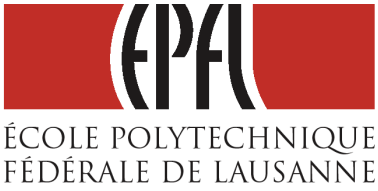


CAMERA EMBARQUEE ETHERNET / INTERNET

ANNEXES

Projet de semestre de

Sylvain Pasini

Professeur : Pr. Paulo Ienne

Présenté le 10 février 2003

Responsable : René Beuchat

Table des matières

1. Organisation des annexes

Afin de comprendre l'utilité de chaque fichier, le schéma ci-dessous a été réalisé. Il permet surtout de comprendre à quelle partie du projet s'attache le code en question :

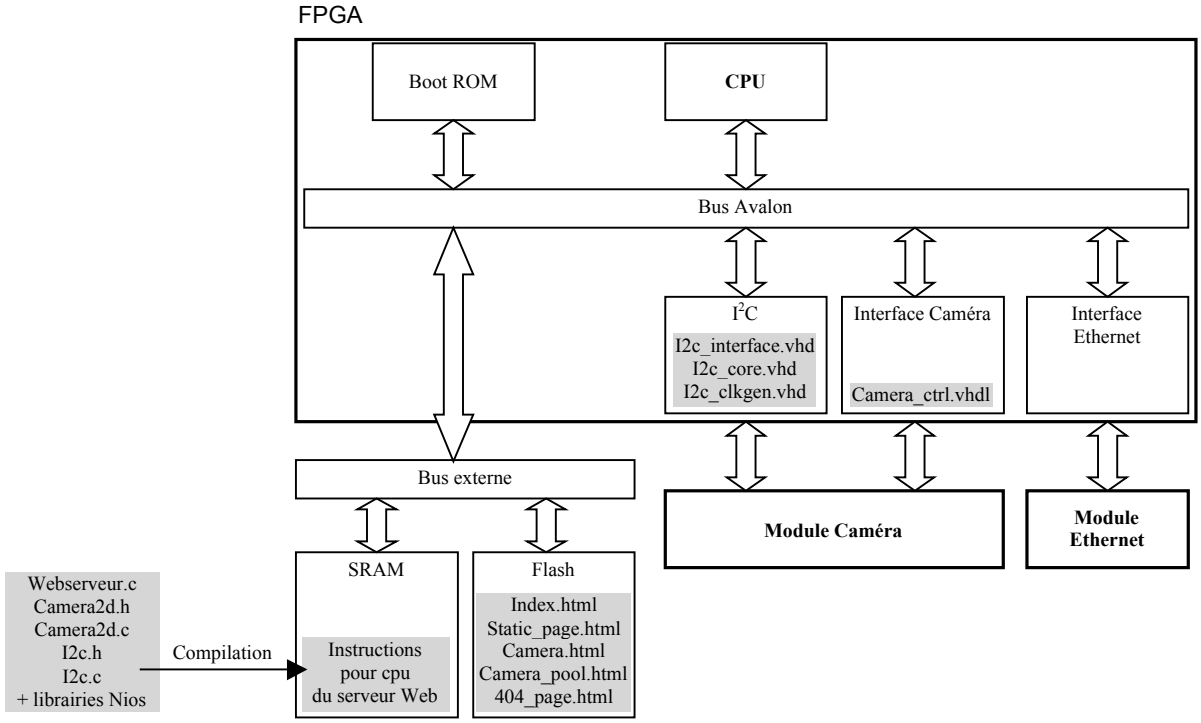
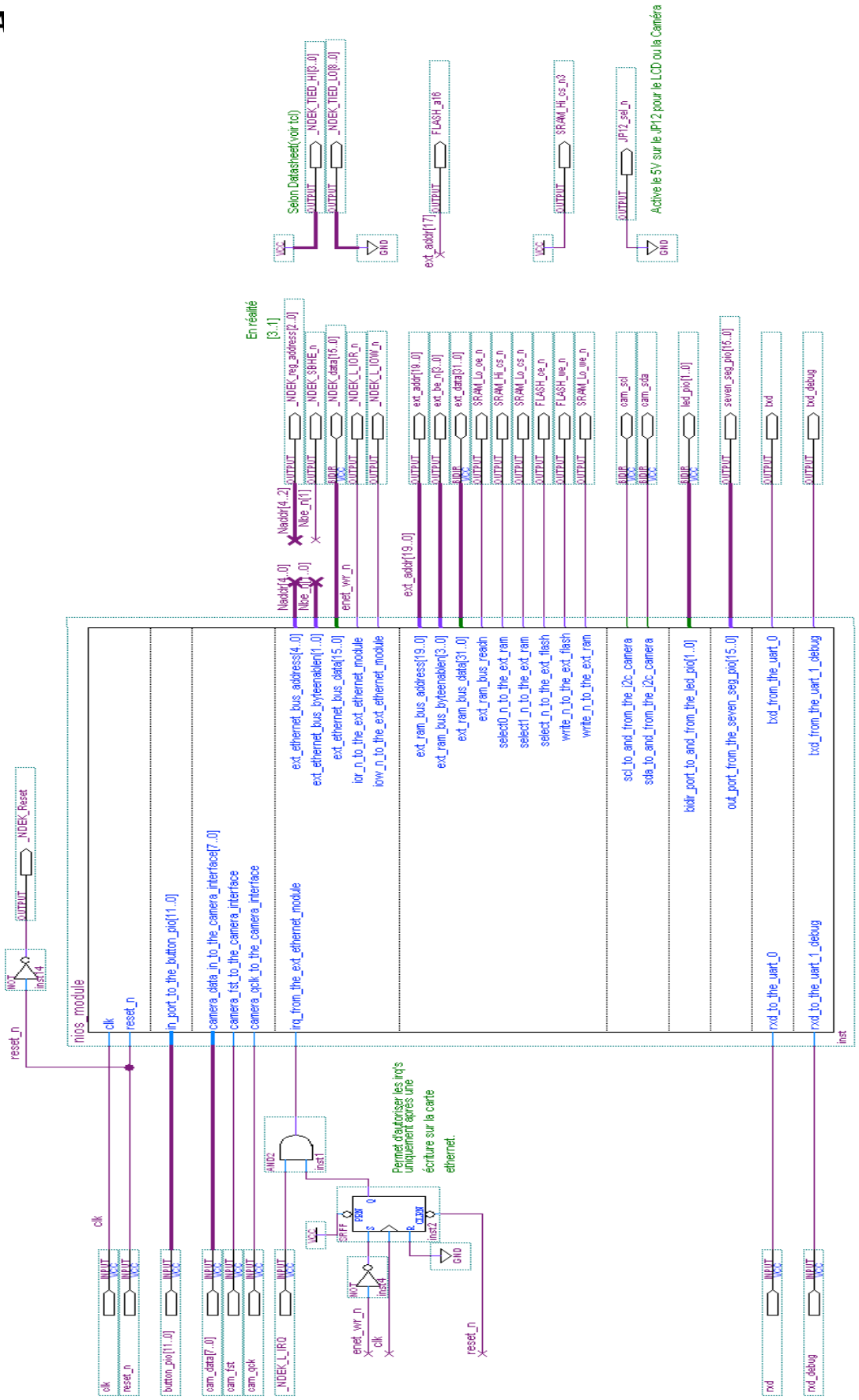


Figure : Schéma représentant l'importance de chaque code

2. Architecture générale pour la carte Excalibur

A



Assignement des pins

```
##############################################################################
# pin_assign.tcl
##############################################################################
# This script allows you to make pin assignments to the Nios tutorial design
#
# You can run this script from Quartus by observing the following steps:
# 1. Place this TCL script in your project directory
# 2. Open your project
# 3. Go to the View Menu and Auxilary Windows -> TCL console
# 4. In the TCL console type:
#                                                  source pin_assign_ethernet.tcl
# 5. The script will assign pins and return an "assignment made" message.
##############################################################################


################ Open a Project if one does not yet exist ####################
set project_name nios_ethernet
set top_name nios_ethernet

if { ![project exists ./$project_name] } {
        project create ./$project_name
}
project open ./$project_name

set cmp_settings_group $top_name
if { ![project cmp_exists $cmp_settings_group] } {
       project create_cmp $top_name
}
project set_active_cmp $top_name

cmp add_assignment $top_name "" "" DEVICE EP20K200EFC484-2X

##############################################################################

###################### Set the pin location variables ########################

#Convention :
#       bit de poids faible d'abord
#       exemple : a<0> a<1> a<2> (-> a(0) de la mémoire) ... a<17> (-> a(15) de la mémoire)


#Horloges
        set clk L6

#Logidule : reset
        set reset_n F12

#Communication série
        set rxd W8
        set txd D15
        set rxd_debug F14
        set txd_debug F13

#Signaux d'interface avec la mémoire SRAM
        set ext_addr {G17 A8 B8 A7 B7 B6 A6 A5 B5 B4 A3 A4 C3 C1 D3 D2 C2 D1 B3 E3}
        set ext_data {C4 H11 G10 D8 E7 D4 D5 G9 F8 E8 C5 D6 C6 F9 H10 D7 C7 E9 E10 D9 C8 F10 G11 C9 C10 H12 D10 G12
G13 F11 B11 B10}

        set SRAM_Lo_oe_n A2
        set SRAM_Lo_we_n E6
        set SRAM_Hi_cs_n E2
        set SRAM_Hi_cs_n3 H4
        set SRAM_Lo_cs_n E4

        set ext_be_n {F5 F2 F4 H5}
        #set ext_addr_unused {B17 Y11 L15 C19 J19}
        set FLASH_ce_n E1
        set FLASH_we_n H13
        set FLASH_a16 F3

        set JP12_sel_n V7

#Port parallèle
        set button_pio {Y9 T9 Y8 W9 V9 U9 T10 U10 V10 P11 U12 Y10}
                        #(0..3)->SW4..SW7 (4..11) -> Registre 8 bits
        set led_pio {T18 T19}
        set seven_seg_pio {W17 U18 Y18 W18 U8 T11 R10 C18 V17 V18 Y17 V8 Y7 U11 R11 D18}

     #Uniquement si Caméra non connectée
        #set lcd_pio {U21 P17 U1 U2 T2 T3 U4 U19 R18 W20 N15}

#Vers Module Ethernet
        set _NDEK_Reset M16
        set _NDEK_L_IRQ J1
        set _NDEK_reg_address {N20 K20 K22}
                              #addr EDK[4..2] -> Nios[3..1]
        set _NDEK_SBHE_n M6
        set _NDEK_data {L20 J18 K18 M17 N18 M20 L16 R21 N6 J3 K5 R4 J7 J5 K3 N2}
        set _NDEK_L_IOR_n N5
```

```
        set _NDEK_L_IOW_n L7
        set _NDEK_TIED_HI {K4 P4 P21 K16}
                          # MEMW MEMR A8  A9                              ( selon docs: ethernet p.26 )
        set _NDEK_TIED_LO {R5 N21 L17 K19 P22 N22 R22 P20 K15}
                          #A0 CSh CS1 A4  A5  A6  A7  A10 A11            ( selon docs: ethernet p.26 )

#Vers Module I2c de la caméra
        set cam_scl P19
        set cam_sda L15

#Vers Module Caméra
        set cam_data { W20 N15 P17 U2 T3 U19 U4 T2 }
        set cam_fst R18
        set cam_qck U21


#############################################################################
################ Make the clock and reset signal assignments ################

#Horloges
        cmp add_assignment $top_name "" clk LOCATION "Pin_$clk"

#Logidule : reset
        cmp add_assignment $top_name "" reset_n LOCATION "Pin_$reset_n"

#Communication série
        cmp add_assignment $top_name "" rxd LOCATION "Pin_$rxd"
        cmp add_assignment $top_name "" txd LOCATION "Pin_$txd"
        cmp add_assignment $top_name "" rxd_debug LOCATION "Pin_$rxd_debug"
        cmp add_assignment $top_name "" txd_debug LOCATION "Pin_$txd_debug"

#Signaux d'interface avec la mémoire SRAM
        set i 0
        foreach {a} $ext_addr {
                cmp add_assignment $top_name "" "ext_addr\[$i\]" LOCATION "Pin_$a"
                set i [expr $i+1]
        }
        #set i 0
        #foreach {a} $ext_addr_unused {
        #       cmp add_assignment $top_name "" "ext_addr_unused\[$i\]" LOCATION "Pin_$a"
        #       set i [expr $i+1]
        #}
        set i 0
        foreach {a} $ext_data {
                cmp add_assignment $top_name "" "ext_data\[$i\]" LOCATION "Pin_$a"
                set i [expr $i+1]
        }
        cmp add_assignment $top_name "" SRAM_Lo_oe_n LOCATION "Pin_$SRAM_Lo_oe_n"
        cmp add_assignment $top_name "" SRAM_Lo_we_n LOCATION "Pin_$SRAM_Lo_we_n"
        cmp add_assignment $top_name "" SRAM_Hi_cs_n LOCATION "Pin_$SRAM_Hi_cs_n"
        cmp add_assignment $top_name "" SRAM_Hi_cs_n3 LOCATION "Pin_$SRAM_Hi_cs_n3"
        cmp add_assignment $top_name "" SRAM_Lo_cs_n LOCATION "Pin_$SRAM_Lo_cs_n"
        set i 0
        foreach {a} $ext_be_n {
                cmp add_assignment $top_name "" "ext_be_n\[$i\]" LOCATION "Pin_$a"
                set i [expr $i+1]
        }
        cmp add_assignment $top_name "" FLASH_ce_n LOCATION "Pin_$FLASH_ce_n"
        cmp add_assignment $top_name "" FLASH_we_n LOCATION "Pin_$FLASH_we_n"
        cmp add_assignment $top_name "" FLASH_a16 LOCATION "Pin_$FLASH_a16"
        cmp add_assignment $top_name "" JP12_sel_n LOCATION "Pin_$JP12_sel_n"

#Port parallèle
        set i 0
        foreach {a} $button_pio {
                cmp add_assignment $top_name "" "button_pio\[$i\]" LOCATION "Pin_$a"
                set i [expr $i+1]
        }
        set i 0
        foreach {a} $led_pio {
                cmp add_assignment $top_name "" "led_pio\[$i\]" LOCATION "Pin_$a"
                set i [expr $i+1]
        }
        set i 0
        foreach {a} $seven_seg_pio {
                cmp add_assignment $top_name "" "seven_seg_pio\[$i\]" LOCATION "Pin_$a"
                set i [expr $i+1]
        }
        #set i 0
        #foreach {a} $lcd_pio {
        #       cmp add_assignment $top_name "" "lcd_pio\[$i\]" LOCATION "Pin_$a"
        #       set i [expr $i+1]
        #}

#Vers Module Ethernet
        cmp add_assignment $top_name "" _NDEK_Reset LOCATION "Pin_$_NDEK_Reset"
        cmp add_assignment $top_name "" _NDEK_L_IRQ LOCATION "Pin_$_NDEK_L_IRQ"
        set i 0
        foreach {a} $_NDEK_reg_address {
                cmp add_assignment $top_name "" "_NDEK_reg_address\[$i\]" LOCATION "Pin_$a"
                set i [expr $i+1]
        }
        cmp add_assignment $top_name "" _NDEK_SBHE_n LOCATION "Pin_$_NDEK_SBHE_n"
```

