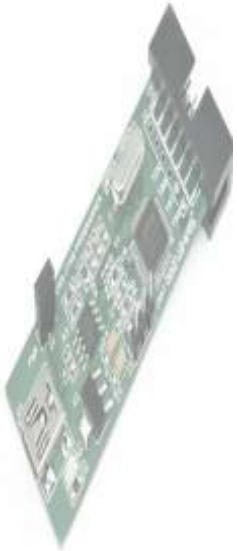
Spartan 6 FPGA Board

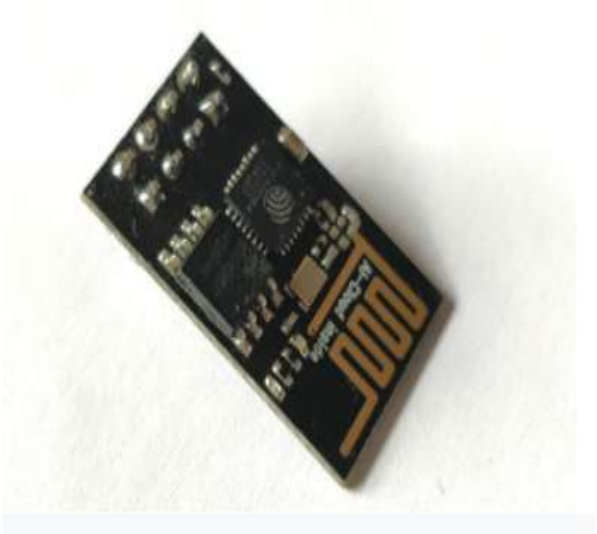
- Xilinx’s Spartan®-6 XC6SLX16-2FTG256C
- MT41J128M16JT-25(DDR3)
- 50MHz Clock
- M25P80 SPI Flash

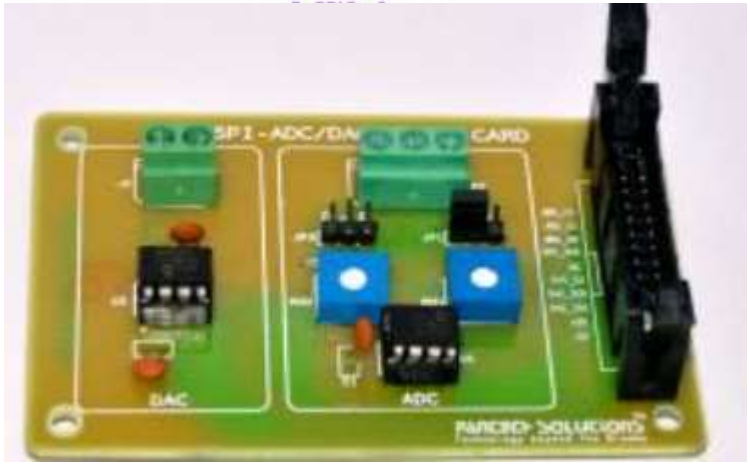
Board Specifications

- On-Board FPGA: XC6SLX16-2FTG256C;
- On-Board FPGA external crystal frequency: 50MHz;
- XC6SLX16-2FTG256C has rich block RAM resource up to 576Kb
- XC6SLX16-2FTG256C has 14,579 logic cells;
- On-Board M25P80 SPI Flash，1M bytes for user configuration code;
- On-Board 256MB Micron DDR3, MT41J128M16JT-125
- On-Board 3.3V power supply for FPGA by using MP2359 wide input range DC/DC
- XC6SLX16 development board has two 64p, 2.54mm pitch headers for extending user IOs.
- All IOs are precisely designed with length matching
- XC6SLX16 development board has 3 user switches
- XC6SLX16 development board has 4 user LEDs;
- XC6SLX16 development board has JTAG interface, by using 6p, 2.54mm pitch header;
- XC6SLX16 development board PCB size is: 6.7cm x 8.4cm;
- Default power source for board is: 1A@5V DC, the DC header type: DC-050, 5.5mmx2.1mm

