

Re: Real Microkernel Need C coders

Source: http://coding.derkeiler.com/Archive/C_CPP/comp.lang.c/2004-12/0741.html

From: Adam (*agdm1_at_netzero.net*)

Date: 12/07/04

Date: 6 Dec 2004 18:13:36 -0800

```
/* Go to hell Jacob whatever the hell your name is*/  
/* Yeah some asshole on the net made me code this*/
```

```
#ifndef VGA_H  
#define VGA_H
```

```
#include <CPU.h>  
#include <Time.h>
```

```
#ifdef __cplusplus  
extern "C"  
{  
#endif
```

```
#ifdef ISIS_OS  
#define caddr_t char *  
#endif
```

```
#define TEXT 0 /* Compatible with VGALib v1.2 */  
#define G320x200x16 1  
#define G640x200x16 2  
#define G640x350x16 3  
#define G640x480x16 4  
#define G320x200x256 5  
#define G320x240x256 6  
#define G320x400x256 7  
#define G360x480x256 8  
#define G640x480x2 9
```

```
#define G640x480x256 10  
#define G800x600x256 11  
#define G1024x768x256 12
```

```
#define G1280x1024x256 13 /* Additional modes. */
```

```
#define G320x200x32K 14  
#define G320x200x64K 15
```

```
#define G320x200x16M 16
#define G640x480x32K 17
#define G640x480x64K 18
#define G640x480x16M 19
#define G800x600x32K 20
#define G800x600x64K 21
#define G800x600x16M 22
#define G1024x768x32K 23
#define G1024x768x64K 24
#define G1024x768x16M 25
#define G1280x1024x32K 26
#define G1280x1024x64K 27
#define G1280x1024x16M 28

#define G800x600x16 29
#define G1024x768x16 30
#define G1280x1024x16 31

#define G720x348x2 32 /* Hercules emulation mode */

#define G320x200x16M32 33 /* 32-bit per pixel modes. */
#define G640x480x16M32 34
#define G800x600x16M32 35
#define G1024x768x16M32 36
#define G1280x1024x16M32 37

/* additional resolutions */
#define G1152x864x16 38
#define G1152x864x256 39
#define G1152x864x32K 40
#define G1152x864x64K 41
#define G1152x864x16M 42
#define G1152x864x16M32 43

#define G1600x1200x16 44
#define G1600x1200x256 45
#define G1600x1200x32K 46
#define G1600x1200x64K 47
#define G1600x1200x16M 48
#define G1600x1200x16M32 49

#define __GLASTMODE G1600x1200x16M32
#define GLASTMODE vga_lastmodenumber()

extern int vga_setmode(int mode);
extern int vga_hasmode(int mode);
extern int vga_setflipchar(int c);

extern int vga_clear(void);
extern int vga_flip(void);
```

```
extern int vga_getxdim(void);
extern int vga_getydim(void);
extern int vga_getcolors(void);

extern int vga_setpalette(int index, int red, int green, int blue);
extern int vga_getpalette(int index, int *red, int *green, int
*blue);
extern int vga_setpalvec(int start, int num, int *pal);
extern int vga_getpalvec(int start, int num, int *pal);

extern int vga_screenoff(void);
extern int vga_screenon(void);

extern int vga_setcolor(int color);
extern int vga_drawpixel(int x, int y);
extern int vga_drawline(int x1, int y1, int x2, int y2);
extern int vga_drawscanline(int line, unsigned char *colors);
extern int vga_drawscansegment(unsigned char *colors, int x, int y,
int length);
extern int vga_getpixel(int x, int y); /* Added. */
extern int vga_getscansegment(unsigned char *colors, int x, int y,
int length);

extern int vga_getch(void);

extern int vga_dumpregs(void);

/* Extensions to VGALib v1.2: */

/* blit flags */
#define HAVE_BITBLIT 1
#define HAVE_FILLBLIT 2
#define HAVE_IMAGEBLIT 4
#define HAVE_HLINELISTBLIT 8
#define HAVE_BLITWAIT 16

/* other flags */
#define HAVE_RWPAGE 1 /* vga_setreadpage() / vga_setwritepage()
available */
#define IS_INTERLACED 2 /* mode is interlaced */
#define IS_MODEX 4 /* ModeX style 256 colors */
#define IS_DYNAMICMODE 8 /* Dynamic defined mode */
#define CAPABLE_LINEAR 16 /* Can go to linear addressing mode. */
#define IS_LINEAR 32 /* Linear addressing enabled. */
#define EXT_INFO_AVAILABLE 64 /* Returned modeinfo contains valid
extended fields */
#define RGB_MISORDERED 128 /* Mach32 32bpp uses 0BGR instead of BGR0.
*/

/* As of this version 1.25 also used to signal if real RGB
(red first in memory) is used instead of BGR (Mach32 DAC 4) */
#define HAVE_EXT_SET 256 /* vga_ext_set() available */
```

```

typedef struct {
int width;
int height;
int bytesperpixel;
int colors;
int linewidth; /* scanline width in bytes */
int maxlogicalwidth; /* maximum logical scanline width */
int startaddressrange; /* changeable bits set */
int maxpixels; /* video memory / bytesperpixel */
int haveblit; /* mask of blit functions available */
int flags; /* other flags */

/* Extended fields: */

int chiptype; /* Chiptype detected */
int memory; /* videomemory in KB */
int linewidth_unit; /* Use only a multiple of this as parameter for
set_logicalwidth and
set_displaystart */
char *linear_aperture; /* points to mmap secondary mem aperture of
card (NULL if unavailable) */
int aperture_size; /* size of aperture in KB if size>=videomemory. 0
if unavail */
void (*set_aperture_page) (int page);
/* if aperture_size<videomemory select a memory page */
void *extensions; /* points to copy of eeprom for mach32 */
/* depends from actual driver/chiptype.. etc. */
} vga_modeinfo;

extern vga_modeinfo *vga_getmodeinfo(int mode);
extern int vga_getdefaultmode(void);
extern int vga_getcurrentmode(void);
extern int vga_getcurrentchipset(void);
extern char *vga_getmodename(int mode);
extern int vga_getmodenumber(char *name);
extern int vga_lastmodenumber(void);

extern unsigned char *graph_mem;
extern unsigned char *vga_getgraphmem(void);

extern void vga_setpage(int p);
extern void vga_setreadpage(int p);
extern void vga_setwritepage(int p);
extern void vga_setlogicalwidth(int w);
extern void vga_setdisplaystart(int a);
extern void vga_waitretrace(void);
extern int vga_claimvideomemory(int n);
extern void vga_disabledriverreport(void);
extern int vga_setmodeX(void);
extern int vga_init(void); /* Used to return void in svgalib <=
1.12. */