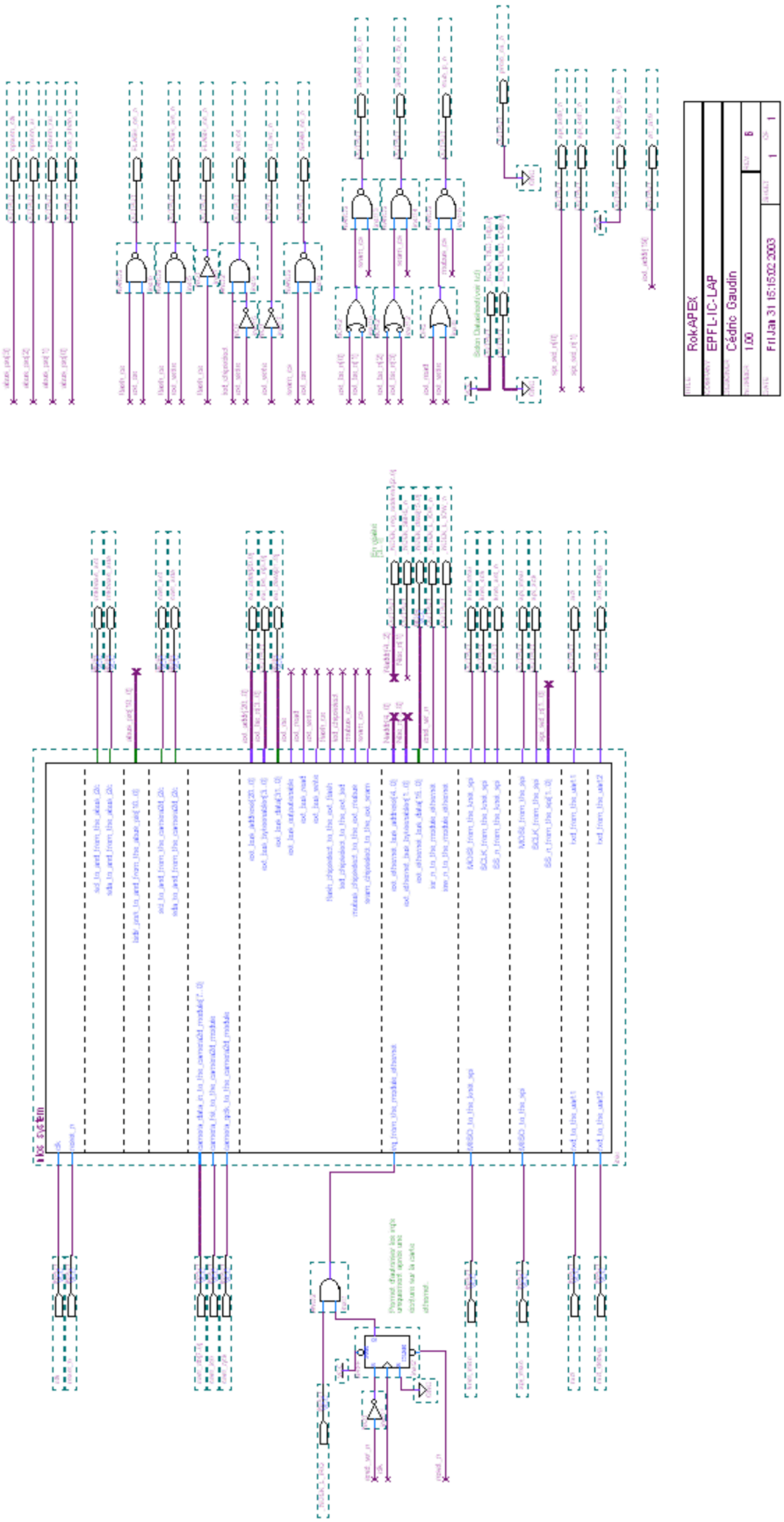
```
        set i 0
        foreach {a} $_NDEK_data {
                cmp add_assignment $top_name "" "_NDEK_data\[$i\]" LOCATION "Pin_$a"
                set i [expr $i+1]
        }
        cmp add_assignment $top_name "" _NDEK_L_IOR_n LOCATION "Pin_$_NDEK_L_IOR_n"
        cmp add_assignment $top_name "" _NDEK_L_IOW_n LOCATION "Pin_$_NDEK_L_IOW_n"
        set i 0
        foreach {a} $_NDEK_TIED_HI {
                cmp add_assignment $top_name "" "_NDEK_TIED_HI\[$i\]" LOCATION "Pin_$a"
                set i [expr $i+1]
        }
        set i 0
        foreach {a} $_NDEK_TIED_LO {
                cmp add_assignment $top_name "" "_NDEK_TIED_LO\[$i\]" LOCATION "Pin_$a"
                set i [expr $i+1]
        }


#Vers Module I2c de la caméra
        cmp add_assignment $top_name "" cam_scl LOCATION "Pin_$cam_scl"
        cmp add_assignment $top_name "" cam_sda LOCATION "Pin_$cam_sda"


#Vers Module Caméra
        set i 0
        foreach {a} $cam_data {
                cmp add_assignment $top_name "" "cam_data\[$i\]" LOCATION "Pin_$a"
                set i [expr $i+1]
        }
        cmp add_assignment $top_name "" cam_fst LOCATION "Pin_$cam_fst"
        cmp add_assignment $top_name "" cam_qck LOCATION "Pin_$cam_qck"
```

3. Architecture générale pour la carte RokApex

Architecture sous Quartus



Assignement des pins

```
#============================================================================
# pin_assign.tcl
#
# This script allows you to make pin assignments for the RokAPEX PCB
# using RokAPEX reference design
#
# Author   : Cédric Gaudin
# Revision : 1.0
# Date     : 27 june 2002
#
# You can run this script from Quartus by observing the following steps:
# 1. Place this TCL script in your project directory
# 2. Open your project
# 3. Go to the View Menu and Auxilary Windows -> TCL console
# 4. In the TCL console type:
#                                                 source pin_assign.tcl
# 5. The script will assign pins and return an "assignment made" message.
#============================================================================


################ Open a Project if one does not yet exist ####################
#set project_name
set top_name RokAPEX

#if { ![project exists ./$project_name] } {
#       project create ./$project_name
#}
#project open ./$project_name

#set cmp_settings_group $top_name
#if { ![project cmp_exists $cmp_settings_group] } {
#        project create_cmp $top_name
#}
#project set_active_cmp $top_name

#cmp add_assignment $top_name "" "" DEVICE EP20K200EFC484-2X

#============================================================================

########## Set the pin location variables

# Main Clock & Software Reset
set clk 31
# Apex_Clk1 - base clock
set reset_n 212
# Apex_Clear* - Software reset for the nios processor

# User Clock
set usr_clk0 151
set usr_clk1  34
set usr_clk2 154

# External BUS (on-board and some connectors)
set ext_addr { 202 201 198 197 196 194 193 192 191 190 187 186 184 183 182  99  98  96  95  94 136 135 }
set ext_data { 203 200 195 189 185 181 169 166 130 129 126 125 124 123 121 143 138 174 173 172 171 170 133 180 131
164 163 161 160 157 156 178 }
set ext_be_n { 105 104 103 102 }
set rd_wr_n  107

# 374 LED register
set led_cs 118

# Push-button
set usr_pb0 110
set usr_pb1 109

# SRAM memories
set a1_a19       134
set SRAM_cs_lo_n  91
set SRAM_cs_hi_n  88
set SRAM_oe_n    106

# FLASH memory (EPC16)
set FLASH_byte_n  204
set FLASH_we_n    205
set FLASH_ce_n    207
set FLASH_oe_n    206

# A-BUS connector
set rokbase_scl  117
set rokbase_sda  116
set odol_a        69
set odol_b        68
set odor_a        66
set odor_b        65
set spi_sel0_n    49
set adc_shdn_n   119
set pwml_pulse    63
set pwml_dir      64
```

```
set pwmr_pulse    61
set pwmr_dir      62
set buz          115
set optsen_clk    37
set optsen_rst    36
set optsen_si     35
set vdd_fault_n  114
set batt_fault_n 113
set wakeup       112
set pwr_en_n     111

# Extension connectors
# set proto_io { 239 238 237 236 235 234 233 232 231 230  228 227 226 225 224 223 222 221 220 219 217 216 215 25 24
23 22 21 20 18 17 16 13 11 10 9 8 7 4 3 }
set proto_cs_n 2
# enable CYBUS 3384 drivers when connected to GND

# MuBUS connector
set mub_p_n   101
# selection signal /P
set mub_irq_n 100
# interrupt request signal /IRQ

# Camera2D connector (Camera2D module)
set cam_io0   76
# FST
set cam_scl   73
# I2C clock
set cam_sda   72
# I2C data
set cam_qck   71
# QCK
set cam_io1   75
# QCKTRI
set cam_io2   74
# SIN
set cam_clki  70
# CLKI
set cam_db    { 85 84 83 82 81 80 79 77 }

# SPI connector
set spi_sck   54
set spi_mosi  53
set spi_miso  50
set spi_sel1_n 48

# K-NET connector
set knet_sck   47
set knet_miso  46
set knet_mosi  44
set knet_sel_n 43
set knet_pai   41
set knet_f7_n  40

# UART connectors
set rxd       59
# RxD0
set txd       58
# TxD0
set rxd_debug 57
# RxD1
set txd_debug 55
# TxD1

#Vers Module Ethernet
        set _NDEK_Reset 3
        set _NDEK_L_IRQ 8
        set _NDEK_reg_address {239 238 234}
                              #addr EDK[4..2] -> Nios[3..1]
        set _NDEK_SBHE_n 4
        set _NDEK_data {217 216 215 25 18 17 16 13 219 24 23 22 21 20 10 9}
        set _NDEK_L_IOR_n 227
        set _NDEK_L_IOW_n 226
        set _NDEK_TIED_HI {236 237 224 223}
                          # MEMW MEMR A8  A9                          ( selon docs: ethernet p.26 )
        set _NDEK_TIED_LO {220 225 11 233 232 231 230 222 221}
                          #A0 CSh CS1 A4  A5  A6  A7  A10 A11         ( selon docs: ethernet p.26 )

##############################################
#### Make all pin assignments

cmp add_assignment $top_name "" clk LOCATION "Pin_$clk"
cmp add_assignment $top_name "" reset_n LOCATION "Pin_$reset_n"

cmp add_assignment $top_name "" usr_clk0 LOCATION "Pin_$usr_clk0"
cmp add_assignment $top_name "" usr_clk1 LOCATION "Pin_$usr_clk1"
cmp add_assignment $top_name "" usr_clk2 LOCATION "Pin_$usr_clk2"

set i 0
foreach {a} $ext_addr {
        cmp add_assignment $top_name "" "ext_addr\[$i\]" LOCATION "Pin_$a"
        set i [expr $i+1]
}
```

```
set i 0
foreach {a} $ext_data {
        cmp add_assignment $top_name "" "ext_data\[$i\]" LOCATION "Pin_$a"
        set i [expr $i+1]
}

set i 0
foreach {a} $ext_be_n {
        cmp add_assignment $top_name "" "ext_be_n\[$i\]" LOCATION "Pin_$a"
        set i [expr $i+1]
}

cmp add_assignment $top_name "" rd_wr_n LOCATION "Pin_$rd_wr_n"
cmp add_assignment $top_name "" led_cs LOCATION "Pin_$led_cs"

cmp add_assignment $top_name "" usr_pb0 LOCATION "Pin_$usr_pb0"
cmp add_assignment $top_name "" usr_pb1 LOCATION "Pin_$usr_pb1"

cmp add_assignment $top_name "" a1_a19 LOCATION "Pin_$a1_a19"
cmp add_assignment $top_name "" SRAM_cs_lo_n LOCATION "Pin_$SRAM_cs_lo_n"
cmp add_assignment $top_name "" SRAM_cs_hi_n LOCATION "Pin_$SRAM_cs_hi_n"
cmp add_assignment $top_name "" SRAM_oe_n    LOCATION "Pin_$SRAM_oe_n"

cmp add_assignment $top_name "" FLASH_byte_n LOCATION "Pin_$FLASH_byte_n"
cmp add_assignment $top_name "" FLASH_we_n   LOCATION "Pin_$FLASH_we_n"
cmp add_assignment $top_name "" FLASH_ce_n   LOCATION "Pin_$FLASH_ce_n"
cmp add_assignment $top_name "" FLASH_oe_n   LOCATION "Pin_$FLASH_oe_n"

cmp add_assignment $top_name "" rokbase_scl  LOCATION "Pin_$rokbase_scl"
cmp add_assignment $top_name "" rokbase_sda  LOCATION "Pin_$rokbase_sda"
cmp add_assignment $top_name "" odol_a       LOCATION "Pin_$odol_a"
cmp add_assignment $top_name "" odol_b       LOCATION "Pin_$odol_b"
cmp add_assignment $top_name "" odor_a       LOCATION "Pin_$odor_a"
cmp add_assignment $top_name "" odor_b       LOCATION "Pin_$odor_b"
cmp add_assignment $top_name "" spi_sel0_n   LOCATION "Pin_$spi_sel0_n"
cmp add_assignment $top_name "" adc_shdn_n   LOCATION "Pin_$adc_shdn_n"
cmp add_assignment $top_name "" pwml_pulse   LOCATION "Pin_$pwml_pulse"
cmp add_assignment $top_name "" pwml_dir     LOCATION "Pin_$pwml_dir"
cmp add_assignment $top_name "" pwmr_pulse   LOCATION "Pin_$pwmr_pulse"
cmp add_assignment $top_name "" pwmr_dir     LOCATION "Pin_$pwmr_dir"
cmp add_assignment $top_name "" buz          LOCATION "Pin_$buz"
cmp add_assignment $top_name "" optsen_clk   LOCATION "Pin_$optsen_clk"
cmp add_assignment $top_name "" optsen_rst   LOCATION "Pin_$optsen_rst"
cmp add_assignment $top_name "" optsen_si    LOCATION "Pin_$optsen_si"

cmp add_assignment $top_name "" vdd_fault_n  LOCATION "Pin_$vdd_fault_n"
cmp add_assignment $top_name "" batt_fault_n LOCATION "Pin_$batt_fault_n"
cmp add_assignment $top_name "" wakeup       LOCATION "Pin_$wakeup"
cmp add_assignment $top_name "" pwr_en_n     LOCATION "Pin_$pwr_en_n"

#set i 0
#foreach {a} $proto_io {
#       cmp add_assignment $top_name "" "proto_io\[$i\]" LOCATION "Pin_$a"
#       set i [expr $i+1]
#}

cmp add_assignment $top_name "" proto_cs_n  LOCATION "Pin_$proto_cs_n"

cmp add_assignment $top_name "" mub_p_n   LOCATION "Pin_$mub_p_n"
cmp add_assignment $top_name "" mub_irq_n LOCATION "Pin_$mub_irq_n"

cmp add_assignment $top_name "" cam_io0   LOCATION "Pin_$cam_io0"
cmp add_assignment $top_name "" cam_scl   LOCATION "Pin_$cam_scl"
cmp add_assignment $top_name "" cam_sda   LOCATION "Pin_$cam_sda"
cmp add_assignment $top_name "" cam_qck   LOCATION "Pin_$cam_qck"
cmp add_assignment $top_name "" cam_io1   LOCATION "Pin_$cam_io1"
cmp add_assignment $top_name "" cam_io2   LOCATION "Pin_$cam_io2"
cmp add_assignment $top_name "" cam_clki  LOCATION "Pin_$cam_clki"

set i 0
foreach {a} $cam_db {
        cmp add_assignment $top_name "" "cam_db\[$i\]" LOCATION "Pin_$a"
        set i [expr $i+1]
}

cmp add_assignment $top_name "" spi_sck    LOCATION "Pin_$spi_sck"
cmp add_assignment $top_name "" spi_mosi   LOCATION "Pin_$spi_mosi"
cmp add_assignment $top_name "" spi_miso   LOCATION "Pin_$spi_miso"
cmp add_assignment $top_name "" spi_sel1_n LOCATION "Pin_$spi_sel1_n"

cmp add_assignment $top_name "" knet_sck    LOCATION "Pin_$knet_sck"
cmp add_assignment $top_name "" knet_mosi   LOCATION "Pin_$knet_mosi"
cmp add_assignment $top_name "" knet_miso   LOCATION "Pin_$knet_miso"
cmp add_assignment $top_name "" knet_sel_n  LOCATION "Pin_$knet_sel_n"
cmp add_assignment $top_name "" knet_pai    LOCATION "Pin_$knet_pai"
cmp add_assignment $top_name "" knet_f7_n   LOCATION "Pin_$knet_f7_n"

cmp add_assignment $top_name "" "rxd" LOCATION "Pin_$rxd"
cmp add_assignment $top_name "" "txd" LOCATION "Pin_$txd"
cmp add_assignment $top_name "" "rxd_debug" LOCATION "Pin_$rxd_debug"
cmp add_assignment $top_name "" "txd_debug" LOCATION "Pin_$txd_debug"
```

```
#Vers Module Ethernet
        cmp add_assignment $top_name "" _NDEK_Reset LOCATION "Pin_$_NDEK_Reset"
        cmp add_assignment $top_name "" _NDEK_L_IRQ LOCATION "Pin_$_NDEK_L_IRQ"
        set i 0
        foreach {a} $_NDEK_reg_address {
                cmp add_assignment $top_name "" "_NDEK_reg_address\[$i\]" LOCATION "Pin_$a"
                set i [expr $i+1]
        }
        cmp add_assignment $top_name "" _NDEK_SBHE_n LOCATION "Pin_$_NDEK_SBHE_n"
        set i 0
        foreach {a} $_NDEK_data {
                cmp add_assignment $top_name "" "_NDEK_data\[$i\]" LOCATION "Pin_$a"
                set i [expr $i+1]
        }
        cmp add_assignment $top_name "" _NDEK_L_IOR_n LOCATION "Pin_$_NDEK_L_IOR_n"
        cmp add_assignment $top_name "" _NDEK_L_IOW_n LOCATION "Pin_$_NDEK_L_IOW_n"
        set i 0
        foreach {a} $_NDEK_TIED_HI {
                cmp add_assignment $top_name "" "_NDEK_TIED_HI\[$i\]" LOCATION "Pin_$a"
                set i [expr $i+1]
        }
        set i 0
        foreach {a} $_NDEK_TIED_LO {
                cmp add_assignment $top_name "" "_NDEK_TIED_LO\[$i\]" LOCATION "Pin_$a"
                set i [expr $i+1]
        }
```

