Part IV

The Mother Of All Appendices

Welcome to Part IV of *Programming The Game Boy Advance*. Part IV includes five appendices that provide reference information for your use, including an ASCII chart, a list of helpful books and Web sites, and an overview of the included CD-ROM, answers to the chapter quizzes, and even a Game Boy Advance hardware reference.

Appendix A

ASCII Chart

This is a standard ASCII chart of character codes 0 to 255. To use an ASCII code, simply hold down the Alt key and type the appropriate value to insert the character.

null	000	←	027	6	054	Q	081
☺	001	∟	028	7	055	R	082
☻	002	↔	029	8	056	S	083
♥	003	▲	030	9	057	T	084
♦	004	▼	031	:	058	U	085
♣	005	space	032	;	059	V	086
♠	006	!	033	<	060	W	087
•	007	"	034	=	061	X	088
◘	008	#	035	>	062	Y	089
○	009	$	036	?	063	Z	090
◙	010	%	037	@	064	[	091
♂	011	&	038	A	065	\	092
♀	012	'	039	B	066	]	093
♪	013	(	040	C	067	^	094
♫	014	)	041	D	068	_	095
☼	015	*	042	E	069	`	096
►	016	+	043	F	070	a	097
◄	017	,	044	G	071	b	098
↕	018	-	045	H	072	c	099
‼	019	.	046	I	073	d	100
¶	020	/	047	J	074	e	101
§	021	0	048	K	075	f	102
▬	022	1	049	L	076	g	103
↨	023	2	050	M	077	h	104
↑	024	3	051	N	078	i	105
↓	025	4	052	O	079	j	106
→	026	5	053	P	080	k	107

l	108	ë	137	ª	166	├	195
m	109	è	138	º	167	─	196
n	110	ï	139	¿	168	┼	197
o	111	î	140	⌐	169	╞	198
p	112	ì	141	¬	170	╟	199
q	113	Ä	142	½	171	╚	200
r	114	Å	143	¼	172	╔	201
s	115	É	144	¡	173	╩	202
t	116	æ	145	«	174	╦	203
u	117	Æ	146	»	175	╠	204
v	118	ô	147	░	176	═	205
w	119	ö	148	▒	177	╬	206
x	120	ò	149	▓	178	╧	207
y	121	û	150	│	179	╨	208
z	122	ù	151	┤	180	╤	209
{	123	ÿ	152	╡	181	╥	210
|	124	Ö	153	╢	182	╙	211
}	125	Ü	154	╖	183	╘	212
~	126	¢	155	╕	184	╒	213
⌂	127	£	156	╣	185	╓	214
Ç	128	¥	157	║	186	╫	215
ü	129	Pts	158	╗	187	╪	216
é	130	ƒ	159	╝	188	┘	217
â	131	á	160	╜	189	┌	218
ä	132	í	161	╛	190	█	219
à	133	ó	162	┐	191	▄	220
å	134	ú	163	└	192	▌	221
ç	135	ñ	164	┴	193	▐	222
ê	136	Ñ	165	┬	194	▀	223

α	224
ß	225
Γ	226
π	227
Σ	228
σ	229
µ	230
τ	231
Φ	232
Θ	233
Ω	234
δ	235
∞	236
φ	237
ε	238
∩	239
≡	240
±	241
≥	242
≤	243
⌠	244
⌡	245
÷	246
≈	247
°	248
·	249
·	250
√	251
ⁿ	252
²	253
■	254
blank	255

Appendix B

Recommended Books And Web Sites

The following books and Web sites are invaluable learning and reference tools that cover programming, video game history, and more.

Recommended Books

Here is a list of good programming books, including some of my favorites and some that I have written myself.You may find a few to be helpful when learning Game Boy programming. Along with beginner books, this list presents advanced books, as well as some references for the ARM chip.

ARM Architecture Reference Manual (2nd Edition) (2000)

Dave Jagger. Addison-Wesley Publishing Company. ISBN 0201737191.

This book is an excellent resource for the ARM processor in all of its variations.

ARM System-On-Chip Architecture (2nd Edition) (2000)

Stephen B. Furber. Addison-Wesley Publishing Company. ISBN 0201675196.

