DALLAS
SEMICONDUCTOR

Application Note 92
DS5002FP Memory Expansion Techniques

OVERVIEW

All members of the Soft Microcontroller Family are designed to directly address up to 64KB of program and data memory. Occasionally, however, an application will require more memory than is available through the 64KB memory map. The Soft Microcontroller Family includes many features which make it easy to address program and/or data memory greater than 64KB. Bit–addressable I/O ports allow single instruction modification of control lines, which can be used to bank switch or page between multiple memory devices. This application note discusses the expansion of both program and data memory. It begins with an introduction to bank switching and software support techniques.

BANK SWITCHING THEORY

Expanded memory access beyond 64KB is done through bank switching. This technique uses one or more general purpose I/O lines as decode lines to address more memory. If a single large–capacity memory device is used, the additional signals can be used directly as address lines. If several smaller capacity memory devices are used, the signals can be used as chip selects. The basic unit of memory switched by the decode logic is called a bank or page. For example, if an I/O line was used to switch between two 64KB EPROMs, the memory would consist of two 64KB banks.

One difficulty in implementing a bank switching scheme is the placement of the interrupt vector table. During most of the device operation, software can perform an orderly switch between banks. When an interrupt occurs, however, the device will immediately jump to the appropriate vector address, below 0030h. The software has no control over the bank configuration at this point, and the device will attempt to jump to the low end of the current bank to seek the vector table. This means that the interrupt vector table must be duplicated in each 64KB bank, so it will be available regardless of the current memory configuration. Additional space may be required if duplication of interrupt service routines is desired on each bank.

Program efficiency is maximized when bank switching is kept to a minimum, i.e., executing straight runs of code. Code efficiency will be improved if the interrupt routines (not just the vectors) are small enough to be duplicated on each bank as well. Data tables or strings accessed with MOVC instructions should be located on the same bank as the instruction.

DESIGNING A BANK SWITCHING SCHEME

The approach described in this application note selects a new memory bank without modifying the program counter. This means that the starting location in the new bank will be relative to the location of the bank switch routine in the “old” bank. Consequently, instruction location between banks is critical. Table 1 shows the timing relationship between a MOV P1, #data instruction and the switching of the bank select signal. In this example, the MOV instruction is located at location 1000h, and the first instruction on the new bank is located at 1004h. The port pin which controls the bank selection will change during the last cycle of the MOV instruction. During this cycle, the DS5002FP will already be prefetching the NOP instruction at location 1003h from the same bank which contains the MOV instruction. The first fetch from the new bank will be at the address following the NOP, 1004h.

There are other ways of modifying port pins, and this scheme will work with 2 cycle instructions such as MOV *direct*, *direct,* and 1 cycle instructions such as SETB *bit*. The only requirement on the bank switching instructions is that the first instruction on the new bank must be at the address following the NOP as described above.

Current Bank		New Bank	
Address	Instruction	Address	Instruction
1000h	MOV P1, #01h	1000h	??
1001h	(second byte)	1001h	??
1002h	(third byte)	1002h	??
1003h	NOP	1003h	NOP
1004h	??	1004h	[First instr.]

For bank switching to work, it is necessary to fragment the code into banks and provide a means for the software to switch between banks. The software techniques presented here show how to switch program banks "on the fly." This approach causes program execution to jump directly from one bank to another without modifying the program counter. One must exercise caution so that the bank switch will occur at a location that corresponds to the start of the next instruction in the next bank. Failure correctly align the instructions may cause the next opcode fetch to occur in the middle of a multi–byte instruction, resulting in loss of program control.

Immediately following a reset, all port pins will go to a logic one state. This means that upon a hardware reset the device will begin operating from the bank which is selected when the port pins are high. Make sure that this bank contains the correct startup code to properly initialize your application.

Many compilers and linkers directly support bank switching, and many of them include library functions for bank switching. The documentation accompanying your compiler will provide information concerning its expanded memory support.

