Z80 BANK-SWITCHING SCHEME AN101

1. INTRODUCTION

1. **Scope:** This Application Note gives a description of a circuit design allowing the classic Z80 microprocessor to access expanded memory, beyond the 64K bytes made readily available by its 16 address lines, A0 through A15.
2. **Z80 microprocessor:** Though it has been over 20 years since the introduction of the Z80, this family of microprocessors still finds application in new designs. This is because the Z80 is still cost-effective for many 8-bit applications; because many users have a large library of tested code for the Z80; and because the parts are readily available from several manufacturers, easing supply concerns that apply to sole-sourced processors.
3. **Applicable chips:** This Application Note applies to the classic Z80 microprocessor. It can also be applied to the newer Z84C15, which comprises a Z80 CPU, a clock generator, four Z80 CTC channels, two Z80 SIO channels, DMA, chip select signals, and glue logic in a 100-pin quad flat pack. However this external bank-switching circuitry is not necessary for members of the Z80180 family, which have a built-in MMU (memory management unit) on-chip.

2. DESIGN GOALS

1. **Program memory:** We wanted to expand program memory space to 128K bytes for our application. We needed to support in-circuit reprogramming, so we chose the AMD 29F010 flash memory device. This +5 volt part does not require a +12 volt power supply for programming. After the flash chip is initially programmed at the factory with the bootstrap loader and the current application code, it can later be reprogrammed in the field over the RS-232 serial port.
2. **RAM memory:** Our communications application needed 512K bytes of RAM, mostly to support storage of long messages. The circuit diagram here shows a fully static 512K byte SRAM. (We have also used a 512K-byte pseudostatic RAM, but that is not shown here.)
3. **Software interface:** We wanted to ease the bank-switching software burden on the programmers.
4. **Cost:** We did not want to add lots of cost. We used readily available ICs.

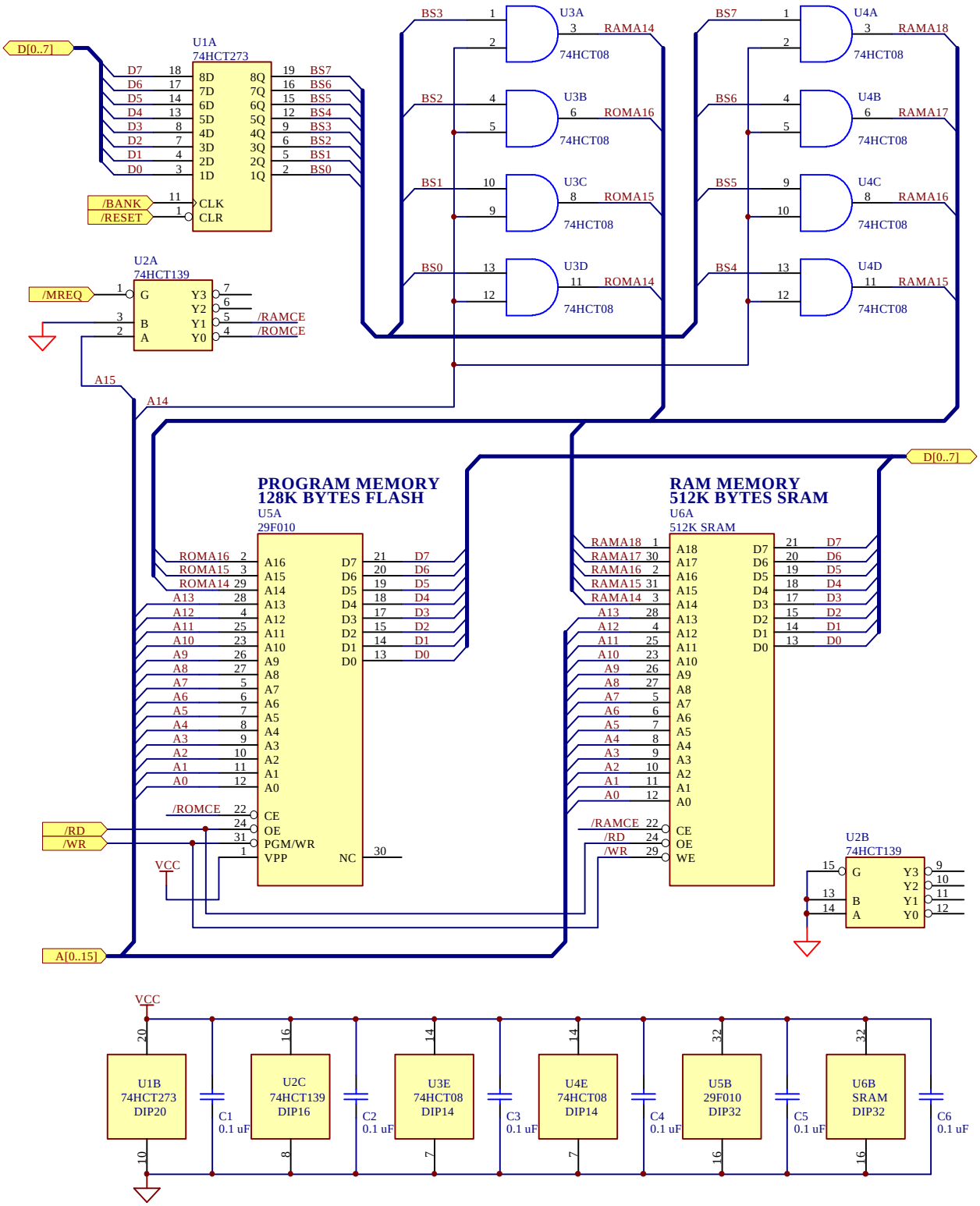
3. SCHEMATIC DIAGRAM

A circuit diagram for the bank-switching logic and the two memory ICs is shown below.

