

#### About this document

This document is being written to give myself a one-stop reference that contains everything I need to know about the Gameboy Advance. The guide will be split into chapters and released slowly as and when I have completed a section.

If there is anything in the document that you feel is incorrect, written by someone else and not referenced, or if you have any comments then please contact me: [gbajunkie@hotmail.com](mailto:gbajunkie@hotmail.com) I would be pleased to hear from you. Although this is a personal reference document I am publishing it so it can help others, feedback can only help this process.

This document is Copyright © gbajunkie.co.uk 2002  
Date written: 17<sup>th</sup> January 2002

## Chapter 2: GBA Graphics

### What is in this chapter

- The 6 screen modes
- Tile modes
- Bitmap modes
- GBA.H
- REG\_DISPCNT register
- My DISPCNT.H header file
- Setting the mode you want

### The 6 Screen Modes

There are 6 different screen modes available to you on the GBA. What follows is a detailed description of each one. This should hopefully give you an insight into which one to use depending on what you are trying to achieve.

The 6 modes (0,1,2,3,4,5) can be split into two specific sets. Modes 0, 1 and 2 are **tile modes** where as the other three modes 3, 4, and 5 are **bitmap modes**. I will deal with modes 0, 1 and 2 first.

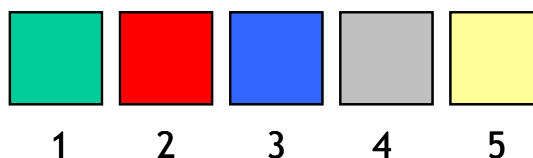
### What are Tile Modes?

When using a tile mode the screen is broken down into 8x8 squares of pixels called tiles. These tiles each contain an image (or part of a bigger image). All you have to do is supply an array of numbers (a map) that represents how these tiles will be drawn on screen. Each number in the map relates to one of your tiles, where ever this number appears in the map the corresponding tile will be drawn. The GBA will draw the tiles in the order you specified to produce the screen layout.

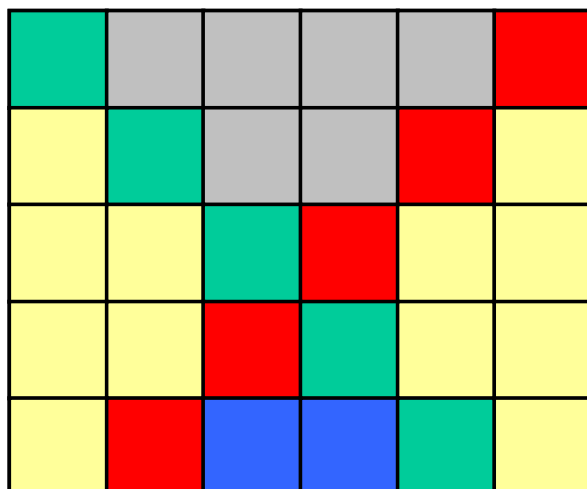
### Map Array

1	4	4	4	4	2
5	1	4	4	2	5
5	5	1	2	5	5
5	5	2	1	5	5
5	2	3	3	1	5

### Available Tiles



## GBA Screen



There are a lot of advantages to using tiles, for example, it can save a lot of memory because you can use tiles multiple times and rotate them for different situations. This allows you to produce very large maps using very few tiles and therefore very little space in memory. The important thing to remember about tile modes is that they are used to produce **backgrounds only**, you cannot produce sprites with them. The sprites can be drawn while in a tile mode but they cannot be made up from the tiles used for the backgrounds.

There can be a maximum of 1024 tiles stored in video memory allowing you to produce screens from 256x256 pixels up to 1024x1024 pixels. You can also have as many as 4 different backgrounds displayed at any one time using different maps and tile data. This can produce some very nice effects (a good example is Nokturn's scrolling clouds over a landscape). Each background can be manipulated independently of the others giving you full control over each. Rotations and scaling can also be achieved depending on which mode you are using.

### Why are there 3 different tile modes you may ask?

Basically each of the three allow you to do different things with the backgrounds as shown below. To begin with Mode 0 is probably the best one to use as it allows you to play with all 4 backgrounds and does not support rotation and scaling which most people wont want to deal with straight away. Below is a list of what each mode allows, notice that rotation and scaling limits the available backgrounds.

