SAMSUNG ELECTRONICS

MEMORY EXPANSION

Application Note: KS88APN
September 1996

Application Engineering Department
LSI Division, Micom Sector

I/O PORT EXPANSION

OVERVIEW

When configured to operate in ROM-less mode, the KS88C0116 microcontroller supports I/O port expansion through an external interface at port 0 and port 1. The port 0 and port 1 pins provide the I/O channels for the external address bus and the data bus. The port 2 pins of the KS88C0116 are available as outputs for bus control signals.

In ROM-less mode, the KS88C0116 can access external data memory and program memory in addition to supporting the I/O port expansion. The expanded I/O ports described in this application note are called EP0 (expansion port 0) and EP1 (expansion port 1).

This application note describes how to expand the KS88C0116's output ports using the 74HCT563 integrated circuit and its input ports using the 74HCT541 integrated circuit. Implementing this type of I/O port expansion is simple and the resulting access times for external port operations are about the same as for on-chip ports.

You can adapt the information in this application note for use with other KS88-series microcontrollers which have an external interface.

OUTPUT PORT EXPANSION

The expanded output ports are called expansion port 0 (EP0) and expansion port 1 (EP1). Although EP0 and EP1 are located in external memory address space, they behave like normal output ports.

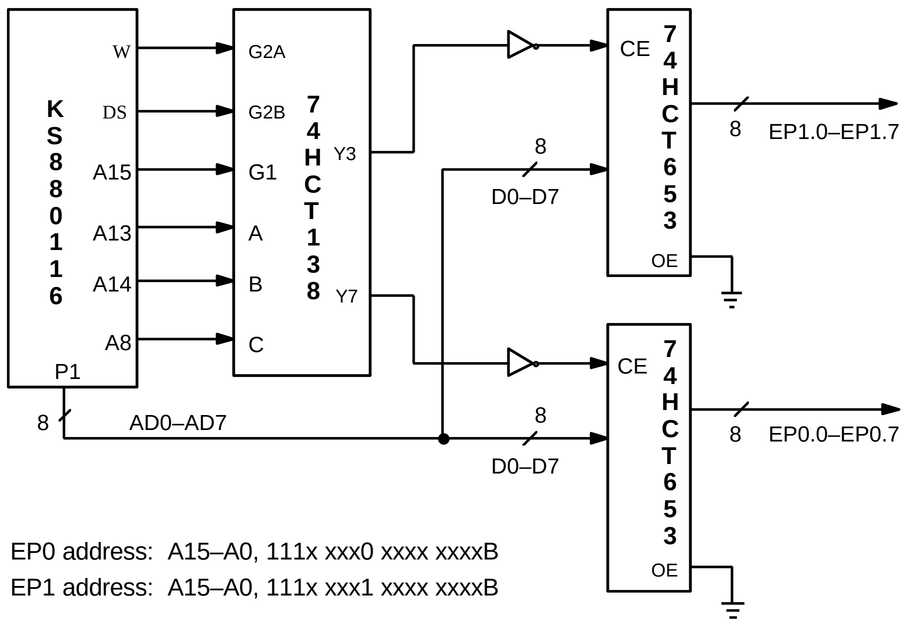
When writing data to an external memory address, you can for example specify location 0E000H or 0E100H at EP0 or EP1, respectively, as shown in the following source code:

```
EP0       equ       0E000H
EP1       equ       0E100H

          lde       EP0,r0              ;  EP0  ←  r0 (output command)
          lde       EP1,r1              ;  EP1  ←  r1 (output command)

; or

          ldw       rr2,#EP0
          lde       @rr2,r0             ;  EP0  ←  rr0 (output command)
          ldw       rr2,#EP1
          lde       @rr2,r1             ;  EP1  ←  rr1 (output command)
```

When specifying expansion port addresses, you can use a comparator logic IC such as the 74HCT138 or 74HCT688. The decoder logic map shown in Table 10-1 shows you how the 74HCT138 logic circuit decodes the addresses for the expanded output port.

