ZILOG Z8® MCU I/O EXPANDER MODULE

ROMLESS MODE CAN BE USED WITHOUT SACRIFICING THE NORMAL I/O FUNCTION OF PORT0 AND PORT1, AND WITHOUT ADDING ADDITIONAL HARDWARE COMPLICATIONS TO THE TARGET APPLICATION.

INTRODUCTION

The Zilog Z8® MCU family of devices supports a ROMless mode that enables the Z8 code to be executed from an external EPROM. Supporting ROMless mode is relatively easy, requiring only minimal additional hardware. Supporting ROMless mode *and* I/O functions at Port0 and Port1, on the other hand, is a bit more complicated. The ROMless mode configures Port0 and Port1 as address and data lines that are used to interface with the EPROM, making the ports unavailable as general input and output signals. Logic must be added to decode an external memory address; however, adding more logic results in additional hardware complications for the target application.

This application note explains how ROMless mode may be used *without sacrificing* the normal I/O function of Port0 and Port1, and *without adding* additional hardware complications. The Z8 I/O Expander Module described here provides the necessary additional logic used to address an output latch and an input buffer—without further complicating the hardware on the target board with extra circuitry.

Description

The Z8 I/O Expander Module is a small surface mount board developed to make ROMless mode easy to implement (see Figure 1). The decoding logic is integrated with the output latch and an input buffer with a 27C256 EPROM socket. All of the logic is self contained so that the target board can be designed as though Port0 and Port1 are used as input and output lines.

The I/O Expander connects to the target board via a miniature 40-pin connector.

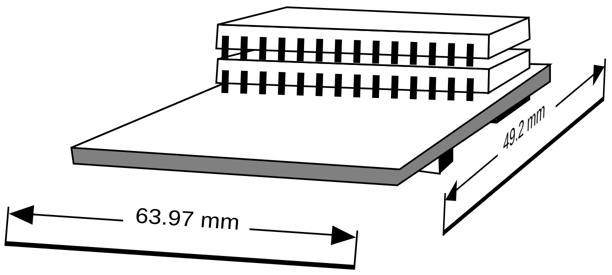

Figure 1. I/O Expander Module (approximate size)

Purpose

The I/O Expander provides a desirable alternative to using an emulator because of size constraints associated with the emulator, ICE pod, and extra power supply. The I/O Expander may be particularly helpful if the prototype is being evaluated by the end customer or a testing agency. In addition, the I/O Expander will improve reliability while enabling the product to be self-contained in the product enclosure.

THEORY OF OPERATION

Refer to the I/O Expander schematic (Figures 3 and 4) for IC references. The I/O Expander board provides a socket for the 27C256 EPROM and its support 74HC374 latch for decoding the upper address lines. A 74HC139 is used to decode address 8000H for both the input and output strobe. Addresses 0 - 7FFFH are decoded by U4B-12 and provide the chip select for the EPROM. Address 8000H is used for both the output latch 74HC374 and input buffer 74HC541, where the R/W signal selects whether U4A-4 or U4A-5 is asserted Low, which depends on whether a Write or a Read function is being performed. The I/O Expander requires all of Port0 to be inputs and all of Port1 to be outputs.

The I/O Expander plugs into the target board using a 40-pin (.050 pin spacing) surface mount connector. The signal pinout was arranged so that when the I/O Expander is removed from the target board a special straight-through shorting block may be plugged in to connect all the pins straight across 1–2, 2–4, and so on. Pins 33 through 40 are not connected by the shorting block since those lines provide the signals from the Z8® MCU to the decoder logic on the I/O expander.