JTAG Interface –To Program FPGA ,WIFI, SPI ADC



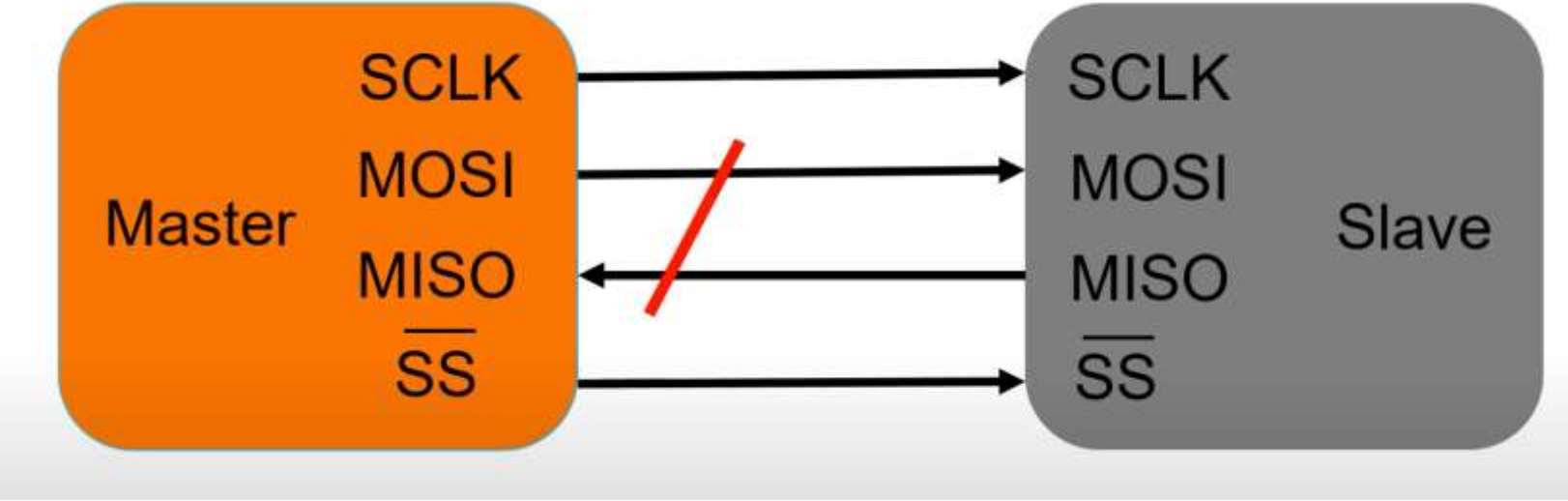




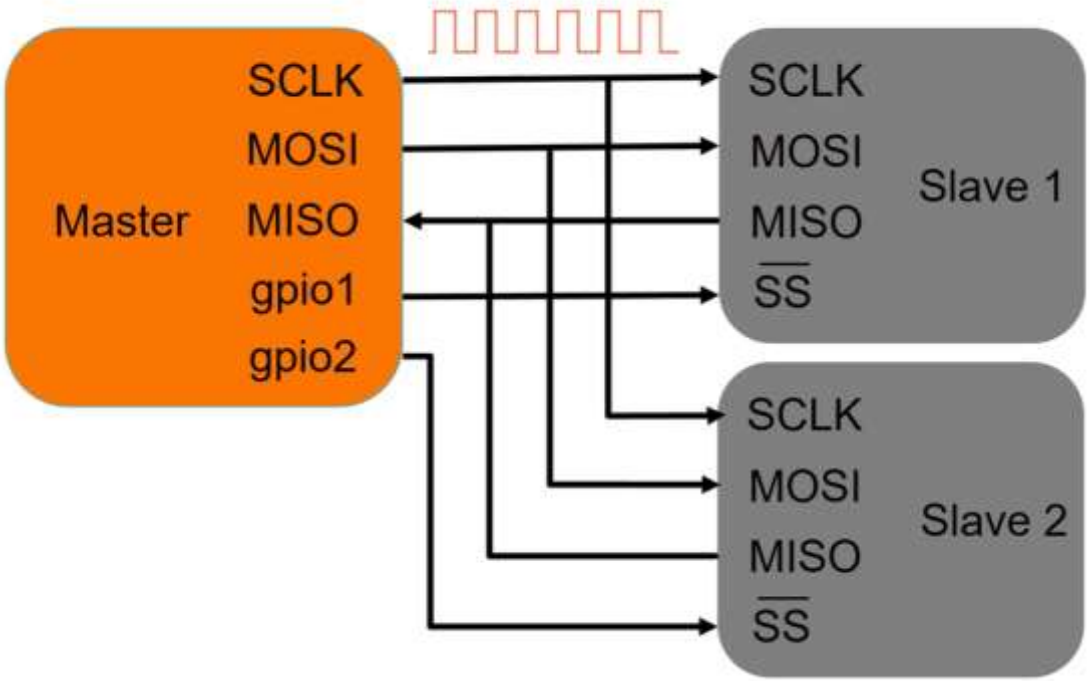
Introduction to SPI

- ✓ **Serial Peripheral Interface**
- ✓ Full-Duplex, Serial Communication Protocol.
- ✓ Synchronous Communication Protocol.
- ✓ Four Wire Communication Protocol.
- ✓ Single-master Multi-slave.
- ✓ Widely used for short-distance communication, primarily in Embedded System.

