SILVER 0x6318
#define CLR_MONEYGREEN 0x6378
#define CLR_FUCHSIA 0x7C1F
#define CLR_SKYBLUE 0x7B34
#define CLR_CREAM 0x7BFF
```

# Functions

INLINE void	<b>vid_vsync</b> (void)
void	<b>vid_wait</b> (uint frames)
u16 *	<b>vid_flip</b> (void) <i>Flip the display page.</i>
void	<b>clr_rotate</b> ( <b>COLOR</b> *clrs, uint nclrs, int ror) <i>Rotate nclrs colors at clrs to the right by ror.</i>
void	<b>clr_blend</b> (const <b>COLOR</b> *srca, const <b>COLOR</b> *srcb, <b>COLOR</b> *dst, u32 nclrs, u32 alpha) <i>Blends color arrays srca and srcb into dst.</i>
void	<b>clr_fade</b> (const <b>COLOR</b> *src, <b>COLOR</b> clr, <b>COLOR</b> *dst, u32 nclrs, u32 alpha) <i>Fades color arrays srca to clr into dst.</i>
void	<b>clr_grayscale</b> ( <b>COLOR</b> *dst, const <b>COLOR</b> *src, uint nclrs) <i>Transform colors to grayscale.</i>
void	<b>clr_rgbscale</b> ( <b>COLOR</b> *dst, const <b>COLOR</b> *src, uint nclrs, <b>COLOR</b> clr) <i>Transform colors to an rgbscale.</i>
void	<b>clr_adj_brightness</b> ( <b>COLOR</b> *dst, const <b>COLOR</b> *src, uint nclrs, <b>FIXED</b> bright) <i>Adjust brightness by bright.</i>
void	<b>clr_adj_contrast</b> ( <b>COLOR</b> *dst, const <b>COLOR</b> *src, uint nclrs, <b>FIXED</b> contrast) <i>Adjust contrast by contrast.</i>
void	<b>clr_adj_intensity</b> ( <b>COLOR</b> *dst, const <b>COLOR</b> *src, uint nclrs, <b>FIXED</b> intensity) <i>Adjust intensity by intensity.</i>
void	<b>pal_gradient</b> ( <b>COLOR</b> *pal, int first, int last) <i>Create a gradient between pal[first] and pal[last].</i>
void	<b>pal_gradient_ex</b> ( <b>COLOR</b> *pal, int first, int last, <b>COLOR</b> clr_first, <b>COLOR</b> clr_last) <i>Create a gradient between pal[first] and pal[last].</i>
IWRAM_CODE void	<b>clr_blend_fast</b> ( <b>COLOR</b> *srca, <b>COLOR</b> *srcb, <b>COLOR</b>

		*dst, uint nclrs, u32 alpha) <i>Blends color arrays srca and srcb into dst.</i>
IWRAM_CODE void	<b>clr_fade_fast</b> ( <b>COLOR</b> *src, <b>COLOR</b> clr, <b>COLOR</b> *dst, uint nclrs, u32 alpha)	<i>Fades color arrays srca to clr into dst.</i>
INLINE <b>COLOR</b>	<b>RGB15</b> (int red, int green, int blue)	<i>Create a 15bit BGR color.</i>
INLINE <b>COLOR</b>	<b>RGB15_SAFE</b> (int red, int green, int blue)	<i>Create a 15bit BGR color, with proper masking of R,G,B components.</i>
INLINE <b>COLOR</b>	<b>RGB8</b> (u8 red, u8 green, u8 blue)	<i>Create a 15bit BGR color, using 8bit components.</i>
INLINE void	<b>se_fill</b> (SCR_ENTRY *sbb, SCR_ENTRY se)	<i>Fill screenblock sbb with se.</i>
INLINE void	<b>se_plot</b> (SCR_ENTRY *sbb, int x, int y, SCR_ENTRY se)	<i>Plot a screen entry at (x,y) of screenblock sbb.</i>
INLINE void	<b>se_rect</b> (SCR_ENTRY *sbb, int left, int top, int right, int bottom, SCR_ENTRY se)	<i>Fill a rectangle on sbb with se.</i>
INLINE void	<b>se_frame</b> (SCR_ENTRY *sbb, int left, int top, int right, int bottom, SCR_ENTRY se)	<i>Create a border on sbb with se.</i>
void	<b>se_window</b> (SCR_ENTRY *sbb, int left, int top, int right, int bottom, SCR_ENTRY se0)	<i>Create a framed rectangle.</i>
void	<b>se_hline</b> (SCR_ENTRY *sbb, int x0, int x1, int y, SCR_ENTRY se)	
void	<b>se_vline</b> (SCR_ENTRY *sbb, int x, int y0, int y1, SCR_ENTRY se)	
INLINE void	<b>bg_aff_set</b> ( <b>BG_AFFINE</b> *bgaff, <b>FIXED</b> pa, <b>FIXED</b> pb, <b>FIXED</b> pc, <b>FIXED</b> pd)	<i>Set the elements of an bg affine matrix.</i>
INLINE void	<b>bg_aff_identity</b> ( <b>BG_AFFINE</b> *bgaff)	<i>Set an bg affine matrix to the identity matrix.</i>
INLINE void	<b>bg_aff_scale</b> ( <b>BG_AFFINE</b> *bgaff, <b>FIXED</b> sx, <b>FIXED</b> sy)	

	<i>Set an bg affine matrix for scaling.</i>
INLINE void	<b>bg_aff_shearx</b> ( <b>BG_AFFINE</b> *bgaff, <b>FIXED</b> hx)
INLINE void	<b>bg_aff_sheary</b> ( <b>BG_AFFINE</b> *bgaff, <b>FIXED</b> hy)
void	<b>bg_aff_rotate</b> ( <b>BG_AFFINE</b> *bgaff, u16 alpha) <i>Set bg matrix to counter-clockwise rotation.</i>
void	<b>bg_aff_rotscale</b> ( <b>BG_AFFINE</b> *bgaff, int sx, int sy, u16 alpha) <i>Set bg matrix to 2d scaling, then counter-clockwise rotation.</i>
void	<b>bg_aff_premul</b> ( <b>BG_AFFINE</b> *dst, const <b>BG_AFFINE</b> *src) <i>Pre-multiply dst by src: D = S*D.</i>
void	<b>bg_aff_postmul</b> ( <b>BG_AFFINE</b> *dst, const <b>BG_AFFINE</b> *src) <i>Post-multiply dst by src: D= D*S.</i>
void	<b>bg_aff_rotscale2</b> ( <b>BG_AFFINE</b> *bgaff, const AFF_SRC *as) <i>Set bg matrix to 2d scaling, then counter-clockwise rotation.</i>
void	<b>bg_rotyscale_ex</b> ( <b>BG_AFFINE</b> *bgaff, const AFF_SRC_EX *asx) <i>Set bg affine matrix to a rot/scale around an arbitrary point.</i>
INLINE void	<b>bg_aff_copy</b> ( <b>BG_AFFINE</b> *dst, const <b>BG_AFFINE</b> *src) <i>Copy bg affine aparameters.</i>

## Generic 8bpp bitmaps

void	<b>bmp8_plot</b> (int x, int y, u32 clr, void *dstBase, uint dstP) <i>Plot a single pixel on a 8-bit buffer.</i>
void	<b>bmp8_hline</b> (int x1, int y, int x2, u32 clr, void *dstBase, uint dstP) <i>Draw a horizontal line on an 8bit buffer.</i>
void	<b>bmp8_vline</b> (int x, int y1, int y2, u32 clr, void *dstBase, uint dstP) <i>Draw a vertical line on an 8bit buffer.</i>
void	<b>bmp8_line</b> (int x1, int y1, int x2, int y2, u32 clr, void

	*dstBase, uint dstP) <i>Draw a line on an 8bit buffer.</i>
void	<b>bmp8_rect</b> (int left, int top, int right, int bottom, u32 clr, void *dstBase, uint dstP) <i>Draw a rectangle in 8bit mode; internal routine.</i>
void	<b>bmp8_frame</b> (int left, int top, int right, int bottom, u32 clr, void *dstBase, uint dstP) <i>Draw a rectangle in 8bit mode; internal routine.</i>

## Generic 16bpp bitmaps

void	<b>bmp16_plot</b> (int x, int y, u32 clr, void *dstBase, uint dstP) <i>Plot a single pixel on a 16-bit buffer.</i>
void	<b>bmp16_hline</b> (int x1, int y, int x2, u32 clr, void *dstBase, uint dstP) <i>Draw a horizontal line on an 16bit buffer.</i>
void	<b>bmp16_vline</b> (int x, int y1, int y2, u32 clr, void *dstBase, uint dstP) <i>Draw a vertical line on an 16bit buffer.</i>
void	<b>bmp16_line</b> (int x1, int y1, int x2, int y2, u32 clr, void *dstBase, uint dstP) <i>Draw a line on an 16bit buffer.</i>
void	<b>bmp16_rect</b> (int left, int top, int right, int bottom, u32 clr, void *dstBase, uint dstP) <i>Draw a rectangle in 16bit mode; internal routine.</i>
void	<b>bmp16_frame</b> (int left, int top, int right, int bottom, u32 clr, void *dstBase, uint dstP) <i>Draw a rectangle in 16bit mode; internal routine.</i>

---

## **Detailed Description**

**Author:**

J Vijn

**Date:**

20060604 - 20080311

---

## Function Documentation

```
u16* vid_flip( void )
```

Flip the display page.

Toggles the display page in REG\_DISPCNT and sets *vid\_page* to point to the back buffer.

**Returns:**

Current back buffer pointer.

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tte\_init\_ase.c File Reference**

```
#include <string.h> #include "tonc_memdef.h"
#include "tonc_core.h"
#include "tonc_bios.h"
#include "tonc_surface.h"
#include "tonc_tte.h"
```

## Functions

void	<b>tte_init_ase</b> (int bgnr, u16 bgcnt, u8 ase0, u32 clrs, u32 bupofs, const <b>TFont</b> *font, <b>fnDrawg</b> proc)
	<i>Initialize text system for affine screen-entry fonts.</i>

---

# Detailed Description

**Author:**

J Vijn

**Date:**

20070701 - 20080515

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tte\_init bmp.c File Reference**

```
#include <string.h> #include "tonc_memdef.h"
#include "tonc_core.h"
#include "tonc_video.h"
#include "tonc_tte.h"
#include "tonc_surface.h"
```

## Functions

void	<b>tte_init_bmp</b> (int vmode, const <b>TFont</b> *font, <b>fnDrawg</b> proc) <i>Initialize text system for bitmap fonts.</i>
------	---

void	<b>bmp8_erase</b> (int left, int top, int right, int bottom) <i>Erase part of the 8bpp text canvas.</i>
------	--

void	<b>bmp16_erase</b> (int left, int top, int right, int bottom) <i>Erase part of the 16bpp text canvas.</i>
------	--

---

# Detailed Description

**Author:**

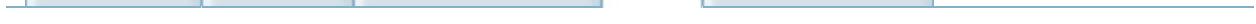
J Vijn

**Date:**

20070517 - 20080229

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tte\_init\_chr4c.c File Reference**

```
#include <string.h> #include <tonc.h>
#include "tonc\_tte.h"
```

## Functions

void	<b>tte_init_chr4c</b> (int bgnr, u16 bgcnt, u16 se0, u32 cattrs, u32 clrs, const <b>TFont</b> *font, <b>fnDrawg</b> proc)
<i>Initialize text system for 4bpp tiled, column-major surfaces.</i>	
void	<b>chr4c_erase</b> (int left, int top, int right, int bottom)
<i>Erase part of the 4bpp text canvas.</i>	

# Detailed Description

**Author:**

J Vijn

**Date:**

20070517 - 20080427

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tte\_init\_chr4r.c File Reference**

```
#include <string.h> #include <tonc.h>
#include "tonc_tte.h"
```

## Functions

void	<b>tte_init_chr4r</b> (int bgnr, u16 bgcnt, u16 se0, u32 cattrs, u32 clrs, const <b>TFont</b> *font, <b>fnDrawg</b> proc)
<i>Initialize text system for 4bpp tiled, column-major surfaces.</i>	
void	<b>chr4r_erase</b> (int left, int top, int right, int bottom)
<i>Erase part of the 4bpp text canvas.</i>	

# Detailed Description

**Author:**

J Vijn

**Date:**

20070517 - 20080515

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tte\_init\_obj.c File Reference**

```
#include <string.h> #include "tonc_memdef.h"
#include "tonc_core.h"
#include "tonc_bios.h"
#include "tonc_tte.h"
```

## Functions

void	<b>tte_init_obj</b> ( <b>OBJ_ATTR</b> *obj, u32 attr0, u32 attr1, u32 attr2, u32 clrs, u32 bupofs, const <b>TFont</b> *font, <b>fnDrawg</b> proc)
	<i>Initialize text system for screen-entry fonts.</i>

---

# Detailed Description

**Author:**

J Vijn

**Date:**

20070715 - 20080229

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tte\_init\_se.c File Reference**

```
#include <string.h> #include "tonc_memdef.h"
#include "tonc_core.h"
#include "tonc_bios.h"
#include "tonc_tte.h"
```

## Functions

void	<b>tte_init_se</b> (int bgnr, u16 bgcnt, SCR_ENTRY se0, u32 clrs, u32 bupofs, const <b>TFont</b> *font, <b>fnDrawg</b> proc)
	<i>Initialize text system for screen-entry fonts.</i>

---

# Detailed Description

**Author:**

J Vijn

**Date:**

20070628 - 20080229

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tte\_iohook.c File Reference**

```
#include <stdio.h> #include <string.h>
#include <stdarg.h>
#include <sys/iosupport.h>
#include "tonc_tte.h"
#include "tonc_nocash.h"
```

# Functions

uint	<b>utf8_decode_char</b> (const char *ptr, char **endptr) <i>Retrieve a single multibyte utf8 character.</i>
void	<b>tte_init_con</b> () <i>Init stdio capabilities.</i>
int	<b>tte_cmd_vt100</b> (const char *text) <i>Parse for VT100-sequences.</i>
int	<b>tte_con_nocash</b> (struct _reent *r, int fd, const char *text, int len)
int	<b>tte_con_write</b> (struct _reent *r, int fd, const char *text, int len) <i>Internal routine for stdio functionality.</i>

## Variables

const devoptab_t	<b>tte_dotab_stdout</b>
const devoptab_t	<b>tte_dotab_nocash</b>

---

## **Detailed Description**

**Author:**

J Vijn

**Date:**

20070517 - 20070517

---

# Variable Documentation

```
const devoptab_t tte_dotab_nocash
```

**Initial value:**

```
{
    "ttetenocash",
    0,
    NULL,
    NULL,
    tte_con_nocash,
    NULL,
    NULL,
    NULL
}
```

```
const devoptab_t tte_dotab_stdout
```

**Initial value:**

```
{
    "ttecon",
    0,
    NULL,
    NULL,
    tte_con_write,
    NULL,
    NULL,
    NULL
}
```

}

---

Generated on Mon Aug 25 17:03:56 2008 for libtonc by  doxygen 1.5.3

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# **tte\_main.c File Reference**

```
#include <stdio.h> #include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <tonc.h>
#include "tonc\_tte.h"
```

# Functions

void	<b>dummy_drawg</b> (uint gid)
void	<b>dummy_erase</b> (int left, int top, int right, int bottom)
INLINE char *	<b>eatwhite</b> (const char *str)
void	<b>tte_set_context</b> (TTC *tc) <i>Set the master context pointer.</i>
void	<b>tte_set_color_attr</b> (eint type, u16 cattr) <i>Set color attribute of type to cattr.</i>
void	<b>tte_set_color_attrs</b> (const u16 cattrs[]) <i>Load important color attribute data.</i>
void	<b>tte_set_color</b> (eint type, u16 color) <i>Set color attribute of type to color.</i>
void	<b>tte_set_colors</b> (const u16 colors[]) <i>Load important color data.</i>
void	<b>tte_init_base</b> (const TFont *font, fnDrawg drawProc, fnErase eraseProc) <i>Base initializer of a TTC.</i>
uint	<b>utf8_decode_char</b> (const char *ptr, char **endptr) <i>Retrieve a single multibyte utf8 character.</i>
char *	<b>tte_cmd_skip</b> (const char *str) <i>Find the string-position after the command.</i>
char *	<b>tte_cmd_next</b> (const char *str) <i>Move to the next command in a sequence.</i>
char *	<b>tte_cmd_default</b> (const char *str) <i>Text command handler.</i>
int	<b>tte_write_ex</b> (int x0, int y0, const char *text, const u16 *cattrs) <i>Extended string writer, with positional and color info.</i>
int	<b>tte_putc</b> (int ch) <i>Plot a single character; does wrapping too.</i>
int	<b>tte_write</b> (const char *text) <i>Render a string.</i>
void	<b>tte_erase_rect</b> (int left, int top, int right, int bottom) <i>Erase a portion of the screen (ignores margins).</i>

void	<b>tte_erase_screen ()</b> <i>Erase the screen (within the margins).</i>
void	<b>tte_erase_line ()</b> <i>Erase the whole line (within the margins).</i>
POINT16	<b>tte_get_text_size</b> (const char *str) <i>Get the size taken up by a string.</i>
void	<b>tte_set_margins</b> (int left, int top, int right, int bottom)

## Variables

<b>TTC</b>	<code>__tte_main_context</code>
<b>TTC *</b>	<code>gp_tte_context = &amp;__tte_main_context</code>

---

## **Detailed Description**

**Author:**

J Vijn

**Date:**

20070517 - 20080229

---

# Function Documentation

```
char* tte_cmd_next ( const char * str )
```

Move to the next command in a sequence.

**Returns:**

Position of EOS ('\0'), EOC ('}') or next cmd token (rest)