This book describes how to design a CPU system-on-chip around a microprocessor core, using the ARM architecture as a case study.

***Beginner's Guide to DarkBASIC Game Programming* (2002)**

Jonathan S. Harbour and Joshua R. Smith. Premier Press. ISBN 1-59200-009-6.

This book provides a good introduction to programming Direct3D, the 3D graphics component of DirectX, using the C language.

***C Programming for the Absolute Beginner* (2002)**

Michael A. Vine. Premier Press. ISBN 1-931841-52-7.

This book teaches C programming using the free GCC compiler as its development platform, which is the same compiler used to write Game Boy programs! As such, I highly recommend this starter book if you are just learning the C language. It sticks to just the basics. You will learn the fundamentals of the C language without any distracting material or commentary, just the fundamentals of what you need to be a successful C programmer.

***C++ Programming for the Absolute Beginner* (2001)**

Dirk Henkemans and Mark Lee. Premier Press. ISBN 1-931841-43-8.

If you are new to programming with C++ and you are looking for a solid introduction, this is the book for you. This book will teach you the skills you need for practical C++ programming applications and how you can put these skills to use in real-world scenarios.

***Game Design: The Art & Business of Creating Games* (2001)**

Bob Bates. Prima Tech. ISBN 0-7615-3165-3.

This very readable and informative book is a great resource for learning how to design games[--]the high-level process of planning the game prior to starting work on the source code or artwork.

***Game Programming All in One* (2002)**

Bruno Miguel Teixeira de Sousa. Premier Press. ISBN 1-931841-23-3.

This book presents everything you need to get started as a game developer using the C language. Divided into increasingly advanced sections, it covers the most important elements of game development. Beginners start with the basics of C programming early in the book. Later chapters move on to Windows programming and the main components of DirectX.

***High Score! The Illustrated History of Electronic Games* (2002)**

Rusel DeMaria and Johnny L. Wilson. McGraw-Hill/Osborne. ISBN 0-07-222428-2.

This gem of a book covers the entire video game industry, including arcade machines, consoles, and computer games. It is jam-packed with wonderful interviews with famous game developers and is chock-full of color photographs, including detailed information about Nintendo and the development of the Game Boy Advance.

***Microsoft C# Programming for the Absolute Beginner* (2002)**

Andy Harris. Premier Press. ISBN 1-931841-16-0.

Using game creation as a teaching tool, this book teaches not only C# but also the fundamental programming concepts you need to grasp to learn any computer language. You will be able to take the skills you learn from this book and apply them to your own situations. *Microsoft C# Programming for the Absolute Beginner* is a unique book aimed at the novice programmer. Developed by computer science instructors, this series is the ideal tool for anyone with little to no programming experience.

***Microsoft Visual Basic .NET Programming for the Absolute Beginner* (2002)**

Jonathan S. Harbour. Premier Press. ISBN 1-59200-002-9.

Whether you are new to programming with Visual Basic .NET or you are upgrading from Visual Basic 6.0 and looking for a solid introduction, this is the book for you. It teaches the basics of Visual Basic .NET by working through simple games that you will learn to create. You will acquire the skills you need for more practical Visual Basic .NET programming applications and learn how to put these skills to use in real-world scenarios.

***Pocket PC Game Programming: Using the Windows CE Game API* (2001)**

Jonathan S. Harbour. Premier Press. ISBN 0-7615-3057-6.

This book will teach you how to program a Pocket PC handheld computer using Visual Basic or Visual C++. It includes coverage of graphics, sound, stylus and button input, and even multiplayer capability. Numerous sample programs and games demonstrate the key topics needed to write complete Pocket PC games.

***Swords & Circuitry: A Designer's Guide to Computer Role-Playing Games* (2001)**

Neal and Jana Hallford. Prima Tech. ISBN 0-7615-3299-4.

This book is a fascinating overview of what it takes to develop a commercial-quality role-playing game, from design to programming to marketing. This is a helpful book if you would like to write a game like *Zelda*.

***Visual Basic Game Programming with DirectX* (2002)**

Jonathan S. Harbour. Premier Press. ISBN 1-931841-25-X.