```

```
extern int vga_getmousetype(void);
extern int vga_getmonitortype(void);
extern void vga_setmousesupport(int s);
extern void vga_lockvc(void);
extern void vga_unlockvc(void);
extern int vga_getkey(void);
extern int vga_oktowitz(void);
extern void vga_copypoplanar256(unsigned char *virtualp, int pitch,
int voffset, int vpitch, int w, int h);
extern void vga_copypoplanar16(unsigned char *virtualp, int pitch,
int voffset, int vpitch, int w, int h);
extern void vga_copypoplanar(unsigned char *virtualp, int pitch,
int voffset, int vpitch, int w, int h, int plane);
extern int vga_setlinearaddressing(void);
extern void vga_safety_fork(void (*shutdown_routine) (void));
```

```
#ifdef EGA /* Kernel headers may define this. */
#undef EGA
#endif
```

```
#define UNDEFINED 0
#define VGA 1
#define ET4000 2
#define CIRRUS 3
#define TVGA8900 4
#define OAK 5
#define EGA 6
#define S3 7
#define ET3000 8
#define MACH32 9
#define GVGA6400 10
#define ARK 11
#define ATI 12
#define ALI 13
#define MACH64 14
#define CHIPS 15
#define APM 16
#define NV3 17
```

```
/* Hor. sync: */
#define MON640_60 0 /* 31.5 KHz (standard VGA) */
#define MON800_56 1 /* 35.1 KHz (old SVGA) */
#define MON1024_43I 2 /* 35.5 KHz (low-end SVGA, 8514) */
#define MON800_60 3 /* 37.9 KHz (SVGA) */
#define MON1024_60 4 /* 48.3 KHz (SVGA non-interlaced) */
#define MON1024_70 5 /* 56.0 KHz (SVGA high frequency) */
#define MON1024_72 6
```

```
extern void vga_setchipset(int c);
extern void vga_setchipsetandfeatures(int c, int par1, int par2);
extern void vga_gettextfont(void *font);
```

```

extern void vga_puttextfont(void *font);
extern void vga_settextmoderegs(void *regs);
extern void vga_gettextmoderegs(void *regs);

extern int vga_white(void);
extern int vga_setegacolor(int c);
extern int vga_setrgbcolor(int r, int g, int b);

extern void vga_bitblt(int srcaddr, int destaddr, int w, int h, int
pitch);
extern void vga_imageblt(void *srcaddr, int destaddr, int w, int h,
int pitch);
extern void vga_fillblt(int destaddr, int w, int h, int pitch, int
c);
extern void vga_hlinelistblt(int ymin, int n, int *xmin, int *xmax,
int pitch, int c);
extern void vga_blitwait(void);
extern int vga_ext_set(unsigned what,...);
extern int vga_accel(unsigned operation,...);

/* Valid values for what in vga_ext_set: */
#define VGA_EXT_AVAILABLE 0 /* supported flags */
#define VGA_EXT_SET 1 /* set flag(s) */
#define VGA_EXT_CLEAR 2 /* clear flag(s) */
#define VGA_EXT_RESET 3 /* set/clear flag(s) */
#define VGA_EXT_PAGE_OFFSET 4 /* set an offset for all subsequent
vga_set*page() calls */
/* Like: vga_ext_set(VGA_EXT_PAGE_OFFSET, 42); */
/* returns the previous offset value. */
#define VGA_EXT_FONT_SIZE 5 /* the (maximal) size of the font buffer */

/* Valid params for VGA_EXT_AVAILABLE: */
#define VGA_AVAIL_SET 0 /* vga_ext_set sub funcs */
#define VGA_AVAIL_ACCEL 1 /* vga_accel sub funcs */
#define VGA_AVAIL_FLAGS 2 /* known flags for VGA_EXT_SET */
#define VGA_AVAIL_ROP 3 /* vga_accel ROP sub funcs */
#define VGA_AVAIL_TRANSPARENCY 4 /* vga_accel TRANSPARENCY sub funcs */
#define VGA_AVAIL_ROPMODES 5 /* vga_accel ROP modes supported funcs */
#define VGA_AVAIL_TRANSMODES 6 /* vga_accel TRANSPARENCY modes
supported */

/* Known flags to vga_ext_set() */
#define VGA_CLUT8 1 /* 8 bit DAC entries */

/* Acceleration interface. */

/* Accel operations. */
#define ACCEL_FILLBOX 1 /* Simple solid fill. */
#define ACCEL_SCREENCOPY 2 /* Simple screen-to-screen BLT. */
#define ACCEL_PUTIMAGE 3 /* Straight image transfer. */
#define ACCEL_DRAWLINE 4 /* General line draw. */

