Using the Bank Switching XDATA Location APNT_130

Author: Andrew Ayre (aayre@keil.com), Keil Software, Inc.
☎ 972-735-8052

OVERVIEW

When you use an XDATA memory location to perform the bank switching in your 8051 target hardware, it is not possible for the unused bits of the location to be used as general purpose I/O. This is because all eight bits are written when performing the bank switching, as it is not possible to selectively write to particular bits in external memory. The result is corruption of data on any unused bits.

It may be desirable to utilize the unused bits of the memory location in order to reduce the amount of hardware required and thereby reduce product size and cost.

This application note describes a modified **L51_BANK.A51** file that allows the unused bits of an XDATA memory location to be used for general purpose I/O. In addition, C code examples are given to show how the location can be safely read from and written to in any bank without causing problems for the bank switching hardware.

CONFIGURING THE BANK SWITCHING

The modified **L51_BANK.A51** file is available from the Keil web site at http://www.keil.com/download/c51/apnt_130.zip.

Simply replace your existing L51_BANK file with this modified version. The default configuration of the file is such that when assembled it produces identical code to the old **L51_BANK.A51**. In order to enable the new functionality the **?B_XDATAIO** variable must be set to 1 (in addition to setting **?B_MODE** to 1).

The **?B_FIRSTBIT** variable allows you to set where the Least Significant Bit (LSB) of the bank switching signals is placed. For example, if you set **?B_FIRSTBIT** to 0 and set the maximum number of banks to 4 then bits 0 and 1 would be used for bank switching. Bits 2 to 7 could then be used for general purpose I/O.

If you set **?B_FIRSTBIT** to 3 and set the maximum number of banks to 16 then bits 3 to 6 would be used for bank switching. Bits 0 to 2 and bit 7 could then be used for I/O.

Note that the bits used for bank switching must be consecutively ordered. It is not possible to use just bits 1 and 3 for bank switching, for example.

Other variables that may required changing are **?B_XDATAPORT** which sets the address of the XDATA memory location, **?B_RTX** which indicates if RTX51 Full is being used or not and **?B_NBANKS** which sets the maximum number of banks.

READING FROM AND WRITING TO THE XDATA LOCATION

Assumptions

The code examples in this section make several assumptions. Modify your code accordingly for your particular hardware arrangement.

- The maximum number of banks (**?B_NBANKS**) is 16, therefore four bits are used for bank switching.
- The first bit (**?B_FIRSTBIT**) is bit 0, therefore bits 0 to 3 are used for bank switching.
- The address of the XDATA memory location (**?B_XDATAPORT**) is FFFFH.

Defining a Variable

In order to read and write to the location a variable has to be defined. The following code will define a variable called **bank** at FFFFH:

```
xdata unsigned char bank _at_ 0xFFFF;
```

To access the variable in other C files use the extern keyword. For example:

```
extern xdata unsigned char bank;
```

Reading the XDATA Location

The following code reads the location and masks off the bits used for bank switching. The result can then be used to determine which bits were set and unset:

```
unsigned char result;

result = bank & 0x0F;
```

Writing the XDATA Location

The folllowing code sets bits 4 and 7

```
bank = (bank & 0x0F) | 0x90;
```

The location is read, the bank switching bits are masked off, bits 4 and 7 are set and the result is written back.

The following code clears bits 4 and 7:

```
bank = bank & 0x6F;
```

NOTE
It is vital when reading, modifying, and writing to the XDATA location that the bank switching bits are NOT modified. This will cause your hardware to switch banks!

INTERRUPTS

It is vital that the I/O pins at the xdata location are not modified during a bank switch. The result will be corruption of the logic levels on those pins. In order to ensure this will not happen, interrupts are disabled when performing a bank switch and **?B_XDATAIO** = 1.

TIME COST

In order to add this extra functionality, additional assembly instructions were added to the bank switching code. The result is an increase in the time required to switch banks, and could adversely affect the performance of an application. Some of this time may be recouped if the number of bank switches is optimized. Possible ways of achieving this are to move functions to the bank they are most frequently called from, or duplicating the most used functions in the banks they are called from.

The following table gives the time cost per bank switch in processor cycles. The test conditions were:

?B_MODE = 1
?B_NBANKS = 16

	?B_RTX = 0			**?B_RTX** = 1		
	?B_XDATAIO = 0 (cycles)	**?B_XDATAIO** = 1 (cycles)	% increase	**?B_XDATAIO** = 0 (cycles)	**?B_XDATAIO** = 1 (cycles)	% increase
Function Call	34	47	38.2%	42	47	11.9%
Function Return	14	26	85.7%	21	26	23.8%

Copyright © 1999 Keil Software, Inc. All rights reserved.

In the USA:
Keil Software, Inc.
16990 Dallas Parkway, Suite 120
Dallas, TX 75248-1903
USA

Sales: 800-348-8051
Phone: 972-735-8052
FAX: 972-735-8055

E-mail: sales.us@keil.com
support.us@keil.com

Internet: http://www.keil.com/

In Europe:
Keil Elektronik GmbH
Bretonischer Ring 15
D-85630 Grasbrunn b. Munchen
Germany

Phone: (49) (089) 45 60 40 - 0
FAX: (49) (089) 46 81 62

E-mail: sales.intl@keil.com
support.intl@keil.com

LIMITATIONS

Because R1 is corrupted in the process of switching banks with ?B_XDATAIO = 1 a generic pointer may not be passed in registers to a function in another code bank. The reason is because the LSB is stored in R1.

In order to avoid this problem you may specifying the NOREGPARAMS for the function in question. Please refer to page 49 in the C51 Compiler User's Guide (01.97) for more information. Alternatively ensure that calling the function does not result in a bank switch.