This book is a comprehensive programmer's tutorial and a reference for everything related to programming games with Visual Basic. After a complete explanation of the Windows API graphics device interface meant to supercharge 2D sprite programming for normal applications, the book delves into DirectX 7.0 and 8.1 and covers every component of DirectX in detail, including Direct3D. Four complete games are included, demonstrating the code developed in the book.

Recommended Web Sites

Following is a list of Web sites that you will find useful when you are learning to program the Game Boy Advance and also as you start writing your own games. For the latest updates to the Web site list, including links to new Web sites dedicated to the Game Boy Advance, please visit my Web site at http://www.jharbour.com and click the Books link to find the official site for this book.

Code Waves

http://www.codewaves.com

The home site of the GBA sound library and other tools.

CowBite Virtual Hardware Specifications

http://www.cs.rit.edu/%7Etjh8300/CowBite/CowBiteSpec.htm

A detailed and invaluable hardware reference for the GBA.

Game Boy Advance Dev'rs

http://www.devrs.com/gba

A useful programming site that focuses on GBA development, including links to GBA tools.

HAM and Hamlib

http://www.ngine.de

The home site for HAM and Hamlib, the GBA development distribution kit included and used in this book.

Microsoft DirectX

http://www.microsoft.com/directx

Microsoft's main DirectX site, where you can download the latest version of DirectX. You will need DirectX in order to run the VisualBoyAdvance emulator.

Visual HAM Home Page

http://visualham.console-dev.de

The home site for the Visual HAM integrated development environment used to write Game Boy programs in this book.

GameDev.net

http://www.gamedev.net

A well-respected online resource for all things related to game development.

Jonathan S. Harbour: Author's Home Page

http://www.jharbour.com

Jonathan's home page, with downloads, links, and resources for this book.

Nintendo Home

http://www.nintendo.com

The primary portal for all Nintendo products.

Nintendo Company History

http://www.nintendo.com/corp/history.html

The source of all historical references used in this book.

Appendix C

Game Boy Advance Hardware Reference

Throughout the book there have been source code listings that made use of standard defines, memory address values, and constants needed to write GBA programs. Here is a complete reference of all those lists in one convenient location. There were some cases where I defined values with slightly different names in the text of the book in order to clarify or explain the purpose of a register or value more easily. The important thing is to know when and how to use these values, rather than being overly specific on naming conventions.

Multiboot

```
#define MULTIBOOT int __gba_multiboot;
```

Bit Values

```
#define BIT00  1
#define BIT01  2
#define BIT02  4
#define BIT03  8
#define BIT04  16
#define BIT05  32
#define BIT06  64
#define BIT07  128
#define BIT08  256
#define BIT09  512
#define BIT10  1024
#define BIT11  2048
#define BIT12  4096
#define BIT13  8192
#define BIT14  16384
#define BIT15  32768
```

Typedefs

```
typedef unsigned char            u8;
typedef unsigned short           u16;
typedef unsigned long            u32;
typedef signed char              s8;
typedef signed short             s16;
typedef signed long              s32;
typedef unsigned char           byte;
typedef unsigned short          hword;
typedef unsigned long           word;
typedef volatile unsigned char  vu8;
typedef volatile unsigned short vu16;
typedef volatile unsigned long  vu32;
typedef volatile signed char    vs8;
typedef volatile signed short   vs16;
typedef volatile signed long    vs32;
```

Buttons

```
volatile u32* BUTTONS = (volatile u32*)0x04000130;
#define BUTTON_A        1
#define BUTTON_B        2
#define BUTTON_SELECT   4
#define BUTTON_START    8
#define BUTTON_RIGHT    16
#define BUTTON_LEFT     32
#define BUTTON_UP       64
#define BUTTON_DOWN     128
#define BUTTON_R        256
#define BUTTON_L        512
```