```

```

#define ACCEL_SETFGCOLOR 5 /* Set foreground color. */
#define ACCEL_SETBGCOLOR 6 /* Set background color. */
#define ACCEL_SETTRANSPARENCY 7 /* Set transparency mode. */
#define ACCEL_SETRASTEROP 8 /* Set raster-operation. */
#define ACCEL_PUTBITMAP 9 /* Color-expand bitmap. */
#define ACCEL_SCREENCOPYBITMAP 10 /* Color-expand from screen. */
#define ACCEL_DRAWHLINELIST 11 /* Draw horizontal spans. */
#define ACCEL_SETMODE 12 /* Set blit strategy. */
#define ACCEL_SYNC 13 /* Wait for blits to finish. */
#define ACCEL_SETOFFSET 14 /* Set screen offset */
#define ACCEL_SCREENCOPYMONO 15 /* Monochrome screen-to-screen BLT. */
#define ACCEL_POLYLINE 16 /* Draw multiple lines. */
#define ACCEL_POLYHLINE 17 /* Draw multiple horizontal spans. */
#define ACCEL_POLYFILLMODE 18 /* Set polygon mode. */

/* Corresponding bitmask. */
#define ACCELFLAG_FILLBOX 0x1 /* Simple solid fill. */
#define ACCELFLAG_SCREENCOPY 0x2 /* Simple screen-to-screen BLT. */
#define ACCELFLAG_PUTIMAGE 0x4 /* Straight image transfer. */
#define ACCELFLAG_DRAWLINE 0x8 /* General line draw. */
#define ACCELFLAG_SETFGCOLOR 0x10 /* Set foreground color. */
#define ACCELFLAG_SETBGCOLOR 0x20 /* Set background color. */
#define ACCELFLAG_SETTRANSPARENCY 0x40 /* Set transparency mode. */
#define ACCELFLAG_SETRASTEROP 0x80 /* Set raster-operation. */
#define ACCELFLAG_PUTBITMAP 0x100 /* Color-expand bitmap. */
#define ACCELFLAG_SCREENCOPYBITMAP 0x200 /* Color-exand from screen. */
#define ACCELFLAG_DRAWHLINELIST 0x400 /* Draw horizontal spans. */
#define ACCELFLAG_SETMODE 0x800 /* Set blit strategy. */
#define ACCELFLAG_SYNC 0x1000 /* Wait for blits to finish. */
#define ACCELFLAG_SETOFFSET 0x2000 /* Set screen offset */
#define ACCELFLAG_SCREENCOPYMONO 0x4000 /* Monochrome screen-to-screen
BLT. */
#define ACCELFLAG_POLYLINE 0x8000 /* Draw multiple lines. */
#define ACCELFLAG_POLYHLINE 0x10000 /* Draw multiple horizontal spans.
*/
#define ACCELFLAG_POLYFILLMODE 0x20000 /* Set polygon mode. */

/* Mode for SetTransparency. */
#define DISABLE_TRANSPARENCY_COLOR 0
#define ENABLE_TRANSPARENCY_COLOR 1
#define DISABLE_BITMAP_TRANSPARENCY 2
#define ENABLE_BITMAP_TRANSPARENCY 3

/* Flags for SetMode (accelerator interface). */
#define BLITS_SYNC 0
#define BLITS_IN_BACKGROUND 0x1

/* Raster ops. */
#define ROP_COPY 0 /* Straight copy. */
#define ROP_OR 1 /* Source OR destination. */
#define ROP_AND 2 /* Source AND destination. */