2101 OXFORD ROAD	PHONE	FAX	INTERNET
DES PLAINES, IL 60018-1919 USA	847.803.6860	847.803.6870	www.siliconengines-ltd.com

Z80 BANK SWITCHING SCHEME

MEMORY BANK CONTROL



4. CIRCUIT DESCRIPTION

1. **Chip select decoder:** One half of a 74HCT139 decoder serves to select ROM versus RAM. When A15 is high, RAM is selected; when A15 is low, ROM is selected. A15 is gated with /MREQ to assure that the memory chips are enabled only when the bus access is for memory, and only when the address lines are stable.

2. **Bank switch latch:** A 74HCT273 eight-bit latch serves as the bank switch latch. The user outputs bank switch information with a Z80 OUT instruction, which generates the output strobe /BANK. Typically a 74HCT138 or similar part (not shown in this schematic) will be used to generate this and other needed I/O strobes. At power-on, the /RESET signal resets all the bank switches to zero.

3. **Eight AND gates:** The latched bank switch output signals, BS0 through BS7, are connected to one input of each of eight 74HCT08 AND gates. The outputs of these gates are used to generate extra address lines for the ROM and RAM chips—three for the ROM, and five for the RAM.

4. **A14 gating:** Each of the eight 74HCT08 AND gates has its other input tied to Z80 address line A14. When A14 is high, the extra address lines will follow the values latched in the 74HCT273 bank switch latch. When A14 is low, the extra address lines will all be set low, regardless of the values latched in the bank switch latch. However the last values stored in the bank switch latch remain there, ready for the next access to a bank-switched memory area.

5. **PAL option:** The logic shown here could be embedded in a PAL to save PC board space.

5. MEMORY MAP

1. **Four memory areas:** Because of the way A14 is gated into the 74HCT08 inputs, the Z80 memory space has been divided into four areas, as shown below:

AREA	CHIP	ADDRESS	A15	A14	FUNCTION	MEMORY SIZE
3	RAM	C000-FFFF	1	1	BANK-SWITCHED RAM	31 BANKS, EACH 16K BYTES
2	RAM	8000-BFFF	1	0	BASE RAM	16K BYTES
1	ROM	4000-7FFF	0	1	BANK-SWITCHED ROM	7 BANKS, EACH 16K BYTES
0	ROM	0000-3FFF	0	0	BASE ROM	16K BYTES

2. **ROM space:** ROM occupies the low half of memory, from 0000 hex to 7FFF hex.

3. **RAM space**: RAM occupies higher memory, from 8000 hex to FFFF hex.

6. SOFTWARE NOTES

1. **Area 2, Base RAM:** The Base RAM area, containing addresses 8000-BFFF, will always be accessible no matter what setting has been stored in the bank-switch latch. The program stack should be assigned within this area, as well as any other temporary information that needs to be accessible from many parts of the software.

2. **Area 0, Base ROM**: The Base ROM area, containing addresses 0000-3FFF, will always be accessible regardless of the bank switch latch. The power-on start-up program must be stored at 0000 H, as usual for the Z80. The Z80 RESTART instruction vectors will automatically be located in Area 0. The Z80 interrupt vectors should also be stored in Area 0, so that they will always be accessible.

3. **Area 3, Bank-Switched RAM:** Bank-switched RAM can now be used to provide additional RAM storage. In a communications application, for example, these banks serve as expanded buffers for incoming and outgoing messages. If the software uses these 31 banks for dedicated purposes like this, it is relatively straightforward to keep the RAM bank-switching straight. As an applications example, we are now receiving a long message, and bank switch bits [BS7..BS3] are set to [01000], meaning that we are storing data into the sixteenth 16K bank in RAM. During the receive process, SIO interrupts are continually being received. Each interrupt requires use of the program stack, and probably of some temporary RAM variables as well, all of which are in the Base RAM area. The bank-switching hardware takes care of accessing these RAM resources, without having to change the RAM bank-switch setting whenever there is an interrupt.

4. **Area 1, Bank-Switched ROM:** The seven switched ROM banks in Area 1 are typically used to store specific software tasks. In our systems we usually use a multi-tasking software executive that controls a number of independent jobs. Control passes from one job to another in round-robin fashion. Each ROM bank in Area 1 contains one or more jobs. Jobs typically do not overlap from one ROM bank to another. The multi-tasking executive is housed in Area 0, Common ROM, and when the executive turns on each job is succession, it also sets the ROM bank switch bits [BS2..BS0] to the appropriate values for the job being started. Taking the communications application again as an example, let us say that a print routine handler resides in ROM bank 3, [BS2..BS0 = 011]. Every time this task is called by the executive, it checks its input pointer against its output pointer. If the input pointer has advanced beyond the output pointer—indicating that a UART receive interrupt service routine has just received another character—this task analyzes the character, does what is needed, and adjusts the pointers. If during this process, another receive interrupt comes in, the bank-switch hardware automatically switches from bank-switched ROM, to common ROM in Area 0, to handle the interrupt service routine.

5. **Bank-switch shadow:** Typically the software must maintain a location in RAM—in Area 2, Base RAM—that contains a RAM shadow of the value stored in the bank switch latch. When the multi-tasking executive needs to change the ROM bank, it reads the shadow, saves the RAM bit settings, changes the ROM bank bits, restores the RAM bit settings, and writes the new value to the bank switch latch. Likewise, when a RAM buffer handler needs to change the RAM bank, it uses the shadow RAM location to avoid disturbing the ROM bank.

☑