Sprites

```
#define OAMmem         (u32*)0x7000000
#define OAMdata        (u16*)0x6100000
#define OBJPaletteMem  (u16*)0x5000200

//Attribute0 values
#define ROTATION_FLAG     0x100
#define SIZE_DOUBLE       0x200
#define MODE_NORMAL       0x0
#define MODE_TRANSPARENT  0x400
#define MODE_WINDOWED    0x800
```

```
#define MOSAIC          0x1000
#define COLOR_16        0x0000
#define COLOR_256       0x2000
#define SQUARE          0x0
#define WIDE            0x4000
#define TALL            0x8000

//Attribute1 values
#define ROTDATA(n)       ((n) << 9)
#define HORIZONTAL_FLIP  0x1000
#define VERTICAL_FLIP    0x2000
#define SIZE_8           0x0
#define SIZE_16          0x4000
#define SIZE_32          0x8000
#define SIZE_64          0xC000

//Attribute2 values
#define PRIORITY(n)      ((n) << 10)
#define PALETTE(n)       ((n) << 12)
```

Backgrounds

```
#define REG_BG0CNT          *(u16*)0x4000008
#define REG_BG1CNT          *(u16*)0x400000A
#define REG_BG2CNT          *(u16*)0x400000C
#define REG_BG3CNT          *(u16*)0x400000E
#define REG_BG0HOFS         *(u16*)0x4000010
#define REG_BG0VOFS         *(u16*)0x4000012
#define REG_BG1HOFS         *(u16*)0x4000014
#define REG_BG1VOFS         *(u16*)0x4000016
#define REG_BG2HOFS         *(u16*)0x4000018
#define REG_BG2VOFS         *(u16*)0x400001A
#define REG_BG3HOFS         *(u16*)0x400001C
#define REG_BG3VOFS         *(u16*)0x400001E
#define REG_BG2PA           *(u16*)0x4000020
#define REG_BG2PB           *(u16*)0x4000022
```

```
#define REG_BG2PC             *(u16*)0x4000024
#define REG_BG2PD             *(u16*)0x4000026
#define REG_BG2X              *(u32*)0x4000028
#define REG_BG2X_L            *(u16*)0x4000028
#define REG_BG2X_H            *(u16*)0x400002A
#define REG_BG2Y              *(u32*)0x400002C
#define REG_BG2Y_L            *(u16*)0x400002C
#define REG_BG2Y_H            *(u16*)0x400002E
#define REG_BG3PA             *(u16*)0x4000030
#define REG_BG3PB             *(u16*)0x4000032
#define REG_BG3PC             *(u16*)0x4000034
#define REG_BG3PD             *(u16*)0x4000036
#define REG_BG3X              *(u32*)0x4000038
#define REG_BG3X_L            *(u16*)0x4000038
#define REG_BG3X_H            *(u16*)0x400003A
#define REG_BG3Y              *(u32*)0x400003C
#define REG_BG3Y_L            *(u16*)0x400003C
#define REG_BG3Y_H            *(u16*)0x400003E
#define BG_MOSAIC_ENABLE      0x40
#define BG_COLOR_256          0x80
#define BG_COLOR_16           0x0
#define TEXTBG_SIZE_256x256   0x0
#define TEXTBG_SIZE_256x512   0x8000
#define TEXTBG_SIZE_512x256   0x4000
#define TEXTBG_SIZE_512x512   0xC000
#define ROTBG_SIZE_128x128    0x0
#define ROTBG_SIZE_256x256    0x4000
#define ROTBG_SIZE_512x512    0x8000
#define ROTBG_SIZE_1024x1024  0xC000
#define WRAPAROUND            0x2000
#define CharBaseBlock(n)      (((n)*0x4000)+0x6000000)
#define ScreenBaseBlock(n)    (((n)*0x800)+0x6000000)
```