```

comp.lang.c: Re: Real Microkernel Need C coders

```
#define ROP_XOR 3 /* Source XOR destination. */
#define ROP_INVERT 4 /* Invert destination. */

/* For the poly funcs */
#define ACCEL_START 1
#define ACCEL_END 2

/*
 * wait for keypress, mousemove, I/O, timeout. cf. select (3) for
 * details on
 * all parameters except which.
 * NULL is a valid argument for any of the ptrs.
 */

extern int vga_waitevent(int which, fd_set * in, fd_set * out,
fd_set * except,
struct timeval *timeout);

/*
 * valid values for what ( | is valid to combine them )
 */
#define VGA_MOUSEEVENT 1
#define VGA_KEYEVENT 2

/*
 * return value >= has bits set for mouse/keyboard events detected.
 * mouse and raw keyboard events are already handled and their bits
 * removed
 * from *in when vga_waitevent returns.
 * VGA_KEYEVENT relates to vga_getch NOT vga_getkey.
 * return values < 0 signal errors. In this case check errno.
 */

/* Background running */
extern void vga_runinbackground(int stat, ...);
#define VGA_GOTOBACK -1
#define VGA_COMEFROMBACK -2
extern int vga_runinbackground_version(void);
#ifdef __cplusplus
}

#endif
#endif /* VGA_H */
```