

Introduction

The Spartan™-3E family of Field-Programmable Gate Arrays (FPGAs) is specifically designed to meet the needs of high volume, cost-sensitive consumer electronic applications. The five-member family offers densities ranging from 100,000 to 1.6 million system gates, as shown in [Table 1](#).

The Spartan-3E family builds on the success of the earlier Spartan-3 family by increasing the amount of logic per I/O, significantly reducing the cost per logic cell. New features improve system performance and reduce the cost of configuration. These Spartan-3E enhancements, combined with advanced 90 nm process technology, deliver more functionality and bandwidth per dollar than was previously possible, setting new standards in the programmable logic industry.

Because of their exceptionally low cost, Spartan-3E FPGAs are ideally suited to a wide range of consumer electronics applications, including broadband access, home networking, display/projection, and digital television equipment.

The Spartan-3E family is a superior alternative to mask programmed ASICs. FPGAs avoid the high initial cost, the lengthy development cycles, and the inherent inflexibility of conventional ASICs. Also, FPGA programmability permits design upgrades in the field with no hardware replacement necessary, an impossibility with ASICs.

Features

- Very low cost, high-performance logic solution for high-volume, consumer-oriented applications
- Proven advanced 90-nanometer process technology
- Multi-voltage, multi-standard SelectIO™ interface pins
 - Up to 376 I/O pins or 156 differential signal pairs
 - LVC MOS, LVTTTL, HSTL, and SSTL single-ended signal standards
 - 3.3V, 2.5V, 1.8V, 1.5V, and 1.2V signaling
 - 622+ Mb/s data transfer rate per I/O
- True LVDS, RS DS, mini-LVDS, differential HSTL/SSTL differential I/O
- Enhanced Double Data Rate (DDR) support
- DDR SDRAM support up to 333 Mb/s
- Abundant, flexible logic resources
 - Densities up to 33,192 logic cells, including optional shift register or distributed RAM support
 - Efficient wide multiplexers, wide logic
 - Fast look-ahead carry logic
 - Enhanced 18 x 18 multipliers with optional pipeline
 - IEEE 1149.1/1532 JTAG programming/debug port
- Hierarchical SelectRAM™ memory architecture
 - Up to 648 Kbits of fast block RAM
 - Up to 231 Kbits of efficient distributed RAM
- Up to eight Digital Clock Managers (DCMs)
 - Clock skew elimination (delay locked loop)
 - Frequency synthesis, multiplication, division
 - High-resolution phase shifting
 - Wide frequency range (5 MHz to over 300 MHz)
- Eight global clocks plus eight additional clocks per each half of device, plus abundant low-skew routing
- Configuration interface to industry-standard PROMs
 - Low-cost, space-saving SPI serial Flash PROM
 - x8 or x8/x16 parallel NOR Flash PROM
 - Low-cost Xilinx [Platform Flash](#) with JTAG
- Complete Xilinx [ISE™](#) and [WebPACK™](#) development system support
- [MicroBlaze™](#) and [PicoBlaze™](#) embedded processor cores
- Fully compliant 32-/64-bit 33 MHz PCI support ([66](#) MHz in some devices)
- Low-cost QFP and BGA packaging options
 - Common footprints support easy density migration
 - Pb-free packaging options

Table 1: Summary of Spartan-3E FPGA Attributes

Device	System Gates	Equivalent Logic Cells	CLB Array (One CLB = Four Slices)				Distributed RAM bits ⁽¹⁾	Block RAM bits ⁽¹⁾	Dedicated Multipliers	DCMs	Maximum User I/O	Maximum Differential I/O Pairs
			Rows	Columns	Total CLBs	Total Slices						
XC3S100E	100K	2,160	22	16	240	960	15K	72K	4	2	108	40
XC3S250E	250K	5,508	34	26	612	2,448	38K	216K	12	4	172	68
XC3S500E	500K	10,476	46	34	1,164	4,656	73K	360K	20	4	232	92
XC3S1200E	1200K	19,512	60	46	2,168	8,672	136K	504K	28	8	304	124
XC3S1600E	1600K	33,192	76	58	3,688	14,752	231K	648K	36	8	376	156

Notes:

1. By convention, one Kb is equivalent to 1,024 bits.

Architectural Overview

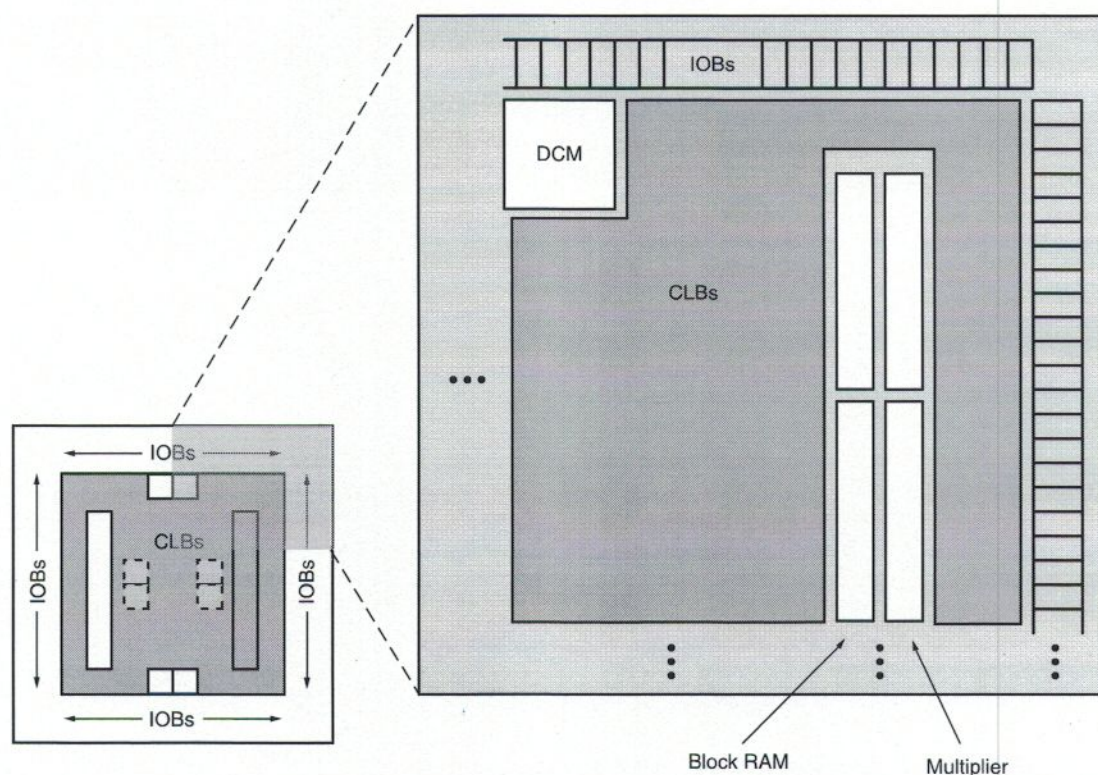
The Spartan-3E family architecture consists of five fundamental programmable functional elements:

- **Configurable Logic Blocks (CLBs)** contain flexible Look-Up Tables (LUTs) that implement logic plus storage elements used as flip-flops or latches. CLBs perform a wide variety of logical functions as well as store data.
- **Input/Output Blocks (IOBs)** control the flow of data between the I/O pins and the internal logic of the device. Each IOB supports bidirectional data flow plus 3-state operation. Supports a variety of signal standards, including four high-performance differential standards. Double Data-Rate (DDR) registers are included.
- **Block RAM** provides data storage in the form of 18-Kbit dual-port blocks.
- **Multiplier Blocks** accept two 18-bit binary numbers as inputs and calculate the product.

- **Digital Clock Manager (DCM) Blocks** provide self-calibrating, fully digital solutions for distributing, delaying, multiplying, dividing, and phase-shifting clock signals.

These elements are organized as shown in Figure 1. A ring of IOBs surrounds a regular array of CLBs. Each device has two columns of block RAM except for the XC3S100E, which has one column. Each RAM column consists of several 18-Kbit RAM blocks. Each block RAM is associated with a dedicated multiplier. The DCMs are positioned in the center with two at the top and two at the bottom of the device. The XC3S100E has only one DCM at the top and bottom, while the XC3S1200E and XC3S1600E add two DCMs in the middle of the left and right sides.

