Application Note - Using 1-wire devices

Author:
Joel - MyFreescaleWebPage
http://myfreescalewebpage.free.fr

Last revision of this document: 1.2 of 2012-07-31.

Table of contents

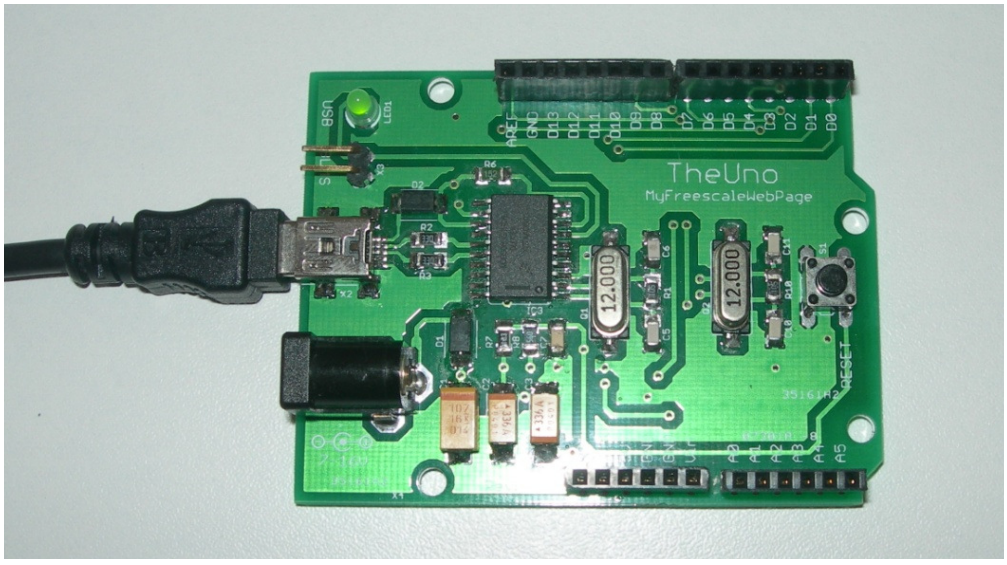
Introduction

The 1-wire is a serial interface which is only using one wire (plus the ground) to transmit data between a microcontroller and a 1-wire component. Moreover, the power supply of the 1-wire component can be drawn by the serial interface itself, making the 1-wire a very interesting communication link to connect various sensors to a microcontroller. A lot of components using 1-wire are available today: RTC, memories, sensors...

I will not deal with the 1-wire protocol and its possibilities in this application note. A very nice tutorial is available on Maxim Semiconductors website to introduce the 1-wire bus: http://www.maxim-ic.com/products/1-wire/flash/overview/index.cfm.

If you are looking for 1-wire components, checkout Maxim Semiconductor website: http://www.maxim-ic.com/products/1-wire.

This application note has been developed on TheUno, a development board created by MyFreescaleWebPage, based on S08DZ60 device and with a build-in open-source debugging cable.



TheUno is compatible with almost every shields designed for the Arduino. It is possible to plug multiple shields simultaneously on the development board, with Ethernet, Wi-Fi, GPS, audio devices, H-Bridge, ... to simply create powerful applications using Freescale CodeWarrior, a nice development environment based on Eclipse. More details about TheUno available on http://myfreescalewebpage.free.fr.

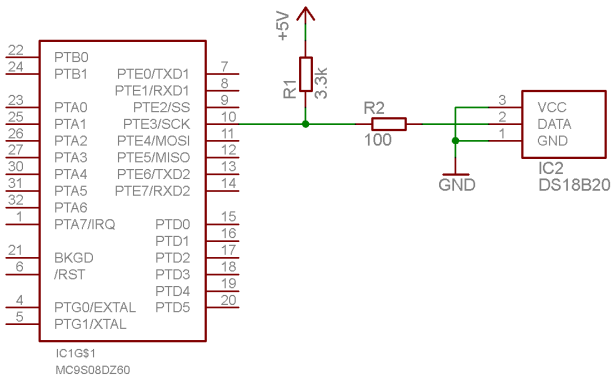
1 DS18B20 Project Example #1

The first project example attached with this application note is a very simple implementation of the 1-wire serial interface. This example is based on the DS18B20 temperature sensor. It can be used on various microcontrollers, from 8 to 32bits devices. It only requires a simple GPIO.

