How to setup GP2X development environment on Windows and use Code::Blocks as IDE

Written by Delrina Chang (<u>gmail.com@delrinachang</u>) version 1.0 January 9th 2007

1. Introduction

The GP2X is an open-source, Linux-based handheld video game console and media player created and sold by GamePark Holdings of South Korea. Released on <u>November 10</u>, <u>2005</u> in South Korea, the GP2X is designed to play video and music, view photos, and play games. It has an open architecture, allowing anybody to develop and run software. Also, there is the possibility for additional features (such as support for new media formats) to be added in the future due to the upgradeable <u>firmware</u>.

A popular use of the GP2X is for running <u>emulators</u>, which allow one to use software from another system on the GP2X.

GP2X and its development kits (breakout board) can be purchased at http://www.gp2x.com

To develop graphics applications (games, multimedia apps) on GP2X, we need to use an IDE such as Visual Studio, Dev C+ or Code::Blocks to write code in host PC. In this case, we use Code:Blocks IDE integrated with MingW32 because it is free. The OS in host PC is Windows XP because it is most popular OS today. Using Code::Blocks, MingW32, we will compile our application into two target operating systems:

- Windows platform for debugging and simulation, *.exe file.
- Linux platform on GP2x, *.gpe file.



2. Set up GP2X development kit

1- Download: gp2x devpack compiled by *Octoate* at here:

<u>http://archive.gp2x.de/cgi-bin/cfiles.cgi?0,0,0,0,14,1362</u>. Unzip the devkitGP2X directory into your root directory. If you chose a development directory other than **C:/devkitGP2X**, you will need to correct the paths in bin\arm-linux-sdl-config and demo\Makefile.

- 2- Upgrade the SDL libraries in C:/devkitGP2X by download this zip file *sdl-libs-211006.zip* compiled by *Guyfawkes* at <u>http://archive.gp2x.de/cgi-bin/cfiles.cgi?0,0,0,0,19,1875</u>. Extract and copy all directories bin, etc, include, info, lib, man and share to C:/devkitGP2X. Overwrite if necessary.
- 3- Add the bin directories to your path. Click on Start->Settings->Control Panel, open System->Advanced->Environment Variables, select PATH and add *C:\devkitGP2X\bin;C:\devkitGP2X\minsys\bin;* to your path.
- 4- Download a zip file SDL Test Program Thingy version 1.51 made by *Guyfawkes*. Unzip it to a directory for example, example SDLTest. Run command line and type "make" to compile this demo program. If compilation is successful, copy sdltest.gpe and sdltest folder to GP2X to test drive. If compilation fails, then check step 1, 2, 3 or go to http://www.gp32x.com/board/index.php to ask for help.

Syst	em Properties			? ×
	System Restore	Automatic Updates	Remote	1
En	vironment Variable	5	?	×
	Edit System Varia	ble	<u>?×</u>	
	Variable <u>n</u> ame:	Path		
	Variable <u>v</u> alue:	in;C:\devkitGP2X\minsys\bin	;C:\MinGW\bin	
		OK	Cancel	
			۲ <u> </u>	
Γ	System variables			111
	Variable	Value	▲	
	LM_LICENSE_FILE	C:\ADAMS\Flexl\icense.dat		
	NUMBER_OF_P	1		
	OS Doth	Windows_N1	indouraus	
	PATHEXT	COM: EXE: BAT: CMD: VBS:	VBE: 1S:	
	1	100.1,12.12,12.11,10.12,1120,1		
		Ne <u>w</u> Edit	Delete	
		OK	Cancel	Ľ

3. Set up Code::Blocks IDE

In the previous section, I mentioned how to setup GP2x development kit, which is used to create *.gpe file for Linux OS in GP2X. In this section, we go further to install Code::Blocks, MingW. The MingW is integrated with Code::Blocks to create *.exe file.

- 1- Down load nightly built Code::Blocks at <u>http://forums.codeblocks.org/index.php?board=20.0</u>, unzip to a folder C:\CodeBlock.
- 2- Down load MinGW Installer 5.0.2 at <u>http://prdownloads.sourceforge.net/mingw/MinGW-5.0.2.exe</u>. This MingGW is compiler to produce executable file for Windows OS. Run installer, we just need gcc and g++ compiler therefore unselect g77, ada and java components. Suppose MingW is installed at C:\MinGW
- 3- Go here <u>http://www.mingw.org/download.shtml</u> to download latest mingw32-gcc.exe, mingw32-g++.exe, ar.exe, gdb.exe, windres.exe, mingw32-make.exe files to update to C:\MinGW\bin folder.
- 4- Click on Start->Settings->Control Panel, open System->Advanced->Environment Variables, select PATH and add C:\MinGW\bin. Otherwise, you cannot start up Code::Blocks because it requires a DLL file *mingwm10.dll*