The Spartan-3E family features a rich network of traces that interconnect all five functional elements, transmitting signals among them. Each functional element has an associated switch matrix that permits multiple connections to the routing.



Notes:

1. The XC3S1200E and XC3S1600E have two additional DCMs on both the left and right sides as indicated by the dashed lines. The XC3S100E has only one DCM at the top and one at the bottom.

Figure 1: Spartan-3E Family Architecture

Configuration

Spartan-3E FPGAs are programmed by loading configuration data into robust, reprogrammable, static CMOS configuration latches (CCLs) that collectively control all functional elements and routing resources. The FPGA's configuration data is stored externally in a PROM or some other non-volatile medium, either on or off the board. After applying power, the configuration data is written to the FPGA using any of seven different modes:

- Master Serial from a Xilinx Platform Flash PROM
- Serial Peripheral Interface (SPI) from an industry-standard SPI serial Flash
- Byte Peripheral Interface (BPI) Up or Down from an industry-standard x8 or x8/x16 parallel NOR Flash
- Slave Serial, typically downloaded from a processor
- Slave Parallel, typically downloaded from a processor
- Boundary Scan (JTAG), typically downloaded from a processor or system tester.

I/O Capabilities

The Spartan-3E FPGA SelectIO interface supports many popular single-ended and differential standards. [Table 2](#) shows the number of user I/Os as well as the number of differential I/O pairs available for each device/package combination.

Spartan-3E FPGAs support the following single-ended standards:

- 3.3V low-voltage TTL (LVTTTL)
- Low-voltage CMOS (LVCMOS) at 3.3V, 2.5V, 1.8V, 1.5V, or 1.2V
- 3V PCI at 33 MHz, and in some devices, [66 MHz](#)
- HSTL I and III at 1.8V, commonly used in memory applications
- SSTL I at 1.8V and 2.5V, commonly used for memory applications

Spartan-3E FPGAs support the following differential standards:

- LVDS
- Bus LVDS
- mini-LVDS
- RSDS
- Differential HSTL (1.8V, Types I and III)
- Differential SSTL (2.5V and 1.8V, Type I)
- 2.5V LVPECL inputs

Table 2: Available User I/Os and Differential (Diff) I/O Pairs

Device	VQ100 VQG100		CP132 CPG132		TQ144 TQG144		PQ208 PQG208		FT256 FTG256		FG320 FGG320		FG400 FGG400		FG484 FGG484	
	User	Diff	User	Diff	User	Diff	User	Diff	User	Diff	User	Diff	User	Diff	User	Diff
XC3S100E	66 (7)	30 (2)	83 (11)	35 (2)	108 (28)	40 (4)	-	-	-	-	-	-	-	-	-	-
XC3S250E	66 (7)	30 (2)	92 (7)	41 (2)	108 (28)	40 (4)	158 (32)	65 (5)	172 (40)	68 (8)	-	-	-	-	-	-
XC3S500E	-	-	92 (7)	41 (2)	-	-	158 (32)	65 (5)	190 (41)	77 (8)	232 (56)	92 (12)	-	-	-	-
XC3S1200E	-	-	-	-	-	-	-	-	190 (40)	77 (8)	250 (56)	99 (12)	304 (72)	124 (20)	-	-
XC3S1600E	-	-	-	-	-	-	-	-	-	-	250 (56)	99 (12)	304 (72)	124 (20)	376 (82)	156 (21)

Notes:

1. All Spartan-3E devices provided in the same package are pin-compatible as further described in Module 4: [Pinout Descriptions](#).
2. The number shown in **bold** indicates the maximum number of I/O and input-only pins. The number shown in *(italics)* indicates the number of input-only pins.

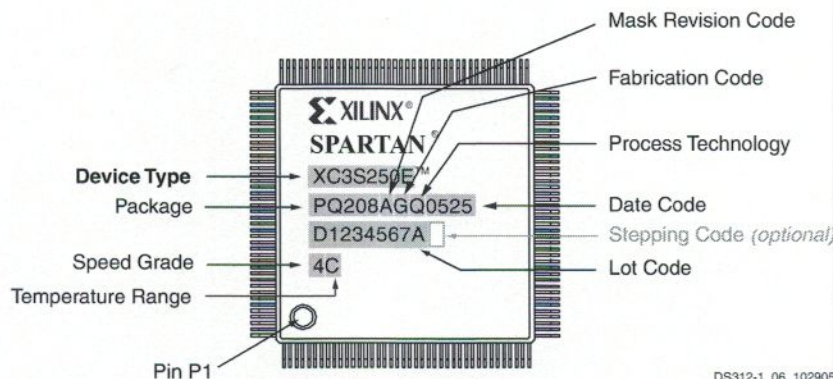
Package Marking

Figure 2 provides a top marking example for Spartan-3E FPGAs in the quad-flat packages. Figure 3 shows the top marking for Spartan-3E FPGAs in BGA packages except the 132-ball chip-scale package (CP132 and CPG132). The markings for the BGA packages are nearly identical to those for the quad-flat packages, except that the marking is rotated with respect to the ball A1 indicator. Figure 4 shows the top marking for Spartan-3E FPGAs in the CP132 and CPG132 packages.

Use the seven digits of the Lot Code to access additional information for a specific device using the Xilinx web-based [Genealogy Viewer](#).

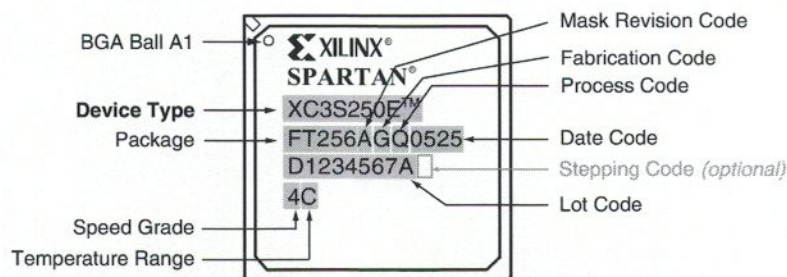
On the QFP and BGA packages, the optional numerical Stepping Code follows the Lot Code. If no Stepping Code appears, then the device is Stepping 0.

The “5C” and “4I” part combinations may be dual marked as “5C/4I”. All “5C” and “4I” part combinations use the Stepping 1 production silicon and have a ‘1’ Stepping Code mark.



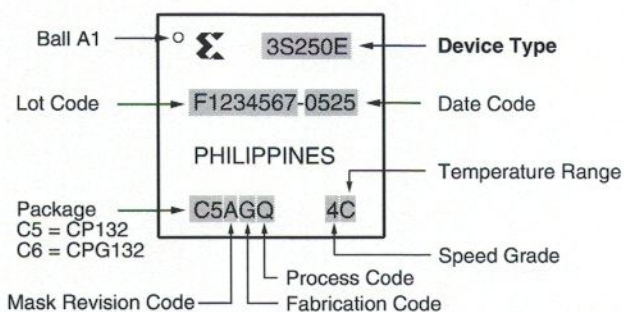
DS312-1_06_102905

Figure 2: Spartan-3E QFP Package Marking Example



DS312-1_02_090105

Figure 3: Spartan-3E BGA Package Marking Example



DS312-1_05_032105

Figure 4: Spartan-3E CP132 and CPG132 Package Marking Example

Features

- In-System Programmable PROMs for Configuration of Xilinx® FPGAs
- Low-Power Advanced CMOS NOR Flash Process
- Endurance of 20,000 Program/Erase Cycles
- Operation over Full Industrial Temperature Range (–40°C to +85°C)
- IEEE Standard 1149.1/1532 Boundary-Scan (JTAG) Support for Programming, Prototyping, and Testing
- JTAG Command Initiation of Standard FPGA Configuration
- Cascadable for Storing Longer or Multiple Bitstreams
- Dedicated Boundary-Scan (JTAG) I/O Power Supply (V_{CCJ})
- I/O Pins Compatible with Voltage Levels Ranging From 1.8V to 3.3V
- Design Support Using the Xilinx ISE® Alliance and Foundation™ Software Packages
- XCF01S/XCF02S/XCF04S
 - 3.3V Supply Voltage
 - Serial FPGA Configuration Interface
 - Available in Small-Footprint VO20 and VOG20 Packages
- XCF08P/XCF16P/XCF32P
 - 1.8V Supply Voltage
 - Serial or Parallel FPGA Configuration Interface
 - Available in Small-Footprint VOG48, FS48, and FSG48 Packages
 - Design Revision Technology Enables Storing and Accessing Multiple Design Revisions for Configuration
 - Built-In Data Decompressor Compatible with Xilinx Advanced Compression Technology

