```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_UNSIGNED.ALL;
use IEEE.std_logic_ARITH.ALL;
--
-- Copyright (C) 2009, Peter C. Wallace, Mesa Electronics
-- http://www.mesanet.com
--
-- This program is is licensed under a disjunctive dual license giving you
-- the choice of one of the two following sets of free software/open source
-- licensing terms:
--
--    * GNU General Public License (GPL), version 2.0 or later
--    * 3-clause BSD License
--
--
-- The GNU GPL License:
--
--     This program is free software; you can redistribute it and/or modify
--     it under the terms of the GNU General Public License as published by
--     the Free Software Foundation; either version 2 of the License, or
--     (at your option) any later version.
--
--     This program is distributed in the hope that it will be useful,
--     but WITHOUT ANY WARRANTY; without even the implied warranty of
--     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
--     GNU General Public License for more details.
--
--     You should have received a copy of the GNU General Public License
--     along with this program; if not, write to the Free Software
--     Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301 USA
--
--
-- The 3-clause BSD License:
--
--     Redistribution and use in source and binary forms, with or without
--     modification, are permitted provided that the following conditions
--     are met:
--
--         * Redistributions of source code must retain the above copyright
--           notice, this list of conditions and the following disclaimer.
--
--         * Redistributions in binary form must reproduce the above
--           copyright notice, this list of conditions and the following
--           disclaimer in the documentation and/or other materials
--           provided with the distribution.
--
--         * Neither the name of Mesa Electronics nor the names of its
--           contributors may be used to endorse or promote products
--           derived from this software without specific prior written
--           permission.
--
--
-- Disclaimer:
--
--     THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
--     "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
--     LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
--     FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
--     COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
--     INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
--     BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
--     LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
--     CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
--     LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
```

```
--     ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
--     POSSIBILITY OF SUCH DAMAGE.
--

package IDROMConst is

    constant QCountRev : std_logic_vector(7 downto 0) := x"02";
        constant MQCRev: std_logic_vector(7 downto 0) := x"03";
    constant KUBStepGenRev : std_logic_vector(7 downto 0) := x"02";

    constant NullAddr : std_logic_vector(7 downto 0) := x"00";
    constant ReadIDAddr : std_logic_vector(7 downto 0) := x"01";

    constant LEDAddr : std_logic_vector(7 downto 0) := x"02";
    constant LEDNumRegs : std_logic_vector(7 downto 0) := x"01";
    constant LEDMPBitMask : std_logic_vector(31 downto 0) := x"00000000";

    constant IDROMAddr : std_logic_vector(7 downto 0) := x"04";
    constant Cookie : std_logic_vector(31 downto 0) := x"55AACAFE";
    constant HostMotNameLow : std_logic_vector(31 downto 0) := x"54534F48";      -- HOST
    constant HostMotNameHigh : std_logic_vector(31 downto 0) := x"32544F4D";     -- MOT2

    constant BoardNameMesa : std_logic_vector(31 downto 0) := x"4153454D";       -- MESA
    constant BoardName4I65 : std_logic_vector(31 downto 0) := x"35364934";       -- 4I65
    constant BoardName4I68 : std_logic_vector(31 downto 0) := x"38364934";       -- 4I68
    constant BoardName4I69 : std_logic_vector(31 downto 0) := x"39364934";       -- 4I69
    constant BoardName4I74 : std_logic_vector(31 downto 0) := x"34374934";       -- 4I74
    constant BoardName5I20 : std_logic_vector(31 downto 0) := x"30324935";       -- 5I20
    constant BoardName5I21 : std_logic_vector(31 downto 0) := x"31324935";       -- 5I21
    constant BoardName5I22 : std_logic_vector(31 downto 0) := x"32324935";       -- 5I22
    constant BoardName5I23 : std_logic_vector(31 downto 0) := x"33324935";       -- 5I23
    constant BoardName5I24 : std_logic_vector(31 downto 0) := x"34324935";       -- 5I24
    constant BoardName5I25 : std_logic_vector(31 downto 0) := x"35324935";       -- 5I25
    constant BoardName6I25 : std_logic_vector(31 downto 0) := x"35324936";       -- 6I25
    constant BoardName7I43 : std_logic_vector(31 downto 0) := x"33344937";       -- 7I43
    constant BoardName7I60 : std_logic_vector(31 downto 0) := x"30364937";       -- 7I60
    constant BoardName7I61 : std_logic_vector(31 downto 0) := x"31364937";       -- 7I61
    constant BoardName7I62 : std_logic_vector(31 downto 0) := x"32364937";       -- 7I62
    constant BoardName7I80HD : std_logic_vector(31 downto 0) := x"30384937";     -- 7I80HD
    constant BoardName7I80DB : std_logic_vector(31 downto 0) := x"30384937";     -- 7I80DB
    constant BoardName7I77E : std_logic_vector(31 downto 0) := x"37374937";      -- 7I77E
    constant BoardName7I76E : std_logic_vector(31 downto 0) := x"36374937";      -- 7I76E
    constant BoardName7I76L : std_logic_vector(31 downto 0) := x"36374937";      -- 7I76L
    constant BoardName3X20 : std_logic_vector(31 downto 0) := x"30325833";       -- 3X20
    constant BoardName3X21 : std_logic_vector(31 downto 0) := x"30325833";       -- 3X21
    constant BoardName7I90 : std_logic_vector(31 downto 0) := x"30394937";       -- 7I90
    constant BoardName7I91 : std_logic_vector(31 downto 0) := x"31394937";       -- 7I90
    constant BoardName7I92 : std_logic_vector(31 downto 0) := x"32394937";       -- 7I90

    constant IDROMOffset : std_logic_vector(31 downto 0) := x"0000"&IDROMAddr&x"00"; -- note
need to change if pitch changed
    constant IDROMWEnAddr : std_logic_vector(7 downto 0) := x"08";

    constant IRQDivAddr  : std_logic_vector(7 downto 0) := x"09";
    constant IRQStatusAddr : std_logic_vector(7 downto 0) := x"0A";
    constant ClearIRQAddr : std_logic_vector(7 downto 0) := x"0B";
