Xport 2.0 User Guide

Version 1.5, June 24, 2003

Quickstart

The steps below will guide you through the Xport installation and setup. You will need the following:

- ✓ PC running Windows 9x/Me/NT/2000/XP with a parallel port, 250 Mbytes of available hard drive space and administrator privileges.
- ✓ Xport software installation CD
- ✓ Xport 2.0
- ✓ Parallel Port Interface and 10-pin ribbon cable
- ✓ Game Boy Advance (GBA) or GBA SP

1. Insert the Xport into GBA cartridge slot. **Figures 1 and 2** show the Xport before and after it is fully engaged. **Note the orientation of the Xport. The two 34-pin user connectors are facing downward.** This applies to both the original GBA and the GBA SP.
2. Verify that the Xport is functioning by turning on the GBA. You should see the "Xport is working" text on the screen and the flashing red and green LEDs after the Nintendo logo sequence. This verifies that the Xport is working properly.
3. Insert the Xport Software CD into your PC's CD-ROM drive and run "setup.exe". This will install the Xport utilities, Cygwin, GCC, eCos, source code and examples.
4. Plug the Parallel Port Interface into your PC's parallel port.
5. With the GBA powered off, plug one end of the 10-pin ribbon cable into the Parallel Port Interface and the other end into the Xport's Cport. Refer to **Figures 3 and 4**.
6. Bring up the "Xport shell". This can either be found on the desktop or through the Start menu (Start->Programs->Xport). Change directories into helloworld_c by typing "cd examples/helloworld_c".
7. Run "make upload". This will configure the Xport logic and flash, but before it does so, it will ask you to toggle the GBA power switch. Toggling the power is necessary before any programming operation.
8. After programming, toggle the GBA power once more to run the demo. After the Nintendo logo sequence, you should see the "Hello world!" text.
9. You may modify main.c as you wish and recompile/upload by running "make upload" again.

If everything worked as described, your Xport is tested and ready for use and development. If you experienced problems, please refer to the troubleshooting section at the end of this manual.

Figure 1: Inserting Xport

Figure 2: Fully Engaged

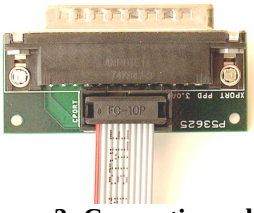

Figure 3: Connecting cable to Parallel Port Interface

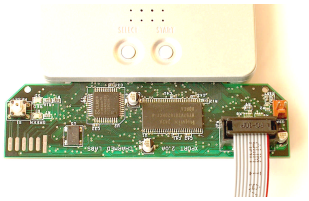

Figure 4: Connecting cable to Xport's Cport connector

Xport 2.0 Overview

Figure 5 shows the complete Xport 2.0 system including Xport and GBA.

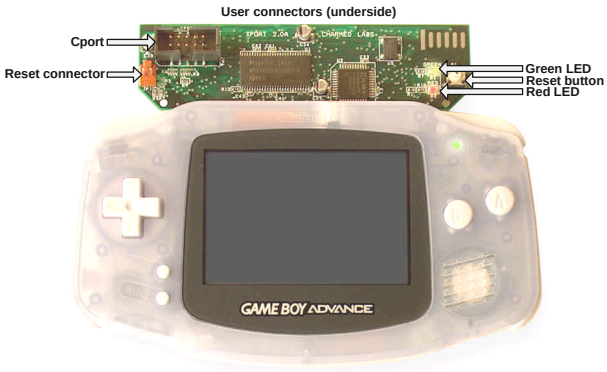

Figure 5: Xport 2.0 System

Reset connector	Provided for those who have installed the automatic reset signal – refer to the section *Automatic Reset Signal* for details.
Reset button	If the automatic reset signal is installed, the reset button will reset the GBA when depressed. This button is can also connected to the FPGA. See the section

	Connector and FPGA Signals for details.
Green LED	User-programmable LED.
Red LED	User-programmable LED.
Cport	Used for programming and communicating with the Xport. Connects to the parallel port of a PC using the Parallel Port Interface and cable.
User connectors	Used to interface to add-on circuits. Refer to section *User Connector Signals* for detailed information.