Description

Xilinx introduces the Platform Flash series of in-system programmable configuration PROMs. Available in 1 to 32 Mb densities, these PROMs provide an easy-to-use, cost-effective, and reprogrammable method for storing large Xilinx FPGA configuration bitstreams. The Platform Flash PROM series includes both the 3.3V XCFxxS PROM and the 1.8V XCFxxP PROM. The XCFxxS version includes 4 Mb, 2 Mb, and 1 Mb PROMs that support Master Serial and Slave Serial FPGA configuration modes (Figure 1, page 2). The XCFxxP version includes 32 Mb, 16 Mb, and

8 Mb PROMs that support Master Serial, Slave Serial, Master SelectMAP, and Slave SelectMAP FPGA configuration modes (Figure 2, page 2).

When driven from a stable, external clock, the PROMs can output data at rates up to 33 MHz. Refer to "AC Electrical Characteristics," page 16 for timing considerations.

A summary of the Platform Flash PROM family members and supported features is shown in Table 1.

Table 1: Platform Flash PROM Features

Device	Density (Mb)	V_{CCINT} (V)	V_{CCO} Range (V)	V_{CCJ} Range (V)	Packages	Program In-system via JTAG	Serial Config.	Parallel Config.	Design Revisioning	Compression
XCF01S	1	3.3	1.8 – 3.3	2.5 – 3.3	VO20/VOG20	✓	✓			
XCF02S	2	3.3	1.8 – 3.3	2.5 – 3.3	VO20/VOG20	✓	✓			
XCF04S	4	3.3	1.8 – 3.3	2.5 – 3.3	VO20/VOG20	✓	✓			
XCF08P	8	1.8	1.8 – 3.3	2.5 – 3.3	VO48/VOG48 FS48/FSG48	✓	✓	✓	✓(1)	✓
XCF16P	16	1.8	1.8 – 3.3	2.5 – 3.3	VO48/VOG48 FS48/FSG48	✓	✓	✓	✓	✓
XCF32P	32	1.8	1.8 – 3.3	2.5 – 3.3	VO48/VOG48 FS48/FSG48	✓	✓	✓	✓	✓

Notes:

1. XCF08P supports storage of a design revision only when cascaded with another XCFxxP PROM. See "Design Revisioning," page 8 for details.

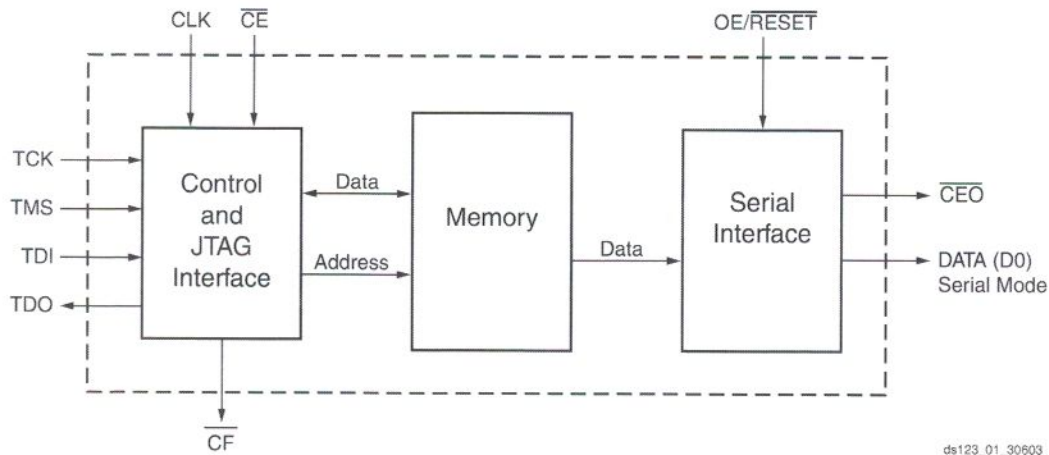


Figure 1: XCFxxS Platform Flash PROM Block Diagram

ds123_01_30603

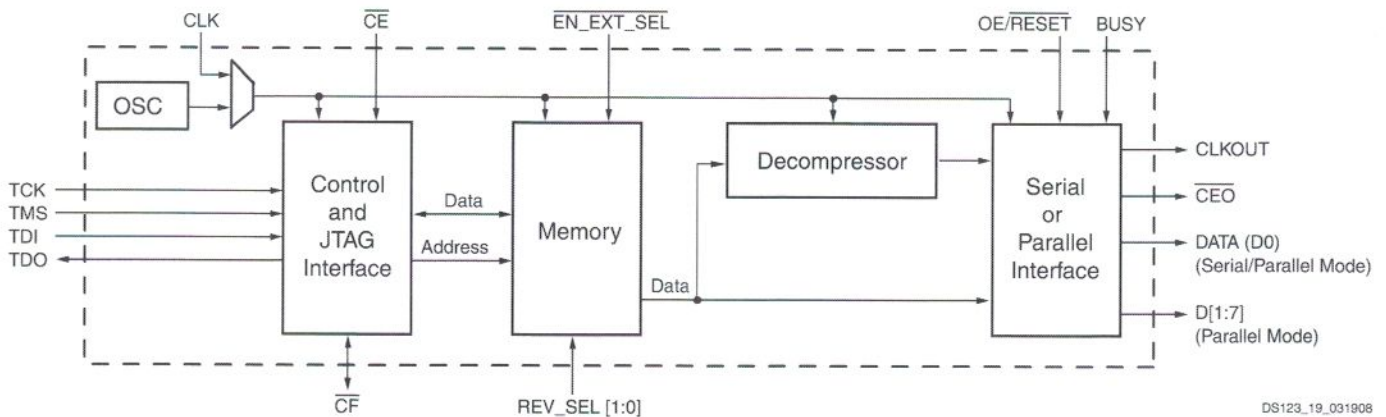


Figure 2: XCFxxP Platform Flash PROM Block Diagram

DS123_19_031908

When the FPGA is in Master Serial mode, it generates a configuration clock that drives the PROM. With \overline{CF} High, a short access time after \overline{CE} and \overline{OE} are enabled, data is available on the PROM DATA (D0) pin that is connected to the FPGA DIN pin. New data is available a short access time after each rising clock edge. The FPGA generates the appropriate number of clock pulses to complete the configuration.

When the FPGA is in Slave Serial mode, the PROM and the FPGA are both clocked by an external clock source, or optionally, for the XCFxxP PROM only, the PROM can be used to drive the FPGA's configuration clock.

The XCFxxP version of the Platform Flash PROM also supports Master SelectMAP and Slave SelectMAP (or Slave Parallel) FPGA configuration modes. When the FPGA is in Master SelectMAP mode, the FPGA generates a configuration clock that drives the PROM. When the FPGA is in Slave SelectMAP Mode, either an external oscillator generates the configuration clock that drives the PROM and the FPGA, or optionally, the XCFxxP PROM can be used to drive the FPGA's configuration clock. With \overline{BUSY} Low and \overline{CF} High, after \overline{CE} and \overline{OE} are enabled, data is available on the PROMs DATA (D0-D7) pins. New data is available a

short access time after each rising clock edge. The data is clocked into the FPGA on the following rising edge of the CCLK. A free-running oscillator can be used in the Slave Parallel/Slave SelectMAP mode.

The XCFxxP version of the Platform Flash PROM provides additional advanced features. A built-in data decompressor supports utilizing compressed PROM files, and design revisioning allows multiple design revisions to be stored on a single PROM or stored across several PROMs. For design revisioning, external pins or internal control bits are used to select the active design revision.

