```
MODULE SysDef;  (*For use with Portable Extended Oberon System*)
  IMPORT SYSTEM;
  CONST VERSION* = "SysDef Feb/11/2020";
(*System Definition Module provides addresses, constants, and simple procedures for
the major hardware interfacing. What is included: The major hardware of the main
board, such as memory size and addresses, CPU interface registers, or framebuffer
memory. What is not included: Minor or transient hardware such as PMOD modules,
temperature sensors, or other application specific hardware which is factored out
to their own subsystems.
The documentation section (after the module) lists all the constants from all
modules. This section will be eventually folded away.
Note: The same constants are hard coded in several modules, copied in the
documentation even when they repeat.
-----------------------------------------------------------------------------*)
(*The peripheral address map from PO.Computer page 13.
#  Unsigned    Signed   input               output              name in    name out
0  0FFFFFFC0H  -64     ms counter          reserved            TIMER      -
4  0FFFFFFC4H  -60     switches            LEDs                SWITCH     LEDS
8  0FFFFFFC8H  -56     RS-232 data         RS-232 data         RSDATA     RSDATA
12 0FFFFFFCCH  -52     RS-232 status       RS-232 control      RSSTATUS   RSCTRL
16 0FFFFFFD0H  -48     SPI data (SD, net) SPI data (SD, net)  SPIDATA    SPIDATA
20 0FFFFFFD4H  -44     SPI status          SPI control         SPISTATUS  SPICTRL
24 0FFFFFFD8H  -40     PS/2 keyboard       --                  KEYBRD     -
28 0FFFFFFDCH  -36     mouse               --                  MOUSE      -
All these are negative numbers in Oberon, which does not provide unsigned integers.
Example: SPI data address calculated with Windows XP Calculator in DWORD display
mode. Switch to Dec. Enter "-48". Switch to Hex --> FFFFFFD0, same as above.

The memory-mapped peripheral address in software is FFFFFFxx. It is NOT used in
RISC5top.v, because the upper 8 address bits are NOT routed from RISC5. The actual
address used in RISC5top.v is rather 3FFFFxx. The prefix 3FFFF will change in
firmware, when more address bits are routed in Verilog. Fortunately, the software
memory map will stay the same. Addresses such as "-40" do not need to change.
-----------------------------------------------------------------------------*)
   (* Peripheral devices *)
   TIMER*     = -64;     (*input*)
   SWITCH*    = -60;     (*input*)
   LEDS*      = -60;     (*output*)
   RSDATA*    = -56;     (*input and output*)
   RSSTATUS*  = -52;     (*input*)
   RSCTRL*    = -52;     (*output*)
   SPIDATA*   = -48;     (*input and output*)
   SPISTATUS* = -44;     (*input*)
   SPICTRL*   = -44;     (*output*)
   KEYBRD*    = -40;     (*input. See WARNING *)
   MOUSE*     = -36;     (*input. See WARNING *)
(*WARNING. In Input.Mod these numbers were swapped: -40 <--> -36 *)
```

```
(*----------------------------------------------------------------------------
The CPU register map from PO.System page 13 page 104.