Figure 6 shows a block diagram overview of the Xport 2.0, which consists primarily of an FPGA, flash memory, optional SDRAM, and support components.

As shown in the diagram, the FPGA is the central component. This topology provides the most flexibility because the FPGA is fully programmable. The Flash device stores the code that gets executed by the GBA, and since the FPGA uses volatile RAM cells to store its logic configuration, the flash is also used to store the FPGA's logic configuration. The CPLD (Complex Programmable Logic Device) assists in the power-up configuration process that is required to program the FPGA when power is applied to the system. The optional 16 Mbytes of SDRAM can either be mapped into the GBA address space or used by the FPGA logic for applications that require data storage and retrieval.

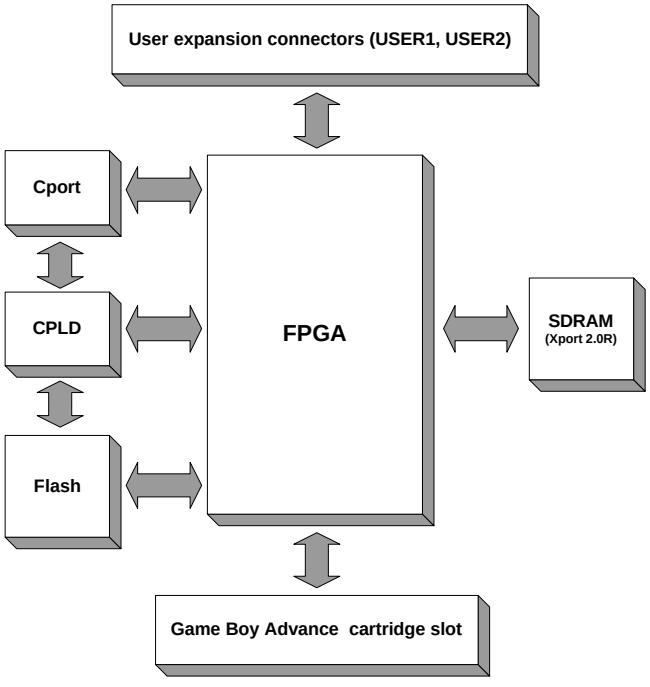

Figure 6: Xport 2.0 Block Diagram

One important difference between the GBA and the original Game Boy is the cartridge design. The GBA uses a 16-bit cartridge design as opposed to the original Game Boy, which uses an 8-bit design. Yet they both share the same physical cartridge connector. So in order to "fit" the extra data signals into the same number of cartridge connections, the GBA multiplexes the address and data signals. This custom multiplexing scheme makes it impossible to interface industry-standard memory devices to the GBA directly. Hence, one of the tasks of the FPGA is to demultiplex the address and data signals coming out of the GBA so they may be presented to the flash device. The demultiplexing is required in order for the GBA to execute the code or access data stored inside the flash.

However, the demultiplexing requires only a small fraction of the FPGA's logic. The rest of the logic is available for the user's particular application. Such applications can use the FPGA logic for interfacing to external devices, coprocessing or both. Examples of coprocessing include:

- Multiplication, division
- FIR filtering
- Image processing
- General purpose RISC processor
- Watchdog timer, real-time clock

For applications that require physical I/O there are 64 I/O signals available through the user expansion connectors USER1 and USER2. Such applications typically make use of additional hardware in the form of an "add-on" circuit. An add-on circuit can be connected to the Xport via USER1 or USER2 with ribbon cables or directly with stackable PCB connectors. Examples of I/O applications:

