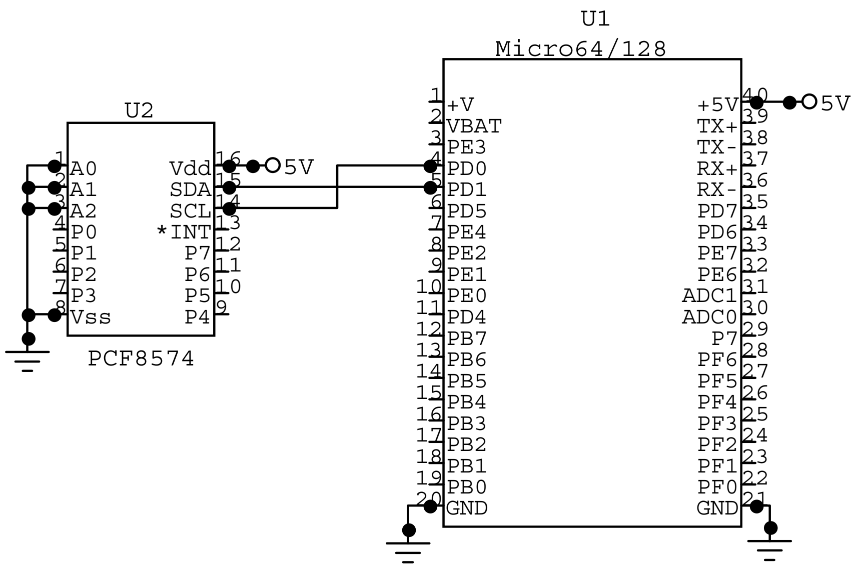
Micromint, Inc	AN701
	Micro64/128
I^{2}C Digital Input/Output Expansion	12/3/04

Introduction: This application note demonstrates how to connect and access a PCF8574 I^{2}C I/O Expander to the Micro64/128 for additional Digital I/O.

Background: Micro64/128 has 29 digital I/O available for the end user to connect digital devices to. Some applications need more than 29 digital I/O. A quick and easy way to add 8 additional digital I/O is to use a PCF8574 I^{2}C I/O Expander manufactured by Philips Semiconductors. The following schematic shows how to connect a PCF8574 I^{2}C I/O Expander to the Micro64/128.



How it works: There are two different PCF8574 I^{2}C I/O Expanders, the PCF8574 and the PCF8574A. The difference between them is the base address. The both have three address lines (A0-A2) which allow the user to set the address of the device. A specific address is set by pulling the lines high or low as shown in the table below. The maximum number of each chip that can be connected to the I^{2}C bus is eight. That can give you a maximum of 128 additional I/O. The CodeVision AVR program demonstrates how to use Micro64/128 Utilities to access a chip with the address of 40H.

PCF8574			
Chip Address	A2	A1	A0
40H	GND	GND	GND
42H	GND	GND	+5V
44H	GND	+5V	GND
46H	GND	+5V	+5V
48H	+5V	GND	GND
4AH	+5V	GND	+5V
4CH	+5V	+5V	GND
4EH	+5V	+5V	+5V

PCF8574A			
Chip Address	A2	A1	A0
70H	GND	GND	GND
72H	GND	GND	+5V
74H	GND	+5V	GND
76H	GND	+5V	+5V
78H	+5V	GND	GND
7AH	+5V	GND	+5V
7CH	+5V	+5V	GND
7EH	+5V	+5V	+5V

Program Listing:

```
/****************************************
Program : I2C I/O Expander example for Micro64 Using the I2C Utilities
Company : Micromint, Inc.
*******************************************/

#include <mega64.h>
#include <MMRS485.h>     // Micromints Library for using both USARTs
#include <stdio.h>      // Standard I/O library
#include <delay.h>                                    // Library for delays


// Declare your global variables here
unsigned int Pass @0xFFE;
unsigned char DATA @ 0xFFD;
unsigned char SLADDR @ 0xFFB;
int COM;         // if COM = 0 then use USART0 if it = 1 then use USART1

void(*I2CSEND)(void)=0x07CB8;
void(*I2CREAD)(void)=0x7CDD;
void(*I2CINIT100KHZ)(void)=0x7C23;
void(*I2CDisable)(void)=0x7C37;

void main(void)
{
// Declare your local variables here

// Set up USART1's Baud rate at 9600 bps with a 11.0592 MHz Crystal
UCSR1A=0x00;      // RX EN, TX EN
UCSR1B=0x18;     // RX EN, TX EN
UCSR1C=0x06;     // 8N1
UBRR1H=0x00;    // Baud rate high - 9600
UBRR1L=0x47;   // Baud rate low

COM = 1;                        // Use USART1
DDRD.6 = 0;                                           // Make PORTD.6 an output
PORTD.6 = 1;                                          // Enable the RS485 control line
printf("Started\r\n");
while (1)
     {

       (*I2CINIT100KHZ)();
        SLADDR = 0x40;
        DATA = 0xFF;
        (*I2CSEND)();
        (*I2CDisable)();
        printf("All of the I/O Expanders IO should be high.\r\n");
        (*I2CINIT100KHZ)();
        SLADDR = 0x40;
        (*I2CREAD)();
        (*I2CDisable)();
        printf("The I/O Expanders port = %d\r\n",Pass);
        delay_ms(2000);
        (*I2CINIT100KHZ)();
        SLADDR = 0x40;
        DATA = 0x00;
        (*I2CSEND)();
        (*I2CDisable)();
        printf("All of the I/O Expanders IO should be low.\r\n");
        (*I2CINIT100KHZ)();
        SLADDR = 0x40;
        (*I2CREAD)();
        (*I2CDisable)();
        printf("The I/O Expanders port = %d\r\n",Pass);
        delay_ms(2000);

     };
}
```