Video

```
#define SetMode(mode)  REG_DISPCNT = (mode)
#define VideoBuffer    (u16*)0x6000000
#define BGPaletteMem   (u16*)0x5000000
#define REG_DISPCNT    *(u32*)0x4000000
#define REG_DISPCNT_L  *(u16*)0x4000000
#define REG_DISPCNT_H  *(u16*)0x4000002
#define REG_DISPSTAT   *(u16*)0x4000004
#define REG_VCOUNT     *(u16*)0x4000006
#define REG_WIN0H      *(u16*)0x4000040
#define REG_WIN1H      *(u16*)0x4000042
#define REG_WIN0V      *(u16*)0x4000044
#define REG_WIN1V      *(u16*)0x4000046
#define REG_WININ      *(u16*)0x4000048
#define REG_WINOUT     *(u16*)0x400004A
#define BACKBUFFER     0x10
#define H_BLANK_OAM    0x20
#define OBJ_MAP_2D     0x0
#define OBJ_MAP_1D     0x40
#define FORCE_BLANK    0x80
#define BG0_ENABLE     0x100
#define BG1_ENABLE     0x200
#define BG2_ENABLE     0x400
#define BG3_ENABLE     0x800
#define OBJ_ENABLE     0x1000
#define WIN1_ENABLE    0x2000
#define WIN2_ENABLE    0x4000
#define WINOBJ_ENABLE  0x8000
```

DMA

```
#define DMA_ENABLE                  0x80000000
#define DMA_INTERRUPT_ENABLE        0x40000000
#define DMA_TIMING_IMMEDIATE       0x00000000
#define DMA_TIMING_VBLANK          0x10000000
```

```
#define DMA_TIMING_HBLANK          0x20000000
#define DMA_TIMING_SYNC_TO_DISPLAY  0x30000000
#define DMA_TIMING_DSOUND          0x30000000
#define DMA_16                      0x00000000
#define DMA_32                      0x04000000
#define DMA_REPEATE                 0x02000000
#define DMA_SOURCE_INCREMENT        0x00000000
#define DMA_SOURCE_DECREMENT        0x00800000
#define DMA_SOURCE_FIXED            0x01000000
#define DMA_DEST_INCREMENT          0x00000000
#define DMA_DEST_DECREMENT          0x00200000
#define DMA_DEST_FIXED              0x00400000
#define DMA_DEST_RELOAD             0x00600000
#define DMA_32NOW (DMA_ENABLE | DMA_TIMING_IMMEDIATE |DMA_32)
#define DMA_16NOW (DMA_ENABLE | DMA_TIMING_IMMEDIATE |DMA_16)
#define REG_DM0SAD      *(u32*)0x40000B0
#define REG_DMA0SAD_L   *(u16*)0x40000B0
#define REG_DMA0SAD_H   *(u16*)0x40000B2
#define REG_DMA0DAD     *(u32*)0x40000B4
#define REG_DMA0DAD_L   *(u16*)0x40000B4
#define REG_DMA0DAD_H   *(u16*)0x40000B6
#define REG_DMA0CNT     *(u32*)0x40000B8
#define REG_DMA0CNT_L   *(u16*)0x40000B8
#define REG_DMA0CNT_H   *(u16*)0x40000BA
#define REG_DMA1SAD     *(u32*)0x40000BC
#define REG_DMA1SAD_L   *(u16*)0x40000BC
#define REG_DMA1SAD_H   *(u16*)0x40000BE
#define REG_DMA1DAD     *(u32*)0x40000C0
#define REG_DMA1DAD_L   *(u16*)0x40000C0
#define REG_DMA1DAD_H   *(u16*)0x40000C2
#define REG_DMA1CNT     *(u32*)0x40000C4
#define REG_DMA1CNT_L   *(u16*)0x40000C4
#define REG_DMA1CNT_H   *(u16*)0x40000C6
#define REG_DMA2SAD     *(u32*)0x40000C8
#define REG_DMA2SAD_L   *(u16*)0x40000C8
```

```
#define REG_DMA2SAD_H   *(u16*)0x40000CA
#define REG_DMA2DAD     *(u32*)0x40000CC
#define REG_DMA2DAD_L   *(u16*)0x40000CC
#define REG_DMA2DAD_H   *(u16*)0x40000CE
#define REG_DMA2CNT     *(u32*)0x40000D0
#define REG_DMA2CNT_L   *(u16*)0x40000D0
#define REG_DMA2CNT_H   *(u16*)0x40000D2
#define REG_DMA3SAD     *(u32*)0x40000D4
#define REG_DMA3SAD_L   *(u16*)0x40000D4
#define REG_DMA3SAD_H   *(u16*)0x40000D6
#define REG_DMA3DAD     *(u32*)0x40000D8
#define REG_DMA3DAD_L   *(u16*)0x40000D8
#define REG_DMA3DAD_H   *(u16*)0x40000DA
#define REG_DMA3CNT     *(u32*)0x40000DC
#define REG_DMA3CNT_L   *(u16*)0x40000DC
#define REG_DMA3CNT_H   *(u16*)0x40000DE
```