BANK LOGIC SELECTION Table 1

P1.0	P1.1	P1.2	P1.3	Memory Selected
0	1	1	1	Program Bank 0
1	1	1	1	Program Bank 1 (default)
0	0	0	1	Program Bank 2
1	0	0	1	Program Bank 3
0	0	1	0	Program Bank 4
1	0	1	0	Program Bank 5
0	X	X	X	Data Bank 0
1	X	X	X	Data Bank 1

PROGRAMMING

The serial bootloader is designed to program one 64 KB bank of memory at a time. Multiple banks can be programmed by using the serial bootloader software to manipulate the general-purpose port pins which control bank switching. In terminal mode, this is done via the "P" command, and via the "PUT_PORT" command using the K2.exe software available to load the DS5002FP using the DS5000TK Evaluation Kit. To properly load the program memory, the MSL bit must be cleared via the bootloader. This is done by the K2.EXE software with the RANGE 128 command or by clearing the MSL bit in terminal mode. To initialize data memory, the MSL bit should be set before doing a load operation. The procedure to load multiple banks is as follows:

1. Start the bootstrap loader.
2. Make sure the memory configuration is correct.
3. Modify the appropriate port pins to select the appropriate 64KB bank.
4. Load the software for that bank.
5. Exit the bootstrap loader.

INTERFACING TO BATTERY–BACKED SIGNALS

The memory expansion method described in this paper uses port pins in a simple fashion as bank selection controls. Please note that the port pins are not battery backed, and as such special care must be used when interfacing non–battery backed signals to battery backed circuits. This is especially important when external logic such as NAND and NOR gates are used in the address or chip select decode circuitry. The low leakage current of 74HCXX logic makes it appropriate for use in battery–backed circuits, as long as all gates of the 74HCXX logic are driven to either a high or low state during the battery backed mode. This is because the input stage of CMOS logic can draw very high current if the input is between the V_{IL} and V_{IH} levels.

The problem of floating CMOS inputs often arises in memory expansion circuits because the DS5002FP general purpose port pins (Ports 0, 1, 2, and 3) are used as inputs to battery backed logic. During battery backed mode, the port pins are tri–stated, and will float to an indeterminate state. Analog CMOS transmission gate logic can be used to isolate battery backed and non–battery logic since the source and drain of the transmission gate do not connect to the gates in the 74HCXX logic. Controls to these transmission gates, however, must be connected to battery backed signals.

A related problem can arise if battery–backed signals are connected to unpowered logic. During data retention mode, the $\overline{CE1}$, $\overline{CE2}$, $\overline{CE3}$, and $\overline{CE4}$ signals are driven to their inactive (high) state. These signals should not be connected to unpowered logic during data retention mode, or the DS5002FP will source an excessive amount of current and shorten battery life.

BANK SWITCHING EXAMPLES

Two examples are presented on the following pages. Both utilize 128KB SRAMs. This is the most cost–efficient method of implementing large amounts of memory. In this configuration, $\overline{CE3}$ serves as the A15 signal to the SRAMs, and $\overline{CE2}$ serves as A16, which is active during data memory operations. The first example is a relatively simple scheme which employs 128KB of program and 128K of data memory. Two general purpose port pins are used to select the appropriate 64KB bank in program and data memory. The second example expands this to 192KB of program and 128KB of data memory. The more complicated memory configuration of the second example requires a 74HC4053 analog switch latch to isolate the bank select signals (P1.0–3) in data retention mode.

Figure 1 shows the memory map for the first example. The P1.0 signal is used to select one of two 64KB blocks of program memory. Note that immediately following a hardware reset, bank 1 will be selected because the P1.0 signal will be high. This bank should contain the code to be executed immediately following reset. P1.1 selects which of two 64KB blocks of data memory will be used by MOVX instructions. Figure 2 shows the interconnection scheme of the hardware.