The RISC processor features 16 registers (of 32 bits). R0 - R11 are used for
expression evaluation. R12 - R15 have fixed, system-wide usage. Note that "12" can
be either a register or a memory address, and these are not the same. Registers are
accessed with REG and LDREG, while memory is accessed with GET and PUT. *)
   (* CPU registers accessible with SYSTEM.REG and SYSTEM.LDREG *)
   REG12*  = 12;  (* Address of the module table MT (typically constant)*)
   REG13*  = 13;  (* Static Base address for variables in the current module SB*)
   REG14*  = 14;  (* Stack Pointer SP *)
   REG15*  = 15;  (* Return address LNK (fixed by RISC's BL instruction) *)
   MTOrg*  = 20H; (* Module Table actual mem address to be kept in MT (Reg 12) *)
   MT*    = REG12; (* Reg 12 alias is Module Table (R12 keeps the address of MT) *)
   SB*    = REG13; (* Reg 13 alias is Static Base *)
   SP*    = REG14; (* Reg 14 alias is Stack Pointer *)
   LNK*   = REG15; (* Reg 15 alias is LNK *)

(*Trap handling: PO.Applications page 77. Trap is installed in the first address of
the Module Table, that is MTOrg = 20H. This is because the module numbers start at
1. There is no module numbered 0. The 0-th location of the Module Table is thus
free. It is used to install the trap.*)
(*----------------------------------------------------------------------------
Memory addresses from addr=0 to FFFF_FFFF. Boot parameters are near 0. Peripherals
are near the max address. Video RAM is below peripherals, but above MemLim. *)
   (* Memory map; PO.System page 104 *)
   STACKSIZE* = 8000H;      (* 32 kB. Was hardwired in Kernel.Init *)
   STACKORG*  = 80000H;     (* stackOrg = 524,288; half a megabyte. *)
   HEAPORG*   = STACKORG;   (* heapOrg = stackOrg *)
   FSoffset*  = STACKORG;   (* Not clear what it is *)
   MEMLIM*    = 0E7EF0H;    (* 1 MB minus 98,575 bytes of VRAM at the end*)
   VRAMORG*   = 0E7F00H;    (* start of memory-mapped display frame*)
   VRAMSIZE*  = 1024*768 DIV 8;(* 1024 x 768 pixel, monocolor display frame*)
   (* heapOrg   = stackOrg on page 104; this is where the heap starts *)
   (* heapLimit = MemLim on page 104; this is where the heap ends     *)
(*-----------------------------------------------------------------------------*)
(*Hardwired memory addresses 0, 4, 12, 16, 20, 24 are explained in PO.Applications,
chapter 14.1, page 74 (2nd part of the book "Project Oberon 2013 Edition"). Address
4 is an interrupt vector explained on page 2 of RISC5.Update.pdf. All these are
memory locations accessible with PUT and GET*)

   (* Memory locations accessible with SYSTEM.GET and SYSTEM.PUT *)
   SysStartAdr*   =  0; (*Branch instruction to the init body of Modules*)
   IntVecAdr*     =  4; (*Interrupt vector*)
   MemLimAdr*     = 12; (*The limit of available memory*)
   ModAllocAdr*   = 16; (*The address of the end of the module space*)
   ModRootAdr*    = 20; (*The current root of the links of loaded modules*)
   ModLimAdr*     = 24; (* The current limit of the module area*)

  (* Copy and paste the entire following block to the relevant modules *)

  (* BLOCK starts on the next line
  (* PO.Applications, chapter 14.1, page 74; RISC5.Update page 2 *)
  SysStartAdr = SysDef.SysStartAdr; (* =  0;*)
  IntVecAdr   = SysDef.IntVecAdr;   (* =  4; *)
  MemLimAdr   = SysDef.MemLimAdr;   (* = 12; *)
  ModAllocAdr = SysDef.ModAllocAdr; (* = 16; *)
  ModRootAdr  = SysDef.ModRootAdr;  (* = 20; *)
  ModLimAdr   = SysDef.ModLimAdr;   (* = 24; *)
  BLOCK ends on the previous line*)

(*----------------------------------------------------------------------------*)
(* Auxiliary definitions gathered by inspecting various modules*)
   CARD0* = 1;      (* The single SD disk with Oberon File System*)
   SPIFAST* = 4;    (* SPI mode. To be verified. Fast is "2" in SCC.Mod *)
(*----------------------------------------------------------------------------*)
(* Gaining access to registers at higher level w/o importing SYSTEM
will require a few regular exported procedures of the form GetReg,
LoadReg, Get, Put. These are defined below.*)

(* SYSTEM services can be used at higher level w/o importing SYSTEM.*)
(* These services cannot be imported by a standalone MODULE*
 Therefore, if you want to use SysDef.Mod for the BootLoader, then
the following must be commented out.*)

   PROCEDURE LoadReg* (adr, value: INTEGER);
     BEGIN SYSTEM.LDREG (adr, value) END LoadReg;

   PROCEDURE GetReg* (adr: INTEGER): INTEGER;
     VAR value: INTEGER;
     BEGIN value := SYSTEM.REG (adr); RETURN value END GetReg;

   PROCEDURE PutMem* (adr, value: INTEGER);
     BEGIN SYSTEM.PUT (adr, value) END PutMem;

   PROCEDURE GetMem* (adr: INTEGER): INTEGER;
     VAR value: INTEGER;
     BEGIN value := SYSTEM.GET (adr); RETURN value END GetMem;

END SysDef.

➔(* Reference and information section is folded here *)➔
```