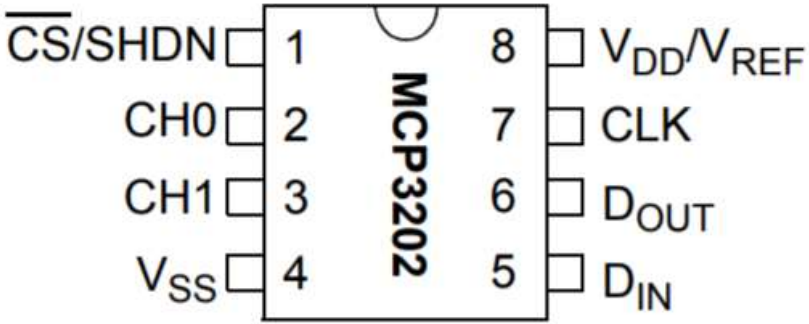
Master and Slave Mode



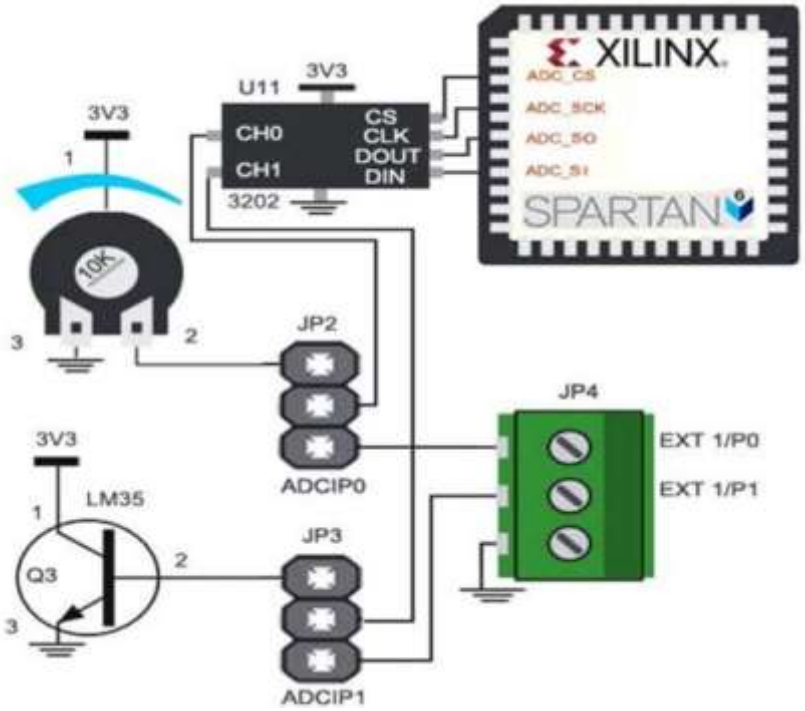
Master and Slave Mode –Multiple Slave



MCP3202 ADC Interface



FPGA Interface with ADC

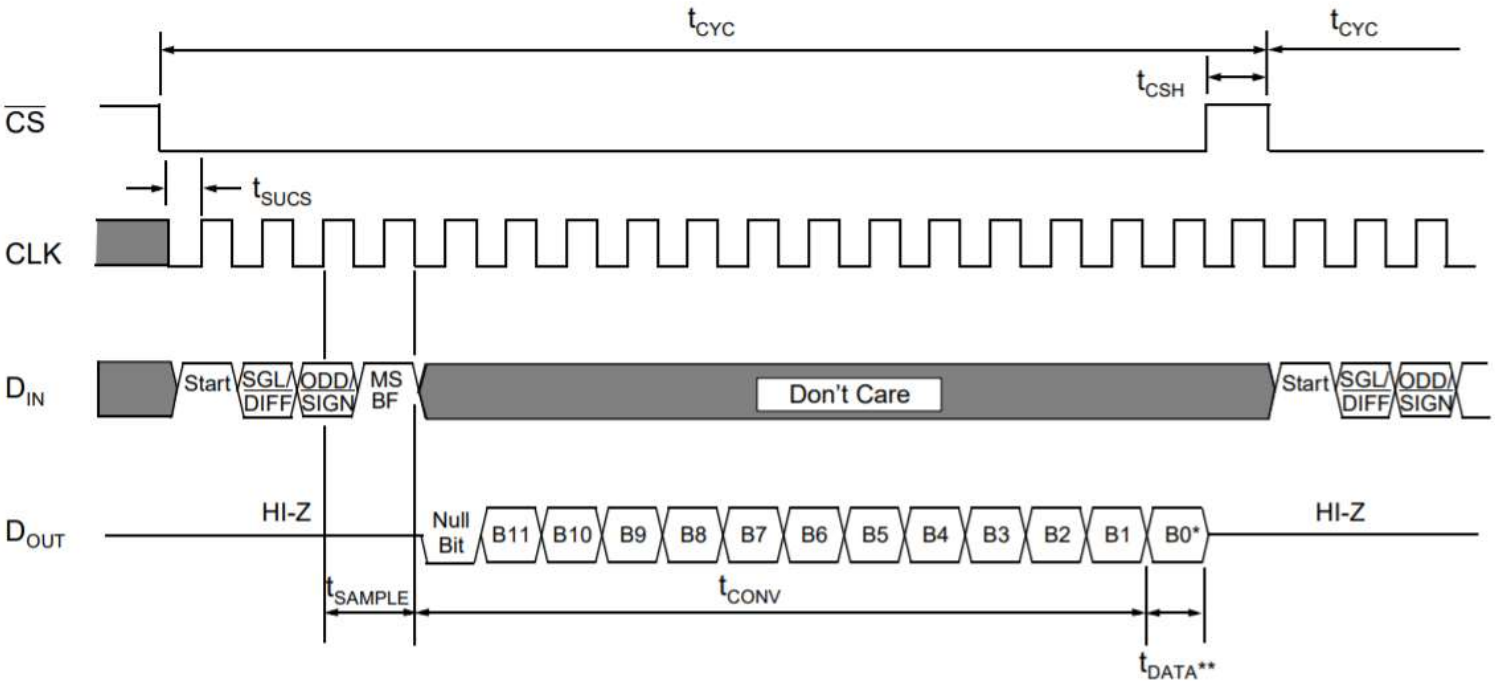


Configuration Bits for MCP3202

	CONFIG BITS		CHANNEL SELECTION		GND
	SGL/$\overline{\text{DIFF}}$	ODD/$\overline{\text{SIGN}}$	0	1	
SINGLE ENDED MODE	1	0	+		-
	1	1		+	-
PSEUDO-DIFFERENTIAL MODE	0	0	IN+	IN-	
	0	1	IN-	IN+	