```
char* tte_cmd_skip ( const char * str )
```

Find the string-position after the command.

**Parameters:**

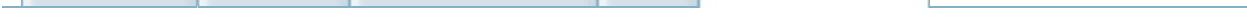
*str* String to check.

**Returns:**

The string-pointer after the current/next command. If there is no command-end, this moves to the end of the string.

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

- `_toncset()` : [tonc\\_core.h](#) , [tonc\\_core.c](#)
- `_BF_GET` : [tonc\\_core.h](#)
- `_BF_PREP` : [tonc\\_core.h](#)
- `_BF_SET` : [tonc\\_core.h](#)
- `_sbmp16_get_pixel()` : [tonc\\_surface.h](#)
- `_sbmp16_plot()` : [tonc\\_surface.h](#)
- `_sbmp8_get_pixel()` : [tonc\\_surface.h](#)
- `_sbmp8_plot()` : [tonc\\_surface.h](#)
- `_schr4c_get_pixel()` : [tonc\\_surface.h](#)
- `_schr4c_plot()` : [tonc\\_surface.h](#)
- `_schr4r_get_pixel()` : [tonc\\_surface.h](#)
- `_schr4r_plot()` : [tonc\\_surface.h](#)
- `_srf_get_ptr()` : [tonc\\_surface.h](#)

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

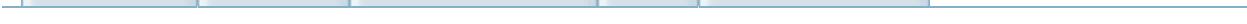
# libtonc Related Pages

Here is a list of all related documentation pages:

- [Todo List](#)
- [Deprecated List](#)

---

*Generated on Mon Aug 25 17:03:58 2008 for libtonc by*  *1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# Todo List

File **tonc\_sbmp16.c**

Code consistency.

File **tonc\_sbmp8.c**

Code consistency.

File **tonc\_schr4c.c**

Code consistency.

Global **tte\_cmd\_default**

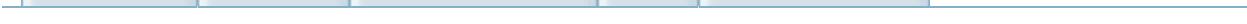
Scrolling and variables ?

Global **tte\_cmd\_vt100**

: check for buffer overflow.

Global **tte\_init\_obj**

Multi-bpp.

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)

# Deprecated List

## Group **grpText**

While potentially still useful, TTE is considerably more advanced. Use that instead.

## Group **grpVideoBmp**

The bmp8/bmp16 functions have been superceded by the surface functions (sbmp8/sbmp16) for the most part. The former group has been kept mostly for reference purposes.

---

*Generated on Mon Aug 25 17:03:56 2008 for libtonc by*  **1.5.3**

[Main Page](#) [Modules](#) [Data Structures](#) [Files](#) [Related Pages](#)

---

[Alphabetical List](#) [Data Structures](#) [Data Fields](#)

# libtonc Data Structure Index

B | D | I | M | O | P | R | T | V

**B**

BG\_POINT  
BgAffineDest  
BgAffineSource  
BLOCK  
BUP

DMA\_REC

I  
IRQ\_REC  
IRQ\_SENDER  
M  
MultiBootParam

**O**

OBJ\_AFFINE  
OBJ\_ATTR  
ObjAffineDest  
ObjAffineSource  
P

POINT32

**R**

RECT32  
REPEAT\_REC  
T  
TFont

**D**

B | D | I | M | O | P | R | T | V

---

Generated on Mon Aug 25 17:03:57 2008 for libtonc by  1.5.3



- a -

- alpha : **ObjAffineSource** , **BgAffineSource**

- b -

- bpp : **TFont**

- c -

- cattr : **TTC**
- cellH : **TFont**
- cellSize : **TFont**
- cellW : **TFont**
- charCount : **TFont**
- charH : **TFont**
- charLut : **TTC**
- charOffset : **TFont**
- charW : **TFont**
- count : **REPEAT\_REC**
- ctrl : **TTC**
- cursorX : **TTC**
- cursorY : **TTC**

- d -

- data : **TFont**

- delay : REPEAT\_REC
- drawgProc : TTC
- dst : TTC
- dst\_bpp : BUP
- dst\_ofs : BUP

- e -

- eraseProc : TTC
- extra : TFont

- f -

- flag : IRQ\_REC , IRQ\_SENDER
- font : TTC
- fontTable : TTC

- h -

- heights : TFont

- i -

- isr : IRQ\_REC

- k -

- keys : REPEAT\_REC

- m -

- mask : REPEAT\_REC

- r -

- reg\_ofs : **IRQ\_SENDER**
- repeat : **REPEAT\_REC**

- s -

- scr\_x : **BgAffineSource**
- scr\_y : **BgAffineSource**
- src\_bpp : **BUP**
- src\_len : **BUP**
- stringTable : **TTC**
- sx : **BgAffineSource , ObjAffineSource**
- sy : **BgAffineSource , ObjAffineSource**

- t -

- tex\_x : **BgAffineSource**
- tex\_y : **BgAffineSource**

- w -

- widths : **TFont**

Main Page Modules Data Structures Files Related Pages

File List Globals

All Functions Variables Typedefs Enumerations Enumerator Defines

a b c d f h i k l m n o p q r s t u v w

- `_toncset()` : **tonc\_core.h**, **tonc\_core.c**
- `_sbmp16_get_pixel()` : **tonc\_surface.h**
- `_sbmp16_plot()` : **tonc\_surface.h**
- `_sbmp8_get_pixel()` : **tonc\_surface.h**
- `_sbmp8_plot()` : **tonc\_surface.h**
- `_schr4c_get_pixel()` : **tonc\_surface.h**
- `_schr4c_plot()` : **tonc\_surface.h**
- `_schr4r_get_pixel()` : **tonc\_surface.h**
- `_schr4r_plot()` : **tonc\_surface.h**
- `_srf_get_ptr()` : **tonc\_surface.h**

- nocash\_buffer : [tonc\\_nocash.h](#)
- sys8Font : [tonc\\_tte.h](#)
- sys8Glyphs : [tonc\\_tte.h](#)
- verdana10Font : [tonc\\_tte.h](#)
- verdana10Glyphs : [tonc\\_tte.h](#)
- verdana10Widths : [tonc\\_tte.h](#)
- verdana9\_b4Font : [tonc\\_tte.h](#)
- verdana9\_b4Glyphs : [tonc\\_tte.h](#)
- verdana9\_b4Widths : [tonc\\_tte.h](#)
- verdana9bFont : [tonc\\_tte.h](#)
- verdana9bGlyphs : [tonc\\_tte.h](#)
- verdana9bWidths : [tonc\\_tte.h](#)
- verdana9Font : [tonc\\_tte.h](#)
- verdana9Glyphs : [tonc\\_tte.h](#)
- verdana9iFont : [tonc\\_tte.h](#)
- verdana9iGlyphs : [tonc\\_tte.h](#)
- verdana9iWidths : [tonc\\_tte.h](#)
- verdana9Widths : [tonc\\_tte.h](#)

- BG\_AFFINE : [tonc\\_types.h](#)
- COLOR : [tonc\\_types.h](#)
- CSTR : [tonc\\_types.h](#)
- FIXED : [tonc\\_types.h](#)
- fn\_i\_i : [tonc\\_types.h](#)
- fn\_v\_i : [tonc\\_types.h](#)
- fnDrawg : [tonc\\_tte.h](#)
- fnErase : [tonc\\_tte.h](#)
- fnptr : [tonc\\_types.h](#)
- PALBANK : [tonc\\_types.h](#)
- SAE : [tonc\\_types.h](#)
- SE : [tonc\\_types.h](#)

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)[File List](#)[Globals](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Defines](#)

- bool : [tonc\\_types.h](#)
- elrqIndex : [tonc\\_irq.h](#)
- eKeyIndex : [tonc\\_input.h](#)
- ESurfaceType : [tonc\\_surface.h](#)

---

Generated on Mon Aug 25 17:03:58 2008 for libtonc by  doxygen 1.5.3

- SRF\_ALLOCATED : [tonc\\_surface.h](#)
- SRF\_BMP16 : [tonc\\_surface.h](#)
- SRF\_BMP8 : [tonc\\_surface.h](#)
- SRF\_CHR4C : [tonc\\_surface.h](#)
- SRF\_CHR4R : [tonc\\_surface.h](#)
- SRF\_CHR8 : [tonc\\_surface.h](#)
- SRF\_NONE : [tonc\\_surface.h](#)

- - -

- `_BF_GET` : [tonc\\_core.h](#)
- `_BF_PREP` : [tonc\\_core.h](#)
- `_BF_SET` : [tonc\\_core.h](#)

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

### - a -

- ABS : [tonc\\_math.h](#)
- ALIGN : [tonc\\_types.h](#)
- align() : [tonc\\_core.h](#)
- ALIGN4 : [tonc\\_types.h](#)
- ArcTan() : [tonc\\_bios.h](#)
- ArcTan2() : [tonc\\_bios.h](#)
- ase\_drawg() : [ase\\_drawg.c](#) , [tonc\\_tte.h](#)
- ase\_drawg\_s() : [tonc\\_tte.h](#) , [ase\\_drawg.c](#)
- ase\_drawg\_w8h16() : [tonc\\_tte.h](#) , [ase\\_drawg.c](#)
- ase\_drawg\_w8h8() : [tonc\\_tte.h](#) , [ase\\_drawg.c](#)
- ase\_erase() : [ase\\_drawg.c](#) , [tonc\\_tte.h](#)
- ASM\_BREAK : [tonc\\_core.h](#)
- ASM\_CMT : [tonc\\_core.h](#)
- ASM\_NOP : [tonc\\_core.h](#)
- ATTR0\_4BPP : [tonc\\_memdef.h](#)
- ATTR0\_8BPP : [tonc\\_memdef.h](#)
- ATTR0\_AFF : [tonc\\_memdef.h](#)
- ATTR0\_AFF\_DBL : [tonc\\_memdef.h](#)
- ATTR0\_BLEND : [tonc\\_memdef.h](#)
- ATTR0\_HIDE : [tonc\\_memdef.h](#)
- ATTR0\_MOSAIC : [tonc\\_memdef.h](#)
- ATTR0\_REG : [tonc\\_memdef.h](#)
- ATTR0\_SQUARE : [tonc\\_memdef.h](#)

- ATTR0\_TALL : [tonc\\_memdef.h](#)
  - ATTR0\_WIDE : [tonc\\_memdef.h](#)
  - ATTR0\_WINDOW : [tonc\\_memdef.h](#)
  - ATTR1\_HFLIP : [tonc\\_memdef.h](#)
  - ATTR1\_SIZE\_16x16 : [tonc\\_memdef.h](#)
  - ATTR1\_SIZE\_16x32 : [tonc\\_memdef.h](#)
  - ATTR1\_SIZE\_16x8 : [tonc\\_memdef.h](#)
  - ATTR1\_SIZE\_32x16 : [tonc\\_memdef.h](#)
  - ATTR1\_SIZE\_32x32 : [tonc\\_memdef.h](#)
  - ATTR1\_SIZE\_32x64 : [tonc\\_memdef.h](#)
  - ATTR1\_SIZE\_32x8 : [tonc\\_memdef.h](#)
  - ATTR1\_SIZE\_64x32 : [tonc\\_memdef.h](#)
  - ATTR1\_SIZE\_64x64 : [tonc\\_memdef.h](#)
  - ATTR1\_SIZE\_8x16 : [tonc\\_memdef.h](#)
  - ATTR1\_SIZE\_8x32 : [tonc\\_memdef.h](#)
  - ATTR1\_SIZE\_8x8 : [tonc\\_memdef.h](#)
  - ATTR1\_VFLIP : [tonc\\_memdef.h](#)
  - ATTR2\_ID\_MASK : [tonc\\_memdef.h](#)
-

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

### - b -

- [bf\\_clamp\(\)](#) : [tonc\\_core.h](#)
- [bf\\_get\(\)](#) : [tonc\\_core.h](#)
- [BF\\_MASK](#) : [tonc\\_core.h](#)
- [bf\\_merge\(\)](#) : [tonc\\_core.h](#)
- [BNF\\_CMP](#) : [tonc\\_core.h](#)
- [BNF\\_GET](#) : [tonc\\_core.h](#)
- [BNF\\_GET2](#) : [tonc\\_core.h](#)
- [BNF\\_PREP](#) : [tonc\\_core.h](#)
- [BNF\\_PREP2](#) : [tonc\\_core.h](#)
- [BNF\\_SET](#) : [tonc\\_core.h](#)
- [BNF\\_SET2](#) : [tonc\\_core.h](#)
- [BG\\_4BPP](#) : [tonc\\_memdef.h](#)
- [BG\\_8BPP](#) : [tonc\\_memdef.h](#)
- [BG\\_AFF\\_128x128](#) : [tonc\\_memdef.h](#)
- [BG\\_AFF\\_16x16](#) : [tonc\\_memdef.h](#)
- [BG\\_AFF\\_32x32](#) : [tonc\\_memdef.h](#)
- [BG\\_AFF\\_64x64](#) : [tonc\\_memdef.h](#)
- [bg\\_aff\\_copy\(\)](#) : [tonc\\_video.h](#)
- [bg\\_aff\\_identity\(\)](#) : [tonc\\_video.h](#)
- [BG\\_AFF\\_OFS](#) : [tonc\\_bios.h](#)
- [bg\\_aff\\_postmul\(\)](#) : [tonc\\_video.h](#) , [tonc\\_bg\\_affine.c](#)
- [bg\\_aff\\_premul\(\)](#) : [tonc\\_video.h](#) , [tonc\\_bg\\_affine.c](#)
- [bg\\_aff\\_rotate\(\)](#) : [tonc\\_video.h](#) , [tonc\\_bg\\_affine.c](#)

- bg\_aff\_rotscale() : [tonc\\_video.h](#) , [tonc\\_bg\\_affine.c](#)
- bg\_aff\_rotscale2() : [tonc\\_video.h](#) , [tonc\\_bg\\_affine.c](#)
- bg\_aff\_scale() : [tonc\\_video.h](#)
- bg\_aff\_set() : [tonc\\_video.h](#)
- BG\_AFFINE : [tonc\\_types.h](#)
- BG\_MOSAIC : [tonc\\_memdef.h](#)
- BG\_REG\_32x32 : [tonc\\_memdef.h](#)
- BG\_REG\_32x64 : [tonc\\_memdef.h](#)
- BG\_REG\_64x32 : [tonc\\_memdef.h](#)
- BG\_REG\_64x64 : [tonc\\_memdef.h](#)
- bg\_rotscale\_ex() : [tonc\\_video.h](#) , [tonc\\_bg\\_affine.c](#)
- BG\_WRAP : [tonc\\_memdef.h](#)
- BgAffineSet() : [tonc\\_bios.h](#)
- BIT : [tonc\\_core.h](#)
- BIT\_CLEAR : [tonc\\_core.h](#)
- BIT\_EQ : [tonc\\_core.h](#)
- BIT\_FLIP : [tonc\\_core.h](#)
- BIT\_MASK : [tonc\\_core.h](#)
- BIT\_SET : [tonc\\_core.h](#)
- BIT\_SHIFT : [tonc\\_core.h](#)
- bit\_tribool() : [tonc\\_core.h](#)
- BLD\_ALL : [tonc\\_memdef.h](#)
- BLD\_BACKDROP : [tonc\\_memdef.h](#)
- BLD\_BG0 : [tonc\\_memdef.h](#)
- BLD\_BG1 : [tonc\\_memdef.h](#)
- BLD\_BG2 : [tonc\\_memdef.h](#)
- BLD\_BG3 : [tonc\\_memdef.h](#)
- BLD\_BLACK : [tonc\\_memdef.h](#)
- BLD\_OBJ : [tonc\\_memdef.h](#)
- BLD\_OFF : [tonc\\_memdef.h](#)
- BLD\_STD : [tonc\\_memdef.h](#)
- BLD\_WHITE : [tonc\\_memdef.h](#)
- bmp16\_drawg() : [bmp16\\_drawg.c](#) , [tonc\\_tte.h](#)

- `bmp16_drawg_b1cos()` : **tonc\_tte.h** ,  
**bmp16\_drawg\_b1cs.c**
- `bmp16_drawg_b1cts()` : **tonc\_tte.h** ,  
**bmp16\_drawg\_b1cs.c**
- `bmp16_drawg_t()` : **tonc\_tte.h** , **bmp16\_drawg.c**
- `bmp16_erase()` : **tonc\_tte.h** , **tte\_init bmp.c**
- `bmp16_frame()` : **tonc\_video.h** , **tonc\_bmp16.c**
- `bmp16_hline()` : **tonc\_video.h** , **tonc\_bmp16.c**
- `bmp16_line()` : **tonc\_video.h** , **tonc\_bmp16.c**
- `bmp16_plot()` : **tonc\_video.h** , **tonc\_bmp16.c**
- `bmp16_rect()` : **tonc\_bmp16.c** , **tonc\_video.h**
- `bmp16_vline()` : **tonc\_video.h** , **tonc\_bmp16.c**
- `bmp8_drawg()` : **tonc\_tte.h** , **bmp8\_drawg.c**
- `bmp8_drawg_b1cos()` : **tonc\_tte.h**
- `bmp8_drawg_b1cts()` : **tonc\_tte.h**
- `bmp8_drawg_b1cts_fast()` : **tonc\_tte.h**
- `bmp8_drawg_t()` : **tonc\_tte.h** , **bmp8\_drawg.c**
- `bmp8_erase()` : **tonc\_tte.h** , **tte\_init bmp.c**
- `bmp8_frame()` : **tonc\_bmp8.c** , **tonc\_video.h**
- `bmp8_hline()` : **tonc\_bmp8.c** , **tonc\_video.h**
- `bmp8_line()` : **tonc\_video.h** , **tonc\_bmp8.c**
- `bmp8_plot()` : **tonc\_bmp8.c** , **tonc\_video.h**
- `bmp8_rect()` : **tonc\_video.h** , **tonc\_bmp8.c**
- `bmp8_vline()` : **tonc\_video.h** , **tonc\_bmp8.c**
- `bool` : **tonc\_types.h**
- `bytes2hword()` : **tonc\_core.h**
- `bytes2word()` : **tonc\_core.h**

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

### - C -

- CBB\_CLEAR : [tonc\\_video.h](#)
- CBB\_SIZE : [tonc\\_memmap.h](#)
- CFS\_CPY : [tonc\\_bios.h](#)
- CFS\_FILL : [tonc\\_bios.h](#)
- chr4\_lmask() : [tonc\\_schr4c.c](#) , [tonc\\_schr4r.c](#)
- chr4\_rmask() : [tonc\\_schr4r.c](#) , [tonc\\_schr4c.c](#)
- chr4c\_colset() : [tonc\\_schr4c.c](#)
- chr4c\_drawg\_b1cts() : [tonc\\_tte.h](#) , [chr4c\\_drawg\\_b1cts.c](#)
- chr4c\_drawg\_b1cts\_fast() : [tonc\\_tte.h](#)
- chr4c\_drawg\_b4cts() : [tonc\\_tte.h](#)
- chr4c\_drawg\_b4cts\_fast() : [tonc\\_tte.h](#)
- chr4c\_erase() : [tte\\_init\\_chr4c.c](#) , [tonc\\_tte.h](#)
- chr4r\_colset() : [tonc\\_schr4r.c](#)
- chr4r\_drawg\_b1cts() : [tonc\\_tte.h](#) , [chr4r\\_drawg\\_b1cts.c](#)
- chr4r\_drawg\_b1cts\_fast() : [tonc\\_tte.h](#)
- chr4r\_erase() : [tonc\\_tte.h](#) , [tte\\_init\\_chr4r.c](#)
- CLAMP : [tonc\\_math.h](#)