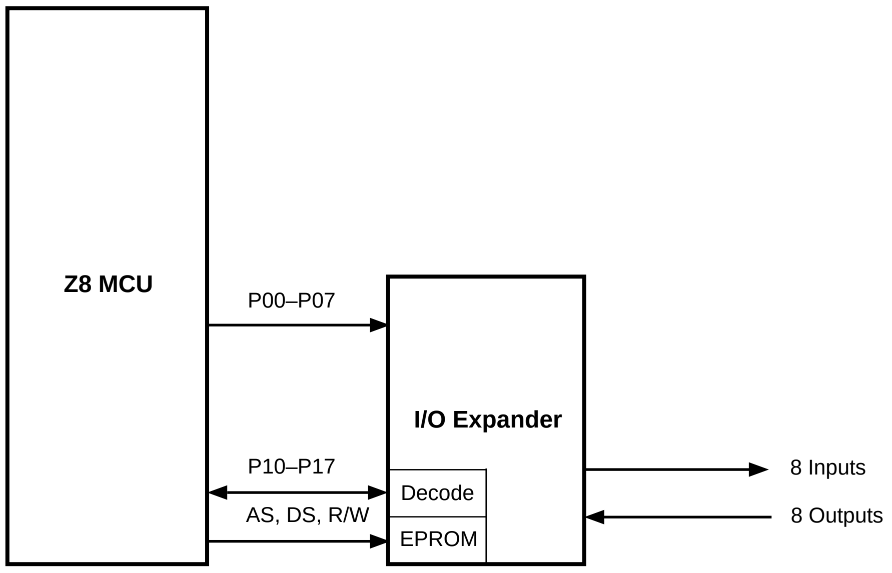

Figure 2. I/O Expander Module Block Diagram

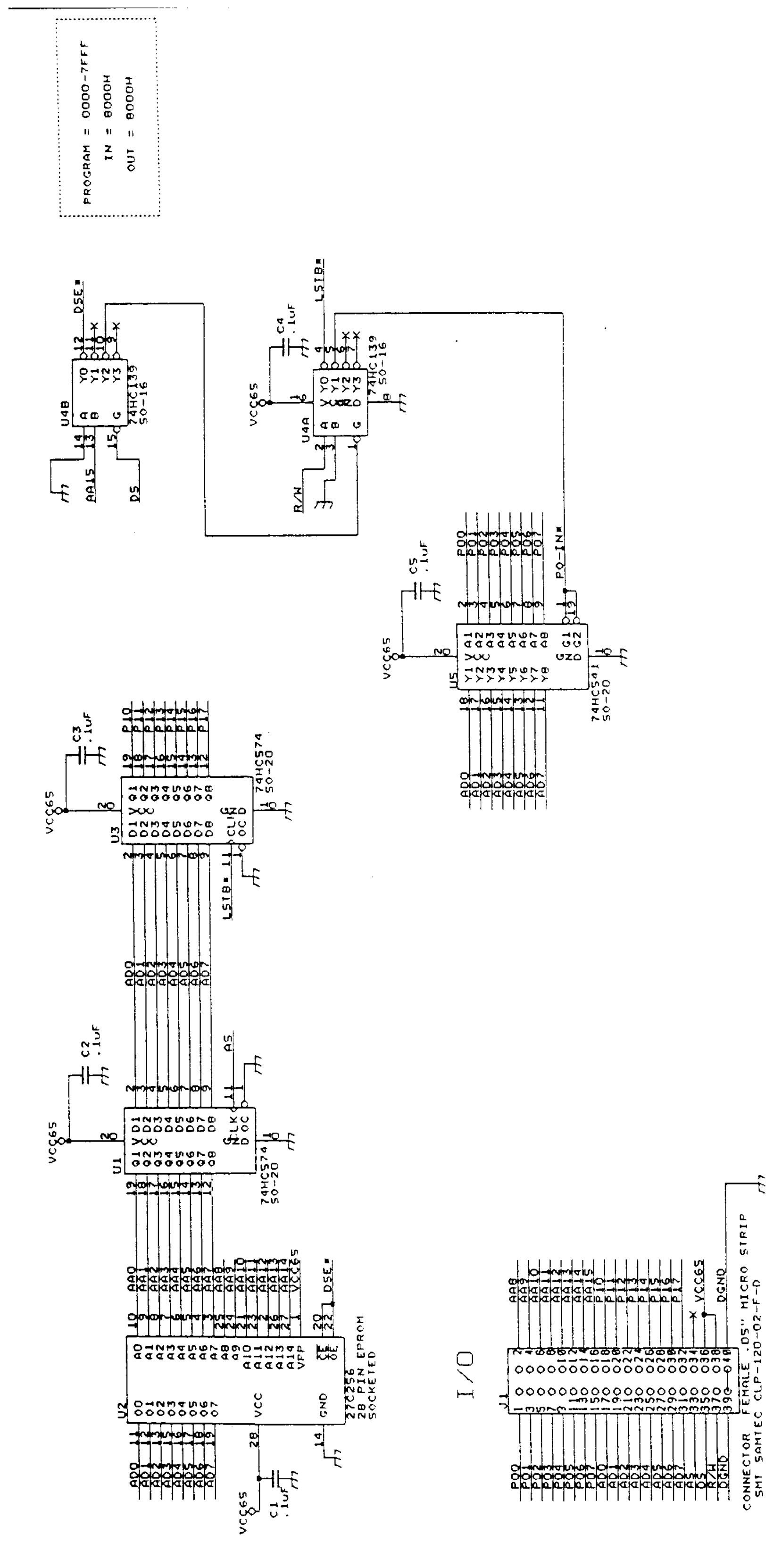

Figure 3. I/O Expander Module Schematic

The schematic (shown in Figures 4 and 5) of the Z89175 FLASH Digital Telephone Answering Device (DTAD) with Caller Identification depicts the I/O Expander implemented in a typical application.