    constant IRQNumRegs : std_logic_vector(7 downto 0) := x"03";
    constant IRQMPBitMask : std_logic_vector(31 downto 0) := x"00000000";

    constant WatchdogTimeAddr : std_logic_vector(7 downto 0) := x"0C";
    constant WatchDogStatusAddr : std_logic_vector(7 downto 0) := x"0D";
    constant WatchDogCookieAddr : std_logic_vector(7 downto 0) := x"0E";
    constant WatchDogNumRegs : std_logic_vector(7 downto 0) := x"03";
    constant WatchDogMPBitMask : std_logic_vector(31 downto 0) := x"00000000";
```

```
    constant DMDMAModeAddr : std_logic_vector(7 downto 0) := x"0F"; -- demand mode DMA
    constant DMDMANumRegs : std_logic_vector(7 downto 0) := x"01";
    constant DMDMAMPBitMask : std_logic_vector(31 downto 0) := x"00000000";

    constant    PortAddr : std_logic_vector(7 downto 0) := x"10";           -- GPIO port
    constant    DDRAddr : std_logic_vector(7 downto 0) := x"11";            -- GPIO/ALT DDR
    constant    AltDataSrcAddr : std_logic_vector(7 downto 0) := x"12";
    constant    OpenDrainModeAddr : std_logic_vector(7 downto 0) := x"13";
    constant OutputInvAddr : std_logic_vector(7 downto 0) := x"14";
    constant IOPortNumRegs : std_logic_vector(7 downto 0) := x"05";
    constant IOPortMPBitMask : std_logic_vector(31 downto 0) := x"0000001F";

    constant FAbsDataAddr0 : std_logic_vector(7 downto 0) := x"15";
    constant FAbsDataAddr1 : std_logic_vector(7 downto 0) := x"16";
    constant FAbsDataAddr2 : std_logic_vector(7 downto 0) := x"17";
    constant FAbsControlAddr0 : std_logic_vector(7 downto 0) := x"18";
    constant FAbsControlAddr1 : std_logic_vector(7 downto 0) := x"19";
    constant    FAbsGlobalPStartAddr : std_logic_vector(7 downto 0) := x"1A";
    constant FAbsNumRegs : std_logic_vector(7 downto 0) := x"03";
    constant FAbsMPBitMask : std_logic_vector(31 downto 0) := x"0000001F";

    constant ScalerCountAddr : std_logic_vector(7 downto 0) := x"1B";
    constant ScalerLatchAddr : std_logic_vector(7 downto 0) := x"1C";
    constant ScalerTimerAddr : std_logic_vector(7 downto 0) := x"1D";
    constant ScalerNumRegs : std_logic_vector(7 downto 0) := x"03";
    constant ScalerMPBitMask : std_logic_vector(31 downto 0) := x"00000003";
    -- free 1E-1F

    constant StepGenRateAddr : std_logic_vector(7 downto 0) := x"20";
    constant StepGenAccumAddr : std_logic_vector(7 downto 0) := x"21";
    constant StepGenModeAddr : std_logic_vector(7 downto 0) := x"22";
    constant StepGenDSUTimeAddr : std_logic_vector(7 downto 0) := x"23";
    constant StepGenDHLDTimeAddr : std_logic_vector(7 downto 0) := x"24";
    constant StepGenPulseATimeAddr : std_logic_vector(7 downto 0) := x"25";
    constant StepGenPulseITimeAddr : std_logic_vector(7 downto 0) := x"26";
    constant StepGenTableAddr : std_logic_vector(7 downto 0) := x"27";
    constant StepGenTableMaxAddr : std_logic_vector(7 downto 0) := x"28";
    constant StepGenBasicRateAddr : std_logic_vector(7 downto 0) := x"29";
    constant StepGenTimerSelectAddr : std_logic_vector(7 downto 0) := x"2A";
    constant StepGenNumRegs : std_logic_vector(7 downto 0) := x"0A";
    constant StepGenMPBitMask : std_logic_vector(31 downto 0) := x"000001FF";

    constant WaveGenRateAddr : std_logic_vector(7 downto 0) := x"2B";
    constant WaveGenPDMRateAddr : std_logic_vector(7 downto 0) := x"2C";
    constant WaveGenLengthAddr : std_logic_vector(7 downto 0) := x"2D";
    constant WaveGenTablePtrAddr : std_logic_vector(7 downto 0) := x"2E";
    constant WaveGenTableDataAddr : std_logic_vector(7 downto 0) := x"2F";
    constant WaveGenNumRegs : std_logic_vector(7 downto 0) := x"05";
    constant WaveGenMPBitMask : std_logic_vector(31 downto 0) := x"0000001F";

    constant QCounterAddr : std_logic_vector(7 downto 0) := x"30";
    constant QCounterCCRAddr : std_logic_vector(7 downto 0) := x"31";
    constant TSDivAddr : std_logic_vector(7 downto 0) := x"32";
    constant TSAddr : std_logic_vector(7 downto 0) := x"33";
    constant QCRateAddr : std_logic_vector(7 downto 0) := x"34";
    constant QCtimerSelectAddr : std_logic_vector(7 downto 0) := x"35";
    constant QCounterNumRegs : std_logic_vector(7 downto 0) := x"05";
    constant QCounterMPBitMask : std_logic_vector(31 downto 0) := x"00000003";

    constant MuxedQCounterAddr : std_logic_vector(7 downto 0) := x"36";
    constant MuxedQCounterCCRAddr : std_logic_vector(7 downto 0) := x"37";
    constant MuxedTSDivAddr : std_logic_vector(7 downto 0) := x"38";
    constant MuxedTSAddr : std_logic_vector(7 downto 0) := x"39";
    constant MuxedQCRateAddr : std_logic_vector(7 downto 0) := x"3A";
    constant MuxedQCTimerSelectAddr : std_logic_vector(7 downto 0) := x"3B";
```

```
    constant MuxedQCounterNumRegs : std_logic_vector(7 downto 0) := x"05";
    constant MuxedQCounterMPBitMask : std_logic_vector(31 downto 0) := x"00000003";

    constant ResModCommandAddr : std_logic_vector(7 downto 0) := x"3C";     -- peculiar addressing, one set of control regs per 6 channels
    constant ResModDataAddr : std_logic_vector(7 downto 0) := x"3D";
    constant ResModStatusAddr : std_logic_vector(7 downto 0) := x"3E";
    constant ResModVelRAMAddr : std_logic_vector(7 downto 0) := x"3F";
    constant ResModPosRAMAddr : std_logic_vector(7 downto 0) := x"40";
    constant    ResModNumRegs : std_logic_vector(7 downto 0) := x"05";
    constant ResModMPBitMask : std_logic_vector(31 downto 0) := x"0000001F";

    constant PWMValAddr : std_logic_vector(7 downto 0) := x"41";
    constant PWMCRAddr : std_logic_vector(7 downto 0) := x"42";
    constant PWMRateAddr : std_logic_vector(7 downto 0) := x"43";
    constant PDMRateAddr : std_logic_vector(7 downto 0) := x"44";
    constant PWMEnasAddr : std_logic_vector(7 downto 0) := x"45";
    constant PWMNumRegs : std_logic_vector(7 downto 0) := x"05";
    constant PWMMPBitMask : std_logic_vector(31 downto 0) := x"00000003";

    constant TPPWMValAddr : std_logic_vector(7 downto 0) := x"46";
    constant TPPWMEnaAddr : std_logic_vector(7 downto 0) := x"47";
    constant TPPWMDZAddr : std_logic_vector(7 downto 0) := x"48";
    constant TPPWMRateAddr : std_logic_vector(7 downto 0) := x"49";
    constant TPPWMNumRegs : std_logic_vector(7 downto 0) := x"04";
    constant TPPWMMPBitMask : std_logic_vector(31 downto 0) := x"00000003";

    constant BISSDataAddr : std_logic_vector(7 downto 0) := x"4A";
    constant BISSControlAddr0 : std_logic_vector(7 downto 0) := x"4B";