- clamp() : [tonc\\_math.h](#)
- clr\_adj\_brightness() : [tonc\\_video.h](#) , [tonc\\_color.c](#)
- clr\_adj\_contrast() : [tonc\\_video.h](#) , [tonc\\_color.c](#)
- clr\_adj\_intensity() : [tonc\\_video.h](#) , [tonc\\_color.c](#)
- clr\_blend() : [tonc\\_color.c](#) , [tonc\\_video.h](#)
- clr\_blend\_fast() : [tonc\\_video.h](#)

- clr\_fade() : [tonc\\_video.h](#) , [tonc\\_color.c](#)
  - clr\_fade\_fast() : [tonc\\_video.h](#)
  - clr\_grayscale() : [tonc\\_video.h](#) , [tonc\\_color.c](#)
  - clr\_rgbscale() : [tonc\\_color.c](#) , [tonc\\_video.h](#)
  - clr\_rotate() : [tonc\\_color.c](#) , [tonc\\_video.h](#)
  - COLOR : [tonc\\_types.h](#)
  - countof : [tonc\\_core.h](#)
  - CpuFastFill() : [tonc\\_bios.h](#)
  - CpuFastSet() : [tonc\\_bios.h](#)
  - CpuSet() : [tonc\\_bios.h](#)
  - CS\_CPY : [tonc\\_bios.h](#)
  - CS\_CPY16 : [tonc\\_bios.h](#)
  - CS\_CPY32 : [tonc\\_bios.h](#)
  - CS\_FILL : [tonc\\_bios.h](#)
  - CS\_FILL32 : [tonc\\_bios.h](#)
  - CSTR : [tonc\\_types.h](#)
- 

Generated on Mon Aug 25 17:03:58 2008 for libtonc by  1.5.3

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

### - d -

- DCNT\_BG0 : [tonc\\_memdef.h](#)
- DCNT\_BG1 : [tonc\\_memdef.h](#)
- DCNT\_BG2 : [tonc\\_memdef.h](#)
- DCNT\_BG3 : [tonc\\_memdef.h](#)
- DCNT\_BLANK : [tonc\\_memdef.h](#)
- DCNT\_GB : [tonc\\_memdef.h](#)
- DCNT\_MODE0 : [tonc\\_memdef.h](#)
- DCNT\_MODE1 : [tonc\\_memdef.h](#)
- DCNT\_MODE2 : [tonc\\_memdef.h](#)
- DCNT\_MODE3 : [tonc\\_memdef.h](#)
- DCNT\_MODE4 : [tonc\\_memdef.h](#)
- DCNT\_MODE5 : [tonc\\_memdef.h](#)
- DCNT\_OAM\_HBL : [tonc\\_memdef.h](#)
- DCNT\_OBJ : [tonc\\_memdef.h](#)
- DCNT\_OBJ\_1D : [tonc\\_memdef.h](#)
- DCNT\_OBJ\_2D : [tonc\\_memdef.h](#)
- DCNT\_PAGE : [tonc\\_memdef.h](#)
- DCNT\_WIN0 : [tonc\\_memdef.h](#)
- DCNT\_WIN1 : [tonc\\_memdef.h](#)
- DCNT\_WINOBJ : [tonc\\_memdef.h](#)
- DEPRECATED : [tonc\\_types.h](#)
- Div() : [tonc\\_bios.h](#)
- DivAbs() : [tonc\\_bios.h](#)

- DivArm() : **tonc\_bios.h**
- DivArmAbs() : **tonc\_bios.h**
- DivArmMod() : **tonc\_bios.h**
- DivSafe() : **tonc\_bios.h**
- dma3\_cpy() : **tonc\_core.h**
- dma3\_fill() : **tonc\_core.h**
- DMA\_16 : **tonc\_memdef.h**
- DMA\_32 : **tonc\_memdef.h**
- DMA\_AT\_FIFO : **tonc\_memdef.h**
- DMA\_AT\_HBLANK : **tonc\_memdef.h**
- DMA\_AT\_NOW : **tonc\_memdef.h**
- DMA\_AT\_REFRESH : **tonc\_memdef.h**
- DMA\_AT\_SPECIAL : **tonc\_memdef.h**
- DMA\_AT\_VBLANK : **tonc\_memdef.h**
- dma\_cpy() : **tonc\_core.h**
- DMA\_DST\_DEC : **tonc\_memdef.h**
- DMA\_DST\_FIXED : **tonc\_memdef.h**
- DMA\_DST\_INC : **tonc\_memdef.h**
- DMA\_DST\_RELOAD : **tonc\_memdef.h**
- DMA\_ENABLE : **tonc\_memdef.h**
- dma\_fill() : **tonc\_core.h**
- DMA\_GAMEPAK : **tonc\_memdef.h**
- DMA\_IRQ : **tonc\_memdef.h**
- DMA\_REPEAT : **tonc\_memdef.h**
- DMA\_SRC\_DEC : **tonc\_memdef.h**
- DMA\_SRC\_FIXED : **tonc\_memdef.h**
- DMA\_SRC\_INC : **tonc\_memdef.h**
- DMA\_TRANSFER : **tonc\_core.h**
- DSTAT\_HBL IRQ : **tonc\_memdef.h**
- DSTAT\_IN\_HBL : **tonc\_memdef.h**
- DSTAT\_IN\_VBL : **tonc\_memdef.h**
- DSTAT\_IN\_VCT : **tonc\_memdef.h**
- DSTAT\_VBL IRQ : **tonc\_memdef.h**

- DSTAT\_VCT\_IRQ : [tonc\\_memdef.h](#)
  - dup16() : [tonc\\_core.h](#)
  - dup8() : [tonc\\_core.h](#)
- 

Generated on Mon Aug 25 17:03:58 2008 for libtonc by  doxygen 1.5.3

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

- e -

- elrqIndex : [tonc\\_irq.h](#)
- eKeyIndex : [tonc\\_input.h](#)
- ESurfaceType : [tonc\\_surface.h](#)
- EWRAM\_BSS : [tonc\\_types.h](#)
- EWRAM\_CODE : [tonc\\_types.h](#)
- EWRAM\_DATA : [tonc\\_types.h](#)

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

### - f -

- FIX\_SHIFT : [tonc\\_math.h](#)
- FIXED : [tonc\\_types.h](#)
- float2fx() : [tonc\\_math.h](#)
- fn\_i\_i : [tonc\\_types.h](#)
- fn\_v\_i : [tonc\\_types.h](#)
- fnDrawg : [tonc\\_tte.h](#)
- fnErase : [tonc\\_tte.h](#)
- fnptr : [tonc\\_types.h](#)
- fwf\_default : [tonc\\_tte.h](#)
- fx2float() : [tonc\\_math.h](#)
- fx2int() : [tonc\\_math.h](#)
- fx2ufrac() : [tonc\\_math.h](#)
- fx2uint() : [tonc\\_math.h](#)
- FX\_RECIMUL : [tonc\\_math.h](#)
- FX\_RECIPROCAL : [tonc\\_math.h](#)
- fxadd() : [tonc\\_math.h](#)
- fxdiv() : [tonc\\_math.h](#)
- fxdiv64() : [tonc\\_math.h](#)
- fmul() : [tonc\\_math.h](#)
- fmul64() : [tonc\\_math.h](#)
- fsub() : [tonc\\_math.h](#)

*Generated on Mon Aug 25 17:03:58 2008 for libtonc by*  **1.5.3**

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)[File List](#)[Globals](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Defines](#)[\\_](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[h](#)[i](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)[x](#)

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

- g -

- GRIT\_CPY : [tonc\\_core.h](#)

---

Generated on Mon Aug 25 17:03:58 2008 for libtonc by  doxygen 1.5.3

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

### - h -

- `hword2word()` : [tonc\\_core.h](#)

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

### - i -

- IN\_RANGE : [tonc\\_math.h](#)
- in\_range() : [tonc\\_math.h](#)
- INLINE : [tonc\\_types.h](#)
- int2fx() : [tonc\\_math.h](#)
- irq\_add() : [tonc\\_irq.h](#) , [tonc\\_irq.c](#)
- irq\_delete() : [tonc\\_irq.c](#) , [tonc\\_irq.h](#)
- IRQ\_DMA0 : [tonc\\_memdef.h](#)
- IRQ\_DMA1 : [tonc\\_memdef.h](#)
- IRQ\_DMA2 : [tonc\\_memdef.h](#)
- IRQ\_DMA3 : [tonc\\_memdef.h](#)
- IRQ\_GAMEPAK : [tonc\\_memdef.h](#)
- IRQ\_HBLANK : [tonc\\_memdef.h](#)
- irq\_init() : [tonc\\_irq.h](#) , [tonc\\_irq.c](#)
- IRQ\_INIT : [tonc\\_irq.h](#)
- IRQ\_KEYPAD : [tonc\\_memdef.h](#)
- IRQ\_SERIAL : [tonc\\_memdef.h](#)
- irq\_set() : [tonc\\_irq.h](#) , [tonc\\_irq.c](#)
- IRQ\_SET : [tonc\\_irq.h](#)
- irq\_set\_master() : [tonc\\_irq.h](#) , [tonc\\_irq.c](#)
- IRQ\_TIMER0 : [tonc\\_memdef.h](#)
- IRQ\_TIMER1 : [tonc\\_memdef.h](#)
- IRQ\_TIMER2 : [tonc\\_memdef.h](#)
- IRQ\_TIMER3 : [tonc\\_memdef.h](#)

- IRQ\_VBLANK : [tonc\\_memdef.h](#)
  - IRQ\_VCOUNTP : [tonc\\_memdef.h](#)
  - ISR\_DEF : [tonc\\_irq.h](#)
  - ISR\_LAST : [tonc\\_irq.h](#)
  - ISR\_PRIO : [tonc\\_irq.h](#)
  - ISR\_PRIO\_MASK : [tonc\\_irq.h](#)
  - ISR\_PRIO\_SHIFT : [tonc\\_irq.h](#)
  - ISR\_REPLACE : [tonc\\_irq.h](#)
  - IWRAM\_CODE : [tonc\\_types.h](#)
  - IWRAM\_DATA : [tonc\\_types.h](#)
- 

*Generated on Mon Aug 25 17:03:58 2008 for libtonc by*  *1.5.3*

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

## - k -

- KCNT\_AND : [tonc\\_memdef.h](#)
- KCNT\_IRQ : [tonc\\_memdef.h](#)
- KCNT\_OR : [tonc\\_memdef.h](#)
- KEY\_A : [tonc\\_memdef.h](#)
- KEY\_ACCEPT : [tonc\\_memdef.h](#)
- KEY\_ANY : [tonc\\_memdef.h](#)
- KEY\_B : [tonc\\_memdef.h](#)
- KEY\_CANCEL : [tonc\\_memdef.h](#)
- key\_curr\_state() : [tonc\\_input.h](#)
- KEY\_DIR : [tonc\\_memdef.h](#)
- KEY\_DOWN : [tonc\\_memdef.h](#)
- KEY\_FIRE : [tonc\\_memdef.h](#)
- KEY\_FULL : [tonc\\_input.h](#)
- key\_held() : [tonc\\_input.h](#)
- key\_hit() : [tonc\\_input.h](#)
- key\_is\_down() : [tonc\\_input.h](#)
- key\_is\_up() : [tonc\\_input.h](#)
- KEY\_L : [tonc\\_memdef.h](#)
- KEY\_LEFT : [tonc\\_memdef.h](#)
- key\_poll() : [tonc\\_input.h](#) , [tonc\\_input.c](#)
- key\_prev\_state() : [tonc\\_input.h](#)
- KEY\_R : [tonc\\_memdef.h](#)
- key\_released() : [tonc\\_input.h](#)

- key\_repeat() : [tonc\\_input.h](#) , [tonc\\_input.c](#)
- key\_repeat\_limits() : [tonc\\_input.h](#) , [tonc\\_input.c](#)
- key\_repeat\_mask() : [tonc\\_input.h](#) , [tonc\\_input.c](#)
- KEY\_RESET : [tonc\\_memdef.h](#)
- KEY\_RIGHT : [tonc\\_memdef.h](#)
- KEY\_SELECT : [tonc\\_memdef.h](#)
- KEY\_SHOULDER : [tonc\\_memdef.h](#)
- KEY\_SPECIAL : [tonc\\_memdef.h](#)
- KEY\_START : [tonc\\_memdef.h](#)
- key\_transit() : [tonc\\_input.h](#)
- key\_tri\_fire() : [tonc\\_input.h](#)
- key\_tri\_horz() : [tonc\\_input.h](#)
- key\_tri\_shoulder() : [tonc\\_input.h](#)
- key\_tri\_vert() : [tonc\\_input.h](#)
- KEY\_UP : [tonc\\_memdef.h](#)
- key\_wait\_till\_hit() : [tonc\\_input.h](#) , [tonc\\_input.c](#)
- key\_was\_down() : [tonc\\_input.h](#)
- key\_was\_up() : [tonc\\_input.h](#)

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

- | -

- lu\_cos() : [tonc\\_math.h](#)
- lu\_div() : [tonc\\_math.h](#)
- lu\_lerp16() : [tonc\\_math.h](#)
- lu\_lerp32() : [tonc\\_math.h](#)
- lu\_sin() : [tonc\\_math.h](#)

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

- m -

- M3\_CLEAR : [tonc\\_video.h](#)
- m3\_clrs() : [tonc\\_text.h](#)
- m3\_fill() : [tonc\\_video.h](#)
- m3\_frame() : [tonc\\_video.h](#)
- m3\_hline() : [tonc\\_video.h](#)
- m3\_line() : [tonc\\_video.h](#)
- m3\_mem : [tonc\\_memmap.h](#)
- m3\_plot() : [tonc\\_video.h](#)
- m3\_putc() : [tonc\\_text.h](#)
- m3\_puts() : [tonc\\_text.h](#)
- m3\_rect() : [tonc\\_video.h](#)
- M3\_SIZE : [tonc\\_memmap.h](#)
- m3\_vline() : [tonc\\_video.h](#)
- M4\_CLEAR : [tonc\\_video.h](#)
- m4\_clrs() : [tonc\\_text.h](#)
- m4\_fill() : [tonc\\_video.h](#)
- m4\_frame() : [tonc\\_video.h](#)
- m4\_hline() : [tonc\\_video.h](#)
- m4\_line() : [tonc\\_video.h](#)
- m4\_mem : [tonc\\_memmap.h](#)
- m4\_mem\_back : [tonc\\_memmap.h](#)
- m4\_plot() : [tonc\\_video.h](#)
- m4\_putc() : [tonc\\_text.h](#)

- m4\_puts() : **tonc\_text.h**
- m4\_rect() : **tonc\_video.h**
- M4\_SIZE : **tonc\_memmap.h**
- m4\_vline() : **tonc\_video.h**
- M5\_CLEAR : **tonc\_video.h**
- m5\_clrs() : **tonc\_text.h**
- m5\_fill() : **tonc\_video.h**
- m5\_frame() : **tonc\_video.h**
- m5\_hline() : **tonc\_video.h**
- m5\_line() : **tonc\_video.h**
- m5\_mem : **tonc\_memmap.h**
- m5\_mem\_back : **tonc\_memmap.h**
- m5\_plot() : **tonc\_video.h**
- m5\_putc() : **tonc\_text.h**
- m5\_puts() : **tonc\_text.h**
- m5\_rect() : **tonc\_video.h**
- M5\_SIZE : **tonc\_memmap.h**
- m5\_vline() : **tonc\_video.h**
- max() : **tonc\_math.h**
- MAX : **tonc\_math.h**
- MEM\_EWRAM : **tonc\_memmap.h**
- MEM\_IO : **tonc\_memmap.h**
- MEM\_IWRAM : **tonc\_memmap.h**
- MEM\_OAM : **tonc\_memmap.h**
- MEM\_PAL : **tonc\_memmap.h**
- MEM\_PAL\_BG : **tonc\_memmap.h**
- MEM\_PAL\_OBJ : **tonc\_memmap.h**
- MEM\_ROM : **tonc\_memmap.h**
- MEM\_SRAM : **tonc\_memmap.h**
- MEM\_VRAM : **tonc\_memmap.h**
- MEM\_VRAM\_BACK : **tonc\_memmap.h**
- MEM\_VRAM\_FRONT : **tonc\_memmap.h**
- MEM\_VRAM\_OBJ : **tonc\_memmap.h**

- `memcpy16()` : **tonc\_core.h**
  - `memcpy32()` : **tonc\_core.h**
  - `memset16()` : **tonc\_core.h**
  - `memset32()` : **tonc\_core.h**
  - `min()` : **tonc\_math.h**
  - `MIN` : **tonc\_math.h**
  - `Mod()` : **tonc\_bios.h**
- 

*Generated on Mon Aug 25 17:03:58 2008 for libtonc by*  **doxygen** 1.5.3

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

### - n -

- nocash\_buffer : [tonc\\_nocash.h](#)
- nocash\_message() : [tonc\\_nocash.h](#)
- nocash\_puts() : [tonc\\_nocash.h](#)

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

### - O -

- OAM\_CLEAR : [tonc\\_oam.h](#)
- oam\_copy() : [tonc\\_oam.h](#)
- oam\_init() : [tonc\\_oam.c](#) , [tonc\\_oam.h](#)
- oam\_mem : [tonc\\_memmap.h](#)
- obj\_aff\_identity() : [tonc\\_oam.h](#)
- obj\_aff\_mem : [tonc\\_memmap.h](#)
- OBJ\_AFF\_OFS : [tonc\\_bios.h](#)
- obj\_aff\_postmul() : [tonc\\_oam.h](#) , [tonc\\_obj\\_affine.c](#)
- obj\_aff\_premul() : [tonc\\_oam.h](#) , [tonc\\_obj\\_affine.c](#)
- obj\_aff\_rotate() : [tonc\\_oam.h](#) , [tonc\\_obj\\_affine.c](#)
- obj\_aff\_rotscale() : [tonc\\_obj\\_affine.c](#) , [tonc\\_oam.h](#)
- obj\_aff\_rotscale2() : [tonc\\_oam.h](#) , [tonc\\_obj\\_affine.c](#)
- obj\_aff\_scale() : [tonc\\_oam.h](#)
- obj\_aff\_set() : [tonc\\_oam.h](#)
- obj\_copy() : [tonc\\_oam.h](#) , [tonc\\_oam.c](#)
- obj\_drawg() : [tonc\\_tte.h](#) , [obj\\_drawg.c](#)
- obj\_erase() : [tonc\\_tte.h](#) , [obj\\_drawg.c](#)
- obj\_get\_height() : [tonc\\_oam.h](#)
- obj\_get\_size() : [tonc\\_oam.h](#)
- obj\_get\_width() : [tonc\\_oam.h](#)
- obj\_hide() : [tonc\\_oam.h](#)
- obj\_hide\_multi() : [tonc\\_oam.c](#) , [tonc\\_oam.h](#)
- obj\_mem : [tonc\\_memmap.h](#)

- obj\_putc2() : [tonc\\_text.h](#)
  - obj\_puts2() : [tonc\\_text.h](#)
  - obj\_rotscale\_ex() : [tonc\\_oam.h](#) , [tonc\\_obj\\_affine.c](#)
  - obj\_set\_attr() : [tonc\\_oam.h](#)
  - obj\_set\_pos() : [tonc\\_oam.h](#)
  - obj\_unhide() : [tonc\\_oam.h](#)
  - ObjAffineSet() : [tonc\\_bios.h](#)
  - octant() : [tonc\\_core.h](#) , [tonc\\_core.c](#)
  - octant\_rot() : [tonc\\_core.h](#) , [tonc\\_core.c](#)
  - octup() : [tonc\\_core.h](#)
- 

Generated on Mon Aug 25 17:03:58 2008 for libtonc by [!\[\]\(39a5cbdc321f4187fbf80623e505b6a3\_img.jpg\) doxygen](#) 1.5.3