Interrupts

```
#define REG_INTERRUPT   *(u32*)0x3007FFC
#define INT_VBLANK        0x0001
#define INT_HBLANK        0x0002
#define INT_VCOUNT        0x0004
#define INT_TIMER0       0x0008
#define INT_TIMER1       0x0010
#define INT_TIMER2       0x0020
#define INT_TIMER3       0x0040
#define INT_COMUNICATION  0x0080
#define INT_DMA0          0x0100
#define INT_DMA1          0x0200
#define INT_DMA2          0x0400
#define INT_DMA3          0x0800
#define INT_KEYBOARD      0x1000
#define INT_CART          0x2000
#define INT_ALL           0x4000
```

Miscellaneous Registers

```
#define REG_MOSAIC     *(u32*)0x400004C
#define REG_MOSAIC_L   *(u32*)0x400004C
#define REG_MOSAIC_H   *(u32*)0x400004E
#define REG_BLDMOD     *(u16*)0x4000050
#define REG_COLEV      *(u16*)0x4000052
#define REG_COLEY      *(u16*)0x4000054
#define REG_SG10       *(u32*)0x4000060
#define REG_SG10_L     *(u16*)0x4000060
#define REG_SG10_H     *(u16*)0x4000062
#define REG_SG11       *(u16*)0x4000064
#define REG_SG20       *(u16*)0x4000068
#define REG_SG21       *(u16*)0x400006C
#define REG_SG30       *(u32*)0x4000070
#define REG_SG30_L     *(u16*)0x4000070
#define REG_SG30_H     *(u16*)0x4000072
#define REG_SG31       *(u16*)0x4000074
#define REG_SG40       *(u16*)0x4000078
#define REG_SG41       *(u16*)0x400007C
#define REG_SGCNT0     *(u32*)0x4000080
#define REG_SGCNT0_L   *(u16*)0x4000080
#define REG_SGCNT0_H   *(u16*)0x4000082
#define REG_SGCNT1     *(u16*)0x4000084
#define REG_SGBIAS     *(u16*)0x4000088
#define REG_SGWR0      *(u32*)0x4000090
#define REG_SGWR0_L    *(u16*)0x4000090
#define REG_SGWR0_H    *(u16*)0x4000092
#define REG_SGWR1      *(u32*)0x4000094
#define REG_SGWR1_L    *(u16*)0x4000094
#define REG_SGWR1_H    *(u16*)0x4000096
#define REG_SGWR2      *(u32*)0x4000098
#define REG_SGWR2_L    *(u16*)0x4000098
#define REG_SGWR2_H    *(u16*)0x400009A
#define REG_SGWR3      *(u32*)0x400009C
#define REG_SGWR3_L    *(u16*)0x400009C
```

```
#define REG_SGWR3_H     *(u16*)0x400009E
#define REG_SGFIF0A     *(u32*)0x40000A0
#define REG_SGFIFOA_L   *(u16*)0x40000A0
#define REG_SGFIFOA_H   *(u16*)0x40000A2
#define REG_SGFIFOB     *(u32*)0x40000A4
#define REG_SGFIFOB_L   *(u16*)0x40000A4
#define REG_SGFIFOB_H   *(u16*)0x40000A6
#define REG_SCD0        *(u16*)0x4000120
#define REG_SCD1        *(u16*)0x4000122
#define REG_SCD2        *(u16*)0x4000124
#define REG_SCD3        *(u16*)0x4000126
#define REG_SCCNT       *(u32*)0x4000128
#define REG_SCCNT_L     *(u16*)0x4000128
#define REG_SCCNT_H     *(u16*)0x400012A
#define REG_P1          *(u16*)0x4000130
#define REG_P1CNT       *(u16*)0x4000132
#define REG_R           *(u16*)0x4000134
#define REG_HS_CTRL     *(u16*)0x4000140
#define REG_JOYRE       *(u32*)0x4000150
#define REG_JOYRE_L     *(u16*)0x4000150
#define REG_JOYRE_H     *(u16*)0x4000152
#define REG_JOYTR       *(u32*)0x4000154
#define REG_JOYTR_L     *(u16*)0x4000154
#define REG_JOYTR_H     *(u16*)0x4000156
#define REG_JSTAT       *(u32*)0x4000158
#define REG_JSTAT_L     *(u16*)0x4000158
#define REG_JSTAT_H     *(u16*)0x400015A
#define REG_IE          *(u16*)0x4000200
#define REG_IF          *(u16*)0x4000202
#define REG_WSCNT       *(u16*)0x4000204
#define REG_IME         *(u16*)0x4000208
#define REG_PAUSE       *(u16*)0x4000300
```