TABLE 5-1: *Configuration Bits for the MCP3202.*

Timing Waveform



Timing Waveform

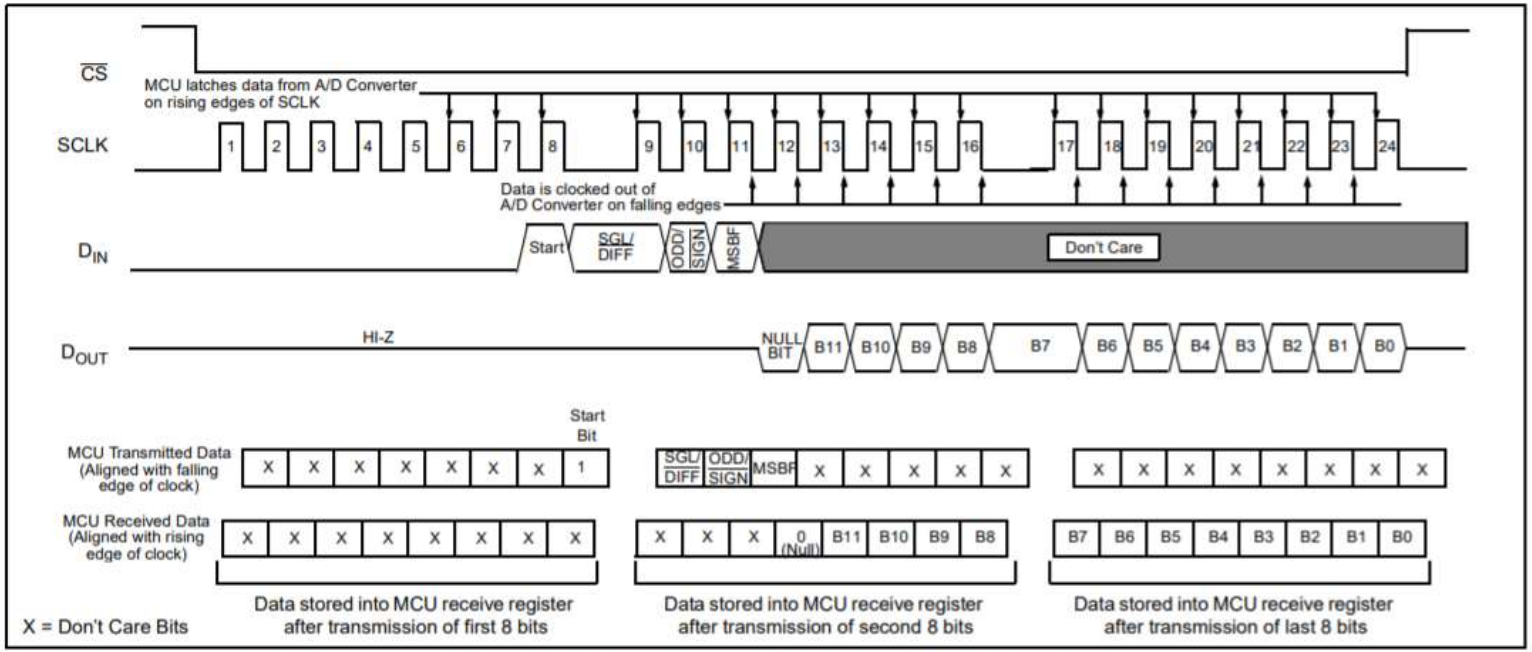


FIGURE 6-1: *SPI Communication using 8-bit segments (Mode 0,0: SCLK idles low).*

Timing Waveform

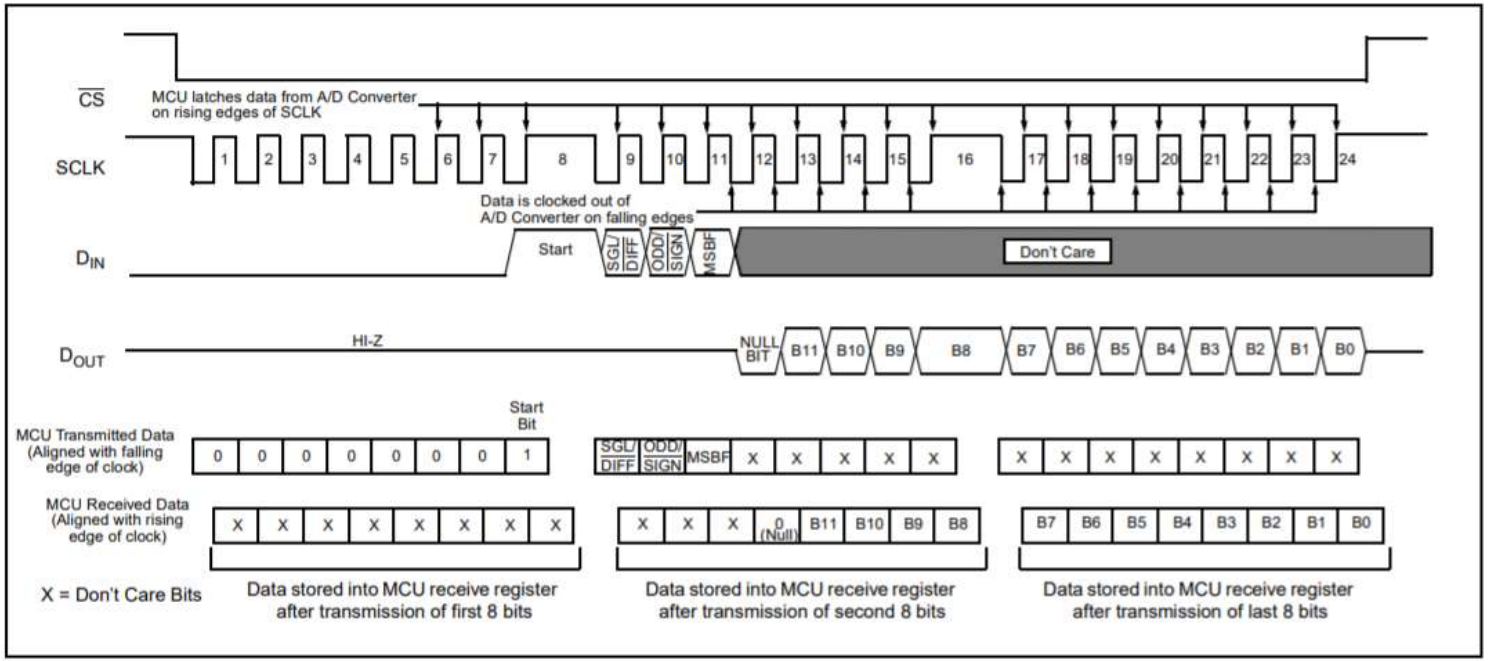


FIGURE 6-2: *SPI Communication using 8-bit segments (Mode 1,1: SCLK idles high).*

Calculation for Digital Output Code

4.2 Digital Output Code

The digital output code produced by an A/D Converter is a function of the input signal and the reference voltage. For the MCP3202, V_{DD} is used as the reference voltage. As the V_{DD} level is reduced, the LSB size is reduced accordingly. The theoretical digital output code produced by the A/D Converter is shown below.

$$\text{Digital Output Code} = \frac{4096 * V_{IN}}{V_{DD}}$$

where:

V_{IN} = *analog input voltage*

V_{DD} = *supply voltage*

AT COMMANDS FOR TRANSMISSION

-- AT commands used for data tramission
-- AT+RST
-- AT+CWJAP="WIFINAME","Password"
-- AT+CIPSTART="TCP","184.106.153.149",80
-- AT+CIPSEND=49
-- GET /update?api_key=XXXXXXXXXXXXXXX&field1=000
-- AT+CLOSE

-- TCP address is from thingspeak.com. You have register it to recieve api key to send the data to cloud id.

Library and Port Declaration

```
--Library Declaration
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
--Port Declaration
entity IoT_Temperature_Monitoring is