- General purpose I/O
- Pulse-width modulation
- Pulse timing
- Analog-to-digital or digital-to-analog conversion
- Quadrature decoding
- Peripheral interfacing (e.g. USB, PCMCIA, Compact Flash)
- Device interfacing (e.g. cameras, sensors, motors, actuators)

Using the FPGA for these applications may require writing (or modifying) a custom logic configuration, if it is not already available through our website. It is recommended that custom configurations be written in an HDL (hardware description language) such as Verilog or VHDL. Refer to the *Xport 2.0 Custom Configuration Tutorial* for detailed instructions regarding how to create your own logic configurations for the Xport. The Xport 2.0 uses the Xilinx Spartan II FPGA, which is supported by the free WebPACK software. This complete FPGA development environment

can be downloaded through the Xilinx website (www.xilinx.com).

Configuration Slots

As described earlier, the FPGA uses the flash to store its logic configuration. Upon power-up the CPLD assists the FPGA by providing it with the configuration data stored in the flash. These logic configurations are stored in 128 Kbyte "slots," which reside at the very top of flash memory. Slot 0 stores the default runtime configuration, which is loaded when the GBA is powered on. Slot 1 stores the programming configuration that is used by Xpcomm to program the flash.

Since these slots reside in flash, they can become corrupted if, for example, a user program accidentally writes into upper flash memory. If slot 0 is corrupted, it can be easily repaired by reprogramming through Xpcomm (using the –pc option). Similarly, if slot 1 is corrupted, it can be repaired by using the update operation (–u option).

Subsequent versions of Xpcomm may contain new configurations for slot 1. Running Xpcomm with the –u option will update slot 1 with the latest configuration. See the section entitled *Using Xpcomm* for more details.

Xport Software Distribution

The Xport is shipped with a CD containing the Xport Software Distribution. Installing the software is accomplished by inserting the CD and running "setup.exe".

The Xport software is installed in C:\xport by default. Contained in this directory are the following subdirectories:

bin	Xport utilities
doc	Xport documentation
examples	ready-to-run Xport examples
devkitadv	GCC toolchain for ARM Thumb targets
include	include files
lib	library files
logic	Xport Logic Library files
share	configuration files (Insight)
src	source code

The ready-to-run examples contained in the examples directory can be compiled and uploaded easily by using the Xport Shell. The Xport Shell can be launched from the desktop or the Start menu. Running "make upload" from within the example directory will compile (if necessary) and upload the example code into the Xport. The example can then be executed after power cycling the GBA.

Using Xpcomm

The Xpcomm utility is used to program the flash and FPGA configuration as well as communicate with the GBA through the Cport.

To program the FPGA configuration, Xpcomm accepts bitstream files (.bit) generated by synthesis software such as the Xilinx ISE 5 or WebPACK. To program the flash, Xpcomm accepts program files as both raw binary (.bin) and S-records (.srec). These files are usually generated by C/C++ compilers such as GCC or ARM SDT.

When invoking Xpcomm with a file argument, it will detect the file type (bitstream, raw binary, or S-record) and program the Xport appropriately. For example,

```
xpcomm redgreen.bit
```

will program the FPGA configuration with the contents redgreen.bit (bitstream), and

```
xpcomm redgreen.bin
```

will program the flash with the contents of the raw binary file redgreen.bin. Similarly,

```
xpcomm redgreen.bit redgreen.bin
```

will perform both programming operations. Most users should find that using Xpcomm in this way is sufficient for almost all programming tasks.