SIMPLE BANK SWITCHING EXAMPLE MEMORY MAP Figure 1

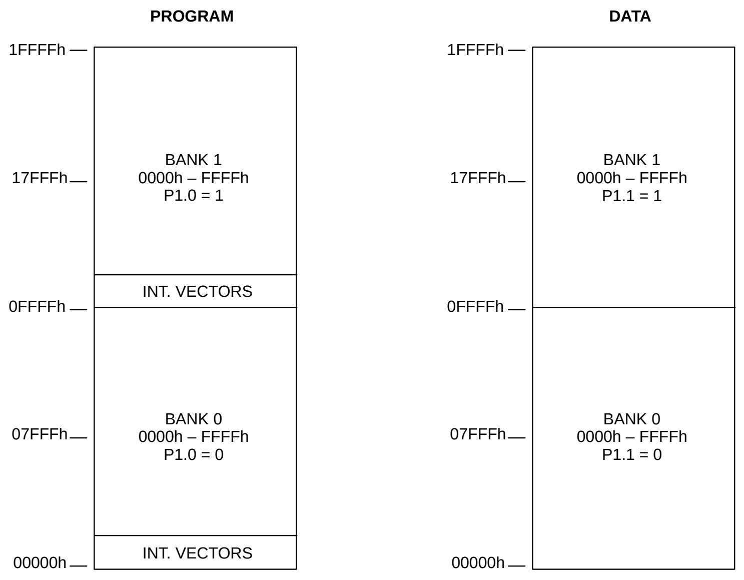
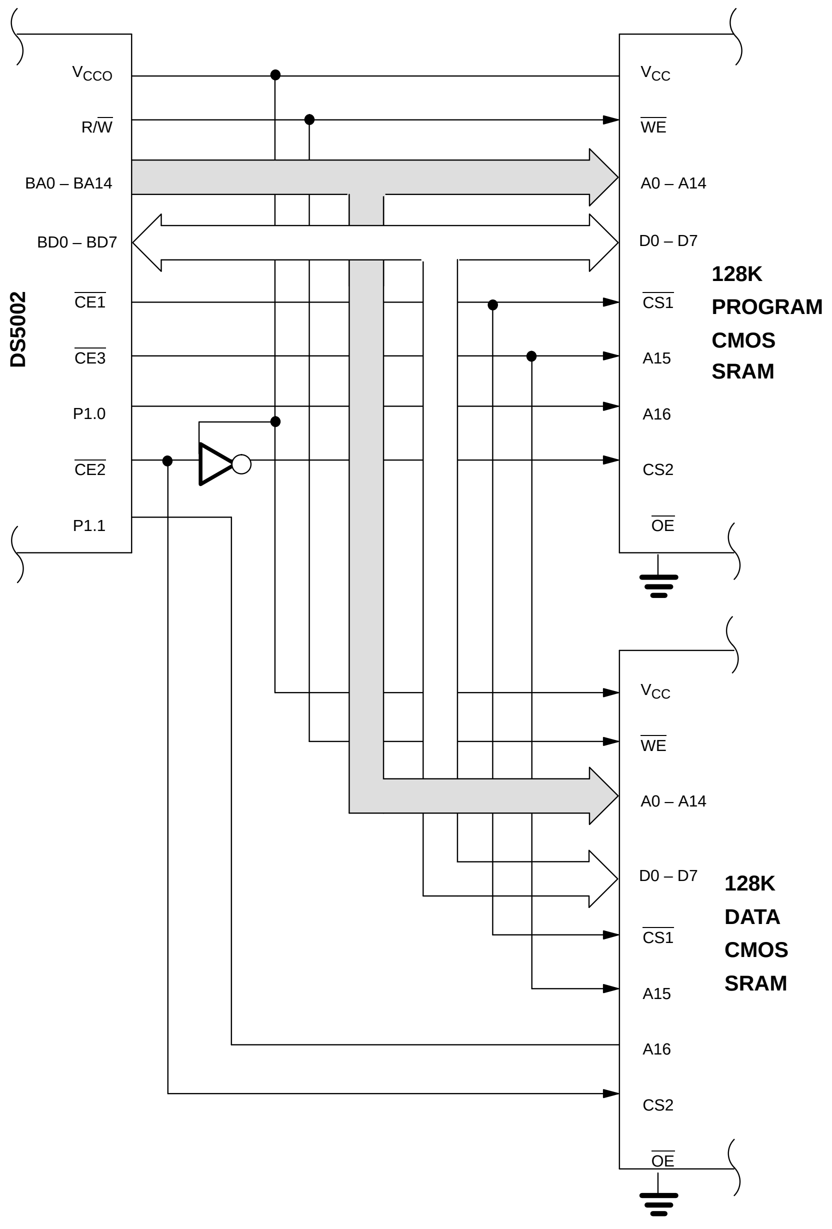


