Working with easyGUI & Linux

easyGUI can be used with almost any compiler and RTOS. Also systems based on embedded Linux can utilize easyGUI. There are three categories of Linux drivers:

1. A driver using the Linux virtual frame buffer device, working from user space.
2. A driver which writes directly to ports, working from user space.
3. A driver which works as a module from kernel space.

Currently easyGUI has several drivers of category 1, e.g. The AT32AP7000 driver and the TS7390 driver. Basically they are more or less identical. The easyGUI drivers for the Linux operating system uses the Linux frame buffer device, which serves as a “virtual” frame buffer for accessing the underlying physical frame buffer. The drivers works from user space, and uses method of mapping the entire frame buffer to a given memory range, because the physical frame buffer in hardware memory can’t be access directly. For gaining access to physical frame buffer memory the driver uses the `mmap()` function, which returns a pointer to the start of the mapped frame buffer. For setting color depth and resolution of the display the data structure `fb_var_screeninfo` is used (declared in fb.h). This structure is also accessible from user space.

GuiDisplay_Init function

This initialization routine first fetches a file descriptor (a small, non-negative integer for use in subsequent I/O as with read, write, etc.). This file descriptor is gained through the system call

`open()`, which is used for conversion from pathname of devices. Next is the system call `ioctl()` for retrieving and setting frame buffer settings x resolution, y resolution and color depth - this is a special subroutine call that transfers the operating system from user mode to kernel mode, because the easyGUI driver works in user space, and can't directly access kernel code. It is the user's responsibility to set only values which are supported by the underlying frame buffer driver module. The last step is to gain a pointer to mapped frame buffer memory through calling function `mmap()`, which can be used to access the frame buffer as a normal standard pointer to array. The initialization routine also clears the frame buffer.

GuiDisplay_Refresh function

This function copies data changed since last call from the internal easyGUI display buffer to the Linux frame buffer. The exact data pointer is calculated from the start address pointer (from the initialization routine) and the chosen resolution.