4. Interface de données de la caméra

Camera_ctrl.vhdl

```
--------------------------------------------------------------------
-- camera_ctrl.vhdl -- Controller for CMOS Camera
--------------------------------------------------------------------
-- $Id: camera_ctrl.vhdl,v 1.2 2002/11/25 13:54:47 Exp $
--------------------------------------------------------------------
-- $Author: Sylvain Pasini $
-- $Revision: 1.2 $
-- $Date: 2002/11/25 13:54:47 $
--------------------------------------------------------------------

library lpm;
use lpm.lpm_components.all;

library ieee;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;

entity camera_ctrl is
  port (

    -- Avalon Bus Interface Signals
    clk             : in  std_logic;
    reset           : in  std_logic;
    address         : in  std_logic_vector(1 downto 0);
    chipselect      : in  std_logic;
    read            : in  std_logic;
    write           : in  std_logic;
    readdata        : out std_logic_vector(31 downto 0);
    writedata       : in  std_logic_vector(31 downto 0);
    irq             : out std_logic;

    -- Camera Signals
    camera_qclk     : in  std_logic;
    camera_fst      : in  std_logic;
    camera_data_in  : in  std_logic_vector(7 downto 0)
    );

end camera_ctrl;

architecture synt of camera_ctrl is

COMPONENT lpm_fifo
  GENERIC (
    lpm_width            : NATURAL;
    lpm_numwords : NATURAL;
    lpm_widthu           : NATURAL;
    lpm_showahead        : STRING;
    lpm_hint             : STRING;
    lpm_type             : STRING
  );
  PORT (
    empty        : OUT STD_LOGIC ;
    clock        : IN STD_LOGIC ;
    rdreq        : IN STD_LOGIC ;
    aclr         : IN STD_LOGIC ;
    full         : OUT STD_LOGIC ;
    sclr         : IN STD_LOGIC ;
    q         : OUT STD_LOGIC_VECTOR (7 DOWNTO 0);
    wrreq        : IN STD_LOGIC ;
    data         : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
    usedw   : OUT STD_LOGIC_VECTOR (9 DOWNTO 0)
    );
END COMPONENT;


type state_type is ( Idle, WaitFrameStart, WaitClockEdge, GetData );

signal state : state_type;

-- Falling Edge Camera Clock Logic Signals
signal qclk_falling_edge : std_logic;
signal qclk_last                        : std_logic_vector(3 downto 0);

-- Signal FST synchronised
signal fst_sync                 : std_logic;

-- Data Counter Signals
signal data_counter                     : std_logic_vector(15 downto 0);
signal data_counter_limit               : std_logic_vector(15 downto 0);
signal data_counter_flag                : std_logic;
signal data_counter_limit_reached : std_logic;

-- FIFO Control & Data Signals
signal fifo_data_in   : std_logic_vector(7 downto 0);
```

```
signal fifo_data_out  : std_logic_vector(7 downto 0);
signal fifo_wrreq     : std_logic;
signal fifo_rdreq     : std_logic;
signal fifo_rddone    : std_logic;
signal fifo_clear     : std_logic;
signal fifo_overrun   : std_logic;  -- Déborde
signal fifo_empty     : std_logic;  -- Vide
signal fifo_usedw     : std_logic_vector(9 downto 0);
signal fifo_flag      : std_logic;  -- Mi-plein

signal start                    : std_logic;
signal wait_fst          : std_logic;

signal intr_pending      : std_logic;
signal intr_enable       : std_logic;

signal running           : std_logic;



begin

irq <= intr_enable AND intr_pending;
data_counter_limit_reached <= '1' when (data_counter >= data_counter_limit) else '0';

-- FIFO data buffer
dp_fifo_buf: lpm_fifo
  generic map (
    lpm_width      => 8,
    lpm_numwords   => 1024,
    lpm_widthu     => 10,
    lpm_showahead  => "ON",
    lpm_type       => "LPM_FIFO",
    lpm_hint       => "USE_EAB=ON")
  port map (
    rdreq => fifo_rdreq,
    sclr  => fifo_clear,
    aclr  => reset,
    clock => clk,
    wrreq => fifo_wrreq,
    data  => fifo_data_in,
    q     => fifo_data_out,
    full  => fifo_overrun,
    empty => fifo_empty,
    usedw => fifo_usedw);

process(clk, reset)
begin
        if reset='1' then
                qclk_last                       <= (others => '0');
                fst_sync                        <= '0';
                qclk_falling_edge <= '0';

        elsif rising_edge(clk) then
                if qclk_last(3)='1' and qclk_last(2)='0' then
                        qclk_falling_edge <= '1';
                else
                        qclk_falling_edge <= '0';
                end if;

                qclk_last(3 downto 1) <= qclk_last(2 downto 0);
                qclk_last(0) <= camera_qclk;
                fst_sync <= camera_fst;
        end if;
end process;

-- Data access
process(clk, reset)
begin
  if (reset = '1') then
    start        <= '0';
    wait_fst     <= '0';
    intr_enable  <= '0';
    intr_pending <= '0';
    fifo_clear   <= '0';
    fifo_rdreq   <= '0';
    readdata     <= (others => '0');
    data_counter_flag  <= '0';
    data_counter_limit <= (others => '0');
    fifo_flag    <= '0';
    fifo_rddone  <= '0';

  elsif (rising_edge(clk)) then
         -- Lecture terminée
    fifo_rdreq <= '0';
    -- Pas de clear
    fifo_clear <= '0';
    -- Pas de flag
    fifo_flag  <= '0';

    -- If running, no start
    if running='1' then
        start<='0';
```

```
    end if;

    -- All picture is readed ?
    if running='1' and data_counter_limit_reached='1' then
      data_counter_flag <= '1';
      intr_pending      <= '1';
      start <= '0';
    end if;

    -- FIFO half full ?
    if fifo_usedw(9)='1' then
      fifo_flag <= '1';
    end if;

    -- Generate Interrupt Pending
    if ( fifo_flag='1' or fifo_overrun='1') then
      intr_pending <= '1';
    end if;

    -- Write access
    if chipselect='1' and write='1' then
      -- Control register
      if address="00" then
        if running='0' then
          start       <= writedata(0);
          wait_fst    <= writedata(1);
          intr_enable <= writedata(2);
          fifo_clear  <= writedata(3);
        end if;
      -- Status register
      elsif address="01" then
          -- Clear Interrupt Pending
          if intr_pending = '1' then
            intr_pending <= '0';
            data_counter_flag <= '0';
          end if;
      -- Counter limit
      elsif address="11" then
        data_counter_limit <= writedata(15 downto 0);
      end if;
    end if;

    -- Read access
    if chipselect='1' and read='1' then
      case address is
        -- Control Register
        when "00" =>
          readdata(0) <= start;
          readdata(1) <= wait_fst;
          readdata(2) <= intr_enable;
          readdata(3) <= fifo_clear;
          readdata(31 downto 4) <= (others => '0');

        -- Status Register
        when "01" =>
          readdata(0) <= running;
          readdata(1) <= intr_pending;
          readdata(2) <= fifo_flag;
          readdata(3) <= fifo_overrun;
          readdata(4) <= data_counter_flag;
          readdata(5) <= fifo_empty;
          readdata(7 downto 6)   <= (others => '0');
          readdata(17 downto 8)  <= fifo_usedw;
          readdata(31 downto 18) <= (others => '0');

        -- FIFO Data Register
        when "10" =>
          -- Vide le fifo de 1 bytes
          readdata(7 downto 0)  <= fifo_data_out;
          readdata(31 downto 8) <= (others => '0');
          if fifo_rddone = '0' then
                 fifo_rdreq <= '1';
                 fifo_rddone <= '1';
                           else
                                  fifo_rddone <= '0';
                           end if;

        -- Data Counter Value & Limit
        when "11" =>
          readdata(15 downto 0)  <= data_counter_limit;
          readdata(31 downto 16) <= data_counter;

        when others => null;
      end case;
    end if; -- read
  end if; -- clk
end process;

-- Camera Data Acquisition
process(reset, clk)
begin
  if reset='1' then
    state <= IDLE;
```

```
    data_counter      <= (others => '0');
    fifo_wrreq        <= '0';
    fifo_data_in      <= (others => '0');
    data_counter      <= (others => '0');
    running           <= '0';

  elsif rising_edge(clk) then
    fifo_wrreq        <= '0';

    case state is
      when Idle =>
        running <= '0';

        -- Start ?
        if start='1' then
          if wait_fst='1' then
            state <= WaitFrameStart;
          else
            state <= WaitClockEdge;
          end if;
        end if;

      when WaitFrameStart =>
        running <= '1';

        -- Wait FST pin set
        if fst_sync='1' then
          state <= WaitClockEdge;
        end if;

      when WaitClockEdge =>
        running <= '1';

        -- Wait falling edge of QCLK
        if qclk_falling_edge='1' then
          state <= GetData;
        end if;

      when GetData =>
        running <= '1';
        -- read camera data and request a write
        -- into the FIFO buffer
        fifo_data_in <= camera_data_in;
        fifo_wrreq   <= '1';

        -- data counter reached limit ?
        if data_counter_limit_reached='1' then
          -- clear counter & set flag
          data_counter <= (others => '0');
          state <= Idle;
        else
          -- increment counter
          data_counter <= data_counter + 1;
          state <= WaitClockEdge;
        end if;

      when others => null;
    end case;
  end if; -- clk
end process;

end synt;
```

Camera2d.h

```
/*-----------------------------------------------------------------
 * camera2d.h - Camera 2D interface library
 *-----------------------------------------------------------------
 * Id: camera2d.h,   v 1.2     2002/11/24 18:20:57
 *-----------------------------------------------------------------
 * Author   : Sylvain Pasini
 * Revision : 1.2
 * Date: 2002/11/24 18:20:57
 *-----------------------------------------------------------------*/

#ifndef _camera2d_h_
#define _camera2d_h_

/*-----------------------------------------------------------------
 * Camera2D Video Data Acquisition Controller Interface
 *-----------------------------------------------------------------*/

typedef volatile struct
{
  int np_cc_ctrl;
  int np_cc_status;
  int np_cc_data;
  int np_cc_limit;

} np_camera_ctrl;

// Control register
// ----------------
enum {
  np_cc_ctrl_start_bit       = 0,
  np_cc_ctrl_waitfst_bit     = 1,
  np_cc_ctrl_intr_enable_bit = 2,
  np_cc_ctrl_fifo_clear_bit  = 3,

  np_cc_ctrl_start_mask       = (1<<0),
  np_cc_ctrl_waitfst_mask     = (1<<1),
  np_cc_ctrl_intr_enable_mask = (1<<2),
  np_cc_ctrl_fifo_clear_mask  = (1<<3)
};

// Status register
// ---------------
enum {
  np_cc_status_running_bit           = 0,
  np_cc_status_intr_pending_bit      = 1,
  np_cc_status_fifo_flag_bit         = 2,
  np_cc_status_fifo_overrun_bit      = 3,
  np_cc_status_data_counter_flag_bit = 4,
  np_cc_status_fifo_empty_bit        = 5,

  np_cc_status_running_mask           = (1<<0),
  np_cc_status_intr_pending_mask      = (1<<1),
  np_cc_status_fifo_flag_mask         = (1<<2),
  np_cc_status_fifo_overrun_mask      = (1<<3),
  np_cc_status_data_counter_flag_mask = (1<<4),
  np_cc_status_fifo_empty_mask        = (1<<5)
};

/*-----------------------------------------------------------------
 * Camera2D I2C Interface
 *-----------------------------------------------------------------*/

/* I2C address of VV6301 2D camera */
#define CAMERA2D_I2C_ADDR     0x20 /* I2C address of camera */

/* Error Codes */
#define CAMERA2D_ENODEV       I2C_ENODEV  /* no device             */
#define CAMERA2D_EBADACK      I2C_EBADACK /* bad acknowledge      */
#define CAMERA2D_EBADIDX      3           /* bad index            */

/* Status registers */
#define CAMERA2D_DEVH         1 /* device identifier */
#define CAMERA2D_DEVL         2
#define CAMERA2D_STATUS0      3
#define CAMERA2D_FRAME_AV     7

/* Setup registers */
#define CAMERA2D_SETUP0       16
#define CAMERA2D_SETUP1       17
#define CAMERA2D_SETUP2       18
#define CAMERA2D_SETUP3       19
#define CAMERA2D_SETUP4       20

/* Exposure registers */
#define CAMERA2D_FINE         33
#define CAMERA2D_COARSE       35
#define CAMERA2D_GAIN         36
#define CAMERA2D_CLKDIV       37
#define CAMERA2D_GN_LIM       38 /* gain limit               */
```

```
#define CAMERA2D_TL           39 /* exposure lower threshold  */
#define CAMERA2D_TC           40 /* exposure center threshold */
#define CAMERA2D_TH           41 /* exposure higher threshold */

/* System registers */
#define CAMERA2D_BDAC         112 /* black calibration         */
#define CAMERA2D_B0           113
#define CAMERA2D_B1           114
#define CAMERA2D_TMS          116 /* test                      */
#define CAMERA2D_CR0          118 /* analogue control          */
#define CAMERA2D_CR1          119

/* Status bits */
#define CAMERA2D_STA_EXPO_PE (1<<0) /* exposure value update pending */
#define CAMERA2D_STA_GAIN_PE (1<<2) /* gain value update pending     */

/* Setup bits */
#define CAMERA2D_SET_AUTO_EX (1<<0) /* automatic exposure control */
#define CAMERA2D_SET_AUTO_GN (1<<2) /* automatic gain control     */
#define CAMERA2D_SET_FMT_SEL (1<<5) /* data format select         */
#define CAMERA2D_SET_ADD_BLK (1<<0) /* additional black lines     */
#define CAMERA2D_SET_SHUFFLE (1<<2) /* horizontal shuffle mode    */
#define CAMERA2D_SET_60HZ    (1<<6) /* 50Hz/60Hz timing selection */

/* standard configuration for Camera2D module */
extern void camera2d_configure(np_i2c *io);

/* display all registers */
extern void camera2d_dump_registers(np_i2c *io);

/* set index register (no data write) */
extern int camera2d_write_index(np_i2c *io, unsigned char index);

/* write into a single register */
extern int camera2d_single_write(np_i2c *io,
                                 unsigned char index,
                                 unsigned char value);

/* read from a single register */
extern int camera2d_single_read(np_i2c *io,
                                unsigned char index,
                                unsigned char *value);

/* write into multiple consecutive registers */
extern int camera2d_multiple_write(np_i2c *io,
                                   unsigned char index,
                                   unsigned char *values,
                                   unsigned char count);

/* read from multiple consecutive registers */
extern int camera2d_multiple_read(np_i2c *io,
                                  unsigned char index,
                                  unsigned char *values,
                                  unsigned char count);

#endif /* _camera2d_h_ */
```

