## Using breakpoints to find graphics data by Bunkai (Jun 2023)

After reading the great tutorial by Labmaster from 24/03/06, which can be found at: <u>https://www.romhacking.net/documents/361/</u>. I wanted to port it over to no\$gba to use along other great documents like the one to find gba pointers using no\$gba by Phonymike. While also adding a few quality life additions like screenshots of the full process to help visualize it.

To follow this tutorial, you will need the following: no\$gba: <u>https://problemkaputt.de/gba.htm</u> yy-chr : <u>https://www.romhacking.net/utilities/119/</u> Target ROM [The Legend of Zelda, The Minish Cap]. Checksum, crc32: ABCEBBB1

Since this is just a port over and my intention is to use this as a learning experience to do it later with mGBA, many of the explanations will be short like in the original (many will indeed be copy pasted), however i will try my best to add links and brief notes on some technical parts where (for me) Labmasters was too scarce. If you need to expand on, it's encouraged to read the gba documentation at <a href="https://rust-console.github.io/gbatek-gbaonly/">https://rust-console.github.io/gbatek-gbaonly/</a> . If you still have any question about the process you may ask them in the rhdn forum <a href="https://www.romhacking.net/forum/index.php">https://www.romhacking.net/forum/index.php</a> or in the discord server, using the proper channels for that, but I won't answer any private questions about your projects.

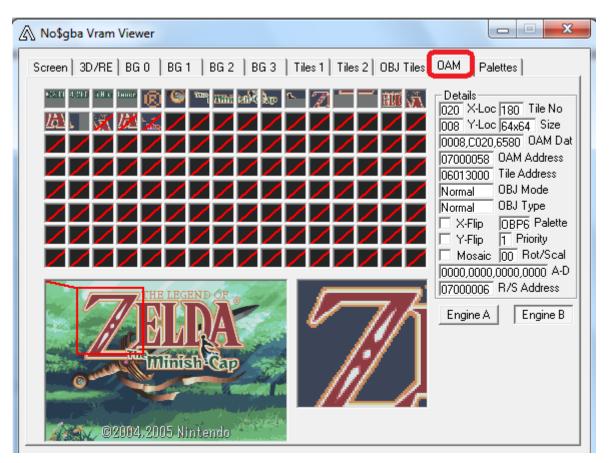
Our Target for this guide is the title screen logo from: The Legend of Zelda, The Minish Cap.

First step is to find where the graphics data is in VRAM, and the first place we look is on the various backgrounds. Open the BG Maps, and see if it's one of the Backgrounds.