    Port (    clk               : in  STD_LOGIC;
                                                    txd             : out  STD_LOGIC;
                                                    cs    : out std_logic;
                                                    sc              : out std_logic;
                                                    do    : out std_logic;
                                                    din   : in std_logic
                                            );
end IoT_Temperature_Monitoring;
```

Signals Declaration

--------Signals for SPI Communication----------

```
type state is (spi,conversion);
signal presentstate : state := spi;
```

-------Signal for Temperature Sensor or Potmeter------------

```
signal temp1:integer:=0;
```

-------Signals for Binary to BCD and BCD to ASCII---------

```
type reg2 is array(1 to 3) of std_logic_vector(7 downto 0);
signal arr_rp1 :reg2;
signal siga,sigb: std_logic_vector(2 downto 0):=(others => '0');
```

Signals for UART Transmission

```
type stat1 is (ready2,start2,stop2);
signal ps2 :stat1 := ready2;
signal start,stop :std_logic;
signal store :std_logic_vector(7 downto 0);
signal baud_clk : std_logic;

type arr is array (1 to 160) of std_logic_vector(7 downto 0);
```

AT commands array Declaration

--AT+RST

constant str : arr := (X"41",X"41",X"54",X"2b",X"52",X"53",X"54",X"0d",X"0a", --9

