0. 使用 avr-gcc(GCC), 版本 4.8.2; avr libc 版本 1.8.0; (返回)
晶振频率 8MHz; 使用的 DS18B20 如图 1; 使用的连线参考图 2, PC1 与 DS18B20 的 DQ 引脚连接.

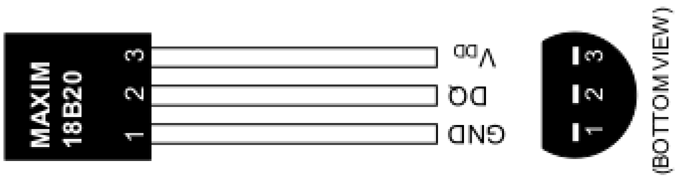

图 1 DS18B20

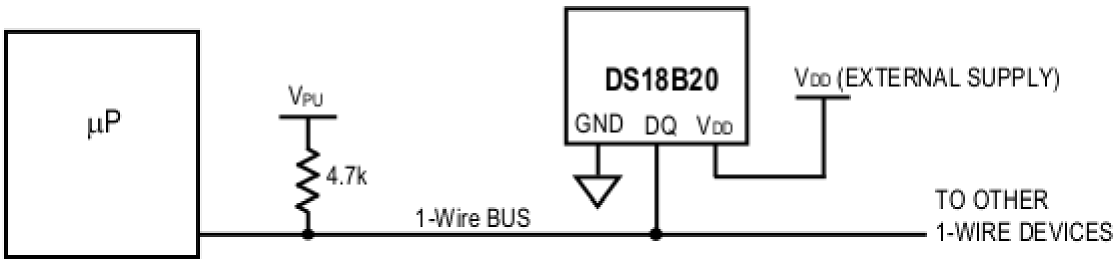

图 2 连线图

1. DS18B20.h (返回)

```
#ifndef  __DS18B20_H__
#define  __DS18B20_H__

#include <inttyppes.h>

extern void DS18B20_reset(void);
extern uint8_t DS18B20_read_byte(void);
extern void DS18B20_write_byte(uint8_t);
extern void DS18B20_start(void);
extern uint16_t DS18B20_read_Temperature(void);

#endif   /* __DS18B20_H__ */
```

2. DS18B20.c (返回)

```
#include <avr/io.h>
#include <inttypes.h>
#include <util/delay.h>
#include <avr/interrupt.h>

#include “DS18B20.h”
```

```
#define  DQ_IN      (DDRC &= ~_BV(PC1))
#define  DQ_OUT     (DDRC |= _BV(PC1))
#define  DQ_CLR     (PORTC &= ~_BV(PC1))
#define  DQ_SET     (PORTC |= _BV(PC1))
#define  DQ_R       (PINC &= _BV(PC1))

uint8_t interrupt_flag;

void  DS18B20_reset(void)
{
	//uint8_t i;
	cli();
	interrupt_flag = SREG;

	DQ_OUT;
	DQ_CLR;
	_delay_us(500);
	DQ_SET;
	DQ_IN;
	_delay_us(80);
	// i = DQ_R;
	DQ_R;
	_delay_us(500);

	if(interrupt_flag & 0x80)
		sei();
}

uint8_t DS18B20_read_byte(void)
{
	uint8_t i;
	uint8_t value = 0;

	cli();
	interrupt_flag = SREG;

	for(i = 8;  i != 0; --i) {
		value >>= 1;
		DQ_OUT;
		DQ_CLR;
		_delay_us(10);
		DQ_SET;
		DQ_IN;
		_delay_us(10);
		if(DQ_R)
			value |= 0x80;
		_delay_us(50);
	}
	if(interrupt_flag & 0x80)
		sei();

	return (value);
}

void DS18B20_write_byte(uint8_t value)
{
	uint8_t i;

	cli();
	interrupt_flag = SREG;

	for(i = 8; i != 0; --i) {
		DQ_OUT;
		DQ_CLR;
		_delay_us(10);
		if(value & 0x01)
			DQ_SET;
		_delay_us(80);
		DQ_SET;
		value >>= 1;
	}

	if(interrupt_flag & 0x80)
		sei();
}

void DS18B20_start(void)
{
	DS18B20_reset();
	DS18B20_write_byte(0xCC);	// Skip ROM
	DS18B20_write_byte(0x44);	// Convert T
}

uint16_t DS18B20_read_Temperature(void)
{
	uint16_t i;
	uint8_t buf[9];

	DS18B20_reset();
	DS18B20_write_byte(0xCC);	// Skip ROM
	DS18B20_write_byte(0xBE);	// Read Scratchpad

	for(i = 0; i < 9; ++i)
		buf[i] = DS18B20_read_byte();

	i = buf[1];
	i <<= 8;
	i |= buf[0];
	return i;
}
```