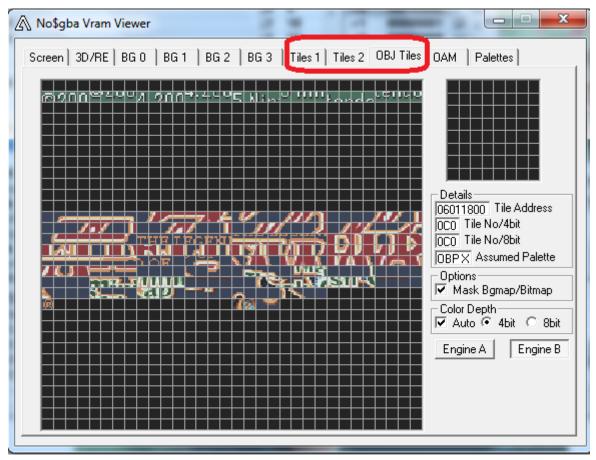
🖉 No\$gba Debugger (Fullvers	ion)	-		
	Window Utility Options	Help		
080B14EC 4770 bx 080B14EE 0000 movs	User Screen	F5	;8 8	r0 5A61E83C n r1 00000001 ₩ 7
080B14EC 0000 movs 080B14F0 2001 movs	3D Rendering	F5	;2 12	
080B14F2 DF19 swi	BG Maps	F5 +	BG0 Map	3 0000000 🗍 💭 🔛 🧨 M M M MA
080B14F4 4770 bx	Tile Viewer	F5 ►	BG1 Map	
080B14F6 0000 movs 080B14F8 DF08 swi				5 00000000
080B14FA 4770 bx	OAM Viewer	F5	BG2 Map	-7 0000000
080B14FC 2200 movs	Palette Viewer	F5	BG3 Map	-8 0000000 🗆 q
080B14FE · DF05 swi 080B1500 4770 bx	I/O Map	F10 ▶	;8 56 ;8 64	r9 00000000 C C 2004,2005 Nintendo
080B1500 4770 bx 080B1502 0000 movs	Profiler	Alt+P ▶	;8 64	
080B1504 4B04 1dr	Profiler	AIL+P V	;9 75	r12 030000B0
080B1506 2200 movs	ARM9 (Main CPU)		;2 77	r13 03007EE8 Reload Edit File Screenshot Upload
080B1508 701A strb 080B150A 4B04 ldr	ARM7 (Sub CPU)		;5 82	r14 08016E87 r15 080B14FE
0808150C 2200 movs	DSP (Teaklite)		;2 93	CDSY 600003F
080B150E 701A strb			;5 98	spsr 00000000
080B1510 3BFA subs	Xtensa (Atheros Wifi)		;2 100 -	D3007EFC D80000F0 Return from ???
00000000 FE FF FF EA FI	RL78 (MCU)			03007EF8 00000000 Pushed r6
00000010 FE FF FF EA FI	38600 (IR.cart.meter)	-	🔳	03007EF4 00000000 Pushed r5
00000020 OF 50 2D E9 0 00000030 OF 50 BD E8 0	38601 (IR.cart.walker		^ X_ ^	03007EF0 00000000 Pushed r4 03007EEC 08055F69 Return from Lxx 8016E78h
00000040 00 B0 4F E1 0	38602 (IR.meter)			D3DD7EE8 · D3DD1000 Pushed r4
00000050 OB FO 29 E1 O	38606 (IR.walker)	-		03007EE4 00000000 Pushed r5 -
0000008	Corelink DMA			System DI GBA 1/1

\Lambda No\$gba Debugger (Ful	version)	
File Search Run Deb	ig Window Utility Options Help	
080B14EC 4770 b 080B14EE 0000 m 080B14F0 2001 m		THE LEGEND OF
080B14F2 DF19 s 080B14F4 4770 b	i Screen   3D/RE BG U   BG 1   BG 2   BG 3 Tiles 1   Tiles 2   OBJ Tiles   OAM   Palettes	MINIA Minista Gap
080B14F8 DF08 s 080B14FA 4770 b 080B14FC 2200 m	i de la companya de la	
	vs	©2004,2005 Nintendo Run Next GBA Specs CPU Specs
080B1504 4B04 1 080B1506 2200 m 080B1508 701A s	vs rb	Edit File Screenshot Upload
080B150E 701A s	vs rb 0600EFEE Map Address 06000000 Tile Address	
080B1510 3BFA s 00000000 FE FF FF E 00000010 FE FF FF E		rom ???
00000020 OF 50 2D E 00000030 OF 50 BD E 00000040 00 B0 4F E	Grid V Seyx	4 rom Lxx_8016E78h r4
00000050 OB FO 29 E		5 -
0000008		System DI GBA 1/1

In this In this case, it's not, so it must be a sprite. To confirm this, open the OAM viewer and scroll across until you find the parts of the logo.

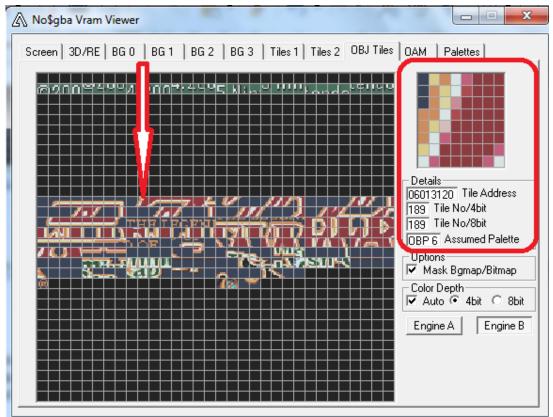


A quick background on GBA sprites. The actual graphics are stored at 0x06010000, (OAM data, which controls position, scaling etc..., is at 0x07000000), that's why now, you have to open the tile viewer and select the 0x601000 char base.