Alternatively, Xpcomm can be instructed to perform specific operations. **Table 1** below details the complete list of supported operations:

Table 1: Xpcomm Operations

-pf <*program file*>	Program the flash with the specified program file (binary or S-record). Note, when programming the flash, Xpcomm performs check-summing to ensure data integrity.
-vf <*program file*>	Verify the flash with the specified program file (binary or S-record).
-pvf <*program file*>	Program and verify the flash with the specified program file (binary or S-record).
-rf <*dump file*> <*length*>	Read and dump the contents of the flash (*length* number of words) to the specified dump file.
-pc <*bitstream file/directory*>	Program the FPGA configuration with the specified bitstream file. If a directory is specified, Xpcomm will automatically choose the bitstream in the directory that is compatible with your Xport's FPGA.
-vc <*bitstream file/directory*>	Verify the FPGA configuration with the specified bitstream file. If a directory is specified, Xpcomm will automatically choose the bitstream in the directory that is compatible with your Xport's FPGA.
-pvc <*bitstream file/directory*>	Program and verify the FPGA configuration with the specified bitstream file. If a directory is specified, Xpcomm will automatically choose the bitstream in the directory that is compatible with your Xport's FPGA.
-c <*bitstream file/directory*>	Program the FPGA with the specified bitstream file using external slave mode. This does not modify the contents of the non-volatile FPGA configuration stored in flash. The configuration will be lost upon the next power cycle. If a directory is specified, Xpcomm will automatically choose the bitstream in the directory that is compatible with your Xport's FPGA.
-i	Retrieve and display Xport information, update FPGA configuration for programming the flash if necessary.
-u	Force an update of the FPGA configuration for programming the flash.
-reset	Reset the GBA and execute the program contained in flash. This will only work if the automatic reset signal is installed.
-startup	Reset the parallel port state so that the slot 0 logic configuration is used.
-console	Run a tty console through the Cport.
-rpc	Run the Remote Procedure Call server through the Cport.
-execute	Same as reset.
-pause <*milliseconds*>	Pause between operations for the specified number of milliseconds.
-loop <*iterations*>	Specifies the number of times to repeat the command sequence.
-time	Output the elapsed time after all operations are completed.
-version	Print the version of Xpcomm.

Operations take place in the order specified. For example,

```
xpcomm –vc redgreen.bit –rf out.bin 0x200000 –pf redgreen.bin
```

will verify the FPGA configuration with redgreen.bit, followed by dumping the first 2 megawords (4 megabytes) of the flash to out.bin, followed by programming the flash with the contents of redgreen.bin.

Xpcomm also supports the properties detailed below in **Table 2**.

Table 2: Xpcomm Properties

-resetauto	Specifies to Xpcomm that the automatic reset signal is installed.
-portaddr <*address*>	(Only applies to Windows 9x and Me platforms.) Specifies the parallel port I/O address. The default address is 0x378.
-portnum <value>	(Only applies to Windows NT, 2000 and XP platforms.) Specifies the parallel port device number (LPT number). By default, LPT1 is used unless otherwise specified. To specify LPT2, for example, -portnum 2 would be added to the commandline.
-debug <*debug level*>	Specifies the relative number of debugging messages that are sent to the console. There are three supported levels: (0) outputs only critical messages, (1) (default) outputs messages considered to be important to the typical user, and (2) outputs verbose information that may be useful for debugging.
-delay <*delay value*>	Specifies the amount of delay that should be used when reading or writing to the parallel port. It is recommended that the delay value not exceed 500.
-readdelay <*delay value*>	Similar to delay, but applies only to read operations from the parallel port.
-writedelay <*delay value*>	Similar to delay, but applies only to write operations to the parallel port.

Properties can appear in any order in the commandline and apply to all specified operations. For example,

```
xpcomm gpio.bit –portaddr 0x3bc gpio.bin -debug 2
```

will communicate with the parallel port through I/O location 0x3bc and output verbose debugging messages for all operations.
It is also possible to specify default properties by setting the XPCOMM_ARGS environment variable. This variable is defined in /etc/xpprofile from within Cygwin (typically c:\cygwin\etc\xpprofile). For example, adding the following line to xpprofile:

```
export  XPCOMM_ARGS= -portaddr 0x3bc –resetauto
```

will modify the parallel port I/O location and specify automatic reset for all subsequent invocations of Xpcomm.