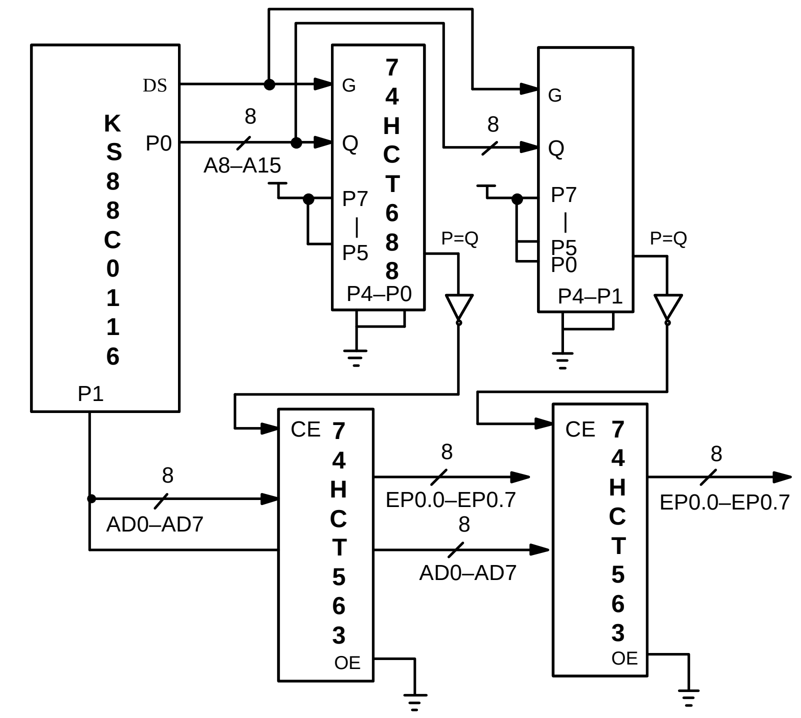
Table 10-1. Logic Map for I/O Port Expansion Using the 74HCT138 Decoder

C	B	A		Y	Address	Comment
0	0	0		0	100x xxx0 xxxx xxxB	
0	0	1		1	101x xxx0 xxxx xxxB	
0	0	1		2	110x xxx0 xxxx xxxB	
0	1	0		3	111x xxx0 xxxx xxxB	; EP0
0	1	1		4	100x xxx1 xxxx xxxB	
1	0	1		5	101x xxx1 xxxx xxxB	
1	1	0		6	110x xxx1 xxxx xxxB	
1	1	1		7	111x xxx1 xxxx xxxB	; EP1



EP0 address: A15–A0, 111x xxx0 xxxx xxxxB
EP1 address: A15–A0, 111x xxx1 xxxx xxxxB

Figure 10-1. Output Port Expansion Block Diagram



EP0 address: 1110 0000 xxxx xxxxB

EP1 address: 1110 0001 xxxx xxxxB

Figure 10-2. Expansion Port Addressing

INPUT PORT EXPANSION

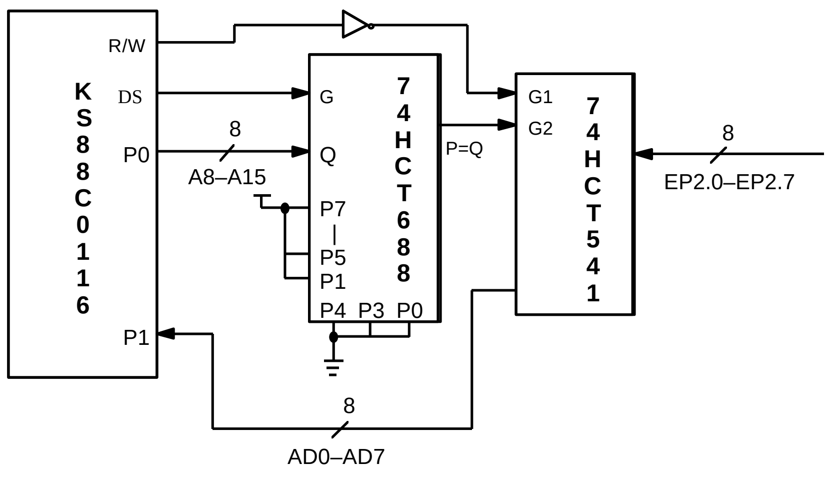
As you will note in Figure 10-3, the address for expansion port 2 (EP2) is 1110 0010 xxxx xxxxB.