Timers

```
#define REG_TM0D        *(u16*)0x4000100
#define REG_TM0CNT      *(u16*)0x4000102
#define REG_TM1D        *(u16*)0x4000104
#define REG_TM1CNT      *(u16*)0x4000106
#define REG_TM2D        *(u16*)0x4000108
#define REG_TM2CNT      *(u16*)0x400010A
#define REG_TM3D        *(u16*)0x400010C
#define REG_TM3CNT      *(u16*)0x400010E
#define FREQUENCY_0     0
#define FREQUENCY_64    1
#define FREQUENCY_256   2
#define FREQUENCY_1024  1 | 2
#define TIMER_CASCADE   4
#define TIMER_IRQ       64
#define TIMER_ENABLE    128
```

Appendix D

Answers to the Chapter Quizzes

This appendix contains the answers to all the quiz questions from each chapter. I hope you got all the answers correct! If you miss more than three answers to any given quiz, I recommend that you go back and reread the relevant chapter and try again before proceeding. Good luck!

Chapter 1

1. B
2. C
3. A
4. D
5. B
6. A
7. C
8. D
9. A
10. B

Chapter 2

1. A
2. C
3. D
4. C
5. B
6. C
7. D
8. A
9. B
10. C

Chapter 3

1. A
2. B
3. C
4. B
5. D
6. B
7. C
8. C
9. A
10. D

Chapter 4

1. C
2. B
3. A
4. A
5. D
6. B
7. B
8. C
9. D
10. A

Chapter 5

1. B
2. A
3. B
4. C
5. D
6. B
7. A
8. C
9. D
10. D

Chapter 6

1. A
2. C
3. B
4. D
5. B
6. A
7. C
8. D
9. B
10. A

Chapter 7

1. B
2. C
3. D
4. A
5. D
6. B
7. B
8. C
9. A
10. A

Chapter 8

1. C
2. A
3. B
4. D
5. A
6. B
7. B
8. A
9. D
10. C

Chapter 9

1. C
2. A
3. C
4. B
5. A
6. D
7. C
8. B
9. A
10. D

Chapter 10

1. C
2. A
3. D
4. A
5. C
6. C
7. B
8. D
9. A
10. B

Chapter 11

1. A
2. B
3. A
4. C
5. D
6. B
7. D
8. A
9. B
10. B

Appendix E

Using The CD-ROM

The CD that comes with this book contains some important files that you will want to use when working through the sample programs in the book. The most important files on the CD are the source code files for the sample programs in the book.

The programs are stored in folders on the CD that are organized by chapter from the root \Sources folder. Inside \Sources, you will find chapter sub-folders: \Sources\Chapter01, \Sources\Chapter02, and so on. I recommend that you copy the entire \Sources folder to your hard drive, turn off the read-only property for all of the files, so you will be able to peruse the sample projects for the book more easily.

This book is about writing Game Boy Advance programs with a GCC compiler chain distribution called HAM, so I have included a version of HAM on the CD that you can install and use while reading the book and typing in sample programs. HAM is free for both personal and professional use, and is based on an open-source C/C++ compiler and ARM assembler, collectively known as a "compiler chain."

Everything you need to write Game Boy Advance programs is installed with HAM, including the emulator. There is also a \Tools folder with all of the various utility programs and other software used in the book, such as gfx2gba and VisualBoyAdvance.