Pseudo-Code Defining the Interface Functions

2/19/2003 N. Kitamura
3/28/2003 (revision)

In the subsequent description, assume
"**use** work.flash_defs.**all**" for the symbol definitions.

DS2401, DS2502 Functions

"**Master**" refers to the 1-Wire Bus Master implemented as a VHDL code. "**Addr**" in the pseudo-code is an offset address relative to the base address.

Base address = **addr_owm_command**

"**Slave**" refers to a 1-Wire device connected to one of the eight slave channels:

DQ0 – DQ5 : DS2502 PROM for LED calibration data
DQ6 : DS2401 Flasher Board ID

Select_Slave_Channel
In order to access a 1-Wire device, DOMMB must first select one of the 1-Wire slaves by writing a word to **addr_owm_slvsel**. Only one slave channel may be selected at any given time.

Data word	Selected channel
0000 0000	(none)
0000 0001	0
0000 0010	1
. .	.
. .	.
. .	.
1000 0000	7

Initialize_Master
Each time a different slave channel is selected, master must be initialized by the following sequence.
1. Reset
2. Set up clock divider (Addr=0x04, Data=0x0D)
3. Set up Interrupt Enable Register (Addr=0x03, Data=0x17)

```
     EPD=1     Enable Presence Detect Interrupt
     IAS=1     Set INTR to be active high
     EBTE=1    Enable Transmit Buffer Empty Interrupt
     ETMT=0    Disable Transmit Shift Register Empty
               Interrupt
     ERBF=1    Enbble Receive Buffer Full Interrupt
     ESINT=0   Disable Slave Interrupt
     ENBSY=0   Disable Not Busy Interrupt
     DQOE=0    Disable DQ Output Enable
```

Send_Reset_Pulse

1. Master Tx "reset" (Addr=0x00, Data=0x01)
2. Wait for "presence" pulse
   Wait for INTR=1, then check to see if PD=1.
   To do so, read a byte at addr=0x02, and check Data[0]

Initializing_Slave

1. Send Reset Pulse
2. Master Tx "skip ROM" commd (Addr=0x00, Data=0xCC)
3. Wait for INTR=1, then check for TBE=1 (Addr=0x02, Data[2]) <--necessary???

Read_Byte

1. Addr=0x01, Data=0xFF, nWR <= '0' (Must write 0xFF first in order to read)
2. Wait for INTR=1, then check to see if TBE=1 (Data[2] at addr=0x02)
3. Addr=0x01, nRD <= '0', Data <= (data)

Read_ROM

1. Initizlize_Slave
2. Master Tx "read memory" command followed by address bytes
   2a. Addr=0x00, Data=0xF0, nWR <='0'
   2b. Addr=0x01, Data <= Address byte1, nWR <= '0'
   2c. Addr=0x01, Data <= Address byte2, nWR <= '0'
3. Read Byte Data <= CRC
4. Check CRC
5. Master "issues read time slots"
6. Read Byte (as many times as necessary)
7. Send Reset Pulse

Read_Status
This is the same as **Read_ROM** except for the command byte.
1. Initialize_Slave
2. Master Tx "read status" command followed by address bytes
    2a. Addr=0x00, Data=0xAA, nWR <='0'
    2b. Addr=0x01, Data <= Address byte1, nWR <= '0'
    2c. Addr=0x01, Data <= Address byte2, nWR <= '0'
3. Read Byte Data <= CRC
4. Check CRC
5. Master "issues read time slots"
6. Read Byte (as many times as necessary)
7. Send Reset Pulse

Write_ROM
1. Initialize Slave
2. Master Tx "Write ROM" command followed by two address bytes and a data byte
    2a. Addr=0x00, Data=0x0F, nWR <='0'
    2b. Addr=0x01, Data <= Address byte1, nWR <= '0'
    2c. Addr=0x01, Data <= Address byte2, nWR <= '0'
    2d. Addr=0x01, Data <= data byte, nWR <= '0'
3. Read Byte Data <= CRC
4. Check CRC
5. Send a programming pulse (12V)
6. Read Byte
7. Verify byte is correct
8. if write more bytes, goto (2d)
9. Send Reset Pulse

Write_Status
Same as Write_ROM except for the command byte
1. Initialize Slave
2. Master Tx "Write status" command followed by two address bytes and a data byte
    2a. Addr=0x00, Data=0x55, nWR <='0'
    2b. Addr=0x01, Data <= Address byte1, nWR <= '0'
    2c. Addr=0x01, Data <= Address byte2, nWR <= '0'
    2d. Addr=0x01, Data <= status byte, nWR <= '0'
3. Read Byte Data <= CRC
4. Check CRC
5. Send a programming pulse (12V)
6. Read Byte
7. Verify byte is correct
8. if write more bytes, goto (2d)
9. Send Reset Pulse

Read_ID
1. Send Reset Pulse
2. Addr=0x00, Data=0x33, nWR <= '0'
3. Read Byte (Familiy code)
4. Read Byte (6 times)
5. Read Byte (CRC)

SPI Functions

Up to eight(8) SPI devices are supported:
nCS0..nCS5 : DAC channels
nCS6..nCS7 : (no device assignment)

SPI_Select_Active_Channel
DOMMB writes a word to **addr_spi_active_list**. SPI channels whose corresponding bit location is a 1 in the word are selected.

SPI_Mode_Start
When DOMMB writes 0xFF to **addr_spi_operate**, the "**SPI_mode**" is started. In SPI_mode, nWR and data(0..2) are used for a special function:

Signal name	Function
nWR	SCLK
data(0)	MOSI
data(1)	MISO
data(2)	nCS

When nCS goes low, all the chip-select lines corresponding to an active SPI channel also go low.

SPI_Mode_End
When DOMMB writes 0x00 to **addr_spi_operate**, the SPI_mode is terminated.

Adjustable Trigger Delay

Each one of the six LED modules, LED_module0..LED_module5, has an adjustable trigger delay device external to the CPLD. The CPLD generates a 3-bit code value and a 1-bit enable signal for each module.

DEL_select_module
DOMMB writes a word specifying which LED module is to be enabled to **addr_del_ena**.

DEL_adjust_module_01
DOMMB writes a word to address **addr_del_dat0**. Bits 0..2 are for LED_module0, bits 3..5 are for LED_module1.

DEL_adjust_module_23
DOMMB writes a word to address **addr_del_dat1**. Bits 0..2 are for LED_module2, bits 3..5 are for LED_module3.

DEL_adjust_module_45
DOMMB writes a word to address **addr_del_dat2**. Bits 0..2 are for LED_module4, bits 3..5 are for LED_module5.