You should see the tiles that make up the logo (note the three different palettes used for the red logo, the green 'Minish Cap' text, and the sword.

So, now we need to figure out where in the ROM this data has come from. Click on one of the tiles that make up the logo (for this example, we'll use the top-right of the 'Z').



The Tile Viewer will give you the address for this tile – 06013020. Now, we will be using this information to do our reverse engineering. In addition, we will use the write breakpoints from no\$gba. Which you can do by pressing ctrl+B or following the image below:

File Search Run	r (Fullversion) Debug Window Utility O	ptions Help									
080B14EC 4770 080B14EE 0000	Toggle Breakpoint	F2	;8 8			5A61E83C 00000001			T	HE LEGEND	ÓF .
080B14F0 2001	Define Break/Condition	^B		2		00000000	Z Z	le l			
080B14F2 DF19	Define Watchpoint	٨I	;8 2	20	r3	00000000	Γ v		<i>///</i> //	Я И Ц	
080B14F4 4770 080B14F6 0000	Clear all breakpoints			28 10		03001000 00000000	Πi	-	1009		
080B14F8 DF08	cicar an breakpoints			8		03001000	E f	and a failed and	Million and and and and and and and and and an	Minisha	Sep 🖓
080B14FA 4770	Reset F10 debug clks		;8 4	16		00000000	🗹 t			CALCULATION OF	A COLUMN 1
080B14FC 2200	Toggle Datazone	^D	,	8		00000000	<u> </u>			and the second second	
080B14FE · DF05 080B1500 4770	Assemble into Memory	it 		56		00000000 00000000	C	1000		and the second	
080B1502 0000	movs r0,r0		,	6		00000000	U	Trace	Run Next	GRA Speci	s CPU Specs
080B1504 4B04	ldr r3,=Lxx_400021	18h		5		03000090			Edit File	<u> </u>	· · · ·
080B1506 2200 080B1508 701A	movs r2,0h strb r2,[r3]			2		03007EE8 08016E87		Reload	EditFile	Screensho	t Upload
080B150A 4B04	ldr r3,=Lxx_3007Fi	Ah	;9 9			080B14FE					
080B150C 2200	movs r2,0h			3		6000003F					
080B150E 701A 080B1510 3BFA	strb r2,[r3]			8	spsr	00000000					
	subs r3,0FAh		;2 1	• 00		'EFC 08000			om ???		
00000000 <b>11</b> FF F 00000010 FE FF F		IO EA FE FF FF EA IO FA FF FF FF FA		···· 🔺		'EF8 00000 'EF4 00000					
	2D E9 01 03 A0 E3 00 E0 8	SF E2 04 F0 10 E5	.P			'EFO 00000					
	,, ,, ,, ,, ,, ,, ,, ,, ,, ,, ,, ,, ,,	D E9 02 C0 5E E5	.P^X	.^.		EEC 08055				16E78h	
00000040 00 B0 4	¥F E1 00 08 2D E9 80 B0 ( 29 E1 03 00 5C E3 00 B0 (	IB E2 1F B0 8B E3 ID F3 80 B0 A0 03	0			'EE8 · 0300 'EE4 00000			4		
	27 ET DO DO OG EO DO BO I	IN CO ON RN HN NO		•••• •	03007	CC4 00000	000 P	usned ro	-		
0000000									System	DIGE	BA 1/1

After that, a prompt will pop up. There, you Place a write breakpoint by typing in: **[06013020]!!** 

No\$gba	×
Enter Breakpoint as [address] [.condition]	OK
106013020]!!	Cancel

This puts a write breakpoint in 06013020, now reset & run the emulation.