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

- p -

- PACKED : [tonc\\_types.h](#)
- pal\_bg\_bank : [tonc\\_memmap.h](#)
- pal\_bg\_mem : [tonc\\_memmap.h](#)
- PAL\_BG\_SIZE : [tonc\\_memmap.h](#)
- pal\_gradient() : [tonc\\_video.h](#) , [tonc\\_color.c](#)
- pal\_gradient\_ex() : [tonc\\_color.c](#) , [tonc\\_video.h](#)
- pal\_obj\_bank : [tonc\\_memmap.h](#)
- pal\_obj\_mem : [tonc\\_memmap.h](#)
- PAL\_OBJ\_SIZE : [tonc\\_memmap.h](#)
- PALBANK : [tonc\\_types.h](#)
- profile\_start() : [tonc\\_core.h](#)
- profile\_stop() : [tonc\\_core.h](#)
- pt\_add() : [tonc\\_math.h](#)
- pt\_add\_eq() : [tonc\\_math.h](#)
- pt\_cross() : [tonc\\_math.h](#)
- pt\_dot() : [tonc\\_math.h](#)
- pt\_in\_rect() : [tonc\\_math.c](#)
- pt\_scale() : [tonc\\_math.h](#)
- pt\_scale\_eq() : [tonc\\_math.h](#)
- pt\_set() : [tonc\\_math.h](#)
- pt\_sub() : [tonc\\_math.h](#)
- pt\_sub\_eq() : [tonc\\_math.h](#)

*Generated on Mon Aug 25 17:03:58 2008 for libtonc by*  **doxygen** 1.5.3

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

- q -

- `qran()` : [tonc\\_core.h](#)
- `qran_range()` : [tonc\\_core.h](#)
- `quad8()` : [tonc\\_core.h](#)

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

- r -

- R\_MODE : [tonc\\_memdef.h](#)
- R\_MODE\_GPIO : [tonc\\_memdef.h](#)
- R\_MODE\_JOYBUS : [tonc\\_memdef.h](#)
- R\_MODE\_MASK : [tonc\\_memdef.h](#)
- R\_MODE\_MULTI : [tonc\\_memdef.h](#)
- R\_MODE\_NORMAL : [tonc\\_memdef.h](#)
- R\_MODE\_SHIFT : [tonc\\_memdef.h](#)
- R\_MODE\_UART : [tonc\\_memdef.h](#)
- RAM\_RESTART : [tonc\\_bios.h](#)
- rc\_height() : [tonc\\_math.h](#)
- rc\_inflate() : [tonc\\_math.h](#)
- rc\_inflate2() : [tonc\\_math.h](#)
- rc\_move() : [tonc\\_math.h](#)
- rc\_set() : [tonc\\_math.h](#)
- rc\_set2() : [tonc\\_math.h](#)
- rc\_set\_pos() : [tonc\\_math.h](#)
- rc\_set\_size() : [tonc\\_math.h](#)
- rc\_width() : [tonc\\_math.h](#)
- reflect() : [tonc\\_math.h](#)
- REFLECT : [tonc\\_math.h](#)
- REG\_BG0CNT : [tonc\\_memmap.h](#)
- REG\_BG0HOFS : [tonc\\_memmap.h](#)
- REG\_BG0VOFS : [tonc\\_memmap.h](#)

- REG\_BG1CNT : [tonc\\_memmap.h](#)
- REG\_BG1HOFS : [tonc\\_memmap.h](#)
- REG\_BG1VOFS : [tonc\\_memmap.h](#)
- REG\_BG2CNT : [tonc\\_memmap.h](#)
- REG\_BG2HOFS : [tonc\\_memmap.h](#)
- REG\_BG2PA : [tonc\\_memmap.h](#)
- REG\_BG2PB : [tonc\\_memmap.h](#)
- REG\_BG2PC : [tonc\\_memmap.h](#)
- REG\_BG2PD : [tonc\\_memmap.h](#)
- REG\_BG2VOFS : [tonc\\_memmap.h](#)
- REG\_BG2X : [tonc\\_memmap.h](#)
- REG\_BG2Y : [tonc\\_memmap.h](#)
- REG\_BG3CNT : [tonc\\_memmap.h](#)
- REG\_BG3HOFS : [tonc\\_memmap.h](#)
- REG\_BG3PA : [tonc\\_memmap.h](#)
- REG\_BG3PB : [tonc\\_memmap.h](#)
- REG\_BG3PC : [tonc\\_memmap.h](#)
- REG\_BG3PD : [tonc\\_memmap.h](#)
- REG\_BG3VOFS : [tonc\\_memmap.h](#)
- REG\_BG3X : [tonc\\_memmap.h](#)
- REG\_BG3Y : [tonc\\_memmap.h](#)
- REG\_BG\_AFFINE : [tonc\\_memmap.h](#)
- REG\_BG\_OFS : [tonc\\_memmap.h](#)
- REG\_BGCNT : [tonc\\_memmap.h](#)
- REG\_BLDALPHA : [tonc\\_memmap.h](#)
- REG\_BLDCNT : [tonc\\_memmap.h](#)
- REG\_BLDMOD : [tonc\\_memmap.h](#)
- REG\_BLDY : [tonc\\_memmap.h](#)
- REG\_DISPCNT : [tonc\\_memmap.h](#)
- REG\_DISPSTAT : [tonc\\_memmap.h](#)
- REG\_DMA : [tonc\\_memmap.h](#)
- REG\_DMA0CNT : [tonc\\_memmap.h](#)
- REG\_DMA0DAD : [tonc\\_memmap.h](#)

- REG\_DMA0SAD : **tonc\_memmap.h**
- REG\_DMA1CNT : **tonc\_memmap.h**
- REG\_DMA1DAD : **tonc\_memmap.h**
- REG\_DMA1SAD : **tonc\_memmap.h**
- REG\_DMA2CNT : **tonc\_memmap.h**
- REG\_DMA2DAD : **tonc\_memmap.h**
- REG\_DMA2SAD : **tonc\_memmap.h**
- REG\_DMA3CNT : **tonc\_memmap.h**
- REG\_DMA3DAD : **tonc\_memmap.h**
- REG\_DMA3SAD : **tonc\_memmap.h**
- REG\_FIFO\_A : **tonc\_memmap.h**
- REG\_FIFO\_B : **tonc\_memmap.h**
- REG\_IE : **tonc\_memmap.h**
- REG\_IF : **tonc\_memmap.h**
- REG\_IFBIOS : **tonc\_memmap.h**
- REG\_IME : **tonc\_memmap.h**
- REG\_ISR\_MAIN : **tonc\_memmap.h**
- REG\_JOY\_RECV : **tonc\_memmap.h**
- REG\_JOY\_TRANS : **tonc\_memmap.h**
- REG\_JOYCNT : **tonc\_memmap.h**
- REG\_JOYSTAT : **tonc\_memmap.h**
- REG\_KEYCNT : **tonc\_memmap.h**
- REG\_KEYINPUT : **tonc\_memmap.h**
- REG\_MOSAIC : **tonc\_memmap.h**
- REG\_PAUSE : **tonc\_memmap.h**
- REG\_RCNT : **tonc\_memmap.h**
- REG\_RESET\_DST : **tonc\_memmap.h**
- REG\_SIOCNT : **tonc\_memmap.h**
- REG\_SIODATA : **tonc\_memmap.h**
- REG\_SIODATA32 : **tonc\_memmap.h**
- REG\_SIODATA8 : **tonc\_memmap.h**
- REG\_SIOMLT\_RECV : **tonc\_memmap.h**
- REG\_SIOMLT\_SEND : **tonc\_memmap.h**

- REG\_SIOMULTI : [tonc\\_memmap.h](#)
- REG\_SIOMULTI0 : [tonc\\_memmap.h](#)
- REG\_SIOMULTI1 : [tonc\\_memmap.h](#)
- REG\_SIOMULTI2 : [tonc\\_memmap.h](#)
- REG\_SIOMULTI3 : [tonc\\_memmap.h](#)
- REG SND1CNT : [tonc\\_memmap.h](#)
- REG SND1FREQ : [tonc\\_memmap.h](#)
- REG SND1SWEEP : [tonc\\_memmap.h](#)
- REG SND2CNT : [tonc\\_memmap.h](#)
- REG SND2FREQ : [tonc\\_memmap.h](#)
- REG SND3CNT : [tonc\\_memmap.h](#)
- REG SND3FREQ : [tonc\\_memmap.h](#)
- REG SND3SEL : [tonc\\_memmap.h](#)
- REG SND4CNT : [tonc\\_memmap.h](#)
- REG SND4FREQ : [tonc\\_memmap.h](#)
- REG SNDBIAS : [tonc\\_memmap.h](#)
- REG SNDSCNT : [tonc\\_memmap.h](#)
- REG SNDDMGCNT : [tonc\\_memmap.h](#)
- REG SNDDSCNT : [tonc\\_memmap.h](#)
- REG SNDSTAT : [tonc\\_memmap.h](#)
- REG TM : [tonc\\_memmap.h](#)
- REG TM0CNT : [tonc\\_memmap.h](#)
- REG TM0D : [tonc\\_memmap.h](#)
- REG TM1CNT : [tonc\\_memmap.h](#)
- REG TM1D : [tonc\\_memmap.h](#)
- REG TM2CNT : [tonc\\_memmap.h](#)
- REG TM2D : [tonc\\_memmap.h](#)
- REG TM3CNT : [tonc\\_memmap.h](#)
- REG TM3D : [tonc\\_memmap.h](#)
- REG VCOUNT : [tonc\\_memmap.h](#)
- REG WAITCNT : [tonc\\_memmap.h](#)
- REG WAVE\_RAM : [tonc\\_memmap.h](#)
- REG WAVE\_RAM0 : [tonc\\_memmap.h](#)

- REG\_WAVE\_RAM1 : [tonc\\_memmap.h](#)
- REG\_WAVE\_RAM2 : [tonc\\_memmap.h](#)
- REG\_WAVE\_RAM3 : [tonc\\_memmap.h](#)
- REG\_WIN0B : [tonc\\_memmap.h](#)
- REG\_WIN0CNT : [tonc\\_memmap.h](#)
- REG\_WIN0H : [tonc\\_memmap.h](#)
- REG\_WIN0L : [tonc\\_memmap.h](#)
- REG\_WIN0R : [tonc\\_memmap.h](#)
- REG\_WIN0T : [tonc\\_memmap.h](#)
- REG\_WIN0V : [tonc\\_memmap.h](#)
- REG\_WIN1B : [tonc\\_memmap.h](#)
- REG\_WIN1CNT : [tonc\\_memmap.h](#)
- REG\_WIN1H : [tonc\\_memmap.h](#)
- REG\_WIN1L : [tonc\\_memmap.h](#)
- REG\_WIN1R : [tonc\\_memmap.h](#)
- REG\_WIN1T : [tonc\\_memmap.h](#)
- REG\_WIN1V : [tonc\\_memmap.h](#)
- REG\_WININ : [tonc\\_memmap.h](#)
- REG\_WINOBJCNT : [tonc\\_memmap.h](#)
- REG\_WINOUT : [tonc\\_memmap.h](#)
- REG\_WINOUTCNT : [tonc\\_memmap.h](#)
- RESET\_EWRAM : [tonc\\_bios.h](#)
- RESET\_GFX : [tonc\\_bios.h](#)
- RESET\_IWRAM : [tonc\\_bios.h](#)
- RESET\_MEM\_MASK : [tonc\\_bios.h](#)
- RESET\_OAM : [tonc\\_bios.h](#)
- RESET\_PALETTE : [tonc\\_bios.h](#)
- RESET\_REG : [tonc\\_bios.h](#)
- RESET\_REG\_MASK : [tonc\\_bios.h](#)
- RESET\_REG\_SIO : [tonc\\_bios.h](#)
- RESET\_REG\_SOUND : [tonc\\_bios.h](#)
- RESET\_VRAM : [tonc\\_bios.h](#)
- RGB15() : [tonc\\_video.h](#)

- RGB15\_SAFE() : [tonc\\_video.h](#)
  - RGB8() : [tonc\\_video.h](#)
  - rom\_mem : [tonc\\_memmap.h](#)
  - ROM\_RESTART : [tonc\\_bios.h](#)
  - ROR() : [tonc\\_core.h](#)
- 

Generated on Mon Aug 25 17:03:58 2008 for libtonc by  doxygen 1.5.3

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

- S -

- SAE : [tonc\\_types.h](#)
- SBB\_SIZE : [tonc\\_memmap.h](#)
- sbmp16.blit() : [tonc\\_sbmp16.c](#) , [tonc\\_surface.h](#)
- sbmp16.floodfill() : [tonc\\_surface.h](#) , [tonc\\_sbmp16.c](#)
- sbmp16.floodfill\_internal() : [tonc\\_sbmp16.c](#)
- sbmp16.frame() : [tonc\\_surface.h](#) , [tonc\\_sbmp16.c](#)
- sbmp16.get\_pixel() : [tonc\\_surface.h](#) , [tonc\\_sbmp16.c](#)
- sbmp16.hline() : [tonc\\_surface.h](#) , [tonc\\_sbmp16.c](#)
- sbmp16.line() : [tonc\\_surface.h](#) , [tonc\\_sbmp16.c](#)
- sbmp16.plot() : [tonc\\_surface.h](#) , [tonc\\_sbmp16.c](#)
- sbmp16.rect() : [tonc\\_surface.h](#) , [tonc\\_sbmp16.c](#)
- sbmp16.vline() : [tonc\\_surface.h](#) , [tonc\\_sbmp16.c](#)
- sbmp8.blit() : [tonc\\_surface.h](#) , [tonc\\_sbmp8.c](#)
- sbmp8.floodfill() : [tonc\\_surface.h](#) , [tonc\\_sbmp8.c](#)
- sbmp8.floodfill\_internal() : [tonc\\_sbmp8.c](#)
- sbmp8.frame() : [tonc\\_surface.h](#) , [tonc\\_sbmp8.c](#)
- sbmp8.get\_pixel() : [tonc\\_surface.h](#) , [tonc\\_sbmp8.c](#)
- sbmp8.hline() : [tonc\\_sbmp8.c](#) , [tonc\\_surface.h](#)
- sbmp8.line() : [tonc\\_surface.h](#) , [tonc\\_sbmp8.c](#)
- sbmp8.plot() : [tonc\\_surface.h](#) , [tonc\\_sbmp8.c](#)
- sbmp8.rect() : [tonc\\_surface.h](#) , [tonc\\_sbmp8.c](#)
- sbmp8.vline() : [tonc\\_surface.h](#) , [tonc\\_sbmp8.c](#)
- schr4c.blit() : [tonc\\_surface.h](#) , [tonc\\_schr4c.c](#)

- `schr4c_floodfill()` : `tonc_surface.h` , `tonc_schr4c.c`
- `schr4c_floodfill_internal()` : `tonc_schr4c.c`
- `schr4c_frame()` : `tonc_surface.h` , `tonc_schr4c.c`
- `schr4c_get_pixel()` : `tonc_surface.h` , `tonc_schr4c.c`
- `schr4c_get_ptr()` : `tonc_surface.h` , `tonc_schr4c.c`
- `schr4c_hline()` : `tonc_surface.h` , `tonc_schr4c.c`
- `schr4c_line()` : `tonc_surface.h` , `tonc_schr4c.c`
- `schr4c_plot()` : `tonc_surface.h` , `tonc_schr4c.c`
- `schr4c_prep_map()` : `tonc_surface.h` , `tonc_schr4c.c`
- `schr4c_rect()` : `tonc_surface.h` , `tonc_schr4c.c`
- `schr4c_vline()` : `tonc_surface.h` , `tonc_schr4c.c`
- `schr4r_frame()` : `tonc_surface.h` , `tonc_schr4r.c`
- `schr4r_get_pixel()` : `tonc_surface.h` , `tonc_schr4r.c`
- `schr4r_get_ptr()` : `tonc_surface.h` , `tonc_schr4r.c`
- `schr4r_hline()` : `tonc_surface.h` , `tonc_schr4r.c`
- `schr4r_line()` : `tonc_surface.h` , `tonc_schr4r.c`
- `schr4r_plot()` : `tonc_surface.h` , `tonc_schr4r.c`
- `schr4r_prep_map()` : `tonc_surface.h` , `tonc_schr4r.c`
- `schr4r_rect()` : `tonc_surface.h` , `tonc_schr4r.c`
- `schr4r_vline()` : `tonc_surface.h` , `tonc_schr4r.c`
- `SDMG_LNOISE` : `tonc_memdef.h`
- `SDMG_LSQR1` : `tonc_memdef.h`
- `SDMG_LSQR2` : `tonc_memdef.h`
- `SDMG_LWAVE` : `tonc_memdef.h`
- `SDMG_RNOISE` : `tonc_memdef.h`
- `SDMG_RSQR1` : `tonc_memdef.h`
- `SDMG_RSQR2` : `tonc_memdef.h`
- `SDMG_RWAVE` : `tonc_memdef.h`
- `SDS_A100` : `tonc_memdef.h`
- `SDS_A50` : `tonc_memdef.h`
- `SDS_AL` : `tonc_memdef.h`
- `SDS_AR` : `tonc_memdef.h`
- `SDS_ARESET` : `tonc_memdef.h`

- SDS\_ATMR0 : **tonc\_memdef.h**
- SDS\_ATMR1 : **tonc\_memdef.h**
- SDS\_B100 : **tonc\_memdef.h**
- SDS\_B50 : **tonc\_memdef.h**
- SDS\_BL : **tonc\_memdef.h**
- SDS\_BR : **tonc\_memdef.h**
- SDS\_BRESET : **tonc\_memdef.h**
- SDS\_BTMR0 : **tonc\_memdef.h**
- SDS\_BTMR1 : **tonc\_memdef.h**
- SDS\_DMG100 : **tonc\_memdef.h**
- SDS\_DMG25 : **tonc\_memdef.h**
- SDS\_DMG50 : **tonc\_memdef.h**
- SE : **tonc\_types.h**
- se\_drawg() : **tonc\_tte.h , se\_drawg.c**
- se\_drawg\_s() : **tonc\_tte.h , se\_drawg.c**
- se\_drawg\_w8h16() : **tonc\_tte.h , se\_drawg.c**
- se\_drawg\_w8h8() : **tonc\_tte.h , se\_drawg.c**
- se\_erase() : **tonc\_tte.h , se\_drawg.c**
- se\_fill() : **tonc\_video.h**
- se\_frame() : **tonc\_video.h**
- SE\_HFLIP : **tonc\_memdef.h**
- se\_mat : **tonc\_memmap.h**
- se\_mem : **tonc\_memmap.h**
- se\_plot() : **tonc\_video.h**
- se\_rect() : **tonc\_video.h**
- SE\_VFLIP : **tonc\_memdef.h**
- se\_window() : **tonc\_bg.c , tonc\_video.h**
- SFREQ\_HOLD : **tonc\_memdef.h**
- SFREQ\_RESET : **tonc\_memdef.h**
- SFREQ\_TIMED : **tonc\_memdef.h**
- sgn() : **tonc\_math.h**
- SGN : **tonc\_math.h**
- SGN2 : **tonc\_math.h**

- SGN3 : **tonc\_math.h**
- sgn3() : **tonc\_math.h**
- SIN\_LUT\_SIZE : **tonc\_math.h**
- SIO\_2MHZ\_CLK : **tonc\_libgba.h**
- SIO\_32BIT : **tonc\_libgba.h**
- SIO\_8BIT : **tonc\_libgba.h**
- SIO\_CLK\_INT : **tonc\_libgba.h**
- SIO\_IRQ : **tonc\_libgba.h , tonc\_memdef.h**
- SIO\_MODE : **tonc\_memdef.h**
- SIO\_MODE\_32BIT : **tonc\_memdef.h**
- SIO\_MODE\_8BIT : **tonc\_memdef.h**
- SIO\_MODE\_MASK : **tonc\_memdef.h**
- SIO\_MODE\_MULTI : **tonc\_memdef.h**
- SIO\_MODE\_SHIFT : **tonc\_memdef.h**
- SIO\_MODE\_UART : **tonc\_memdef.h**
- SIO\_MULTI : **tonc\_libgba.h**
- SIO\_RDY : **tonc\_libgba.h**
- SIO\_SI\_HIGH : **tonc\_memdef.h**
- SIO\_SO\_HIGH : **tonc\_libgba.h**
- SIO\_UART : **tonc\_libgba.h**
- SIOM\_115200 : **tonc\_memdef.h**
- SIOM\_38400 : **tonc\_memdef.h**
- SIOM\_57600 : **tonc\_memdef.h**
- SIOM\_9600 : **tonc\_memdef.h**
- SIOM\_BAUD : **tonc\_memdef.h**
- SIOM\_BAUD\_MASK : **tonc\_memdef.h**
- SIOM\_BAUD\_SHIFT : **tonc\_memdef.h**
- SIOM\_CONNECTED : **tonc\_memdef.h**
- SIOM\_ENABLE : **tonc\_memdef.h**
- SIOM\_ERROR : **tonc\_memdef.h**
- SIOM\_ID : **tonc\_memdef.h**
- SIOM\_ID\_MASK : **tonc\_memdef.h**
- SIOM\_ID\_SHIFT : **tonc\_memdef.h**

- SIOM\_SD : **tonc\_memdef.h**