Multiple Platform Flash PROM devices can be cascaded to support the larger configuration files required when targeting larger FPGA devices or targeting multiple FPGAs daisy chained together. When utilizing the advanced features for the XCFxxP Platform Flash PROM, such as design revisioning, programming files which span cascaded PROM devices can only be created for cascaded chains containing only XCFxxP PROMs. If the advanced XCFxxP features are not enabled, then the cascaded chain can include both XCFxxP and XCFxxS PROMs.

See [UG161](#), *Platform Flash PROM User Guide*, for detailed guidelines on PROM-to-FPGA configuration hardware connections, for software usage, for a reference list of Xilinx FPGAs, and for the respective compatible Platform Flash PROMs. [Table 2](#) lists the Platform Flash PROMs and their capacities.

Table 2: Platform Flash PROM Capacity

Platform Flash PROM	Configuration Bits	Platform Flash PROM	Configuration Bits
XCF01S	1,048,576	XCF08P	8,388,608
XCF02S	2,097,152	XCF16P	16,777,216
XCF04S	4,194,304	XCF32P	33,554,432

Programming

The Platform Flash PROM is a reprogrammable NOR flash device (refer "[Quality and Reliability Characteristics](#)," [page 14](#) for the program/erase specifications). Reprogramming requires an erase followed by a program operation. A verify operation is recommended after the program operation to validate the correct transfer of data from the programmer source to the Platform Flash PROM.

Several programming solutions are available.

In-System Programming

In-System Programmable PROMs can be programmed individually, or two or more can be daisy-chained together and programmed in-system via the standard 4-pin JTAG protocol as shown in [Figure 3](#).

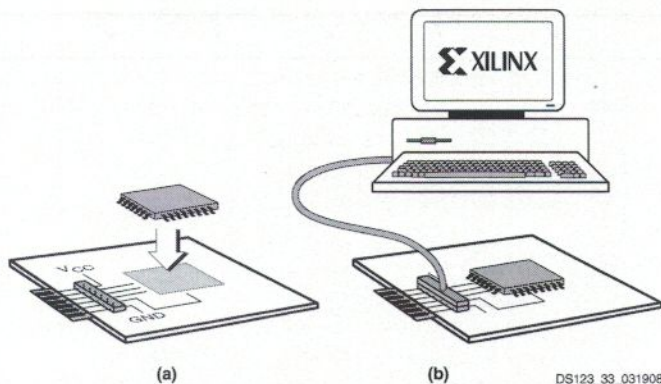


Figure 3: JTAG In-System Programming Operation

- (a) Solder Device to PCB
- (b) Program Using Download Cable

In-system programming offers quick and efficient design iterations and eliminates unnecessary package handling or socketing of devices. The programming data sequence is delivered to the device using either Xilinx iMPACT software and a Xilinx download cable, a third-party JTAG development system, a JTAG-compatible board tester, or a simple microprocessor interface that emulates the JTAG

instruction sequence. The iMPACT software also outputs serial vector format (SVF) files for use with any tools that accept SVF format, including automatic test equipment. During in-system programming, the \overline{CEO} output is driven High. All other outputs are held in a high-impedance state or held at clamp levels during in-system programming. All non-JTAG input pins are ignored during in-system programming, including CLK, CE, CF, OE/RESET, BUSY, EN_EXT_SEL, and REV_SEL[1:0]. In-system programming is fully supported across the recommended operating voltage and temperature ranges.

Embedded, in-system programming reference designs, such as [XAPP058](#), *Xilinx In-System Programming Using an Embedded Microcontroller*, are available on the Xilinx web page for [PROM Programming and Data Storage Application Notes](#). See [UG161](#), *Platform Flash PROM User Guide*, for an advanced update methodology that uses the Design Revisioning feature in the Platform Flash XCFxxP PROMs.

OE/RESET

The 1/2/4 Mb XCFxxS Platform Flash PROMs in-system programming algorithm results in issuance of an internal device reset that causes OE/RESET to pulse Low.

External Programming

In traditional manufacturing environments, third-party device programmers can program Platform Flash PROMs with an initial memory image before the PROMs are assembled onto boards. Contact a preferred third-party programmer vendor for Platform Flash PROM support information. A sample list of third-party programmer vendors with Platform Flash PROM support is available on the Xilinx web page for [Third-Party Programmer Device Support](#). See [UG161](#), *Platform Flash PROM User Guide*, for the PROM data file format required for programmers.

Pre-programmed PROMs can be assembled onto boards using the typical soldering process guidelines in [UG112](#), *Device Package User Guide*. A pre-programmed PROM's memory image can be updated after board assembly using an in-system programming solution.

Reliability and Endurance

Xilinx in-system programmable products provide a guaranteed endurance level of 20,000 in-system program-erase cycles and a minimum data retention of 20 years. Each device meets all functional, performance, and data retention specifications within this endurance limit.

See [UG116](#), *Xilinx Device Reliability Report*, for device quality, reliability, and process node information.

Design Security

The Xilinx in-system programmable Platform Flash PROM devices incorporate advanced data security features to fully protect the FPGA programming data against unauthorized reading via JTAG. The XCFxxP PROMs can also be programmed to prevent inadvertent writing via JTAG.

Table 3 and Table 4 show the security settings available for the XCFxxS PROM and XCFxxP PROM, respectively.

Read Protection

The read protect security bit can be set by the user to prevent the internal programming pattern from being read or copied via JTAG. Read protection does not prevent write operations. For the XCFxxS PROM, the read protect security bit is set for the entire device, and resetting the read protect security bit requires erasing the entire device. For the XCFxxP PROM the read protect security bit can be set for individual design revisions, and resetting the read protect bit requires erasing the particular design revision.

Write Protection

The XCFxxP PROM device also allows the user to write protect (or lock) a particular design revision or PROM option settings. Write protection helps to prevent an inadvertent JTAG instruction from modifying an area by write protecting the area and by locking the erase instruction. The write-protection setting can be cleared by erasing the protected area. However, an XSC_UNLOCK instruction must first be issued to the XCFxxP PROM to unlock the ISC_ERASE instruction. Refer to the XCFxxP PROM BSDL file for the XSC_UNLOCK and ISC_ERASE instructions.

Caution! The iMPACT software always issues a XSC_UNLOCK when performing an Erase operation on an XCFxxP PROM and, thus, always unlocks the write protection.

Table 3: XCFxxS Device Data Security Options

Read Protect	Read/Verify Inhibited	Program Inhibited	Erase Inhibited
Reset (default)			
Set	✓		

Table 4: XCFxxP Design Revision Data Security Options

Read Protect	Write Protect	Read/Verify Inhibited	Program Inhibited	Erase Inhibited
Reset (default)	Reset (default)			
Reset (default)	Set		✓	✓
Set	Reset (default)	✓		
Set	Set	✓	✓	✓


```

1  # use the reset switch on the Core3S500E
2  NET "reset_0" LOC = "P101";
3  NET "clk" LOC = "P184";
4
5  # LEDs on the Core3S500E
6  NET "led01" LOC = "p89";
7  NET "led02" LOC = "p83";
8  NET "led03" LOC = "p78";
9  NET "led04" LOC = "p68";
10
11 # Use 16I/Os_2
12 NET "counter_tp[0]" LOC = "p119"; #16I/Os_2_1
13 NET "counter_tp[1]" LOC = "p116"; #16I/Os_2_2
14 NET "counter_tp[2]" LOC = "p115"; #16I/Os_2_3
15 NET "counter_tp[3]" LOC = "p113"; #16I/Os_2_4
16 NET "counter_tp[4]" LOC = "p112"; #16I/Os_2_5
17 NET "counter_tp[5]" LOC = "p109"; #16I/Os_2_6
18 NET "counter_tp[6]" LOC = "p108"; #16I/Os_2_7
19 NET "counter_tp[7]" LOC = "p107"; #16I/Os_2_8
20 NET "counter_tp[8]" LOC = "p106"; #16I/Os_2_9
21 NET "counter_tp[9]" LOC = "p102"; #16I/Os_2_10
22 NET "counter_tp[10]" LOC = "p100"; #16I/Os_2_11
23 NET "counter_tp[11]" LOC = "p99"; #16I/Os_2_12
24
25 # Use 8I/Os_2
26 NET "sub_clk_tp" LOC = "p132"; #8I/Os_2_1 - 1000 Hz
27 NET "clk_10_tp" LOC = "p129"; #8I/Os_2_2 - 100 Hz
28 NET "clk_100_tp" LOC = "p128"; #8I/Os_2_3 100 Hz 10 Hz
29 NET "clk_1000_tp" LOC = "p127"; #8I/Os_2_4 - 1 Hz
30 NET "clk_0_tp" LOC = "p126"; #8I/Os_2_5