File Search	Ru	n Debug Window	Utility Options Help						
080B14EC	4	Run	F9	;8	8	-	rØ	5A61E83C n	
080B14EE	01	Kun	13	;2	10		r1	00000001 🔽 z	
080B14F0	21	Reset & Run	Keypad *	;2	12		r2	00000000 🔽 c	
080B14F2	DI	Undo Last Run		;8	20		r3	0000000 -	
080B14F4	4	ondo Last Null		:8	28		r4	03001000	

## It should break thus:

ile Search Run	Debug	Window Utility O	ptions Help	1			_							
80ADF2C 6898	ldr	r0,[r3,8h]			;9	9		rØ	84000580	l <b>v</b> r	)			Autor the second
80ADF2E 6010	str	r0,[r2]			;5	14		r1	84000000	Γz	100	and and a lot		1215-1
BDADF30 6051	str	r1,[r2,4h]			;5	19		r2	04000004	E c	6-650		and the	
30ADF32 88D8	ldrh	rØ,[r3,6h]			;6	25		r3	020244C4			and the	1300	
BOADF34 DOCD	lsls	r0,r0,3h			;2	27		r4	00000004		Profession of the			C. 10. 10
BDADF36 2184	MOVS	r1,84h			;2	29		r5	00000030	12 2	100		in product the second	
30ADF38 0609	lsls	r1,r1,18h			;2	31		r6	03001000		6	and the second		1 1
30ADF3A 4308	orrs	rØ,r1			;2	33		r7	00000000				and the second	all har file
BOADF3C 6090	str	rØ,[r2,8h]			;5	38		r8	00000000		1			
BDADF3E • 6890	ldr	rØ,[r2,8h]			;4	42		r9	00000000	_ C	100	a tel destruction		
BOADF40 88DA	ldrh	r2,[r3,6h]			;6	48		r10	00000000					and the strength of a
BOADF42 3A10	subs	r2,10h			;2	50		r11	00000000		Trace	Run Next	GBA Specs	CPU Spec
30ADF44 2A00 30ADF46 DD19	cmp ble	r2,0h	- 1		;2	52 60		r12 r13	03000090 03007ECC		Reload	Edit File	Screenshol	t Upload
BOADF48 E012	bie	Lxx_80ADF7Ch Lxx 80ADF7Dh	;↓ ;↓		;8	68		r13	DSDD7ECC		1101000	Lakino		
BOADF48 CD12	D Movs	r0,r0	;+		;8	70		r14 r15	D8DADF3E					
30ADF4C 2800	CMP	r0,0h			;2	72			8000003F					
BDADF4E 0601	lsls	r1,r0,18h			;2	74			00000000					
BOADFSO FFFF	b1	lr+0FFEh			;10		-	•						
			<b>,</b> ,				_		7EE0 0000					
<global memor<="" td=""><td>'y cnange</td><td>preak2</td><td></td><td></td><td>[06013]</td><td>nsnl:</td><td><u> </u></td><td></td><td>7EDC 0300</td><td></td><td></td><td>-</td><td>00F746</td><td></td></global>	'y cnange	preak2			[06013]	nsnl:	<u> </u>		7EDC 0300			-	00F746	
									7ED8 080A 7ED4 0300			rom Lxx_80 4	HUC/4N	
									7ED4 0300 7ED0 0000			-		
									7ECC • 000					
									7EC8 0300					

How do we know that this is the right break? If you open the memory viewer and see what 06013020 is whilst on the menu, you'll know that these are the correct values.