For example.

```
EP2         equ         0E200H

            lde         r0,EP2                 ;  r0 ← EP2

; or

            ldw         rr2,#EP2
            lde         r0,@rr2                ;  r0 ← EP2
```

NOTE

By combining the two circuits shown in Figures 10-1 and 10-3 for output and input port expansion, you can complete the external I/O implementation for reading the port status and writing data to the port.



EP2 address: 1110 0010 xxxx xxxxB

Figure 10-3. Input Port Expansion Block Diagram

External Circuits and Programs to Support Hardware and Software Power-On/Reset Operations

Application Note: KS88APN
September 1996

Application Engineering Department
LSI Division, Micom Sector

MEMORY EXPANSION

OVERVIEW

When designing an application, it is often impossible to estimate early in the project the exact amount of ROM and RAM that will be required. Expansion program ROM or data RAM allows for a margin of error and eliminates problems associated with estimation. If the internal program ROM or data RAM proves to be insufficient for an application, a designer can implement the additional memory space externally.

To expand its available on-chip memory space, the KS88C0116 microcontroller provides an external interface through port 0 and port 1. These ports serve as the address bus and data bus, and port 2 is used to carry the bus control signals.

You can access program memory and data memory locations over this 16-bit multiplexed address/data bus. Instructions can be fetched or data read, from an external program memory device. If external program memory is implemented using a RAM-type device, you can write either program code or data to this memory space.

This application note describes a program memory expansion for the KS88C0116 using a 32-Kbyte EEPROM (part number 27C256, see Figure 11-1) and a data memory expansion using a 32-KByte SRAM (part number KM62256, see Figure 11-4). This type of memory expansion is simple to implement and the external access times are almost as fast as those for on-chip ROM or RAM locations.

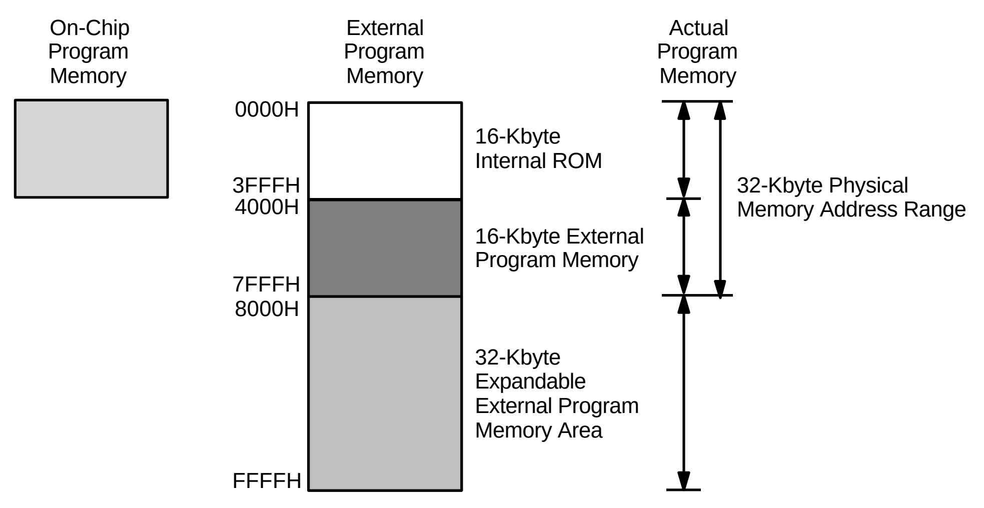

On-Chip Program Memory
External Program Memory
Actual Program Memory
0000H
3FFFH
4000H
7FFFH
8000H
FFFFH
16-Kbyte Internal ROM
16-Kbyte External Program Memory
32-Kbyte Expandable External Program Memory Area
32-Kbyte Physical Memory Address Range

Figure 11-1. Address Map of External Memory Using a 32-Kbyte EEPROM (27C256)

EXPANDING PROGRAM MEMORY

Transferring Program Control to the Expanded Program Memory

The following instructions show you two ways you can transfer program control from internal memory to the expanded external program memory area: You can 1) jump to an address located in the expanded memory area or you can 2) call a subroutine whose address is in the expanded memory area:

```
            jp          t,addr                  ;  addr is located in the expanded program memory
; or
            call        addr                    ;  addr is subroutine located in off-chip program memory
```

