RTEMS i386 VBE framebuffer support

Generated by Doxygen 1.8.6

Mon Dec 22 2014 01:27:49

Contents

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Module Documentation

4.1 Basic Definitions

Macros

- #define **TRUE** 1

  This ensures that RTEMS has TRUE defined in all situations.
- #define **SCORE_EXTERN** extern

  The following ensures that all data is declared in the space of the initialization routine for either the Initialization Manager or the initialization file for the appropriate API.
- #define **SAPI_EXTERN** extern

  The following ensures that all data is declared in the space of the initialization routine for either the Initialization Manager or the initialization file for the appropriate API.
- #define **RTEMS_EXTERN** extern

  The following ensures that all data is declared in the space of the initialization routine for either the Initialization Manager or the initialization file for the appropriate API.
- #define **POSIX_EXTERN** extern

  The following ensures that all data is declared in the space of the initialization routine for either the Initialization Manager or the initialization file for the appropriate API.
- #define **RTEMS_INLINE_ROUTINE** static inline

  The following (in conjunction with compiler arguments) are used to choose between the use of static inline functions and macro functions.
- #define **RTEMS_COMPILER_MEMORY_BARRIER**()

  The following macro is a compiler specific way to ensure that memory writes are not reordered around certian points.
- #define **RTEMS_COMPILER_NO_RETURN_ATTRIBUTE**

  The following macro is a compiler specific way to indicate that the method will NOT return to the caller.
- #define **RTEMS_COMPILER_PURE_ATTRIBUTE**

  The following defines a compiler specific attribute which informs the compiler that the method has no effect except the return value and that the return value depends only on parameters and/or global variables.
- #define **RTEMS_COMPILER_DEPRECATED_ATTRIBUTE**

  Instructs the compiler to issue a warning whenever a variable or function with this attribute will be used.
- #define **RTEMS_COMPILER_UNUSED_ATTRIBUTE**

  Instructs the compiler that a specific variable is deliberately unused.
- #define **RTEMS_COMPILER_PACKED_ATTRIBUTE**

  Instructs the compiler that a specific structure or union members will be placed so that the least memory is used.
- #define **RTEMS_STATIC_ASSERT**(cond, msg) typedef int rtems_static_assert_ ## msg [(cond) ? 1 : -1]
- #define **RTEMS_ARRAY_SIZE**(array) (sizeof(array) / sizeof((array)[0]))
- #define **RTEMS_ZERO_LENGTH_ARRAY** 0

- #define **RTEMS_CONTAINER_OF**(_m, _type, _member_name) ((_type *) ((uintptr_t) (_m) - offsetof(_type, _member_name)))

  Returns a pointer to the container of a specified member pointer.
- #define **RTEMS_TYPEOF_REFX**(_ptr_level, _ptr_type) typeof(_ptr_level(union { int z; typeof(_ptr_type) x; }){0}.x)
- #define **RTEMS_DECONST**(_type, _var) ((_type)(uintptr_t)(const void *) (_var))

  Removes the const qualifier from a type of a variable.
- #define **RTEMS_DEVOLATILE**(_type, _var) ((_type)(uintptr_t)(volatile void *) (_var))

  Removes the volatile qualifier from a type of a variable.
- #define **RTEMS_DEQUALIFY**(_type, _var) ((_type)(uintptr_t)(const volatile void *) (_var))

  Removes the all qualifiers from a type of a variable.

Typedefs

- typedef void * **proc_ptr**

  XXX: Eventually proc_ptr needs to disappear!!!

4.1.1 Detailed Description

4.1.2 Macro Definition Documentation

4.1.2.1 #define POSIX_EXTERN extern

The following ensures that all data is declared in the space of the initialization routine for either the Initialization Manager or the initialization file for the appropriate API.

It is referenced as "external" in every other file.

4.1.2.2 #define RTEMS_COMPILER_MEMORY_BARRIER( )

The following macro is a compiler specific way to ensure that memory writes are not reordered around certian points.

This specifically can impact interrupt disable and thread dispatching critical sections.

Referenced by i386_raw_gdt_entry().

4.1.2.3 #define RTEMS_COMPILER_NO_RETURN_ATTRIBUTE

The following macro is a compiler specific way to indicate that the method will NOT return to the caller.

This can assist the compiler in code generation and avoid unreachable paths. This can impact the code generated following calls to rtems_fatal_error_occurred and _Terminate.

4.1.2.4 #define RTEMS_COMPILER_UNUSED_ATTRIBUTE

Instructs the compiler that a specific variable is deliberately unused.

This can occur when reading volatile device memory or skipping arguments in a variable argument method.

4.1.2.5 #define RTEMS_CONTAINER_OF(*_m, _type, _member_name*) ((_type *) ((uintptr_t) (_m) - offsetof(_type, _member_name)))

Returns a pointer to the container of a specified member pointer.

Parameters

in	*_m*	The pointer to a member of the container.
in	*_type*	The type of the container.
in	*_member_name*	The designator name of the container member.

4.1.2.6 #define RTEMS_DECONST(_type, _var) ((_type)(uintptr_t)(const void *) (_var))

Removes the const qualifier from a type of a variable.

Parameters

in	*_type*	The target type for the variable.
in	*_var*	The variable.

4.1.2.7 #define RTEMS_DEQUALIFY(_type, _var) ((_type)(uintptr_t)(const volatile void *) (_var))

Removes the all qualifiers from a type of a variable.

Parameters

in	*_type*	The target type for the variable.
in	*_var*	The variable.

4.1.2.8 #define RTEMS_DEVOLATILE(_type, _var) ((_type)(uintptr_t)(volatile void *) (_var))

Removes the volatile qualifier from a type of a variable.

Parameters

in	*_type*	The target type for the variable.
in	*_var*	The variable.

4.1.2.9 #define RTEMS_EXTERN extern

The following ensures that all data is declared in the space of the initialization routine for either the Initialization Manager or the initialization file for the appropriate API.

It is referenced as "external" in every other file.

4.1.2.10 #define RTEMS_INLINE_ROUTINE static inline

The following (in conjunction with compiler arguments) are used to choose between the use of static inline functions and macro functions.

The static inline implementation allows better type checking with no cost in code size or execution speed.

4.1.2.11 #define SAPI_EXTERN extern

The following ensures that all data is declared in the space of the initialization routine for either the Initialization Manager or the initialization file for the appropriate API.

It is referenced as "external" in every other file.

4.1.2.12 #define SCORE_EXTERN extern

The following ensures that all data is declared in the space of the initialization routine for either the Initialization Manager or the initialization file for the appropriate API.

It is referenced as "external" in every other file.

Chapter 5

Data Structure Documentation

5.1 __rtems_raw_irq_connect_data__ Struct Reference

Data Fields

- rtems_vector_offset **idtIndex**
- rtems_raw_irq_hdl **hdl**
- rtems_raw_irq_enable **on**
- rtems_raw_irq_disable **off**
- rtems_raw_irq_is_enabled **isOn**

The documentation for this struct was generated from the following file:

- c/src/lib/libcpu/i386/cpu.h

5.2 EDID_color_point_data Struct Reference

Data Fields

- uint8_t **ColorPointWhitePointIndexNumber**
- uint8_t **ColorPointWhiteLowBits**
- uint8_t **ColorPointWhite_x**
- uint8_t **ColorPointWhite_y**
- uint8_t **ColorPointWhiteGamma**

The documentation for this struct was generated from the following file:

- c/src/lib/libbsp/i386/pc386/include/**edid.h**

5.3 EDID_CVT_3_byte_code_descriptor Struct Reference

Data Fields

- uint8_t **AddressableLinesLow**
- uint8_t **AspectRatio_AddressableLinesHigh**
- uint8_t **VerticalRate_PreferredVerticalRate**

The documentation for this struct was generated from the following file:

- c/src/lib/libbsp/i386/pc386/include/**edid.h**

5.4 EDID_CVT_timing_codes_3B Struct Reference

Collaboration diagram for EDID_CVT_timing_codes_3B:

Data Fields

- uint8_t **VersionNumber**
- **EDID_CVT_3_byte_code_descriptor cvt** [4]

The documentation for this struct was generated from the following file:

- c/src/lib/libbsp/i386/pc386/include/**edid.h**

5.5 EDID_detailed_timing_descriptor Struct Reference

Data Fields

- uint8_t **PixelClock_div10000** [2]
- uint8_t **HorizontalActiveLow**
- uint8_t **HorizontalBlankingLow**
- uint8_t **HorizontalBlanking_ActiveHigh**
- uint8_t **VerticalActiveLow**
- uint8_t **VerticalBlankingLow**
- uint8_t **VerticalBlanking_ActiveHigh**
- uint8_t **HorizontalSyncOffsetLow**
- uint8_t **HorizontalSyncPulseWidthLow**
- uint8_t **VerticalSyncPulseWidth_OffsetLow**
- uint8_t **Vert_Hor_SyncPulseWidth_Offset_High**
- uint8_t **HorizontalImageSizeLow**
- uint8_t **VerticalImageSizeLow**
- uint8_t **Vertical_HorizontalImageSizeHigh**
- uint8_t **HorizontalBorder**
- uint8_t **VerticalBorder**
- uint8_t **Flags**

The documentation for this struct was generated from the following file:

- c/src/lib/libbsp/i386/pc386/include/**edid.h**

5.6 EDID_DTD_MD Union Reference

Collaboration diagram for EDID_DTD_MD:

Data Fields

- **EDID_detailed_timing_descriptor dtd**
- **EDID_monitor_descriptor md**

The documentation for this union was generated from the following file:

- c/src/lib/libbsp/i386/pc386/include/**edid.h**

5.7 EDID_edid1 Struct Reference

Collaboration diagram for EDID_edid1:

Data Fields

- uint8_t **Header** [8]
- uint8_t **IDManufacturerName** [2]
- uint8_t **IDProductCode** [2]
- uint8_t **IDSerialNumber** [4]
- uint8_t **WeekofManufacture**
- uint8_t **YearofManufacture**
- uint8_t **Version**
- uint8_t **Revision**
- uint8_t **VideoInputDefinition**
- uint8_t **MaxHorizontalImageSize**
- uint8_t **MaxVerticalImageSize**
- uint8_t **DisplayTransferCharacteristic**
- uint8_t **Features**
- uint8_t **GreenRedLow**
- uint8_t **WhiteBlueLow**
- uint8_t **RedXHigh**
- uint8_t **RedYHigh**
- uint8_t **GreenXHigh**
- uint8_t **GreenYHigh**
- uint8_t **BlueXHigh**
- uint8_t **BlueYHigh**
- uint8_t **WhiteXHigh**
- uint8_t **WhiteYHigh**
- uint8_t **EST_I_II_Man** [3]
- **EDID_standard_timing_identification STI** [8]
- union **EDID_DTD_MD dtd_md** [4]
- uint8_t **ExtensionFlag**
- uint8_t **Checksum**

The documentation for this struct was generated from the following file:

- c/src/lib/libbsp/i386/pc386/include/**edid.h**

5.8 EDID_established_timings_III Struct Reference

Data Fields

- uint8_t **RevisionNumber**
- uint8_t **EST_III** [12]

The documentation for this struct was generated from the following file:

- c/src/lib/libbsp/i386/pc386/include/**edid.h**

5.9 EDID_monitor_descriptor Struct Reference

Data Fields

- uint8_t **Flag0** [2]
- uint8_t **Flag1**
- uint8_t **DataTypeTag**
- uint8_t **Flag2**
- uint8_t **DescriptorData** [13]

The documentation for this struct was generated from the following file:

- c/src/lib/libbsp/i386/pc386/include/**edid.h**

5.10 EDID_monitor_range_limits Struct Reference

Data Fields

- uint8_t **MinVerticalRateInHz**
- uint8_t **MaxVerticalRateInHz**
- uint8_t **MinHorizontalInKHz**
- uint8_t **MaxHorizontalInKHz**
- uint8_t **MaxSupportedPixelClockIn10MHz**
- uint8_t **GTFStandard** [8]

The documentation for this struct was generated from the following file:

- c/src/lib/libbsp/i386/pc386/include/**edid.h**

5.11 EDID_standard_timing_identification Struct Reference

Data Fields

- uint8_t **HorizontalActivePixels**
- uint8_t **ImageAspectRatio_RefreshRate**

The documentation for this struct was generated from the following file:

- c/src/lib/libbsp/i386/pc386/include/**edid.h**

5.12 i386_realmode_interrupt_registers Struct Reference

Used for passing and retrieving registers content to/from real mode interrupt call.

`#include <realmode_int.h>`

Data Fields

- uint32_t **reg_eax**
- uint32_t **reg_ebx**
- uint32_t **reg_ecx**
- uint32_t **reg_edx**
- uint32_t **reg_esi**
- uint32_t **reg_edi**
- uint16_t **reg_ds**
- uint16_t **reg_es**
- uint16_t **reg_fs**
- uint16_t **reg_gs**

5.12.1 Detailed Description

Used for passing and retrieving registers content to/from real mode interrupt call.