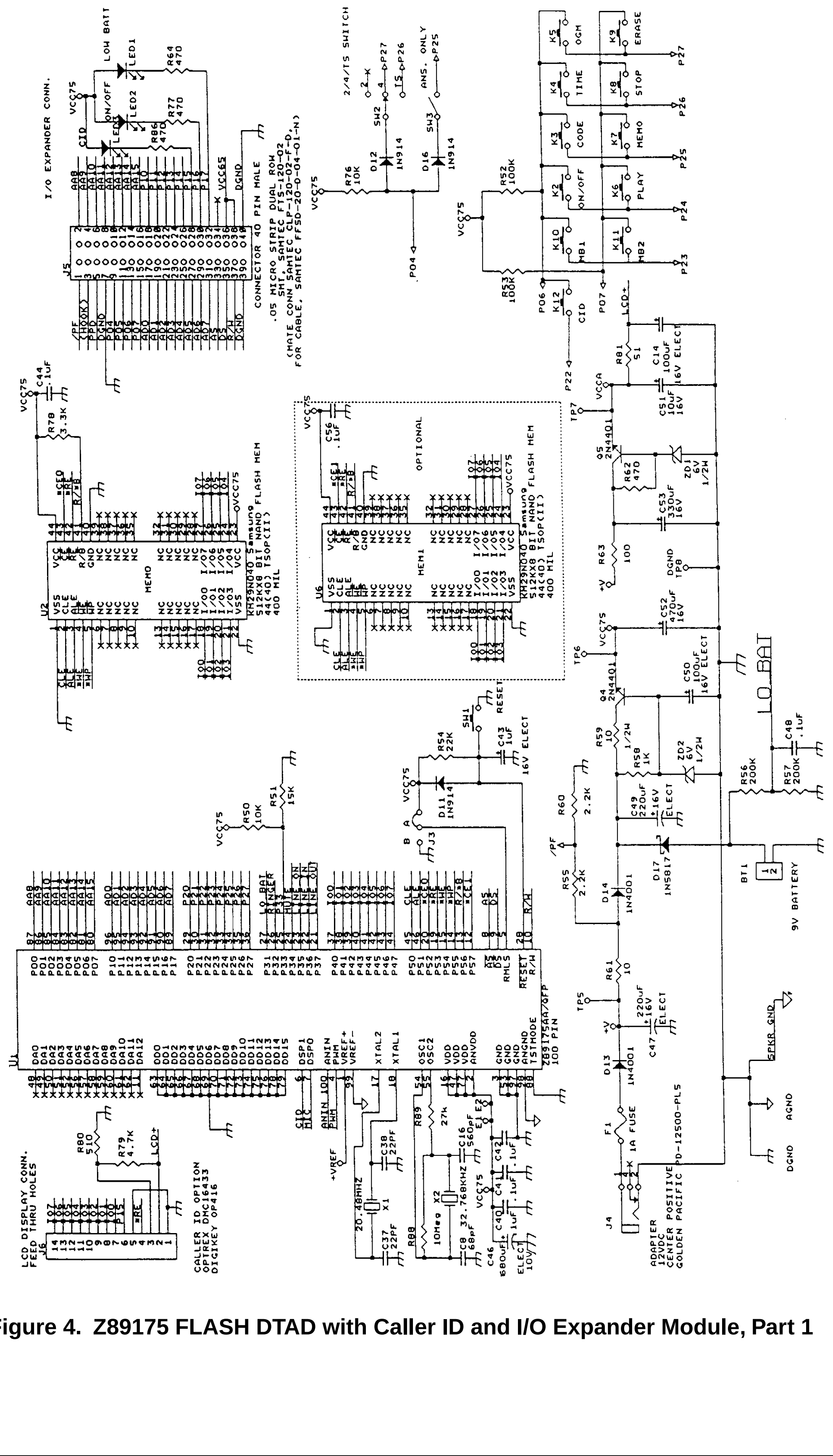

Figure 4. Z89175 FLASH DTAD with Caller ID and I/O Expander Module, Part 1

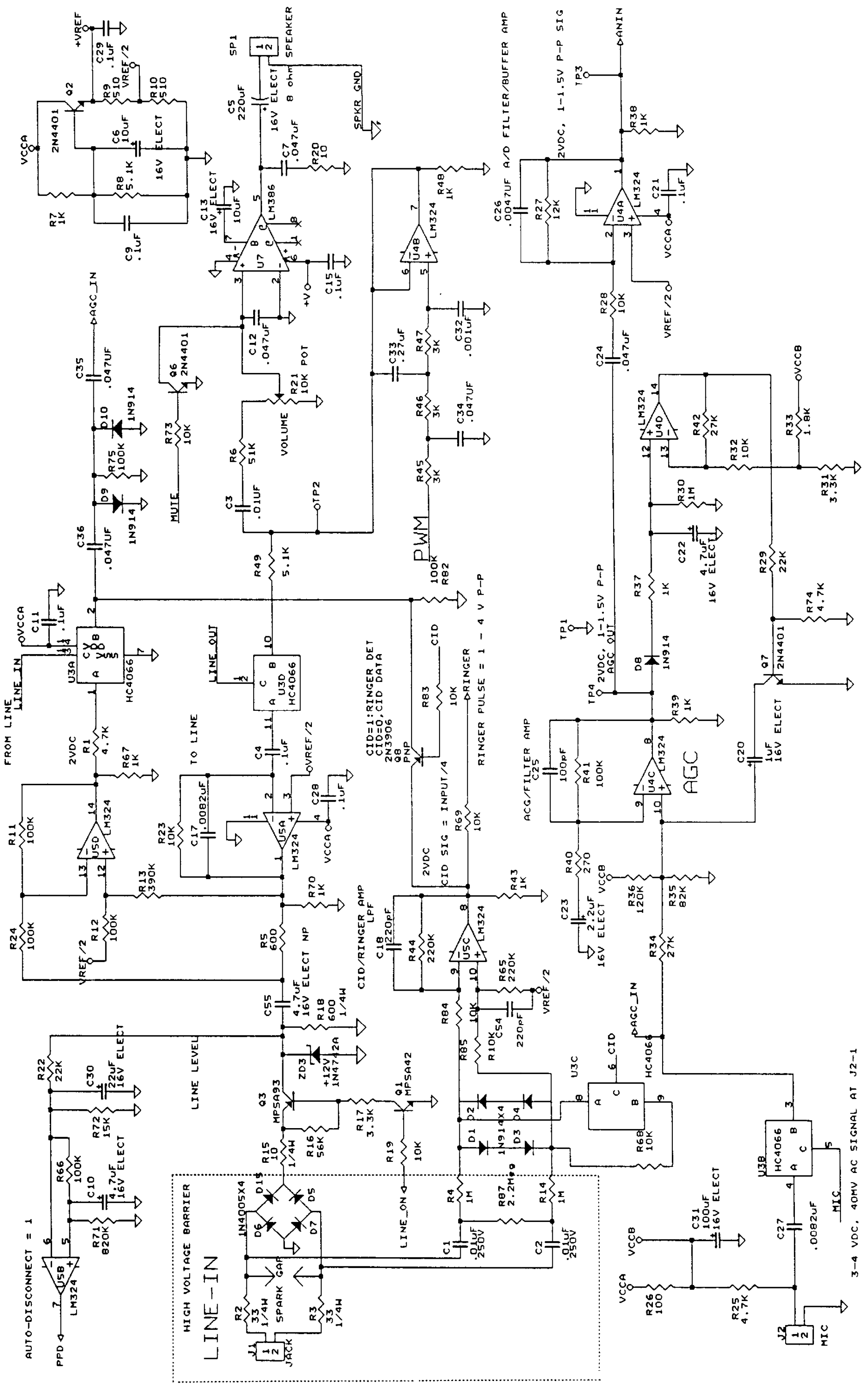

Figure 5. Z89175 FLASH DTAD with Caller ID and I/O Expander Module, Part 2

HARDWARE ISSUES

EPROM access times in the 150–200 ns range are suitable for crystal clock speeds of 10 MHz for most applications. (Reference the appropriate Zilog data sheets for the exact access time requirement for the particular device being used.)

You should carefully consider compatibility of the I/O Expander ports compared to the Z8® MCU ports. The 74HC541 IC used for the inputs does not have an autolatch feature like the Z8 MCU; therefore, if an input is not used and left open it may cause oscillations that could result in noise in adjacent circuits. Any unused inputs should be connected to V_{CC} or GND.

Also, consider that the 74HC374 used for the outputs has a higher output drive capability than the Z8. The hardware should be designed to be within the specifications of the output drive capabilities for the Z8 MCU.

The I/O Expander is hard-wired so that Port0 is all inputs and Port1 is all outputs. The outputs are driven by a 74HC374, which has totem pole outputs. The totem pole outputs should not present a problem in most cases, if the target design uses open-drain on Port1.