    constant BISSControlAddr1 : std_logic_vector(7 downto 0) := x"4C";
    constant    BISSGlobalPStartAddr : std_logic_vector(7 downto 0) := x"4D";
    constant BISSNumRegs : std_logic_vector(7 downto 0) := x"04";
    constant BISSMPBitMask : std_logic_vector(31 downto 0) := x"00000007";

    constant TwiddlerCommandAddr : std_logic_vector(7 downto 0) := x"4E";       -- peculiar addressing, one set of control regs per 4-16 channels
    constant TwiddlerDataAddr : std_logic_vector(7 downto 0) := x"4F";
    constant TwiddlerRAMAddr : std_logic_vector(7 downto 0) := x"50";
    constant    TwiddlerNumRegs : std_logic_vector(7 downto 0) := x"03";
    constant TwiddlerMPBitMask : std_logic_vector(31 downto 0) := x"00000007";

    constant SPIDataAddr : std_logic_vector(7 downto 0) := x"51";
    constant SPIBitCountAddr : std_logic_vector(7 downto 0) := x"52";
    constant SPIBitrateAddr : std_logic_vector(7 downto 0) := x"53";
    constant SPINumRegs : std_logic_vector(7 downto 0) := x"03";
    constant SPIMPBitMask : std_logic_vector(31 downto 0) := x"00000007";

    constant BinOscEnaAddr : std_logic_vector(7 downto 0) := x"54";
    constant BinOscNumRegs : std_logic_vector(7 downto 0) := x"01";
    constant BinOscMPBitMask : std_logic_vector(31 downto 0) := x"00000001";

    constant BSPIDataAddr : std_logic_vector(7 downto 0) := x"55";
    constant BSPIDescriptorAddr : std_logic_vector(7 downto 0) := x"56";
    constant BSPIFIFOCountAddr : std_logic_vector(7 downto 0) := x"57";
    constant BSPINumRegs : std_logic_vector(7 downto 0) := x"03";
    constant BSPIMPBitMask : std_logic_vector(31 downto 0) := x"00000007";

    constant DBSPIDataAddr : std_logic_vector(7 downto 0) := x"58";         -- should be same as BSPI
    constant DBSPIDescriptorAddr : std_logic_vector(7 downto 0) := x"59";
    constant DBSPIFIFOCountAddr : std_logic_vector(7 downto 0) := x"5A";
    constant DBSPINumRegs : std_logic_vector(7 downto 0) := x"03";
    constant DBSPIMPBitMask : std_logic_vector(31 downto 0) := x"00000007";

    constant SSerialCommandAddr : std_logic_vector(7 downto 0) := x"5B";        -- peculiar addressing, one set of control regs per 4-16 channels
```

```
    constant SSerialCommandAddr : std_logic_vector(7 downto 0) := x"5B";        -- peculiar
addressing, one set of control regs per 4-16 channels
    constant SSerialDataAddr : std_logic_vector(7 downto 0) := x"5C";
    constant SSerialRAMAddr0 : std_logic_vector(7 downto 0) := x"5D";           -- CSR
    constant SSerialRAMAddr1 : std_logic_vector(7 downto 0) := x"5E";           -- User0
    constant SSerialRAMAddr2 : std_logic_vector(7 downto 0) := x"5F";           -- User1
    constant SSerialRAMAddr3 : std_logic_vector(7 downto 0) := x"60";           -- User2
--  constant SSerialRAMAddr4 : std_logic_vector(7 downto 0) := x"61";           -- User3
    constant    SSerialNumRegs : std_logic_vector(7 downto 0) := x"06";
    constant SSerialMPBitMask : std_logic_vector(31 downto 0) := x"0000003C";

    constant UARTTDataAddr : std_logic_vector(7 downto 0) := x"61";
    constant UARTTFIFOCountAddr : std_logic_vector(7 downto 0) := x"62";
    constant UARTTBitrateAddr: std_logic_vector(7 downto 0) := x"63";
    constant UARTTModeRegAddr : std_logic_vector(7 downto 0) := x"64";
    constant UARTTNumRegs : std_logic_vector(7 downto 0) := x"04";
    constant UARTTMPBitMask : std_logic_vector(31 downto 0) := x"0000000F";

    constant UARTRDataAddr : std_logic_vector(7 downto 0) := x"65";
    constant UARTRFIFOCountAddr : std_logic_vector(7 downto 0) := x"66";
    constant UARTRBitrateAddr : std_logic_vector(7 downto 0) := x"67";
    constant UARTRModeRegAddr : std_logic_vector(7 downto 0) := x"68";
    constant UARTRNumRegs : std_logic_vector(7 downto 0) := x"04";
    constant UARTRMPBitMask : std_logic_vector(31 downto 0) := x"0000000F";

-- note PktUART uses same addresses as normal UART with the assumption you would not use both
in one config

    constant PktUARTTDataAddr : std_logic_vector(7 downto 0) := x"61";
    constant PktUARTTFrameCountAddr : std_logic_vector(7 downto 0) := x"62";
    constant PktUARTTBitrateAddr: std_logic_vector(7 downto 0) := x"63";
    constant PktUARTTModeRegAddr : std_logic_vector(7 downto 0) := x"64";
    constant PktUARTTNumRegs : std_logic_vector(7 downto 0) := x"04";
    constant PktUARTTMPBitMask : std_logic_vector(31 downto 0) := x"0000000F";

    constant PktUARTRDataAddr : std_logic_vector(7 downto 0) := x"65";
    constant PktUARTRFrameCountAddr : std_logic_vector(7 downto 0) := x"66";
    constant PktUARTRBitrateAddr : std_logic_vector(7 downto 0) := x"67";
    constant PktUARTRModeRegAddr : std_logic_vector(7 downto 0) := x"68";
    constant PktUARTRNumRegs : std_logic_vector(7 downto 0) := x"04";
    constant PktUARTRMPBitMask : std_logic_vector(31 downto 0) := x"0000000F";

    constant SSSIDataAddr0 : std_logic_vector(7 downto 0) := x"69";
    constant SSSIDataAddr1 : std_logic_vector(7 downto 0) := x"6A";
    constant SSSIControlAddr : std_logic_vector(7 downto 0) := x"6B";
    constant    SSSIGlobalPStartAddr : std_logic_vector(7 downto 0) := x"6C";
    constant SSSINumRegs : std_logic_vector(7 downto 0) := x"04";
    constant SSSIMPBitMask : std_logic_vector(31 downto 0) := x"00000003";

    constant DAQFIFODataAddr : std_logic_vector(7 downto 0) := x"6D";
    constant DAQFIFOCountAddr : std_logic_vector(7 downto 0) := x"6E";
    constant DAQFIFOModeAddr : std_logic_vector(7 downto 0) := x"6F";
    constant DAQFIFONumRegs : std_logic_vector(7 downto 0) := x"03";
    constant DAQFIFOMPBitMask : std_logic_vector(31 downto 0) := x"00000007";

    constant HM2DPLLBaseRateAddr : std_logic_vector(7 downto 0) := x"70";
    constant HM2PhaseErrAddr : std_logic_vector(7 downto 0) := x"71";
    constant HM2DPLLControl0Addr : std_logic_vector(7 downto 0) := x"72";
    constant HM2DPLLControl1Addr : std_logic_vector(7 downto 0) := x"73";
    constant HM2DPLLTimer12Addr : std_logic_vector(7 downto 0) := x"74";
    constant HM2DPLLTimer34Addr : std_logic_vector(7 downto 0) := x"75";
    constant HM2DPLLSyncAddr : std_logic_vector(7 downto 0) := x"76";
    constant HM2DPLLNumRegs : std_logic_vector(7 downto 0) := x"07";
    constant HM2DPLLMPBitMask : std_logic_vector(31 downto 0) := x"00000000";
```

```
    -- free 77

    -- custom and probably deprecated:
    constant DPLLFreqLowAddr : std_logic_vector(7 downto 0) := x"70";  -- note overlaps
translate RAM!