Figure 2


DS5002
V_{CCO}
R/$\overline{W}$
BA0 – BA14
BD0 – BD7
$\overline{CE1}$
$\overline{CE3}$
P1.0
$\overline{CE2}$
P1.1
V_{CC}
$\overline{WE}$
A0 – A14
D0 – D7
$\overline{CS1}$
A15
A16
CS2
$\overline{OE}$
128K PROGRAM CMOS SRAM
V_{CC}
$\overline{WE}$
A0 – A14
D0 – D7
$\overline{CS1}$
A15
A16
CS2
$\overline{OE}$
128K DATA CMOS SRAM

ADVANCED BANK SWITCHING EXAMPLE MEMORY MAP Figure 3

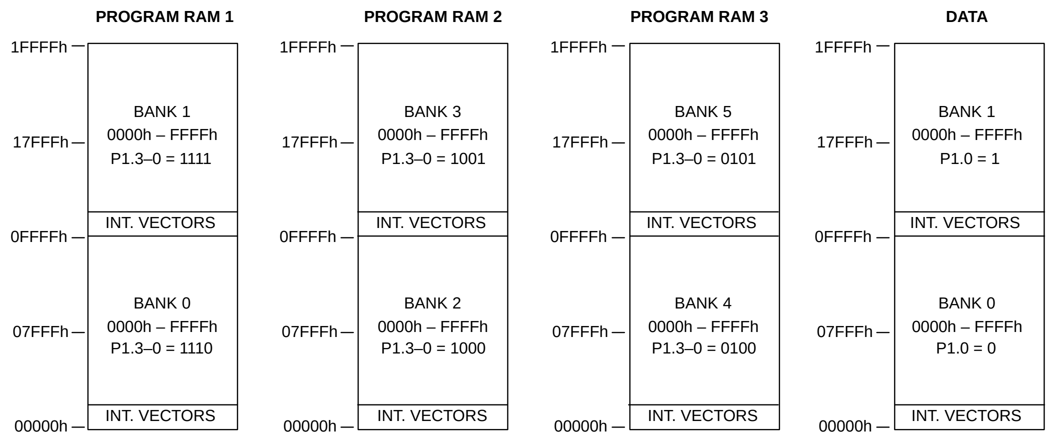
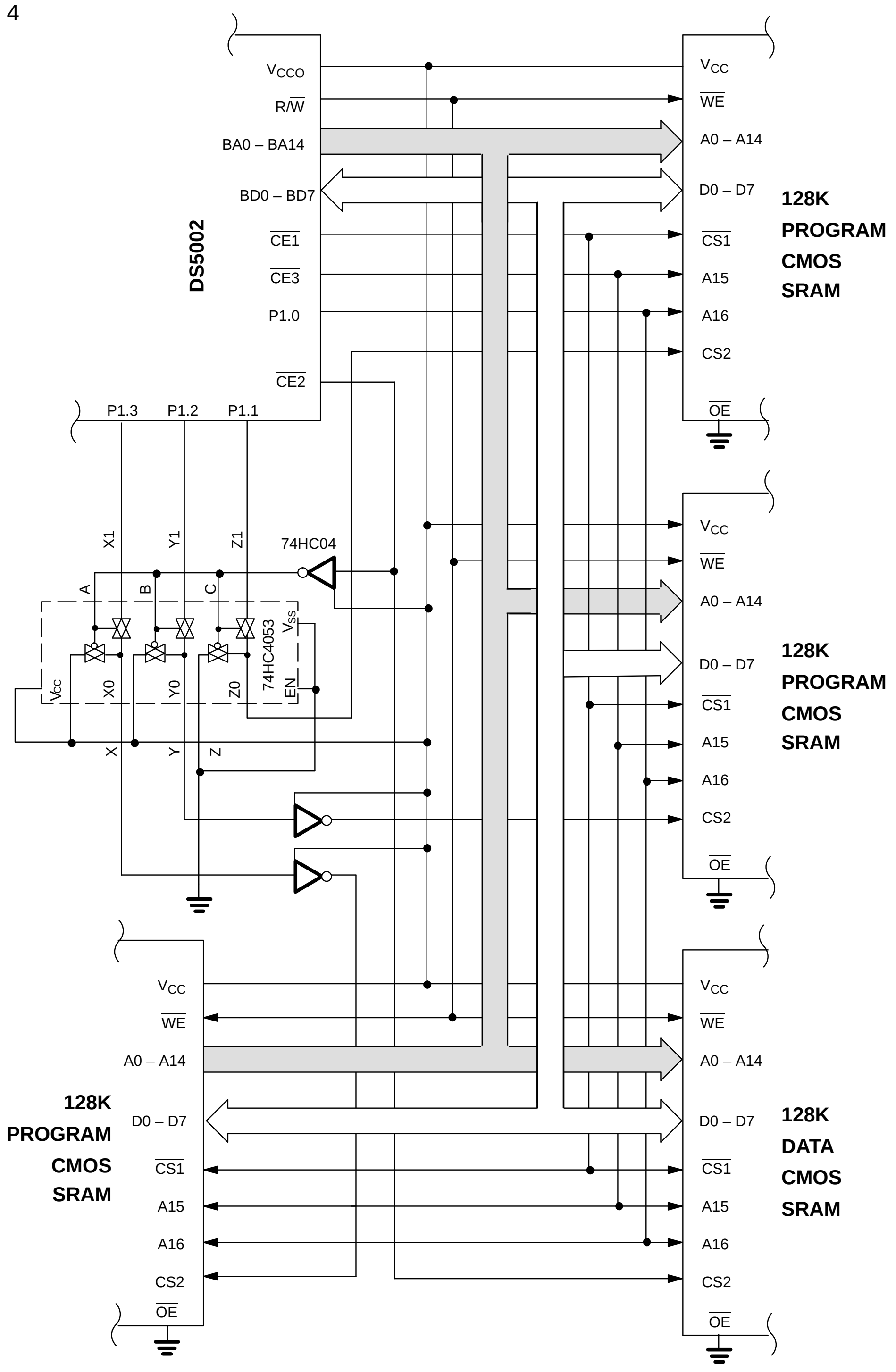


Figure 3 shows the memory map for the second example. The P1.0–3 signals are used to select one of six 64KB blocks of program memory. Note that immediately following a hardware reset, bank 1 will be selected because all port pins will be high. This bank should contain the code to be executed immediately following reset. The decode logic through the 74HC4053 is designed so that each port pin selects one device. The bank select logic is described in Table 1. P1.0 selects which of two 64KB blocks of data memory will be used by MOVX instructions. Figure 4 shows the interconnection scheme of the hardware.

Figure 4


DS5002
V_{CCO}
R/$\overline{W}$
BA0 – BA14
BD0 – BD7
$\overline{CE1}$
$\overline{CE3}$
P1.0
$\overline{CE2}$
P1.3
P1.2
P1.1
X1
Y1
Z1
74HC04
A
B
C
V_{SS}
74HC4053
$\overline{EN}$
V_{CC}
X0
Y0
Z0
X
Y
Z
V_{CC}
$\overline{WE}$
A0 – A14
D0 – D7
$\overline{CS1}$
A15
A16
CS2
$\overline{OE}$
128K PROGRAM CMOS SRAM
128K DATA CMOS SRAM