- 5- Next, we need install SDL library for MinGW, go here <u>http://www.libsdl.org/download-1.2.php</u> and download <u>http://www.libsdl.org/release/SDL-devel-1.2.11-mingw32.tar.gz</u> unzip to folder C:\SDLMingW
- 6- In this step, we configure Code::Blocks global compiler settings. Notes, in Code::Blocks, the project compiler settings augment to the global compiler settings. First we configure GCC compiler settings for compiling Win32 *.exe file.
 - a. Start Code::Blocks and go to menu Settings > Compiler and Debugger.
 - b. In the combo box labeled Selected Compiler, select GNU GCC, you may copy this compiler profile as new one such as "GCC Win32 SDL". As for me, I make changes directly to "GNU GCC".
 - c. Go to tab Compiler, then tab #defines, add WIN32. It is equivalent to #define WIN32 in our source code. It will tell the compiler to compile only Win32 code not GP2X code when creating *.exe file.
 - d. Go to tab Linker, add following libraries: mingw32, SDLmain, SDL.dll, SDL_image, user32, gdi32, winmn, dxguid. (*Notes this is very basic libraries for a simple SDL application. For specific project, you may include additional library to project's linker settings*).

Global compiler settings Selected compiler Glubal compiler settings Glubal compiler settings Global compiler settings Compiler linker Directories Programs Custom variables Other Policy: Image Variable Solution Batch builds Solution Debugger settings Add Edit Delete Clear	c	Compiler and debugger settings					
Selected compiler GNU GCC Set as default Copy Rename Delete Reset defaults Compiler Linker Directories Programs Custom variables Other Polcy: Link libraries: Polcy: Link libraries: Diman SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SDL.di SD	Γ		Global compiler settings				
GNU GCC Global compiler settings Global compiler settings Profiler settings Batch builds Debugger settings Add Edit Delete Clopy all to			⊂ Selected compiler				
Set as default Copy Rename Delete Reset defaults Global compiler settings Compiler Linker Directories Programs Custom variables Other Profiler settings Compiler Linker Directories Programs Custom variables Other Batch builds SDL_dil SDL_main SDL_main Other linker options: Image: graph of the graph		500 -		-			
Global compiler settings Compiler Linker Directories Programs Custom variables Other Policy: Image: Custom variables Other SDL. dil SDL. dil Global compiler settings Add Add Edit Delete Copy all to Image: Custom variables Other		502	Set as default Copy Rename Delete Reset def	aults			
Global compiler settings Compiler Linker Directories Programs Custom variables Other Policy: > Profiler settings Clink libraries: Batch builds SDL_minge Debugger settings Add Edit Delete Clear Copy all to Image							
Policy: Profiler settings Batch builds Debugger settings Add Edit Delete Clear Copy all to		Global compiler settings	Compiler Linker Directories Programs Custom variables Other				
Profiler settings Batch builds Debugger settings Add Edit Delugger settings Other linker options: Other linker options: SDL and SDL			Policy:				
Profiler settings Profiler settings Batch builds Debugger settings Add Edit Delete Clear Copy all to				-			
Profiler settings Batch builds Debugger settings Add Edit Delete Clear Copy all to			Link libraries:				
Profiler settings SDL.dll SDL_image user 32 gdl32 winnm dxguid Add Edit Delete Clear Copy all to			SDLmain				
Image: Signal		Profiler settings	SDL.dll SDL image				
Batch builds Debugger settings Add Edit Delete Clear Copy all to			user32				
Batch builds Debugger settings Add Edit Delete Clear Copy all to		and a	winnm drawid				
Batch builds Debugger settings Add Edit Delete Clear Copy all to							
Batch builds							
Debugger settings		Batch builds					
Debugger settings							
Debugger settings							
Debugger settings							
Debugger settings							
Copy all to		Debugger settings	Add Edit Delete Clear				
		bebugger betangb		-			
OK Cancel							

- e. Go to tab Directories, to add location storing header files and libraries files. First, select tab Compiler, add:
 - i. C:\SDLMingW\include, location of header files of SDL libraries.
 - ii. C:\MinGW\include, location of standard header files of MinGW such as string.h, stdio.h...