    constant DPLLFreqHighAddr : std_logic_vector(7 downto 0) := x"71"; -- will fix in the great
re-alignment
    constant DPLLPostScaleAddr : std_logic_vector(7 downto 0) := x"72";
    constant DPLLIRateAddr : std_logic_vector(7 downto 0) := x"73";
    constant DPLLILimitAddr : std_logic_vector(7 downto 0) := x"74";
    constant DPLLPTweakAddr : std_logic_vector(7 downto 0) := x"75";
    constant DPLLITweakAddr : std_logic_vector(7 downto 0) := x"76";
    constant DPLLCountAddr : std_logic_vector(7 downto 0) := x"77";
    constant DPLLPhaseErrAddr : std_logic_vector(7 downto 0) := x"78";
    constant DPLLPostErrAddr : std_logic_vector(7 downto 0) := x"79";
    constant DPLLPostCountAddr : std_logic_vector(7 downto 0) := x"7A";
    constant DPLLControlAddr : std_logic_vector(7 downto 0) := x"7B";
    constant DPLLNumRegs : std_logic_vector(7 downto 0) := x"0C";
    constant DPLLMPBitMask : std_logic_vector(31 downto 0) := x"000003FF";

    constant TranslateRamAddr : std_logic_vector(7 downto 0) := x"78";
    constant TranslateRegionAddr : std_logic_vector(7 downto 0) := x"7C";
    constant TranslateNumRegs : std_logic_vector(7 downto 0) := x"04";
    constant TranslateMPBitMask : std_logic_vector(31 downto 0) := x"00000000";

    constant ClockLow20: integer :=  33333333;          -- 5I20/4I65 low speed clock
    constant ClockLow22: integer :=  48000000;      -- 5I22/5I23 low speed clock
    constant ClockLow23: integer :=  48000000;      -- 5I22/5I23 low speed clock
    constant ClockLow24: integer :=  33333333;      -- 5I24 low speed clock
    constant ClockLow25: integer :=  33333333;      -- 5I25 low speed clock
    constant ClockLow6I25: integer :=  66666666;        -- 6I25 low speed clock
    constant ClockLow21: integer :=  48000000;      -- 5I21 low speed clock
    constant ClockLow43: integer :=  50000000;      -- 7I43 low speed clock
    constant ClockLow43U: integer := 33333333;      -- 7I43U low speed clock
    constant ClockLow61: integer :=  50000000;      -- 7I61 low speed clock
    constant ClockLow68: integer :=  48000000;      -- 4I68 low speed clock
    constant ClockLow69: integer :=  50000000;      -- 4I69 low speed clock
    constant ClockLowx20: integer :=  50000000;     -- 3X20 low speed clock
    constant ClockLow76: integer :=  100000000;     -- 7I76E low speed clock
    constant ClockLow80: integer :=  100000000;     -- 7I80 low speed clock
    constant ClockLow90: integer :=  100000000;     -- 7I90 low speed clock
    constant ClockLow91: integer :=  100000000;     -- 7I90 low speed clock
    constant ClockLow92: integer :=  100000000;     -- 7I90 low speed clock

    constant ClockMed20: integer    := 50000000;        -- 5I20/4I65 medium speed clock
    constant ClockMed21: integer    := 72000000;        -- 5I21 medium speed clock
    constant ClockMed22: integer    := 72000000;        -- 5I22/5I23 medium speed clock
    constant ClockMed23: integer    := 72000000;        -- 5I22/5I23 medium speed clock
    constant ClockMed24: integer    := 100000000;   -- 5I24 medium speed clock
    constant ClockMed25: integer    := 100000000;   -- 5I25 medium speed clock
    constant ClockMed6I25: integer  := 100000000;   -- 6I25 medium speed clock
    constant ClockMed43: integer    := 75000000;        -- 7I43 medium speed clock
    constant ClockMed43U: integer   := 75000000;        -- 7I43U medium speed clock
    constant ClockMed61: integer    := 100000000;   -- 7I61 medium speed clock
    constant ClockMed68: integer    := 72000000;        -- 4I68 medium speed clock
    constant ClockMed69: integer    := 100000000;   -- 4I69 medium speed clock
    constant ClockMedx20: integer   := 75000000;        -- 3X20 medium speed clock
    constant ClockMed76: integer    := 100000000;   -- 7I76E medium speed clock
    constant ClockMed80: integer    := 100000000;   -- 7I80 medium speed clock
    constant ClockMed90: integer    := 100000000;   -- 7I90 medium speed clock
    constant ClockMed91: integer    := 100000000;   -- 7I90 medium speed clock
    constant ClockMed92: integer    := 100000000;   -- 7I90 medium speed clock

    constant ClockHigh20: integer    := 100000000;  -- 5I20/4I65 high speed clock
    constant ClockHigh21: integer    := 96000000;   -- 5I21 high speed clock
```

```
    constant ClockHigh22: integer    := 96000000;   -- 5I22/5I23 high speed clock
    constant ClockHigh23: integer    := 96000000;   -- 5I22/5I23 high speed clock
    constant ClockHigh24: integer    := 200000000;  -- 5I24 high speed clock
    constant ClockHigh25: integer    := 200000000;  -- 5I25 high speed clock
    constant ClockHigh6I25: integer  := 200000000;  -- 6I25 high speed clock
    constant ClockHigh43: integer    := 100000000;  -- 7I43 high speed clock
    constant ClockHigh43U: integer   := 100000000;  -- 7I43U high speed clock
    constant ClockHigh61: integer    := 200000000;  -- 7I61 high speed clock
    constant ClockHigh61u: integer   := 200000000;  -- 7I61U high speed clock
    constant ClockHigh68: integer    := 96000000;   -- 4I68 high speed clock
    constant ClockHigh69: integer    := 100000000;  -- 4I69 high speed clock
    constant ClockHighx20: integer   := 100000000;  -- 3X20 high speed clock
    constant ClockHigh76: integer    := 200000000;  -- 7I76E high speed clock
    constant ClockHigh80: integer    := 200000000;  -- 7I80 high speed clock
    constant ClockHigh90: integer    := 200000000;  -- 7I90 high speed clock
    constant ClockHigh91: integer    := 200000000;  -- 7I91 high speed clock
    constant ClockHigh92: integer    := 200000000;  -- 7I92 high speed clock

    constant ClockLowTag: std_logic_vector(7 downto 0) := x"01";

    constant ClockHighTag: std_logic_vector(7 downto 0) := x"02";

    constant NullTag : std_logic_vector(7 downto 0) := x"00";
        constant NullPin : std_logic_vector(7 downto 0) := x"00";

    constant IRQTag : std_logic_vector(7 downto 0) := x"01";

    constant WatchDogTag : std_logic_vector(7 downto 0) := x"02";

    constant IOPortTag : std_logic_vector(7 downto 0) := x"03";

    constant    QCountTag : std_logic_vector(7 downto 0) := x"04";
        constant QCountQAPin : std_logic_vector(7 downto 0) := x"01";
        constant QCountQBPin : std_logic_vector(7 downto 0) := x"02";
        constant QCountIdxPin : std_logic_vector(7 downto 0) := x"03";
        constant QCountIdxMaskPin : std_logic_vector(7 downto 0) := x"04";
        constant QCountProbePin : std_logic_vector(7 downto 0) := x"05";

    constant    StepGenTag : std_logic_vector(7 downto 0) := x"05";
        constant    StepGenStepPin : std_logic_vector(7 downto 0) := x"81";