Loading a Byte of Data from Expanded Program Memory

The following instruction sequence shows you one example of how to load one byte of data from an expanded program memory area:

```
            ld          rr2,#455FH
            ldc         r0,@rr2                 ;  r0 ← Contents of program memory 455FH
            ldc         r0,#01H[rr2]            ;  r0 ← Contents of program memory 4560H
            ldc         r0,#1000H[rr2]          ;  r0 ← Contents of program memory 555FH
            ldc         r0,455FH                ;  r0 ← Contents of program memory 455FH
            ldcd        r0,@rr2                 ;  r0 ← Contents of program memory 455FH
                                                ;  rr2 ← rr2 – 1
            ldci        r0,@rr2                 ;  r0 ← Contents of program memory 455FH
                                                ;  rr2 ← rr2 + 1
```

Writing a Byte of Data to Expanded Program Memory (SRAM, or a similar device)

The following instruction sequency shows you one example of how to write one byte of data to expanded program memory (in this case, SRAM):

```
            ld          rr2,#455FH
            ld          r0,#5AH
            ldc         @rr2,r0                 ;  Contents of program memory 455FH ← r0
            ldc         #01h[rr2],r0            ;  Contents of program memory 4560H ← r0
            ldc         #1000h[rr2],r0          ;  Contents of program memory 555FH ← r0
            ldc         455FH,r0                ;  Contents of program memory 455FH ← r0
            ldcpd       @rr2,r0                 ;  rr2 ← rr2 – 1
                                                ;  Contents of program memory 455EH ← r0
            ldcpi       r0,@rr2                 ;  rr2 ← rr2 + 1
                                                ;  Contents of program memory 4560H ← r0
```