The documentation for this struct was generated from the following file:

- c/src/lib/libbsp/i386/shared/realmode_int/**realmode_int.h**

5.13 la_bits Struct Reference

Data Fields

- unsigned int **offset**: 12
- unsigned int **page**: 10
- unsigned int **directory**: 10

The documentation for this struct was generated from the following file:

- c/src/lib/libcpu/i386/cpu.h

5.14 linear_address Union Reference

Collaboration diagram for linear_address:

Data Fields

- **la_bits bits**
- unsigned int **address**

The documentation for this union was generated from the following file:

- c/src/lib/libcpu/i386/cpu.h

5.15 Mode_params Struct Reference

Basic graphic's mode parameters.

Data Fields

- uint16_t **mode_number**

  number of the graphic's mode
- uint16_t **resX**

  number of pixels in one line
- uint16_t **resY**

  number of lines
- uint8_t **bpp**

  bits per pixel

5.15.1 Detailed Description

Basic graphic's mode parameters.

The documentation for this struct was generated from the following file:

- c/src/lib/libbsp/i386/pc386/console/**fb_vesa_rm.c**

5.16 page_dir_bits Struct Reference

Data Fields

- unsigned int **present**: 1
- unsigned int **writable**: 1
- unsigned int **user**: 1
- unsigned int **write_through**: 1
- unsigned int **cache_disable**: 1
- unsigned int **accessed**: 1
- unsigned int **reserved1**: 1
- unsigned int **page_size**: 1
- unsigned int **reserved2**: 1
- unsigned int **available**: 3
- unsigned int **page_frame_address**: 20

The documentation for this struct was generated from the following file:

- c/src/lib/libcpu/i386/cpu.h

5.17 page_dir_entry Union Reference

Collaboration diagram for page_dir_entry:

Data Fields

- **page_dir_bits bits**
- unsigned int **dir_entry**

The documentation for this union was generated from the following file:

- c/src/lib/libcpu/i386/cpu.h

5.18 page_directory Struct Reference

Collaboration diagram for page_directory:

Data Fields

- **page_dir_entry pageDirEntry** [MAX_ENTRY]

The documentation for this struct was generated from the following file:

- c/src/lib/libcpu/i386/cpu.h

5.19 page_table Struct Reference

Collaboration diagram for page_table:

Data Fields

- **page_table_entry pageTableEntry** [MAX_ENTRY]

The documentation for this struct was generated from the following file:

- c/src/lib/libcpu/i386/cpu.h

5.20 page_table_bits Struct Reference

Data Fields

- unsigned int **present**: 1
- unsigned int **writable**: 1
- unsigned int **user**: 1
- unsigned int **write_through**: 1
- unsigned int **cache_disable**: 1
- unsigned int **accessed**: 1
- unsigned int **dirty**: 1
- unsigned int **reserved2**: 2
- unsigned int **available**: 3
- unsigned int **page_frame_address**: 20

The documentation for this struct was generated from the following file:

- c/src/lib/libcpu/i386/cpu.h

5.21 page_table_entry Union Reference

Collaboration diagram for page_table_entry:

Data Fields

- **page_table_bits bits**
- unsigned int **table_entry**

The documentation for this union was generated from the following file:

- c/src/lib/libcpu/i386/cpu.h

5.22 pm_bkp_and_param Struct Reference

backup values, pointers/parameters accessible in protected mode

Data Fields

- uint16_t **idtr_lim_bkp**

  spot for backup protected mode interrupt descriptor table register
- uint32_t **idtr_base_bkp**
- uint16_t **es_bkp**

  spot to backup of ES register value in 32bit protected mode
- uint16_t **fs_bkp**

  spot to backup of FS register value in 32bit protected mode
- uint16_t **gs_bkp**

  spot to backup of GS register value in 32bit protected mode
- uint32_t **rml_entry**

  values for indirect jump to 16bit protected mode
- uint16_t **rml_code_selector**
- uint16_t **rml_data_selector**

  data selector for 16bit protected mode
- uint16_t **rm_stack_segment**

  values determinig location of real mode stack
- uint16_t **rm_stack_pointer**
- uint16_t **rm_data_segment**

  data segment for real mode

5.22.1 Detailed Description

backup values, pointers/parameters accessible in protected mode

Note

Struct members not necessarily used in C. This serves also as layout of memory and it is used within inline assembler.

5.22.2 Field Documentation

5.22.2.1 uint32_t pm_bkp_and_param::idtr_base_bkp

See Also

idtr_lim_bkp (p. 18)

5.22.2.2 uint16_t pm_bkp_and_param::rm_stack_pointer

See Also

rm_stack_segment (p. 18)

Referenced by i386_real_interrupt_call().

5.22.2.3 uint16_t pm_bkp_and_param::rml_code_selector

See Also

rml_entry (p. 18)

Referenced by i386_real_interrupt_call().

The documentation for this struct was generated from the following file:

- c/src/lib/libbsp/i386/shared/realmode_int/**realmode_int.c**

5.23 rm_int_regs_bkp_param Struct Reference

parameters, results, backup values accessible in real mode

Collaboration diagram for rm_int_regs_bkp_param:

Data Fields

- **i386_realmode_interrupt_registers inoutregs**
- uint32_t **pm_esp_bkp**
  spot for back up of protected mode stack pointer
- uint16_t **pm_ss_bkp**
  spot for back up of protected mode stack selector
- uint16_t **ds_bkp**
  spot for back up of protected mode data selector
- uint16_t **rm_entry**
  spot for setting up long indirect jump offset to real mode from 16bit protected mode
- uint16_t **rm_code_segment**
  spot for setting up long indirect jump segment to real mode from 16bit protected mode
- uint32_t **pm_entry**
  returning offset for long indirect jump back to 32bit protected mode
- uint16_t **pm_code_selector**
  returning selector for long indirect jump back to 32bit protected mode

5.23.1 Detailed Description

parameters, results, backup values accessible in real mode

Note

Struct members not necessarily used in C. This serves also as layout of memory and it is used within inline assembler.

The documentation for this struct was generated from the following file:

- c/src/lib/libbsp/i386/shared/realmode_int/**realmode_int.c**

5.24 rtems_raw_irq_global_settings Struct Reference

Collaboration diagram for rtems_raw_irq_global_settings:

Data Fields

- unsigned int **idtSize**
- **rtems_raw_irq_connect_data defaultRawEntry**
- **rtems_raw_irq_connect_data** * **rawIrqHdlTbl**

The documentation for this struct was generated from the following file:

- c/src/lib/libcpu/i386/cpu.h

5.25 segment_descriptors Struct Reference

describes one entry of Global/Local Descriptor Table

`#include <cpu.h>`

Data Fields

- unsigned int **limit_15_0**: 16
- unsigned int **base_address_15_0**: 16
- unsigned int **base_address_23_16**: 8
- unsigned int **type**: 4
- unsigned int **descriptor_type**: 1
- unsigned int **privilege**: 2
- unsigned int **present**: 1
- unsigned int **limit_19_16**: 4
- unsigned int **available**: 1
- unsigned int **fixed_value_bits**: 1
- unsigned int **operation_size**: 1
- unsigned int **granularity**: 1
- unsigned int **base_address_31_24**: 8

5.25.1 Detailed Description

describes one entry of Global/Local Descriptor Table

The documentation for this struct was generated from the following file:

- c/src/lib/libcpu/i386/cpu.h

5.26 VBE_CRTC_info_block Struct Reference

Describes monitor synchronization.

`#include <vbe3.h>`

Data Fields

- uint16_t **HorizontalTotal**

  Horizontal total in pixels.
- uint16_t **HorizontalSyncStart**

  Horizontal sync start in pixels.
- uint16_t **HorizontalSyncEnd**

  Horizontal sync end in pixels.
- uint16_t **VerticalTotal**

  Vertical total in lines.
- uint16_t **VerticalSyncStart**

  Vertical sync start in lines.
- uint16_t **VerticalSyncEnd**

  Vertical sync end in lines.
- uint8_t **Flags**

  Flags (Interlaced, Double Scan etc)
- uint32_t **PixelClock**

  Pixel clock in units of Hz.
- uint16_t **RefreshRate**

  Refresh rate in units of 0.01 Hz.
- uint8_t **Reserved** [40]

  *remainder of **VBE_mode_info_block** (p. 22)*

5.26.1 Detailed Description

Describes monitor synchronization.

The documentation for this struct was generated from the following file:

- c/src/lib/libbsp/i386/pc386/include/**vbe3.h**

5.27 VBE_far_pointer Struct Reference

Far pointer as defined by VBE standard.

`#include <vbe3.h>`

Data Fields

- uint16_t **offset**

  Offset to segment described by selector.
- uint16_t **selector**

  Selector or Segment depending on whether this is used from 16bit protected mode or from real mode.

5.27.1 Detailed Description

Far pointer as defined by VBE standard.

5.27.2 Field Documentation

5.27.2.1 uint16_t VBE_far_pointer::offset

Offset to segment described by *selector*.

5.27.2.2 uint16_t VBE_far_pointer::selector

Selector or Segment depending on whether this is used from 16bit protected mode or from real mode.

The documentation for this struct was generated from the following file:

- c/src/lib/libbsp/i386/pc386/include/**vbe3.h**

5.28 VBE_mode_info_block Struct Reference

Describes graphic's mode parameter.

`#include <vbe3.h>`

Data Fields

- uint16_t **ModeAttributes**
  mode attributes
- uint8_t **WinAAttributes**
  window A attributes
- uint8_t **WinBAttributes**
  window B attributes
- uint16_t **WinGranularity**
  window granularity
- uint16_t **WinSize**
  window size
- uint16_t **WinASegment**
  window A start segment
- uint16_t **WinBSegment**
  window B start segment
- uint32_t * **WinFuncPtr**
  real mode pointer to window function
- uint16_t **BytesPerScanLine**
  bytes per scan line
- uint16_t **XResolution**
  horizontal resolution in px or chars
- uint16_t **YResolution**
  vertical resolution in px or chars
- uint8_t **XCharSize**
  character cell width in pixels
- uint8_t **YCharSize**
  character cell height in pixels
- uint8_t **NumberOfPlanes**
  number of memory planes
- uint8_t **BitsPerPixel**

bits per pixel

- uint8_t **NumberOfBanks**

  number of banks
- uint8_t **MemoryModel**

  memory model type
- uint8_t **BankSize**

  bank size in KB
- uint8_t **NumberOfImagePages**

  number of images
- uint8_t **Reserved0**

  reserved for page function
- uint8_t **RedMaskSize**

  size of direct color red mask in bits
- uint8_t **RedFieldPosition**

  bit position of lsb of red mask
- uint8_t **GreenMaskSize**

  size of direct color green mask in b
- uint8_t **GreenFieldPosition**

  bit position of lsb of green mask
- uint8_t **BlueMaskSize**

  size of direct color blue mask in b
- uint8_t **BlueFieldPosition**

  bit position of lsb of blue mask
- uint8_t **RsvdMaskSize**

  size of direct color reserved mask
- uint8_t **RsvdFieldPosition**

  bit position of lsb of reserved mask
- uint8_t **DirectColorModeInfo**

  direct color mode attributes
- uint32_t * **PhysBasePtr**

  physical address for flat memory frame buffer
- uint32_t **Reserved1**

  Reserved - always set to 0.
- uint16_t **Reserved2**

  Reserved - always set to 0.
- uint16_t **LinBytesPerScanLine**

  bytes per scan line for linear modes
- uint8_t **BnkNumberOfImagePages**

  number of images for banked modes
- uint8_t **LinNumberOfImagePages**

  number of images for linear modes
- uint8_t **LinRedMaskSize**

  size of direct color red mask
- uint8_t **LinRedFieldPosition**

  bit position of lsb of red mask
- uint8_t **LinGreenMaskSize**

  size of direct color green mask
- uint8_t **LinGreenFieldPosition**

  bit position of lsb of green mask
- uint8_t **LinBlueMaskSize**

  size of direct color blue mask

- uint8_t **LinBlueFieldPosition**

  bit position of lsb of blue mask
- uint8_t **LinRsvdMaskSize**

  size of direct color reserved mask
- uint8_t **LinRsvdFieldPosition**

  bit position of lsb of reserved mask
- uint32_t **MaxPixelClock**

  maximum pixel clock (in Hz) for graphics mode
- uint8_t **Reserved3** [189]

  *remainder of **VBE_mode_info_block** (p. 22)*

5.28.1 Detailed Description

Describes graphic's mode parameter.

The documentation for this struct was generated from the following file:

- c/src/lib/libbsp/i386/pc386/include/**vbe3.h**

5.29 VBE_palette_entry Struct Reference

Describes palette entry.

`#include <vbe3.h>`

Data Fields

- uint8_t **Blue**

  Blue channel value (6 or 8 bits)
- uint8_t **Green**

  Green channel value (6 or 8 bits)
- uint8_t **Red**

  Red channel value(6 or 8 bits)
- uint8_t **Alignment**

  DWORD alignment byte (unused)

5.29.1 Detailed Description

Describes palette entry.

The documentation for this struct was generated from the following file:

- c/src/lib/libbsp/i386/pc386/include/**vbe3.h**

5.30 VBE_protected_mode_info_block Struct Reference

Protected mode info block as defined by VBE standard.