```

Counter02.vct


```

1  -- VHDL library Declarations
2  LIBRARY IEEE;
3  USE IEEE.STD_LOGIC_1164.ALL;
4  USE IEEE.STD_LOGIC_UNSIGNED.ALL;
5  USE IEEE.STD_LOGIC_ARITH.ALL;
6
7  -- The Entity Declarations
8  ENTITY counter02 IS
9      PORT
10         (
11             -- rst & clk
12             reset_0:    IN STD_LOGIC;
13             clk:        IN STD_LOGIC;    -- 50 MHz
14
15             -- test points for the logic analyzer
16             counter_tp: OUT STD_LOGIC_VECTOR(11 DOWNTO 0); -- for logic analyzer
17             sub_clk_tp:  OUT STD_LOGIC;    -- 1000 Hz clock for counter
18             clk_10_tp:   OUT STD_LOGIC;    -- Every 10 Count Blip
19             clk_100_tp:  OUT STD_LOGIC;    -- Every 100 Count Blip
20             clk_1000_tp: OUT STD_LOGIC;    -- Every 1000 Count Blip
21             clk_0_tp:    OUT STD_LOGIC;    -- Counter 0 Blip
22             led01:       OUT STD_LOGIC;    -- LED01 - Flash every 10 counts
23             led02:       OUT STD_LOGIC;    -- LED02 - Flash every 100 counts
24             led03:       OUT STD_LOGIC;    -- LED03 - Flash every 1000 counts
25             led04:       OUT STD_LOGIC);   -- LED04 - Flash when the counter is 0
26 END counter02;
27
28 -- The Architecture of Entity Declarations
29 ARCHITECTURE Behavioral OF counter02 IS
30     SIGNAL counter: STD_LOGIC_VECTOR(11 downto 0) := "000000000000"; -- main counter
31     SIGNAL sub_clk: STD_LOGIC := '0'; -- 1000 Hz - Sub Clock
32     SIGNAL counter1: INTEGER RANGE 0 TO 49999 := 0;
33     SIGNAL counter_10: INTEGER RANGE 0 TO 9 := 0;
34     SIGNAL counter_100: INTEGER RANGE 0 TO 99 := 0;
35     SIGNAL counter_1000: INTEGER RANGE 0 TO 999 := 0;
36     SIGNAL clk_10: STD_LOGIC := '0';
37     SIGNAL clk_100: STD_LOGIC := '0';
38     SIGNAL clk_1000: STD_LOGIC := '0';
39 BEGIN
40
41     -- Main Process for counter
42     PROCESS(reset_0, clk)
43     BEGIN
44         IF reset_0 = '0' THEN
45             counter <= "000000000000";
46         ELSIF RISING_EDGE(clk) THEN
47
48             IF clk_10 = '1' THEN
49                 led01 <= '0';
50             ELSE
51                 led01 <= '1';
52             END IF;
53
54             IF clk_100 = '1' THEN
55                 led02 <= '0';
56             ELSE
57                 led02 <= '1';
58             END IF;
59
60             IF clk_1000 = '1' THEN
61                 led03 <= '0';
62             ELSE
63                 led03 <= '1';
64             END IF;
65
66             IF counter = "000000000000" THEN
67                 led04 <= '0';

```



```

68         clk_0_tp <= '1';
69     ELSE
70         led04 <= '1';
71         clk_0_tp <= '0';
72     END IF;
73
74     IF sub_clk = '1' THEN
75         counter <= counter + 1; -- main counter
76     END IF;
77 END IF;
78 END PROCESS;
79
80 counter_tp <= counter;
81
82 -- 1000 Hz sub_clk
83 PROCESS(reset_0,clk)
84 BEGIN
85     IF reset_0 = '0' THEN
86         counter1 <= 49999;
87         sub_clk <= '0';
88     ELSIF RISING_EDGE(clk) THEN
89         IF counter1 /= 0 THEN
90             counter1 <= counter1 - 1;
91         ELSE
92             counter1 <= 49999;
93         END IF;
94         IF counter1 = 0 THEN
95             sub_clk <= '1';
96         ELSE
97             sub_clk <= '0';
98         END IF;
99     END IF;
100 END PROCESS;
101
102 sub_clk_tp <= sub_clk;
103
104 -- Every 10 Counter
105 PROCESS(reset_0,clk)
106 BEGIN
107     IF reset_0 = '0' THEN
108         counter_10 <= 9;
109         clk_10 <= '0';
110     ELSIF RISING_EDGE(clk) THEN
111         IF sub_clk = '1' THEN
112             IF counter_10 /= 0 THEN
113                 counter_10 <= counter_10 - 1;
114             ELSE
115                 counter_10 <= 9;
116             END IF;
117             IF counter_10 = 0 THEN
118                 clk_10 <= '1';
119             ELSE
120                 clk_10 <= '0';
121             END IF;
122         END IF;
123     END IF;
124 END PROCESS;
125
126 clk_10_tp <= clk_10;
127
128 -- Every 100 Counter
129 PROCESS(reset_0,clk)
130 BEGIN
131     IF reset_0 = '0' THEN
132         counter_100 <= 99;
133         clk_100 <= '0';
134     ELSIF RISING_EDGE(clk) THEN

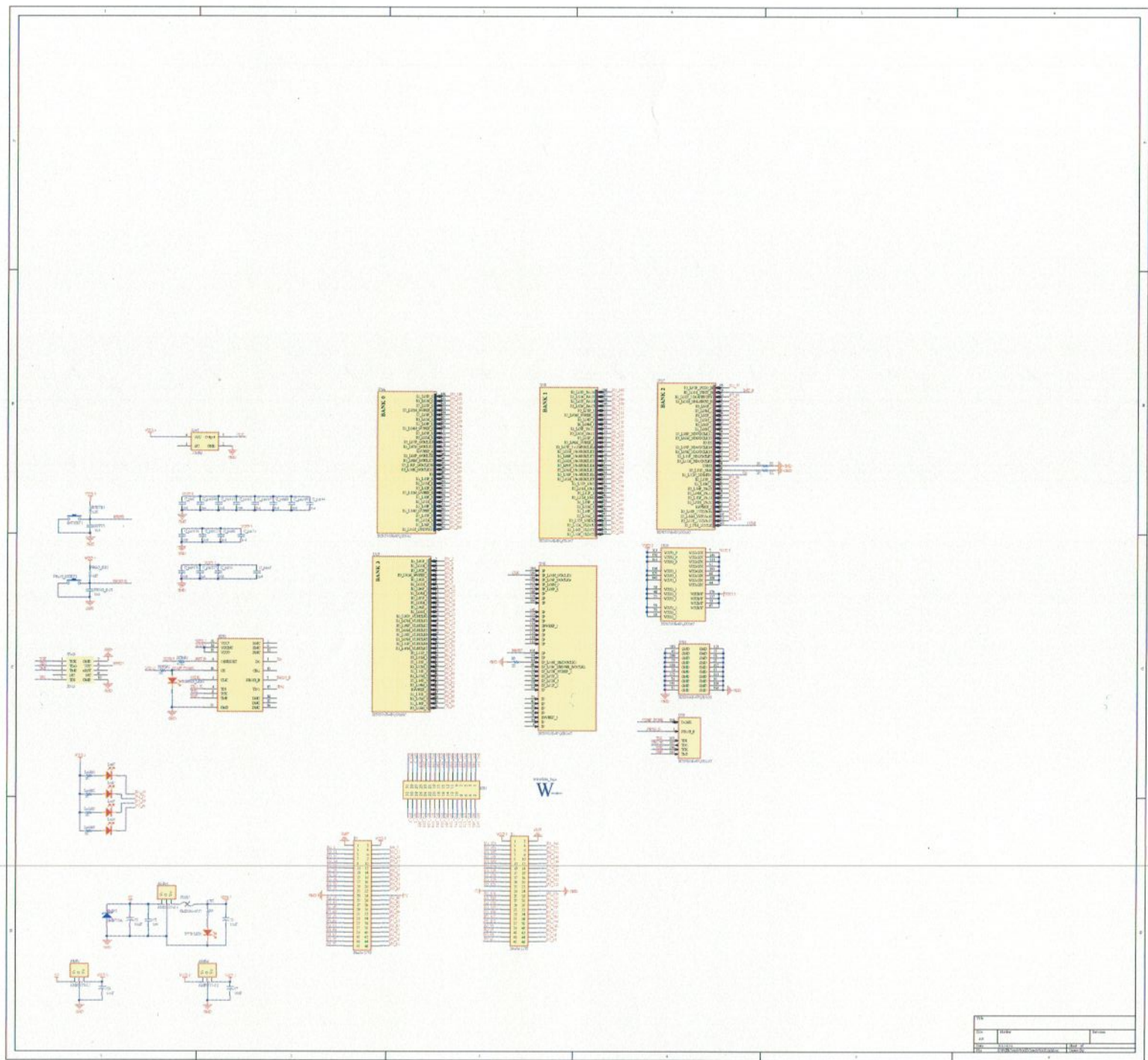
```



```

135     IF sub_clk = '1' THEN
136         IF counter_100 /= 0 THEN
137             counter_100 <= counter_100 - 1;
138         ELSE
139             counter_100 <= 99;
140         END IF;
141         IF counter_100 = 0 THEN
142             clk_100 <= '1';
143         ELSE
144             clk_100 <= '0';
145         END IF;
146     END IF;
147 END IF;
148 END PROCESS;
149
150 clk_100_tp <= clk_100;
151
152 -- Every 1000 Counter
153 PROCESS(reset_0,clk)
154 BEGIN
155     IF reset_0 = '0' THEN
156         counter_1000 <= 999;
157         clk_1000 <= '0';
158     ELSIF RISING_EDGE(clk) THEN
159         IF sub_clk = '1' THEN
160             IF counter_1000 /= 0 THEN
161                 counter_1000 <= counter_1000 - 1;
162             ELSE
163                 counter_1000 <= 999;
164             END IF;
165             IF counter_1000 = 0 THEN
166                 clk_1000 <= '1';
167             ELSE
168                 clk_1000 <= '0';
169             END IF;
170         END IF;
171     END IF;
172 END PROCESS;
173
174 clk_1000_tp <= clk_1000;
175
176 END Behavioral;
177
178
179
180