---AT+CWJAP="WIFINAME","Password"

x"41",x"54",x"2b",x"43",x"57",x"4a",x"41",x"50",x"3d",x"22",x"5a",x"54",x"45",x"2d",x"61",x"64",x"64",x"65",x"37",x"37",x"22",
x"2c",x"22",x"38",x"38",x"35" ,x"64",x"66",x"62",x"61",x"64",x"22",X"0d",X"0a", --34

---AT+CIPSTART="TCP","184.106.153.149",80

X"41",X"54",X"2b",X"43",X"49",X"50",X"53",X"54",X"41",X"52",X"54",X"3D",X"22",X"54",X"43",X"50",X"22",X"2C",X"22",
X"31",X"38",X"34",X"2E",X"31",X"30",X"36",X"2E",X"31",X"35",X"33",X"2E",X"31",X"34",X"39",X"22",X"2C",X"38",X"30",X"0d",X"0a",
--40

---AT+CIPSEND=60

X"41",X"54",X"2b",X"43",X"49",X"50",X"53",X"45",X"4e",X"44",X"3d",X"36",X"30",X"0d",X"0a", --15

X"47",X"45",X"54",X"20",X"2F",X"75",X"70",X"64",X"61",X"74",X"65",X"3F",X"61",X"70",X"69",X"5F",X"6B",X"65",X"79",X"3D",

--ZMERJO9Q36JZ3I63 –Write api key

x"37",x"53",x"34",x"46",x"50",x"41",x"41",x"36",x"4e",x"47",x"53",x"4b",x"31",x"32",x"58",x"30",

--GET /update?api_key=ZMERJO9Q36JZ3I63&field1=000

X"26",X"66",X"69",X"65",X"6C",X"64",X"31",X"3D",X"30",X"31",X"32",X"0d",X"0a", --49

X"41",X"54",X"2b",X"43",X"49",X"50",X"43",X"4c",X"4f",X"53",X"45",X"0d",X"0a"); --13

SPI ADC PROCESS

```
process(clk)
variable i,j,k : integer := 0;
variable tot : std_logic_vector(12 downto 0) := "0000000000000";
variable temp : std_logic_vector(23 downto 0) := "000000000000000000000000";
begin
if rising_edge(clk) then
                if presentstate = spi then
                        if i <= 50 then
                                i := i + 1;
                                sc <= '1';
                        elsif i > 50 and i < 100 then
                                i := i + 1;
                                sc <= '0';
                        elsif i = 100 then
                                i := 0;
                                if j < 19 then
                                        j := j + 1;
                                elsif j = 19 then
                                        presentstate <= conversion;
                                        j := 0;
                                end if;
                        end if;
```

Channel selection

```
if j = 0 or j >= 18 then
                cs <= '1';
else
                cs <= '0';
end if;

if i > 40 and i < 60 then
case j is
                when 0 =>
                                do <= '0';
                when 1 =>
                                do <= '1';
                when 2 =>
                                do <= '1';
                when 3 =>
                                do <= '1';        -----channel bit
                when 4 =>
                                do <= '1';
                when 5 =>
                                do <= '1';
                when others =>
                                null;
end case;
end if;
```

DATA RECEIVED

```
if i >= 0 and i < 10 then
                                                    case j is
                                                                when 6 =>
                                                                tot(12) := din;
                                                                when 7 =>
                                                                tot(11) := din;
                                                                when 8 =>
                                                                tot(10) := din;
                                                                when 9 =>
                                                                tot(9) := din;
                                                                when 10 =>
                                                                tot(8) := din;
                                                                when 11 =>
                                                                tot(7) := din;
                                                                when 12 =>
                                                                tot(6) := din;
                                                                when 13 =>
                                                                tot(5) := din;
                                                                when 14 =>
                                                                tot(4) := din;
                                                                when 15 =>
                                                                tot(3) := din;
                                                                when 16 =>
                                                                tot(2) := din;
                                                                when 17 =>
                                                                tot(1) := din;
                                                                when 18 =>
                                                                tot(0) := din;
                                                                when others =>
                                                                null;
                                                    end case;
                                        end if;
                            end if;
```

TEMPERATURE CONVERSION

```
-------------------------------------------------------------
                if presentstate = conversion then
                                cs <= '1';
                                temp1 <= (conv_integer(tot(11 downto 0)) * 330)/ 4096;  -------- Temperature Calculation
                                presentstate <= spi;
                end if;
end if;
end process;
```