`#include <vbe3.h>`

Data Fields

- uint8_t **Signature** [4]
  PM Info Block Signature.
- uint16_t **EntryPoint**
  Offset of PM entry point within BIOS.
- uint16_t **PMInitialize**
  Offset of PM initialization entry point.
- uint16_t **BIOSDataSel**
  Selector to BIOS data area emulation block.
- uint16_t **A0000Sel**
  Selector to access A0000h physical memmory.
- uint16_t **B0000Sel**
  Selector to access B0000h physical memmory.
- uint16_t **B8000Sel**
  Selector to access B8000h physical memmory.
- uint16_t **CodeSegSel**
  Selector to access code segment as data.
- uint8_t **InProtectMode**
  Set to 1 when in protected mode.
- uint8_t **Checksum**
  Checksum byte for structure.

5.30.1 Detailed Description

Protected mode info block as defined by VBE standard.

5.30.2 Field Documentation

5.30.2.1 uint8_t VBE_protected_mode_info_block::Checksum

Checksum byte for structure.

Sum over all structure bytes gives 0.

The documentation for this struct was generated from the following file:

- c/src/lib/libbsp/i386/pc386/include/**vbe3.h**

5.31 VBE_supplemental_vbe_info_block Struct Reference

Supplemental VBE info block.

`#include <vbe3.h>`

Data Fields

- uint8_t **SupVbeSignature** [7]
  Supplemental VBE Signature.
- uint16_t **SupVbeVersion**
  Supplemental VBE Version.
- uint8_t **SupVbeSubFunc** [8]

Bitfield of supported subfunctions.

- uint16_t **OemSoftwareRev**

  OEM Software revision.

- uint8_t * **OemVendorNamePtr**

  VBE_far_pointer *(p. 21) to Vendor Name String.*

- uint8_t * **OemProductNamePtr**

  VBE_far_pointer *(p. 21) to Product Name String.*

- uint8_t * **OemProductRevPtr**

  VBE_far_pointer *(p. 21) to Product Revision String.*

- uint8_t * **OemStringPtr**

  VBE_far_pointer *(p. 21) to OEM String.*

- uint8_t **Reserved** [221]

  Reserved for description strings and future expansion.

5.31.1 Detailed Description

Supplemental VBE info block.

The documentation for this struct was generated from the following file:

- c/src/lib/libbsp/i386/pc386/include/**vbe3.h**

5.32 VBE_vbe_info_block Struct Reference

Information about VBE implementation.

`#include <vbe3.h>`

Data Fields

- uint8_t **VbeSignature** [4]

  VBE Signature.

- uint16_t **VbeVersion**

  VBE Version.

- uint8_t * **OemStringPtr**

  VBE_far_pointer *(p. 21) to OEM String.*

- uint8_t **Capabilities** [4]

  Capabilities of graphics controller.

- uint32_t * **VideoModePtr**

  VBE_far_pointer *(p. 21) to VideoModeList.*

- uint16_t **TotalMemory**

  Number of 64kb memory blocks.

- uint16_t **OemSoftwareRev**

  VBE implementation Software revision.

- uint8_t * **OemVendorNamePtr**

  VBE_far_pointer *(p. 21) to Vendor Name String.*

- uint8_t * **OemProductNamePtr**

  VBE_far_pointer *(p. 21) to Product Name String.*

- uint8_t * **OemProductRevPtr**

  VBE_far_pointer *(p. 21) to Product Revision String.*

- uint8_t **Reserved** [222]

Reserved for VBE implementation scratch.

- uint8_t **OemData** [256]

  Data Area for OEM Strings.

5.32.1 Detailed Description

Information about VBE implementation.

The documentation for this struct was generated from the following file:

- c/src/lib/libbsp/i386/pc386/include/**vbe3.h**

Chapter 6

File Documentation

6.1 c/src/lib/libbsp/i386/pc386/console/fb_vesa_rm.c File Reference

FB driver for graphic hardware compatible with VESA Bios Extension Real mode interface utilized Tested on real HW.

```
#include <bsp.h>
#include <bsp/fb_vesa.h>
#include <bsp/realmode_int.h>
#include <pthread.h>
#include <rtems/libio.h>
#include <rtems/fb.h>
#include <rtems/framebuffer.h>
#include <stdlib.h>
```

Include dependency graph for fb_vesa_rm.c:

Data Structures

- struct **Mode_params**

  Basic graphic's mode parameters.

Macros

- #define **FB_VESA_NAME** "FB_VESA_RM"
- #define **MAX_NO_OF_SORTED_MODES** 100

Functions

- void **vesa_realmode_bootup_init** (void)

  Initializes VBE framebuffer during bootup.
- uint32_t **VBE_controller_information** (**VBE_vbe_info_block** *info_block, uint16_t queried_VBE_Version)

  Returns information about graphic's controller in the `info_block` structure.
- uint32_t **VBE_mode_information** (**VBE_mode_info_block** *info_block, uint16_t mode_number)

  Fills structure `info_block` with informations about selected mode in `mode_number` variable.
- uint32_t **VBE_set_mode** (uint16_t mode_number, **VBE_CRTC_info_block** *info_block)

  Sets graphics mode selected.
- uint32_t **VBE_current_mode** (uint16_t *mode_number)

  Get currently set mode number.

- uint32_t **VBE_report_DDC_capabilities** (uint16_t controller_unit_number, uint8_t *seconds_to_transfer_EDID_block, uint8_t *DDC_level_supported)

  Gets information about display data channel implemented in the graphic's controller.
- uint32_t **VBE_read_EDID** (uint16_t controller_unit_number, uint16_t EDID_block_number, **EDID_edid1** *buffer)

  Reads selected EDID block from display attached to controller's interface.
- rtems_device_driver **frame_buffer_initialize** (rtems_device_major_number major, rtems_device_minor_number minor, void *arg)
- rtems_device_driver **frame_buffer_open** (rtems_device_major_number major, rtems_device_minor_number minor, void *arg)
- rtems_device_driver **frame_buffer_close** (rtems_device_major_number major, rtems_device_minor_number minor, void *arg)
- rtems_device_driver **frame_buffer_read** (rtems_device_major_number major, rtems_device_minor_number minor, void *arg)
- rtems_device_driver **frame_buffer_write** (rtems_device_major_number major, rtems_device_minor_number minor, void *arg)
- rtems_device_driver **frame_buffer_control** (rtems_device_major_number major, rtems_device_minor_number minor, void *arg)

6.1.1 Detailed Description

FB driver for graphic hardware compatible with VESA Bios Extension Real mode interface utilized Tested on real HW. Public sources related:

- VESA BIOS EXTENSION (VBE) Core Function Standard, Ver: 3.0, Sep 16, 1998
- VESA Enhanced Extended Display Identification Data (E-EDID) Standard Release A, Revision 2, September 25, 2006

Hardware is completely initialized upon boot of the system. Therefore there is no way to change graphics mode later.

Interrupt 0x10 is used for entering graphics BIOS.

Driver reads parameter from multiboot command line to setup video: "--video=<resX>x<resY>[-<bpp>]" If cmdline parameter is not specified an attempt for obtaining resolution from display attached is made.

6.1.2 Function Documentation

6.1.2.1 uint32_t VBE_controller_information (VBE_vbe_info_block * *info_block*, uint16_t *queried_VBE_Version*)

Returns information about graphic's controller in the `info_block` structure.

Parameters

`out`	*info_block*	pointer to the struct to be filled with controller information
`in`	*queried_VBE_Version*	if >0x200 then video bios is asked to fill in parameters which appeared with second version of VBE.

Return values

ax	register content as defined in VBE RETURN STATUS paragraph
-1	error calling graphical bios

References i386_get_default_rm_buffer(), i386_Physical_to_real(), i386_real_interrupt_call(), VBE20plus_SIGNATURE, VBE_callSuccessful, VBE_functionSupported, VBE_RetVBEConInf, and VBE_vbe_info_block::VbeSignature.

Referenced by vesa_realmode_bootup_init().

6.1.2.2 uint32_t VBE_current_mode (uint16_t * *mode_number*)

Get currently set mode number.

Parameters

out	*mode_number*	variable to be filled with current mode number

Return values

ax	register content as defined in VBE RETURN STATUS paragraph
-1	error calling graphical bios

References i386_real_interrupt_call(), and VBE_RetCurVBEMod.

6.1.2.3 uint32_t VBE_mode_information (VBE_mode_info_block * *info_block*, uint16_t *mode_number*)

Fills structure `info_block` with informations about selected mode in `mode_number` variable.

Parameters

out	*info_block*	pointer to the struct to be filled with mode information
in	*mode_number*	detailes of this mode to be filled

Return values

ax	register content as defined in VBE RETURN STATUS paragraph
-1	error calling graphical bios

References i386_get_default_rm_buffer(), i386_Physical_to_real(), i386_real_interrupt_call(), VBE_callSuccessful, VBE_functionSupported, and VBE_RetVBEModInf.

Referenced by vesa_realmode_bootup_init().

6.1.2.4 uint32_t VBE_read_EDID (uint16_t *controller_unit_number*, uint16_t *EDID_block_number*, EDID_edid1 * *buffer*)

Reads selected EDID block from display attached to controller's interface.

Parameters

in	*controller_unit_number*	
in	*EDID_block_number*	block no. to be read from the display
out	*buffer*	place to store block fetched from the display

Return values

ax	register content as defined in VBE RETURN STATUS paragraph
-1	error calling graphical bios

References i386_get_default_rm_buffer(), i386_Physical_to_real(), i386_real_interrupt_call(), VBE_callSuccessful, VBE_DisDatCha, VBE_functionSupported, and VBEDDC_ReadEDID.

6.1.2.5 uint32_t VBE_report_DDC_capabilities (uint16_t *controller_unit_number*, uint8_t * *seconds_to_transfer_EDID_block*, uint8_t * *DDC_level_supported*)

Gets information about display data channel implemented in the graphic's controller.

Parameters

in	*controller_unit_number*	

out	*seconds_to_transfer_EDID_block*	approximate time to transfer one EDID block rounded up to seconds
out	*DDC_level_supported*	contains DDC version supported and screen blanking state during transfer

Return values

ax	register content as defined in VBE RETURN STATUS paragraph
-1	error calling graphical bios

References i386_real_interrupt_call(), VBE_DisDatCha, and VBEDDC_Capabilities.

6.1.2.6 uint32_t VBE_set_mode (uint16_t *mode_number,* VBE_CRTC_info_block * *info_block*)

Sets graphics mode selected.

If mode has refreshRateCtrl bit set, than the `info_block` must be filled accordingly.

Parameters

in	*mode_number*	number of mode to be set
in	*info_block*	pointer to struct containing refresh rate control info

Return values

ax	register content as defined in VBE RETURN STATUS paragraph
-1	error calling graphical bios

References i386_get_default_rm_buffer(), i386_Physical_to_real(), i386_real_interrupt_call(), and VBE_SetVBEMod.

Referenced by vesa_realmode_bootup_init().

6.1.2.7 void vesa_realmode_bootup_init (void)

Initializes VBE framebuffer during bootup.

utilizes switches to real mode interrupts and therefore must be called during bootup before tick is set up and real-time interrupt vectors utilized

References VBE_mode_info_block::BitsPerPixel, Mode_params::bpp, i386_get_default_rm_buffer(), i386_Real_to_physical(), VBE_mode_info_block::LinBlueFieldPosition, VBE_mode_info_block::LinBlueMaskSize, VBE_mode_info_block::LinBytesPerScanLine, VBE_mode_info_block::LinGreenFieldPosition, VBE_mode_info_block::LinGreenMaskSize, VBE_mode_info_block::LinRedFieldPosition, VBE_mode_info_block::LinRedMaskSize, VBE_mode_info_block::LinRsvdFieldPosition, VBE_mode_info_block::LinRsvdMaskSize, Mode_params::mode_number, VBE_mode_info_block::ModeAttributes, VBE_mode_info_block::PhysBasePtr, Mode_params::resX, Mode_params::resY, VBE_callFailed, VBE_callSuccessful, VBE_ColorModeMask, VBE_controller_information(), VBE_functionSupported, VBE_GraphicsModeMask, VBE_linearFlatFrameBufMask, VBE_LinFraBufModeAvaiMask, VBE_mode_information(), VBE_modSupInHWMask, VBE_set_mode(), VBE_STUB_VideoModeList, VBE_vbe_info_block::VideoModePtr, VBE_mode_info_block::XResolution, and VBE_mode_info_block::YResolution.