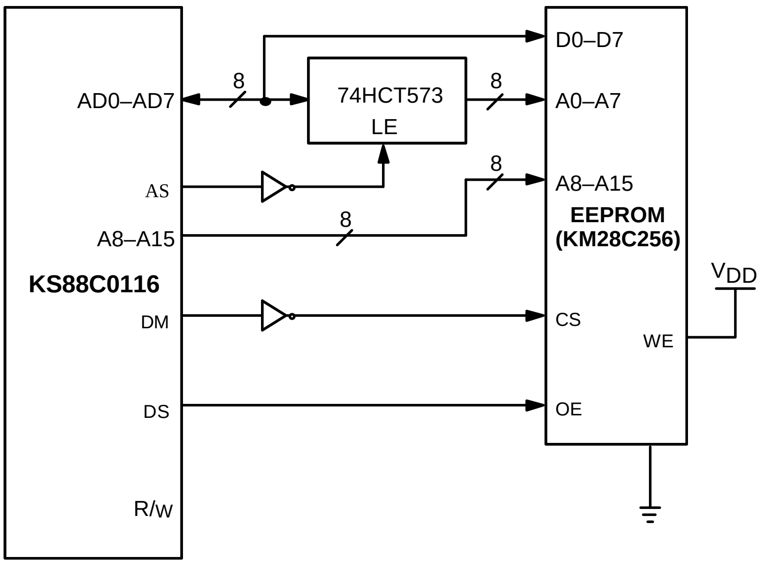


Figure 11-2. Using an EEPROM (KM28C256) as Program Memory

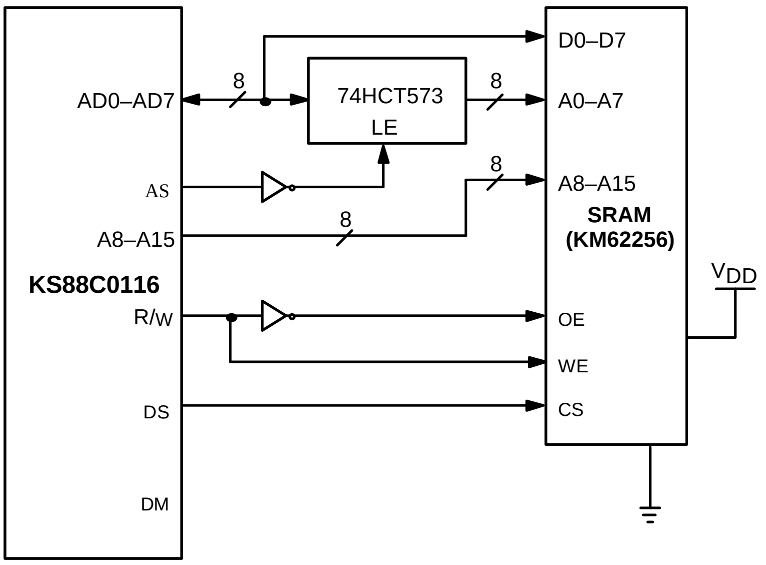


Figure 11-3. Using an SRAM (KM62256) as Program Memory

EXPANDING DATA MEMORY

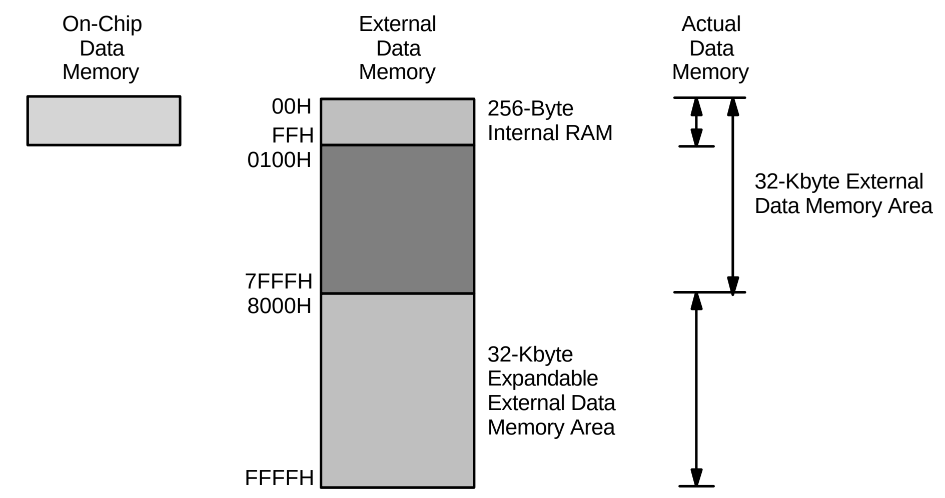


Figure 11-4. Address Map of Expanded Memory Using a 32-Kbyte SRAM (KM62256)

READING AND WRITING EXTERNAL DATA MEMORY

1. Read operation from external data memory:

```
ld        rr0,#123FH
```

2. Read operation using indirect addressing:

```
lde       r2,@rr0               ;  r2 ← 123FH address of data memory
lded      r2,@rr0               ;  r2 ← 123FH address of data memory
                                ;  rr0 ← rr0 – 1
ldei      r2,@rr0               ;  r2 ← 123FH address of data memory
                                ;  rr0 ← rr0 + 1
```

3. Read operation using indexed addressing:

```
lde       r2,#01H[rr0]          ;  r2 ← 1240H address of data memory
lde       r2,#1000H[rr0]        ;  r2 ← 223FH address of data memory
```

4. Read operation using direct addressing:

```
lde       r2,123FH              ;  r2 ← 123FH address of data memory
```

5. Write operation to external data memory:

```
ld        rr0,#123FH
ld        r2,#5AH
```

6. Write operation using indirect addressing:

```
lde       @rr0,r2               ;  123FH address of data memory ← r2
ldepd     @rr0,r2               ;  rr0 ← rr0 – 1
                                ;  123EH address of data memory ← r2
ldepi     @rr0,r2               ;  rr0 ← rr0 + 1
                                ;  1240H address of data memory ← r2
```

7. Write operation using indexed addressing:

```
lde       #01H[rr0],r2          ;  1240H address of data memory ← r2
lde       #1000H[rr0],r2        ;  223FH address of data memory ← r2
```

8. Write operation using direct addressing:

```
lde       addr,r2               ;  1240H address of data memory ← r2
```

