easyGUI and code size

We get questions on easyGUI and code sizes all the time. It's an important issue when designing embedded interfaces, and when deciding on what system to use for a project.

"Do I have enough RAM or ROM for my project?" is probably the most common question we get. Unfortunately this is also one of the hardest to answer. To answer these kinds of questions with any degree of certainty, we would need a ton of detailed information on both your product and your hardware setup. Information that we do not have access to.

We have however tried to make a guide that approximately shows the required easyGUI's memory sizes. Please bear in mind that a lot of factors can and will affect these numbers.

Old style alphanumeric mode (character based) displays for embedded systems require a minimum of processor resources and memory, and are very fast. The disadvantages are the limited user interface design possibilities, and the lack of true graphics. Virtually all modern displays are either custom displays made individually per project, or true graphics types, with pixel addressing capabilities. The latter allow much more complex and artistic user interfaces, making better products, with higher market penetration, but also taxes the system more. The processor must thus have sufficient power to run the display in the selected graphical mode, with the highest resolution and deepest color depths requiring the most power.

But how much power is needed? Very difficult to answer, as it depends on a lot of factors, among them user interface complexity, how dynamic the user interface will be (scrolling, animation, ...), how much else the processor is supposed to handle, the effectiveness of the compiler and kernel, etc. Be aware that a graphical user interface like that possible with easyGUI needs at least 10 times the processor power of corresponding old style displays. That goes with the nature of the graphical approach. The benefit is of course the much improved user interface.

easyGUI can theoretically handle any display size, but there is a natural limit both at the bottom and the top of the scale. Very small displays are unsuitable for a graphical user interface, because the microcontroller in use is normally rather slow and with limited resources. At the other end of the scale there is a tendency to use advanced operating systems like e.g. Windows CE, which contain their own display control libraries, but also require much more memory, typically by at least a factor of 10-50.

Company: IBIS Solutions ApS. Torvevangen 24, DK-4550 Asnaes, Denmark. Phone: +45 7022 0495 Fax: +45 7023 0495
VAT No.: DK 27 06 03 07 Mail: sales@ibissolutions.com Web: www.ibissolutions.com
Please observe that our office hours follow the Greenwich Mean Time (GMT) + 1 hour.

easyGUI normally uses an internal RAM display buffer for building the display image, before sending it to the display controller, plus an additional amount for general housekeeping. The amount of RAM and ROM relies heavily on three factors: Display size, color depth, and complexity of the user interface.

RAM

RAM usage can be calculated as 6KB + (Display width x Display height x Bits per color) / 8. Some typical example values are:

- 128x64 pixels monochrome: 6KB + (128x64x1)/8: ~7KB RAM
- 128x64 pixels 16 colors: 4KB + (128x64x4)/8: ~8KB RAM
- 240x128 pixels monochrome: 4KB + (240x128x1)/8: ~8KB RAM
- 240x128 pixels 256 colors: 4KB + (240x128x8)/8: ~36KB RAM
- 320x240 pixels (quarter VGA) monochrome: 4KB + (320x240x1)/8: ~14KB RAM
- 320x240 pixels (quarter VGA) 64K colors (16 bits): 4KB + (320x240x16)/8: ~158KB RAM

The internal RAM display buffer can be dispensed with, provided that the display controller memory is mapped into ordinary memory space, allowing the easyGUI library to write effectively directly to the display controller RAM. The RAM usage then drops to the ~6KB level.

ROM

ROM size is more difficult to estimate than RAM size, as it is very dependent on application complexity, but also the display size matters to some degree, as bigger displays tends to have more complicated user interfaces, and use bigger fonts. The easyGUI graphical library is actually a little smaller for displays using many colors, as it is simpler to write data to these display types. The system will be needing more power, as bigger color depths necessitates more code execution, despite the code being simpler.

ROM/flash usage can be calculated as 21KB + font data + screen data. The following examples are only rough guidelines:

- Low complexity GUI (50 structures), 2 full text fonts, 1 partial big font, 2 icon fonts: 30KB~50KB ROM.
- Medium complexity GUI (250 structures), 2 full text fonts, 2 partial big fonts, 4 icon fonts: 60KB~100KB ROM.
- High complexity GUI (400 structures), 4 full text fonts, 2 partial big fonts, 6 icon fonts: 100~150KB ROM.

Adding additional languages will increase the ROM requirements.

The RAM and ROM requirements can be somewhat lessened by excluding features of the easyGUI library, like e.g. cursors, scrolling, clipping, bitmaps, etc.

The easyGUI user interface data normally placed in ROM/flash can also be moved to external memory (e.g. SPI accessed Flash RAM), thus freeing up ROM. The back side is that this can result in a significant speed penalty, as data now have to be fetched using some kind of connection method to external memory, instead of simply being read from the ordinary memory bus. If the user interface can tolerate this, either because speed requirements are low, and/or because of plenty of processor power, external memory data placement might be a viable option.

As can be seen from the above easyGUI places comparably low requirements on RAM and ROM, and there are several options for reducing or moving the data requirements.