#### Mode 0

Backgrounds available: All 4 backgrounds  
Rotation and Scaling: No

#### Mode 1

Backgrounds Available: 0, 1, and 2  
Rotation and Scaling: Background 2

#### Mode 2

Backgrounds Available: 2 and 3  
Rotation and Scaling: Backgrounds 2 and 3

### What are Bitmap Modes?

Bitmap modes as the name suggests allows you to access the screen as if it was a single bitmap image. The three available modes differ slightly and each is better depending on what you are trying to achieve. What follows is a description of each mode:

#### Mode 3

Backgrounds Available: Background 2 which is 240x160 pixels  
No. of frame buffers used: 1  
Colour: 15-bit true colour

Mode 3 consists of a linear array of 16-bit pixels. What this means is that the screen is accessed through a 1D array structure. If you put a value into any position in the array it will affect one pixel on the screen. All you have to do is setup the array to point to the start of video memory and then access each element of the array to access individual pixels.

This mode is excellent for displaying single screen static images but that is about it. It is too slow to do any animation, this is due to the single frame buffer meaning that to achieve animation the GBA would have to redraw the scene every frame. This is why double buffering is so desirable (this is where modes 4 & 5 come into play).

#### Mode 4

Background Available: Background 2 which is 240x160 pixels  
No. of frame buffers used: 2  
Colour: 8-bit indexed palette

This mode is very similar to Mode 3 except it uses a 16-bit palette with 8-bit indexes. Each pixel on the screen is an 8-bit index into this 16-bit palette. You store 256 16-bit colours that you wish to use for you background in palette memory. Then you access the screen as an array, you provide a number between 0-255 which then colours the active pixel. There are two major advantages to this approach:

1. A pixel can be drawn in half the time
2. Two screens can be stored in memory at any one time due to the time saving pixel drawing (double buffering)

This double buffering technique allows you to draw to the screen not being currently displayed and then swap the screens. This produces very smooth animation.

#### Mode 5

Background Available: Background 2 which is 160x128 pixels  
No. of frame buffers used: 2  
Colour: 15-bit true colour

This mode combines modes 3 and 4. It allows true colour display and double buffering but the drawback is a smaller screen size 160x128 pixels. Not much else to say about this mode.

### GBA.H

Programming the GBA is all about registers. GBA.H that was put together by Eloist contains definitions of all the major registers you will need to use. It is important that you have a copy and include it with every program you create. It will be included in every program you download from my site so don't worry about where to get it from. As you progress in your knowledge of GBA programming so more of the registers will become familiar to you. Taking a look at the file right now it looks very daunting due to the amount of definitions, but they will all make sense eventually.

By far the most important register defined in GBA.H is REG\_DISPCNT which I will discuss next.

## REG\_DISPCNT

REG\_DISPCNT is probably the most important register in GBA programming because it allows you to do the following:

- Set the screen mode
- Enable backgrounds
- Set the layout of sprites in memory
- Gives you control over video hardware

It is a 16-bit register which is split up to control different areas as outlined above. Below is a description of what each bit or set of bits does within the register.

### Bits 0-2

These three bits allow you to set the screen mode from 0 to 5. So you can choose which of the modes I discussed earlier, you want to use.

### Bit 3

This bit is only set if a Gameboy or Gameboy Colour cartridge is to be run from your GBA. Therefore it does not affect coding the GBA in any way so you can forget about it.

### Bit 4

As you should know from reading the screen modes section, modes 4 and 5 have two frame buffers. You need some way of switching between these two buffers when your game is running. This is what bit 4 of the register is used for, by setting and un-setting the bit you can switch the buffers.

### Bit 5

When set this bit allows you to update the OAM (Object Attribute Memory) during a horizontal blank (H\_BLANK). I have not discussed the OAM yet because it is used to deal with sprites. You will find a detailed description either later on in this chapter or in the next chapter. For the moment you do not need to worry about it.