- SIOM\_SI : **tonc\_memdef.h**
- SIOM\_SLAVE : **tonc\_memdef.h**
- SION\_256KHZ : **tonc\_memdef.h**
- SION\_2MHZ : **tonc\_memdef.h**
- SION\_CLK\_EXT : **tonc\_memdef.h**
- SION\_CLK\_INT : **tonc\_memdef.h**
- SION\_ENABLE : **tonc\_memdef.h**
- SION\_RECV\_HIGH : **tonc\_memdef.h**
- SION\_SEND\_HIGH : **tonc\_memdef.h**
- SIOU\_115200 : **tonc\_memdef.h**
- SIOU\_38400 : **tonc\_memdef.h**
- SIOU\_57600 : **tonc\_memdef.h**
- SIOU\_7BIT : **tonc\_memdef.h**
- SIOU\_8BIT : **tonc\_memdef.h**
- SIOU\_9600 : **tonc\_memdef.h**
- SIOU\_BAUD : **tonc\_memdef.h**
- SIOU\_BAUD\_MASK : **tonc\_memdef.h**
- SIOU\_BAUD\_SHIFT : **tonc\_memdef.h**
- SIOU\_CTS : **tonc\_memdef.h**
- SIOU\_ERROR : **tonc\_memdef.h**
- SIOU\_PARITY\_EVEN : **tonc\_memdef.h**
- SIOU\_PARITY\_ODD : **tonc\_memdef.h**
- SIOU\_RECV : **tonc\_memdef.h**
- SIOU\_RECV\_EMPTY : **tonc\_memdef.h**
- SIOU\_SEND : **tonc\_memdef.h**
- SIOU\_SEND\_FULL : **tonc\_memdef.h**
- SND\_RATE : **tonc\_core.h**
- Sqrt() : **tonc\_bios.h**
- sram\_mem : **tonc\_memmap.h**
- srf\_align() : **tonc\_surface.h**
- SRF\_ALLOCATED : **tonc\_surface.h**
- SRF\_BMP16 : **tonc\_surface.h**

- SRF\_BMP8 : [tonc\\_surface.h](#)
  - SRF\_CHR4C : [tonc\\_surface.h](#)
  - SRF\_CHR4R : [tonc\\_surface.h](#)
  - SRF\_CHR8 : [tonc\\_surface.h](#)
  - srf\_get\_ptr() : [tonc\\_surface.c](#) , [tonc\\_surface.h](#)
  - srf\_init() : [tonc\\_surface.c](#) , [tonc\\_surface.h](#)
  - SRF\_NONE : [tonc\\_surface.h](#)
  - srf\_pal\_copy() : [tonc\\_surface.c](#) , [tonc\\_surface.h](#)
  - srf\_set\_pal() : [tonc\\_surface.h](#)
  - srf\_set\_ptr() : [tonc\\_surface.h](#)
  - SSQR\_DEC : [tonc\\_memdef.h](#)
  - SSQR\_DUTY1\_2 : [tonc\\_memdef.h](#)
  - SSQR\_DUTY1\_4 : [tonc\\_memdef.h](#)
  - SSQR\_DUTY1\_8 : [tonc\\_memdef.h](#)
  - SSQR\_DUTY3\_4 : [tonc\\_memdef.h](#)
  - SSQR\_INC : [tonc\\_memdef.h](#)
  - SSTAT\_DISABLE : [tonc\\_memdef.h](#)
  - SSTAT\_ENABLE : [tonc\\_memdef.h](#)
  - SSTAT\_NOISE : [tonc\\_memdef.h](#)
  - SSTAT\_SQR1 : [tonc\\_memdef.h](#)
  - SSTAT\_SQR2 : [tonc\\_memdef.h](#)
  - SSTAT\_WAVE : [tonc\\_memdef.h](#)
  - SSW\_DEC : [tonc\\_memdef.h](#)
  - SSW\_INC : [tonc\\_memdef.h](#)
  - SSW\_OFF : [tonc\\_memdef.h](#)
  - STR : [tonc\\_core.h](#)
  - SWAP : [tonc\\_math.h](#)
  - SWAP2 : [tonc\\_math.h](#)
  - SWAP3 : [tonc\\_math.h](#)
  - swi\_call : [tonc\\_bios.h](#)
  - sys8Font : [tonc\\_tte.h](#)
  - sys8Glyphs : [tonc\\_tte.h](#)
-

*Generated on Mon Aug 25 17:03:58 2008 for libtonc by*  **doxygen** 1.5.3

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

### - t -

- tile8\_mem : [tonc\\_memmap.h](#)
- tile8\_mem\_obj : [tonc\\_memmap.h](#)
- tile\_mem : [tonc\\_memmap.h](#)
- tile\_mem\_obj : [tonc\\_memmap.h](#)
- TM CASCADE : [tonc\\_memdef.h](#)
- TM\_ENABLE : [tonc\\_memdef.h](#)
- TM\_FREQ\_1 : [tonc\\_memdef.h](#)
- TM\_FREQ\_1024 : [tonc\\_memdef.h](#)
- TM\_FREQ\_256 : [tonc\\_memdef.h](#)
- TM\_FREQ\_64 : [tonc\\_memdef.h](#)
- TM\_FREQ\_SYS : [tonc\\_memdef.h](#)
- TM\_IRQ : [tonc\\_memdef.h](#)
- tonccpy() : [tonc\\_core.h](#) , [tonc\\_core.c](#)
- toncset() : [tonc\\_core.h](#)
- toncset16() : [tonc\\_core.h](#)
- toncset32() : [tonc\\_core.h](#)
- TTE\_BASE\_VARS : [tonc\\_tte.h](#)
- TTE\_CHAR\_VARS : [tonc\\_tte.h](#)
- tte\_cmd\_default() : [tonc\\_tte.h](#) , [tte\\_main.c](#)
- tte\_cmd\_next() : [tte\\_main.c](#)
- tte\_cmd\_skip() : [tte\\_main.c](#)
- tte\_cmd\_vt100() : [tonc\\_tte.h](#) , [tte\\_iohook.c](#)
- tte\_con\_write() : [tonc\\_tte.h](#) , [tte\\_iohook.c](#)

- TTE\_DST\_VARS : **tonc\_tte.h**
- tte\_erase\_line() : **tonc\_tte.h** , **tte\_main.c**
- tte\_erase\_rect() : **tte\_main.c** , **tonc\_tte.h**
- tte\_erase\_screen() : **tonc\_tte.h** , **tte\_main.c**
- tte\_get\_context() : **tonc\_tte.h**
- tte\_get\_drawg() : **tonc\_tte.h**
- tte\_get\_erase() : **tonc\_tte.h**
- tte\_get\_font() : **tonc\_tte.h**
- tte\_get\_font\_table() : **tonc\_tte.h**
- tte\_get\_glyph\_data() : **tonc\_tte.h**
- tte\_get\_glyph\_height() : **tonc\_tte.h**
- tte\_get\_glyph\_id() : **tonc\_tte.h**
- tte\_get\_glyph\_width() : **tonc\_tte.h**
- tte\_get\_ink() : **tonc\_tte.h**
- tte\_get\_paper() : **tonc\_tte.h**
- tte\_get\_pos() : **tonc\_tte.h**
- tte\_get\_shadow() : **tonc\_tte.h**
- tte\_get\_special() : **tonc\_tte.h**
- tte\_get\_string\_table() : **tonc\_tte.h**
- tte\_get\_surface() : **tonc\_tte.h**
- tte\_get\_text\_size() : **tonc\_tte.h** , **tte\_main.c**
- tte\_init\_ase() : **tonc\_tte.h** , **tte\_init\_ase.c**
- tte\_init\_base() : **tonc\_tte.h** , **tte\_main.c**
- tte\_init\_bmp() : **tonc\_tte.h** , **tte\_init\_bmp.c**
- tte\_init\_chr4c() : **tonc\_tte.h** , **tte\_init\_chr4c.c**
- tte\_init\_chr4r() : **tonc\_tte.h** , **tte\_init\_chr4r.c**
- tte\_init\_con() : **tonc\_tte.h** , **tte\_iohook.c**
- tte\_init\_obj() : **tonc\_tte.h** , **tte\_init\_obj.c**
- tte\_init\_se() : **tte\_init\_se.c** , **tonc\_tte.h**
- tte\_printf : **tonc\_tte.h**
- tte\_putc() : **tonc\_tte.h** , **tte\_main.c**
- tte\_set\_color() : **tonc\_tte.h** , **tte\_main.c**
- tte\_set\_color\_attr() : **tonc\_tte.h** , **tte\_main.c**

- `tte_set_color_attrs()` : **tonc\_tte.h** , **tte\_main.c**
  - `tte_set_colors()` : **tte\_main.c** , **tonc\_tte.h**
  - `tte_set_context()` : **tonc\_tte.h** , **tte\_main.c**
  - `tte_set_drawg()` : **tonc\_tte.h**
  - `tte_set_erase()` : **tonc\_tte.h**
  - `tte_set_font()` : **tonc\_tte.h**
  - `tte_set_font_table()` : **tonc\_tte.h**
  - `tte_set_ink()` : **tonc\_tte.h**
  - `tte_set_paper()` : **tonc\_tte.h**
  - `tte_set_pos()` : **tonc\_tte.h**
  - `tte_set_shadow()` : **tonc\_tte.h**
  - `tte_set_special()` : **tonc\_tte.h**
  - `tte_set_string_table()` : **tonc\_tte.h**
  - `tte_set_surface()` : **tonc\_tte.h**
  - `TTE_TAB_WIDTH` : **tonc\_tte.h**
  - `tte_write()` : **tte\_main.c** , **tonc\_tte.h**
  - `tte_write_ex()` : **tte\_main.c** , **tonc\_tte.h**
  - `txt_init_se()` : **tonc\_text.h**
  - `txt_init_std()` : **tonc\_text.h**
-

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

- **U** -

- `utf8_decode_char()` : [tte\\_iohook.c](#) , [tte\\_main.c](#)

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

- v -

- VBlankIntrDelay() : [tonc\\_bios.h](#)
- VBlankIntrWait() : [tonc\\_bios.h](#)
- vec\_add() : [tonc\\_math.h](#)
- vec\_add\_eq() : [tonc\\_math.h](#)
- vec\_dot() : [tonc\\_math.h](#)
- vec\_mul() : [tonc\\_math.h](#)
- vec\_mul\_eq() : [tonc\\_math.h](#)
- vec\_scale() : [tonc\\_math.h](#)
- vec\_scale\_eq() : [tonc\\_math.h](#)
- vec\_set() : [tonc\\_math.h](#)
- vec\_sub() : [tonc\\_math.h](#)
- vec\_sub\_eq() : [tonc\\_math.h](#)
- verdana10Font : [tonc\\_tte.h](#)
- verdana10Glyphs : [tonc\\_tte.h](#)
- verdana10Widths : [tonc\\_tte.h](#)
- verdana9\_b4Font : [tonc\\_tte.h](#)
- verdana9\_b4Glyphs : [tonc\\_tte.h](#)
- verdana9\_b4Widths : [tonc\\_tte.h](#)
- verdana9bFont : [tonc\\_tte.h](#)
- verdana9bGlyphs : [tonc\\_tte.h](#)
- verdana9bWidths : [tonc\\_tte.h](#)
- verdana9Font : [tonc\\_tte.h](#)
- verdana9Glyphs : [tonc\\_tte.h](#)

- verdana9iFont : [tonc\\_tte.h](#)
  - verdana9iGlyphs : [tonc\\_tte.h](#)
  - verdana9iWidths : [tonc\\_tte.h](#)
  - verdana9Widths : [tonc\\_tte.h](#)
  - vid\_flip() : [tonc\\_video.c](#) , [tonc\\_video.h](#)
  - vid\_mem : [tonc\\_memmap.h](#)
  - vid\_mem\_back : [tonc\\_memmap.h](#)
  - vid\_mem\_front : [tonc\\_memmap.h](#)
  - VRAM\_BG\_SIZE : [tonc\\_memmap.h](#)
  - VRAM\_OBJ\_SIZE : [tonc\\_memmap.h](#)
  - VRAM\_PAGE\_SIZE : [tonc\\_memmap.h](#)
  - vwf\_default : [tonc\\_tte.h](#)
- 

Generated on Mon Aug 25 17:03:58 2008 for libtonc by  1.5.3

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

- W -

- WIN\_ALL : [tonc\\_memdef.h](#)
- WIN\_BG0 : [tonc\\_memdef.h](#)
- WIN\_BG1 : [tonc\\_memdef.h](#)
- WIN\_BG2 : [tonc\\_memdef.h](#)
- WIN\_BG3 : [tonc\\_memdef.h](#)
- WIN\_BLD : [tonc\\_memdef.h](#)
- WIN\_BUILD : [tonc\\_memdef.h](#)
- WIN\_LAYER : [tonc\\_memdef.h](#)
- WIN\_LAYER\_MASK : [tonc\\_memdef.h](#)
- WIN\_LAYER\_SHIFT : [tonc\\_memdef.h](#)
- WIN\_OBJ : [tonc\\_memdef.h](#)
- WININ\_BUILD : [tonc\\_memdef.h](#)
- WINOUT\_BUILD : [tonc\\_memdef.h](#)
- WRAP : [tonc\\_math.h](#)
- wrap() : [tonc\\_math.h](#)
- WS\_SRAM\_4 : [tonc\\_memdef.h](#)

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

- X -

- XSTR : [tonc\\_core.h](#)

## - a -

- align() : [tonc\\_core.h](#)
- ArcTan() : [tonc\\_bios.h](#)
- ArcTan2() : [tonc\\_bios.h](#)
- ase\_drawg() : [tonc\\_tte.h](#) , [ase\\_drawg.c](#)
- ase\_drawg\_s() : [tonc\\_tte.h](#) , [ase\\_drawg.c](#)
- ase\_drawg\_w8h16() : [tonc\\_tte.h](#) , [ase\\_drawg.c](#)
- ase\_drawg\_w8h8() : [ase\\_drawg.c](#) , [tonc\\_tte.h](#)
- ase\_erase() : [ase\\_drawg.c](#) , [tonc\\_tte.h](#)

## - b -

- `bf_clamp()` : [tonc\\_core.h](#)
- `bf_get()` : [tonc\\_core.h](#)
- `bf_merge()` : [tonc\\_core.h](#)
- `bg_aff_copy()` : [tonc\\_video.h](#)
- `bg_aff_identity()` : [tonc\\_video.h](#)
- `bg_aff_postmul()` : [tonc\\_video.h](#) , [tonc\\_bg\\_affine.c](#)
- `bg_aff_premul()` : [tonc\\_video.h](#) , [tonc\\_bg\\_affine.c](#)
- `bg_aff_rotate()` : [tonc\\_video.h](#) , [tonc\\_bg\\_affine.c](#)
- `bg_aff_rotscale()` : [tonc\\_video.h](#) , [tonc\\_bg\\_affine.c](#)
- `bg_aff_rotscale2()` : [tonc\\_video.h](#) , [tonc\\_bg\\_affine.c](#)
- `bg_aff_scale()` : [tonc\\_video.h](#)
- `bg_aff_set()` : [tonc\\_video.h](#)
- `bg_rotscale_ex()` : [tonc\\_video.h](#) , [tonc\\_bg\\_affine.c](#)
- `BgAffineSet()` : [tonc\\_bios.h](#)
- `bit_tribool()` : [tonc\\_core.h](#)
- `bmp16_drawg()` : [tonc\\_tte.h](#) , [bmp16\\_drawg.c](#)
- `bmp16_drawg_b1cos()` : [tonc\\_tte.h](#) ,  
[bmp16\\_drawg\\_b1cs.c](#)
- `bmp16_drawg_b1cts()` : [tonc\\_tte.h](#) ,  
[bmp16\\_drawg\\_b1cs.c](#)
- `bmp16_drawg_t()` : [tonc\\_tte.h](#) , [bmp16\\_drawg.c](#)
- `bmp16_erase()` : [tonc\\_tte.h](#) , [tte\\_init bmp.c](#)
- `bmp16_frame()` : [tonc\\_video.h](#) , [tonc\\_bmp16.c](#)
- `bmp16_hline()` : [tonc\\_video.h](#) , [tonc\\_bmp16.c](#)

- `bmp16_line()` : [tonc\\_video.h](#) , [tonc\\_bmp16.c](#)
- `bmp16_plot()` : [tonc\\_video.h](#) , [tonc\\_bmp16.c](#)
- `bmp16_rect()` : [tonc\\_video.h](#) , [tonc\\_bmp16.c](#)
- `bmp16_vline()` : [tonc\\_video.h](#) , [tonc\\_bmp16.c](#)
- `bmp8_drawg()` : [tonc\\_tte.h](#) , [bmp8\\_drawg.c](#)
- `bmp8_drawg_b1cos()` : [tonc\\_tte.h](#)
- `bmp8_drawg_b1cts()` : [tonc\\_tte.h](#)
- `bmp8_drawg_b1cts_fast()` : [tonc\\_tte.h](#)
- `bmp8_drawg_t()` : [bmp8\\_drawg.c](#) , [tonc\\_tte.h](#)
- `bmp8_erase()` : [tte\\_init bmp.c](#) , [tonc\\_tte.h](#)
- `bmp8_frame()` : [tonc\\_video.h](#) , [tonc\\_bmp8.c](#)
- `bmp8_hline()` : [tonc\\_bmp8.c](#) , [tonc\\_video.h](#)
- `bmp8_line()` : [tonc\\_video.h](#) , [tonc\\_bmp8.c](#)
- `bmp8_plot()` : [tonc\\_video.h](#) , [tonc\\_bmp8.c](#)
- `bmp8_rect()` : [tonc\\_bmp8.c](#) , [tonc\\_video.h](#)
- `bmp8_vline()` : [tonc\\_bmp8.c](#) , [tonc\\_video.h](#)
- `bytes2hword()` : [tonc\\_core.h](#)
- `bytes2word()` : [tonc\\_core.h](#)

## - C -

- chr4\_lmask() : [tonc\\_schr4c.c](#) , [tonc\\_schr4r.c](#)
- chr4\_rmask() : [tonc\\_schr4r.c](#) , [tonc\\_schr4c.c](#)
- chr4c\_colset() : [tonc\\_schr4c.c](#)
- chr4c\_drawg\_b1cts() : [tonc\\_tte.h](#) , [chr4c\\_drawg\\_b1cts.c](#)
- chr4c\_drawg\_b1cts\_fast() : [tonc\\_tte.h](#)
- chr4c\_drawg\_b4cts() : [tonc\\_tte.h](#)
- chr4c\_drawg\_b4cts\_fast() : [tonc\\_tte.h](#)
- chr4c\_erase() : [tonc\\_tte.h](#) , [tte\\_init\\_chr4c.c](#)
- chr4r\_colset() : [tonc\\_schr4r.c](#)
- chr4r\_drawg\_b1cts() : [tonc\\_tte.h](#) , [chr4r\\_drawg\\_b1cts.c](#)
- chr4r\_drawg\_b1cts\_fast() : [tonc\\_tte.h](#)
- chr4r\_erase() : [tonc\\_tte.h](#) , [tte\\_init\\_chr4r.c](#)
- clamp() : [tonc\\_math.h](#)
- clr\_adj\_brightness() : [tonc\\_video.h](#) , [tonc\\_color.c](#)
- clr\_adj\_contrast() : [tonc\\_color.c](#) , [tonc\\_video.h](#)
- clr\_adj\_intensity() : [tonc\\_video.h](#) , [tonc\\_color.c](#)
- clr\_blend() : [tonc\\_color.c](#) , [tonc\\_video.h](#)
- clr\_blend\_fast() : [tonc\\_video.h](#)
- clr\_fade() : [tonc\\_color.c](#) , [tonc\\_video.h](#)
- clr\_fade\_fast() : [tonc\\_video.h](#)
- clr\_grayscale() : [tonc\\_video.h](#) , [tonc\\_color.c](#)
- clr\_rgbscale() : [tonc\\_color.c](#) , [tonc\\_video.h](#)
- clr\_rotate() : [tonc\\_color.c](#) , [tonc\\_video.h](#)
- CpuFastFill() : [tonc\\_bios.h](#)

- CpuFastSet() : **tonc\_bios.h**
  - CpuSet() : **tonc\_bios.h**
- 

*Generated on Mon Aug 25 17:03:58 2008 for libtonc by*  **doxygen** 1.5.3

>Main Page   Modules   Data Structures   Files   Related Pages

File List   Globals