3. 编译接口与实现 (返回)
新建目录 include, lib, 以及 src, 将 DS18B20.h 放入 include 中; 将 DS18B20.c 放入 src 中:

```
$ cd /home/gwh/clutter/avr/DS18B20
$ mkdir include src lib
$ mv DS18B20.h include/
$ mv DS18B20.c src/
```

进入 lib 目录, 新建文件 Makefile:

```
$ cd lib
$ touch Makefile && vim Makefile
```

Makefile 内容如下:

```
# Makefile Start
AR=avr-ar
CC=avr-gcc
MCU=atmega16
CFLAGS=-g -mmcu=$(MCU) \
		-Wall -Wstrict-prototypes \
		-Os -mcall-prologues

libds18b20.a : DS18B20.o
	$(AR) -cr libds18b20.a DS18B20.o
DS18B20.o : ../src/DS18B20.c
	$(CC) $(CFLAGS) -I../include -DF_CPU=8000000
-D__DELAY_BACKWARD_COMPATIBLE__ -c ../src/DS18B20.c

clean :
	rm -f *.o
# Makefile End
```

编译:

```
$ make libds18b20.a
```

4. 编写基于上述接口的程序 (返回)
新建目录 Card01:

```
$ cd /home/gwh/clutter/avr/DS18B20
$ mkdir Card01
```

新建文档 main.c:

```
$ cd Card01
$ touch main.c && vim main.c
```

内容如下:

```
/* main.c */
#include <avr/io.h>
#include <inttypes.h>
#include <util/delay.h>

#include “DS18B20.h”
#include “LCD1602.h”

void Display_Temperature(int16_t, uint8_t);

int main(int argc, char *argv[])
{
    int16_t min = 0x0000;
    int16_t max = 0x0000;
    int16_t temp = 0x0000;

    Lcd1602_Init();

    Lcd1602_WriteData(0x0C, 0);     // 开显示 关光标 光标不闪烁

    DS18B20_start();
    _delay_ms(800);
    max = DS18B20_read_Temperature();
    min = max;

    while(1) {
        DS18B20_start();
        _delay_ms(800);
        temp = DS18B20_read_Temperature();
        Display_Temperature(temp, 0x80);

        if(temp > max)
            max = temp;
        Dispaly_Temperature(max, 0xCA);
        if(min > temp)
            min = temp;
        Display_Temperature(min, 0xC0);
    }

    return 0;
}

void Display_Temperature(int16_t temperature, uint8_t pos)
{
    uint8_t digits[4];

    Lcd1602_WriteData(pos, 0);
    if(temperature < 0) {
        Lcd1602_WriteData('-', 1);
        temperature = - temperature;
    }
    else
        Lcd1602_WriteData('+', 1);

    temperature *= 0.625;

    digits[0] = temperature / 1000 % 10 + '0';
    if(digits[0] == '0')
        digits[0] = ' ';
    digits[1] = temperature / 100 % 10 + '0';
    digits[2] = temperature / 10 % 10 + '0';
    digits[3] = temperature % 10 + '0';

    Lcd1602_WriteData(pos + 1, 0);

    Lcd1602_WriteData(digits[0], 1);
    Lcd1602_WriteData(digits[1], 1);
    Lcd1602_WriteData(digits[2], 1);
    Lcd1602_WriteData('.', 1);
    Lcd1602_WriteData(digits[3], 1);
}
```

5. 编译

(返回)

新建文件 Makefile, 其中-L../../lib 是指出 liblcd1602.a 库的位置; -I../../include 是指出 LCD1602.h 头文件的位置, 根据需要, 你可以修改这两处来包含你自己的头文件和链接自己的库:

```
$ touch Makefile && vim Makefile
```

内容如下:

```
# Makefile Start
CC=avr-gcc
MCU=atmega16
OBJCOPY=avr-objcopy
CFLAGS=-g -mmcu=$(MCU) \
            -Os -mcall-prologues \
            -Wall -Wstrict-prototypes

all : main.hex
main.hex : main.out
    $(OBJCOPY) -R .eeprom -O ihex main.out main.hex
main.out : main.o
    $(CC) $(CFLAGS) -L../lib -L../../lib -o main.out -Wl,-Map,main.map main.o -lds18b20
-llcd1602
main.o : main.c
    $(CC) $(CFLAGS) -I../include -I../../include -DF_CPU=8000000
```

```
-D__DELAY_BACKWARD_COMPATIBLE__ -c main.c

load : main.hex
	avrdude -c usbasp -p m16 -B 1 -U flash:w:main.hex

clean :
	rm -f *.o *.map *.out
# Makefile End
```

编译, 下载:

```
$ make
$ make load
```