### Bit 6

Allows you to choose the mapping mode for sprites. This will be discussed at another time with reference to sprites.

### Bit 7

When set the screen is forced to go white (a blank). There has been no real use highlighted for this yet. You may find one but until you do don't worry about it.

### Bits 8-11

As you know from reading earlier in this document there are 4 available backgrounds. These four bits are used to enable them based on the mode you are currently in e.g. Mode 0 requires all 4 bits to be set.

### Bit 12

Used to enable object sprites

### Bit 13

Allows you to display separate windows. Not too sure about this as I have not found much information about them. At a guess I would assume it allows you to split the screen into 2 windows and display different images within them. But don't quote me on that.

## Setting the mode you want to use

Although I haven't covered actually coding anything yet (I will soon) I think now is the right time to explain how you set the mode you want to use.

The mode will have to be set every time you create a game/program so it would be good if we could have a reusable function that just allows you to specify what mode you want this time.

As explained above REG\_DISPCNT is just a single 16-bit register with the different bits being set for different modes etc. Each of the bits needs to be defined within your code and then combined somehow to choose a mode. To define each bit in the register I use a header file I have called **DISPCNT.H** that I have included below:

#### DISPCNT.H

```
//define the bits that control the screen mode 0-5
#define MODE_0                0x0
#define MODE_1                0x1
#define MODE_2                0x2
#define MODE_3                0x3
#define MODE_4                0x4
#define MODE_5                0x5

//define the buffer which is used to set the active buffer
//when using double buffering
#define backbuffer            0x10

//This bit, when set allows OAM(Object Attribute Memory) to
//be updated during a horizontal blank
#define H_BLANK_OAM          0x20

//use these two defines to choose which mapping mode is used
//for sprite graphics 2D or 1D
#define OBJ_MAP_2D            0x0
#define OBJ_MAP_1D            0x40

//Causes the screen to go white by using a forced blank
#define FORCE_BLANK            0x80

//define the flags for enabling backgrounds and objects(sprites)
#define BG0_ENABLE            0x100
#define BG1_ENABLE            0x200
#define BG2_ENABLE            0x400
#define BG3_ENABLE            0x800
#define OBJ_ENABLE            0x1000

//allows window displays (don't worry about these)
#define WIN1_ENABLE            0x2000
#define WIN2_ENABLE            0x4000
#define WINOBJ_ENABLE         0x8000
```

All the bits are given meaningful names and a hexadecimal value in this header file. The value corresponds to where the bit is within the register. So all we need to do now is set the relevant bits in this register to set the mode. The easiest way to do this is with a **macro**. All you have to do is logically AND the different bits together to create the mode you desire, for example, if you create a macro called SetMode() which is defined as below:

```
#define SetMode(mode) REG_DISPCNT = (mode)
```

where REG\_DISPCNT is the register as defined in GBA.H. That is the macro defined, now how to call it. This is easy now, if you wanted to set the mode to Mode 3 and you look at my handy description of modes above, you will notice that mode 3 only allows you to use Background 2. So to set the mode to Mode 3 you would call the macro with the following:

```
SetMode( MODE_3 | BG2_ENABLE);
```

That's it! REG\_DISPCNT will now have the correct bits set for Mode 3. As another example, to set the mode to Mode 4 with objects (sprites) visible, you would call it as follows:

```
SetMode( MODE_4 | BG2_ENABLE | OBJ_ENABLE);
```

You will see it in action when I do some more coding. The program I included with chapter one does use it however.

**What you should know from reading this chapter:**

- What the 6 different graphics modes are and what to use them for
- The difference between tile and bitmap modes
- What the REG\_DISPCNT register is and what it is used for
- How to set the modes you wish to use

**Bibliography**

Title: Gameboy Advance Quick Reference Guide

Author: Djiin

Source: <http://www.gbadev.org>

Title: The Pern Project Tutorials

Author: Dovoto

Source: <http://216.167.73.47/~dovoto/>

Title: Gameboy Advance Resource Management

Author: Rafael Baptista

Source: [http://www.gamasutra.com/features/20010509/baptista\\_01.htm](http://www.gamasutra.com/features/20010509/baptista_01.htm)