Camera2d.c

```
/*-------------------------------------------------------------------
 * camera2d.c - Camera 2D interface library
 *-------------------------------------------------------------------
 * $Id: camera2d.c,v 1.1.1.1 2002/10/22 19:30:07 cgaudin Exp $
 *-------------------------------------------------------------------
 * $Author: cgaudin $
 * $Revision: 1.1.1.1 $
 * $Date: 2002/10/22 19:30:07 $
 *-------------------------------------------------------------------*/

#include "nios.h"
#include "i2c.h"
#include  "camera2d.h"

/* wait some microseconds */
#define WAIT { int i; for (i=0; i<100; ) i++; }

/* wait for end of transfer */
#define WAIT_FOR_EOT(io,t) { while ((t=io->np_i2c_status) & I2C_TIP) WAIT; }


struct {
  unsigned char index;
  char         *name;
}

camera2d_register_names[] = {
  { 0,   "DevH" },
  { 1,   "DevL" },
  { 2,   "Status0" },
  { 16,  "Setup0" },
  { 17,  "Setup1" },
  { 18,  "Setup2" },
  { 19,  "Setup3" },
  { 20,  "Setup4" },
  { 33,  "Fine" },
  { 35,  "Coarse" },
  { 36,  "Gain" },
  { 37,  "ClkDiv" },
  { 38,  "GainLimit" },
  { 39,  "LowThreshold" },
  { 40,  "CenterThreshold" },
  { 41,  "HighThreshold" },
  { 112, "BlackCalibration" },
  { 113, "Black0" },
  { 114, "Black1" },
  { 116, "SystemTest" },
  { 118, "AnCtrlReg0" },
  { 119, "AnCtrlReg1" },
  { 0, 0 }
};

/*-------------------------------------------------------------------
 * Configuration Primitives
 *-------------------------------------------------------------------*/

/*-------------------------------------------------------------------
 * Standard configuration for Camera2D module
 *-------------------------------------------------------------------*/
void camera2d_configure(np_i2c *io)
{
  /*
   * automatic exposure control on
   * automatic gain control on
   * data format 8 wire parallel output
   * + --------------------------------+
   * | 0 : Automatic exposure                      |
   * | 2 : Automatic Gain                                  |
   * | 5 : Data format 0=8bits 1=4bits |
   * | 1,3,4,6,7 : Unused                                  |
   * + --------------------------------+
   */
  camera2d_single_write(io, CAMERA2D_SETUP0, 0x02 |
                        CAMERA2D_SET_AUTO_EX |
                        CAMERA2D_SET_AUTO_GN);

  /*
   * disable additional black lines (3-8)
   * disable horizontal shuffle mode
   * 50Hz timing
        * + --------------------------------+
   * | 0 : Enable balck lines                      |
   * | 2 : Enable horiz shuffle                    |
   * | 6 : 50Hz / 60Hz                                             |
   * | 1,3-5,7 : Unused                                            |
   * + --------------------------------+
   */
  camera2d_single_write(io, CAMERA2D_SETUP1, 0x32);
```

```
        /*
         * Exposure Step Size
         * + ------------------------------------------------+
    * | 6-5 : Exposure step size 1/8 fast or 1/64 slow |
    * | 0-4,7 : Unused                                                                  |
    * + ------------------------------------------------+
         */
        camera2d_single_write(io, CAMERA2D_SETUP3, 0x00);

  /*
   * FST/QCK pin modes : FST_pin = FST, QCK_pin = QCK
   * QCK modes = valid only during data period of line
   * FST modes = normal behaviour FST will qualify
   *             the visible pixels in status line
        * + --------------------------------+
   * | 1-0 : FST/QCK pin modes                    |
   * | 3-2 : QCK modes                                          |
   * | 7 : FST modes                                            |
   * | 4-5 : Unused                                                      |
   * + --------------------------------+
   */
  camera2d_single_write(io, CAMERA2D_SETUP4, 0x4C);

        camera2d_single_write(io, CAMERA2D_CLKDIV, 0x00);

        //camera2d_single_write(io, CAMERA2D_BDAC, 0x01);
        //camera2d_single_write(io, CAMERA2D_CR0, 0x0C);
        //camera2d_single_write(io, CAMERA2D_CR1, 0x0C);

}

/*-----------------------------------------------------------------
 * Low-Level I2C Communication Primitives
 *-----------------------------------------------------------------*/

static unsigned char regs[128];

/*
 * display all registers
 */
void camera2d_dump_registers(np_i2c *io)
{
  int i;

  printf("Reading Camera2D(VV6301) Registers...\n");

  if (camera2d_multiple_read(io, 0, (unsigned char *)&regs, 128)<0) {
    printf("No device found at address %02X !\n", CAMERA2D_I2C_ADDR);
    return;
  }

  printf("done !\n");

  printf("---------------------------------\n");
  for (i=0; camera2d_register_names[i].name != 0; i++) {
    printf( "(%3u) %20s = %03u, %02X \n",
            camera2d_register_names[i].index,
            camera2d_register_names[i].name,
            regs[camera2d_register_names[i].index],
            regs[camera2d_register_names[i].index]);
  }
}

/*-----------------------------------------------------------------
 * Set register index (no data write)
 *-----------------------------------------------------------------*/
int camera2d_write_index(np_i2c *io,
                         unsigned char index)
{
  unsigned char t;

  /* wait for end of transfer */
  WAIT_FOR_EOT(io, t);

  /* write device address + r/w=0 */
  io->np_i2c_data = ( CAMERA2D_I2C_ADDR & 0xFE );
  io->np_i2c_ctrl = ( I2C_START | I2C_WRITE );

  /* wait for end of transfer */
  WAIT_FOR_EOT(io, t);

  /* no device found ? */
  if (t & I2C_LRA)
    return -CAMERA2D_ENODEV;

  /* write index */
  io->np_i2c_data = ( index );
  io->np_i2c_ctrl = ( I2C_WRITE );

  /* wait for end of transfer */
  WAIT_FOR_EOT(io, t);
```

```
  /* bad acknowledge */
  if (t & I2C_LRA)
    return -CAMERA2D_EBADACK;

  /* clear interrupt */
  io->np_i2c_data = 0;

  return 0;
}

/*-----------------------------------------------------------------
 * Write into a single register
 *-----------------------------------------------------------------*/
int camera2d_single_write(np_i2c *io,
                          unsigned char index,
                          unsigned char value)
{
  int rc;
  unsigned char t;

  /* set index */
  if ((rc=camera2d_write_index(io, index & 0x7F)) < 0)
    return rc;

  /* write data byte */
  io->np_i2c_data = ( value );
  io->np_i2c_ctrl = ( I2C_WRITE | I2C_STOP );

  /* wait for end of transfer */
  WAIT_FOR_EOT(io, t);

  /* bad acknowledge ? */
  if (t & I2C_LRA)
    return -CAMERA2D_EBADACK;

  /* dummy access cycle for clearing interrupt flag */
  io->np_i2c_data = 0;

  return 0;
}

/*-----------------------------------------------------------------
 * Read a single register
 *-----------------------------------------------------------------*/
int camera2d_single_read(np_i2c *io,
                         unsigned char index,
                         unsigned char *value)
{
  int rc;
  unsigned char t;

  /* set index */
  if ((rc=camera2d_write_index(io, index & 0x7F)) < 0)
    return rc;

  /* do repstart for reading - write address + r/w=1 */
  io->np_i2c_data = ( CAMERA2D_I2C_ADDR | 0x01 );
  io->np_i2c_ctrl = ( I2C_WRITE | I2C_START );

  /* wait for end of transfer */
  WAIT_FOR_EOT(io, t);

  /* no device found ? */
  if (t & I2C_LRA)
    return -CAMERA2D_ENODEV;

  /* read back the index */
  io->np_i2c_ctrl = ( I2C_READ );

  /* wait for end of transfer */
  WAIT_FOR_EOT(io, t);

  /* read register value */
  t = io->np_i2c_data;
  io->np_i2c_ctrl = ( I2C_READ | I2C_STOP | I2C_ACK);

  /* wait for end of transfer */
  WAIT_FOR_EOT(io, t);

  (*value) = io->np_i2c_data;

  return 0;
}

/*-----------------------------------------------------------------
 * write into multiple consecutive registers
 *-----------------------------------------------------------------*/
int camera2d_multiple_write(np_i2c *io,
                            unsigned char index,
                            unsigned char *values,
                            unsigned char count)
{
  int rc;
```

```
  unsigned char t;

  /* set index with autoincrement mode on */
  if ((rc=camera2d_write_index(io, index | 0x80)) < 0)
    return rc;

  while (count > 0) {

    /* write a byte */
    io->np_i2c_data = ( *values );

    if (count==1)
      io->np_i2c_ctrl = ( I2C_WRITE | I2C_STOP );
    else
      io->np_i2c_ctrl = ( I2C_WRITE );

    /* wait for end of transfer */
    WAIT_FOR_EOT(io, t);

    /* next byte */
    count --;
    values ++;
  }

  /* dummy access cycle for clearing interrupt flag */
  io->np_i2c_data = 0;

  return 0;
}

/*-------------------------------------------------------------------
 * read consecutive registers
 *-------------------------------------------------------------------*/
int camera2d_multiple_read(np_i2c *io,
                           unsigned char index,
                           unsigned char *values,
                           unsigned char count)
{
   unsigned char t;
   int rc;

   /* set index with autoincrement mode on */
   if ((rc = camera2d_write_index(io, index|0x80)) < 0)
     return rc;

   /* do a repstart and write device address + r/w=1 */
   io->np_i2c_data = ( CAMERA2D_I2C_ADDR | 0x01 );
   io->np_i2c_ctrl = ( I2C_START | I2C_WRITE );

   /* wait for end of transfer */
   WAIT_FOR_EOT(io, t);

   /* no device found ? */
   if (t & I2C_LRA)
     return -CAMERA2D_ENODEV;

   /* read back the index, first time */
   io->np_i2c_ctrl = ( I2C_READ );

   /* wait for end of transfer */
   WAIT_FOR_EOT(io, t);

   /* interrupt clearing */
   t=io->np_i2c_data;

   /* read back the index, second time */
   io->np_i2c_ctrl = ( I2C_READ );

   /* wait for end of transfer */
   WAIT_FOR_EOT(io, t);

   /* interrupt clearing */
   t=io->np_i2c_data;

   while (count > 0) {

     /* last byte */
     if (count == 1)
       io->np_i2c_ctrl = ( I2C_READ | I2C_ACK | I2C_STOP );
     else
       io->np_i2c_ctrl = ( I2C_READ );

     /* wait for end of transfer */
     WAIT_FOR_EOT(io, t);

     (*values) = io->np_i2c_data;

     values ++;
     count --;
   }

  return 0;
}
```

5. Interface de configuration de la camera (i2c)

i2c_interface.vhd

```
------------------------------------------------------------------
-- i2c_interface.vhd -- I2C Master Interface for
--                      Avalon Bus Slave Interface
------------------------------------------------------------------
-- Author  : Cédric Gaudin
-- Version : 0.8 alpha
-- History :
--           20-mar-2002 CG 0.1 initial alpha release
--           20-mar-2002 CG 0.2 minor corrections
--           27-mar-2002 CG 0.6 interface is working yet
--           02-apr-2002 CG 0.7 minor corrections
------------------------------------------------------------------
-- Registers description:
--
-- Adr RW Name
-- 00  RW DR - transmit and receive data register
-- 01  WO CR - control register
-- 10  RO SR - status register
-- 11  RW CD - clock divisor
--
-- SR - status register bits
--
--  +---------+-------+-------+-------+-------+
--  |bit 5-7  | bit 3 | bit 2 | bit 1 | bit 0 |
--  +---------+-------+-------+-------+-------+
--  | UNUSED  | TIP   | IPE   | BSY   | LAR   |
--  +---------+-------+-------+-------+-------+
--
-- TIP   - transfer in progress
-- IPE   - interrupt pending
-- BSY   - I2C bus busy
-- LAR   - last acknowledge received
--
-- CR - control register bits
--
--  +---------+-------+-------+-------+-------+-------+-------+
--  | bit 6-7 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
--  +---------+-------+-------+-------+-------+-------+-------+
--  | UNUSED  | IEN   | WR    | RD    | STA   | STP   | ACK   |
--  +---------+-------+-------+-------+-------+-------+-------+
--
-- ACK    - Acknowledge bit for reading
-- STP    - Generate a I2C Stop Sequence
-- STA    - Generate a I2C Start Sequence
-- RD     - Read command bit
-- WR     - Write command bit
-- IEN    - Interrupt Enable
--
-- To start a transfer WR or RD *MUST* BE set.
-- When command transfer has started TIP goes high
-- and write to CR are ignored until TIP goes low.
-- At end of transfer IRQ goes high if interrupt is enabled (IEN=1).
--
------------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;

entity i2c_interface is
        port (
                clk         : in std_logic;
                reset       : in std_logic;

                -- Avalon bus signals
                address     : in std_logic_vector(1 downto 0);
                chipselect  : in std_logic;
                write         : in std_logic;
                writedata   : in std_logic_vector(7 downto 0);
                read        : in std_logic;
                readdata    : out std_logic_vector(7 downto 0);
                irq         : out std_logic;

                -- I2C signals
                scl                                     : inout std_logic;
                sda                                     : inout std_logic

        );

end i2c_interface;

architecture structural of i2c_interface is

component i2c_core
        port (
```

```
                -- I2C signals
                sda_in       : in  std_logic;
                scl_in       : in  std_logic;
                sda_out      : out std_logic;
                scl_out      : out std_logic;

                -- interface signals
                clk          : in std_logic;
                rst          : in std_logic;
                sclk           : in std_logic;
                ack_in       : in  std_logic;
                ack_out      : out std_logic;
                data_in      : in  std_logic_vector(7 downto 0);
                data_out     : out std_logic_vector(7 downto 0);
                cmd_start    : in  std_logic;
                cmd_stop     : in  std_logic;
                cmd_read     : in  std_logic;
                cmd_write    : in  std_logic;
                cmd_done_ack : in  std_logic;
                cmd_done     : out std_logic;
                busy         : out std_logic

                -- debug signals
                --state                  : out std_logic_vector(5 downto 0)
        );
end component;

component i2c_clkgen
        port (
                signal clk     : in std_logic;
                signal rst     : in std_logic;

                signal clk_cnt : in std_logic_vector(7 downto 0);

                -- I2C clock generated
                signal sclk    : out std_logic;

                -- I2C clock line SCL (used for clock stretching)
                signal scl_in  : in std_logic;
                signal scl_out : in std_logic
        );
end component;



-- I2C base clock
signal i_sclk         : std_logic;

-- I2C serial clock output
signal i_scl_out      : std_logic;

-- clock divisor register
signal i_clkdiv_reg   : std_logic_vector(7 downto 0);

-- status register bits
signal i_tip_reg      : std_logic; -- transfer in progress      ( bit 3 )
signal i_int_pe_reg   : std_logic; -- interrupt pending         ( bit 2 )
signal i_busy_reg     : std_logic; -- busy                      ( bit 1 )
signal i_lar_reg      : std_logic; -- last acknowledge received ( bit 0 )

-- control register bits
signal i_int_en_reg   : std_logic; -- interrupt enable          ( bit 5 )
signal i_write_reg    : std_logic; -- write command             ( bit 4 )
signal i_read_reg     : std_logic; -- read command              ( bit 3 )
signal i_start_reg    : std_logic; -- command with a start      ( bit 2 )
signal i_stop_reg     : std_logic; -- command with a stop       ( bit 1 )
signal i_ack_reg      : std_logic; -- acknowledge to send       ( bit 0 )

-- data register
signal i_data_out     : std_logic_vector(7 downto 0);
signal i_data_in      : std_logic_vector(7 downto 0);

-- command done & acknowledge signals
signal i_cmd_done_ack : std_logic;
signal i_cmd_done     : std_logic;

-- internal signals
signal i_readdata     : std_logic_vector(7 downto 0);
signal i_irq          : std_logic;

-- write strobe
signal i_write_strobe : std_logic;

-- read strobe
signal i_read_strobe  : std_logic;

-- interrupt clear
signal i_int_clr     : std_logic;


-- just implement open collector in module
signal scl_in        : std_logic;
signal sda_in        : std_logic;
```

```
signal scl_out       : std_logic;
signal sda_out       : std_logic;

begin

scl_in <= scl;
sda_in <= sda;

scl <= 'Z' when (scl_out = '1') else '0';
sda <= 'Z' when (sda_out = '1') else '0';

clkgen: i2c_clkgen port map (
        clk     => clk,
        rst      => reset,
        clk_cnt => i_clkdiv_reg,
        sclk    => i_sclk,
        scl_in  => scl_in,
        scl_out => i_scl_out
);

core: i2c_core port map (
        clk          => clk,
        rst              => reset,
        sclk            => i_sclk,
        ack_in       => i_ack_reg,
        ack_out      => i_lar_reg,
        data_in      => i_data_in,
        data_out     => i_data_out,
        cmd_start    => i_start_reg,
        cmd_stop     => i_stop_reg,
        cmd_read     => i_read_reg,
        cmd_write    => i_write_reg,
        cmd_done_ack => i_cmd_done_ack,
        cmd_done     => i_cmd_done,
        busy         => i_busy_reg,
        sda_in       => sda_in,
        scl_in       => scl_in,
        sda_out      => sda_out,
        scl_out      => i_scl_out
--      state        => state
);

-- read strobe
i_read_strobe <= (chipselect and read);

-- output to avalon bus
data_out_sync: process(clk, reset)
begin
        if (reset = '1' ) then
                readdata <= (others => '0');
                irq <= '0';
        elsif (rising_edge(clk)) then
                if (i_read_strobe = '1') then
                        readdata <= i_readdata;
                end if;
                irq <= i_irq;
        end if;
end process;

-- output multiplexer
data_out_comb: process(address, i_data_out, i_lar_reg, i_busy_reg, i_int_pe_reg, i_tip_reg, i_clkdiv_reg)
begin
        case address is
                when "00"   => i_readdata    <= i_data_out;
                when "10"   => i_readdata(0) <= i_lar_reg;
                  i_readdata(1) <= i_busy_reg;
                  i_readdata(2) <= i_int_pe_reg;
                  i_readdata(3) <= i_tip_reg;
                  i_readdata(7 downto 4) <= "0000";
                when "11"   => i_readdata    <= i_clkdiv_reg;
                when others => i_readdata    <= (others => '0');
        end case;
end process;

-- output scl already syncronized in core
scl_out <= i_scl_out;

-- transfer in progress
i_tip_reg <= ( i_read_reg OR i_write_reg );

-- write strobe
i_write_strobe <= (chipselect and write);

-- interrupt output
i_irq <= ( i_int_pe_reg AND i_int_en_reg );

-- interrupt clear signal coming from outside
i_int_clr  <= '1' when (address = "00" and chipselect='1') else '0';

data_in_sync: process(clk, reset)
begin
        if ( reset = '1' ) then
                i_int_pe_reg <= '0';
```

```
                i_data_in    <= (others => '0');
        i_int_en_reg <= '0';
                i_write_reg  <= '0';
                i_read_reg   <= '0';
                i_start_reg  <= '0';
                i_stop_reg   <= '0';
                i_ack_reg    <= '0';
                i_clkdiv_reg <= "10000011";
        i_cmd_done_ack <= '0';

        elsif ( rising_edge(clk) ) then
                i_cmd_done_ack <= '0';

                -- no transfer in progress
                if ( i_tip_reg = '0' ) then

                        if ( i_write_strobe = '1' ) then
                                case address is
                                        when "00" => i_data_in       <= writedata;
                                        when "01" => i_ack_reg       <= writedata(0);
                                                i_stop_reg      <= writedata(1);
                                                i_start_reg     <= writedata(2);
                                                      i_read_reg      <= writedata(3);
                                                      i_write_reg     <= writedata(4);

                                                          i_int_en_reg    <= writedata(5);

                                        when "11" => i_clkdiv_reg    <= writedata;
                                        when others => null;
                                end case;
                        end if;

                        if (i_int_clr = '1') then
                                i_int_pe_reg <= '0';
                        end if;
                else
                        if (i_cmd_done = '1') then
                                -- clear command bits
                                i_write_reg <= '0';
                                i_read_reg  <= '0';
                                i_start_reg <= '0';
                                i_stop_reg  <= '0';

                                -- set interrupt pending
                                i_int_pe_reg <= '1';

                                -- acknowledge cmd done to core controller
                                i_cmd_done_ack <= '1';
                        end if;
                end if;
        end if;
end process data_in_sync;

end structural;
```