        constant    StepGenDirPin : std_logic_vector(7 downto 0) := x"82";
        constant    StepGenTable2Pin : std_logic_vector(7 downto 0) := x"83";
        constant    StepGenTable3Pin : std_logic_vector(7 downto 0) := x"84";
        constant    StepGenTable4Pin : std_logic_vector(7 downto 0) := x"85";
        constant    StepGenTable5Pin : std_logic_vector(7 downto 0) := x"86";
        constant    StepGenTable6Pin : std_logic_vector(7 downto 0) := x"87";
        constant    StepGenTable7Pin : std_logic_vector(7 downto 0) := x"88";
        constant    StepGenIndexPin  : std_logic_vector(7 downto 0) := x"01";
        constant    StepGenProbePin  : std_logic_vector(7 downto 0) := x"02";

    constant PWMTag : std_logic_vector(7 downto 0) := x"06";
        constant PWMAOutPin : std_logic_vector(7 downto 0) := x"81";
        constant PWMBDirPin : std_logic_vector(7 downto 0) := x"82";
        constant PWMCEnaPin : std_logic_vector(7 downto 0) := x"83";

    constant SPITag : std_logic_vector(7 downto 0) := x"07";
        constant SPIFramePin : std_logic_vector(7 downto 0) := x"81";
        constant SPIOutPin : std_logic_vector(7 downto 0) := x"82";
        constant SPIClkPin : std_logic_vector(7 downto 0) := x"83";
        constant SPIInPin : std_logic_vector(7 downto 0) := x"04";

    constant SSSITag : std_logic_vector(7 downto 0) := x"08";
        constant SSSIClkPin : std_logic_vector(7 downto 0) := x"81";
        constant SSSIClkEnPin : std_logic_vector(7 downto 0) := x"82";
        constant SSSIDataPin : std_logic_vector(7 downto 0) := x"03";
```

```
        constant    SSSIDAVPin : std_logic_vector(7 downto 0) := x"84";

    constant UARTTTag : std_logic_vector(7 downto 0) := x"09";
        constant UTDataPin : std_logic_vector(7 downto 0) := x"81";
        constant UTDrvEnPin : std_logic_vector(7 downto 0) := x"82";

    constant UARTRTag : std_logic_vector(7 downto 0) := x"0A";
        constant URDataPin : std_logic_vector(7 downto 0) := x"01";

    constant AddrXTag : std_logic_vector(7 downto 0) := x"0B";

    constant MuxedQCountTag: std_logic_vector(7 downto 0) := x"0C";
        constant MuxedQCountQAPin : std_logic_vector(7 downto 0) := x"01";
        constant MuxedQCountQBPin : std_logic_vector(7 downto 0) := x"02";
        constant MuxedQCountIdxPin : std_logic_vector(7 downto 0) := x"03";
        constant MuxedQCountIdxMaskPin : std_logic_vector(7 downto 0) := x"04";
        constant MuxedQCountProbePin : std_logic_vector(7 downto 0) := x"05";

    constant MuxedQCountSelTag: std_logic_vector(7 downto 0) := x"0D";
        constant MuxedQCountSel0Pin : std_logic_vector(7 downto 0) := x"81";
        constant MuxedQCountSel1Pin : std_logic_vector(7 downto 0) := x"82";

    constant BSPITag : std_logic_vector(7 downto 0) := x"0E";
        constant BSPIFramePin : std_logic_vector(7 downto 0) := x"81";
        constant BSPIOutPin : std_logic_vector(7 downto 0) := x"82";
        constant BSPIClkPin : std_logic_vector(7 downto 0) := x"83";
        constant BSPIInPin : std_logic_vector(7 downto 0) := x"04";
        constant BSPICS0Pin : std_logic_vector(7 downto 0) := x"85";
        constant BSPICS1Pin : std_logic_vector(7 downto 0) := x"86";
        constant BSPICS2Pin : std_logic_vector(7 downto 0) := x"87";
        constant BSPICS3Pin : std_logic_vector(7 downto 0) := x"88";
        constant BSPICS4Pin : std_logic_vector(7 downto 0) := x"89";
        constant BSPICS5Pin : std_logic_vector(7 downto 0) := x"8A";
        constant BSPICS6Pin : std_logic_vector(7 downto 0) := x"8B";
        constant BSPICS7Pin : std_logic_vector(7 downto 0) := x"8C";

    constant DBSPITag : std_logic_vector(7 downto 0) := x"0F";
        constant DBSPIOutPin : std_logic_vector(7 downto 0) := x"82";
        constant DBSPIClkPin : std_logic_vector(7 downto 0) := x"83";
        constant DBSPIInPin : std_logic_vector(7 downto 0) := x"04";
        constant DBSPICS0Pin : std_logic_vector(7 downto 0) := x"85";
        constant DBSPICS1Pin : std_logic_vector(7 downto 0) := x"86";
        constant DBSPICS2Pin : std_logic_vector(7 downto 0) := x"87";
        constant DBSPICS3Pin : std_logic_vector(7 downto 0) := x"88";
        constant DBSPICS4Pin : std_logic_vector(7 downto 0) := x"89";
        constant DBSPICS5Pin : std_logic_vector(7 downto 0) := x"8A";
        constant DBSPICS6Pin : std_logic_vector(7 downto 0) := x"8B";
        constant DBSPICS7Pin : std_logic_vector(7 downto 0) := x"8C";

    constant DPLLTag: std_logic_vector(7 downto 0) := x"10";            --
        constant DPLLSyncInPin : std_logic_vector(7 downto 0) := x"01";
        constant DPLLMSBOutPin : std_logic_vector(7 downto 0) := x"82";
        constant DPLLFOutPin : std_logic_vector(7 downto 0) := x"83";
        constant DPLLPostOutPin : std_logic_vector(7 downto 0) := x"84";
        constant DPLLSyncTogPin : std_logic_vector(7 downto 0) := x"85";


-- these are a muxed index mask varient of the muxed q counter
-- since they will never co-exist with the non muxed index mask varient
-- they share the same register decodes
    constant MuxedQCountMIMTag: std_logic_vector(7 downto 0) := x"11";
        constant MuxedQCountMIMQAPin : std_logic_vector(7 downto 0) := x"01";
        constant MuxedQCountMIMQBPin : std_logic_vector(7 downto 0) := x"02";
        constant MuxedQCountMIMIdxPin : std_logic_vector(7 downto 0) := x"03";
        constant MuxedQCountMIMIdxMaskPin : std_logic_vector(7 downto 0) := x"04";
```

```
    constant MuxedQCountSelMIMTag: std_logic_vector(7 downto 0) := x"12";
        constant MuxedQCountSelMIM0Pin : std_logic_vector(7 downto 0) := x"81";
        constant MuxedQCountSelMIM1Pin : std_logic_vector(7 downto 0) := x"82";

    constant TPPWMTag : std_logic_vector(7 downto 0) := x"13";
        constant TPPWMAOutPin : std_logic_vector(7 downto 0) := x"81";
        constant TPPWMBOutPin : std_logic_vector(7 downto 0) := x"82";
        constant TPPWMCOutPin : std_logic_vector(7 downto 0) := x"83";
        constant NTPPWMAOutPin : std_logic_vector(7 downto 0) := x"84";
        constant NTPPWMBOutPin : std_logic_vector(7 downto 0) := x"85";
        constant NTPPWMCOutPin : std_logic_vector(7 downto 0) := x"86";
        constant TPPWMEnaPin : std_logic_vector(7 downto 0) := x"87";
        constant TPPWMFaultPin : std_logic_vector(7 downto 0) := x"08";

    constant WavegenTag : std_logic_vector(7 downto 0) := x"14";
        constant PDMAOutPin : std_logic_vector(7 downto 0) := x"81";
        constant PDMBOutPin : std_logic_vector(7 downto 0) := x"82";
        constant Trigger0OutPin : std_logic_vector(7 downto 0) := x"83";
        constant Trigger1OutPin: std_logic_vector(7 downto 0) := x"84";