Second, select tab Linker, add C:\SDLMingW\lib and C:\MinGW\lib

f. Go to tab Program, and configure as bellow:

С	Compiler Linker Directories Programs Custom variables Other					
[Compiler's installation dir	ectory				
	C: WinGW	Auto-detect				
	NOTE: All programs belo or in any of the "Additio	ow, must exist either in the "bin" sub-directory of this path				
	or an any or and shakado					
	Program Files Additiona	al Paths				
	C compiler:	mingw32-gcc.exe				
	C++ compiler:	mingw32-g++.exe				
	Linker for dynamic libs:	mingw32-g++.exe				
	Linker for static libs:	ar.exe				
	Debugger:	gdb.exe				
	Resource compiler:	windres.exe				
	Make program:	mingw32-make.exe				
		↓				

- g. To test GNU GCC compiler profiles, lets create a SDL project, in menu, select File > New > Project then choose SDL Project icon. Follow wizard:
 - i. Enter C:\SDLMingW as SDL's location
 - ii. Select GNU GCC as compiler and click button Finish.
 - iii. Compile and you should see a simple demo SDL application "SDL_app".
 - iv. Save this project for later modification.

Projects Ruid targets	Category:	Il categories>			•	Go	SDL_app	
Files Custom	R		FLTK	GLFW		<u>C</u> ancel		
	Dynamic Link Library	Empty project	FLTK project	GLFW project				
	GLUT	F	IRELICHT	S				
	GLUT project	GTK+project	Irrlicht project	Ogre project			2	
	(DenGL	3	SDL)	Smartwin				~
	OpenGL project	QT4 project	SDL project	SmartWin project				6
	8	5	Ř			View as		
1	Static library	Win32 GUI	wxSmith	wxWidgets	-	C List		
			Why	are some wizards r	marked in red?			
1. Select a wizard 2. Select a specific 3. Press Go	type first on the wizard from the	left main window (f	ilter by categori	ies if needed)				

We can optimize a little for our Win32 SDL application by settings compiler flags in tab Compiler > Compiler Flags...

Since we use gdb.exe as debugger, we can debug our Win32 SDL as easily as in Microsoft Visual Studio. More important, you don't have to pay a penny !

- 7- Next, we configure compiler for GP2X. Make a copy of GNU GCC compiler profile settings and rename it to "*DevKitGP2X*"
 - a. Go to tab Compiler > #define GP2X
 - b. Go to tab Directories > tab Compiler, add path to location of header files C:\devkitGP2X\include, C:\devkitGP2X\lib\gcc\arm-linux\4.0.2\include, C:\devkitGP2X\include\SDL
 - Go to tab Directories > tab Linker add path to location of library files C:\devkitGP2X\lib
 - d. Go to tab Programs and configure as follow:

Compiler Linker Directories Programs Custom variables Other						
Compiler's installation directory						
C:\devkitGP2X	Auto-detect					
NOTE: All programs belo or in any of the "Addition	ow, must exist either in the "bin" sub-directory of this path anal paths"					
Program Files Addition	al Paths					
C compiler:	arm-linux-gcc.exe					
C++ compiler:	arm-linux-g++.exe					
Linker for dynamic libs:	arm-linux-g++.exe					
Linker for static libs:	ar.exe					
Debugger:	adb.exe					
Resource compiler:						
resource complet.	winares.exe					
Make program:	mingw32-make.exe					
	▲ · · · · · · · · · · · · · · · · · · ·					
	•					

- 8- After creating compiler settings for GP2XdevKit, we can modify our Code::Blocks project so it can compile to *.gpe file.
 - a. Go to menu Project > Properties. Select tab Targets and click button "Add" to create new target.
 - b. Modify extension of output file name from *.exe to *.gpe.
 - c. Turn on the check boxes, Auto-generate file name prefix and Auto-generate file name extension. Then click button Ok to save settings.

Project/targets options						
Project Targets Scripts C/C++ par	ser options					
Build targets		Selected build target	options			
Debug	Add		Console application			
GP2X	Rename	Type:	Pause when execution ends Create import library			
	Duplicate		Create .DEF exports file			
	Delete	Output filename:	GP2X\01 ope			
		ouput mename:	Auto-generate filename prefix			
			Auto-generate filename extension			
	Virtual targets	Execution working dir:	·			
	Dependencies	Objects output dir:	GP2X\obj			
	Re-order	Build target files:				
	Build options	✓ cb.bmp				
✓ main.cpp						
	Create project					
	from target	Toggle checkmarks	Selected file properties			
OK Cancel						