i2c_core.vhd

```
-------------------------------------------------------
-- i2c_core.vhd - I2C core V2 logic
-------------------------------------------------------
-- Author  : Cédric Gaudin
-- Version : 0.4 alpha
-- History :
--           20-mar-2002 CG 0.1 initial alpha release
--           22-mar-2002 CG 0.2 complete rewrite
--           27-mar-2002 CG 0.3 minor corrections
--           02-apr-2002 CG 0.4 sync. of outputs
-------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;

entity i2c_core is
        port(
                -- I2C signals
                sda_in    : in  std_logic;
                scl_in    : in  std_logic;
                sda_out   : out std_logic;
                scl_out   : out std_logic;

                -- interface signals
                clk          : in  std_logic;
                rst              : in  std_logic;
                sclk         : in  std_logic;
                ack_in       : in  std_logic;
                ack_out      : out std_logic;
                data_in      : in  std_logic_vector(7 downto 0);
                data_out     : out std_logic_vector(7 downto 0);
                cmd_start    : in  std_logic;
                cmd_stop     : in  std_logic;
                cmd_read     : in  std_logic;
                cmd_write    : in  std_logic;
                cmd_done_ack : in  std_logic;
                cmd_done     : out std_logic;
                busy         : out std_logic
        );
end i2c_core;

architecture behavorial of i2c_core is

type state_type is (
        s_Reset,   s_Idle,    s_Done, s_DoneAck,
        s_Start_A, s_Start_B, s_Start_C, s_Start_D,
        s_Stop_A,  s_Stop_B,  s_Stop_C,
        s_Rd_A,    s_Rd_B,    s_Rd_C, s_Rd_D, s_Rd_E, s_Rd_F,
        s_RdAck_A, s_RdAck_B, s_RdAck_C, s_RdAck_D, s_RdAck_E,
        s_Wr_A,    s_Wr_B,    s_Wr_C, s_Wr_D, s_Wr_E,
        s_WrAck_A, s_WrAck_B, s_WrAck_C, s_WrAck_D
        );

-- data output register
signal i_dout_ld    : std_logic;
signal i_dout        : std_logic_vector(7 downto 0);

-- ack output register
signal i_ack_out_ld : std_logic;
signal i_ack_out    : std_logic;

-- data input bit
signal i_data_in    : std_logic;

-- bit counter
signal i_ctr        : unsigned(2 downto 0);
signal i_ctr_incr   : std_logic;
signal i_ctr_clr    : std_logic;

signal p_state     : state_type;
signal n_state     : state_type;


signal i_scl_out    : std_logic;
signal i_sda_out    : std_logic;

signal i_sclk_en    : std_logic;

signal i_cmd_done   : std_logic;
signal i_cmd_go     : std_logic;
signal i_busy       : std_logic;

begin

-- syncronize output signals
output_sync: process (clk, rst)
begin
        if (rst = '1') then
```

```
                scl_out  <= '1';
                sda_out  <= '1';
                data_out <= (others => '0');
                ack_out  <= '0';
                busy     <= '0';
                cmd_done <= '0';
        elsif (rising_edge(clk)) then
                scl_out  <= i_scl_out;
                sda_out  <= i_sda_out;
                data_out <= i_dout;
                ack_out  <= i_ack_out;
                busy     <= i_busy;
                cmd_done <= i_cmd_done;
        end if;
end process output_sync;

-- select current bit
data_input_selector: process(i_ctr, data_in)
begin
  case i_ctr is
        when "000" => i_data_in <= data_in(7);
        when "001" => i_data_in <= data_in(6);
        when "010" => i_data_in <= data_in(5);
        when "011" => i_data_in <= data_in(4);
        when "100" => i_data_in <= data_in(3);
        when "101" => i_data_in <= data_in(2);
        when "110" => i_data_in <= data_in(1);
        when "111" => i_data_in <= data_in(0);
        when others => null;
  end case;
end process data_input_selector;

-- indicate start of command
i_cmd_go <= ( cmd_read OR cmd_write ) AND NOT i_busy;

-- i2c bit counter
counter: process(clk, rst)
begin
        if (rst = '1' ) then
                i_ctr <= (others => '0');
        elsif (rising_edge(clk)) then
                if (i_ctr_clr = '1') then
                        i_ctr <= (others => '0');
                elsif (i_ctr_incr = '1') then
                        i_ctr <= i_ctr + 1;
                end if;
        end if;
end process counter;


-- data output register
dout_reg: process(clk, rst)
begin
        if ( rst = '1' ) then
                i_dout <= (others => '0');
        elsif (rising_edge(clk)) then
                if (i_dout_ld = '1') then
                        case i_ctr is
                                when "000" => i_dout(7) <= sda_in;
                                when "001" => i_dout(6) <= sda_in;
                                when "010" => i_dout(5) <= sda_in;
                                when "011" => i_dout(4) <= sda_in;
                                when "100" => i_dout(3) <= sda_in;
                                when "101" => i_dout(2) <= sda_in;
                                when "110" => i_dout(1) <= sda_in;
                                when "111" => i_dout(0) <= sda_in;
                          when others => null;
                        end case;
                end if;
        end if;
end process dout_reg;

-- ack bit output register
ack_out_reg: process(clk, rst)
begin
        if (rst = '1' ) then
                i_ack_out <= '0';
        elsif (rising_edge(clk)) then
                if (i_ack_out_ld = '1') then
                        i_ack_out <= sda_in;
                end if;
        end if;
end process ack_out_reg;

-- i2c send / receive byte
i2c_sync: process(rst, clk)
begin
        if ( rst = '1' ) then
                p_state <= s_Reset;
        elsif  ( rising_edge(clk) ) then
                if ( ( sclk = '1' and i_sclk_en = '1' ) or i_sclk_en='0' ) then
                        p_state <= n_state;
                end if;
```

```
        end if;
end process i2c_sync;

i2c_comb: process( p_state, sda_in, scl_in, i_cmd_go, i_ctr, ack_in, i_data_in,
                   cmd_start, cmd_stop, cmd_write, cmd_read, cmd_done_ack)
begin
        n_state <= p_state;
        i_sclk_en    <= '0';
        i_busy       <= '0';
        i_ctr_clr    <= '0';
        i_ctr_incr   <= '0';
        i_cmd_done   <= '0';
        i_sda_out    <= sda_in;
        i_scl_out    <= scl_in;

        case p_state is

                when s_Reset =>
                        --state        <= "000000";
                        i_sclk_en    <= '0';
                        i_busy       <= '0';
                        i_ctr_clr    <= '0';
                        i_ctr_incr   <= '0';
                        i_cmd_done   <= '0';
                        i_dout_ld    <= '0';
                        i_ack_out_ld <= '0';
                        i_sda_out    <= '1';
                        i_scl_out    <= '1';
                        n_state      <= s_Idle;

                when s_Idle =>
                        --state        <= "000001";
                        i_sclk_en    <= '0';
                        i_busy       <= '0';
                        i_ctr_clr    <= '1';
                        i_ctr_incr   <= '0';
                        i_cmd_done   <= '0';
                        i_dout_ld    <= '0';
                        i_ack_out_ld <= '0';
                        i_sda_out    <= sda_in;
                        i_scl_out    <= scl_in;

                        if ( i_cmd_go = '1' ) then

                                if ( cmd_start = '1' ) then

                                        -- do a START
                                        n_state <= s_Start_A;
                                elsif ( cmd_write = '1' ) then

                                        -- do a WRITE
                                        n_state <= s_Wr_A;
                                elsif ( cmd_read = '1' ) then

                                        -- do a READ
                                        n_state <= s_Rd_A;
                                end if;
                        end if;


                when s_Start_A =>
                        i_sclk_en    <= '1';
                        i_busy       <= '1';
                        i_ctr_clr    <= '0';
                        i_ctr_incr   <= '0';
                        i_cmd_done   <= '0';
                        i_dout_ld    <= '0';
                        i_ack_out_ld <= '0';
                        i_sda_out    <= '1';
                        i_scl_out    <= scl_in;

                        n_state <= s_Start_B;

                when s_Start_B =>
                        i_sclk_en    <= '1';
                        i_busy       <= '1';
                        i_ctr_clr    <= '0';
                        i_ctr_incr   <= '0';
                        i_cmd_done   <= '0';
                        i_dout_ld    <= '0';
                        i_ack_out_ld <= '0';
                        i_sda_out    <= '1';
                        i_scl_out    <= '1';

                        n_state  <= s_Start_C;

                when s_Start_C =>
                        i_sclk_en    <= '1';
                        i_busy       <= '1';
                        i_ctr_clr    <= '0';
                        i_ctr_incr   <= '0';
                        i_cmd_done   <= '0';
                        i_dout_ld    <= '0';
```

```
                i_ack_out_ld <= '0';
                i_sda_out    <= '0';
                i_scl_out    <= '1';

                n_state  <= s_Start_D;

        when s_Start_D =>
                i_sclk_en    <= '1';
                i_busy       <= '1';
                i_ctr_clr    <= '0';
                i_ctr_incr   <= '0';
                i_cmd_done   <= '0';
                i_dout_ld    <= '0';
                i_ack_out_ld <= '0';
                i_sda_out    <= '0';
                i_scl_out    <= '0';

                if ( cmd_write = '1' ) then
                        -- do a WRITE
                        n_state <= s_Wr_A;
                elsif (cmd_read = '1') then
                        -- do a READ
                        n_state <= s_Rd_A;
                end if;

        when s_Rd_A =>
                i_sclk_en    <= '1';
                i_busy       <= '1';
                i_ctr_clr    <= '0';
                i_ctr_incr   <= '0';
                i_cmd_done   <= '0';
                i_dout_ld    <= '0';
                i_ack_out_ld <= '0';
                i_sda_out    <= '1';
                i_scl_out    <= '0';

                n_state      <= s_Rd_B;

        when s_Rd_B =>
                i_sclk_en    <= '1';
                i_busy       <= '1';
                i_ctr_clr    <= '0';
                i_ctr_incr   <= '0';
                i_cmd_done   <= '0';
                i_dout_ld    <= '0';
                i_ack_out_ld <= '0';
                i_sda_out    <= '1';
                i_scl_out    <= '1';

                n_state  <= s_Rd_C;

        when s_Rd_C =>
                i_sclk_en    <= '0';
                i_busy       <= '1';
                i_ctr_clr    <= '0';
                i_ctr_incr   <= '0';
                i_cmd_done   <= '0';
                i_dout_ld    <= '1';
                i_ack_out_ld <= '0';
                i_sda_out    <= '1';
                i_scl_out    <= '1';

                n_state  <= s_Rd_D;

        when s_Rd_D =>
                i_sclk_en    <= '1';
                i_busy       <= '1';
                i_ctr_clr    <= '0';
                i_ctr_incr   <= '0';
                i_cmd_done   <= '0';
                i_dout_ld    <= '0';
                i_ack_out_ld <= '0';
                i_sda_out    <= '1';
                i_scl_out    <= '1';

                n_state  <= s_Rd_E;

        when s_Rd_E =>
                i_sclk_en    <= '1';
                i_busy       <= '1';
                i_ctr_clr    <= '0';
                i_ctr_incr   <= '0';
                i_cmd_done   <= '0';
                i_dout_ld    <= '0';
                i_ack_out_ld <= '0';
                i_sda_out    <= '1';
                i_scl_out    <= '0';

                if ( i_ctr = 7 ) then
                        -- do ACKOUT
                        n_state <= s_WrAck_A;
                else
                        -- increment bit counter
```

```
                        n_state <= s_Rd_F;
                end if;

        when s_Rd_F =>
                i_sclk_en    <= '0';
                i_busy       <= '1';
                i_ctr_clr    <= '0';
                i_ctr_incr   <= '1';
                i_cmd_done   <= '0';
                i_dout_ld    <= '0';
                i_ack_out_ld <= '0';
                i_sda_out    <= '1';
                i_scl_out    <= '0';

                n_state     <= s_Rd_A;

        when s_WrAck_A =>
                i_sclk_en    <= '1';
                i_busy       <= '1';
                i_ctr_clr    <= '0';
                i_ctr_incr   <= '0';
                i_cmd_done   <= '0';
                i_dout_ld    <= '0';
                i_ack_out_ld <= '0';
                i_sda_out    <= ack_in;
                i_scl_out    <= '0';

                n_state     <= s_WrAck_B;

        when s_WrAck_B =>
                i_sclk_en    <= '1';
                i_busy       <= '1';
                i_ctr_clr    <= '0';
                i_ctr_incr   <= '0';
                i_cmd_done   <= '0';
                i_dout_ld    <= '0';
                i_ack_out_ld <= '0';
                i_sda_out    <= ack_in;
                i_scl_out    <= '1';

                n_state      <= s_WrAck_C;

        when s_WrAck_C =>
                i_sclk_en    <= '1';
                i_busy       <= '1';
                i_ctr_clr    <= '0';
                i_ctr_incr   <= '0';
                i_cmd_done   <= '0';
                i_dout_ld    <= '0';
                i_ack_out_ld <= '0';
                i_sda_out    <= ack_in;
                i_scl_out    <= '1';

                n_state      <= s_WrAck_D;

        when s_WrAck_D =>
                i_sclk_en    <= '1';
                i_busy       <= '1';
                i_ctr_clr    <= '0';
                i_ctr_incr   <= '0';
                i_cmd_done   <= '0';
                i_dout_ld    <= '0';
                i_ack_out_ld <= '0';
                i_sda_out    <= ack_in;
                i_scl_out    <= '0';

                -- do a STOP ?
                if (cmd_stop = '1') then
                        n_state <= s_Stop_A;
                else
                        -- we are DONE
                        n_state <= s_Done;
                end if;

        when s_Wr_A =>
                i_sclk_en    <= '1';
                i_busy       <= '1';
                i_ctr_clr    <= '0';
                i_ctr_incr   <= '0';
                i_cmd_done   <= '0';
                i_dout_ld    <= '0';
                i_ack_out_ld <= '0';
                i_sda_out    <= i_data_in;
                i_scl_out    <= '0';

                n_state     <= s_Wr_B;

        when s_Wr_B =>
                i_sclk_en    <= '1';
                i_busy       <= '1';
                i_ctr_clr    <= '0';
                i_ctr_incr   <= '0';
                i_cmd_done   <= '0';
```

```
                i_dout_ld    <= '0';
                i_ack_out_ld <= '0';
                i_sda_out    <= i_data_in;
                i_scl_out    <= '1';

                n_state  <= s_Wr_C;

        when s_Wr_C =>
                i_sclk_en    <= '1';
                i_busy       <= '1';
                i_ctr_clr    <= '0';
                i_ctr_incr   <= '0';
                i_cmd_done   <= '0';
                i_dout_ld    <= '0';
                i_ack_out_ld <= '0';
                i_sda_out    <= i_data_in;
                i_scl_out    <= '1';

                n_state  <= s_Wr_D;

        when s_Wr_D =>
                i_sclk_en    <= '1';
                i_busy       <= '1';
                i_ctr_clr    <= '0';
                i_ctr_incr   <= '0';
                i_cmd_done   <= '0';
                i_dout_ld    <= '0';
                i_ack_out_ld <= '0';
                i_sda_out    <= i_data_in;
                i_scl_out    <= '0';

                if ( i_ctr = 7 ) then
                        -- do ACKIN
                        n_state <= s_RdAck_A;
                else
                        -- increment bit counter
                        n_state <= s_Wr_E;
                end if;

        when s_Wr_E =>
                i_sclk_en    <= '0';
                i_busy       <= '1';
                i_ctr_clr    <= '0';
                i_ctr_incr   <= '1';
                i_cmd_done   <= '0';
                i_dout_ld    <= '0';
                i_ack_out_ld <= '0';
                i_sda_out    <= i_data_in;
                i_scl_out    <= '0';

                n_state <= s_Wr_A;

        when s_RdAck_A =>
                i_sclk_en    <= '1';
                i_busy       <= '1';
                i_ctr_clr    <= '0';
                i_ctr_incr   <= '0';
                i_cmd_done   <= '0';
                i_dout_ld    <= '0';
                i_ack_out_ld <= '0';
                i_sda_out    <= '1';
                i_scl_out    <= '0';

                n_state   <= s_RdAck_B;

        when s_RdAck_B =>
                i_sclk_en    <= '1';
                i_busy       <= '1';
                i_ctr_clr    <= '0';
                i_ctr_incr   <= '0';
                i_cmd_done   <= '0';
                i_dout_ld    <= '0';
                i_ack_out_ld <= '0';
                i_sda_out    <= '1';
                i_scl_out    <= '1';

                n_state      <= s_RdAck_C;

        when s_RdAck_C =>
                i_sclk_en    <= '0';
                i_busy       <= '1';
                i_ctr_clr    <= '0';
                i_ctr_incr   <= '0';
                i_cmd_done   <= '0';
                i_dout_ld    <= '0';
                i_ack_out_ld <= '1';
                i_sda_out    <= '1';
                i_scl_out    <= '1';

                n_state      <= s_RdAck_D;


        when s_RdAck_D =>
```

```
                i_sclk_en    <= '1';
                i_busy       <= '1';
                i_ctr_clr    <= '0';
                i_ctr_incr   <= '0';
                i_cmd_done   <= '0';
                i_dout_ld    <= '0';
                i_ack_out_ld <= '0';
                i_sda_out    <= '1';
                i_scl_out    <= '1';

                n_state      <= s_RdAck_E;

        when s_RdAck_E =>
                i_sclk_en    <= '1';
                i_busy       <= '1';
                i_ctr_clr    <= '0';
                i_ctr_incr   <= '0';
                i_cmd_done   <= '0';
                i_dout_ld    <= '0';
                i_ack_out_ld <= '0';
                i_sda_out    <= '1';
                i_scl_out    <= '0';

                if ( cmd_stop = '1' ) then
                        -- do a STOP
                        n_state <= s_Stop_A;
                else
                        -- we are DONE
                        n_state <= s_Done;
                end if;

        when s_Stop_A =>
                i_sclk_en    <= '1';
                i_busy       <= '1';
                i_ctr_clr    <= '0';
                i_ctr_incr   <= '0';
                i_cmd_done   <= '0';
                i_dout_ld    <= '0';
                i_ack_out_ld <= '0';
                i_sda_out    <= '0';
                i_scl_out    <= '0';

                n_state      <= s_Stop_B;

        when s_Stop_B =>
                i_sclk_en    <= '1';
                i_busy       <= '1';
                i_ctr_clr    <= '0';
                i_ctr_incr   <= '0';
                i_cmd_done   <= '0';
                i_dout_ld    <= '0';
                i_ack_out_ld <= '0';
                i_sda_out    <= '0';
                i_scl_out    <= '1';

                n_state      <= s_Stop_C;

        when s_Stop_C =>
                i_sclk_en    <= '1';
                i_busy       <= '1';
                i_ctr_clr    <= '0';
                i_ctr_incr   <= '0';
                i_cmd_done   <= '0';
                i_dout_ld    <= '0';
                i_ack_out_ld <= '0';
                i_sda_out    <= '1';
                i_scl_out    <= '1';

                n_state <= s_Done;

        when s_Done =>
                i_sclk_en    <= '0';
                i_busy       <= '1';
                i_ctr_clr    <= '0';
                i_ctr_incr   <= '0';
                i_cmd_done   <= '1';
                i_dout_ld    <= '0';
                i_ack_out_ld <= '0';
                i_sda_out    <= sda_in;
                i_scl_out    <= scl_in;

                n_state      <= s_DoneAck;

        when s_DoneAck =>
                i_sclk_en    <= '0';
                i_busy       <= '1';
                i_ctr_clr    <= '0';
                i_ctr_incr   <= '0';
                i_cmd_done   <= '1';
                i_dout_ld    <= '0';
                i_ack_out_ld <= '0';
                i_sda_out    <= sda_in;
                i_scl_out    <= scl_in;
```

```
                        if (cmd_done_ack = '1') then
                                n_state <= s_Idle;
                        end if;

        end case;

end process i2c_comb;

end behavorial;
```