6.2 c/src/lib/libbsp/i386/pc386/include/bsp.h File Reference

Global BSP definitions.

```
#include <bspopts.h>
#include <bsp/default-initial-extension.h>
#include <bsp/tblsizes.h>
#include <rtems.h>
#include <rtems/iosupp.h>
#include <rtems/console.h>
#include <rtems/clockdrv.h>
#include <libcpu/cpu.h>
#include <rtems/bspIo.h>
```

Include dependency graph for bsp.h: This graph shows which files directly or indirectly include this file:

Macros

- #define **BSP_HAS_FRAME_BUFFER** 1

  pc386_i386 PC386 Support
- #define **BSP_NE2000_NETWORK_DRIVER_NAME** "ne1"
- #define **BSP_NE2000_NETWORK_DRIVER_ATTACH** rtems_ne_driver_attach
- #define **BSP_WD8003_NETWORK_DRIVER_NAME** "wd1"
- #define **BSP_WD8003_NETWORK_DRIVER_ATTACH** rtems_wd_driver_attach
- #define **BSP_DEC21140_NETWORK_DRIVER_NAME** "dc1"
- #define **BSP_DEC21140_NETWORK_DRIVER_ATTACH** rtems_dec21140_driver_attach
- #define **BSP_3C509_NETWORK_DRIVER_NAME** "3c1"
- #define **BSP_3C509_NETWORK_DRIVER_ATTACH** rtems_3c509_driver_attach
- #define **RTEMS_BSP_NETWORK_DRIVER_NAME** BSP_DEC21140_NETWORK_DRIVER_NAME
- #define **RTEMS_BSP_NETWORK_DRIVER_ATTACH** BSP_DEC21140_NETWORK_DRIVER_ATTACH
- #define **IO_TIMER1** 0x40
- #define **TIMER_CNTR0** (IO_TIMER1 + 0) /∗ timer 0 counter port ∗/
- #define **TIMER_CNTR1** (IO_TIMER1 + 1) /∗ timer 1 counter port ∗/
- #define **TIMER_CNTR2** (IO_TIMER1 + 2) /∗ timer 2 counter port ∗/
- #define **TIMER_MODE** (IO_TIMER1 + 3) /∗ timer mode port ∗/
- #define **TIMER_SEL0** 0x00 /∗ select counter 0 ∗/
- #define **TIMER_SEL1** 0x40 /∗ select counter 1 ∗/
- #define **TIMER_SEL2** 0x80 /∗ select counter 2 ∗/
- #define **TIMER_INTTC** 0x00 /∗ mode 0, intr on terminal cnt ∗/
- #define **TIMER_ONESHOT** 0x02 /∗ mode 1, one shot ∗/
- #define **TIMER_RATEGEN** 0x04 /∗ mode 2, rate generator ∗/
- #define **TIMER_SQWAVE** 0x06 /∗ mode 3, square wave ∗/
- #define **TIMER_SWSTROBE** 0x08 /∗ mode 4, s/w triggered strobe ∗/
- #define **TIMER_HWSTROBE** 0x0a /∗ mode 5, h/w triggered strobe ∗/
- #define **TIMER_LATCH** 0x00 /∗ latch counter for reading ∗/
- #define **TIMER_LSB** 0x10 /∗ r/w counter LSB ∗/
- #define **TIMER_MSB** 0x20 /∗ r/w counter MSB ∗/
- #define **TIMER_16BIT** 0x30 /∗ r/w counter 16 bits, LSB first ∗/
- #define **TIMER_BCD** 0x01 /∗ count in BCD ∗/
- #define **TIMER_RD_BACK** 0xc0 /∗ Read Back Command ∗/
- #define **RB_NOT_COUNT** 0x40 /∗ Don't select counter latch ∗/
- #define **RB_NOT_STATUS** 0x20 /∗ Don't select status latch ∗/
- #define **RB_COUNT_0** 0x02 /∗ Counter 0 latch ∗/
- #define **RB_COUNT_1** 0x04 /∗ Counter 1 latch ∗/
- #define **RB_COUNT_2** 0x08 /∗ Counter 2 latch ∗/
- #define **RB_OUTPUT** 0x80 /∗ Output of the counter is 1 ∗/
- #define **TIMER_TICK** 1193182 /∗ The internal tick rate in ticks per second ∗/
- #define **BSP_CONSOLE_VGA** 0
- #define **BSP_CONSOLE_COM1** 1

- #define **BSP_CONSOLE_COM2** 2
- #define **US_TO_TICK**(us) (((us)∗105+44)/88)
- #define **TICK_TO_US**(tk) (((tk)∗88+52)/105)
- #define **BSP_CONSOLE_PORT_CONSOLE** (-1)
- #define **BSP_CONSOLE_PORT_COM1** (BSP_UART_COM1)
- #define **BSP_CONSOLE_PORT_COM2** (BSP_UART_COM2)
- #define **RTEMS_BSP_HAS_IDE_DRIVER**
- #define **BSP_MAXIMUM_DEVICES** 6

Typedefs

- typedef __FILE **FILE**

Functions

- void **BSP_runtime_console_select** (int ∗pPrintkPort, int ∗pConsolePort)
- int **rtems_ne_driver_attach** (struct rtems_bsdnet_ifconfig ∗, int)
- int **rtems_wd_driver_attach** (struct rtems_bsdnet_ifconfig ∗, int)
- int **rtems_dec21140_driver_attach** (struct rtems_bsdnet_ifconfig ∗, int)
- int **rtems_3c509_driver_attach** (struct rtems_bsdnet_ifconfig ∗config)
- void **_IBMPC_initVideo** (void)
- void **_IBMPC_outch** (char)
- char **_IBMPC_inch** (void)
- char **_IBMPC_inch_sleep** (void)
- int **BSP_wait_polled_input** (void)
- int **rtems_kbpoll** (void)
- int **getch** (void)
- void **add_to_queue** (unsigned short b)
- void **Wait_X_ms** (unsigned int timeToWait)
- void **Calibrate_loop_1ms** (void)
- void **rtems_irq_mngt_init** (void)
- void **bsp_size_memory** (void)
- void **Clock_driver_install_handler** (void)
- void **Clock_driver_support_initialize_hardware** (void)
- void **kbd_reset_setup** (char ∗str, int ∗ints)
- size_t **read_aux** (char ∗buffer, size_t count)
- bool **bsp_get_serial_mouse_device** (const char ∗∗name, const char ∗∗type)
- void **register_leds** (int console, unsigned int led, unsigned int ∗addr, unsigned int mask)
- const char ∗ **bsp_cmdline** (void)
- const char ∗ **bsp_cmdline_arg** (const char ∗arg)
- void **bsp_ide_cmdline_init** (void)
- void **init_remote_gdb** (void)
- void **i386_stub_glue_init** (int uart)
- void **i386_stub_glue_init_breakin** (void)
- void **breakpoint** (void)
- uint32_t **BSP_irq_count_dump** (FILE ∗f)
- void **raw_idt_notify** (void)
- void **C_dispatch_isr** (int vector)

Variables

- interrupt_gate_descriptor **Interrupt_descriptor_table** [IDT_SIZE]
- **segment_descriptors _Global_descriptor_table** [GDT_SIZE]

6.2.1 Detailed Description

Global BSP definitions.

6.2.2 Macro Definition Documentation

6.2.2.1 #define BSP_HAS_FRAME_BUFFER 1

pc386_i386 PC386 Support

PC386 support.

6.3 c/src/lib/libbsp/i386/pc386/include/edid.h File Reference

VESA EDID definitions.

```
#include <stdint.h>
#include <rtems/score/basedefs.h>
```

Include dependency graph for edid.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct **EDID_detailed_timing_descriptor**
- struct **EDID_color_point_data**
- struct **EDID_monitor_range_limits**
- struct **EDID_CVT_3_byte_code_descriptor**
- struct **EDID_CVT_timing_codes_3B**
- struct **EDID_established_timings_III**
- struct **EDID_monitor_descriptor**
- union **EDID_DTD_MD**
- struct **EDID_standard_timing_identification**
- struct **EDID_edid1**

Macros

- #define **EDID_INLINE_ROUTINE RTEMS_INLINE_ROUTINE**
- #define **EDID1_DTD_Flag_InterlacedOff** 7
- #define **EDID1_DTD_Flag_InterlacedMask** 0x1
- #define **EDID1_DTD_Flag_StereoModeOff** 0
- #define **EDID1_DTD_Flag_StereoModeMask** 0xC1
- #define **EDID1_DTD_Stereo_FldSeqRightOnSync** 0x40
- #define **EDID1_DTD_Stereo_FldSeqLeftOnSync** 0x80
- #define **EDID1_DTD_Stereo_2wItlvdRightOnEven** 0x41
- #define **EDID1_DTD_Stereo_2wItlvdLeftOnEven** 0x81
- #define **EDID1_DTD_Stereo_4wInterleaved** 0xC0
- #define **EDID1_DTD_Stereo_SideBySideItlvd** 0xC1
- #define **EDID1_DTD_Flag_DigitalOff** 4
- #define **EDID1_DTD_Flag_DigitalMask** 0x1
- #define **EDID1_DTD_BipolarAnalogComposSyncOff** 3
- #define **EDID1_DTD_BipolarAnalogComposSyncMask** 0x1
- #define **EDID1_DTD_WithSerrationsOff** 2
- #define **EDID1_DTD_WithSerrationsMask** 0x1
- #define **EDID1_DTD_DigitalSeparateSyncOff** 3

- #define **EDID1_DTD_DigitalSeparateSyncMask** 0x1
- #define **EDID1_DTD_VerticalSyncIsPositiveOff** 2
- #define **EDID1_DTD_VerticalSyncIsPositiveMask** 0x1
- #define **EDID1_DTD_HorizontalSyncIsPositiveOff** 1
- #define **EDID1_DTD_HorizontalSyncIsPositiveMask** 0x1
- #define **EDID_DTT_MonitorSerialNumber** 0xFF
- #define **EDID_DTT_ASCIIString** 0xFE
- #define **EDID_DTT_MonitorRangeLimits** 0xFD
- #define **EDID_DTT_MonitorName** 0xFC
- #define **EDID_DTT_AdditionalColorPointData** 0xFB
- #define **EDID_DTT_AdditionalSTI** 0xFA
- #define **EDID_DTT_DisplayColorManagement** 0xF9
- #define **EDID_DTT_CVT3ByteTimingCodes** 0xF8
- #define **EDID1_CVT_AspectRatioOff** 2
- #define **EDID1_CVT_AspectRatioMask** 0x3
- #define **EDID1_CVT_AddressableLinesHighOff** 4
- #define **EDID1_CVT_AddressableLinesHighMask** 0xF
- #define **EDID1_CVT_VerticalRate60HzRBOff** 0
- #define **EDID1_CVT_VerticalRate60HzRBMask** 0x1
- #define **EDID1_CVT_VerticalRate85HzOff** 1
- #define **EDID1_CVT_VerticalRate85HzMask** 0x1
- #define **EDID1_CVT_VerticalRate75HzOff** 2
- #define **EDID1_CVT_VerticalRate75HzMask** 0x1
- #define **EDID1_CVT_VerticalRate60HzOff** 3
- #define **EDID1_CVT_VerticalRate60HzMask** 0x1
- #define **EDID1_CVT_VerticalRate50HzOff** 4
- #define **EDID1_CVT_VerticalRate50HzMask** 0x1
- #define **EDID1_CVT_PreferredVerticalRateOff** 5
- #define **EDID1_CVT_PreferredVerticalRateMask** 0x3
- #define **EDID_CVT_AspectRatio_4_3** 0
- #define **EDID_CVT_AspectRatio_16_9** 1
- #define **EDID_CVT_AspectRatio_16_10** 2
- #define **EDID_CVT_AspectRatio_15_9** 3
- #define **EDID_CVT_PrefVertRate50Hz** 0
- #define **EDID_CVT_PrefVertRate60Hz** 1
- #define **EDID_CVT_PrefVertRate75Hz** 2
- #define **EDID_CVT_PrefVertRate85Hz** 3
- #define **EDID_DTT_EstablishedTimingsIII** 0xF7
- #define **EDID_DTT_DescriptorSpaceUnused** 0x10
- #define **EDID1_STI_ImageAspectRatioOff** 0
- #define **EDID1_STI_ImageAspectRatioMask** 0x3
- #define **EDID1_STI_RefreshRateOff** 2
- #define **EDID1_STI_RefreshRateMask** 0x3F
- #define **EDID_STI_DescriptorUnused** 0x0101
- #define **EDID_STI_AspectRatio_16_10** 0
- #define **EDID_STI_AspectRatio_4_3** 1
- #define **EDID_STI_AspectRatio_5_4** 2
- #define **EDID_STI_AspectRatio_16_9** 3
- #define **EDID1_VID_DigitalSignalLevelOff** 7
- #define **EDID1_VID_DigitalSignalLevelMask** 0x1
- #define **EDID1_VID_ColorBitDepthOff** 4
- #define **EDID1_VID_ColorBitDepthMask** 0x7 /∗ see CBD ∗/
- #define **EDID1_VID_DigitalVideoStandardSuppOff** 0
- #define **EDID1_VID_DigitalVideoStandardSuppMask** 0xF /∗ see DVS ∗/
- #define **EDID1_VID_SignalLevelStandardOff** 5

