# Xport 2.0 General Purpose I/O (GPIO)

Version 2.0, May 7, 2003 Rich LeGrand (rich@charmedlabs.com)

**Summary**

This application note explains how to use the general purpose I/O (GPIO) bitstream with the Xport 2.0.

## Introduction

General purpose I/O (GPIO) is perhaps the simplest way to interface to the outside world. The GPIO bitstream allows you to configure each I/O signal or the Xport individually as either digital input or digital output and then use the signals to read or write digital information.

## Usage

Each I/O signal has two control bits: a direction bit and a data bit. Setting the direction bit to 1 configures the corresponding I/O signal as an output. Setting the direction bit to 0 configures the I/O signal as an input. The data bit reflects the logic state of the corresponding I/O signal regardless of whether it is configured as an input or output. For convenience, these bits are grouped into 16-bit registers. The direction bits collectively form the "data direction registers" (DDRs) and the data bits form the "data registers" (DRs). **Table 1** below details these registers and their I/O signal mapping.

**Table 1: GPIO Register Mapping**

| | | Register contents (individual bits shown) | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | Address | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DDR0 | 0x9ffc400 | PA15 | PA14 | PA13 | PA12 | PA11 | PA10 | PA9 | PA8 | PA7 | PA6 | PA5 | PA4 | PA3 | PA2 | PA1 | PA0 |
| DDR1 | 0x9ffc402 | 0 | PA30 | PA29 | PA28 | PA27 | PA26 | PA25 | PA24 | PA23 | PA22 | PA21 | PA20 | PA19 | PA18 | PA17 | PA16 |
| DDR2 | 0x9ffc404 | PB15 | PB14 | PB13 | PB12 | PB11 | PB10 | PB9 | PB8 | PB7 | PB6 | PB5 | PB4 | PB3 | PB2 | PB1 | PB0 |
| DDR3 | 0x9ffc406 | 0 | PB30 | PB29 | PB28 | PB27 | PB26 | PB25 | PB24 | PB23 | PB22 | PB21 | PB20 | PB19 | PB18 | PB17 | PB16 |
| DR0 | 0x9ffc408 | PA15 | PA14 | PA13 | PA12 | PA11 | PA10 | PA9 | PA8 | PA7 | PA6 | PA5 | PA4 | PA3 | PA2 | PA1 | PA0 |
| DR1 | 0x9ffc40a | 0 | PA30 | PA29 | PA28 | PA27 | PA26 | PA25 | PA24 | PA23 | PA22 | PA21 | PA20 | PA19 | PA18 | PA17 | PA16 |
| DR2 | 0x9ffc40c | PB15 | PB14 | PB13 | PB12 | PB11 | PB10 | PB9 | PB8 | PB7 | PB6 | PB5 | PB4 | PB3 | PB2 | PB1 | PB0 |
| DR3 | 0x9ffc40e | 0 | PB30 | PB29 | PB28 | PB27 | PB26 | PB25 | PB24 | PB23 | PB22 | PB21 | PB20 | PB19 | PB18 | PB17 | PB16 |

Where Pn = the nth I/O signal – see the *Connector Pinouts* section in the Xport User's Manual.

For example, to set I/O signals PA0 through PA7 as output and PA8 through PA15 as input, set DDR0 as follows:

```
*((volatile unsigned short *)0x9ffc400) = 0x00ff;
```

Setting PA0 to PA3 as logic high and PA4 through PA7 as logic low, set DR0 as follows:

```
*((volatile unsigned short *)0x9ffc408) = 0x000f;
```

To read the state of PA8 through PA15, read DR0 as follows:

```
unsigned short val = *((volatile unsigned short *)0x9ffc408);   // read
val >>= 8;   // shift down PA8 through PA15 for convenience
```

Reading data bits that are configured as output should return the previously assigned value, as illustrated below:

```
*((volatile unsigned short *)0x9ffc400) = 0xffff;   // set PA0 through PA15 as output
*((volatile unsigned short *)0x9ffc408) = 0xabcd;
if (*((volatile unsigned short *)0x9ffc408)!=0xabcd)
   printf("Error: this should not happen\n");
```

# Electrical Specifications

**Table 2: DC Characteristics**

| Description | Value |
|---|---|
| Current source per output | 12mA |
| Current sink per output | -12mA |
| High-level output voltage | 3.3V max |
| Low-level output voltage | 0.0V min |
| Current leakage per input | ±10µA |
| High-level input voltage | 1.7V min, 5.5V max |
| Low-level input voltage | 0.0V min, 1.0V max |
| Input capacitance per input | 20pF |

Note: as detailed, all inputs are 5V tolerant