`ifndef APB\_BUS\_SV

`define APB\_BUS\_SV

`include "config.sv"

// SOC PERIPHERALS APB BUS PARAMETRES

`define NB\_MASTER 9

// MASTER PORT TO CVP

`define UART\_START\_ADDR 32'h1A10\_0000

`define UART\_END\_ADDR 32'h1A10\_0FFF

// MASTER PORT TO GPIO

`define GPIO\_START\_ADDR 32'h1A10\_1000

`define GPIO\_END\_ADDR 32'h1A10\_1FFF

// MASTER PORT TO SPI MASTER

`define SPI\_START\_ADDR 32'h1A10\_2000

`define SPI\_END\_ADDR 32'h1A10\_2FFF

// MASTER PORT TO TIMER

`define TIMER\_START\_ADDR 32'h1A10\_3000

`define TIMER\_END\_ADDR 32'h1A10\_3FFF

// MASTER PORT TO EVENT UNIT

`define EVENT\_UNIT\_START\_ADDR 32'h1A10\_4000

`define EVENT\_UNIT\_END\_ADDR 32'h1A10\_4FFF

// MASTER PORT TO I2C

`define I2C\_START\_ADDR 32'h1A10\_5000

`define I2C\_END\_ADDR 32'h1A10\_5FFF

// MASTER PORT TO FLL

`define FLL\_START\_ADDR 32'h1A10\_6000

`define FLL\_END\_ADDR 32'h1A10\_6FFF

// MASTER PORT TO SOC CTRL

`define SOC\_CTRL\_START\_ADDR 32'h1A10\_7000

`define SOC\_CTRL\_END\_ADDR 32'h1A10\_7FFF

// MASTER PORT TO DEBUG

`define DEBUG\_START\_ADDR 32'h1A11\_0000

`define DEBUG\_END\_ADDR 32'h1A11\_7FFF

`define APB\_ASSIGN\_SLAVE(lhs, rhs) \

assign lhs.paddr = rhs.paddr; \

assign lhs.pwdata = rhs.pwdata; \

assign lhs.pwrite = rhs.pwrite; \

assign lhs.psel = rhs.psel; \

assign lhs.penable = rhs.penable; \

assign rhs.prdata = lhs.prdata; \

assign rhs.pready = lhs.pready; \

assign rhs.pslverr = lhs.pslverr;

`define APB\_ASSIGN\_MASTER(lhs, rhs) `APB\_ASSIGN\_SLAVE(rhs, lhs)

////////////////////////////////////////////////////////////////////////////////

// Only general functions and definitions are defined here //

// These functions are not intended to be modified //

////////////////////////////////////////////////////////////////////////////////

interface APB\_BUS

#(

parameter APB\_ADDR\_WIDTH = 32,

parameter APB\_DATA\_WIDTH = 32

);

logic [APB\_ADDR\_WIDTH-1:0] paddr;

logic [APB\_DATA\_WIDTH-1:0] pwdata;

logic pwrite;

logic psel;

logic penable;

logic [APB\_DATA\_WIDTH-1:0] prdata;

logic pready;

logic pslverr;

// Master Side

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

modport Master

(

output paddr, pwdata, pwrite, psel, penable,

input prdata, pready, pslverr

);

// Slave Side

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

modport Slave

(

input paddr, pwdata, pwrite, psel, penable,

output prdata, pready, pslverr

);

endinterface

`endif