- #define **EDID1_VID_SignalLevelStandardMask** 0x3
- #define **EDID1_VID_VideoSetupBlankOff** 4
- #define **EDID1_VID_VideoSetupBlankMask** 0x1
- #define **EDID1_VID_SeparateSyncHandVSignalsOff** 3
- #define **EDID1_VID_SeparateSyncHandVSignalsMask** 0x1
- #define **EDID1_VID_SyncSignalOnHorizontalOff** 2
- #define **EDID1_VID_SyncSignalOnHorizontalMask** 0x1
- #define **EDID1_VID_SyncSignalOnGreenOff** 1
- #define **EDID1_VID_SyncSignalOnGreenMask** 0x1
- #define **EDID1_VID_SerationOnVerticalSyncOff** 0
- #define **EDID1_VID_SerationOnVerticalSyncMask** 0x1
- #define **EDID_SLS_0700_0300_1000Vpp** 0x0
- #define **EDID_SLS_0714_0286_1000Vpp** 0x1
- #define **EDID_SLS_1000_0400_1400Vpp** 0x2
- #define **EDID_SLS_0700_0000_0700Vpp** 0x3
- #define **CBD_undef** 0x0
- #define **CBD_6bPerPrimaryColor** 0x1
- #define **CBD_8bPerPrimaryColor** 0x2
- #define **CBD_10bPerPrimaryColor** 0x3
- #define **CBD_12bPerPrimaryColor** 0x4
- #define **CBD_14bPerPrimaryColor** 0x5
- #define **CBD_16bPerPrimaryColor** 0x6
- #define **CBD_reserved** 0x7
- #define **DVS_undef** 0x0
- #define **DVS_DVI** 0x1
- #define **DVS_HDMI** -a 0x2
- #define **DVS_HDMI** -b 0x3
- #define **DVS_MDDI** 0x4
- #define **DVS_DiplayPort** 0x5
- #define **EDID1_Feature_GTFSupported_mask** 0x1
- #define **EDID1_Feature_GTFSupported_off** 0
- #define **EDID1_Feature_PreferredTimingMode_mask** 0x1
- #define **EDID1_Feature_PreferredTimingMode_off** 1
- #define **EDID1_Feature_StandardDefaultColorSpace_mask** 0x1
- #define **EDID1_Feature_StandardDefaultColorSpace_off** 2
- #define **EDID1_Feature_DisplayType_mask** 0x2
- #define **EDID1_Feature_DisplayType_off** 3
- #define **EDID1_Feature_ActiveOff_mask** 0x1
- #define **EDID1_Feature_ActiveOff_off** 5
- #define **EDID1_Feature_Suspend_mask** 0x1
- #define **EDID1_Feature_Suspend_off** 6
- #define **EDID1_Feature_StandBy_mask** 0x1
- #define **EDID1_Feature_StandBy_off** 7
- #define **EDID_DisplayType_Monochrome** 0
- #define **EDID_DisplayType_RGBcolor** 1
- #define **EDID_DisplayType_nonRGBcolor** 2
- #define **EDID_DisplayType_undef** 3
- #define **EDID_DisplayType_RGB444** 0
- #define **EDID_DisplayType_RGB444YCrCb444** 1
- #define **EDID_DisplayType_RGB444YCrCb422** 2
- #define **EDID_DisplayType_RGB444YCrCb444YCrCb422** 3

Enumerations

- enum **EST_III** {
  EST_1152x864_75Hz = 0, **EST_1024x768_85Hz** = 1, **EST_800x600_85Hz** = 2, **EST_848x480_60Hz** = 3,
  EST_640x480_85Hz = 4, **EST_720x400_85Hz** = 5, **EST_640x400_85Hz** = 6, **EST_640x350_85Hz** = 7,
  EST_1280x1024_85Hz = 8, **EST_1280x1024_60Hz** = 9, **EST_1280x960_85Hz** = 10, **EST_1280x960_60Hz** = 11,
  EST_1280x768_85Hz = 12, **EST_1280x768_75Hz** = 13, **EST_1280x768_60Hz** = 14, **EST_1280x768_60-HzRB** = 15,
  EST_1400x1050_75Hz = 16, **EST_1400x1050_60Hz** = 17, **EST_1400x1050_60HzRB** = 18, **EST_1400x900-_85Hz** = 19,
  EST_1400x900_75Hz = 20, **EST_1400x900_60Hz** = 21, **EST_1400x900_60HzRB** = 22, **EST_1360x768_-60Hz** = 23,
  EST_1600x1200_70Hz = 24, **EST_1600x1200_65Hz** = 25, **EST_1600x1200_60Hz** = 26, **EST_1680x1050-_85Hz** = 27,
  EST_1680x1050_75Hz = 28, **EST_1680x1050_60Hz** = 29, **EST_1680x1050_60HzRB** = 30, **EST_-1400x1050_85Hz** = 31,
  EST_1920x1200_60Hz = 32, **EST_1920x1200_60HzRB** = 33, **EST_1856x1392_75Hz** = 34, **EST_-1856x1392_60Hz** = 35,
  EST_1792x1344_75Hz = 36, **EST_1792x1344_60Hz** = 37, **EST_1600x1200_85Hz** = 38, **EST_1600x1200-_75Hz** = 39,
  EST_1920x1440_75Hz = 44, **EST_1920x1440_60Hz** = 45, **EST_1920x1200_85Hz** = 46, **EST_1920x1200-_75Hz** = 47 }
- enum **edid1_established_timings** {
  EST_800x600_60Hz = 0, **EST_800x600_56Hz** = 1, **EST_640x480_75Hz** = 2, **EST_640x480_72Hz** = 3,
  EST_640x480_67Hz = 4, **EST_640x480_60Hz** = 5, **EST_720x400_88Hz** = 6, **EST_720x400_70Hz** = 7,
  EST_1280x1024_75Hz = 8, **EST_1024x768_75Hz** = 9, **EST_1024x768_70Hz** = 10, **EST_1024x768_60Hz** = 11,
  EST_1024x768_87Hz = 12, **EST_832x624_75Hz** = 13, **EST_800x600_75Hz** = 14, **EST_800x600_72Hz** = 15,
  EST_1152x870_75Hz = 23 }

Functions

- EDID_INLINE_ROUTINE uint16_t **DTD_horizontal_active** (**EDID_detailed_timing_descriptor** *dtd)
- EDID_INLINE_ROUTINE uint16_t **DTD_horizontal_blanking** (**EDID_detailed_timing_descriptor** *dtd)
- EDID_INLINE_ROUTINE uint16_t **DTD_vertical_active** (**EDID_detailed_timing_descriptor** *dtd)
- EDID_INLINE_ROUTINE uint16_t **DTD_vertical_blanking** (**EDID_detailed_timing_descriptor** *dtd)
- EDID_INLINE_ROUTINE uint16_t **DTD_vertical_sync_pulse_width** (**EDID_detailed_timing_descriptor** *dtd)
- EDID_INLINE_ROUTINE uint16_t **DTD_vertical_sync_offset** (**EDID_detailed_timing_descriptor** *dtd)
- EDID_INLINE_ROUTINE uint16_t **DTD_horizontal_sync_pulse_width** (**EDID_detailed_timing_-descriptor** *dtd)
- EDID_INLINE_ROUTINE uint16_t **DTD_horizontal_sync_offset** (**EDID_detailed_timing_descriptor** *dtd)
- EDID_INLINE_ROUTINE uint16_t **DTD_vertical_image_size** (**EDID_detailed_timing_descriptor** *dtd)
- EDID_INLINE_ROUTINE uint16_t **DTD_horizontal_image_size** (**EDID_detailed_timing_descriptor** *dtd)
- EDID_INLINE_ROUTINE uint16_t **edid1_CVT_addressable_lines_high** (**EDID_CVT_3_byte_code_-descriptor** *cvt)
- EDID_INLINE_ROUTINE uint8_t **edid1_CVT_aspect_ratio** (**EDID_CVT_3_byte_code_descriptor** *cvt)
- EDID_INLINE_ROUTINE uint16_t **edid1_RedX** (**EDID_edid1** *edid)
- EDID_INLINE_ROUTINE uint16_t **edid1_RedY** (**EDID_edid1** *edid)
- EDID_INLINE_ROUTINE uint16_t **edid1_GreenX** (**EDID_edid1** *edid)
- EDID_INLINE_ROUTINE uint16_t **edid1_GreenY** (**EDID_edid1** *edid)
- EDID_INLINE_ROUTINE uint16_t **edid1_BlueX** (**EDID_edid1** *edid)
- EDID_INLINE_ROUTINE uint16_t **edid1_BlueY** (**EDID_edid1** *edid)
- EDID_INLINE_ROUTINE uint16_t **edid1_WhiteX** (**EDID_edid1** *edid)

- EDID_INLINE_ROUTINE uint16_t **edid1_WhiteY** (**EDID_edid1** ∗edid)
- EDID_INLINE_ROUTINE uint8_t **edid1_established_tim** (**EDID_edid1** ∗edid, enum edid1_established_timings est)

6.3.1 Detailed Description

VESA EDID definitions. This file contains definitions for constants related to VESA Extended Display Identification Data. More information can be found at `http://www.vesa.org/vesa-standards/free-standards/`
VESA public standards may be found at `http://www.vesa.org/wp-content/uploads/2010/12/thankspublic.htm`

6.4 c/src/lib/libbsp/i386/pc386/include/fb_vesa.h File Reference

Headers specific for framebuffer drivers utilizing VESA VBE.

```
#include <bsp/vbe3.h>
#include <edid.h>
#include <stdint.h>
```

Include dependency graph for fb_vesa.h:

Functions

- uint32_t **VBE_controller_information** (**VBE_vbe_info_block** ∗info_block, uint16_t queried_VBE_Version)

  Returns information about graphic's controller in the `info_block` structure.
- uint32_t **VBE_mode_information** (**VBE_mode_info_block** ∗info_block, uint16_t mode_number)

  Fills structure `info_block` with informations about selected mode in `mode_number` variable.
- uint32_t **VBE_set_mode** (uint16_t mode_number, **VBE_CRTC_info_block** ∗info_block)

  Sets graphics mode selected.
- uint32_t **VBE_current_mode** (uint16_t ∗mode_number)

  Get currently set mode number.
- uint32_t **VBE_report_DDC_capabilities** (uint16_t controller_unit_number, uint8_t ∗seconds_to_transfer_EDID_block, uint8_t ∗DDC_level_supported)

  Gets information about display data channel implemented in the graphic's controller.
- uint32_t **VBE_read_EDID** (uint16_t controller_unit_number, uint16_t EDID_block_number, **EDID_edid1** ∗buffer)

  Reads selected EDID block from display attached to controller's interface.

6.4.1 Detailed Description

Headers specific for framebuffer drivers utilizing VESA VBE.

6.4.2 Function Documentation

6.4.2.1 uint32_t VBE_controller_information (VBE_vbe_info_block ∗ *info_block,* uint16_t *queried_VBE_Version*)

Returns information about graphic's controller in the `info_block` structure.

Parameters

out	*info_block*	pointer to the struct to be filled with controller information
in	*queried_VBE_Version*	if >0x200 then video bios is asked to fill in parameters which appeared with second version of VBE.

Return values

ax	register content as defined in VBE RETURN STATUS paragraph
-1	error calling graphical bios

References i386_get_default_rm_buffer(), i386_Physical_to_real(), i386_real_interrupt_call(), VBE20plus_SIGNATURE, VBE_callSuccessful, VBE_functionSupported, VBE_RetVBEConInf, and VBE_vbe_info_block::VbeSignature.

Referenced by vesa_realmode_bootup_init().

6.4.2.2 uint32_t VBE_current_mode (uint16_t * *mode_number*)

Get currently set mode number.

Parameters

out	*mode_number*	variable to be filled with current mode number

Return values

ax	register content as defined in VBE RETURN STATUS paragraph
-1	error calling graphical bios

References i386_real_interrupt_call(), and VBE_RetCurVBEMod.

6.4.2.3 uint32_t VBE_mode_information (VBE_mode_info_block * *info_block,* uint16_t *mode_number*)

Fills structure `info_block` with informations about selected mode in `mode_number` variable.

Parameters

out	*info_block*	pointer to the struct to be filled with mode information
in	*mode_number*	detailes of this mode to be filled

Return values

ax	register content as defined in VBE RETURN STATUS paragraph
-1	error calling graphical bios

References i386_get_default_rm_buffer(), i386_Physical_to_real(), i386_real_interrupt_call(), VBE_callSuccessful, VBE_functionSupported, and VBE_RetVBEModInf.

Referenced by vesa_realmode_bootup_init().

6.4.2.4 uint32_t VBE_read_EDID (uint16_t *controller_unit_number,* uint16_t *EDID_block_number,* EDID_edid1 * *buffer*)

Reads selected EDID block from display attached to controller's interface.

Parameters

in	*controller_unit_number*	

in	*EDID_block_number*	block no. to be read from the display
out	*buffer*	place to store block fetched from the display

Return values

ax	register content as defined in VBE RETURN STATUS paragraph
-1	error calling graphical bios

References i386_get_default_rm_buffer(), i386_Physical_to_real(), i386_real_interrupt_call(), VBE_callSuccessful, VBE_DisDatCha, VBE_functionSupported, and VBEDDC_ReadEDID.

6.4.2.5 uint32_t VBE_report_DDC_capabilities (uint16_t *controller_unit_number,* uint8_t * *seconds_to_transfer_EDID_block,* uint8_t * *DDC_level_supported*)

Gets information about display data channel implemented in the graphic's controller.

Parameters

in	*controller_unit_number*	
out	*seconds_to_transfer_EDID_block*	approximate time to transfer one EDID block rounded up to seconds
out	*DDC_level_supported*	contains DDC version supported and screen blanking state during transfer

Return values

ax	register content as defined in VBE RETURN STATUS paragraph
-1	error calling graphical bios

References i386_real_interrupt_call(), VBE_DisDatCha, and VBEDDC_Capabilities.

6.4.2.6 uint32_t VBE_set_mode (uint16_t *mode_number,* **VBE_CRTC_info_block** * *info_block*)

Sets graphics mode selected.