        constant Trigger2OutPin : std_logic_vector(7 downto 0) := x"85";
        constant Trigger3OutPin : std_logic_vector(7 downto 0) := x"86";

    constant DAQFIFOTag : std_logic_vector(7 downto 0) := x"15";
        constant DAQFIFOStrobePin : std_logic_vector(7 downto 0) := x"41";
        constant DAQFIFOFullPin : std_logic_vector(7 downto 0) := x"81";
        constant DAQFIFOData0Pin : std_logic_vector(7 downto 0) := x"01";
        constant DAQFIFOData1Pin : std_logic_vector(7 downto 0) := x"02";
        constant DAQFIFOData2Pin : std_logic_vector(7 downto 0) := x"03";
        constant DAQFIFOData3Pin : std_logic_vector(7 downto 0) := x"04";
        constant DAQFIFOData4Pin : std_logic_vector(7 downto 0) := x"05";
        constant DAQFIFOData5Pin : std_logic_vector(7 downto 0) := x"06";
        constant DAQFIFOData6Pin : std_logic_vector(7 downto 0) := x"07";
        constant DAQFIFOData7Pin : std_logic_vector(7 downto 0) := x"08";
        constant DAQFIFOData8Pin : std_logic_vector(7 downto 0) := x"09";
        constant DAQFIFOData9Pin : std_logic_vector(7 downto 0) := x"0A";
        constant DAQFIFODataAPin : std_logic_vector(7 downto 0) := x"0B";
        constant DAQFIFODataBPin : std_logic_vector(7 downto 0) := x"0C";
        constant DAQFIFODataCPin : std_logic_vector(7 downto 0) := x"0D";
        constant DAQFIFODataDPin : std_logic_vector(7 downto 0) := x"0E";
        constant DAQFIFODataEPin : std_logic_vector(7 downto 0) := x"0F";
        constant DAQFIFODataFPin : std_logic_vector(7 downto 0) := x"10";
        constant DAQFIFOData10Pin : std_logic_vector(7 downto 0) := x"11";
        constant DAQFIFOData11Pin : std_logic_vector(7 downto 0) := x"12";
        constant DAQFIFOData12Pin : std_logic_vector(7 downto 0) := x"13";
        constant DAQFIFOData13Pin : std_logic_vector(7 downto 0) := x"14";
        constant DAQFIFOData14Pin : std_logic_vector(7 downto 0) := x"15";
        constant DAQFIFOData15Pin : std_logic_vector(7 downto 0) := x"16";
        constant DAQFIFOData16Pin : std_logic_vector(7 downto 0) := x"17";
        constant DAQFIFOData17Pin : std_logic_vector(7 downto 0) := x"18";
        constant DAQFIFOData18Pin : std_logic_vector(7 downto 0) := x"19";
        constant DAQFIFOData19Pin : std_logic_vector(7 downto 0) := x"1A";
        constant DAQFIFOData1APin : std_logic_vector(7 downto 0) := x"1B";
        constant DAQFIFOData1BPin : std_logic_vector(7 downto 0) := x"1C";
        constant DAQFIFOData1CPin : std_logic_vector(7 downto 0) := x"1D";
        constant DAQFIFOData1DPin : std_logic_vector(7 downto 0) := x"1E";

    constant BinOscTag : std_logic_vector(7 downto 0) := x"16";
        constant BinOscOut0Pin : std_logic_vector(7 downto 0) := x"81";
        constant BinOscOut1Pin : std_logic_vector(7 downto 0) := x"82";
        constant BinOscOut2Pin : std_logic_vector(7 downto 0) := x"83";
        constant BinOscOut3Pin : std_logic_vector(7 downto 0) := x"84";
        constant BinOscOut4Pin : std_logic_vector(7 downto 0) := x"85";
        constant BinOscOut5Pin : std_logic_vector(7 downto 0) := x"86";
        constant BinOscOut6Pin : std_logic_vector(7 downto 0) := x"87";
```

```
        constant BinOscOut7Pin : std_logic_vector(7 downto 0) := x"88";
        constant BinOscOut8Pin : std_logic_vector(7 downto 0) := x"89";
        constant BinOscOut9Pin : std_logic_vector(7 downto 0) := x"8A";
        constant BinOscOutAPin : std_logic_vector(7 downto 0) := x"8B";
        constant BinOscOutBPin : std_logic_vector(7 downto 0) := x"8C";
        constant BinOscOutCPin : std_logic_vector(7 downto 0) := x"8D";
        constant BinOscOutDPin : std_logic_vector(7 downto 0) := x"8E";
        constant BinOscOutEPin : std_logic_vector(7 downto 0) := x"8F";
        constant BinOscOutFPin : std_logic_vector(7 downto 0) := x"90";
        constant BinOscOut10Pin : std_logic_vector(7 downto 0) := x"91";
        constant BinOscOut11Pin : std_logic_vector(7 downto 0) := x"92";
        constant BinOscOut12Pin : std_logic_vector(7 downto 0) := x"93";
        constant BinOscOut13Pin : std_logic_vector(7 downto 0) := x"94";
        constant BinOscOut14Pin : std_logic_vector(7 downto 0) := x"95";
        constant BinOscOut15Pin : std_logic_vector(7 downto 0) := x"96";
        constant BinOscOut16Pin : std_logic_vector(7 downto 0) := x"97";
        constant BinOscOut17Pin : std_logic_vector(7 downto 0) := x"98";
        constant BinOscOut18Pin : std_logic_vector(7 downto 0) := x"99";
        constant BinOscOut19Pin : std_logic_vector(7 downto 0) := x"9A";
        constant BinOscOut1APin : std_logic_vector(7 downto 0) := x"9B";
        constant BinOscOut1BPin : std_logic_vector(7 downto 0) := x"9C";
        constant BinOscOut1CPin : std_logic_vector(7 downto 0) := x"9D";
        constant BinOscOut1DPin : std_logic_vector(7 downto 0) := x"9E";
        constant BinOscOut1EPin : std_logic_vector(7 downto 0) := x"9F";

    constant DMDMATag : std_logic_vector(7 downto 0) := x"17";

    constant BISSTag : std_logic_vector(7 downto 0) := x"18";
        constant BISSClkPin : std_logic_vector(7 downto 0) := x"81";
        constant BISSClkEnPin : std_logic_vector(7 downto 0) := x"82";
        constant BISSDataPin : std_logic_vector(7 downto 0) := x"03";
        constant    BISSDAVPin : std_logic_vector(7 downto 0) := x"84";
        constant    BISSTestDataPin : std_logic_vector(7 downto 0) := x"85";             --
    debug pin
        constant    BISSSampleTimePin : std_logic_vector(7 downto 0) := x"86";      -- debug
    pin

    constant FAbsTag : std_logic_vector(7 downto 0) := x"19";
        constant FAbsRQPin : std_logic_vector(7 downto 0) := x"81";
        constant FAbsRQEnPin : std_logic_vector(7 downto 0) := x"82";
        constant FAbsDataPin : std_logic_vector(7 downto 0) := x"03";
        constant    FAbsDAVPin : std_logic_vector(7 downto 0) := x"84";
        constant FAbsTestClkPin : std_logic_vector(7 downto 0) := x"85";

    constant HM2DPLLTag: std_logic_vector(7 downto 0) := x"1A";
        constant HM2DPLLSyncInPin : std_logic_vector(7 downto 0) := x"01";
        constant HM2DPLLRefOutPin : std_logic_vector(7 downto 0) := x"82";
        constant HM2DPLLTimer1Pin : std_logic_vector(7 downto 0) := x"83";
        constant HM2DPLLTimer2Pin : std_logic_vector(7 downto 0) := x"84";
        constant HM2DPLLTimer3Pin : std_logic_vector(7 downto 0) := x"85";
        constant HM2DPLLTimer4Pin : std_logic_vector(7 downto 0) := x"86";

    constant PktUARTTTag : std_logic_vector(7 downto 0) := x"1B";
        constant PktUTDataPin : std_logic_vector(7 downto 0) := x"81";
        constant PktUTDrvEnPin : std_logic_vector(7 downto 0) := x"82";

