APPLICATION NOTE 145

Interfacing the Dallas Semiconductor 1-Wire Master (DS1WM) to an ARM7 Processor

This application note provides the reader with information concerning how to interface the Dallas Semiconductor 1-Wire® Master (DS1WM) to an ARM7 processor. Both hardware and software concerns are addressed, including block diagrams and sample ANSI C code to enable communication between the 1-Wire Master ARM7 processor over the Dallas Semiconductor 1-Wire communication protocol. Software routines for initialization of the ARM7 and common 1-Wire commands are provided and therefore can be used on any Dallas Semiconductor device that communicates using the 1-Wire protocol.

Introduction

The 1-Wire Master (DS1WM) was designed to interface easily into any 8-bit system bus and generate all 1-Wire related timing freeing the system processor to perform other tasks. The following example shows how to interface the DS1WM to an ARM7 processor and how to communicate to devices on the 1-Wire bus with standard C routines.

Interfacing to the ARM7 Bus

The 1-Wire Master requires no external components to map into the ARM7's data bus. Figure 1 shows all necessary connections. The example figure uses one of the ARM7's programmable chip select lines (NCS2) to generate the enable pulse for the DS1WM. However, any negative logic chip select generated by the ARM can be used. The DS1WM's Master Reset (MR) and Clock (CLK) should be connected directly to the system reset and clock. The Address Strobe (ADS) is not used for interfacing to the ARM7 and must be grounded.
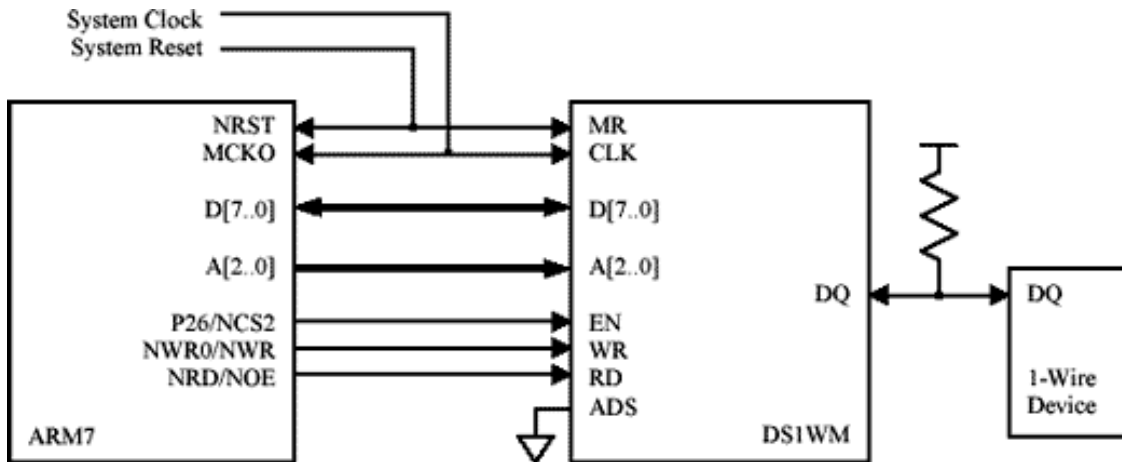

Figure 1: Bus connections between ARM7 and DS1WM

Generating 1-Wire Communication

The coding examples below are written in standard ANSI C and can be compiled with any standard ARM7 compiler. Before the ARM can read and write DS1WM registers, the ARM chip select line (NCS2 in this example) must be configured. The following code enables NCS2 and sets the base address for the DS1WM to 0x20000000. This code also enables the maximum number of wait states. The actual number of wait states required will depend on the system clock frequency and how the DS1WM is synthesized. Refer to the External Bus Interface (EBI) section of the ARM7 datasheet for a more in-depth description of initialization.

```
long* ChipSelectRegister = (long*)0xFFE00008; /* Address of Chip Select 2 Register */

*ChipSelectRegister = 0x2000203E; /* Base Address of 0x20000000 BA = 0x200 */
                                  /* Chip Select 2 Enabled CSEN = 1 */
                                  /* Write Byte Access Enabled BAT = 0 */
                                  /* Data Float Time Zero TDF = 0 */
                                  /* Page Size of 1 Meg PAGES = 0 */
```

```
                              /* Wait State Enabled WSE = 0 */
                              /* 8 Standard Wait States NWS = 0x7 */
                              /* 8 bit Data Bus Width DBW = 0x2 */
```