- d. Go to menu Project > Build Options. Set GNU GCC as compiler for Debug and Release targets. Set DevKitGP2X as compiler for GP2X.
- e. In dialog "Project build options", switch to tab Linker, paste following line to Other linker options: -*static -lSDL_gfx -lSDL_image -ljpeg -lz -lSDL_mixer -lgcc -lm -lc lexpat -lpthread -lmad -dgp2x*. We will use static link to create *.gpe file. The result *.gpe file will be big. But we cannot create *.gpe file that dynamically links to SDL libraries since DevKitGP2X use SDL and gcc which is newer than those supplied by Games Park Holdings.
- *f*. Menu Build > Select Target > GP2X and then Build. Transfer output *.gpe with cb.bmp file to GP2x and test. (*Be warned, when you run this program, you cannot exit because, there is no appropriate handling code to terminate this sample program in GP2X environment*).

4. Compile additional libraries for SDL

While devkitGP2X supplies many additional libraries such as SDL_image, SDL_ttf, SDL_mixer, so you can easily build *.gpe file for GP2X. We better to make similar environment in host PC, Windows XP so we can create and simulate our game conveniently in Windows. As we satisfy with Win32 game version, we switch target to GP2X and make *.gpe file. In this section, I explain briefly how to install (or compile if necessary) SDL_image, SDL_mixer, SDL_gfx and SDL_ttf. You can visit here for more detail http://lazyfoo.net/SDL_tutorials/lesson03/windows/codeblocks/index.php

- 1- SDL_Image: Go to <u>http://www.libsdl.org/projects/SDL_image/</u> download SDL_image-devel-1.2.5-VC6.zip file and copy *.h, *.lib and *.dll file to appropriate folder.
- 2- SDL_mixer: Go to <u>http://www.libsdl.org/projects/SDL_mixer/</u> and download SDL_mixerdevel-1.2.7-VC6.zip

- 3- SDL_gfx: Go to <u>http://www.ferzkopp.net/joomla/content/view/19/14/</u> and download this file SDL_gfx-2.0.15.tar.gz.
 - a. Go to Other builds unzip "Makefile" to source folder.
 - b. Replace "c:/dev/local" to the path to SDL library folder "C:/SDLMingW"
 - c. Replace "c:/dev/local/include/SDL" by "C:/SDLMingW/include"
 - d. Use command line, go to source folder and type "*mingw32-make –B*" to compile. The output files are SDL_gfx.dll, libSDL_gfx.a, libSDL_gfx.dll.a
- 4- SDL_ttf: Go to <u>http://www.libsdl.org/projects/SDL_ttf/</u> and download this file SDL_ttf-devel-2.0.8-VC6.zip. SDL_ttf requires FreeType 2 library. Go here to download FreeType 2. <u>http://gnuwin32.sourceforge.net/packages/freetype.htm</u>
- 5- To compile SDL Test Program Thingy version 1.51,
 - a. Go to menu Project > Build options
 - b. Select Release build target (GNU GCC compiler)
 - c. Go to tab Linker, in the list box Link Libraries, add SDL_mixer, SDL_ttf, SDL_gfx
 - d. Menu Build > Build and Run

Project build options		
SDLTest	Selected compiler	-
GP2X 1	Compiler Linker Directories Commands Custom variables "Make" commands Policy: Append target options to project options	
	Link libraries: SDL_mixer SDL_gfx 4 Add Edit Delete Clear	
	Copy all to	
	<u>QK</u> <u>C</u> ancel	

SDL Library Test	_ 🗆 🗙
SDL Test Program Thingy :) By Guyfawkes Test SDL_TTF Test SDL_Image Test SDL_Image Test SDL_Mixer WAV Test SDL_Mixer WGG Test SDL_Mixer MO3 Test SDL_Mixer MOD Test SDL_Timer 5 second test New joystick stuff Quit	

SDL Test Program Thingy is running on Windows XP OS.

Notes: Test SDL_Input, you need to use keyboard to simulate GP2X joystick and buttons.

5. Some good SDL programming tutorials

Firstly, I recommend the demo of **SDL Test Program Thingy** version 1.51 made by *Guyfawkes*. This demo works well in GP2X and Windows.

Some good SDL tutorials: <u>http://lazyfoo.net/SDL_tutorials/index.php</u> Lazy Foo's Productions <u>http://sol.gfxile.net/gp/</u> Sol Tutorials

More tutor please see here http://www.libsdl.org/tutorials.php

Book about games programming in SDL: Data Structures for Game Programmers by Ron Penton, 2002.

Special thanks to all members in http://archive.gp2x.de

Chang,