    constant PktUARTRTag : std_logic_vector(7 downto 0) := x"1C";
        constant PktURDataPin : std_logic_vector(7 downto 0) := x"01";

    constant ScalerCounterTag : std_logic_vector(7 downto 0)    := x"1D";
        constant ScalerCounterInA : std_logic_vector(7 downto 0)    := x"01";
        constant ScalerCounterInB : std_logic_vector(7 downto 0)    := x"02";

    constant LIOPortTag : std_logic_vector(7 downto 0) := x"40";
```

```
    constant ResModTag: std_logic_vector(7 downto 0) := x"C0";
        constant ResModPwrEnPin : std_logic_vector(7 downto 0) := x"81";
        constant ResModPDMPPin : std_logic_vector(7 downto 0) := x"82";
        constant ResModPDMMPin : std_logic_vector(7 downto 0) := x"83";
        constant ResModChan0Pin : std_logic_vector(7 downto 0) := x"84";
        constant ResModChan1Pin : std_logic_vector(7 downto 0) := x"85";
        constant ResModChan2Pin : std_logic_vector(7 downto 0) := x"86";
        constant ResModSPICSPin : std_logic_vector(7 downto 0) := x"87";
        constant ResModSPIClkPin : std_logic_vector(7 downto 0) := x"88";
        constant ResModTestBitPin : std_logic_vector(7 downto 0) := x"89";
        constant ResModSPIDI0Pin : std_logic_vector(7 downto 0) := x"09";
        constant ResModSPIDI1Pin : std_logic_vector(7 downto 0) := x"0A";

    constant SSerialTag: std_logic_vector(7 downto 0) := x"C1";
        constant SSerialRX0Pin : std_logic_vector(7 downto 0) := x"01";  -- note, 15 ports max
    per SSerial module
        constant SSerialRX1Pin : std_logic_vector(7 downto 0) := x"02";
        constant SSerialRX2Pin : std_logic_vector(7 downto 0) := x"03";
        constant SSerialRX3Pin : std_logic_vector(7 downto 0) := x"04";
        constant SSerialRX4Pin : std_logic_vector(7 downto 0) := x"05";
        constant SSerialRX5Pin : std_logic_vector(7 downto 0) := x"06";
        constant SSerialRX6Pin : std_logic_vector(7 downto 0) := x"07";
        constant SSerialRX7Pin : std_logic_vector(7 downto 0) := x"08";
        constant SSerialRX8Pin : std_logic_vector(7 downto 0) := x"09";  -- note, 15 ports max
    per SSerial module
        constant SSerialRX9Pin : std_logic_vector(7 downto 0) := x"0A";
        constant SSerialRXAPin : std_logic_vector(7 downto 0) := x"0B";
        constant SSerialRXBPin : std_logic_vector(7 downto 0) := x"0C";
        constant SSerialRXCPin : std_logic_vector(7 downto 0) := x"0D";
        constant SSerialRXDPin : std_logic_vector(7 downto 0) := x"0E";
        constant SSerialRXEPin : std_logic_vector(7 downto 0) := x"0F";
        constant SSerialTX0Pin : std_logic_vector(7 downto 0) := x"81";
        constant SSerialTX1Pin : std_logic_vector(7 downto 0) := x"82";
        constant SSerialTX2Pin : std_logic_vector(7 downto 0) := x"83";
        constant SSerialTX3Pin : std_logic_vector(7 downto 0) := x"84";
        constant SSerialTX4Pin : std_logic_vector(7 downto 0) := x"85";
        constant SSerialTX5Pin : std_logic_vector(7 downto 0) := x"86";
        constant SSerialTX6Pin : std_logic_vector(7 downto 0) := x"87";
        constant SSerialTX7Pin : std_logic_vector(7 downto 0) := x"88";
        constant SSerialTX8Pin : std_logic_vector(7 downto 0) := x"89";
        constant SSerialTX9Pin : std_logic_vector(7 downto 0) := x"8A";
        constant SSerialTXAPin : std_logic_vector(7 downto 0) := x"8B";
        constant SSerialTXBPin : std_logic_vector(7 downto 0) := x"8C";
        constant SSerialTXCPin : std_logic_vector(7 downto 0) := x"8D";
        constant SSerialTXDPin : std_logic_vector(7 downto 0) := x"8E";
        constant SSerialTXEPin : std_logic_vector(7 downto 0) := x"8F";
        constant SSerialTXEn0Pin : std_logic_vector(7 downto 0) := x"91";
        constant SSerialTXEn1Pin : std_logic_vector(7 downto 0) := x"92";
        constant SSerialTXEn2Pin : std_logic_vector(7 downto 0) := x"93";
        constant SSerialTXEn3Pin : std_logic_vector(7 downto 0) := x"94";
        constant SSerialTXEn4Pin : std_logic_vector(7 downto 0) := x"95";
        constant SSerialTXEn5Pin : std_logic_vector(7 downto 0) := x"96";
        constant SSerialTXEn6Pin : std_logic_vector(7 downto 0) := x"97";
        constant SSerialTXEn7Pin : std_logic_vector(7 downto 0) := x"98";
        constant SSerialTXEn8Pin : std_logic_vector(7 downto 0) := x"99";
        constant SSerialTXEn9Pin : std_logic_vector(7 downto 0) := x"9A";
        constant SSerialTXEnAPin : std_logic_vector(7 downto 0) := x"9B";
        constant SSerialTXEnBPin : std_logic_vector(7 downto 0) := x"9C";
        constant SSerialTXEnCPin : std_logic_vector(7 downto 0) := x"9D";
        constant SSerialTXEnDPin : std_logic_vector(7 downto 0) := x"9E";
        constant SSerialTXEnEPin : std_logic_vector(7 downto 0) := x"9F";
        constant SSerialTestPin : std_logic_vector(7 downto 0) := x"A1";

    constant TwiddlerTag: std_logic_vector(7 downto 0) := x"C2";
        constant    TwiddlerIn0Pin: std_logic_vector(7 downto 0) := x"01";  -- note 31 pins max
```

```
    constant    TwiddlerIn1Pin: std_logic_vector(7 downto 0) := x"02";
    constant    TwiddlerIn2Pin: std_logic_vector(7 downto 0) := x"03";
    constant    TwiddlerIn3Pin: std_logic_vector(7 downto 0) := x"04";
    constant    TwiddlerIn4Pin: std_logic_vector(7 downto 0) := x"05";
    constant    TwiddlerIn5Pin: std_logic_vector(7 downto 0) := x"06";
    constant    TwiddlerIn6Pin: std_logic_vector(7 downto 0) := x"07";
    constant    TwiddlerIn7Pin: std_logic_vector(7 downto 0) := x"08";
    constant    TwiddlerIn8Pin: std_logic_vector(7 downto 0) := x"09";
    constant    TwiddlerIn9Pin: std_logic_vector(7 downto 0) := x"0A";
    constant    TwiddlerInAPin: std_logic_vector(7 downto 0) := x"0B";
    constant    TwiddlerInBPin: std_logic_vector(7 downto 0) := x"0C";
    constant    TwiddlerInCPin: std_logic_vector(7 downto 0) := x"0D";
    constant    TwiddlerInDPin: std_logic_vector(7 downto 0) := x"0E";
    constant    TwiddlerInEPin: std_logic_vector(7 downto 0) := x"0F";
    constant    TwiddlerInFPin: std_logic_vector(7 downto 0) := x"10";
    constant    TwiddlerIn10Pin: std_logic_vector(7 downto 0) := x"11";
    constant    TwiddlerIn11Pin: std_logic_vector(7 downto 0) := x"12";
    constant    TwiddlerIn12Pin: std_logic_vector(7 downto 0) := x"13";
    constant    TwiddlerIn13Pin: std_logic_vector(7 downto 0) := x"14";
    constant    TwiddlerIn14Pin: std_logic_vector(7 downto 0) := x"15";
    constant    TwiddlerIn15Pin: std_logic_vector(7 downto 0) := x"16";
    constant    TwiddlerIn16Pin: std_logic_vector(7 downto 0) := x"17";
    constant    TwiddlerIn17Pin: std_logic_vector(7 downto 0) := x"18";
    constant    TwiddlerIn18Pin: std_logic_vector(7 downto 0) := x"19";