```

Rev	001	Date	10/10/2010
By	W	Drawn	W
App	W	Checked	W
Rev	001	Date	10/10/2010


```

1 #####Core3S500E#####
2 #####
3 NET "clk" LOC = "P184" ;
4 NET "RST" LOC = "P101" ;
5
6 NET "led[0]" LOC = "P68" ;
7 NET "led[1]" LOC = "P78" ;
8 NET "led[2]" LOC = "P83" ;
9 NET "led[3]" LOC = "P89";
10
11 ##### Mother Board
12 #####
13 NET "Buzzer" LOC = "p123" ;
14 NET "DS18B20" LOC = "p126" ;
15
16 #JOYSTICK
17 NET "down" LOC = "p116" ;
18 NET "up" LOC = "p122" ;
19 NET "left" LOC = "p119" ;
20 NET "right" LOC = "p120" ;
21 NET "press" LOC = "p115" ;
22
23 #8I/Os_1
24 NET "8I/Os_1_1" LOC = "p75" ;
25 NET "8I/Os_1_2" LOC = "p76" ;
26 NET "8I/Os_1_3" LOC = "p69" ;
27 NET "8I/Os_1_4" LOC = "p74" ;
28 NET "8I/Os_1_5" LOC = "p64" ;
29 NET "8I/Os_1_6" LOC = "p65" ;
30 NET "8I/Os_1_7" LOC = "p62" ;
31 NET "8I/Os_1_8" LOC = "p63" ;
32
33 #8I/Os_2
34 NET "8I/Os_2_1" LOC = "p132" ;
35 NET "8I/Os_2_2" LOC = "p129" ;
36 NET "8I/Os_2_3" LOC = "p128" ;
37 NET "8I/Os_2_4" LOC = "p127" ;
38 NET "8I/Os_2_5" LOC = "p126" ;
39 NET "8I/Os_2_6" LOC = "p123" ;
40 NET "8I/Os_2_7" LOC = "p122" ;
41 NET "8I/Os_2_8" LOC = "p120" ;
42
43 #16I/Os_2
44 NET "16I/Os_2_1" LOC = "p119" ;
45 NET "16I/Os_2_2" LOC = "p116" ;
46 NET "16I/Os_2_3" LOC = "p115" ;
47 NET "16I/Os_2_4" LOC = "p113" ;
48 NET "16I/Os_2_5" LOC = "p112" ;
49 NET "16I/Os_2_6" LOC = "p109" ;
50 NET "16I/Os_2_7" LOC = "p108" ;
51 NET "16I/Os_2_8" LOC = "p107" ;
52 NET "16I/Os_2_9" LOC = "p106" ;
53 NET "16I/Os_2_10" LOC = "p102" ;
54 NET "16I/Os_2_11" LOC = "p100" ;
55 NET "16I/Os_2_12" LOC = "p99" ;
56 NET "16I/Os_2_13" LOC = "p98" ;
57 NET "16I/Os_2_14" LOC = "p97" ;
58 NET "16I/Os_2_15" LOC = "p96" ;
59 NET "16I/Os_2_16" LOC = "p94" ;
60
61 #16I/Os_1
62 NET "16I/Os_1_1" LOC = "p160" ;
63 NET "16I/Os_1_2" LOC = "p153" ;
64 NET "16I/Os_1_3" LOC = "p152" ;
65 NET "16I/Os_1_4" LOC = "p151" ;

```

3S500E.ucf


```

66 NET "16I/Os_1_5" LOC = "p150" ;
67 NET "16I/Os_1_6" LOC = "p147" ;
68 NET "16I/Os_1_7" LOC = "p146" ;
69 NET "16I/Os_1_8" LOC = "p145" ;
70 NET "16I/Os_1_9" LOC = "p144" ;
71 NET "16I/Os_1_10" LOC = "p140" ;
72 NET "16I/Os_1_11" LOC = "p139" ;
73 NET "16I/Os_1_12" LOC = "p138" ;
74 NET "16I/Os_1_13" LOC = "p137" ;
75 NET "16I/Os_1_14" LOC = "p135" ;
76 NET "16I/Os_1_15" LOC = "p134" ;
77 NET "16I/Os_1_16" LOC = "p133" ;
78
79 #32I/Os_1
80 NET "32I/Os_1_1" LOC = "p61";
81 NET "32I/Os_1_2" LOC = "p60";
82 NET "32I/Os_1_3" LOC = "p55";
83 NET "32I/Os_1_4" LOC = "p50";
84 NET "32I/Os_1_5" LOC = "p49";
85 NET "32I/Os_1_6" LOC = "p48";
86 NET "32I/Os_1_7" LOC = "p47";
87 NET "32I/Os_1_8" LOC = "p45";
88 NET "32I/Os_1_9" LOC = "p42";
89 NET "32I/Os_1_10" LOC = "p41";
90 NET "32I/Os_1_11" LOC = "p40";
91 NET "32I/Os_1_12" LOC = "p39";
92 NET "32I/Os_1_13" LOC = "p36";
93 NET "32I/Os_1_14" LOC = "p35";
94 NET "32I/Os_1_15" LOC = "p34";
95 NET "32I/Os_1_16" LOC = "p33";
96 NET "32I/Os_1_17" LOC = "p31";
97 NET "32I/Os_1_18" LOC = "p30";
98 NET "32I/Os_1_19" LOC = "p29";
99 NET "32I/Os_1_20" LOC = "p28";
100 NET "32I/Os_1_21" LOC = "p25";
101 NET "32I/Os_1_22" LOC = "p24";
102 NET "32I/Os_1_23" LOC = "p23";
103 NET "32I/Os_1_24" LOC = "p22";
104 NET "32I/Os_1_25" LOC = "p19";
105 NET "32I/Os_1_26" LOC = "p18";
106 NET "32I/Os_1_27" LOC = "p16";
107 NET "32I/Os_1_28" LOC = "p15";
108 NET "32I/Os_1_29" LOC = "p12";
109 NET "32I/Os_1_30" LOC = "p11";
110 NET "32I/Os_1_31" LOC = "p9";
111 NET "32I/Os_1_32" LOC = "p8";
112
113
114
115 #SDRAM_L
116 NET "SDRAM_L_1" LOC = "GND" ;
117 NET "SDRAM_L_2" LOC = "3.3V" ;
118 NET "SDRAM_L_3" LOC = "p9";
119 NET "SDRAM_L_4" LOC = "p8";
120 NET "SDRAM_L_5" LOC = "p12";
121 NET "SDRAM_L_6" LOC = "p11";
122 NET "SDRAM_L_7" LOC = "p16";
123 NET "SDRAM_L_8" LOC = "p15";
124 NET "SDRAM_L_9" LOC = "p19";
125 NET "SDRAM_L_10" LOC = "p18";
126 NET "SDRAM_L_11" LOC = "p23";
127 NET "SDRAM_L_12" LOC = "p22" ;
128 NET "SDRAM_L_13" LOC = "p25";
129 NET "SDRAM_L_14" LOC = "p24";
130 NET "SDRAM_L_15" LOC = "p29";
131 NET "SDRAM_L_16" LOC = "p28";
132 NET "SDRAM_L_17" LOC = "p31";