All   Functions   Variables   Typedefs   Enumerations   Enumerator   Defines

\_ a b c d f h i k l m n o p q r s t u v w

## - d -

- Div() : [tonc\\_bios.h](#)
- DivAbs() : [tonc\\_bios.h](#)
- DivArm() : [tonc\\_bios.h](#)
- DivArmAbs() : [tonc\\_bios.h](#)
- DivArmMod() : [tonc\\_bios.h](#)
- DivSafe() : [tonc\\_bios.h](#)
- dma3\_cpy() : [tonc\\_core.h](#)
- dma3\_fill() : [tonc\\_core.h](#)
- dma\_cpy() : [tonc\\_core.h](#)
- dma\_fill() : [tonc\\_core.h](#)
- dup16() : [tonc\\_core.h](#)
- dup8() : [tonc\\_core.h](#)

>Main Page   Modules   Data Structures   Files   Related Pages

File List   Globals

All   Functions   Variables   Typedefs   Enumerations   Enumerator   Defines

\_   a   b   c   d   f   h   i   k   l   m   n   o   p   q   r   s   t   u   v   w

## - f -

- float2fx() : [tonc\\_math.h](#)
- fx2float() : [tonc\\_math.h](#)
- fx2int() : [tonc\\_math.h](#)
- fx2ufrac() : [tonc\\_math.h](#)
- fx2uint() : [tonc\\_math.h](#)
- fxadd() : [tonc\\_math.h](#)
- fxdiv() : [tonc\\_math.h](#)
- fxdiv64() : [tonc\\_math.h](#)
- fxml() : [tonc\\_math.h](#)
- fxml64() : [tonc\\_math.h](#)
- fxsub() : [tonc\\_math.h](#)

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)[File List](#)[Globals](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Defines](#)[\\_](#)[a](#)[b](#)[c](#)[d](#)[f](#)[h](#)[i](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)

## - h -

- `hword2word()` : [tonc\\_core.h](#)

Generated on Mon Aug 25 17:03:58 2008 for libtonc by  1.5.3

## - i -

- `in_range()` : [tonc\\_math.h](#)
- `int2fx()` : [tonc\\_math.h](#)
- `irq_add()` : [tonc\\_irq.c](#) , [tonc\\_irq.h](#)
- `irq_delete()` : [tonc\\_irq.h](#) , [tonc\\_irq.c](#)
- `irq_init()` : [tonc\\_irq.c](#) , [tonc\\_irq.h](#)
- `irq_set()` : [tonc\\_irq.c](#) , [tonc\\_irq.h](#)
- `irq_set_master()` : [tonc\\_irq.h](#) , [tonc\\_irq.c](#)

Main Page Modules Data Structures Files Related Pages

File List Globals

All Functions Variables Typedefs Enumerations Enumerator Defines

\_ a b c d f h i k l m n o p q r s t u v w

## - k -

- key\_curr\_state() : [tonc\\_input.h](#)
- key\_held() : [tonc\\_input.h](#)
- key\_hit() : [tonc\\_input.h](#)
- key\_is\_down() : [tonc\\_input.h](#)
- key\_is\_up() : [tonc\\_input.h](#)
- key\_poll() : [tonc\\_input.h](#) , [tonc\\_input.c](#)
- key\_prev\_state() : [tonc\\_input.h](#)
- key\_released() : [tonc\\_input.h](#)
- key\_repeat() : [tonc\\_input.h](#) , [tonc\\_input.c](#)
- key\_repeat\_limits() : [tonc\\_input.h](#) , [tonc\\_input.c](#)
- key\_repeat\_mask() : [tonc\\_input.c](#) , [tonc\\_input.h](#)
- key\_transit() : [tonc\\_input.h](#)
- key\_tri\_fire() : [tonc\\_input.h](#)
- key\_tri\_horz() : [tonc\\_input.h](#)
- key\_tri\_shoulder() : [tonc\\_input.h](#)
- key\_tri\_vert() : [tonc\\_input.h](#)
- key\_wait\_till\_hit() : [tonc\\_input.h](#) , [tonc\\_input.c](#)
- key\_was\_down() : [tonc\\_input.h](#)
- key\_was\_up() : [tonc\\_input.h](#)

- | -

- lu\_cos() : [tonc\\_math.h](#)
- lu\_div() : [tonc\\_math.h](#)
- lu\_lerp16() : [tonc\\_math.h](#)
- lu\_lerp32() : [tonc\\_math.h](#)
- lu\_sin() : [tonc\\_math.h](#)

## - m -

- `m3_clrs()` : [tonc\\_text.h](#)
- `m3_fill()` : [tonc\\_video.h](#)
- `m3_frame()` : [tonc\\_video.h](#)
- `m3_hline()` : [tonc\\_video.h](#)
- `m3_line()` : [tonc\\_video.h](#)
- `m3_plot()` : [tonc\\_video.h](#)
- `m3_putc()` : [tonc\\_text.h](#)
- `m3_puts()` : [tonc\\_text.h](#)
- `m3_rect()` : [tonc\\_video.h](#)
- `m3_vline()` : [tonc\\_video.h](#)
- `m4_clrs()` : [tonc\\_text.h](#)
- `m4_fill()` : [tonc\\_video.h](#)
- `m4_frame()` : [tonc\\_video.h](#)
- `m4_hline()` : [tonc\\_video.h](#)
- `m4_line()` : [tonc\\_video.h](#)
- `m4_plot()` : [tonc\\_video.h](#)
- `m4_putc()` : [tonc\\_text.h](#)
- `m4_puts()` : [tonc\\_text.h](#)
- `m4_rect()` : [tonc\\_video.h](#)
- `m4_vline()` : [tonc\\_video.h](#)
- `m5_clrs()` : [tonc\\_text.h](#)
- `m5_fill()` : [tonc\\_video.h](#)
- `m5_frame()` : [tonc\\_video.h](#)
- `m5_hline()` : [tonc\\_video.h](#)

- m5\_line() : **tonc\_video.h**
  - m5\_plot() : **tonc\_video.h**
  - m5\_putc() : **tonc\_text.h**
  - m5\_puts() : **tonc\_text.h**
  - m5\_rect() : **tonc\_video.h**
  - m5\_vline() : **tonc\_video.h**
  - max() : **tonc\_math.h**
  - memcpy16() : **tonc\_core.h**
  - memcpy32() : **tonc\_core.h**
  - memset16() : **tonc\_core.h**
  - memset32() : **tonc\_core.h**
  - min() : **tonc\_math.h**
  - Mod() : **tonc\_bios.h**
- 

*Generated on Mon Aug 25 17:03:58 2008 for libtonc by*  **doxygen** 1.5.3

## - n -

- `nocash_message()` : [tonc\\_nocash.h](#)
- `nocash_puts()` : [tonc\\_nocash.h](#)

>Main Page Modules Data Structures Files Related Pages

File List Globals

All Functions Variables Typedefs Enumerations Enumerator Defines

\_ a b c d f h i k l m n o p q r s t u v w

## - O -

- `oam_copy()` : [tonc\\_oam.h](#)
- `oam_init()` : [tonc\\_oam.h](#) , [tonc\\_oam.c](#)
- `obj_aff_identity()` : [tonc\\_oam.h](#)
- `obj_aff_postmul()` : [tonc\\_oam.h](#) , [tonc\\_obj\\_affine.c](#)
- `obj_aff_premul()` : [tonc\\_obj\\_affine.c](#) , [tonc\\_oam.h](#)
- `obj_aff_rotate()` : [tonc\\_oam.h](#) , [tonc\\_obj\\_affine.c](#)
- `obj_aff_rotscale()` : [tonc\\_oam.h](#) , [tonc\\_obj\\_affine.c](#)
- `obj_aff_rotscale2()` : [tonc\\_oam.h](#) , [tonc\\_obj\\_affine.c](#)
- `obj_aff_scale()` : [tonc\\_oam.h](#)
- `obj_aff_set()` : [tonc\\_oam.h](#)
- `obj_copy()` : [tonc\\_oam.h](#) , [tonc\\_oam.c](#)
- `obj_drawg()` : [tonc\\_tte.h](#) , [obj\\_drawg.c](#)
- `obj_erase()` : [tonc\\_tte.h](#) , [obj\\_drawg.c](#)
- `obj_get_height()` : [tonc\\_oam.h](#)
- `obj_get_size()` : [tonc\\_oam.h](#)
- `obj_get_width()` : [tonc\\_oam.h](#)
- `obj_hide()` : [tonc\\_oam.h](#)
- `obj_hide_multi()` : [tonc\\_oam.h](#) , [tonc\\_oam.c](#)
- `obj_putc2()` : [tonc\\_text.h](#)
- `obj_puts2()` : [tonc\\_text.h](#)
- `obj_rotscale_ex()` : [tonc\\_obj\\_affine.c](#) , [tonc\\_oam.h](#)
- `obj_set_attr()` : [tonc\\_oam.h](#)
- `obj_set_pos()` : [tonc\\_oam.h](#)
- `obj_unhide()` : [tonc\\_oam.h](#)

- ObjAffineSet() : **tonc\_bios.h**
  - octant() : **tonc\_core.h** , **tonc\_core.c**
  - octant\_rot() : **tonc\_core.c** , **tonc\_core.h**
  - octup() : **tonc\_core.h**
- 

*Generated on Mon Aug 25 17:03:58 2008 for libtonc by  1.5.3*

>Main Page   Modules   Data Structures   Files   Related Pages

File List   Globals

All   Functions   Variables   Typedefs   Enumerations   Enumerator   Defines

\_   a   b   c   d   f   h   i   k   l   m   n   o   p   q   r   s   t   u   v   w

## - p -

- pal\_gradient() : [tonc\\_video.h](#) , [tonc\\_color.c](#)
- pal\_gradient\_ex() : [tonc\\_color.c](#) , [tonc\\_video.h](#)
- profile\_start() : [tonc\\_core.h](#)
- profile\_stop() : [tonc\\_core.h](#)
- pt\_add() : [tonc\\_math.h](#)
- pt\_add\_eq() : [tonc\\_math.h](#)
- pt\_cross() : [tonc\\_math.h](#)
- pt\_dot() : [tonc\\_math.h](#)
- pt\_in\_rect() : [tonc\\_math.c](#)
- pt\_scale() : [tonc\\_math.h](#)
- pt\_scale\_eq() : [tonc\\_math.h](#)
- pt\_set() : [tonc\\_math.h](#)
- pt\_sub() : [tonc\\_math.h](#)
- pt\_sub\_eq() : [tonc\\_math.h](#)

## - q -

- `qran()` : [tonc\\_core.h](#)
- `qran_range()` : [tonc\\_core.h](#)
- `quad8()` : [tonc\\_core.h](#)

## - r -

- rc\_height() : [tonc\\_math.h](#)
- rc\_inflate() : [tonc\\_math.h](#)
- rc\_inflate2() : [tonc\\_math.h](#)
- rc\_move() : [tonc\\_math.h](#)
- rc\_set() : [tonc\\_math.h](#)
- rc\_set2() : [tonc\\_math.h](#)
- rc\_set\_pos() : [tonc\\_math.h](#)
- rc\_set\_size() : [tonc\\_math.h](#)
- rc\_width() : [tonc\\_math.h](#)
- reflect() : [tonc\\_math.h](#)
- RGB15() : [tonc\\_video.h](#)
- RGB15\_SAFE() : [tonc\\_video.h](#)
- RGB8() : [tonc\\_video.h](#)
- ROR() : [tonc\\_core.h](#)

## - S -

- `sbmp16.blit()` : [tonc\\_surface.h](#) , [tonc\\_sbmp16.c](#)
- `sbmp16_floodfill()` : [tonc\\_sbmp16.c](#) , [tonc\\_surface.h](#)
- `sbmp16_floodfill_internal()` : [tonc\\_sbmp16.c](#)
- `sbmp16_frame()` : [tonc\\_surface.h](#) , [tonc\\_sbmp16.c](#)
- `sbmp16_get_pixel()` : [tonc\\_surface.h](#) , [tonc\\_sbmp16.c](#)
- `sbmp16_hline()` : [tonc\\_surface.h](#) , [tonc\\_sbmp16.c](#)
- `sbmp16_line()` : [tonc\\_surface.h](#) , [tonc\\_sbmp16.c](#)
- `sbmp16_plot()` : [tonc\\_surface.h](#) , [tonc\\_sbmp16.c](#)
- `sbmp16_rect()` : [tonc\\_surface.h](#) , [tonc\\_sbmp16.c](#)
- `sbmp16_vline()` : [tonc\\_surface.h](#) , [tonc\\_sbmp16.c](#)
- `sbmp8.blit()` : [tonc\\_surface.h](#) , [tonc\\_sbmp8.c](#)
- `sbmp8_floodfill()` : [tonc\\_surface.h](#) , [tonc\\_sbmp8.c](#)
- `sbmp8_floodfill_internal()` : [tonc\\_sbmp8.c](#)
- `sbmp8_frame()` : [tonc\\_surface.h](#) , [tonc\\_sbmp8.c](#)
- `sbmp8_get_pixel()` : [tonc\\_surface.h](#) , [tonc\\_sbmp8.c](#)
- `sbmp8_hline()` : [tonc\\_surface.h](#) , [tonc\\_sbmp8.c](#)
- `sbmp8_line()` : [tonc\\_sbmp8.c](#) , [tonc\\_surface.h](#)
- `sbmp8_plot()` : [tonc\\_surface.h](#) , [tonc\\_sbmp8.c](#)
- `sbmp8_rect()` : [tonc\\_surface.h](#) , [tonc\\_sbmp8.c](#)
- `sbmp8_vline()` : [tonc\\_surface.h](#) , [tonc\\_sbmp8.c](#)
- `schr4c.blit()` : [tonc\\_surface.h](#) , [tonc\\_schr4c.c](#)
- `schr4c_floodfill()` : [tonc\\_surface.h](#) , [tonc\\_schr4c.c](#)
- `schr4c_floodfill_internal()` : [tonc\\_schr4c.c](#)
- `schr4c_frame()` : [tonc\\_surface.h](#) , [tonc\\_schr4c.c](#)

- `schr4c_get_pixel()` : `tonc_surface.h` , `tonc_schr4c.c`
- `schr4c_get_ptr()` : `tonc_surface.h` , `tonc_schr4c.c`
- `schr4c_hline()` : `tonc_surface.h` , `tonc_schr4c.c`
- `schr4c_line()` : `tonc_surface.h` , `tonc_schr4c.c`
- `schr4c_plot()` : `tonc_surface.h` , `tonc_schr4c.c`
- `schr4c_prep_map()` : `tonc_surface.h` , `tonc_schr4c.c`
- `schr4c_rect()` : `tonc_schr4c.c` , `tonc_surface.h`
- `schr4c_vline()` : `tonc_surface.h` , `tonc_schr4c.c`
- `schr4r_frame()` : `tonc_schr4r.c` , `tonc_surface.h`
- `schr4r_get_pixel()` : `tonc_surface.h` , `tonc_schr4r.c`
- `schr4r_get_ptr()` : `tonc_surface.h` , `tonc_schr4r.c`
- `schr4r_hline()` : `tonc_surface.h` , `tonc_schr4r.c`
- `schr4r_line()` : `tonc_surface.h` , `tonc_schr4r.c`
- `schr4r_plot()` : `tonc_surface.h` , `tonc_schr4r.c`
- `schr4r_prep_map()` : `tonc_surface.h` , `tonc_schr4r.c`
- `schr4r_rect()` : `tonc_surface.h` , `tonc_schr4r.c`
- `schr4r_vline()` : `tonc_surface.h` , `tonc_schr4r.c`
- `se_drawg()` : `se_drawg.c` , `tonc_tte.h`
- `se_drawg_s()` : `tonc_tte.h` , `se_drawg.c`
- `se_drawg_w8h16()` : `tonc_tte.h` , `se_drawg.c`
- `se_drawg_w8h8()` : `tonc_tte.h` , `se_drawg.c`
- `se_erase()` : `se_drawg.c` , `tonc_tte.h`
- `se_fill()` : `tonc_video.h`
- `se_frame()` : `tonc_video.h`
- `se_plot()` : `tonc_video.h`
- `se_rect()` : `tonc_video.h`
- `se_window()` : `tonc_video.h` , `tonc_bg.c`
- `sgn()` : `tonc_math.h`
- `sgn3()` : `tonc_math.h`
- `Sqrt()` : `tonc_bios.h`
- `srf_align()` : `tonc_surface.h`
- `srf_get_ptr()` : `tonc_surface.h` , `tonc_surface.c`
- `srf_init()` : `tonc_surface.c` , `tonc_surface.h`

- `srf_pal_copy()` : **tonc\_surface.h** , **tonc\_surface.c**
  - `srf_set_pal()` : **tonc\_surface.h**
  - `srf_set_ptr()` : **tonc\_surface.h**
- 

Generated on Mon Aug 25 17:03:58 2008 for libtonc by  doxygen 1.5.3

## - t -

- tonccpy() : [tonc\\_core.h](#) , [tonc\\_core.c](#)
- toncset() : [tonc\\_core.h](#)
- toncset16() : [tonc\\_core.h](#)
- toncset32() : [tonc\\_core.h](#)
- tte\_cmd\_default() : [tonc\\_tte.h](#) , [tte\\_main.c](#)
- tte\_cmd\_next() : [tte\\_main.c](#)
- tte\_cmd\_skip() : [tte\\_main.c](#)
- tte\_cmd\_vt100() : [tonc\\_tte.h](#) , [tte\\_iohook.c](#)
- tte\_con\_write() : [tonc\\_tte.h](#) , [tte\\_iohook.c](#)
- tte\_erase\_line() : [tonc\\_tte.h](#) , [tte\\_main.c](#)
- tte\_erase\_rect() : [tonc\\_tte.h](#) , [tte\\_main.c](#)
- tte\_erase\_screen() : [tonc\\_tte.h](#) , [tte\\_main.c](#)
- tte\_get\_context() : [tonc\\_tte.h](#)
- tte\_get\_drawg() : [tonc\\_tte.h](#)
- tte\_get\_erase() : [tonc\\_tte.h](#)
- tte\_get\_font() : [tonc\\_tte.h](#)
- tte\_get\_font\_table() : [tonc\\_tte.h](#)
- tte\_get\_glyph\_data() : [tonc\\_tte.h](#)
- tte\_get\_glyph\_height() : [tonc\\_tte.h](#)
- tte\_get\_glyph\_id() : [tonc\\_tte.h](#)
- tte\_get\_glyph\_width() : [tonc\\_tte.h](#)
- tte\_get\_ink() : [tonc\\_tte.h](#)
- tte\_get\_paper() : [tonc\\_tte.h](#)
- tte\_get\_pos() : [tonc\\_tte.h](#)

- tte\_get\_shadow() : **tonc\_tte.h**
- tte\_get\_special() : **tonc\_tte.h**
- tte\_get\_string\_table() : **tonc\_tte.h**
- tte\_get\_surface() : **tonc\_tte.h**
- tte\_get\_text\_size() : **tonc\_tte.h , tte\_main.c**
- tte\_init\_ase() : **tonc\_tte.h , tte\_init\_ase.c**
- tte\_init\_base() : **tonc\_tte.h , tte\_main.c**
- tte\_init\_bmp() : **tonc\_tte.h , tte\_init\_bmp.c**