Electrical Specifications

Table 3: DC Characteristics

Description	Value
Maximum supply voltage	3.3V[3]
Minimum supply voltage	3.0V[4]
Maximum supply current	200mA[5]
Nominal supply current	100mA[6]
Current source per output[1]	24mA
Current sink per output[1]	-24mA
Current leakage per input[2]	±10µA
High-level output voltage[1]	3.3V max
Low-level output voltage[1]	0.0V min
High-level input voltage	1.7V min, 5.5V max[7]
Low-level input voltage	0.0V min, 1.0V max
Input capacitance per input[2]	20pF

[1]For programmable I/O pins configured for output.
[2]For programmable I/O pins configured for input.
[3]Typical voltage at Vcc pin of JP1 and JP2 with less than 50mA current draw from add-on circuit.
[4]Voltage at Vcc pin of JP1 and JP2 with 200mA of total current draw from add-on circuit.
[5]Maximum total current available to add-on circuit assuming GBA is using two AA batteries for power.
[6]It is recommended that total average current draw from add-on circuit not exceed 100mA.
[7]All inputs are 5V tolerant.

Automatic Reset Signal

The Xport has provision for an automatic reset signal. When this signal is installed, the PC can reset the GBA automatically, and the user can reset the GBA via the reset button (see **Figure 5**). Installing the reset signal can simplify development by preventing the user from having to manually reset the GBA by toggling the power switch before each programming operation.

Installing this signal requires modifying the GBA. Instructions can be found on Jeff Frohwein's GBA developer's site:

http://www.devrs.com/gba/files/gbadevfaqs.php#ResetButton

Once installed, crimp a connector pin (supplied with the Xport) to the newly installed reset wire and insert the pin into the connector housing. Be sure to follow the correct pin orientation and location depicted in **Figure 7**. The connector housing can then be plugged into the 2-pin connector (JP1) on the Xport. The newly installed reset signal can be tested by depressing the reset button while the GBA is powered on. After releasing the button the GBA should reset immediately.

Figure 7: Correct Reset Pin Orientation and Location During Insertion into the Connector Housing

In order for the Xpcomm utility to be made aware of the reset signal, you must add the text "-resetauto" to the XPCOMM_ARGS environment variable. See the section entitled *Using Xpcomm.*

Troubleshooting Guide

- **GBA powers up without Nintendo logo and program does not run.**
  This is typically caused by an incorrect parallel port state causing slot 1 instead of slot 0 logic configuration to be loaded into the FPGA upon power-up. Running

  `xpcomm –startup`

  will reset the parallel port such that slot 0 is selected. Note, the xpprofile script runs Xpcomm with the –startup option when the Xport shell is run.

- **When running Xpcomm, it prints the message "Failed to initialize" and exits.**
  This is usually caused by another copy of Xpcomm running. Be sure to check the process list and kill any Xpcomm processes. Checking the process list within Cygwin is accomplished by running "`ps`" followed by "`kill` ***`pid`***" where *pid* is the process ID of the Xpcomm process.

  This can also be caused by another program or device locking the parallel port.

- **Programming operations or Cport communications fail.**
  Programming and communication problems can be caused by an incompatible parallel port configuration on your PC.

  Please review the information below as it applies to your system.
  - Xpcomm is not compatible with EPP (enhanced parallel port) mode. It may be necessary to reboot your PC and change the parallel port mode to either Bidirectional or ECP from the BIOS setup screen. Bidirectional is sometimes referred to as PS2 mode from BIOS.

  If you are running Windows 9x or Me:
  - Xpcomm will not request exclusive access to the parallel port. Thus, if another program or driver is trying the access the parallel port, problems will result. Be sure that no driver or application is accessing the parallel port already.
  - Xpcomm will assume the parallel port is located at location 0x378 unless otherwise specified. If your parallel port is located elsewhere, be sure to pass this location to Xpcomm by using the -portaddr property.