```



```

133 NET "SDRAM_L_18" LOC = "p30";
134 NET "SDRAM_L_19" LOC = "p34";
135 NET "SDRAM_L_20" LOC = "p33";
136 NET "SDRAM_L_21" LOC = "p36";
137 NET "SDRAM_L_22" LOC = "p35";
138 #SDRAM_R
139 NET "SDRAM_R_1" LOC = "3.3V";
140 NET "SDRAM_R_2" LOC = "GND";
141 NET "SDRAM_R_3" LOC = "p153";
142 NET "SDRAM_R_4" LOC = "p160";
143 NET "SDRAM_R_5" LOC = "p151";
144 NET "SDRAM_R_6" LOC = "p152";
145 NET "SDRAM_R_7" LOC = "p147";
146 NET "SDRAM_R_8" LOC = "p150";
147 NET "SDRAM_R_9" LOC = "p145";
148 NET "SDRAM_R_10" LOC = "P146";
149 NET "SDRAM_R_11" LOC = "p140";
150 NET "SDRAM_R_12" LOC = "p144";
151 NET "SDRAM_R_13" LOC = "p138";
152 NET "SDRAM_R_14" LOC = "p139";
153 NET "SDRAM_R_15" LOC = "p135";
154 NET "SDRAM_R_16" LOC = "p137";
155 NET "SDRAM_R_17" LOC = "p133";
156 NET "SDRAM_R_18" LOC = "p134";
157 NET "SDRAM_R_19" LOC = "p129";
158 NET "SDRAM_R_20" LOC = "p132";
159 NET "SDRAM_R_21" LOC = "p127";
160 NET "SDRAM_R_22" LOC = "p128";

```

```

161
162
163 #LCD1602
164 NET "RS" LOC = "p36";
165 NET "R/W" LOC = "p40";
166 NET "EN" LOC = "p39";
167 NET "D0" LOC = "p42";
168 NET "D1" LOC = "p41";
169 NET "D2" LOC = "p47";
170 NET "D3" LOC = "p45";
171 NET "D4" LOC = "p49";
172 NET "D5" LOC = "p48";
173 NET "D6" LOC = "p55";
174 NET "D7" LOC = "p50";
175 NET "A" LOC = "p61";
176 NET "K" LOC = "p60";

```

```

177
178 #LCD12864
179 NET "RS" LOC = "p36";
180 NET "R/W" LOC = "p40";
181 NET "EN" LOC = "p39";
182 NET "D0" LOC = "p42";
183 NET "D1" LOC = "p41";
184 NET "D2" LOC = "p47";
185 NET "D3" LOC = "p45";
186 NET "D4" LOC = "p49";
187 NET "D5" LOC = "p48";
188 NET "D6" LOC = "p55";
189 NET "D7" LOC = "p50";
190 NET "PSB" LOC = "p61";
191
192 NET "RST" LOC = "p63";
193
194 NET "A" LOC = "p65";
195 NET "K" LOC = "p64";

```

```

196
197
198
199

```


PQ208: 208-pin Plastic Quad Flat Package

The 208-pin plastic quad flat package, PQ208, supports two different Spartan-3E FPGAs, including the XC3S250E and the XC3S500E.

Table 140 lists all the PQ208 package pins. They are sorted by bank number and then by pin name. Pairs of pins that form a differential I/O pair appear together in the table. The table also shows the pin number for each pin and the pin type, as defined earlier.

An electronic version of this package pinout table and footprint diagram is available for download from the Xilinx website at http://www.xilinx.com/bvdocs/publications/s3e_pin.zip.

Pinout Table

Table 140: PQ208 Package Pinout

Bank	XC3S250E XC3S500E Pin Name	PQ208 Pin	Type
0	IO	P187	I/O
0	IO/VREF_0	P179	VREF
0	IO_L01N_0	P161	I/O
0	IO_L01P_0	P160	I/O
0	IO_L02N_0/VREF_0	P163	VREF
0	IO_L02P_0	P162	I/O
0	IO_L03N_0	P165	I/O
0	IO_L03P_0	P164	I/O
0	IO_L04N_0/VREF_0	P168	VREF
0	IO_L04P_0	P167	I/O
0	IO_L05N_0	P172	I/O
0	IO_L05P_0	P171	I/O
0	IO_L07N_0/GCLK5	P178	GCLK
0	IO_L07P_0/GCLK4	P177	GCLK
0	IO_L08N_0/GCLK7	P181	GCLK
0	IO_L08P_0/GCLK6	P180	GCLK
0	IO_L10N_0/GCLK11	P186	GCLK
0	IO_L10P_0/GCLK10	P185	GCLK
0	IO_L11N_0	P190	I/O
0	IO_L11P_0	P189	I/O
0	IO_L12N_0/VREF_0	P193	VREF
0	IO_L12P_0	P192	I/O
0	IO_L13N_0	P197	I/O
0	IO_L13P_0	P196	I/O
0	IO_L14N_0/VREF_0	P200	VREF
0	IO_L14P_0	P199	I/O
0	IO_L15N_0	P203	I/O
0	IO_L15P_0	P202	I/O

Table 140: PQ208 Package Pinout (Continued)

Bank	XC3S250E XC3S500E Pin Name	PQ208 Pin	Type
0	IO_L16N_0/HSWAP	P206	DUAL
0	IO_L16P_0	P205	I/O
0	IP	P159	INPUT
0	IP	P169	INPUT
0	IP	P194	INPUT
0	IP	P204	INPUT
0	IP_L06N_0	P175	INPUT
0	IP_L06P_0	P174	INPUT
0	IP_L09N_0/GCLK9	P184	GCLK
0	IP_L09P_0/GCLK8	P183	GCLK
0	VCCO_0	P176	VCCO
0	VCCO_0	P191	VCCO
0	VCCO_0	P201	VCCO
1	IO_L01N_1/A15	P107	DUAL
1	IO_L01P_1/A16	P106	DUAL
1	IO_L02N_1/A13	P109	DUAL
1	IO_L02P_1/A14	P108	DUAL
1	IO_L03N_1/VREF_1	P113	VREF
1	IO_L03P_1	P112	I/O
1	IO_L04N_1	P116	I/O
1	IO_L04P_1	P115	I/O
1	IO_L05N_1/A11	P120	DUAL
1	IO_L05P_1/A12	P119	DUAL
1	IO_L06N_1/VREF_1	P123	VREF
1	IO_L06P_1	P122	I/O
1	IO_L07N_1/A9/RHCLK1	P127	RHCLK/DUAL
1	IO_L07P_1/A10/RHCLK0	P126	RHCLK/DUAL
1	IO_L08N_1/A7/RHCLK3	P129	RHCLK/DUAL
1	IO_L08P_1/A8/RHCLK2	P128	RHCLK/DUAL
1	IO_L09N_1/A5/RHCLK5	P133	RHCLK/DUAL
1	IO_L09P_1/A6/RHCLK4	P132	RHCLK/DUAL
1	IO_L10N_1/A3/RHCLK7	P135	RHCLK/DUAL
1	IO_L10P_1/A4/RHCLK6	P134	RHCLK/DUAL
1	IO_L11N_1/A1	P138	DUAL
1	IO_L11P_1/A2	P137	DUAL
1	IO_L12N_1/A0	P140	DUAL
1	IO_L12P_1	P139	I/O
1	IO_L13N_1	P145	I/O
1	IO_L13P_1	P144	I/O