1.1 Architecture

1.1.1 Hardware schematic

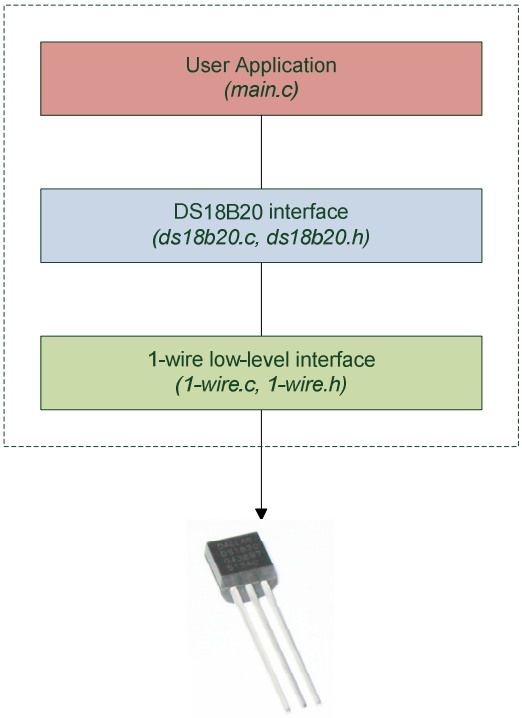
The 1-wire device is connected to the microcontroller using two simple resistors, has shown on the following schematic.



The pull-up resistor R1 is used to power the sensor. The microcontroller will also be used to draw current during temperature conversion, and it's why R2 is added, in order to limit the current in case of shortcut on the 1-wire device.

1.1.2 Software architecture

The software architecture is very simple.



The 1-wire interface contains low level functions used to communicate with the 1-wire device. Those functions are basic functions which can be used to interface any 1-wire component to the microcontroller, and not only temperature sensors.

DS18B20 interface contains specific functions to use the DS18B20 device. It uses the 1-wire functions to communicate on the 1-wire bus.

Finally, the application is simply using the DS18B20 functions to get the current temperature.

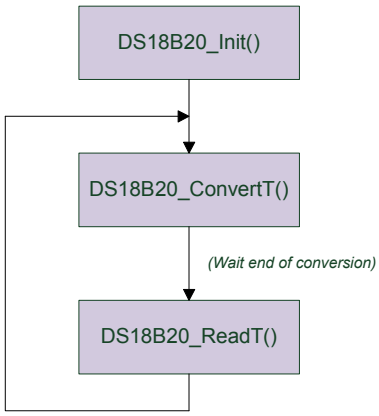
1.1.3 User Application architecture

The DS18B20 header file must be included in the user application:

```
#include "ds18b20.h"
```

The DS18B20 interface contains three functions:
- DS18B20_Init() to initialize the sensor;
- DS18B20_ConvertT() to initiate temperature conversion;
- DS18B20_ReadT() to read the temperature.

Those functions are called according to the following schema.



Please note that the temperature conversion is taking a lot of time, up to 750ms, depending of the temperature resolution required. Do not call DS18B20_ReadT() before the end of the temperature conversion.

1.2 Configurations

1.2.1 Hardware

The hardware configuration (pin used to connect the sensor) is defined in the 1-wire low level interface. It can be modified in order to match the board requirements.

In the project attached to this application note, the pin which is used to connect the sensor is PTE3:

```
#define ONEWIRE_PIN             PTED_PTED3
#define ONEWIRE_DIR             PTEDD_PTEDD3
```

If you modify the pin, please modify both registers in the above definitions.

1.2.2 Delays

The 1-wire interface is using delays based on one of the internal timer of the microcontroller.

In order to get the right delays, you have to define the bus frequency of the microcontroller (in MHz) in the 1-wire low level interface, according to the configuration of the system clock on your microcontroller:

```
#define BUSCLK                  18
```

Finally, it is possible to modify the timer used to make delays by modifying the "ONEWIRE_DELAY" macro.

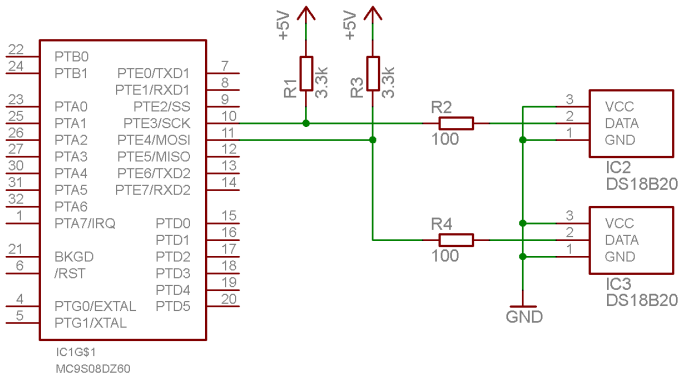
2 DS18B20 Project Example #2

The previous project is very simple but do not permit the use of multiple 1-wire devices on the same microcontroller. In order to connect several devices, I have chosen to create several 1-wire interfaces on the microcontroller, simply using several GPIO. This example uses two DS18B20 temperature sensors.