🕼 No\$gba Debugger (Fullversion)			×
File Search Run Debug Window Utility Options Help			
06012FFC 00000000 andeg r0,r0,r0	;2 2	<b>r0 84000580 </b> ✓ n	Julian
06013000 00000000 andeq r0,r0,r0	;2 4	r1 84000000 🗆 z	1. Sand Carlos and Car
06013004 00000000 andeq r0,r0,r0	;2 6	r2 04000004 🗖 6	
06013008 00000000 andeq r0,r0,r0	;28	r3 020244C4	2000
0601300C 00000000 andeq r0,r0,r0	;2 10	r4 0000004 — 🚺 🔤 😂	and allowed a
06013010 00000000 andeq r0,r0,r0	;2 12	r5 0000030 🔚 📜	The second s
D6D13D14 DDDDDDDD andeq rD,rD,rD	;2 14	r6 03001000 🔤 🚺 🕜 🍘	
D6D13D18 DDDDDDDD andeq rD,D,rD	;2 16	r7 0000000 🗹 t	and the second of the second
0601301C 00000000 andeq r0,r0,r0	;2 18	r8 0000000 🗌 q	Internet and a first of the later
06013020 db 000h	<bookmark< th=""><th>r9 0000000 C</th><th></th></bookmark<>	r9 0000000 C	
06013021		r10 0000000 P	and the second
06013024 00000000 andeq r0,r0,r0	;2 20	r11 00000000 U Trace Run Nex	t GBA Specs CPU Specs
06013028 00000000 andeq r0,r0,r0	;2 22	r12 03000090 Reload Edit File	Screenshot Upload
0601302C CCCC0000 stclgt p0,c0,[r12]	.0 04	r13 03007ECC Reload Edit File	Screensnor Opload
06013030 EECCC000 cdp p0,12,c12,c12,c0,0 06013034 FFFEC000 swinv (ARMv2) 0FEC000h ;Invalid!!!	;2 24 ;6 30	r15 080ADF3A	
06013034 124FCC00 subne r12,=Lxx 6013040h	;2 32	cpsr 8000003F	
06013030 1116ECOD tstne r6,r0,lsl 18h	;2 32	spsr 0000000	
06013040 00000000 andeg r0,r0,r0	;2 34	· · ·	
	•	03007EE0 00000000 Pushed r5	*
<pre><global break="" change="" memory=""></global></pre>	[06013020]!	03007EDC 03001000 Pushed r4	
Run		03007ED8 080ADDCF Return from Lxx_8	UHUE/AN
Define Breakpoint		03007ED4 03001000 Pushed r6	
		03007ED0 0000030 Pushed r5 03007ECC · 00000004 Pushed r4	
Define Watchpoint		03007EC8 03001000 Pushed r4	
Data Window		USUDICO USUDIDUD PUSNed P4	•
		System	DI GBA 1/1
		0004000F 0.4 4 4040F744	

06013020 00 <mark>0</mark> 11 лл	<u> </u>	00	00	00	CC	CC	
06013030 00 C	Run	12	00	EC	16	11	
06013040 00 0	Goto						
06013050 EE E		1.1					
06013060 00 0							
06013070 EE Ec cc	EE TT TT TT TT 11 11 11	-11	11	11	11	11	••••••
06013021							

	1116ECOD tstne r6,r0,lsl 18h	;2 34	
06013040	No\$gba	;2 36	۳Ì
06013020			-
06013030	Address to position to: OK 1		
06013040	06013020		
06013050			
06013060	EE EE EE FF FF FF FF 11 11 11 11 11 11 11 11		
06013070	EE EE EE EE FF FF FF FF II II II II II II II II II		• ]
06013021			

So now we need to see what opcode triggered this breakpoint. In the diassembly viewer, go to 080adf3e and scroll up a bit. The instruction that triggered the breakpoint is the one immediately before, as you can see in the image below:

he instruction	\land No\$gba	Debugg	ger (Fullve	rsion)		_	_	_		_		_	_		_
hat triggered	File Searc	h Rur	Debug	Window	Utility	Options	Help								
he breakpoint	<b>D8DADF2C</b>	6898	ldr	r0,[r	3,8h]	The	address	ia x02	;9	9		rØ	84000580	🗸 n	set/
the one	080ADF2E		str	r0,[r					;5	14		r1		z	- And
	080ADF30		str	r1,[r		is O	<b>04000</b>	004 🗖				r2	040000D4	С	Sec. 11
mediatly before.	080ADF32		ldrh						;6	25		r3	02024404	Τv	State of the second
-	080ADF34		lsls						;2	27		r4	00000004	Ξi.	And a state of the
	080ADF36		movs						;2	29		r5	00000030	ΞĒ	the state of the
he game is 👘	080ADF38 080ADF38		lsls						;2	31		r6	03001000	₹t	(Q ***
topped at this	D8DADF3C		str	r0,[r					;2	33 38		r7 r8	00000000	Γq.	
	D8DADF3C				2,8n] r2,8h]				;5			re r9		-	march - 2 March
struction	USUAUF SE		ldrh						;6	48		r7	00000000	CP	
_	080ADF42		subs						;2	50		r11	00000000	PU	1 408/2000A/%/
	080ADF44		cmp	r2,0h					;2	52		r12	03000090	Ο,	Trace F
	080ADF46		ble		ØADF7Ch	;+			;8	60		r13	03007ECC		Reload
	080ADF48		b		DADF70h				;8	68		r14	OSOADDCF	1	i
	080ADF4A		movs						;2	70		r15	080ADF3E		
	080ADF4C	2800	cmp	r0,0h					;2	72		cpsr	8000003F		
	080ADF4E	0601	lsls	r1,r0	,18h				;2	74		spsr	00000000		
	080ADF50	FFFF	b 1	lr+OF	FEh	; 66)	•		;10	84	Ŧ	0.000	, 7EEO 000000	<u></u>	ichad n5
	06013020	<u> </u>		<u>a aa aa a</u>	<u>n nn nn</u>		IO OO CC C	c			_		7EDC 030010		
	06013030						IO EC 16 1		0				7ED8 080ADD		
	06013040				n nn nn								7ED4 030010		
					F 11 11		1 11 11 1						7EDO 000000		
	06013060						C CC CC C						7ECC • 00000		
							1 11 11 1				-		7EC8 030010		
															S

080adf3c 6090 str r0, [r2, #0x8]

This writes the value in r0 to the address in r02 + 0x8.

Now this is where it's handy to know your GBA IO registers. The address in r02, 0x040000d4, is one of those that control DMA transfers (<u>https://problemkaputt.de/gbatek.htm#gbadmatransfers</u>). 0X040000D4 + 0x8 is 040000DC, and we're writing a word to it - this actually corresponds to both the word count register and control register for DMA channel 3. (you can read the reason in the gbatek link, gbatek is one of the most useful document a GBA hacker could have).

The source register for DMA channel 3 is at 0x040000D4, so take a peek at the value there by using right clicking as shown in the image below:

	080B150E 080B1510			strb subs		-	[r3] Ofat		;5 ;2	98 100	-
	00000000 00000010 00000020 00000030 00000040 00000040	FE F OF 5 OF 5 00 B	F FF 0 20 0 BD 0 4F	EA FE E9 01 E8 04 E1 00	FF 03 F0 08	F A 5 2		Break/Watch Window	)		*
And g	00000006 oto: <b>04000</b>	0d4	1	lo\$gba Address t 040000d		sitior	n to:	OK Cancel			

You can see the result here:

File Search Run	Debug V	Vindow Utility (	Options Help	)							
080ADF2C 6898	ldr	r0,[r3,8h]			;9	9		rØ	84000580	🗹 n	and /
080ADF2E 6010	str	r0,[r2]			;5			r1	84000000	Γz	and they
080ADF30 6051	str	r1,[r2,4h]			;5			r2	04000004	□ c	Self all
080ADF32 88D8	ldrh	rØ,[r3,6h]			;6			r3	020244C4		
080ADF34 00C0	lsls	r0,r0,3h			;2			r4	00000004	Ē.	Profession of the
080ADF36 2184	movs	r1,84h			;2			r5	00000030		11 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
080ADF38 0609	lsls	r1,r1,18h			;2			r6	03001000		0*
080ADF3A 4308	orrs	r0,r1			;2			r7	00000000	₹ t	and the second second
080ADF3C 6090	str	rØ,[r2,8h]			;5			r8	00000000	q	1 1
080ADF3E • 6890	ldr	r0,[r2,8h]			,	4 42		r9	00000000	С	and the second second
080ADF40 88DA	ldrh	r2,[r3,6h]			;6			r10	00000000	P	
080ADF42 3A10	subs	r2,10h			;2			r11	00000000	U	Trace
080ADF44 2ADD	cmp	r2,0h			;2			r12	03000090	1	Dalaad
080ADF46 DD19	ble	Lxx_80ADF7Ch	;+		;8			r13	03007ECC		Reload
080ADF48 E012	b	Lxx 80ADF70h	;↓		;8	68	Ŧ	r14	080ADDCF	8 - C	
04000000 <mark>00 00 0</mark>	0 00 60	67 8D 08 00 30	01 06 80 0	5 00 04	···· `g0			r15	080ADF3E		
U4UUUUEU 00 00 0	0 00 00		00 00 00 00	0 00 00				cpsr			
040000F0 00 00 🖌	Allres	s in little e	ndian 0 01	0 00 00				spsr	00000000		
04000100 A3 1E Ú			0 01	0 00 00				0300	7EDO 00000	030 Pr	ished r5
04000110 00 00 0	0 00 00	00 00 00 00 00	00 00 00 00	0 00 00				0300	7ECC • 0000	0004 P	ushed r4
04000120 00 00 0	0 00 00	<u> </u>	00 00 00 00	n n n n			-	0300	7EC8 03001	000 P.	ished rå