i2c_clkgen.vhd

```
----------------------------------------------------------------------
-- i2c_clkgen.vhd -- I2C base clock generator
--                   with clock stretching feature
----------------------------------------------------------------------
-- Author  : Cédric Gaudin
-- Version : 0.2 alpha
-- History :
--           20-apr-2002 CG 0.1 Initial alpha release
--           27-apr-2002 CG 0.2 minor cosmetic changes
----------------------------------------------------------------------

library ieee;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;

entity i2c_clkgen is
        port (
                signal clk     : in std_logic;
                signal rst     : in std_logic;

                -- count used for dividing clk signal
                signal clk_cnt : in std_logic_vector(7 downto 0);

                -- I2C clock output generated
                signal sclk    : out std_logic;

                -- I2C clock line SCL (used for clock stretching)
                signal scl_in  : in std_logic;
                signal scl_out : in std_logic
        );

end i2c_clkgen;

architecture behavioral of i2c_clkgen is

signal clk_ctr   : unsigned(7 downto 0);
signal clk_wait  : std_logic;
signal i_clk_out : std_logic;

begin

sclk <= i_clk_out;

process(clk, rst)
begin
        if ( rst = '1' ) then
                clk_ctr   <= (others => '0');
                i_clk_out <= '1';
        elsif (rising_edge(clk)) then
                if ( clk_ctr >= unsigned(clk_cnt) ) then
                        clk_ctr   <= (others => '0');
                        i_clk_out <= '1';
                else
                        if ( clk_wait = '0' ) then
                                clk_ctr <= clk_ctr + 1;
                        end if;
                        i_clk_out <= '0';
                end if;
        end if;
end process;

clk_wait <= '1' when (scl_out = '1' and scl_in = '0') else '0';


end behavioral;
```

i2c.h

```
/*-------------------------------------------------------------------
 * i2c.h - I2C interface library
 *-------------------------------------------------------------------
 * $Id: i2c.h,v 1.4 2002/04/14 21:50:56 cgaudin Exp $
 *-------------------------------------------------------------------
 * Author   : $Author: cgaudin $
 * Revision : $Revision: 1.4 $
 * Date     : $Date: 2002/04/14 21:50:56 $
 *-------------------------------------------------------------------*/

#ifndef _i2c_h_
#define _i2c_h_


/* I2C status register bits
 * ------------------------ */
#define I2C_TIP         (1<<3) // transfer in progress
#define I2C_IRQ         (1<<2) // interrupt pending
#define I2C_BUSY        (1<<1) // I2C busy
#define I2C_LRA         (1<<0) // last received ack

/* I2C control register bits
 * ------------------------- */
#define I2C_IRQ_ENABLE (1<<5) // interrupt enable
#define I2C_WRITE      (1<<4) // write
#define I2C_READ       (1<<3) // read
#define I2C_START      (1<<2) // start
#define I2C_STOP       (1<<1) // stop
#define I2C_ACK        (1<<0) // ack to send

/* I2C error codes
 * --------------- */
#define I2C_ENODEV     1      // no device
#define I2C_EBADACK    2      // bad acknowledge received


/* I2C Clock Divisor Standard Values
 * --------------------------------- */

/* 50 kHz */
#define I2C_SLOW   0xA3

/* 100 kHz */
#define I2C_NORMAL 0x54

/* 400 kHz */
#define I2C_FAST   0x15


/* I2C registers
 * ------------- */
typedef volatile struct
{
  int np_i2c_data;   // Read/Write, 8-bit
  int np_i2c_ctrl;   // Write-only, 8-bit
  int np_i2c_status; // Read-only,  8-bit
  int np_i2c_clkdiv; // Read/Write, 8-bit
} np_i2c;


/* function declarations
 * --------------------- */

/* Fast mode (400 kHz) */
extern void i2c_fastmode(np_i2c *io);

/* Normal mode (100 kHz) */
extern void i2c_normal_mode(np_i2c *io);

/* write one byte to a device on I2C bus */
extern int i2c_write_byte(np_i2c *io,
                          unsigned char addr,
                          unsigned char data);

/* read a byte from a device on I2C bus */
extern int i2c_read_byte(np_i2c *io,
                         unsigned char addr,
                         unsigned char *data);

#endif /* _i2c_h_ */
```

i2c.c

```
/*-------------------------------------------------------------------
 * i2c.c - I2C interface library
 *-------------------------------------------------------------------
 * $Id: i2c.c,v 1.3 2002/04/14 21:50:56 cgaudin Exp $
 *-------------------------------------------------------------------
 * Author   : $Author: cgaudin $
 * Revision : $Revision: 1.3 $
 * Date     : $Date: 2002/04/14 21:50:56 $
 *-------------------------------------------------------------------*/

#include "nios.h"
#include "i2c.h"

/* wait some microseconds */
#define WAIT { int i; for (i=0; i<100;) i++; }

/* wait for end of transfer */
#define WAIT_FOR_EOT(io,t) { while ((t=io->np_i2c_status) & I2C_TIP) WAIT; }


/* Set Serial Clock Speed in hz
 * --------------------------- */
void i2c_set_speed(np_i2c *io, int speed_in_hz)
{
  int div = nasys_clock_freq_1000 / (4 * speed_in_hz);

  if (nasys_clock_freq_1000 % (4 * speed_in_hz))
    div++;

  io->np_i2c_clkdiv = (unsigned char)div;

  /* wait a while */
  nr_delay(1);
}


/* Set FAST mode (400 kHz)
 * ---------------------- */
void i2c_fastmode(np_i2c *io)
{
  i2c_set_speed(io, 400);
}


/* Set NORMAL mode (100 kHz)
 * ------------------------ */
void i2c_normal_mode(np_i2c *io)
{
  i2c_set_speed(io, 100);
}


/* write one byte to a device on I2C bus
 * ------------------------------------ */
int i2c_write_byte(np_i2c *io,
                   unsigned char addr,
                   unsigned char data)
{
  unsigned char t;

  /* wait for end of transfer */
  WAIT_FOR_EOT(io,t);

  /* write device address + r/w=0 */
  io->np_i2c_data = (addr & 0xFE);
  io->np_i2c_ctrl = (I2C_WRITE | I2C_START);

  /* wait for end of transfer */
  WAIT_FOR_EOT(io,t);

  /* device found ? */
  if (t & I2C_LRA)
    return -I2C_ENODEV;

  /* write data byte to device */
  io->np_i2c_data = data;
  io->np_i2c_ctrl = (I2C_WRITE | I2C_STOP);

  /* wait for end of transfer */
  WAIT_FOR_EOT(io,t);

  /* bad acknowledge received ? */
  if ( t & I2C_LRA)
    return -I2C_EBADACK;

  /* clear interrupt */
  io->np_i2c_data = 0;

  /* successful completion */
```

```
  return 0;
}


/* read a byte from a device on I2C bus
 * ---------------------------------- */
int i2c_read_byte(np_i2c *io,
                  unsigned char addr,
                  unsigned char *data)
{
  unsigned char t;

  /* wait for end of transfer */
  WAIT_FOR_EOT(io, t);

  /* write device address + r/w=1 */
  io->np_i2c_data = (addr|0x01);
  io->np_i2c_ctrl = (I2C_WRITE | I2C_START);

  /* wait for end of transfer */
  WAIT_FOR_EOT(io, t);

  /* device found ? */
  if (t & I2C_LRA)
    return -I2C_ENODEV;

  /* read data byte from device */
  io->np_i2c_ctrl = (I2C_READ | I2C_STOP);

  /* wait for end of transfer */
  WAIT_FOR_EOT(io, t);

  /* read data and clear interrupt */
  (*data) = io->np_i2c_data;

  /* successful completion */
  return 0;
}
```