Note: If the port was used to scan a keyboard matrix, do not press more than one key at a time unless isolation diodes are used.

Power Consumption Effects

The additional power consumption from the EPROM and I/O Expander should not affect most designs. Most of the extra current (approximately 10 mA) will be used by the EPROM. The three CMOS parts are very low power. The quiescent current on the 74HC374, for example, is about 160 µa (worst case).

The most notable impact the I/O Expander will have on power consumption will be when the application is in a Low-Power mode. Increased power consumption caused by the I/O Expander can be minimized by setting the outputs High or Low, so that minimal current is drawn during either Halt or Stop mode. If the application uses Stop mode, the current from the EPROM will be equal to the standby current listed in the particular Zilog data sheet for the EPROM being used. (A typical value might be 100–250 µa.) For Halt Mode, the current should be significantly less than the active current of the EPROM, since the EPROM will be in standby most of the time. In applications where the Halt Mode is automatically exited by a programmed interrupt, then the standby time will depend on the interrupt rate and the processing time after each interrupt.

SOFTWARE ISSUES

The I/O Expander takes some additional code space, but this is reduced somewhat when the `EXTMEMORY` equate is set to `0` for a mask release. (Refer to the "Program Listings" section, which follows.) In this case, the `InputPort1` and `OutputPort0` subroutines reduce to a single output instruction, the equivalent of what it would be without the I/O Expander. Reading or writing to these ports will require two instructions instead of one.

If the Z8® MCU is a part that includes a DSP processor, the assigned engineer should verify that the burned-in DSP code is the correct version for the intended application. If there was a recent DSP update, that version may not be available in a ROMless part.

The customer should test with both the I/O Expander and the emulator to ensure that there is nothing that could result in a problem with the masked part. This is especially important if the intended masked DSP code is not the version in the ROMless part used with the I/O Expander. In this case, the final software verification should be performed with the emulator.

OutPutPort1 Timing Considerations

When the foreground software uses the `OutputPort1` routine to change the port setting, the exact timing could be affected by the same `OutputPort1` routine used in the interrupt handler.