If mode has refreshRateCtrl bit set, than the `info_block` must be filled accordingly.

Parameters

in	*mode_number*	number of mode to be set
in	*info_block*	pointer to struct containing refresh rate control info

Return values

ax	register content as defined in VBE RETURN STATUS paragraph
-1	error calling graphical bios

References i386_get_default_rm_buffer(), i386_Physical_to_real(), i386_real_interrupt_call(), and VBE_SetVBEMod.

Referenced by vesa_realmode_bootup_init().

6.5 c/src/lib/libbsp/i386/pc386/include/tblsizes.h File Reference

Sizes of Global and Interrupt descriptor tables.

```
#include <bspopts.h>
```
Include dependency graph for tblsizes.h:

Macros

- #define **IDT_SIZE** (256)
- #define **GDT_SIZE** (3 + NUM_APP_DRV_GDT_DESCRIPTORS)

6.5.1 Detailed Description

Sizes of Global and Interrupt descriptor tables.

6.6 c/src/lib/libbsp/i386/pc386/include/vbe3.h File Reference

VESA Bios Extension definitions.

```
#include <stdint.h>
#include <rtems/score/basedefs.h>
```
Include dependency graph for vbe3.h:

Data Structures

- struct **VBE_far_pointer**

  Far pointer as defined by VBE standard.
- struct **VBE_protected_mode_info_block**

  Protected mode info block as defined by VBE standard.
- struct **VBE_vbe_info_block**

  Information about VBE implementation.
- struct **VBE_mode_info_block**

  Describes graphic's mode parameter.
- struct **VBE_CRTC_info_block**

  Describes monitor synchronization.
- struct **VBE_palette_entry**

  Describes palette entry.
- struct **VBE_supplemental_vbe_info_block**

  Supplemental VBE info block.

Macros

- #define **VBE_functionSupported** 0x4F

  AL == 4Fh: Function is supported
  AL != 4Fh: Function is not supported.
- #define **VBE_callSuccessful** 0x00

  AH == 00h: Function call successful.
- #define **VBE_callFailed** 0x01

  AH == 01h: Function call failed.
- #define **VBE_notSupportedInCurHWConf** 0x02

  AH == 02h: Function is not supported in the current hardware configuration.
- #define **VBE_callInvalid** 0x03

  AH == 03h: Function call invalid in current video mode.

- #define **VBE_modeNumberMask** 0x01FF

  D0-D8 = Mode number.
- #define **VBE_modeNumberShift** 0x00
- #define **VBE_VESAmodeMask** 0x0100

  If D8 == 0, this is not a VESA defined VBE mode
  If D8 == 1, this is a VESA defined VBE mode.
- #define **VBE_VESAmodeShift** 0x08
- #define **VBE_refreshRateCtrlMask** 0x0800

  If D11 == 0, Use current BIOS default refresh rate
  If D11 == 1, Use user specified CRTC values for refresh rate.
- #define **VBE_refreshRateCtrlShift** 0x0B
- #define **VBE_linearFlatFrameBufMask** 0x4000

  If D14 == 0, Use Banked/Windowed Frame Buffer
  If D14 == 1, Use Linear/Flat Frame Buffer.
- #define **VBE_linearFlatFrameBufShift** 0x0E
- #define **VBE_preserveDispMemMask** 0x8000

  If D15 == 0, Clear display memory
  If D15 == 1, Preserve display memory.
- #define **VBE_preserveDispMemShift** 0x0F
- #define **VBE_R640x400C256** 0x100

  15-bit mode, Resolution: 640x400, Colors: 256
- #define **VBE_R640x480C256** 0x101

  15-bit mode, Resolution: 640x480, Colors: 256
- #define **VBE_R800x600C16** 0x102

  15-bit mode, Resolution: 800x600, Colors: 16
- #define **VBE_b7R800x600C16** 0x6A

  7-bit mode, Resolution: 800x600, Colors: 16
- #define **VBE_R800x600C256** 0x103

  15-bit mode, Resolution: 800x600, Colors: 256
- #define **VBE_R1024x768C16** 0x104

  15-bit mode, Resolution: 1024x768, Colors: 16
- #define **VBE_R1024x768C256** 0x105

  15-bit mode, Resolution: 1024x768, Colors: 256
- #define **VBE_R1280x1024C16** 0x106

  15-bit mode, Resolution: 1280x1024, Colors: 16
- #define **VBE_R1280x1024C256** 0x107

  15-bit mode, Resolution: 1280x1024, Colors: 256
- #define **VBE_R320x200C32K** 0x10D

  15-bit mode, Resolution: 320x200, Colors: 32K (1:5:5:5)
- #define **VBE_R320x200C64K** 0x10E

  15-bit mode, Resolution: 320x200, Colors: 64K (5:6:5)
- #define **VBE_R320x200C17M** 0x10F

  15-bit mode, Resolution: 320x200, Colors: 16.8M (8:8:8)
- #define **VBE_R640x480C32K** 0x110

  15-bit mode, Resolution: 640x480, Colors: 32K (1:5:5:5)
- #define **VBE_R640x480C64K** 0x111

  15-bit mode, Resolution: 640x480, Colors: 64K (5:6:5)
- #define **VBE_R640x480C17M** 0x112

  15-bit mode, Resolution: 640x480, Colors: 16.8M (8:8:8)
- #define **VBE_R800x600C32K** 0x113

  15-bit mode, Resolution: 800x600, Colors: 32K (1:5:5:5)
- #define **VBE_R800x600C64K** 0x114

15-bit mode, Resolution: 800x600, Colors: 64K (5:6:5)

- #define **VBE_R800x600C17M** 0x115

  15-bit mode, Resolution: 800x600, Colors: 16.8M (8:8:8)
- #define **VBE_R1024x768C32K** 0x116

  15-bit mode, Resolution: 1024x768, Colors: 32K (1:5:5:5)
- #define **VBE_R1024x768C64K** 0x117

  15-bit mode, Resolution: 1024x768, Colors: 64K (5:6:5)
- #define **VBE_R1024x768C17M** 0x118

  15-bit mode, Resolution: 1024x768, Colors: 16.8M (8:8:8)
- #define **VBE_R1280x1024C32K** 0x119

  15-bit mode, Resolution: 1280x1024, Colors: 32K (1:5:5:5)
- #define **VBE_R1280x1024C64K** 0x11A

  15-bit mode, Resolution: 1280x1024, Colors: 64K (5:6:5)
- #define **VBE_R1280x1024C17M** 0x11B

  15-bit mode, Resolution: 1280x1024, Colors: 16.8M (8:8:8)
- #define **VBE_SpecialMode** 0x81FF
- #define **VBE_C80R60** 0x108

  15-bit mode, Columns: 80, Rows: 60
- #define **VBE_C132R25** 0x109

  15-bit mode, Columns: 132, Rows: 25
- #define **VBE_C132R43** 0x10A

  15-bit mode, Columns: 132, Rows: 43
- #define **VBE_C132R50** 0x10B

  15-bit mode, Columns: 132, Rows: 50
- #define **VBE_C132R60** 0x10C

  15-bit mode, Columns: 132, Rows: 60
- #define **VBE_RetVBEConInf** 0x4F00

  VBE function - Return VBE Controller Information.
- #define **VBE_RetVBEModInf** 0x4F01

  VBE function - Return VBE Mode Information.
- #define **VBE_SetVBEMod** 0x4F02

  VBE function - Set VBE Mode.
- #define **VBE_RetCurVBEMod** 0x4F03

  VBE function - Return Current VBE Mode.
- #define **VBE_SavResSta** 0x4F04

  VBE function - Save/Restore State.
- #define **VBE_DisWinCon** 0x4F05

  VBE function - Display Window Control.
- #define **VBE_SetGetLogScaLinLen** 0x4F06

  VBE function - Set/Get Logical Scan Line Length.
- #define **VBE_SetGetDisSta** 0x4F07

  VBE function - Set/Get Display Start.
- #define **VBE_SetGetDACPalFor** 0x4F08

  VBE function - Set/Get DAC Palette Format.
- #define **VBE_SetGetPalDat** 0x4F09

  VBE function - Set/Get Palette Data.
- #define **VBE_RetVBEProModInt** 0x4F0A

  VBE function - Return VBE Protected Mode Interface.
- #define **VBE_GetSetpixclo** 0x4F0B

  VBE function - Get/Set pixel clock.
- #define **VBE_PowManExt** 0x4F10

VBE function - Power Management Extensions (PM)

- #define **VBE_FlaPanIntExt** 0x4F11

  VBE function - Flat Panel Interface Extensions (FP)

- #define **VBE_AudIntExt** 0x4F13

  VBE function - Audio Interface Extensions (AI)

- #define **VBE_OEMExt** 0x4F14

  VBE function - OEM Extensions.

- #define **VBE_DisDatCha** 0x4F15

  VBE function - Display Data Channel (DDC), Serial Control Interface (SCI)

- #define **VBE_RetVBESupSpeInf** 0x00

  Return VBE Supplemental Specification Information.

- #define **VBE_SIGNATURE** "VESA"

  General VBE signature.

- #define **VBE20plus_SIGNATURE** "VBE2"

  Signature for VBE 2.0 and higher.

- #define **VBE_STUB_VideoModeList** 0xFFFF

  for STUB see VBE CORE FUNCTIONS VERSION 3.0 - Appendix 1

- #define **VBE_END_OF_VideoModeList** 0xFFFF
- #define **VBE_DACswitchableMask** 0x0001

  VBE Info Block - Capabilities
  D0 = 0 DAC is fixed width, with 6 bits per primary color
  D0 = 1 DAC width is switchable to 8 bits per primary color.

- #define **VBE_notVGAcompatibleMask** 0x0002

  VBE Info Block - Capabilities
  D1 = 0 Controller is VGA compatible
  D1 = 1 Controller is not VGA compatible.

- #define **VBE_specialRAMDACopMask** 0x0004

  VBE Info Block - Capabilities
  D2 = 0 Normal RAMDAC operation
  D2 = 1 When programming large blocks of information to the RAMDAC, use the blank bit in Function 09h.

- #define **VBE_hwStereoscopicMask** 0x0008

  VBE Info Block - Capabilities
  D3 = 0 No hardware stereoscopic signaling support
  D3 = 1 Hardware stereoscopic signaling supported by controller.

- #define **VBE_supportEVCconnMask** 0x0010

  VBE Info Block - Capabilities
  D4 = 0 Stereo signaling supported via external VESA stereo connector
  D4 = 1 Stereo signaling supported via VESA EVC connector.

- #define **VBE_modSupInHWMask** 0x0001

  Mode Info Block - Mode Attributes
  D0 = Mode supported by hardware configuration.

- #define **VBE_TTYOutSupByBIOSMask** 0x0004

  Mode Info Block - Mode Attributes
  D2 = TTY Output functions supported by BIOS.

- #define **VBE_ColorModeMask** 0x0008

  Mode Info Block - Mode Attributes
  D3 = Monochrome/color mode (see note below).

- #define **VBE_GraphicsModeMask** 0x0010

  Mode Info Block - Mode Attributes
  D4 = Mode type.

- #define **VBE_VGACompModeMask** 0x0020

  Mode Info Block - Mode Attributes
  D5 = VGA compatible mode.

- #define **VBE_VGACompWinMemModeMask** 0x0040

Mode Info Block - Mode Attributes
D6 = VGA compatible windowed memory mode is available.

- #define **VBE_LinFraBufModeAvaiMask** 0x0080

  Mode Info Block - Mode Attributes
  D7 = Linear frame buffer mode is available.
- #define **VBE_DblScnModeAvaiMask** 0x0100

  Mode Info Block - Mode Attributes
  D8 = Double scan mode is available.
- #define **VBE_InterlModeAvaiMask** 0x0200

  Mode Info Block - Mode Attributes
  D9 = Interlaced mode is available.
- #define **VBE_HWTripBufSupMask** 0x0400

  Mode Info Block - Mode Attributes
  D10 = Hardware triple buffering support.
- #define **VBE_HWSterDispSupMask** 0x0800

  Mode Info Block - Mode Attributes
  D11 = Hardware stereoscopic display support.
- #define **VBE_DualDispStAdrSupMask** 0x1000

  Mode Info Block - Mode Attributes
  D12 = Dual display start address support.
- #define **VBE_RelocWinSupMask** 0x01

  D0 = Relocatable window(s) supported.
- #define **VBE_WinReadableMask** 0x02

  D1 = Window readable.
- #define **VBE_WinWritableMask** 0x04

  D2 = Window writeable.
- #define **VBE_TextMode** 0x00
- #define **VBE_CGAGraphics** 0x01
- #define **VBE_HerculesGraphics** 0x02
- #define **VBE_Planar** 0x03
- #define **VBE_PackedPixel** 0x04
- #define **VBE_NonChain4Color256** 0x05
- #define **VBE_DirectColor** 0x06
- #define **VBE_YUV** 0x07
- #define **VBE_ColRampProgMask** 0x01

  D0 = Color ramp is fixed/programmable.
- #define **VBE_RsvdBitsUsableMask** 0x02

  D1 = Bits in Rsvd field are usable/reserved.
- #define **VBE_GrModeDblScanMask** 0x01

  CRTC Info Block - Flags
  D0 = Double Scan Mode Enable.
- #define **VBE_GrModeInterlMask** 0x02

  CRTC Info Block - Flags
  D1 = Interlaced Mode Enable.
- #define **VBE_HorSncPolNegMask** 0x04

  CRTC Info Block - Flags
  D2 = Horizontal sync polarity.
- #define **VBE_VerSncPolNegMask** 0x08

  CRTC Info Block - Flags
  D3 = Vertical sync polarity.
- #define **VBEDDC_Capabilities** 0x0

  VBE/DDC subfunction - Report VBE/DDC Capabilities.
- #define **VBEDDC_ReadEDID** 0x1

VBE/DDC subfunction - Read EDID.