- tte\_init\_chr4c() : **tonc\_tte.h , tte\_init\_chr4c.c**
- tte\_init\_chr4r() : **tte\_init\_chr4r.c , tonc\_tte.h**
- tte\_init\_con() : **tonc\_tte.h , tte\_iohook.c**
- tte\_init\_obj() : **tonc\_tte.h , tte\_init\_obj.c**
- tte\_init\_se() : **tonc\_tte.h , tte\_init\_se.c**
- tte\_putc() : **tonc\_tte.h , tte\_main.c**
- tte\_set\_color() : **tte\_main.c , tonc\_tte.h**
- tte\_set\_color\_attr() : **tte\_main.c , tonc\_tte.h**
- tte\_set\_color\_attrs() : **tte\_main.c , tonc\_tte.h**
- tte\_set\_colors() : **tonc\_tte.h , tte\_main.c**
- tte\_set\_context() : **tonc\_tte.h , tte\_main.c**
- tte\_set\_drawg() : **tonc\_tte.h**
- tte\_set\_erase() : **tonc\_tte.h**
- tte\_set\_font() : **tonc\_tte.h**
- tte\_set\_font\_table() : **tonc\_tte.h**
- tte\_set\_ink() : **tonc\_tte.h**
- tte\_set\_paper() : **tonc\_tte.h**
- tte\_set\_pos() : **tonc\_tte.h**
- tte\_set\_shadow() : **tonc\_tte.h**
- tte\_set\_special() : **tonc\_tte.h**
- tte\_set\_string\_table() : **tonc\_tte.h**
- tte\_set\_surface() : **tonc\_tte.h**
- tte\_write() : **tonc\_tte.h , tte\_main.c**
- tte\_write\_ex() : **tonc\_tte.h , tte\_main.c**
- txt\_init\_se() : **tonc\_text.h**

- `txt_init_std()` : [tonc\\_text.h](#)
- 

*Generated on Mon Aug 25 17:03:58 2008 for libtonc by  doxygen 1.5.3*

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)[File List](#)[Globals](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Defines](#)[\\_](#)[a](#)[b](#)[c](#)[d](#)[f](#)[h](#)[i](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)

## - U -

- `utf8_decode_char()` : [tte\\_iohook.c](#) , [tte\\_main.c](#)

Main Page Modules Data Structures Files Related Pages

File List Globals

All Functions Variables Typedefs Enumerations Enumerator Defines

\_ a b c d f h i k l m n o p q r s t u v w

- V -

- VBlankIntrDelay() : [tonc\\_bios.h](#)
- VBlankIntrWait() : [tonc\\_bios.h](#)
- vec\_add() : [tonc\\_math.h](#)
- vec\_add\_eq() : [tonc\\_math.h](#)
- vec\_dot() : [tonc\\_math.h](#)
- vec\_mul() : [tonc\\_math.h](#)
- vec\_mul\_eq() : [tonc\\_math.h](#)
- vec\_scale() : [tonc\\_math.h](#)
- vec\_scale\_eq() : [tonc\\_math.h](#)
- vec\_set() : [tonc\\_math.h](#)
- vec\_sub() : [tonc\\_math.h](#)
- vec\_sub\_eq() : [tonc\\_math.h](#)
- vid\_flip() : [tonc\\_video.h](#) , [tonc\\_video.c](#)

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)[File List](#)[Globals](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Defines](#)[\\_](#)[a](#)[b](#)[c](#)[d](#)[f](#)[h](#)[i](#)[k](#)[l](#)[m](#)[n](#)[o](#)[p](#)[q](#)[r](#)[s](#)[t](#)[u](#)[v](#)[w](#)

## - W -

- wrap() : [tonc\\_math.h](#)

**- a -**

- ABS : [tonc\\_math.h](#)
- ALIGN : [tonc\\_types.h](#)
- ALIGN4 : [tonc\\_types.h](#)
- ASM\_BREAK : [tonc\\_core.h](#)
- ASM\_CMT : [tonc\\_core.h](#)
- ASM\_NOP : [tonc\\_core.h](#)
- ATTR0\_4BPP : [tonc\\_memdef.h](#)
- ATTR0\_8BPP : [tonc\\_memdef.h](#)
- ATTR0\_AFF : [tonc\\_memdef.h](#)
- ATTR0\_AFF\_DBL : [tonc\\_memdef.h](#)
- ATTR0\_BLEND : [tonc\\_memdef.h](#)
- ATTR0\_HIDE : [tonc\\_memdef.h](#)
- ATTR0\_MOSAIC : [tonc\\_memdef.h](#)
- ATTR0\_REG : [tonc\\_memdef.h](#)
- ATTR0\_SQUARE : [tonc\\_memdef.h](#)
- ATTR0\_TALL : [tonc\\_memdef.h](#)
- ATTR0\_WIDE : [tonc\\_memdef.h](#)
- ATTR0\_WINDOW : [tonc\\_memdef.h](#)
- ATTR1\_HFLIP : [tonc\\_memdef.h](#)
- ATTR1\_SIZE\_16x16 : [tonc\\_memdef.h](#)
- ATTR1\_SIZE\_16x32 : [tonc\\_memdef.h](#)
- ATTR1\_SIZE\_16x8 : [tonc\\_memdef.h](#)
- ATTR1\_SIZE\_32x16 : [tonc\\_memdef.h](#)
- ATTR1\_SIZE\_32x32 : [tonc\\_memdef.h](#)

- ATTR1\_SIZE\_32x64 : [tonc\\_memdef.h](#)
  - ATTR1\_SIZE\_32x8 : [tonc\\_memdef.h](#)
  - ATTR1\_SIZE\_64x32 : [tonc\\_memdef.h](#)
  - ATTR1\_SIZE\_64x64 : [tonc\\_memdef.h](#)
  - ATTR1\_SIZE\_8x16 : [tonc\\_memdef.h](#)
  - ATTR1\_SIZE\_8x32 : [tonc\\_memdef.h](#)
  - ATTR1\_SIZE\_8x8 : [tonc\\_memdef.h](#)
  - ATTR1\_VFLIP : [tonc\\_memdef.h](#)
  - ATTR2\_ID\_MASK : [tonc\\_memdef.h](#)
- 

Generated on Mon Aug 25 17:03:58 2008 for libtonc by  doxygen 1.5.3

## - b -

- BF\_MASK : [tonc\\_core.h](#)
- BFN\_CMP : [tonc\\_core.h](#)
- BFN\_GET : [tonc\\_core.h](#)
- BFN\_GET2 : [tonc\\_core.h](#)
- BFN\_PREP : [tonc\\_core.h](#)
- BFN\_PREP2 : [tonc\\_core.h](#)
- BFN\_SET : [tonc\\_core.h](#)
- BFN\_SET2 : [tonc\\_core.h](#)
- BG\_4BPP : [tonc\\_memdef.h](#)
- BG\_8BPP : [tonc\\_memdef.h](#)
- BG\_AFF\_128x128 : [tonc\\_memdef.h](#)
- BG\_AFF\_16x16 : [tonc\\_memdef.h](#)
- BG\_AFF\_32x32 : [tonc\\_memdef.h](#)
- BG\_AFF\_64x64 : [tonc\\_memdef.h](#)
- BG\_AFF\_OFS : [tonc\\_bios.h](#)
- BG\_MOSAIC : [tonc\\_memdef.h](#)
- BG\_REG\_32x32 : [tonc\\_memdef.h](#)
- BG\_REG\_32x64 : [tonc\\_memdef.h](#)
- BG\_REG\_64x32 : [tonc\\_memdef.h](#)
- BG\_REG\_64x64 : [tonc\\_memdef.h](#)
- BG\_WRAP : [tonc\\_memdef.h](#)
- BIT : [tonc\\_core.h](#)
- BIT\_CLEAR : [tonc\\_core.h](#)
- BIT\_EQ : [tonc\\_core.h](#)

- BIT\_FLIP : [tonc\\_core.h](#)
  - BIT\_MASK : [tonc\\_core.h](#)
  - BIT\_SET : [tonc\\_core.h](#)
  - BIT\_SHIFT : [tonc\\_core.h](#)
  - BLD\_ALL : [tonc\\_memdef.h](#)
  - BLD\_BACKDROP : [tonc\\_memdef.h](#)
  - BLD\_BG0 : [tonc\\_memdef.h](#)
  - BLD\_BG1 : [tonc\\_memdef.h](#)
  - BLD\_BG2 : [tonc\\_memdef.h](#)
  - BLD\_BG3 : [tonc\\_memdef.h](#)
  - BLD\_BLACK : [tonc\\_memdef.h](#)
  - BLD\_OBJ : [tonc\\_memdef.h](#)
  - BLD\_OFF : [tonc\\_memdef.h](#)
  - BLD\_STD : [tonc\\_memdef.h](#)
  - BLD\_WHITE : [tonc\\_memdef.h](#)
- 

Generated on Mon Aug 25 17:03:58 2008 for libtonc by  1.5.3

Main Page Modules Data Structures Files Related Pages

File List Globals

All Functions Variables Typedefs Enumerations Enumerator Defines

\_ a b c d e f g i k m o p r s t v w x

## - C -

- CBB\_CLEAR : [tonc\\_video.h](#)
- CBB\_SIZE : [tonc\\_memmap.h](#)
- CFS\_CPY : [tonc\\_bios.h](#)
- CFS\_FILL : [tonc\\_bios.h](#)
- CLAMP : [tonc\\_math.h](#)
- countof : [tonc\\_core.h](#)
- CS\_CPY : [tonc\\_bios.h](#)
- CS\_CPY16 : [tonc\\_bios.h](#)
- CS\_CPY32 : [tonc\\_bios.h](#)
- CS\_FILL : [tonc\\_bios.h](#)
- CS\_FILL32 : [tonc\\_bios.h](#)

>Main Page Modules Data Structures Files Related Pages

File List Globals

All Functions Variables Typedefs Enumerations Enumerator Defines

\_ a b c d e f g i k m o p r s t v w x

## - d -

- DCNT\_BG0 : [tonc\\_memdef.h](#)
- DCNT\_BG1 : [tonc\\_memdef.h](#)
- DCNT\_BG2 : [tonc\\_memdef.h](#)
- DCNT\_BG3 : [tonc\\_memdef.h](#)
- DCNT\_BLANK : [tonc\\_memdef.h](#)
- DCNT\_GB : [tonc\\_memdef.h](#)
- DCNT\_MODE0 : [tonc\\_memdef.h](#)
- DCNT\_MODE1 : [tonc\\_memdef.h](#)
- DCNT\_MODE2 : [tonc\\_memdef.h](#)
- DCNT\_MODE3 : [tonc\\_memdef.h](#)
- DCNT\_MODE4 : [tonc\\_memdef.h](#)
- DCNT\_MODE5 : [tonc\\_memdef.h](#)
- DCNT\_OAM\_HBL : [tonc\\_memdef.h](#)
- DCNT\_OBJ : [tonc\\_memdef.h](#)
- DCNT\_OBJ\_1D : [tonc\\_memdef.h](#)
- DCNT\_OBJ\_2D : [tonc\\_memdef.h](#)
- DCNT\_PAGE : [tonc\\_memdef.h](#)
- DCNT\_WIN0 : [tonc\\_memdef.h](#)
- DCNT\_WIN1 : [tonc\\_memdef.h](#)
- DCNT\_WINOBJ : [tonc\\_memdef.h](#)
- DEPRECATED : [tonc\\_types.h](#)
- DMA\_16 : [tonc\\_memdef.h](#)
- DMA\_32 : [tonc\\_memdef.h](#)
- DMA\_AT\_FIFO : [tonc\\_memdef.h](#)

- DMA\_AT\_HBLANK : [tonc\\_memdef.h](#)
- DMA\_AT\_NOW : [tonc\\_memdef.h](#)
- DMA\_AT\_REFRESH : [tonc\\_memdef.h](#)
- DMA\_AT\_SPECIAL : [tonc\\_memdef.h](#)
- DMA\_AT\_VBLANK : [tonc\\_memdef.h](#)
- DMA\_DST\_DEC : [tonc\\_memdef.h](#)
- DMA\_DST\_FIXED : [tonc\\_memdef.h](#)
- DMA\_DST\_INC : [tonc\\_memdef.h](#)
- DMA\_DST\_RELOAD : [tonc\\_memdef.h](#)
- DMA\_ENABLE : [tonc\\_memdef.h](#)
- DMA\_GAMEPAK : [tonc\\_memdef.h](#)
- DMA\_IRQ : [tonc\\_memdef.h](#)
- DMA\_REPEAT : [tonc\\_memdef.h](#)
- DMA\_SRC\_DEC : [tonc\\_memdef.h](#)
- DMA\_SRC\_FIXED : [tonc\\_memdef.h](#)
- DMA\_SRC\_INC : [tonc\\_memdef.h](#)
- DMA\_TRANSFER : [tonc\\_core.h](#)
- DSTAT\_HBL\_IRQ : [tonc\\_memdef.h](#)
- DSTAT\_IN\_HBL : [tonc\\_memdef.h](#)
- DSTAT\_IN\_VBL : [tonc\\_memdef.h](#)
- DSTAT\_IN\_VCT : [tonc\\_memdef.h](#)
- DSTAT\_VBL\_IRQ : [tonc\\_memdef.h](#)
- DSTAT\_VCT\_IRQ : [tonc\\_memdef.h](#)

Main Page Modules Data Structures Files Related Pages

File List Globals

All Functions Variables Typedefs Enumerations Enumerator Defines

\_ a b c d e f g i k m o p r s t v w x

- e -

- EWRAM\_BSS : [tonc\\_types.h](#)
- EWRAM\_CODE : [tonc\\_types.h](#)
- EWRAM\_DATA : [tonc\\_types.h](#)

---

Generated on Mon Aug 25 17:03:58 2008 for libtonc by [doxygen](#) 1.5.3

Main Page Modules Data Structures Files Related Pages

File List Globals

All Functions Variables Typedefs Enumerations Enumerator Defines

\_ a b c d e f g i k m o p r s t v w x

- f -

- FIX\_SHIFT : [tonc\\_math.h](#)
- fwf\_default : [tonc\\_tte.h](#)
- FX\_RECIMUL : [tonc\\_math.h](#)
- FX\_RECIPROCAL : [tonc\\_math.h](#)

---

Generated on Mon Aug 25 17:03:58 2008 for libtonc by [!\[\]\(edc33152edb73be4cc8a0bfd67e3c338\_img.jpg\) doxygen](#) 1.5.3

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)[File List](#)[Globals](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Defines](#)[\\_](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[i](#)[k](#)[m](#)[o](#)[p](#)[r](#)[s](#)[t](#)[v](#)[w](#)[x](#)

## - g -

- GRIT\_CPY : [tonc\\_core.h](#)

Generated on Mon Aug 25 17:03:58 2008 for libtonc by  doxygen 1.5.3

>Main Page Modules Data Structures Files Related Pages

File List Globals

All Functions Variables Typedefs Enumerations Enumerator Defines

\_ a b c d e f g i k m o p r s t v w x

- i -

- IN\_RANGE : [tonc\\_math.h](#)
- INLINE : [tonc\\_types.h](#)
- IRQ\_DMA0 : [tonc\\_memdef.h](#)
- IRQ\_DMA1 : [tonc\\_memdef.h](#)
- IRQ\_DMA2 : [tonc\\_memdef.h](#)
- IRQ\_DMA3 : [tonc\\_memdef.h](#)
- IRQ\_GAMEPAK : [tonc\\_memdef.h](#)
- IRQ\_HBLANK : [tonc\\_memdef.h](#)
- IRQ\_INIT : [tonc\\_irq.h](#)
- IRQ\_KEYPAD : [tonc\\_memdef.h](#)
- IRQ\_SERIAL : [tonc\\_memdef.h](#)
- IRQ\_SET : [tonc\\_irq.h](#)
- IRQ\_TIMER0 : [tonc\\_memdef.h](#)
- IRQ\_TIMER1 : [tonc\\_memdef.h](#)
- IRQ\_TIMER2 : [tonc\\_memdef.h](#)
- IRQ\_TIMER3 : [tonc\\_memdef.h](#)
- IRQ\_VBLANK : [tonc\\_memdef.h](#)
- IRQ\_VCOUNT : [tonc\\_memdef.h](#)
- ISR\_DEF : [tonc\\_irq.h](#)
- ISR\_LAST : [tonc\\_irq.h](#)
- ISR\_PRIO : [tonc\\_irq.h](#)
- ISR\_PRIO\_MASK : [tonc\\_irq.h](#)
- ISR\_PRIO\_SHIFT : [tonc\\_irq.h](#)
- ISR\_REPLACE : [tonc\\_irq.h](#)

- IWRAM\_CODE : [tonc\\_types.h](#)
  - IWRAM\_DATA : [tonc\\_types.h](#)
- 

Generated on Mon Aug 25 17:03:58 2008 for libtonc by  doxygen 1.5.3

## - k -

- KCNT\_AND : [tonc\\_memdef.h](#)
- KCNT\_IRQ : [tonc\\_memdef.h](#)
- KCNT\_OR : [tonc\\_memdef.h](#)
- KEY\_A : [tonc\\_memdef.h](#)
- KEY\_ACCEPT : [tonc\\_memdef.h](#)
- KEY\_ANY : [tonc\\_memdef.h](#)
- KEY\_B : [tonc\\_memdef.h](#)
- KEY\_CANCEL : [tonc\\_memdef.h](#)
- KEY\_DIR : [tonc\\_memdef.h](#)
- KEY\_DOWN : [tonc\\_memdef.h](#)
- KEY\_FIRE : [tonc\\_memdef.h](#)
- KEY\_FULL : [tonc\\_input.h](#)
- KEY\_L : [tonc\\_memdef.h](#)
- KEY\_LEFT : [tonc\\_memdef.h](#)
- KEY\_R : [tonc\\_memdef.h](#)
- KEY\_RESET : [tonc\\_memdef.h](#)
- KEY\_RIGHT : [tonc\\_memdef.h](#)
- KEY\_SELECT : [tonc\\_memdef.h](#)
- KEY\_SHOULDER : [tonc\\_memdef.h](#)
- KEY\_SPECIAL : [tonc\\_memdef.h](#)
- KEY\_START : [tonc\\_memdef.h](#)
- KEY\_UP : [tonc\\_memdef.h](#)

*Generated on Mon Aug 25 17:03:58 2008 for libtonc by*  **doxygen** 1.5.3

>Main Page Modules Data Structures Files Related Pages

File List Globals

All Functions Variables Typedefs Enumerations Enumerator Defines

\_ a b c d e f g i k m o p r s t v w x

## - M -

- M3\_CLEAR : [tonc\\_video.h](#)
- m3\_mem : [tonc\\_memmap.h](#)
- M3\_SIZE : [tonc\\_memmap.h](#)
- M4\_CLEAR : [tonc\\_video.h](#)
- m4\_mem : [tonc\\_memmap.h](#)
- m4\_mem\_back : [tonc\\_memmap.h](#)
- M4\_SIZE : [tonc\\_memmap.h](#)
- M5\_CLEAR : [tonc\\_video.h](#)
- m5\_mem : [tonc\\_memmap.h](#)
- m5\_mem\_back : [tonc\\_memmap.h](#)
- M5\_SIZE : [tonc\\_memmap.h](#)
- MAX : [tonc\\_math.h](#)
- MEM\_EWRAM : [tonc\\_memmap.h](#)
- MEM\_IO : [tonc\\_memmap.h](#)
- MEM\_IWRAM : [tonc\\_memmap.h](#)
- MEM\_OAM : [tonc\\_memmap.h](#)
- MEM\_PAL : [tonc\\_memmap.h](#)
- MEM\_PAL\_BG : [tonc\\_memmap.h](#)
- MEM\_PAL\_OBJ : [tonc\\_memmap.h](#)
- MEM\_ROM : [tonc\\_memmap.h](#)
- MEM\_SRAM : [tonc\\_memmap.h](#)
- MEM\_VRAM : [tonc\\_memmap.h](#)
- MEM\_VRAM\_BACK : [tonc\\_memmap.h](#)
- MEM\_VRAM\_FRONT : [tonc\\_memmap.h](#)

- MEM\_VRAM\_OBJ : [tonc\\_memmap.h](#)
  - MIN : [tonc\\_math.h](#)
- 

*Generated on Mon Aug 25 17:03:58 2008 for libtonc by*  **doxygen** 1.5.3

Main Page Modules Data Structures Files Related Pages

File List Globals

All Functions Variables Typedefs Enumerations Enumerator Defines

\_ a b c d e f g i k m o p r s t v w x