2.1 Architecture

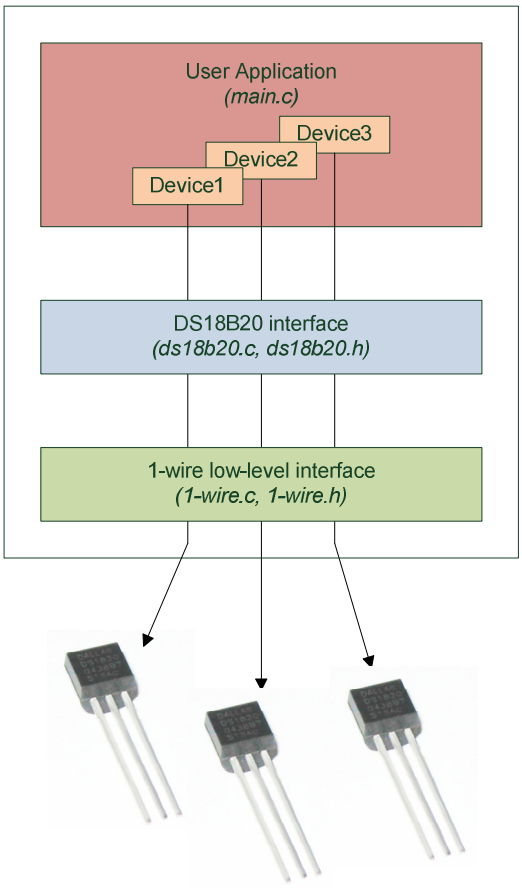
2.1.1 Hardware schematic

The DS18B20 sensors are connected to the microcontroller according to the following schematic.



2.1.2 Software architecture

The software architecture is quite the same than the previous one, as shown on the following diagram.



The 1-wire interface always contains the low level functions used to communicate with the 1-wire devices, and the DS18B20 interface still contains the specific functions useful to communicate with the DS18B20 sensors.

However, the 1-wire and DS18B20 functions are now able to manage multiple interfaces. This is simply done using a description of the interface for each device declared in the user application.

The 1-wire interface structure is the following one:

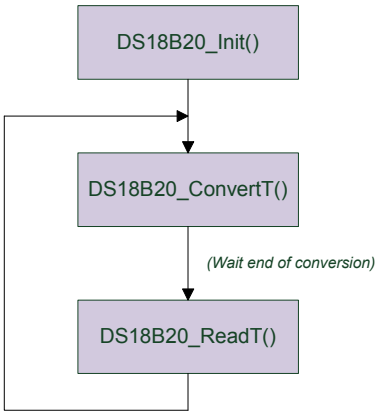
```
typedef struct
{
	UINT8 * PortData;
	UINT8 * PortDataDirection;
	UINT8 Bit;
}
t_OneWireInterface;
```

It is used to declare the DS18B20 device structure:

```
typedef struct
{
	t_OneWireInterface OneWireInterface;
	t_OneWireResetResult Status;
	UINT16 Temperature;
}
t_DS18B20Device;
```

2.1.3 User Application architecture

Each DS18B20 device is declared and then initialized by setting its interface parameters. It is then used exactly as shown in the first project example attached with this application note.



The software architecture permits the creation and the use of multiple devices at the same time: RTC, memories, etc.

2.2 Configurations

2.2.1 Hardware

As previously explained, the hardware configuration is chosen calling DS18B20_Init() function.

2.2.2 Delays

The 1-wire interface is using delays based on one of the internal timer of the microcontroller.

In order to get the right delays, you have to define the bus frequency of the microcontroller (in MHz) in the 1-wire low level interface, according to the configuration of the system clock on your microcontroller:

```
#define BUSCLK                  18
```

Finally, it is possible to modify the timer used to make delays by modifying the "ONEWIRE_DELAY" macro.

Conclusion

The projects attached with this application note are showing two simple ways to use 1-wire devices on Freescale microcontrollers. The 1-wire and DS18B20 libraries can be used with 8 to 32-bits microcontrollers.