- #define **VBEDDC_1SupportedMask** 0x1

  0 - DDC1 not supported; 1 - DDC1 supported

- #define **VBEDDC_2SupportedMask** 0x2

  0 - DDC2 not supported; 1 - DDC2 supported

- #define **VBEDDC_scrBlnkDatTrMs** 0x4

  0 - Screen not blanked during data transfer
  1 - Screen blanked during data transfer

- #define **VBESCI_ReportCapabil** 0x10

  VBE/SCI subfunction - Report VBE/SCI Capabilities.

- #define **VBESCI_BegSCLSDACtrl** 0x11

  VBE/SCI subfunction - Begin SCL/SDA control.

- #define **VBESCI_EndSCLSDACtrl** 0x12

  VBE/SCI subfunction - End SCL/SDA control.

- #define **VBESCI_WrtSCLClkLine** 0x13

  VBE/SCI subfunction - Write SCL clock line.

- #define **VBESCI_WrtSDADatLine** 0x14

  VBE/SCI subfunction - Write SDA data line.

- #define **VBESCI_RdySCLClkLine** 0x15

  VBE/SCI subfunction - Read SCL clock line.

- #define **VBESCI_RdySDADatLine** 0x16

  VBE/SCI subfunction - Read SDA data line.

- #define **VBESCI_capSCLwrtMask** 0x1

  Can write to SCL clock line.

- #define **VBESCI_capSDAwrtMask** 0x2

  Can write to SDA data line.

- #define **VBESCI_capSCLrdyMask** 0x4

  Can read from SCL clock line.

- #define **VBESCI_capSDArdyMask** 0x8

  Can read from SDA data line.

6.6.1 Detailed Description

VESA Bios Extension definitions. This file contains definitions for constants related to VBE. More information can be found at `http://www.vesa.org/vesa-standards/free-standards/`. VESA public standards may be found at `http://www.vesa.org/wp-content/uploads/2010/12/thankspublic.htm`.

6.6.2 Macro Definition Documentation

6.6.2.1 #define VBE_ColorModeMask 0x0008

Mode Info Block - Mode Attributes

D3 = Monochrome/color mode (see note below).

0 = Monochrome mode

1 = Color mode

Referenced by vesa_realmode_bootup_init().

6.6.2.2 #define VBE_ColRampProgMask 0x01

D0 = Color ramp is fixed/programmable.

0 = Color ramp is fixed

1 = Color ramp is programmable

6.6.2.3 #define VBE_DblScnModeAvaiMask 0x0100

Mode Info Block - Mode Attributes

D8 = Double scan mode is available.

0 = No

1 = Yes

6.6.2.4 #define VBE_DualDispStAdrSupMask 0x1000

Mode Info Block - Mode Attributes

D12 = Dual display start address support.

0 = No

1 = Yes

6.6.2.5 #define VBE_GraphicsModeMask 0x0010

Mode Info Block - Mode Attributes

D4 = Mode type.

0 = Text mode

1 = Graphics mode

Referenced by vesa_realmode_bootup_init().

6.6.2.6 #define VBE_GrModeDblScanMask 0x01

CRTC Info Block - Flags

D0 = Double Scan Mode Enable.

0 = Graphics mode is not double scanned

1 = Graphics mode is double scanned

6.6.2.7 #define VBE_GrModeInterlMask 0x02

CRTC Info Block - Flags

D1 = Interlaced Mode Enable.

0 = Graphics mode is non-interlaced

1 = Graphics mode is interlaced

6.6.2.8 #define VBE_HorSncPolNegMask 0x04

CRTC Info Block - Flags

D2 = Horizontal sync polarity.

0 = Horizontal sync polarity is positive (+)

1 = Horizontal sync polarity is negative (-)

6.6.2.9 #define VBE_HWSterDispSupMask 0x0800

Mode Info Block - Mode Attributes

D11 = Hardware stereoscopic display support.

0 = No

1 = Yes

6.6.2.10 #define VBE_HWTripBufSupMask 0x0400

Mode Info Block - Mode Attributes

D10 = Hardware triple buffering support.

0 = No

1 = Yes

6.6.2.11 #define VBE_InterlModeAvaiMask 0x0200

Mode Info Block - Mode Attributes

D9 = Interlaced mode is available.

0 = No

1 = Yes

6.6.2.12 #define VBE_LinFraBufModeAvaiMask 0x0080

Mode Info Block - Mode Attributes

D7 = Linear frame buffer mode is available.

0 = No

1 = Yes

Referenced by vesa_realmode_bootup_init().

6.6.2.13 #define VBE_modSupInHWMask 0x0001

Mode Info Block - Mode Attributes

D0 = Mode supported by hardware configuration.

0 = Mode not supported in hardware

1 = Mode supported in hardware

Referenced by vesa_realmode_bootup_init().

6.6.2.14 #define VBE_RelocWinSupMask 0x01

D0 = Relocatable window(s) supported.

0 = Single non-relocatable window only

1 = Relocatable window(s) are supported

6.6.2.15 #define VBE_RsvdBitsUsableMask 0x02

D1 = Bits in Rsvd field are usable/reserved.

0 = Bits in Rsvd field are reserved

1 = Bits in Rsvd field are usable by the application

6.6.2.16 #define VBE_specialRAMDACopMask 0x0004

VBE Info Block - Capabilities

D2 = 0 Normal RAMDAC operation

D2 = 1 When programming large blocks of information to the RAMDAC, use the blank bit in Function 09h.

6.6.2.17 #define VBE_TTYOutSupByBIOSMask 0x0004

Mode Info Block - Mode Attributes

D2 = TTY Output functions supported by BIOS.

0 = TTY Output functions not supported by BIOS

1 = TTY Output functions supported by BIOS

6.6.2.18 #define VBE_VerSncPolNegMask 0x08

CRTC Info Block - Flags

D3 = Vertical sync polarity.

0 = Vertical sync polarity is positive (+)

1 = Vertical sync polarity is negative (-)

6.6.2.19 #define VBE_VGACompModeMask 0x0020

Mode Info Block - Mode Attributes

D5 = VGA compatible mode.

0 = Yes

1 = No

6.6.2.20 #define VBE_VGACompWinMemModeMask 0x0040

Mode Info Block - Mode Attributes

D6 = VGA compatible windowed memory mode is available.

0 = Yes

1 = No

6.6.2.21 #define VBE_WinReadableMask 0x02

D1 = Window readable.

0 = Window is not readable

1 = Window is readable

6.6.2.22 #define VBE_WinWritableMask 0x04

D2 = Window writeable.

0 = Window is not writeable

1 = Window is writeable

6.7 c/src/lib/libbsp/i386/shared/irq/idt.c File Reference

```
#include <libcpu/cpu.h>
#include <bsp/irq.h>
```

Include dependency graph for idt.c:

Functions

- void **create_interrupt_gate_descriptor** (interrupt_gate_descriptor *idtEntry, rtems_raw_irq_hdl hdl)
- rtems_raw_irq_hdl **get_hdl_from_vector** (rtems_vector_offset index)
- int **i386_set_idt_entry** (const **rtems_raw_irq_connect_data** *irq)
- void **_CPU_ISR_install_vector** (uint32_t vector, **proc_ptr** hdl, **proc_ptr** *oldHdl)
- int **i386_get_current_idt_entry** (**rtems_raw_irq_connect_data** *irq)
- int **i386_delete_idt_entry** (const **rtems_raw_irq_connect_data** *irq)
- int **i386_init_idt** (**rtems_raw_irq_global_settings** *config)
- int **i386_get_idt_config** (**rtems_raw_irq_global_settings** **config)
- uint32_t **i386_raw_gdt_entry** (uint16_t segment_selector_index, **segment_descriptors** *sd)

  Allows to set a GDT entry.
- void **i386_fill_segment_desc_base** (uint32_t base, **segment_descriptors** *sd)

  fills `sd` *with provided* `base` *in appropriate fields of* `sd`
- void **i386_fill_segment_desc_limit** (uint32_t limit, **segment_descriptors** *sd)

  fills `sd` *with provided* `limit` *in appropriate fields of* `sd`
- uint32_t **i386_set_gdt_entry** (uint16_t segment_selector_index, uint32_t base, uint32_t limit)
- uint16_t **i386_next_empty_gdt_entry** ()

  Returns next empty descriptor in GDT.
- uint16_t **i386_cpy_gdt_entry** (uint16_t segment_selector_index, **segment_descriptors** *struct_to_fill)

  Copies GDT entry at index `segment_selector` *to structure pointed to by* `struct_to_fill`*.*
- **segment_descriptors** * **i386_get_gdt_entry** (uint16_t segment_selector_index)

  Returns pointer to GDT table at index given by `segment_selector`*.*
- uint32_t **i386_limit_gdt_entry** (**segment_descriptors** *gdt_entry)

  Extracts limit in bytes from GDT entry pointed to by `gdt_entry`*.*

6.7.1 Function Documentation

6.7.1.1 uint16_t i386_cpy_gdt_entry (uint16_t *segment_selector,* segment_descriptors * *struct_to_fill*)

Copies GDT entry at index `segment_selector` to structure pointed to by `struct_to_fill`.

Parameters

in	*segment_selector*	index to GDT table specifying descriptor to copy
out	*struct_to_fill*	pointer to memory where will be descriptor coppied

Return values

0	FAILED segment_selector out of GDT range
<1;65535>	retrieved segment_selector

6.7.1.2 void i386_fill_segment_desc_base (uint32_t *base*, segment_descriptors * *sd*)

fills `sd` with provided `base` in appropriate fields of `sd`

Parameters

in	*base*	32-bit address to be set as descriptor's base
out	*sd*	descriptor being filled with `base`

6.7.1.3 void i386_fill_segment_desc_limit (uint32_t *limit*, segment_descriptors * *sd*)

fills `sd` with provided `limit` in appropriate fields of `sd`

sets granularity bit if necessary

Parameters

in	*limit*	32-bit value representing number of limit bytes
out	*sd*	descriptor being filled with `limit`

6.7.1.4 segment_descriptors* i386_get_gdt_entry (uint16_t *sgmnt_selector*)

Returns pointer to GDT table at index given by `segment_selector`.

Parameters

in	*sgmnt_selector*	index to GDT table for specifying descriptor to get

Return values

NULL	FAILED segment_selector out of GDT range
pointer	to GDT table at `segment_selector`

6.7.1.5 uint32_t i386_limit_gdt_entry (segment_descriptors * *gdt_entry*)

Extracts limit in bytes from GDT entry pointed to by `gdt_entry`.

Parameters

in	*gdt_entry*	pointer to entry from which limit should be retrieved

Return values

limit	value in bytes from GDT entry

6.7.1.6 uint16_t i386_next_empty_gdt_entry (void)

Returns next empty descriptor in GDT.

Number of descriptors that can be returned depends on *GDT_SIZE*

Return values

0	FAILED GDT is full
<1;65535>	segment_selector number as index to GDT

6.7.1.7 uint32_t i386_raw_gdt_entry (uint16_t *segment_selector_index,* segment_descriptors * *sd*)

Allows to set a GDT entry.

Puts global descriptor `sd` to the global descriptor table on index `segment_selector_index`

Parameters

`in`	*segment_selector_index*	index to GDT entry
`in`	*sd*	structure to be coppied to given `segment_selector` in GDT

Return values

0	FAILED out of GDT range or index is 0, which is not valid index in GDT
1	SUCCESS

References RTEMS_COMPILER_MEMORY_BARRIER.