    constant    TwiddlerIn19Pin: std_logic_vector(7 downto 0) := x"1A";
    constant    TwiddlerIn1APin: std_logic_vector(7 downto 0) := x"1B";
    constant    TwiddlerIn1BPin: std_logic_vector(7 downto 0) := x"1C";
    constant    TwiddlerIn1CPin: std_logic_vector(7 downto 0) := x"1D";
    constant    TwiddlerIn1DPin: std_logic_vector(7 downto 0) := x"1E";
    constant    TwiddlerIn1EPin: std_logic_vector(7 downto 0) := x"1F";

    constant    TwiddlerIO0Pin: std_logic_vector(7 downto 0) := x"C1";
    constant    TwiddlerIO1Pin: std_logic_vector(7 downto 0) := x"C2";
    constant    TwiddlerIO2Pin: std_logic_vector(7 downto 0) := x"C3";
    constant    TwiddlerIO3Pin: std_logic_vector(7 downto 0) := x"C4";
    constant    TwiddlerIO4Pin: std_logic_vector(7 downto 0) := x"C5";
    constant    TwiddlerIO5Pin: std_logic_vector(7 downto 0) := x"C6";
    constant    TwiddlerIO6Pin: std_logic_vector(7 downto 0) := x"C7";
    constant    TwiddlerIO7Pin: std_logic_vector(7 downto 0) := x"C8";
    constant    TwiddlerIO8Pin: std_logic_vector(7 downto 0) := x"C9";
    constant    TwiddlerIO9Pin: std_logic_vector(7 downto 0) := x"CA";
    constant    TwiddlerIOAPin: std_logic_vector(7 downto 0) := x"CB";
    constant    TwiddlerIOBPin: std_logic_vector(7 downto 0) := x"CC";
    constant    TwiddlerIOCPin: std_logic_vector(7 downto 0) := x"CD";
    constant    TwiddlerIODPin: std_logic_vector(7 downto 0) := x"CE";
    constant    TwiddlerIOEPin: std_logic_vector(7 downto 0) := x"CF";
    constant    TwiddlerIOFPin: std_logic_vector(7 downto 0) := x"D0";
    constant    TwiddlerIO10Pin: std_logic_vector(7 downto 0) := x"D1";
    constant    TwiddlerIO11Pin: std_logic_vector(7 downto 0) := x"D2";
    constant    TwiddlerIO12Pin: std_logic_vector(7 downto 0) := x"D3";
    constant    TwiddlerIO13Pin: std_logic_vector(7 downto 0) := x"D4";
    constant    TwiddlerIO14Pin: std_logic_vector(7 downto 0) := x"D5";
    constant    TwiddlerIO15Pin: std_logic_vector(7 downto 0) := x"D6";
    constant    TwiddlerIO16Pin: std_logic_vector(7 downto 0) := x"D7";
    constant    TwiddlerIO17Pin: std_logic_vector(7 downto 0) := x"D8";
    constant    TwiddlerIO18Pin: std_logic_vector(7 downto 0) := x"D9";
    constant    TwiddlerIO19Pin: std_logic_vector(7 downto 0) := x"DA";
    constant    TwiddlerIO1APin: std_logic_vector(7 downto 0) := x"DB";
    constant    TwiddlerIO1BPin: std_logic_vector(7 downto 0) := x"DC";
    constant    TwiddlerIO1CPin: std_logic_vector(7 downto 0) := x"DD";
    constant    TwiddlerIO1DPin: std_logic_vector(7 downto 0) := x"DE";
    constant    TwiddlerIO1EPin: std_logic_vector(7 downto 0) := x"DF";

    constant    TwiddlerOut0Pin: std_logic_vector(7 downto 0) := x"81";
    constant    TwiddlerOut1Pin: std_logic_vector(7 downto 0) := x"82";
```

```
        constant    TwiddlerOut2Pin: std_logic_vector(7 downto 0) := x"83";
        constant    TwiddlerOut3Pin: std_logic_vector(7 downto 0) := x"84";
        constant    TwiddlerOut4Pin: std_logic_vector(7 downto 0) := x"85";
        constant    TwiddlerOut5Pin: std_logic_vector(7 downto 0) := x"86";
        constant    TwiddlerOut6Pin: std_logic_vector(7 downto 0) := x"87";
        constant    TwiddlerOut7Pin: std_logic_vector(7 downto 0) := x"88";
        constant    TwiddlerOut8Pin: std_logic_vector(7 downto 0) := x"89";
        constant    TwiddlerOut9Pin: std_logic_vector(7 downto 0) := x"8A";
        constant    TwiddlerOutAPin: std_logic_vector(7 downto 0) := x"8B";
        constant    TwiddlerOutBPin: std_logic_vector(7 downto 0) := x"8C";
        constant    TwiddlerOutCPin: std_logic_vector(7 downto 0) := x"8D";
        constant    TwiddlerOutDPin: std_logic_vector(7 downto 0) := x"8E";
        constant    TwiddlerOutEPin: std_logic_vector(7 downto 0) := x"8F";
        constant    TwiddlerOutFPin: std_logic_vector(7 downto 0) := x"90";
        constant    TwiddlerOut10Pin: std_logic_vector(7 downto 0) := x"91";
        constant    TwiddlerOut11Pin: std_logic_vector(7 downto 0) := x"92";
        constant    TwiddlerOut12Pin: std_logic_vector(7 downto 0) := x"93";
        constant    TwiddlerOut13Pin: std_logic_vector(7 downto 0) := x"94";
        constant    TwiddlerOut14Pin: std_logic_vector(7 downto 0) := x"95";
        constant    TwiddlerOut15Pin: std_logic_vector(7 downto 0) := x"96";
        constant    TwiddlerOut16Pin: std_logic_vector(7 downto 0) := x"97";
        constant    TwiddlerOut17Pin: std_logic_vector(7 downto 0) := x"98";
        constant    TwiddlerOut18Pin: std_logic_vector(7 downto 0) := x"99";
        constant    TwiddlerOut19Pin: std_logic_vector(7 downto 0) := x"9A";
        constant    TwiddlerOut1APin: std_logic_vector(7 downto 0) := x"9B";
        constant    TwiddlerOut1BPin: std_logic_vector(7 downto 0) := x"9C";
        constant    TwiddlerOut1CPin: std_logic_vector(7 downto 0) := x"9D";
        constant    TwiddlerOut1DPin: std_logic_vector(7 downto 0) := x"9E";
        constant    TwiddlerOut1EPin: std_logic_vector(7 downto 0) := x"9F";


    constant LEDTag : std_logic_vector(7 downto 0) := x"80";

    constant GlobalChan: std_logic_vector(7 downto 0) := x"80";

    constant emptypin : std_logic_vector(31 downto 0) := x"00000000";
    constant empty : std_logic_vector(31 downto 0) := x"00000000";
    constant PadT : std_logic_vector(7 downto 0) := x"00";
    constant MaxModules : integer := 32;            -- maximum number of module types
    constant MaxPins : integer := 144;              -- maximum number of I/O pins

-- would be better to change all the pindescs to records
-- but that requires reversing the byte order of the constant
-- pindesc arrays, some other day...

    type PinDescRecord is  -- not used yet!
    record
        SecPin : std_logic_vector(7 downto 0);
        SecFunc : std_logic_vector(7 downto 0);
        SecInst : std_logic_vector(7 downto 0);
        PriFunc : std_logic_vector(7 downto 0);
    end record;

    type PinDescType is array(0 to MaxPins -1) of std_logic_vector(31 downto 0);

    type ModuleRecord is                            -- probably need an alternate way for smart
modules
    record
        GTag : std_logic_vector(7 downto 0);
        Version : std_logic_vector(7 downto 0);
        Clock : std_logic_vector(7 downto 0);
        NumInstances : std_logic_vector(7 downto 0);
        BaseAddr : std_logic_vector(15 downto 0);
        NumRegisters : std_logic_vector(7 downto 0);
        Strides : std_logic_vector(7 downto 0);
```

```
        MultRegs : std_logic_vector(31 downto 0);
    end record;


    type ModuleIDType is array(0 to MaxModules-1) of ModuleRecord;


end package IDROMConst;
```