When the `OutputPort1` routine is called, the changes take effect immediately. If only the "port1save" register is modified, the change will take effect on the next system interrupt. This simpler approach can be used where timing is not critical.

For applications requiring critical timing sequences, the user can temporarily disable interrupts.

SUMMARY

The I/O Expander is useful in applications where Port0 and Port1 are needed by the application as I/O and ROMless mode is also desired. The small size of the module, combined with the software support provided in this application note, makes implementation easy.

The I/O Expander is implemented on a Z89175 Evaluation Kit Board (Z8917500CZ0) that demonstrates digital answering machine functions with FLASH memory and Caller ID. For more information, contact a Zilog sales office.

PROGRAM LISTINGS

The I/O Expander software support includes Port0 and Port1 initialization, input and output subroutines, and some example code that demonstrates how the I/O Expander is used. The names of registers are the same as those used in the Z89175 Evaluation Kit (Z8917500ZC0) application code.

A conditional assembly is used to assemble the code to initialize Port0 and Port1 as address and data if the EXTMEMORY equate is = 1. If the equate is = 0, then the code is assembled to initialize Port0 as an input and Port1 as an output.

```
EXTMEMORY.equu    1                         ; 1=I/O Expander. 0=silicon or emulator

       srp #%F0                             ; Point to control registers
       .if     EXTMEMORY
       ld      r8,#%96                      ; P01M, p0/p1=add/data lines
       .else
       ld      r8,#01000101B                ; P01M, p0/p1=outputs
       .endif
```

OutputPort1 Subroutine

The OutputPort1 subroutine is used to output a new value to the port. A copy of the output port is always kept in the "port1save" register. Any changes to the port require either an AND or an OR instruction to be performed on the register, then the OutputPort1 subroutine can be called to update the port. If the code is running with the I/O Expander, then the output is performed by an external address write. To run with the emulator or for the mask release, the output is performed by writing to the port directly with a single instruction. The InportPort0 subroutine works in a similar fashion.

```
extadd  .equ   %8000                        ; External Address

OutputPort1:
       .if     EXTMEMORY                    ; Output to Port1
       ld      r0,#^HB(extadd)              ; Point to External I/O
       ld      r1,#^LB(extadd)
       ld      r7,port1save                 : Get New Data
       lde     @rr0,r7                      ; Output Data to “Fake” Port1
       ret

       .else
       ld      port1,port1save              ; Output Real Port1 Data
       ret
       .endif
```

InputPort0 Subroutine

The InputPort0 subroutine is used to read the input port. After calling the InputPort0 routine, the port value is stored in a working register. Permanent data needs to be saved by the application to a permanent register. In most cases, a permanent register is not needed because the port value can be read using the routine anytime.

```
InputPort0:
       .if     EXTMEMORY                    ; Input Port0
       ld      r0,#^HB(extadd)              ; Point to External I/O
       ld      r1,#^LB(extadd)
       lde     nport0,@rr0                  ; Get Expanded Port0 Data
       ret

       .else
       ld      nport0,port0                 ; Get Real Port0 Data
       ret
       .endif
```

Note the usage of the LDE instruction as opposed to an LDC instruction. The LDE instruction is used to write or read from external memory.

Setting Bit 6 Low

This example shows how to set Bit 6 Low, while not affecting any other bits on the output port.

```
       and     port1save,#%bf              ; Turn on LED by Setting the Bit Low
       call    OutputPort1
```

Reading in New Data from the Port

This example shows how to read in new data from the port. The data is read into a temporary register to direct a conditional jump instruction. R7 is used for the temporary “nport0” register in the Z89175 Evaluation Kit application code. A regular register also could be used, but a working register was used in the kit code to conserve regular registers for the application.

```
       Call    InputPort0                  ; Get Port0
       tm      nport0,#1                   ; Check if bit0 is High
       jr      nz,PinWasHigh
       .
       .
PinWasHigh:
       .
       .
```

Zilog, Inc., 210 East Hacienda Ave.
Campbell, CA 95008-6600
Telephone (408) 370-8000
FAX (408) 370-8056
BBS (408) 370-8024
Internet: http://www.zilog.com