  If you are running Windows NT, 2000, or XP:
  - Xpcomm will attempt to gain exclusive access of the parallel port. If it cannot do so, it will return an error and exit. This error results when another driver or application has already gained exclusive access of the parallel port. Disabling or eliminating the contending driver or application will remedy this issue.
  - Xpcomm will use the first parallel port (LPT1) by default. If you are using a different port, pass the parallel port number to xpcomm with the –portnum property.

Contacting Us

Please send your bug reports, questions and suggestions to support@charmedlabs.com

Connector and FPGA Signals

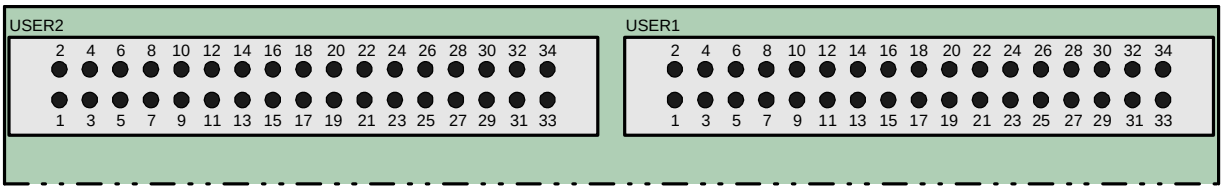

Figure 8: Connectors USER1 and USER2 (top view of connectors)

Table 4: USER1 Pinouts

Pin	Signal	Description
1	GND	0V
2	PA0	Programmable I/O
3	PA1	Programmable I/O
4	PA2	Programmable I/O
5	PA3	Programmable I/O
6	PA4	Programmable I/O
7	PA5	Programmable I/O
8	PA6	Programmable I/O
9	PA7	Programmable I/O
10	PA8	Programmable I/O
11	PA9	Programmable I/O
12	PA10	Programmable I/O
13	PA11	Programmable I/O
14	PA12	Programmable I/O
15	PA13	Programmable I/O
16	PA14	Programmable I/O
17	PA15	Programmable I/O
18	PA16	Programmable I/O
19	PA17	Programmable I/O
20	PA18	Programmable I/O
21	PA19	Programmable I/O
22	PA20	Programmable I/O
23	PA21	Programmable I/O
24	PA22	Programmable I/O
25	PA23	Programmable I/O
26	PA24	Programmable I/O
27	PA25	Programmable I/O
28	PA26	Programmable I/O
29	PA27	Programmable I/O
30	PA28	Programmable I/O
31	PA29	Programmable I/O
32	PA30	Programmable I/O
33	CLKINA	Clock or generic input
34	Vcc	3.3V

Table 5: USER2 Pinouts

Pin	Signal	Description
1	GND	0V
2	PB0	Programmable I/O
3	PB1/RData5	Programmable I/O, SDRAM DQ5
4	PB2/RAddr4	Programmable I/O, SDRAM A4
5	PB3/RAddr3	Programmable I/O, SDRAM A3
6	PB4/RAddr5	Programmable I/O, SDRAM A5
7	PB5/RAddr2	Programmable I/O, SDRAM A2
8	PB6/RAddr6	Programmable I/O, SDRAM A6
9	PB7/RAddr1	Programmable I/O, SDRAM A1
10	PB8/RAddr7	Programmable I/O, SDRAM A7
11	PB9/RAddr0	Programmable I/O, SDRAM A0
12	PB10/RAddr8	Programmable I/O, SDRAM A8
13	PB11/RAddr10	Programmable I/O, SDRAM A10
14	PB12/RAddr9	Programmable I/O, SDRAM A9
15	PB13/RBS1	Programmable I/O, SDRAM BS1
16	PB14/RAddr11	Programmable I/O, SDRAM A11
17	PB15/RBS0	Programmable I/O, SDRAM BS0
18	PB16/RCKE	Programmable I/O, SDRAM CKE
19	PB17/RData4	Programmable I/O, SDRAM DQ4
20	PB18/RUDQM	Programmable I/O, SDRAM UDQM
21	PB19/RRAS	Programmable I/O, SDRAM RAS
22	PB20/RData8	Programmable I/O, SDRAM DQ8
23	PB21/RCAS	Programmable I/O, SDRAM CAS
24	PB22/RData9	Programmable I/O, SDRAM DQ9
25	PB23/RWE	Programmable I/O, SDRAM WE
26	PB24/RData10	Programmable I/O, SDRAM DQ10
27	PB25/RLDQM	Programmable I/O, SDRAM LDQM
28	PB26/RData11	Programmable I/O, SDRAM DQ11
29	PB27/RData7	Programmable I/O, SDRAM DQ7
30	PB28/RData12	Programmable I/O, SDRAM DQ12
31	PB29/RData6	Programmable I/O, SDRAM DQ6
32	PB30/RData13	Programmable I/O, SDRAM DQ13
33	CLKINB	Clock or generic input, reset button[1]
34	Vcc	3.3V