The address value you want is: 08 8d 67 60

Little Endian > denoting or relating to a system of ordering data in which the least significant units are put first.

You could also do it checking the DMA Registers with:

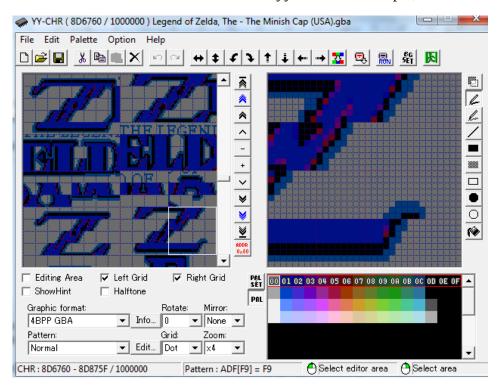
🕼 No\$gba Debugge	r (Fullvers	iion)				-				-		-	-		- <b>X</b>
File Search Run	Debug	Window Utility Options	Help												
080ADF2C 6898	ldr	User Screen	F5	1	;9	9		rØ	84000580		n	and /			10000-73
080ADF2E 6010	str				;5	14		r1	84000000		z	////	handle 1	- 100 M	all and the
080ADF30 6051	str	3D Rendering	F5		;5	19		r2	04000004		с	Sel Sel.	Section 11	and the state	
080ADF32 88D8	ldrh	BG Maps	F5 🕨		;6	25	_	r3	020244C4		v		1	2	Ser Starles
080ADF34 00C0	lsls	Tile Viewer	F5 ►	L	;2	27		r4	00000004	E i	i	ALC: NO. T.		1000	The start of the
080ADF36 2184	movs			L	;2	29		r5	00000030		F	CONTRACTOR			
080ADF38 0609 080ADF3A 4308	lsls	OAM Viewer	F5	L	;2	31 33		r6 r7	03001000 00000000			@'		1000	100
080ADF3C 6090	orrs str	Palette Viewer	F5		;2	38		r7 r8	00000000						and the second s
080ADF3E · 6890	ldr				,,,	10		10	00000000	-	•	mul 2			1 Harley Store
080ADF40 88DA	ldrh	I/O Map	F10 >	١	/ideo	Contr	ol		00000000		Ç	A DEL	1.60		- Sec. 4.
080ADF42 3A10	subs	Profiler	Alt+P ▶			ontrol			00000000			T.		lonvo	
080ADF44 2A00	cmp	Fromer	ALTER						03000090		-	Trace	Run Next	GBA Specs	CPU Specs
080ADF46 DD19	ble	ARM9 (Main CPU)		5	Soun	d			03007ECC			Reload	Edit File	Screenshot	Upload
080ADF48 E012	b	· · · · ·		[	DMA	Regist	ers 🖌								
080ADF4A 0000	movs	ARM7 (Sub CPU)			Time	-			DODNOT		5			-	
080ADF4C 2800	cmp	DSP (Teaklite)			IIme	rs -			8000003F						
080ADF4E 0601	lsls	Xtensa (Atheros Wifi)		(	Other				00000000						
080ADF50 FFFF	b1								'EEO 00001	1000	Pu	shed r5			
04000000 00 00 C	0 🔟 6	RL78 (MCU)		1	All in	one			'EDC 0300'	1000	Pus	shed r4			
040000E0 00 00 0	0 00 01	38600 (IR.cart.meter)	L					0300	I7ED8 080AI	DDCF	Ret	turn fra	m Lxx_80	ADE74h	
040000F0 00 00 (	1	38601 (IR.cart.walker							17ED4 0300 <sup>.</sup>						
04000100 A3 1E 8	1								17EDO 00001						
04000110 00 00 0	1	38602 (IR.meter)						_	17ECC • 0001				•		
04000120 00 00 0	0 00 01	38606 (IR.walker)		•••••			Ŧ	0300	17EC8 0300 <sup>4</sup>	1000	Pus	shed r4			-
Execution Time: 91,490,	189 Cycle	Corelink DMA											System	DI GB	A 1/1