Once the chip enable has been initialized, communication to the DS1WM can begin. The following ANSI C code shows how to set pointers to the DS1W M's registers, initialize the bus clock, and how to perform the three major bus operations—read, write, and reset. The code assumes that the DS1WM's base address is 0x20000000h and the system clock is 32Mhz as before. The Clock Divider register should be programmed as soon as communication to the DS1WM has been established and before any communication on the 1-Wire bus takes place. The WriteByte, ReadByte, and Reset functions can then be called whenever needed. The Write, Read, and Reset functions in this example poll the Interrupt register waiting for the command to be completed. The user's functions, especially in a time sensitive application, should use this time to perform other tasks.

```c
/* Initialize pointers to DS1WM addresses. Base address of DS1WM is 0x20000000 */
unsigned char* Command1WM = (unsigned char*)0x20000000; /* Address of Command Register */
unsigned char* Data1WM = (unsigned char*)0x20000001; /* Address of Data Register */
unsigned char* Int1WM = (unsigned char*)0x20000002; /* Address of Interrupt Register */
unsigned char* Clock1WM = (unsigned char*)0x20000004; /* Address DS1WM Clock Divider */

/* The clock divider must be programmed before 1-Wire communication can take place */
/* Refer to the DS1WM datasheet page 4 for the proper programming value for you system clock */
*Clock1WM = (unsigned char)0x12; /* Setup for 32Mhz Clock */

/* Reset will generate a reset on the 1-Wire bus. If no presence detect was seen, it will return a 1, */
/* otherwise it returns 0. */
int Reset(void)
{
        unsigned char result;
        int loop;
        int DELAY = 30000; /* Time to Poll for Completion */
                                        /* Choose a length to Poll that is slightly greater than */
                                        /* the maximum possible time to complete the operation */
        *Command1WM=0x01; /* Send 1WR Reset */
        for( loop=0; loop < DELAY; loop++ ) /* Poll INT Register for command completion */
        {
                result=*Int1WM;
                if(result&0x01) break;
        }
        if((result&0x02) != 0x00) return 1; /* No Presence Detect Found, Return Error */
        return 0;
}

/* Send a byte on the 1-Wire Bus. Poll for completion afterwards */
/* Return 1 if operation timed out, 0 if successful */
int WriteByte(char Data)
{
        unsigned char result;
        int loop;
        int DELAY = 30000; /* Time to Poll for Completion */
                                /* Choose a length to Poll that is slightly greater than */
                                /* the maximum possible time to complete the operation */

        *Data1WM = Data; /* Transmit Data Byte on the Bus */
        for( loop=0; loop < DELAY; loop++ ) /* Poll INT Register for command completion */
        {
                result=*Int1WM;
                if(result&0x0C) == 0x0C ) break; /* Poll TBE & TEMT until both are empty */
        }
        if(loop == DELAY) return 1; /* Operation Timed Out */
        return 0;
}

/* Read a byte from the 1-Wire Bus. Write a 0xFF to create read time slots and poll for receive */
/* buffer full before moving the result to the location pointed to by *Data. Returns a 1 if either */
/* the write 0xFF or read times out, 0 otherwise. */
int ReadByte(char *Data)
{
        unsigned char result;
```

```
        int loop;
        int DELAY = 30000; /* Time to Poll for Completion */
                                  /* Choose a length to Poll that is slightly greater than */
                                  /* the maximum possible time to complete the operation */

        if(WriteByte((char)0xFF)) return 1; /* Generate 1-Wire read timeslots */
        for( loop=0; loop < DELAY; loop++ ) /* Poll INT Register for command completion */
        {
                result=*Int1WM;
                if( result&0x10) == 0x10 ) break; /* Poll RBF until set */
        }
        if(loop == DELAY) return 1; /* Operation Timed Out */
        *Data = *Data1WM; /* Retrieve data that was returned */
        return 0;
}
```

Summary

The Dallas Semiconductor 1-Wire Master will interface easily to an ARM7 processor without the need for any external components. Standard ANSI C functions can then be used to generate all necessary 1-Wire communication while freeing the processor to perform other tasks. Refer to the DS1WM datasheet or AN120 "Communicating through the 1-Wire Master" for more information on how to interface to and program the 1-Wire Master.

More Information

DS1WM: QuickView   -- Full (PDF) Data Sheet