[1]When depressed the reset button will pull CLKINB to GND through a 4.7K resistor.

Table 6: FPGA Signals

Pin number	Signal	Description
165	CartData0	GBA cartridge data/address
166	CartData1	GBA cartridge data/address
167	CartData2	CBA cartridge data/address
168	CartData3	GBA cartridge data/address
172	CartData4	GBA cartridge data/address
173	CartData5	GBA cartridge data/address
174	CartData6	GBA cartridge data/address
175	CartData7	GBA cartridge data/address
176	CartData8	GBA cartridge data/address
178	CartData9	GBA cartridge data/address
179	CartData10	GBA cartridge data/address
180	CartData11	GBA cartridge data/address
181	CartData12	GBA cartridge data/address
187	CartData13	GBA cartridge data/address
188	CartData14	GBA cartridge data/address
189	CartData15	GBA cartridge data/address
191	CartAddr0	GBA cartridge address
192	CartAddr1	GBA cartridge address
193	CartAddr2	GBA cartridge address
194	CartAddr3	GBA cartridge address
195	CartAddr4	GBA cartridge address
199	CartAddr5	GBA cartridge address
200	CartAddr6	GBA cartridge address
201	CartAddr7	GBA cartridge address
164	CartCs	GBA cartridge select
163	CartRd	GBA cartridge read
162	CartWr	GBA cartridge write
203	CartIReq	GBA interrupt request
182	CartClk	GBA clock output
150	FAddr0	Flash address
149	FAddr1	Flash address
148	FAddr2	Flash address
147	FAddr3	Flash address
141	FAddr4	Flash address
140	FAddr5	Flash address
139	FAddr6	Flash address
138	FAddr7	Flash address
136	FAddr8	Flash address
134	FAddr9	Flash address
133	FAddr10	Flash address
129	FAddr11	Flash address
127	FAddr12	Flash address
125	FAddr13	Flash address
123	FAddr14	Flash address
122	FAddr15	Flash address
121	FAddr16	Flash address
120	FAddr17	Flash address
114	FAddr18	Flash address
113	FAddr19	Flash address
112	FAddr20	Flash address
45[1], 96[2]	FAddr21	Flash address
66[3]	FAddr22	Flash address
153	FData0	Flash data
146	FData1	Flash data
142	FData2	Flash data
135	FData3	Flash data
126	FData4	Flash data
119	FData5	Flash data
115	FData6	Flash data
108	FData7	Flash data
152	Foe	Flash output enable
132	Fce	Flash enable
151	Fwe	Flash write
185	Clk	50MHz clock
204	GreenLED	Green LED control (0V=illuminated)
205	RedLED	Red LED control (0V=illuminated)
20	PA0	Programmable I/O
21	PA1	Programmable I/O
22	PA2	Programmable I/O
23	PA3	Programmable I/O
24	PA4	Programmable I/O
27	PA5	Programmable I/O
29	PA6	Programmable I/O
30	PA7	Programmable I/O
31	PA8	Programmable I/O
33	PA9	Programmable I/O
34	PA10	Programmable I/O
35	PA11	Programmable I/O
36	PA12	Programmable I/O
37	PA13	Programmable I/O
41	PA14	Programmable I/O
42	PA15	Programmable I/O
43	PA16	Programmable I/O
206	PA17	Programmable I/O
18	PA18	Programmable I/O
17	PA19	Programmable I/O
16	PA20	Programmable I/O
15	PA21	Programmable I/O
14	PA22	Programmable I/O
10	PA23	Programmable I/O
9	PA24	Programmable I/O
8	PA25	Programmable I/O
7	PA26	Programmable I/O
6	PA27	Programmable I/O
5	PA28	Programmable I/O
4	PA29	Programmable I/O
3	PA30	Programmable I/O
77	CLKINA	Clock or generic input
97	PB0	Programmable I/O
96	PB1/RData5	Programmable I/O, SDRAM DQ5
95	PB2/RAddr4	Programmable I/O, SDRAM A4