6. Serveur Web

Webserveur.c

```
/*-------------------------------------------------------------------------------------------------------------
 * webserver.c -- Server HTTP
 *-------------------------------------------------------------------------------------------------------------
 * Author: Sylvain Pasini
 * Revision: 0
 * Date: 20.01.03
 *-------------------------------------------------------------------------------------------------------------*/

// +------------------------------------------------------------------------------------------------------------
// | LIBRAIRIES
// +------------------------------------------------------------------------------------------------------------
// Librairie Nios
#include "nios.h"
// Librairie Module Ethernet
#include "plugs.h"
#include "wosfs.h"
#include "cs8900.h"
// Libraire Camera
#include "i2c.h"
#include "camera2d.h"

// +------------------------------------------------------------------------------------------------------------
// | CONNECTIONS
// +------------------------------------------------------------------------------------------------------------
// Define communication constants
#define k_http_port 80
#define k_string_length 256
#define k_tcp_timeout 1000000 // milliseconds

// Define the structure of a connection
typedef struct {
        int tcp_plug_handle;
        int inquiry_number;
        int button_presses[4];
        long tcp_opened_time; // So we can time out the connection
                              // in case of a dangled-open connect.
} globals;

// One connection...
static globals g = {0};

// +------------------------------------------------------------------------------------------------------------
// | PICTURES
// +------------------------------------------------------------------------------------------------------------
// Define size of camera pictures (constants)
#define width 164
#define height 126

// Define size of BMP pictures (constants)
#define width_bmp (width-2)
#define height_bmp (height-2)

// Define size of pictures sended, they are rounded at mod 4 (constants)
#define width_send ( (width_bmp) + (width_bmp)%4 )
#define height_send ( (height_bmp) + (height_bmp)%4 )

// Camera buffer in memory for camera pictures
typedef unsigned char type_pict[width*height];
volatile static type_pict picture; // 2 buffers
volatile static int picture_pointer;
volatile static int CameraAcqDone;
volatile static int nbr;

#define PACKED __attribute__ ((packed))

// Pixel structure of a BMP picture
typedef struct PACKED {
        unsigned char b ;
        unsigned char g ;
        unsigned char r ;
} rgb;

// Bitmap structure
typedef struct {
        unsigned char  bmIdent[2]    PACKED; // Identifier = bitmap for Windows
        unsigned long  bmFileSize   PACKED; // File size
        unsigned long  bmReserved    PACKED; // Reserved
        unsigned long  bmDataOffset  PACKED; // Data Offset
        unsigned long  bmHead_size   PACKED; // Header size
        unsigned long  bmWidth       PACKED; // Width
        unsigned long  bmHeight      PACKED; // Height
        unsigned short bmPlanes      PACKED; // Planes
        unsigned short bmBPP         PACKED; // Bits Per Pixel = RGB
        unsigned long  bmCompression PACKED; // Compression = BI_RGB (uncompressed)
        unsigned long  bmData_size   PACKED; // Bitmap Data size
```

```
        unsigned long  bmHres        PACKED; // Hresolution [pixels/meter] = 1640
        unsigned long  bmVres        PACKED; // Vresolution [pixels/meter] = 1260
        unsigned long  bmColors      PACKED; // Colors = any
        unsigned long  bmImp_colors  PACKED; // Imortant colors = all
        //unsigned long bmPalette;                                  // Palette = none because RGB
        rgb   bmData[width_send*width_send]   PACKED; // Bitmap Data in RGB
} bmp_file;

// File containing the RGB image with header.
static bmp_file img_bmp;

// +-----------------------------------------------------------------------------------------------------
// | PROTOTYPES
// +-----------------------------------------------------------------------------------------------------
static int  r_handle_http_request(int plug_handle,unsigned char *message,int message_length);
static void r_get_cgi_param(char *url_and_args,char *parameter,char *value_out);
static void r_set_7_pattern(int control);
static void r_increment_button_counts(void);
static char *r_printable_string(char *s);
static int  r_get_html_from_wosfs(char *url,char *html_out,int *html_length_out);
static int  r_get_html_404(char *url,char *html_out);
static int  r_get_picture(char *url,char *url_and_args);
static int  r_get_html_dynamically(char *url,char *url_and_args,char *html_out);
static int  r_substitute_string(char *fmt,char *s);

static int r_tcp_proc(int plug_handle,void *context,ns_plugs_packet *p,void *payload,int payload_length);

static int r_listen_proc(int plug_handle,void *context,host_32 remote_ip_address,host_16 remote_port);

static int r_start_listening(void);

void irq_route_camera(int contexte,int irq_number,int adresse);
static void start_complete_picture(void);
static void WaitOnCamera(void);
static void ConvertInBmp(void);

// +-----------------------------------------------------------------------------------------------------
// | MAIN
// +-----------------------------------------------------------------------------------------------------

// All printf are deleted for better performances

int main(void)
{
        int result, adr;
        np_camera_ctrl *cc = (np_camera_ctrl *)na_camera_interface;
        np_i2c *i2c        = (np_i2c *)na_i2c_camera;

        printf("[Xnedk_example_web_server] starting\n");
        /* Configure camera and initialize IRQ from camera */
        /* ----------------------------------------------- */
           /* set I2C interface to 100kHz speed */
           i2c_normal_mode(i2c);

           /* Configure the camera registers via I2C */
           camera2d_configure(i2c);

           /* Define camera interrupt routine */
           nr_installuserisr2(na_camera_interface_irq,irq_route_camera,(int)cc);

           /* start at image 0*/
           nbr = 0;


        /* Initialize the plugs library */
        /* ---------------------------- */
                result = nr_plugs_initialize(0,(void
*)0,na_ext_ethernet_module,na_ext_ethernet_module_irq,&ng_cs8900);
                // Warning, use nr_installuserisr
                if(result)
                        goto error_exit;

                /* Print the IP address we're using */
                {
                ns_plugs_network_settings settings;
                result = nr_plugs_get_settings(0,&settings);

                printf("[nedk_example_web_server] IP address = ");
                nr_plugs_print_ip_address(settings.ip_address);
                printf("\n");
                }

                result = nr_plugs_create
                                (
                                &g.tcp_plug_handle, // Plug to create
                                ne_plugs_tcp,       // Kind of plug (tcp connection)
                                k_http_port,        // Port to listen on
                                r_tcp_proc,         // To call with incoming data
                                0,                  // Context for plug callback
                                0                   // Flags for plug (none)
                                );
                if(result)
                        goto error_exit;
```

```
                /* Accept incoming connections */
                result = r_start_listening();
                if(result)
                        goto error_exit;



        /* Test camera */
        /* ----------- */
           printf("[MAIN] Start camera\n");
           start_complete_picture();

           printf("[MAIN] Wait for camera done... ");
                WaitOnCamera();
           printf("Done\n");

           printf("[MAIN] Convert in Bmp... ");
           ConvertInBmp();
           printf("Done\n");

        /* Main loop: spin through here forever
           All this application does is serve
           web pages and blink the 7-segment
                display a little. If there were other tasks
                to do, like operating your factory
                equipment or something, you could
                have it running here in this main
                event loop (if timing weren't important),
                or as a timer task.
        */
        printf("\n");
        printf("Start infinte loop... \n");

        while(1){
                nr_plugs_idle();
                r_set_7_pattern(0);

                /* Check if tcp connection's been opened too long */
                if( g.tcp_opened_time && ((nr_timer_milliseconds() - g.tcp_opened_time) > k_tcp_timeout) ){
                        char *s = "nedk_web_server: Connection timing out. Bye.\n";
                        result = nr_plugs_send(g.tcp_plug_handle,s,strlen(s),0);
                        result = r_start_listening();
                }
        }

        /* Only get here on an error condition... */
        error_exit:
                nr_plugs_print_error_message("[nedk_example_web_server]", result);
                return result;
        }

// +-----------------------------------------------------------------------------------------------------
// | R_SET_7_PATTERN
// |
// | Advance the little blinky-pattern that
// | moves around the 7-segment display
// |
// | Control Values
// |  0 advance tick
// |  1 all segments on
// +-----------------------------------------------------------------------------------------------------
static void r_set_7_pattern(int control)
{
        long x;
        static long y;
        static long last_time = 0;
        static long current_step = 0;
        static long d_pattern[] =  // 8 steps in the pattern
        {
                0xffffffbf, 0xffffffdf, 0xffffffef, 0xfffffff7,
                0xfffff7ff, 0xffffbfff, 0xffffdfff, 0xfffbffff,
        };

        // All on if in ctrl
        if (control == 1) {
                x = 0;
        }
        // One on, and if a new image forward
        else{
                if(nbr != last_time){
                        last_time = nbr;
                        current_step = (current_step + 1) & 7;
                        // Get a image
                        start_complete_picture();
                        WaitOnCamera();
                        ConvertInBmp();
                }
                else{
                        x = d_pattern[current_step];
                }
        }
```

```
        na_seven_seg_pio->np_piodirection = 0xffff;
        na_seven_seg_pio->np_piodata = x;
}

// +-----------------------------------------------------------------------------------------------------------
// | R_START_LISTENING
// |
// | This routine closes the current tcp
// | connection (if any) and starts listening
// | again.
// +-----------------------------------------------------------------------------------------------------------
static int r_start_listening(void)
{
        int result;

        result = nr_plugs_listen(g.tcp_plug_handle,r_listen_proc,0);
        g.tcp_opened_time = 0; // so it looks like we're "closed"

        return result;
}

// +-----------------------------------------------------------------------------------------------------------
// | R_LISTEN_PROC
// |
// | This is a plugs listener callback proc.
// | It gets called when a remote client
// | requests a connection. We always accept
// | the connection, but we do take the opportunity
// | to note the time of the connection. Then, in
// | the main loop, we see if the connection has
// | been left open too long, and maybe force it closed.
// +-----------------------------------------------------------------------------------------------------------
static int r_listen_proc(int plug_handle,
                void *context,
                host_32 remote_ip_address,
                host_16 remote_port)
{

        g.tcp_opened_time = nr_timer_milliseconds();

        return 0;
}

// +-----------------------------------------------------------------------------------------------------------
// | R_TCP_PROC
// |
// | This is a plugs receive callback proc.
// | It gets called when a remote client connects
// | to our web server and sends a request. We
// | assume that the request arrives in a single
// | packet (or, at least, that the first line
// | of it, containing the URL, does).
// |
// | If the payload length is zero, this means
// | for a tcp_plug that the remote network device
// | has disconnected from us, and we listen for
// | the next connection.
// +-----------------------------------------------------------------------------------------------------------
static int r_tcp_proc (
                                                                int plug_handle,
                                                                void *context,
                                                                ns_plugs_packet *p,
                                                                void *payload,
                                                                int payload_length )
{
        unsigned char *w;
        int i;
        int c;

        if(payload_length){
                // Incoming message, it must be a http request
                // We let the http_server handle this request.
                r_handle_http_request(plug_handle,payload, payload_length);
        }
        else{
                // Zero-length payload means "they disconnected",
                // so, begin listening again on same port.
                r_start_listening();
        }

go_home:
        return 0;
}

// +-----------------------------------------------------------------------------------------------------------
// | R_HANDLE_HTTP_REQUEST
// |
// | Actually, we just assume that the message
// | is, in fact, an HTTP request. An HTTP
// | request is of the form:
// |
// |    GET <url>[ <http-version>](CR)(LF)
// |
```

```
// | It may also continue with more information
// | on subsequent lines, each of the form:
// |
// |    <variable>: <value>
// |
// | And eventually ends with two CR-LF's.
// | But we ignore everything past the first
// | line here.
// |
// | To reply, we send a status line and
// | a content-type, followed by to CRLF's and the HTML.
// |
// |    HTTP/1.1 200 OK
// |    Content-type: <content-type>
// |
// |    <HTML>
// |
// | The content type is either "text/html", "image/bmp"  or "image/gif".
// | We presume that if the content comes from WOSFS, and the last letter
// | of the filename is 'f', then it's a GIF image.
// | We presume that if the content comes dynamically, and the last letter
// | of the filename is 'p', then it's a BMP image.
// +-----------------------------------------------------------------------------------------------
static int r_handle_http_request(int plug_handle,unsigned char *message, int message_length)
{
        unsigned char *message_end;
        unsigned char *w;
        int i;
        int result = 0;
        char url_and_args[k_string_length];
        char url[k_string_length];
        int url_and_args_length;

   // flash the 7-seg display
        r_set_7_pattern(1);

        ++g.inquiry_number;

        message_end = message + message_length;
        w = message;

        // Find the url_and_args portion of the packet, by looking for the
        // first space (just after the word "GET").
        while(w < message_end && *w++ != ' ');

   // omit the omnipresent leading slash
        w = w + 1;
        url_and_args_length = 0;

        // Copy the URL and any CGI arguments into url_and_args.
        // look for the end of the URL and CGI parameters as a ' ' or crlf
        // break on blank or ctrl char
        while(w < message_end && *w > ' ')
                url_and_args[url_and_args_length++] = *w++;

        url_and_args[url_and_args_length] = 0;  // | Terminate string

        // Asking for the root of the server, "/", means
        // they want "index.html".
        // empty url?
        if(url_and_args_length == 0){
                strcpy(url_and_args,"index.html");
                url_and_args_length = strlen(url_and_args);
        }

   // null param = URL
        r_get_cgi_param(url_and_args,0,url);

        // | We've extracted the url with arguments, and just the url
        // | by itself. We'll now look for a file, a dynamic page, or a 404 error.
        {
                char html_buf[100000];
                char *html_response = html_buf;
                int html_response_length = 0;
                char *http_result_code_and_message;
                char *http_content_type;
                int file_index;

                // Default to 200 OK status
                http_result_code_and_message = "200 OK";
                html_response[0] = 0;

                // Check last char of URL for content type
                if (url[strlen(url) - 1] == 'p'){
                                http_content_type = "image/bmp";

                                // Try to get the picture dynamically
                                result = r_get_picture(url,url_and_args);
                                if (! result) {
                                        html_response = (char *)&img_bmp;
                                        html_response_length = 36 + (height*width)*3 + ((height*width)*3)%4;
//=(int)img_bmp.bmFileSize; !!!
                                }
```

```
                                // Try to get the picture from wosfs
                                if (result){
                                        result = r_get_html_from_wosfs(url,html_response,&html_response_length);
                                }
                }
                else if (url[strlen(url) - 1] == 'f'){
                                http_content_type = "image/gif";

                                // Try to get the picture from wosfs
                                result = r_get_html_from_wosfs(url,html_response,&html_response_length);
                }
                else{
                                http_content_type = "text/html";

                                // Try to get the file from wosfs
                                result = r_get_html_from_wosfs(url,html_response,&html_response_length);

                                // If not wosfs, try dynamic
                                if(result){
                                        result = r_get_html_dynamically(url,url_and_args,html_response);
                                }
                }

                // And if even that fails, call our error generation routine.
                if(result){
                        http_content_type = "text/html";
                        r_get_html_404(url,html_response);
                        result = 0; // above routine not allowed to fail
                }

                // If nobody filled out the html_response_length, get it as a string length.
                if(html_response_length == 0)
                        html_response_length = strlen(html_response);

                // | Lastly, send the HTTP response.
                // | The plugs library will not automatically
                // | break up the packet into ethernet-legal
                // | chunks, so we do it here.
                {
                        char http_intro[k_string_length];

                        // Send the HTTP stuff
                        sprintf(http_intro,"HTTP/1.1 %s\r\nContent-type: %s\r\n\r\n",
                                        http_result_code_and_message,
                                        http_content_type
                                        );

                        nr_plugs_send(plug_handle,http_intro,strlen(http_intro),0); // Error : message line 2307 in
plugs.c
                }


                {
                        int this_packet;
                        int size_left = html_response_length;
                        char *w = html_response;
                        int mtu = 1024;//512;

        // if necessary wait on Camera done
        WaitOnCamera;
                        // | And send the HTML, broken up into MTU sized bits
                        while(size_left){
                                this_packet = size_left;
                                if(this_packet > mtu)
                                        this_packet = mtu;
                                nr_plugs_send(plug_handle,w,this_packet,0);
                                size_left -= this_packet;
                                w += this_packet;
                        }
                }

                // And close the connection by re-listening
                r_start_listening();
        }
}


// +-----------------------------------------------------------------------------------------------------
// | R_GET_HTML_FROM_WOSFS
// |
// | Look for a WOSFS file with a file name
// | matching the URL, and fill html_out with
// | its contents if found. If no such file
// | is found, return result -1.
// |
// | The length is also returned, since it
// | might be a gif image, and strlen will not
// | work on that.
// +-----------------------------------------------------------------------------------------------------
static int r_get_html_from_wosfs(char *url,char *html_out,int *html_length_out)
{
```

```
        ns_wosfs_directory_entry de;
        int result = 0;
        int file_index;

        file_index = nr_wosfs_get_directory_entry_by_name(url,&de);

        if(file_index >= 0){
                // | Found a wosfs file with the same name
                // | as the url requested. Get the file contents.
                // | Also, add a null terminator, to make it
                // | a C string.
                nr_wosfs_read(file_index,0,de.file_length,html_out); // should check length...
                html_out[de.file_length] = 0; // c string terminator
                *html_length_out = de.file_length; // and return the length, too
        }
        else
                result = -1;

        return result;
}


// +-----------------------------------------------------------------------------------------------------------
// | R_GET_HTML_404
// |
// | Construct a 404 error message page.
// +-----------------------------------------------------------------------------------------------------------
static int r_get_html_404(char *url,char *html_out)
{
        ns_wosfs_directory_entry de;
        int result = 0;
        int file_index;

        file_index = nr_wosfs_get_directory_entry_by_name("404_page.html",&de);
        if(file_index >= 0){
                nr_wosfs_read(file_index,0,de.file_length,html_out);
                html_out[de.file_length] = 0; // C string terminator
                // | r_substitute_string() searches for the first
                // | occurence of "%s" and replaces it with the string
                // | (in this case the URL, so the error page can say
                // | what the bad URL was)
                r_substitute_string(html_out,url);
        }
        else{
                // | This shouldnt happen, only if our wosfs
                // | is missing the file "404_page.html".
                strcpy(html_out,"<h1>404: File Not Found</h1>");
        }

        return result;
}

// +-----------------------------------------------------------------------------------------------------------
// | R_GET_PICTURE
// |
// | Construct a picture from the camera
// +-----------------------------------------------------------------------------------------------------------
static int r_get_picture(char *url,char *url_and_args)
{
        int result = 0;
        int i;                  // point on column
        int j;                  // point on ligne
        int k;                  // point on the buffer
        int l;                  // point on the bitmap
        int r,g,b;              // temporary color foar a pixel

        if(!strcmp(url,"camera.bmp")){
                // This page contain the camera picture...

                // For get a new image
                nbr = nbr+1;

                // Directly from the bmp buffer...
        }

        else if(!strcmp(url,"test.bmp")){
        // This page contain the camera picture...

                // Bitmap header
                img_bmp.bmIdent[0] = 'B';
                img_bmp.bmIdent[1] = 'M';
                img_bmp.bmReserved = 0;
                img_bmp.bmData_size = (height_send*width_send)*3 + ((height_send*width_send)*3)%4;
                img_bmp.bmFileSize = 0x36 + img_bmp.bmData_size;
                img_bmp.bmDataOffset = 0x36;
                img_bmp.bmHead_size = 40;
                img_bmp.bmWidth =  width_send;
                img_bmp.bmHeight =  height_send;
                img_bmp.bmPlanes = 1;
                img_bmp.bmBPP = 24;
                img_bmp.bmCompression = 0;
                img_bmp.bmHres = 100;
                img_bmp.bmVres = 100;
```

```
                img_bmp.bmColors = 0;
                img_bmp.bmImp_colors = 0;

                // Bitmap data
                /* Up of picture */
                l = 0;
                for (j=0; j<(height_bmp)/2; j++) {
                        for (i=0; i<(width_bmp); i++) {
                                img_bmp.bmData[l].r = 254;
                                img_bmp.bmData[l].g = 0;
                                img_bmp.bmData[l].b = 0;
                                l = l+1;
                        }
                        /* Fill the line */
                        for (r=width_bmp; r<width_send; r++){
                                img_bmp.bmData[l].r = 0;
                                img_bmp.bmData[l].g = 0;
                                img_bmp.bmData[l].b = 0;
                                l=l+1;
                        }

                }
                /* Down of picture */
                for (j=(height_bmp)/2; j<(height_bmp); j++) {
                        for (i=0; i<(width_bmp); i++) {
                                img_bmp.bmData[l].r = 0;
                                img_bmp.bmData[l].g = 255;
                                img_bmp.bmData[l].b = 0;
                                l = l+1;
                        }
                        /* Fill the line */
                        for (r=width_bmp; r<width_send; r++){
                                img_bmp.bmData[l].r = 0;
                                img_bmp.bmData[l].g = 0;
                                img_bmp.bmData[l].b = 0;
                                l = l+1;
                        }
                }

                /* Fill the picture */
                for (r=height_bmp; r<height_send; r++){
                        for (r=0; r<width_send; r++){
                                img_bmp.bmData[l].r = 0;
                                img_bmp.bmData[l].g = 0;
                                img_bmp.bmData[l].b = 0;
                                l = l+1;
                        }
                }
        }

        else{
                // | Else, it's a dynamic url we dont know about. return error.
                result = -1;
        }

        return result;
}

// +----------------------------------------------------------------------------------------------------------
// | R_GET_HTML_DYNAMICALLY
// |
// | This routine generates any dynamic html for the web server
// | It first retrieves a wosfs file named "<url>.template", if
// | any, and then checks for several possible dynamically
// | generated pages. If it's not one of those, return an error.
// +----------------------------------------------------------------------------------------------------------
static int r_get_html_dynamically(char *url,char *url_and_args,char *html_out)
{
        char template_file[k_string_length];
        ns_wosfs_directory_entry de;
        int result = 0;
        int file_index;

        // Verify if the page is a template
        strcpy(template_file,url);
        strcat(template_file,".template");
        file_index = nr_wosfs_get_directory_entry_by_name(template_file,&de);

        // If existe send the template page
        if(file_index >= 0){
                nr_wosfs_read(file_index,0,de.file_length,html_out); // should check length...
                html_out[de.file_length] = 0;
        }

        // Else is a dynamic page
        else
                html_out[0] = 0;

        if(!strcmp(url,"template_page.html")){
                // | General Info page
                // | Fill in a couple of blanks
                // | with the number of web inquiries so far,
                // | and the number of milliseconds since starting
```

```
                char s[k_string_length];

                sprintf(s,"%4d",g.inquiry_number); // s must be more than 2 chars long!
                r_substitute_string(html_out,s);

                sprintf(s,"%4d",nr_timer_milliseconds()); // s must be more than 2 chars long!
                r_substitute_string(html_out,s);
        }

        else if(!strcmp(url,"dynamic_page.html")){
                // | The memory dump page is generated entirely in this
                // | routine. No WOSFS file is used.
                char x[k_string_length];
                char *w;
                char c;
                int i;
                int j;
                int address;

                sprintf(html_out,"
                        <HTML>
                        <HEAD>
                        <TITLE>Nios</TITLE>
                        <BODY BGCOLOR=#FFFFFF>
                        <BLOCKQUOTE>
                        <H1><A HREF=index.html><IMG SRC=exc-nios.gif BORDER=0></A>Nios Memory Dump</H1>
                        <HR><BLOCKQUOTE><PRE>
                        ");
                                        r_get_cgi_param(url_and_args,"address",x);
                                        w = x;

                                        address = 0;
                                        while((c = *w++) != 0)
                                                {
                                                if(c > '9')
                                                        c += 9;
                                                address = (address << 4) + (c & 0x0f);
                                                }

                                        for(j = address; j < address + 1024; j += 32)
                                                {
                                                sprintf(html_out+strlen(html_out),"%08x: ",j);
                                                for(i = 0; i < 32; i++)
                                                        sprintf(html_out+strlen(html_out),"%02x ",*(unsigned char
*)(j + i));

                                                sprintf(html_out+strlen(html_out),"  :  ");
                                                for(i = 0; i < 32; i++)
                                                        {
                                                        c = *(unsigned char *)(j + i);
                                                        if(c < 32 || c > 0x7e)
                                                                c = '.';
                                                        if(c == '<')
                                                                sprintf(html_out+strlen(html_out),"&lt;");
                                                        else
                                                                sprintf(html_out+strlen(html_out),"%c",c);
                                                        }
                                                sprintf(html_out+strlen(html_out),"\n");
                                                }
                                        sprintf(html_out+strlen(html_out),"
                        </PRE></BLOCKQUOTE>
                        <HR>
                         <FORM ACTION=dynamic_page.html name=a_form>
                         <INPUT TYPE=submit name=next value=\"Show Next 1024 bytes\">
                         <INPUT TYPE=hidden name=address value=%08x>
                         </FORM>
                        <HR>
                         <FORM ACTION=dynamic_page.html name=b_form>
                         Base Address: <INPUT TYPE=text name=address size=30>
                         <INPUT TYPE=submit name=submit value=Show>
                         </FORM>
                        <HR>

                        <A HREF=index.html>Home</A>
                        </BLOCKQUOTE>
                        </BODY>
                        </HTML>
                        ",address+1024);
        }

        else{
                // | Else, it's a dynamic url we dont know about. return error.
                result = -1;
        }

        return result;
}

// +-----------------------------------------------------------------------------------------------
// | R_SUBSTITUTE_STRING
// |
// | Scan fmt for the first occurence of
// | "%s", and replace it with string s.
```

```
// |
// | Very useful for implementing dynamic
// | pages where a few parts are written
// | in with values or something.
// +--------------------------------------------------------------------------------------------------------
static int r_substitute_string(char *fmt,char *s)
{
        int result = 0;
        int i;
        int fmt_len;
        int s_len;

        fmt_len = strlen(fmt);
        s_len = strlen(s);
        for(i = 0; i < (fmt_len - 1); i++){
                if(fmt[i] == '%' && fmt[i + 1] == 's'){
                        // | found "%s", move stuff around a bit.
                        bcopy(fmt + i,fmt + i + s_len - 2,fmt_len - i + 1); // -2 for %s itself
                        bcopy(s,fmt + i,s_len);
                        goto go_home;
                }
        }

        result = -1; // didnt find %s, so didnt do substitution

go_home:
        return result;
}


// +--------------------------------------------------------------------------------------------------------
// | R_READ_STRING_UNTIL
// |
// | Given a pointer to a char *, read a string until
// | either a null terminator or the until_character
// | is hit. Leave the char * pointing to this
// | last character.
// +--------------------------------------------------------------------------------------------------------
static void r_read_string_until(char **wp,char *string_out,char until_character)
{
        char *w = *wp;
        char c;

        while(1){
                c = *w;

                if(c == 0 || c == until_character)
                        goto go_home;

                if(c == '+')        // | http, + means blank
                        c = ' ';
                else if(c == '%'){  // | http, %AB is hex encoding
                        char a;

                        w++;
                        c = *w;
                        if(!c)
                                goto go_home;
                        c -= '0';
                        if(c > 9)
                                c = (c & 15) + 9;

                        a = *w;
                        if(!a)
                                goto go_home;
                        a -= '0';
                        if(a > 9)
                                a = (a & 15) + 9;

                        c += (a<<4);
                }

                *string_out++ = c;
                w++;
        }
go_home:
        *string_out = 0;
        *wp = w;
}

// +--------------------------------------------------------------------------------------------------------
// | R_GET_CGI_PARAM
// |
// | Extract a single CGI parameter from the arguments
// |
// | URL and arguments are of the form
// |
// |    <url>?<param>=<val>+<param>=<val>...
// |
// | Example:  /controls/index.html?number=300&menu=fish
// |
// | This routine takes that URL and arguments string and
// | pulls out the named parameter into a string you
```

```
// | pass in. If you pass NULL for the parameter, it
// | extracts just the URL.
// | If the named parameter is not present, it returns
// | a zero-length string for the value_out.
// +-----------------------------------------------------------------------------------------------------------
static void r_get_cgi_param(char *url_and_args,char *parameter, char *value_out)
{
        char *w;
        char this_param[k_string_length];

        value_out[0] = 0; // assume empty string output

        // | First, advance past url, to the parameters
        w = url_and_args;

        // | Read the first thing, the URL
        r_read_string_until(&w,value_out,'?');

        // | parameter=NULL means we wanted the URL
        // | Which we have just found.
        if(parameter == 0)
                goto go_home;

        // | Read each param name & param value, and then see if it's
        // | the one we are looking for.

read_next_param_name:
        // | *w was left pointing at either 0 or the until_char of last segment
        value_out[0] = 0;
        if(*w++ == 0)
                goto go_home;

        value_out[0] = 0;

        r_read_string_until(&w,this_param,'=');

        if(*w++ == 0)
                goto go_home;

        r_read_string_until(&w,value_out,'&');

        if(nr_wosfs_string_match(this_param,parameter))
                return;

        // | if not, go up there & try again
        goto read_next_param_name;

go_home:
        return;
}

// +-----------------------------------------------------------------------------------------------------------
// | R_PRINTABLE_STRING
// |
// | return a char * to the string with
// | all unprintable characters replaced
// | with some oddball symbol. (Uses a
// | static string buffer, so only the
// | most recent result of this function
// | is available.)
// +-----------------------------------------------------------------------------------------------------------
static char *r_printable_string(char *s)
{
        int c;
        static char printable_string[k_string_length];
        int i = 0;

        while((c = *s++) != 0 && i < (k_string_length - 1)){
                if(c < 0x20 || c > 0x7e)
                        c = 21;
                printable_string[i++] = c;
        }
        printable_string[i] = 0;

        return printable_string;
}


// +-----------------------------------------------------------------------------------------------------------
// | IRQ_ROUTE_CAMERA(int p_img)
// |
// | Leer the fifo of the camera interface when an interrupt
// | is pending.
// +-----------------------------------------------------------------------------------------------------------
void irq_route_camera(int context,int irq_number,int adresse)
{
  register int t;
  np_camera_ctrl *cc = (np_camera_ctrl *)context;

  /* Get the status register */
  t = cc->np_cc_status;

  /* If somthing to leer */
```

```
  if (t & np_cc_status_fifo_flag_mask){

    /* Read all data in fifo */
    while ( !( (t=cc->np_cc_status) & np_cc_status_fifo_empty_mask ) ){
                  picture[picture_pointer++] = (cc->np_cc_data&0xff);
    }
  }

  if (t & np_cc_status_data_counter_flag_mask) {
                /* Read all data in fifo */
     while ( !( (t=cc->np_cc_status) & np_cc_status_fifo_empty_mask ) ){
                        picture[picture_pointer++] = (cc->np_cc_data&0xff);
                }

        /* Acknoledgment */
     CameraAcqDone = 1;
  }

  /* Clear interrupt */
  cc->np_cc_status = 0;
}

// +-----------------------------------------------------------------------------------------------------------
// | START_COMPLETE_PICTURE()
// |
// | Start the camera to get a complete picture.
// | The picture is saved in the variable "picture"
// +-----------------------------------------------------------------------------------------------------------
static void start_complete_picture()
{
                  np_camera_ctrl *cc = (np_camera_ctrl *)na_camera_interface;

            CameraAcqDone = 0;
        picture_pointer  = 0;

        /* Get all pixels */
        /* -1 because the counter start at 0 */
        cc->np_cc_limit = (width*height)-1;

        /* Clear fifo before start */
        cc->np_cc_ctrl |= np_cc_ctrl_fifo_clear_mask;

        /* Start & wait fst & interrupt enable */
        cc->np_cc_ctrl =
                    ( np_cc_ctrl_start_mask |
                      np_cc_ctrl_waitfst_mask |
                      np_cc_ctrl_intr_enable_mask);
}


// +-----------------------------------------------------------------------------------------------------------
// | WAITONCAMERA()
// |
// | Wait on the camera have termined
// +-----------------------------------------------------------------------------------------------------------
static void WaitOnCamera()
{
        while(!CameraAcqDone);
}


// +-----------------------------------------------------------------------------------------------------------
// | CONVERTINBMP()
// |
// | Wait on the camera have termined
// +-----------------------------------------------------------------------------------------------------------
static void ConvertInBmp()
{
                int i;                  // point on column
                int j;                  // point on ligne
                int k;                  // point on the buffer
                int l;                  // point on the bitmap
                unsigned int r, g ,b;   // pixel'colors (temp)

                // Bitmap header
                img_bmp.bmIdent[0] = 'B';
                img_bmp.bmIdent[1] = 'M';
                img_bmp.bmReserved = 0;
                img_bmp.bmData_size = (height_send*width_send)*3 + ((height_send*width_send)*3)%4;
                img_bmp.bmFileSize = 36 + img_bmp.bmData_size;
                img_bmp.bmDataOffset = 0x36;
                img_bmp.bmHead_size = 40;
                img_bmp.bmWidth = width_send;
                img_bmp.bmHeight = height_send;
                img_bmp.bmPlanes = 1;
                img_bmp.bmBPP = 24;
                img_bmp.bmCompression = 0;
                img_bmp.bmHres = 100;
                img_bmp.bmVres = 100;
                img_bmp.bmColors = 0;
                img_bmp.bmImp_colors = 0;

                // Bitmap Data
```

```
                l=height_send * width_send;
                for (k=j=0; j<height_bmp; j++, k+=2) {
                        for (i=0; i<width_bmp; i++, k++) {
                                r=g=b=0;

                                if (j%2) {
                                // ligne paire
                                        if (i%2) {
                                        // colonne paire
                                                r = ((unsigned int)picture[k ] +
                                                        (unsigned int)picture[k + 2 ] +
                                                        (unsigned int)picture[k + ( 2 * width ) ] +
                                                        (unsigned int)picture[k + ( 2 * width ) + 2 ]) / 4;
                                                g = ((unsigned int)picture[k + 1 ] +
                                                        (unsigned int)picture[k + width ] +
                                                        (unsigned int)picture[k + width + 2 ] +
                                                        (unsigned int)picture[k + ( 2 * width ) + 1 ]) / 4;
                                                b = ((unsigned int)picture[k + width + 1]);
                                        }
                                        else {
                                        // colonne impaire
                                                r = ((unsigned int)picture[k + 1 ] +
                                                        (unsigned int)picture[k + ( 2 * width ) + 1 ]) / 2;
                                                g = ((unsigned int)picture[k + width + 1 ]);
                                                b = ((unsigned int)picture[k + width ] +
                                                        (unsigned int)picture[k + width + 2]) / 2;
                                        }
                                }
                                else {
                                // ligne impaire
                                        if (i%2) {
                                        // colonne paire
                                                b = ((unsigned int)picture[k + 1 ] +
                                                        (unsigned int)picture[k + ( 2 * width ) + 1 ]) / 2;
                                                g = ((unsigned int)picture[k + width + 1 ]);
                                                r = ((unsigned int)picture[k + width ] +
                                                        (unsigned int)picture[k + width + 2]) / 2;
                                                }
                                        else {
                                        // colonne impaire
                                                b = ((unsigned int)picture[k ] +
                                                        (unsigned int)picture[k + 2 ] +
                                                        (unsigned int)picture[k + ( 2 * width ) ] +
                                                        (unsigned int)picture[k + ( 2 * width ) + 2 ]) / 4;
                                                g = ((unsigned int)picture[k + 1 ] +
                                                        (unsigned int)picture[k + width ] +
                                                        (unsigned int)picture[k + width + 2 ] +
                                                        (unsigned int)picture[k + ( 2 * width ) + 1 ]) / 4;
                                                r = ((unsigned int)picture[k + width + 1]);

                                        }
                                }
                                img_bmp.bmData[l].r = (unsigned char)r;
                                img_bmp.bmData[l].g = (unsigned char)g;
                                img_bmp.bmData[l].b = (unsigned char)b;
                                l = l-1;
                        }
                        /* Fill the line */
                        for (r=width_bmp; r<width_send; r++){
                                img_bmp.bmData[l].r = 0;
                                img_bmp.bmData[l].g = 0;
                                img_bmp.bmData[l].b = 0;
                                l = l-1;
                        }
                }
                /* Fill the picture */
                for (r=height_bmp; r<height_send; r++){
                        for (g=0; g<width_send; g++){
                                img_bmp.bmData[l].r = 0;
                                img_bmp.bmData[l].g = 0;
                                img_bmp.bmData[l].b = 0;
                                l = l-1;
                        }
                }
}


// end of file
```