## - O -

- OAM\_CLEAR : [tonc\\_oam.h](#)
- oam\_mem : [tonc\\_memmap.h](#)
- obj\_aff\_mem : [tonc\\_memmap.h](#)
- OBJ\_AFF\_OFS : [tonc\\_bios.h](#)
- obj\_mem : [tonc\\_memmap.h](#)

## - p -

- PACKED : [tonc\\_types.h](#)
- pal\_bg\_bank : [tonc\\_memmap.h](#)
- pal\_bg\_mem : [tonc\\_memmap.h](#)
- PAL\_BG\_SIZE : [tonc\\_memmap.h](#)
- pal\_obj\_bank : [tonc\\_memmap.h](#)
- pal\_obj\_mem : [tonc\\_memmap.h](#)
- PAL\_OBJ\_SIZE : [tonc\\_memmap.h](#)

## - r -

- R\_MODE : [tonc\\_memdef.h](#)
- R\_MODE\_GPIO : [tonc\\_memdef.h](#)
- R\_MODE\_JOYBUS : [tonc\\_memdef.h](#)
- R\_MODE\_MASK : [tonc\\_memdef.h](#)
- R\_MODE\_MULTI : [tonc\\_memdef.h](#)
- R\_MODE\_NORMAL : [tonc\\_memdef.h](#)
- R\_MODE\_SHIFT : [tonc\\_memdef.h](#)
- R\_MODE\_UART : [tonc\\_memdef.h](#)
- RAM\_RESTART : [tonc\\_bios.h](#)
- REFLECT : [tonc\\_math.h](#)
- REG\_BG0CNT : [tonc\\_memmap.h](#)
- REG\_BG0HOFS : [tonc\\_memmap.h](#)
- REG\_BG0VOFS : [tonc\\_memmap.h](#)
- REG\_BG1CNT : [tonc\\_memmap.h](#)
- REG\_BG1HOFS : [tonc\\_memmap.h](#)
- REG\_BG1VOFS : [tonc\\_memmap.h](#)
- REG\_BG2CNT : [tonc\\_memmap.h](#)
- REG\_BG2HOFS : [tonc\\_memmap.h](#)
- REG\_BG2PA : [tonc\\_memmap.h](#)
- REG\_BG2PB : [tonc\\_memmap.h](#)
- REG\_BG2PC : [tonc\\_memmap.h](#)
- REG\_BG2PD : [tonc\\_memmap.h](#)
- REG\_BG2VOFS : [tonc\\_memmap.h](#)
- REG\_BG2X : [tonc\\_memmap.h](#)

- REG\_BG2Y : [tonc\\_memmap.h](#)
- REG\_BG3CNT : [tonc\\_memmap.h](#)
- REG\_BG3HOFS : [tonc\\_memmap.h](#)
- REG\_BG3PA : [tonc\\_memmap.h](#)
- REG\_BG3PB : [tonc\\_memmap.h](#)
- REG\_BG3PC : [tonc\\_memmap.h](#)
- REG\_BG3PD : [tonc\\_memmap.h](#)
- REG\_BG3VOFS : [tonc\\_memmap.h](#)
- REG\_BG3X : [tonc\\_memmap.h](#)
- REG\_BG3Y : [tonc\\_memmap.h](#)
- REG\_BG\_AFFINE : [tonc\\_memmap.h](#)
- REG\_BG\_OFS : [tonc\\_memmap.h](#)
- REG\_BGCNT : [tonc\\_memmap.h](#)
- REG\_BLDALPHA : [tonc\\_memmap.h](#)
- REG\_BLDCNT : [tonc\\_memmap.h](#)
- REG\_BLDMOD : [tonc\\_memmap.h](#)
- REG\_BLDY : [tonc\\_memmap.h](#)
- REG\_DISPCNT : [tonc\\_memmap.h](#)
- REG\_DISPSTAT : [tonc\\_memmap.h](#)
- REG\_DMA : [tonc\\_memmap.h](#)
- REG\_DMA0CNT : [tonc\\_memmap.h](#)
- REG\_DMA0DAD : [tonc\\_memmap.h](#)
- REG\_DMA0SAD : [tonc\\_memmap.h](#)
- REG\_DMA1CNT : [tonc\\_memmap.h](#)
- REG\_DMA1DAD : [tonc\\_memmap.h](#)
- REG\_DMA1SAD : [tonc\\_memmap.h](#)
- REG\_DMA2CNT : [tonc\\_memmap.h](#)
- REG\_DMA2DAD : [tonc\\_memmap.h](#)
- REG\_DMA2SAD : [tonc\\_memmap.h](#)
- REG\_DMA3CNT : [tonc\\_memmap.h](#)
- REG\_DMA3DAD : [tonc\\_memmap.h](#)
- REG\_DMA3SAD : [tonc\\_memmap.h](#)
- REG\_FIFO\_A : [tonc\\_memmap.h](#)

- REG\_FIFO\_B : **tonc\_memmap.h**
- REG\_IE : **tonc\_memmap.h**
- REG\_IF : **tonc\_memmap.h**
- REG\_IFBIOS : **tonc\_memmap.h**
- REG\_IME : **tonc\_memmap.h**
- REG\_ISR\_MAIN : **tonc\_memmap.h**
- REG\_JOY\_RECV : **tonc\_memmap.h**
- REG\_JOY\_TRANS : **tonc\_memmap.h**
- REG\_JOYCNT : **tonc\_memmap.h**
- REG\_JOYSTAT : **tonc\_memmap.h**
- REG\_KEYCNT : **tonc\_memmap.h**
- REG\_KEYINPUT : **tonc\_memmap.h**
- REG\_MOSAIC : **tonc\_memmap.h**
- REG\_PAUSE : **tonc\_memmap.h**
- REG\_RCNT : **tonc\_memmap.h**
- REG\_RESET\_DST : **tonc\_memmap.h**
- REG\_SIOCNT : **tonc\_memmap.h**
- REG\_SIODATA : **tonc\_memmap.h**
- REG\_SIODATA32 : **tonc\_memmap.h**
- REG\_SIODATA8 : **tonc\_memmap.h**
- REG\_SIOMLT\_RECV : **tonc\_memmap.h**
- REG\_SIOMLT\_SEND : **tonc\_memmap.h**
- REG\_SIOMULTI : **tonc\_memmap.h**
- REG\_SIOMULTI0 : **tonc\_memmap.h**
- REG\_SIOMULTI1 : **tonc\_memmap.h**
- REG\_SIOMULTI2 : **tonc\_memmap.h**
- REG\_SIOMULTI3 : **tonc\_memmap.h**
- REG SND1CNT : **tonc\_memmap.h**
- REG SND1FREQ : **tonc\_memmap.h**
- REG SND1SWEEP : **tonc\_memmap.h**
- REG SND2CNT : **tonc\_memmap.h**
- REG SND2FREQ : **tonc\_memmap.h**
- REG SND3CNT : **tonc\_memmap.h**

- REG\_SND3FREQ : [tonc\\_memmap.h](#)
- REG\_SND3SEL : [tonc\\_memmap.h](#)
- REG\_SND4CNT : [tonc\\_memmap.h](#)
- REG\_SND4FREQ : [tonc\\_memmap.h](#)
- REG\_SNDBIAS : [tonc\\_memmap.h](#)
- REG\_SNDCNT : [tonc\\_memmap.h](#)
- REG\_SNDDMGCNT : [tonc\\_memmap.h](#)
- REG\_SNDDSCNT : [tonc\\_memmap.h](#)
- REG SNDSTAT : [tonc\\_memmap.h](#)
- REG\_TM : [tonc\\_memmap.h](#)
- REG\_TM0CNT : [tonc\\_memmap.h](#)
- REG\_TM0D : [tonc\\_memmap.h](#)
- REG\_TM1CNT : [tonc\\_memmap.h](#)
- REG\_TM1D : [tonc\\_memmap.h](#)
- REG\_TM2CNT : [tonc\\_memmap.h](#)
- REG\_TM2D : [tonc\\_memmap.h](#)
- REG\_TM3CNT : [tonc\\_memmap.h](#)
- REG\_TM3D : [tonc\\_memmap.h](#)
- REG\_VCOUNT : [tonc\\_memmap.h](#)
- REG\_WAITCNT : [tonc\\_memmap.h](#)
- REG\_WAVE\_RAM : [tonc\\_memmap.h](#)
- REG\_WAVE\_RAM0 : [tonc\\_memmap.h](#)
- REG\_WAVE\_RAM1 : [tonc\\_memmap.h](#)
- REG\_WAVE\_RAM2 : [tonc\\_memmap.h](#)
- REG\_WAVE\_RAM3 : [tonc\\_memmap.h](#)
- REG\_WIN0B : [tonc\\_memmap.h](#)
- REG\_WIN0CNT : [tonc\\_memmap.h](#)
- REG\_WIN0H : [tonc\\_memmap.h](#)
- REG\_WIN0L : [tonc\\_memmap.h](#)
- REG\_WIN0R : [tonc\\_memmap.h](#)
- REG\_WIN0T : [tonc\\_memmap.h](#)
- REG\_WIN0V : [tonc\\_memmap.h](#)
- REG\_WIN1B : [tonc\\_memmap.h](#)

- REG\_WIN1CNT : [tonc\\_memmap.h](#)
- REG\_WIN1H : [tonc\\_memmap.h](#)
- REG\_WIN1L : [tonc\\_memmap.h](#)
- REG\_WIN1R : [tonc\\_memmap.h](#)
- REG\_WIN1T : [tonc\\_memmap.h](#)
- REG\_WIN1V : [tonc\\_memmap.h](#)
- REG\_WININ : [tonc\\_memmap.h](#)
- REG\_WINOBJCNT : [tonc\\_memmap.h](#)
- REG\_WINOUT : [tonc\\_memmap.h](#)
- REG\_WINOUTCNT : [tonc\\_memmap.h](#)
- RESET\_EWRAM : [tonc\\_bios.h](#)
- RESET\_GFX : [tonc\\_bios.h](#)
- RESET\_IWRAM : [tonc\\_bios.h](#)
- RESET\_MEM\_MASK : [tonc\\_bios.h](#)
- RESET\_OAM : [tonc\\_bios.h](#)
- RESET\_PALETTE : [tonc\\_bios.h](#)
- RESET\_REG : [tonc\\_bios.h](#)
- RESET\_REG\_MASK : [tonc\\_bios.h](#)
- RESET\_REG\_SIO : [tonc\\_bios.h](#)
- RESET\_REG\_SOUND : [tonc\\_bios.h](#)
- RESET\_VRAM : [tonc\\_bios.h](#)
- rom\_mem : [tonc\\_memmap.h](#)
- ROM\_RESTART : [tonc\\_bios.h](#)

>Main Page Modules Data Structures Files Related Pages

File List Globals

All Functions Variables Typedefs Enumerations Enumerator Defines

\_ a b c d e f g i k m o p r s t v w x

## - S -

- SBB\_SIZE : [tonc\\_memmap.h](#)
- SDMG\_LNOISE : [tonc\\_memdef.h](#)
- SDMG\_LSQR1 : [tonc\\_memdef.h](#)
- SDMG\_LSQR2 : [tonc\\_memdef.h](#)
- SDMG\_LWAVE : [tonc\\_memdef.h](#)
- SDMG\_RNOISE : [tonc\\_memdef.h](#)
- SDMG\_RSQR1 : [tonc\\_memdef.h](#)
- SDMG\_RSQR2 : [tonc\\_memdef.h](#)
- SDMG\_RWAVE : [tonc\\_memdef.h](#)
- SDS\_A100 : [tonc\\_memdef.h](#)
- SDS\_A50 : [tonc\\_memdef.h](#)
- SDS\_AL : [tonc\\_memdef.h](#)
- SDS\_AR : [tonc\\_memdef.h](#)
- SDS\_ARESET : [tonc\\_memdef.h](#)
- SDS\_ATMR0 : [tonc\\_memdef.h](#)
- SDS\_ATMR1 : [tonc\\_memdef.h](#)
- SDS\_B100 : [tonc\\_memdef.h](#)
- SDS\_B50 : [tonc\\_memdef.h](#)
- SDS\_BL : [tonc\\_memdef.h](#)
- SDS\_BR : [tonc\\_memdef.h](#)
- SDS\_BRESET : [tonc\\_memdef.h](#)
- SDS\_BTMR0 : [tonc\\_memdef.h](#)
- SDS\_BTMR1 : [tonc\\_memdef.h](#)
- SDS\_DMG100 : [tonc\\_memdef.h](#)

- SDS\_DMG25 : **tonc\_memdef.h**
- SDS\_DMG50 : **tonc\_memdef.h**
- SE\_HFLIP : **tonc\_memdef.h**
- se\_mat : **tonc\_memmap.h**
- se\_mem : **tonc\_memmap.h**
- SE\_VFLIP : **tonc\_memdef.h**
- SFREQ\_HOLD : **tonc\_memdef.h**
- SFREQ\_RESET : **tonc\_memdef.h**
- SFREQ\_TIMED : **tonc\_memdef.h**
- SGN : **tonc\_math.h**
- SGN2 : **tonc\_math.h**
- SGN3 : **tonc\_math.h**
- SIN\_LUT\_SIZE : **tonc\_math.h**
- SIO\_2MHZ\_CLK : **tonc\_libgba.h**
- SIO\_32BIT : **tonc\_libgba.h**
- SIO\_8BIT : **tonc\_libgba.h**
- SIO\_CLK\_INT : **tonc\_libgba.h**
- SIO\_IRQ : **tonc\_libgba.h , tonc\_memdef.h**
- SIO\_MODE : **tonc\_memdef.h**
- SIO\_MODE\_32BIT : **tonc\_memdef.h**
- SIO\_MODE\_8BIT : **tonc\_memdef.h**
- SIO\_MODE\_MASK : **tonc\_memdef.h**
- SIO\_MODE\_MULTI : **tonc\_memdef.h**
- SIO\_MODE\_SHIFT : **tonc\_memdef.h**
- SIO\_MODE\_UART : **tonc\_memdef.h**
- SIO\_MULTI : **tonc\_libgba.h**
- SIO\_RDY : **tonc\_libgba.h**
- SIO\_SI\_HIGH : **tonc\_memdef.h**
- SIO\_SO\_HIGH : **tonc\_libgba.h**
- SIO\_UART : **tonc\_libgba.h**
- SIOM\_115200 : **tonc\_memdef.h**
- SIOM\_38400 : **tonc\_memdef.h**
- SIOM\_57600 : **tonc\_memdef.h**

- SIOM\_9600 : **tonc\_memdef.h**
- SIOM\_BAUD : **tonc\_memdef.h**
- SIOM\_BAUD\_MASK : **tonc\_memdef.h**
- SIOM\_BAUD\_SHIFT : **tonc\_memdef.h**
- SIOM\_CONNECTED : **tonc\_memdef.h**
- SIOM\_ENABLE : **tonc\_memdef.h**
- SIOM\_ERROR : **tonc\_memdef.h**
- SIOM\_ID : **tonc\_memdef.h**
- SIOM\_ID\_MASK : **tonc\_memdef.h**
- SIOM\_ID\_SHIFT : **tonc\_memdef.h**
- SIOM\_SD : **tonc\_memdef.h**
- SIOM\_SI : **tonc\_memdef.h**
- SIOM\_SLAVE : **tonc\_memdef.h**
- SION\_256KHZ : **tonc\_memdef.h**
- SION\_2MHZ : **tonc\_memdef.h**
- SION\_CLK\_EXT : **tonc\_memdef.h**
- SION\_CLK\_INT : **tonc\_memdef.h**
- SION\_ENABLE : **tonc\_memdef.h**
- SION\_RECV\_HIGH : **tonc\_memdef.h**
- SION\_SEND\_HIGH : **tonc\_memdef.h**
- SIOU\_115200 : **tonc\_memdef.h**
- SIOU\_38400 : **tonc\_memdef.h**
- SIOU\_57600 : **tonc\_memdef.h**
- SIOU\_7BIT : **tonc\_memdef.h**
- SIOU\_8BIT : **tonc\_memdef.h**
- SIOU\_9600 : **tonc\_memdef.h**
- SIOU\_BAUD : **tonc\_memdef.h**
- SIOU\_BAUD\_MASK : **tonc\_memdef.h**
- SIOU\_BAUD\_SHIFT : **tonc\_memdef.h**
- SIOU\_CTS : **tonc\_memdef.h**
- SIOU\_ERROR : **tonc\_memdef.h**
- SIOU\_PARITY EVEN : **tonc\_memdef.h**
- SIOU\_PARITY ODD : **tonc\_memdef.h**

- SIOU\_RECV : [tonc\\_memdef.h](#)
- SIOU\_RECV\_EMPTY : [tonc\\_memdef.h](#)
- SIOU\_SEND : [tonc\\_memdef.h](#)
- SIOU\_SEND\_FULL : [tonc\\_memdef.h](#)
- SND\_RATE : [tonc\\_core.h](#)
- sram\_mem : [tonc\\_memmap.h](#)
- SSQR\_DEC : [tonc\\_memdef.h](#)
- SSQR\_DUTY1\_2 : [tonc\\_memdef.h](#)
- SSQR\_DUTY1\_4 : [tonc\\_memdef.h](#)
- SSQR\_DUTY1\_8 : [tonc\\_memdef.h](#)
- SSQR\_DUTY3\_4 : [tonc\\_memdef.h](#)
- SSQR\_INC : [tonc\\_memdef.h](#)
- SSTAT\_DISABLE : [tonc\\_memdef.h](#)
- SSTAT\_ENABLE : [tonc\\_memdef.h](#)
- SSTAT\_NOISE : [tonc\\_memdef.h](#)
- SSTAT\_SQR1 : [tonc\\_memdef.h](#)
- SSTAT\_SQR2 : [tonc\\_memdef.h](#)
- SSTAT\_WAVE : [tonc\\_memdef.h](#)
- SSW\_DEC : [tonc\\_memdef.h](#)
- SSW\_INC : [tonc\\_memdef.h](#)
- SSW\_OFF : [tonc\\_memdef.h](#)
- STR : [tonc\\_core.h](#)
- SWAP : [tonc\\_math.h](#)
- SWAP2 : [tonc\\_math.h](#)
- SWAP3 : [tonc\\_math.h](#)
- swi\_call : [tonc\\_bios.h](#)

>Main Page   Modules   Data Structures   Files   Related Pages

File List   Globals

All   Functions   Variables   Typedefs   Enumerations   Enumerator   Defines

\_ a b c d e f g i k m o p r s t v w x

## - t -

- tile8\_mem : [tonc\\_memmap.h](#)
- tile8\_mem\_obj : [tonc\\_memmap.h](#)
- tile\_mem : [tonc\\_memmap.h](#)
- tile\_mem\_obj : [tonc\\_memmap.h](#)
- TM CASCADE : [tonc\\_memdef.h](#)
- TM\_ENABLE : [tonc\\_memdef.h](#)
- TM\_FREQ\_1 : [tonc\\_memdef.h](#)
- TM\_FREQ\_1024 : [tonc\\_memdef.h](#)
- TM\_FREQ\_256 : [tonc\\_memdef.h](#)
- TM\_FREQ\_64 : [tonc\\_memdef.h](#)
- TM\_FREQ\_SYS : [tonc\\_memdef.h](#)
- TM\_IRQ : [tonc\\_memdef.h](#)
- TTE\_BASE\_VARS : [tonc\\_tte.h](#)
- TTE\_CHAR\_VARS : [tonc\\_tte.h](#)
- TTE\_DST\_VARS : [tonc\\_tte.h](#)
- tte\_printf : [tonc\\_tte.h](#)
- TTE\_TAB\_WIDTH : [tonc\\_tte.h](#)

## - V -

- vid\_mem : [tonc\\_memmap.h](#)
- vid\_mem\_back : [tonc\\_memmap.h](#)
- vid\_mem\_front : [tonc\\_memmap.h](#)
- VRAM\_BG\_SIZE : [tonc\\_memmap.h](#)
- VRAM\_OBJ\_SIZE : [tonc\\_memmap.h](#)
- VRAM\_PAGE\_SIZE : [tonc\\_memmap.h](#)
- vwf\_default : [tonc\\_tte.h](#)

>Main Page   Modules   Data Structures   Files   Related Pages

File List   Globals

All   Functions   Variables   Typedefs   Enumerations   Enumerator   Defines

\_   a   b   c   d   e   f   g   i   k   m   o   p   r   s   t   v   w   x

## - W -

- WIN\_ALL : [tonc\\_memdef.h](#)
- WIN\_BG0 : [tonc\\_memdef.h](#)
- WIN\_BG1 : [tonc\\_memdef.h](#)
- WIN\_BG2 : [tonc\\_memdef.h](#)
- WIN\_BG3 : [tonc\\_memdef.h](#)
- WIN\_BLD : [tonc\\_memdef.h](#)
- WIN\_BUILD : [tonc\\_memdef.h](#)
- WIN\_LAYER : [tonc\\_memdef.h](#)
- WIN\_LAYER\_MASK : [tonc\\_memdef.h](#)
- WIN\_LAYER\_SHIFT : [tonc\\_memdef.h](#)
- WIN\_OBJ : [tonc\\_memdef.h](#)
- WININ\_BUILD : [tonc\\_memdef.h](#)
- WINOUT\_BUILD : [tonc\\_memdef.h](#)
- WRAP : [tonc\\_math.h](#)
- WS\_SRAM\_4 : [tonc\\_memdef.h](#)

[Main Page](#)[Modules](#)[Data Structures](#)[Files](#)[Related Pages](#)[File List](#)[Globals](#)[All](#)[Functions](#)[Variables](#)[Typedefs](#)[Enumerations](#)[Enumerator](#)[Defines](#)[\\_](#)[a](#)[b](#)[c](#)[d](#)[e](#)[f](#)[g](#)[i](#)[k](#)[m](#)[o](#)[p](#)[r](#)[s](#)[t](#)[v](#)[w](#)[x](#)

- X -

- XSTR : [tonc\\_core.h](#)

---

Generated on Mon Aug 25 17:03:58 2008 for libtonc by  doxygen 1.5.3