94	PB3/RAddr3	Programmable I/O, SDRAM A3
90	PB4/RAddr5	Programmable I/O, SDRAM A5
89	PB5/RAddr2	Programmable I/O, SDRAM A2
88	PB6/RAddr6	Programmable I/O, SDRAM A6
87	PB7/RAddr1	Programmable I/O, SDRAM A1
86	PB8/RAddr7	Programmable I/O, SDRAM A7
84	PB9/RAddr0	Programmable I/O, SDRAM A0
83	PB10/RAddr8	Programmable I/O, SDRAM A8
82	PB11/RAddr10	Programmable I/O, SDRAM A10
81	PB12/RAddr9	Programmable I/O, SDRAM A9
75	PB13/RBS1	Programmable I/O, SDRAM BS1
74	PB14/RAddr11	Programmable I/O, SDRAM A11
73	PB15/RBS0	Programmable I/O, SDRAM BS0
71	PB16/RCKE	Programmable I/O, SDRAM CKE
70	PB17/RData4	Programmable I/O, SDRAM DQ4
69	PB18/RUDQM	Programmable I/O, SDRAM UDQM
68	PB19/RRAS	Programmable I/O, SDRAM RAS
67	PB20/RData8	Programmable I/O, SDRAM DQ8
63	PB21/RCAS	Programmable I/O, SDRAM CAS
62	PB22/RData9	Programmable I/O, SDRAM DQ9
61	PB23/RWE	Programmable I/O, SDRAM WE
60	PB24/RData10	Programmable I/O, SDRAM DQ10
59	PB25/RDQM	Programmable I/O, SDRAM LDQM
58	PB26/RData11	Programmable I/O, SDRAM DQ11
57	PB27/RData7	Programmable I/O, SDRAM DQ7
49	PB28/RData12	Programmable I/O, SDRAM DQ12
48	PB29/RData6	Programmable I/O, SDRAM DQ6
47	PB30/RData13	Programmable I/O, SDRAM DQ13
80	CLKINB	Clock or generic input, reset button
162	RData0	SDRAM DQ0
161	RData1	SDRAM DQ1
107	RData2	SDRAM DQ2
154	RData3	SDRAM DQ3
46	RData14	SDRAM DQ14
44	RData15	SDRAM DQ15
45	RCS	SDRAM CS
110	CPDir	Cport data direction
111	CPStrobe	Cport data strobe
99	CPReset	Cport reset
98	CPReady	Cport data ready
109	CPData0	Cport data
102	CPData1	Cport data
101	CPData2	Cport data
100	CPData3	Cport data

[1]Applies if JP10 is installed
[2]Applies if JP8 is installed
[3]Applies if JP9 is installed