7. Pages HTML

404_page.html

```
<HTML>
<HEAD>
        <TITLE>404: File Not Found</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
        <BLOCKQUOTE>

        <H1>
        <A HREF=index.html><IMG BORDER=0 SRC="exc-nios.gif"></A>
        404: File Not Found
        </H1>

        <TABLE><TR><TD WIDTH=1024>
        <HR>
        <H3> Error </H3>
        The URL you tried to access, <b><i>%s</b></i>, was not found
        on this server, nedk_example_web_server.
        <HR>
        <A HREF=index.html>Home</A>

        </TD></TR></TABLE>

        </BLOCKQUOTE>
</BODY>
</HTML>
```

Camera.html

```
<HTML>
<HEAD>
         <TITLE>Welcome to Camera 2D</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
         <BLOCKQUOTE>

         <H1>
         <IMG SRC="exc-nios.gif">
         Welcome To Nios EDK
         </H1>

         <TABLE><TR><TD WIDTH=1024>
         <HR>
         <H3> This image is refreshed every periode time </H3>

         <HR>
        <iframe name="I1" height="1500" width="1024"  border="0" frameborder="0" src="camera_pool.html">
        Your browser does not support inline frames or is currently configured not to display inline
frames.</iframe></td>
        <td width="45" valign="top">

         <HR>
         <FONT SIZE="+1">
         <A HREF=index.html>Home</A><BR>
         <HR>


         </TD></TR></TABLE>

         </BLOCKQUOTE>

</BODY>
</HTML>
```

Camera_pool.html

```
<HTML><HEAD><TITLE>JavaScript updated 164x124 BMP Image</TITLE>

<META content="text/html; charset=windows-1252" http-equiv=Content-Type>

<META content=no-cache http-equiv=Pragma>


<SCRIPT language="JavaScript">
 // Set the BaseURL to the url of your camera
 // Note: Since this file is located inside the unit itself, no base url is specified here
 // Example:  var BaseURL = "http://172.21.1.122/";
 var BaseURL = "http://206.48.56.159/";

 var vitesse = 500;
 var taille = 1;
 var cpt = 30;

 // This is the filepath to the image generating file inside the camera itself
 var File = "camera.bmp?resolution=164x124";

 // Force an immediate image load
 var theTimer = setTimeout("first()",1);

 function first() {
    now = new Date();
    var url = File;
    url += "&dummy=";
    url += now.getTime().toLocaleString();
    document.theImage.src = url;
    document.theImage.width = 100*taille;
    clearTimeout(theTimer);
    theTimer = setTimeout("reloadImage()",vitesse);

    afficher();
    afficher_size()
 }

 function reloadImage() {
  if (document.theImage.complete==true){
    now = new Date();
    var url = File;
    url += "&dummy=";
    url += now.getTime().toLocaleString();
    document.theImage.src = url;
    document.theImage.width = 128*taille;
    clearTimeout(theTimer);
    theTimer = setTimeout("reloadImage()",vitesse);
    cpt = 0;
  }

  if (document.theImage.complete==false){
    //image non complete, retour dans 50 ms
    clearTimeout(theTimer);
    theTimer = setTimeout("reloadImage()",50);
    cpt = cpt+1;
  }
 }

 function taille_plus(delta) {
   vitesse = vitesse + delta;
   afficher();
 }
 function taille_moins(delta) {
   if (delta==100 && vitesse>100){
     vitesse=vitesse-100;
   }
   if (delta==1000 && vitesse>=1100){
     vitesse=vitesse-1000;
   }
   afficher();
 }
 function afficher() {
  document.form2.output.value=(vitesse+' ms');
 }

 function size_plus(coeff) {
   taille = taille*coeff;
   afficher_size();
 }
 function afficher_size() {
  document.form3.output.value=(taille+' x');
 }

</SCRIPT>
<!-- End of image display part  -->


<BODY topmargin="0" leftmargin="0" bgcolor="#FFFFFF">
```

```
         Période :  (>=400ms)<FORM NAME="form2">
          <INPUT TYPE="text" NAME="output" VALUE="">
         </FORM>
         <FORM NAME="form1">
          <INPUT TYPE="button" NAME="plus 100ms" VALUE="  + 100ms  " onClick="taille_plus(100)">
          <INPUT TYPE="button" NAME="moins 100ms" VALUE="  - 100ms  " onClick="taille_moins(100)">
         </FORM>
         <FORM NAME="form5">
          <INPUT TYPE="button" NAME="plus 1s" VALUE=" + 1000ms " onClick="taille_plus(1000)">
          <INPUT TYPE="button" NAME="moins 1s" VALUE=" - 1000ms " onClick="taille_moins(1000)">
         </FORM>

          <HR>
          Taille : <FORM NAME="form3">
          <INPUT TYPE="text" NAME="output" VALUE=""></FORM>
         <FORM NAME="form4">
          <INPUT TYPE="button" NAME="plus2" VALUE=" x 2 " onClick="size_plus(2)">
          <INPUT TYPE="button" NAME="moin2" VALUE=" / 2 " onClick="size_plus(1/2)"></FORM>

          <HR>
         <p align="left" style="margin-top: 0; margin-bottom: 0"><IMG alt="Live Image" name=theImage> </p>

</BODY>
</HTML>
```

Index.html

```
<HTML>
<HEAD>
        <TITLE>Welcome to the Nios EDK</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
        <BLOCKQUOTE>

        <H1>
        <IMG SRC="exc-nios.gif">
        Welcome To Nios EDK
        </H1>

        <TABLE><TR><TD WIDTH=1024>
        <HR>
        This is the <b>web server<i></i></b>. He can serve different kinds of pages :

        <OL>
                <LI> <b><i>Static Pages.</i></b> The page is served directly from flash memory.
                        (GIF images can also be served from flash memory.) </LI>
                <LI> <b><i>Demo Picture in BMP.</i></b> The picture is served directly from flash memory. </LI>
                <LI> <b><i>Test Picture in BMP.</i></b> The picture is generated entirely by code. </LI>
                <LI> <b><i>Camera Picture in BMP.</i></b> The picture from camera is generated entirely by code.
</LI>
                <LI> <b><i>Page of Camera Picture.</i></b> The page is served directly from flash memory.
                        Picture from camera is generated entirely by code. </LI>
                <LI> <b><i>Template Pages.</i></b> The page is retrieved from flash memory, and
                        code in the server replaces each occurence of %s with some new value. </LI>
                <LI> <b><i>Dynamic Pages.</i></b> The page is generated entirely by code
                        in the server. </LI>
                <LI> <b><i>404 Error Page.</i></b> If none of previous pages can
                        be served for a particular URL, then an error page is generated. </LI>
        </OL>

        <HR>
        <H3> Example of Each Kind of Page </H3>

        <FONT SIZE="+1">
        <b>Test pictures :</b><BR>
          <LI><A HREF=img.bmp>A Démo Picture in BMP</A><BR></LI>
          <LI><A HREF=test.bmp>A Test Picture in BMP</A><BR></LI>
        <b>Pictures from camera :</b><BR>
          <LI><A HREF=camera.bmp>A Camera Picture in BMP</A><BR></LI>
          <LI><A HREF=camera.html>Page of camera pictures</A><BR></LI>
          <LI><A HREF=camera_pool.html>Camera pictures in live</A><BR></LI>
        <b>Some pages exemples :</b><BR>
          <LI><A HREF=static_page.html>A Static Page</A><BR></LI>
          <LI><A HREF=template_page.html>A Template Page</A><BR></LI>
          <LI><A HREF=dynamic_page.html>A Dynamic Page</A><BR></LI>
        No existent link :</b><BR>
        <LI><A HREF=a_bad_link.zzzhtml>A Bad Link</A><BR></LI>
        <HR>


        </TD></TR></TABLE>

        </BLOCKQUOTE>
</BODY>
</HTML>
```

Static_page.html

```
<HTML>
<HEAD>
        <TITLE>Welcome to the Nios EDK</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
        <BLOCKQUOTE>

        <H1>
        <A HREF=index.html><IMG BORDER=0 SRC="exc-nios.gif"></A>
        Welcome To Nios EDK
        </H1>

        <TABLE><TR><TD WIDTH=1024>
        <HR>
        <H3> Static Page </H3>
        This page is served statically, straight out of flash memory.
        The great thing about <H1> HTML </H1> is that the
        browser does all the <BLINK><FONT FACE=arial><b>HARD WORK</b></FONT></BLINK>.
        <HR>
        <A HREF=index.html>Home</A>

        </TD></TR></TABLE>

        </BLOCKQUOTE>
</BODY>
</HTML>
```