PROCESS ADC CONVERSION –BINARY TO ASCII

```
process (clk)
                variable q1 : integer range 0 to 100 := 0;
                variable p1,p2,p3,p4 : integer range 0 to 10 := 0;
begin
if rising_edge(clk) then
                sigb <= sigb + 1;
                case sigb is
                                when "000" =>
                                  q1 := temp1;
                                p1 := 0;
                                  p2 := 0;
                                  p3 := 0;
                                                when "001" =>
                                                if (q1 >= 100) then
                                                                q1 := q1 - 100;
                                                                p1 := p1 + 1;
                                                                sigb <= "001";
                                                elsif (q1 >= 10) then
                                                                q1 := q1 - 10;
                                                                p2 := p2 + 1;
                                                                sigb <= "001";
                                                else
                                                                p3 := q1;
                                                end if;

                                when "010" =>
                                arr_rp1(1) <= conv_std_logic_vector (p1, 8) + x"30";
                                when "011" =>
                                arr_rp1(2) <= conv_std_logic_vector (p2, 8) + x"30";
                                when "100" =>
                                arr_rp1(3) <= conv_std_logic_vector (p3, 8) + x"30";
                                when others =>
                                                                sigb <= "000";
                end case;
end if;
end process;
```

Process baud clk generation

```
process(clk)
--------------------50 x 10^6 / 16*115200 = 27
variable baud_count : integer range 0 to 27 := 0;
begin
if rising_edge(clk) then
if baud_count = 27 then
baud_clk <= '1';
baud_count := 0;
else
baud_count := baud_count + 1;
baud_clk <= '0';
end if;
end if;
end process;
```

```
process(baud_clk)
variable i,k : integer := 0;
variable j : integer := 0;
begin
if rising_edge(baud_clk)then
if ps2 = ready2 then
i := i + 1;
if i = 8 then
                                    txd <= '0';
                                    i := 0;
                                    ps2 <= start2;
 end if;
end if;
```

```
----------16xbaudrate sampling mathod----------
if ps2 = start2 then
i := i + 1;

if j = 144 then
store <= arr_rp1(1);
--store <= x"31";
elsif j = 145 then
store <= arr_rp1(2);
--store <= x"32";
elsif j = 146 then
store <= arr_rp1(3);
--store <= x"33";
else
store <= str(j)(7 downto 0);
end if;
```

```
if i = 16 then
txd <= store(0);
end if;

if i = 32 then
txd <= store(1);
end if;

if i = 48 then
txd <= store(2);
end if;

if i = 64 then
txd <= store(3);
end if;

if i = 80 then
txd <= store(4);
end if;

if i = 96 then
txd <= store(5);
end if;

if i = 112 then
txd <= store(6);
end if;

if i = 128 then
txd <= store(7);
end if;

if i = 144 then
txd <= '1';
end if;

if i = 160 then
i := 0;
ps2 <= stop2;
end if;
```

```
elsif ps2 = stop2 then
if j = 9 then
                    ps2 <= stop2;
                    k := k + 1;
                    if k = 10000000 then
                    j := j + 1;
                    k := 0;
                    ps2 <= ready2;
                    end if;
elsif j = 43 then
                    ps2 <= stop2;
                    k := k + 1;
                    if k = 50000000 then
                    j := j + 1;
                    k := 0;
                    ps2 <= ready2;
                    end if;
elsif j = 83 then
                    ps2 <= stop2;
                    k := k + 1;
                    if k = 5000000 then
                    j := j + 1;
                    k := 0;
                    ps2 <= ready2;
                    end if;
```

```
elsif  j = 98 then
                        ps2 <= stop2;
                        k := k + 1;
                        if k = 5000000 then
                        j := j + 1;
                        k := 0;
                        ps2 <= ready2;
                        end if;
elsif  j = 147 then
                        ps2 <= stop2;
                        k := k + 1;
                        if k = 5000000 then
                        j := j + 1;
                        k := 0;
                        ps2 <= ready2;
                        end if;
elsif j = 160 then
                        ps2 <= stop2;
                        k := k + 1;
                        if k = 5000000 then
                        k := 0;
                        j := 43;
                        ps2 <= ready2;
                        end if;
else
                        j := j + 1;
                        ps2 <= ready2;
end if;


end if;
end if;
end process;
```