6.8 c/src/lib/libbsp/i386/shared/realmode_int/realmode_int.c File Reference

Real mode interrupt call implementation.

```
#include <bsp/realmode_int.h>
#include <string.h>
#include <rtems/score/cpu.h>
```

Include dependency graph for realmode_int.c:

Data Structures

- struct **rm_int_regs_bkp_param**

  parameters, results, backup values accessible in real mode
- struct **pm_bkp_and_param**

  backup values, pointers/parameters accessible in protected mode

Macros

- #define **IR_EAX_OFF** "0x00"
- #define **IR_EBX_OFF** "0x04"
- #define **IR_ECX_OFF** "0x08"
- #define **IR_EDX_OFF** "0x0C"

- #define **IR_ESI_OFF** "0x10"
- #define **IR_EDI_OFF** "0x14"
- #define **IR_DS_OFF** "0x18"
- #define **IR_ES_OFF** "0x1A"
- #define **IR_FS_OFF** "0x1C"
- #define **IR_GS_OFF** "0x1E"
- #define **BKP_ESP_OFF** "0x20"
- #define **BKP_SS_OFF** "0x24"
- #define **BKP_DS_OFF** "0x26"
- #define **RM_ENTRY** "0x28"
- #define **PM_ENTRY** "0x2C"
- #define **BKP_IDTR_LIM** "0x00"
- #define **BKP_IDTR_BASE** "0x02"
- #define **BKP_ES_OFF** "0x06"
- #define **BKP_FS_OFF** "0x08"
- #define **BKP_GS_OFF** "0x0A"
- #define **RML_ENTRY** "0x0C"
- #define **RML_D_SEL** "0x12"
- #define **RM_SS** "0x14"
- #define **RM_SP** "0x16"
- #define **RM_DS** "0x18"
- #define **REAL_MODE_SPOT** 0x12000
- #define **DEFAULT_BUFFER_SIZE** 512
- #define **STACK_SIZE** 8192
- #define **INT_STACK_TOP** REAL_MODE_SPOT
- #define **__DP_TYPE** uint8_t
- #define **__DP_YES** ((__DP_TYPE)1)
- #define **__DP_NO** ((__DP_TYPE)-1)
- #define **__DP_FAIL** ((__DP_TYPE)0)
- #define **rml_limit** 0xFFFF

Functions

- void ∗ **i386_get_default_rm_buffer** (uint16_t ∗size)

  Returns buffer and its size usable with real mode interrupt call.
- int **i386_real_interrupt_call** (uint8_t interrupt_number, **i386_realmode_interrupt_registers** ∗ir)

  Call to real mode interrupt with specified int NO and processor registers.

6.8.1 Detailed Description

Real mode interrupt call implementation.

6.8.2 Function Documentation

6.8.2.1 void∗ i386_get_default_rm_buffer (uint16_t ∗ *size*)

Returns buffer and its size usable with real mode interrupt call.

Provides position to real mode buffer. It is buffer accessible from real mode context - it is located below address ∼0x100000 in order for it to be accessible This buffer is meant to be pointed to by segReg:GenPurpReg and through this get bigger portion of an information to/from interrupt service routine than just by using register.

Parameters

out	*size*	pointer to variable, where the size of buffer will be filled

Return values

pointer	to buffer

Referenced by VBE_controller_information(), VBE_mode_information(), VBE_read_EDID(), VBE_set_mode(), and vesa_realmode_bootup_init().

6.8.2.2 int i386_real_interrupt_call (uint8_t *interrupt_number,* i386_realmode_interrupt_registers * *ir*)

Call to real mode interrupt with specified int NO and processor registers.

This function allows calling interrupts in real mode and to set processor registers as desired before interrupt call is made and to retrieve the registers content after call was made.

Parameters

in	*interrupt_number*	interrupt number to be called
in	*ir*	pointer to structure containing registers to be passed to interrupt and to retrieve register content after call was made.

Return values

0	call failed (GDT too small or pagin is on)
1	call successful

References i386_Physical_to_real(), rm_int_regs_bkp_param::pm_code_selector, rm_int_regs_bkp_param::pm_entry, rm_int_regs_bkp_param::rm_code_segment, pm_bkp_and_param::rm_data_segment, rm_int_regs_bkp_param::rm_entry, pm_bkp_and_param::rm_stack_pointer, pm_bkp_and_param::rm_stack_segment, pm_bkp_and_param::rml_code_selector, pm_bkp_and_param::rml_data_selector, and pm_bkp_and_param::rml_entry.

Referenced by VBE_controller_information(), VBE_current_mode(), VBE_mode_information(), VBE_read_EDID(), VBE_report_DDC_capabilities(), and VBE_set_mode().

6.9 c/src/lib/libbsp/i386/shared/realmode_int/realmode_int.h File Reference

Definitioins supporting real mode interrupt calls.

```
#include <libcpu/cpu.h>
#include <stdint.h>
```

Include dependency graph for realmode_int.h:

Data Structures

- struct **i386_realmode_interrupt_registers**

  Used for passing and retrieving registers content to/from real mode interrupt call.

Macros

- #define **INTERRUPT_NO_VIDEO_SERVICES** 0x10

Functions

- void * **i386_get_default_rm_buffer** (uint16_t *size)

Returns buffer and its size usable with real mode interrupt call.

- int **i386_real_interrupt_call** (uint8_t interrupt_number, **i386_realmode_interrupt_registers** *ir)

  Call to real mode interrupt with specified int NO and processor registers.

6.9.1 Detailed Description

Definitioins supporting real mode interrupt calls. Interface allows calling given interrupt number with content of the registers defined. For passing or receiving higher amounts of the data there is a buffer accessible from real mode available. Real mode pointer to this buffer is passed to the interrupt in the registers.

6.9.2 Function Documentation

6.9.2.1 void* i386_get_default_rm_buffer (uint16_t * *size*)

Returns buffer and its size usable with real mode interrupt call.

Provides position to real mode buffer. It is buffer accessible from real mode context - it is located below address ~0x100000 in order for it to be accessible This buffer is meant to be pointed to by segReg:GenPurpReg and through this get bigger portion of an information to/from interrupt service routine than just by using register.

Parameters

out	*size*	pointer to variable, where the size of buffer will be filled

Return values

pointer	to buffer

Referenced by VBE_controller_information(), VBE_mode_information(), VBE_read_EDID(), VBE_set_mode(), and vesa_realmode_bootup_init().

6.9.2.2 int i386_real_interrupt_call (uint8_t *interrupt_number,* **i386_realmode_interrupt_registers** * *ir*)

Call to real mode interrupt with specified int NO and processor registers.

This function allows calling interrupts in real mode and to set processor registers as desired before interrupt call is made and to retrieve the registers content after call was made.

Parameters

in	*interrupt_number*	interrupt number to be called
in	*ir*	pointer to structure containing registers to be passed to interrupt and to retrieve register content after call was made.

Return values

0	call failed (GDT too small or pagin is on)
1	call successful

References i386_Physical_to_real(), rm_int_regs_bkp_param::pm_code_selector, rm_int_regs_bkp_param::pm_entry, rm_int_regs_bkp_param::rm_code_segment, pm_bkp_and_param::rm_data_segment, rm_int_regs_bkp_param::rm_entry, pm_bkp_and_param::rm_stack_pointer, pm_bkp_and_param::rm_stack_segment, pm_bkp_and_param::rml_code_selector, pm_bkp_and_param::rml_data_selector, and pm_bkp_and_param::rml_entry.

Referenced by VBE_controller_information(), VBE_current_mode(), VBE_mode_information(), VBE_read_EDID(), VBE_report_DDC_capabilities(), and VBE_set_mode().

6.10 cpukit/score/cpu/i386/rtems/score/i386.h File Reference

Intel I386 CPU Dependent Source.

Macros

- #define **I386_HAS_FPU** 1
- #define **I386_HAS_BSWAP** 1
- #define **CPU_NAME** "Intel i386"
- #define **get_cs**() i386_get_cs()
- #define **get_ds**() i386_get_ds()
- #define **get_es**() i386_get_es()
- #define **get_ss**() i386_get_ss()
- #define **get_fs**() i386_get_fs()
- #define **get_gs**() i386_get_gs()
- #define **CPU_swap_u32**(_value) i386_swap_u32(_value)
- #define **CPU_swap_u16**(_value) i386_swap_u16(_value)
- #define **outport_byte**(_port, _value) i386_outport_byte(_port, _value)
- #define **outport_word**(_port, _value) i386_outport_word(_port, _value)
- #define **outport_long**(_port, _value) i386_outport_long(_port, _value)
- #define **inport_byte**(_port, _value) i386_inport_byte(_port, _value)
- #define **inport_word**(_port, _value) i386_inport_word(_port, _value)
- #define **inport_long**(_port, _value) i386_inport_long(_port, _value)

Functions

- void ∗ **i386_Logical_to_physical** (unsigned short segment, void ∗address)
- void ∗ **i386_Physical_to_logical** (unsigned short segment, void ∗address)
- **RTEMS_INLINE_ROUTINE** void ∗ **i386_Real_to_physical** (uint16_t segment, uint16_t offset)

  Converts real mode pointer {segment, offset} to physical address.
- int **i386_Physical_to_real** (void ∗address, uint16_t ∗segment, uint16_t ∗offset)

  Retreives real mode pointer elements {segmnet, offset} from physical address.

6.10.1 Detailed Description

Intel I386 CPU Dependent Source. This include file contains information pertaining to the Intel i386 processor.

6.10.2 Function Documentation

6.10.2.1 int i386_Physical_to_real (void ∗ *address,* uint16_t ∗ *segment,* uint16_t ∗ *offset*)

Retreives real mode pointer elements {segmnet, offset} from physical address.

i386_Physical_to_real Function returns the highest segment (base) address possible. Example: input address - 0x4B3A2 output segment - 0x4B3A offset - 0x2 input address - 0x10F12E output segment - 0xFFFF offset - 0xF13-E

Parameters

in	*address*	address to be converted, must be less than 0x10FFEF

out	*segment*	segment computed from address
out	*offset*	offset computed from address

Return values

0	address not convertible
1	segment and offset extracted

Referenced by i386_real_interrupt_call(), VBE_controller_information(), VBE_mode_information(), VBE_read_EDID(), and VBE_set_mode().

6.10.2.2 RTEMS_INLINE_ROUTINE void* i386_Real_to_physical (uint16_t *segment*, uint16_t *offset*)

Converts real mode pointer {segment, offset} to physical address.

i386_Real_to_physical

Parameters

in	*segment*	used with offset to compute physical address
in	*offset*	used with segment to compute physical address

Return values

physical	address

Referenced by vesa_realmode_bootup_init().

6.11 cpukit/score/include/rtems/score/basedefs.h File Reference

Basic Definitions.

```
#include <rtems/score/cpuopts.h>
#include <stddef.h>
#include <stdbool.h>
#include <stdint.h>
#include <limits.h>
#include <string.h>
```

Include dependency graph for basedefs.h: This graph shows which files directly or indirectly include this file:

Macros

- #define **TRUE** 1

  This ensures that RTEMS has TRUE defined in all situations.
- #define **SCORE_EXTERN** extern

  The following ensures that all data is declared in the space of the initialization routine for either the Initialization Manager or the initialization file for the appropriate API.
- #define **SAPI_EXTERN** extern

  The following ensures that all data is declared in the space of the initialization routine for either the Initialization Manager or the initialization file for the appropriate API.
- #define **RTEMS_EXTERN** extern

  The following ensures that all data is declared in the space of the initialization routine for either the Initialization Manager or the initialization file for the appropriate API.
- #define **POSIX_EXTERN** extern

  The following ensures that all data is declared in the space of the initialization routine for either the Initialization Manager or the initialization file for the appropriate API.
- #define **RTEMS_INLINE_ROUTINE** static inline

The following (in conjunction with compiler arguments) are used to choose between the use of static inline functions and macro functions.

- #define **RTEMS_COMPILER_MEMORY_BARRIER**()

  The following macro is a compiler specific way to ensure that memory writes are not reordered around certian points.
- #define **RTEMS_COMPILER_NO_RETURN_ATTRIBUTE**

  The following macro is a compiler specific way to indicate that the method will NOT return to the caller.
- #define **RTEMS_COMPILER_PURE_ATTRIBUTE**

  The following defines a compiler specific attribute which informs the compiler that the method has no effect except the return value and that the return value depends only on parameters and/or global variables.
- #define **RTEMS_COMPILER_DEPRECATED_ATTRIBUTE**

  Instructs the compiler to issue a warning whenever a variable or function with this attribute will be used.
- #define **RTEMS_COMPILER_UNUSED_ATTRIBUTE**

  Instructs the compiler that a specific variable is deliberately unused.
- #define **RTEMS_COMPILER_PACKED_ATTRIBUTE**

  Instructs the compiler that a specific structure or union members will be placed so that the least memory is used.
- #define **RTEMS_STATIC_ASSERT**(cond, msg) typedef int rtems_static_assert_ ## msg [(cond) ? 1 : -1]
- #define **RTEMS_ARRAY_SIZE**(array) (sizeof(array) / sizeof((array)[0]))
- #define **RTEMS_ZERO_LENGTH_ARRAY** 0
- #define **RTEMS_CONTAINER_OF**(_m, _type, _member_name) ((_type *) ((uintptr_t) (_m) - offsetof(_type, _member_name)))

  Returns a pointer to the container of a specified member pointer.
- #define **RTEMS_TYPEOF_REFX**(_ptr_level, _ptr_type) typeof(_ptr_level(union { int z; typeof(_ptr_type) x; }){0}.x)
- #define **RTEMS_DECONST**(_type, _var) ((_type)(uintptr_t)(const void *) (_var))

  Removes the const qualifier from a type of a variable.
- #define **RTEMS_DEVOLATILE**(_type, _var) ((_type)(uintptr_t)(volatile void *) (_var))

  Removes the volatile qualifier from a type of a variable.
- #define **RTEMS_DEQUALIFY**(_type, _var) ((_type)(uintptr_t)(const volatile void *) (_var))

  Removes the all qualifiers from a type of a variable.

Typedefs

- typedef void * **proc_ptr**

  XXX: Eventually proc_ptr needs to disappear!!!

6.11.1 Detailed Description

Basic Definitions.

Index