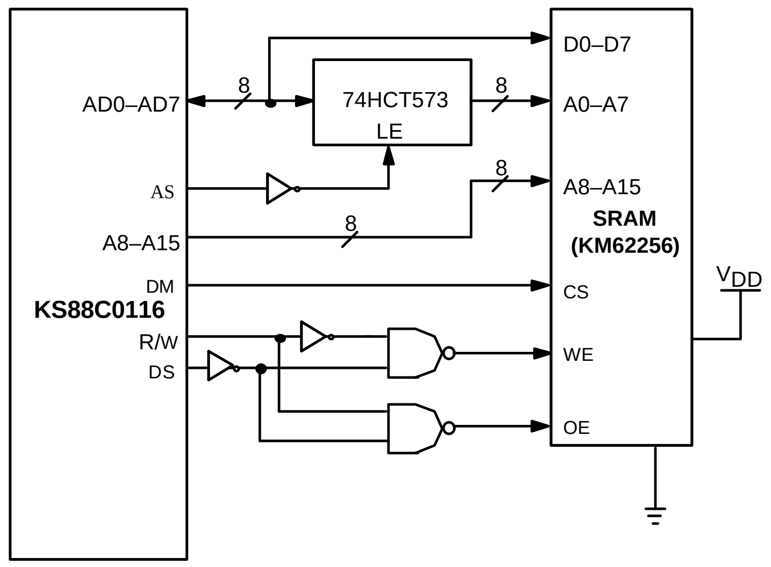


Figure 11-5. Using an SRAM (KM62256) as Data Memory

ADDRESS MAP OF THE EXTERNAL MEMORY AREA

Table 11-1. Logic Map for Memory Expansion Using the 74HCT138 Decoder

C (V_{SS})	B (A15)	A (A14)		Y	Address Range	Comment
0	0	0		Y0	0000H–3FFFH (16 K bytes)	On-chip ROM, no connection
0	0	1		Y1	4000H–7FFFH (16 K bytes)	Off-chip SRAM, data memory
0	1	0		Y2	8000H–BFFFH (16 K bytes)	Off-chip SRAM, PGM memory
0	1	1		Y3	C000H –FFFFH (16 K bytes)	Off-chip I/O port

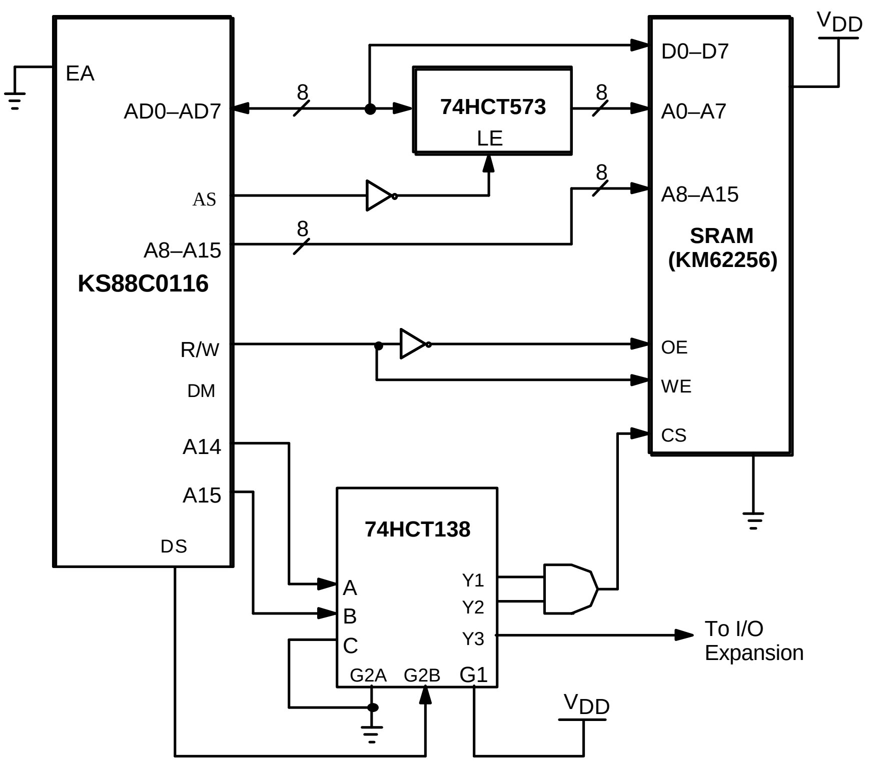


Figure 11-6. Block Diagram of External Memory Configuration

NOTES