And see the value here:

4 IRQ 🔽	DMA1SAD DMA1DAD 1CNT_L 1CNT_H Increment Increment
1 4720 0BC 0040 0C0 1 0C4 DMA 0C6 DMA -6 Dest -8 Src 2-13 Time 3 Repeat <b>V</b> 4 IRQ <b>V</b>	DMA1SAD DMA1DAD 1CNT_L 1CNT_H Increment Increment SndFifo 7 b10 32bit
4720 0BC 00A0 0C0 1 0C4 DMA 0C6 DMA -6 Dest -8 Src 2-13 Time 3 Repeat <b>▼</b> 4 IRQ <b>▼</b>	DMA1DAD 1CNT_L 1CNT_H Increment Increment SndFifo 5 b10 32bit
00A0 0C0 1 0C4 DMA 0C6 DMA -6 Dest -8 Src 2-13 Time 9 Repeat <b>1</b> 4 IRQ <b>1</b>	DMA1DAD 1CNT_L 1CNT_H Increment Increment SndFifo 5 b10 32bit
OC4 DMA OC6 DMA -6 Dest -8 Src 2-13 Time 9 Repeat 🔽	1CNT_L 1CNT_H Increment Increment SndFifo 7 b10 32bit
OC6 DMA -6 Dest -8 Src 2-13 Time Repeat I 4 IRQ I	1CNT_H Increment Increment SndFifo 7 b10 32bit
-6 Dest -8 Src 2-13 Time 3 Repeat I⊽ 4 IRQ IIV	Increment Increment SndFifo b10 32bit
-8 Src 2-13 Time 9 Repeat I⊽ 4 IRQ I⊽	Increment SndFifo b10 32bit
2-13 Time ∂Repeat 🔽  4 IRQ 💽	SndFifo 510 32bit
BRepeat Iv 4 IRQ Iv	b10 32bit
4 IRQ 🔽	
	b15 Active
4A40 -> 040	000040 Curr
ODC DMA	3CNT_L
ODE DMA	3CNT_H
-6 Dest	Increment
-8 Src	Increment
2-13 Time	Burst
Repeat 🔽	b10 32bit
	b15 Active
7D60 -> 06	014600 Curr
1 5 7 1 9 1	14 IRQ 🛛

This means that, the data has been copied from 088D6760. Note that the next word in memory is 06013000, sounds familiar? It's the destination address (040000d8 is the destination register for channel 3).

Now, we convert that location into a gba rom memory address by substracting hexidecimal 8000000: **0x088D6760 - 0x8000000 = 0x8D6760** and go to that address in our graphics editor of preference, for this tutorial we will show the results in yy-chr as an example, below:



Very Important Caveat: Although the steps up to setting the breakpoint will be the same for most games (though the graphics might be in one of the other Character bases in the tile viewer), the way the game gets that data into VRAM will differ. Some may use a Software Interrupt (SWI) to copy or decompress the data from a ROM address - others will have their own functions to decompress graphics (sometimes directly to VRAM, others into WRAM, then copied into VRAM using DMA or a SWI, or even with stmia/ldmia opcodes), so it's up to you to figure out what's going on past this point.

That concludes this brief tutorial, happy hacking and reverse engineering.

- Curiosity leads to knowledge, be curious!