Table 140: PQ208 Package Pinout (Continued)

Bank	XC3S250E XC3S500E Pin Name	PQ208 Pin	Type
1	IO_L14N_1	P147	I/O
1	IO_L14P_1	P146	I/O
1	IO_L15N_1/LDC0	P151	DUAL
1	IO_L15P_1/HDC	P150	DUAL
1	IO_L16N_1/LDC2	P153	DUAL
1	IO_L16P_1/LDC1	P152	DUAL
1	IP	P110	INPUT
1	IP	P118	INPUT
1	IP	P124	INPUT
1	IP	P130	INPUT
1	IP	P142	INPUT
1	IP	P148	INPUT
1	IP	P154	INPUT
1	IP/VREF_1	P136	VREF
1	VCCO_1	P114	VCCO
1	VCCO_1	P125	VCCO
1	VCCO_1	P143	VCCO
2	IO/D5	P76	DUAL
2	IO/M1	P84	DUAL
2	IO/VREF_2	P98	VREF
2	IO_L01N_2/INIT_B	P56	DUAL
2	IO_L01P_2/CSO_B	P55	DUAL
2	IO_L03N_2/MOSI/CSI_B	P61	DUAL
2	IO_L03P_2/DOOUT/BUSY	P60	DUAL
2	IO_L04N_2	P63	I/O
2	IO_L04P_2	P62	I/O
2	IO_L05N_2	P65	I/O
2	IO_L05P_2	P64	I/O
2	IO_L06N_2	P69	I/O
2	IO_L06P_2	P68	I/O
2	IO_L08N_2/D6/GCLK13	P75	DUAL/GCLK
2	IO_L08P_2/D7/GCLK12	P74	DUAL/GCLK
2	IO_L09N_2/D3/GCLK15	P78	DUAL/GCLK
2	IO_L09P_2/D4/GCLK14	P77	DUAL/GCLK
2	IO_L11N_2/D1/GCLK3	P83	DUAL/GCLK
2	IO_L11P_2/D2/GCLK2	P82	DUAL/GCLK
2	IO_L12N_2/DIN/D0	P87	DUAL
2	IO_L12P_2/M0	P86	DUAL
2	IO_L13N_2	P90	I/O
2	IO_L13P_2	P89	I/O

Table 140: PQ208 Package Pinout (Continued)

Bank	XC3S250E XC3S500E Pin Name	PQ208 Pin	Type
2	IO_L14N_2/A22	P94	DUAL
2	IO_L14P_2/A23	P93	DUAL
2	IO_L15N_2/A20	P97	DUAL
2	IO_L15P_2/A21	P96	DUAL
2	IO_L16N_2/VS1/A18	P100	DUAL
2	IO_L16P_2/VS2/A19	P99	DUAL
2	IO_L17N_2/CCLK	P103	DUAL
2	IO_L17P_2/VS0/A17	P102	DUAL
2	IP	P54	INPUT
2	IP	P91	INPUT
2	IP	P101	INPUT
2	IP_L02N_2	P58	INPUT
2	IP_L02P_2	P57	INPUT
2	IP_L07N_2/VREF_2	P72	VREF
2	IP_L07P_2	P71	INPUT
2	IP_L10N_2/M2/GCLK1	P81	DUAL/GCLK
2	IP_L10P_2/RDWR_B/ GCLK0	P80	DUAL/GCLK
2	VCCO_2	P59	VCCO
2	VCCO_2	P73	VCCO
2	VCCO_2	P88	VCCO
3	IO/VREF_3	P45	VREF
3	IO_L01N_3	P3	I/O
3	IO_L01P_3	P2	I/O
3	IO_L02N_3/VREF_3	P5	VREF
3	IO_L02P_3	P4	I/O
3	IO_L03N_3	P9	I/O
3	IO_L03P_3	P8	I/O
3	IO_L04N_3	P12	I/O
3	IO_L04P_3	P11	I/O
3	IO_L05N_3	P16	I/O
3	IO_L05P_3	P15	I/O
3	IO_L06N_3	P19	I/O
3	IO_L06P_3	P18	I/O
3	IO_L07N_3/LHCLK1	P23	LHCLK
3	IO_L07P_3/LHCLK0	P22	LHCLK
3	IO_L08N_3/LHCLK3	P25	LHCLK
3	IO_L08P_3/LHCLK2	P24	LHCLK
3	IO_L09N_3/LHCLK5	P29	LHCLK
3	IO_L09P_3/LHCLK4	P28	LHCLK
3	IO_L10N_3/LHCLK7	P31	LHCLK

Table 140: PQ208 Package Pinout (Continued)

Bank	XC3S250E XC3S500E Pin Name	PQ208 Pin	Type
3	IO_L10P_3/LHCLK6	P30	LHCLK
3	IO_L11N_3	P34	I/O
3	IO_L11P_3	P33	I/O
3	IO_L12N_3	P36	I/O
3	IO_L12P_3	P35	I/O
3	IO_L13N_3	P40	I/O
3	IO_L13P_3	P39	I/O
3	IO_L14N_3	P42	I/O
3	IO_L14P_3	P41	I/O
3	IO_L15N_3	P48	I/O
3	IO_L15P_3	P47	I/O
3	IO_L16N_3	P50	I/O
3	IO_L16P_3	P49	I/O
3	IP	P6	INPUT
3	IP	P14	INPUT
3	IP	P26	INPUT
3	IP	P32	INPUT
3	IP	P43	INPUT
3	IP	P51	INPUT
3	IP/VREF_3	P20	VREF
3	VCCO_3	P21	VCCO
3	VCCO_3	P38	VCCO
3	VCCO_3	P46	VCCO
GND	GND	P10	GND
GND	GND	P17	GND
GND	GND	P27	GND
GND	GND	P37	GND
GND	GND	P52	GND
GND	GND	P53	GND
GND	GND	P70	GND
GND	GND	P79	GND
GND	GND	P85	GND
GND	GND	P95	GND
GND	GND	P105	GND
GND	GND	P121	GND
GND	GND	P131	GND
GND	GND	P141	GND
GND	GND	P156	GND
GND	GND	P173	GND
GND	GND	P182	GND

Table 140: PQ208 Package Pinout (Continued)

Bank	XC3S250E XC3S500E Pin Name	PQ208 Pin	Type
GND	GND	P188	GND
GND	GND	P198	GND
GND	GND	P208	GND
VCCAUX	DONE	P104	CONFIG
VCCAUX	PROG_B	P1	CONFIG
VCCAUX	TCK	P158	JTAG
VCCAUX	TDI	P207	JTAG
VCCAUX	TDO	P157	JTAG
VCCAUX	TMS	P155	JTAG
VCCAUX	VCCAUX	P7	VCCAUX
VCCAUX	VCCAUX	P44	VCCAUX
VCCAUX	VCCAUX	P66	VCCAUX
VCCAUX	VCCAUX	P92	VCCAUX
VCCAUX	VCCAUX	P111	VCCAUX
VCCAUX	VCCAUX	P149	VCCAUX
VCCAUX	VCCAUX	P166	VCCAUX
VCCAUX	VCCAUX	P195	VCCAUX
VCCINT	VCCINT	P13	VCCINT
VCCINT	VCCINT	P67	VCCINT
VCCINT	VCCINT	P117	VCCINT
VCCINT	VCCINT	P170	VCCINT

User I/Os by Bank

Table 141 indicates how the 158 available user-I/O pins are distributed between the four I/O banks on the PQ208 package.

Footprint Migration Differences

The XC3S250E and XC3S500E FPGAs have identical footprints in the PQ208 package. Designs can migrate between the XC3S250E and XC3S500E without further consideration.

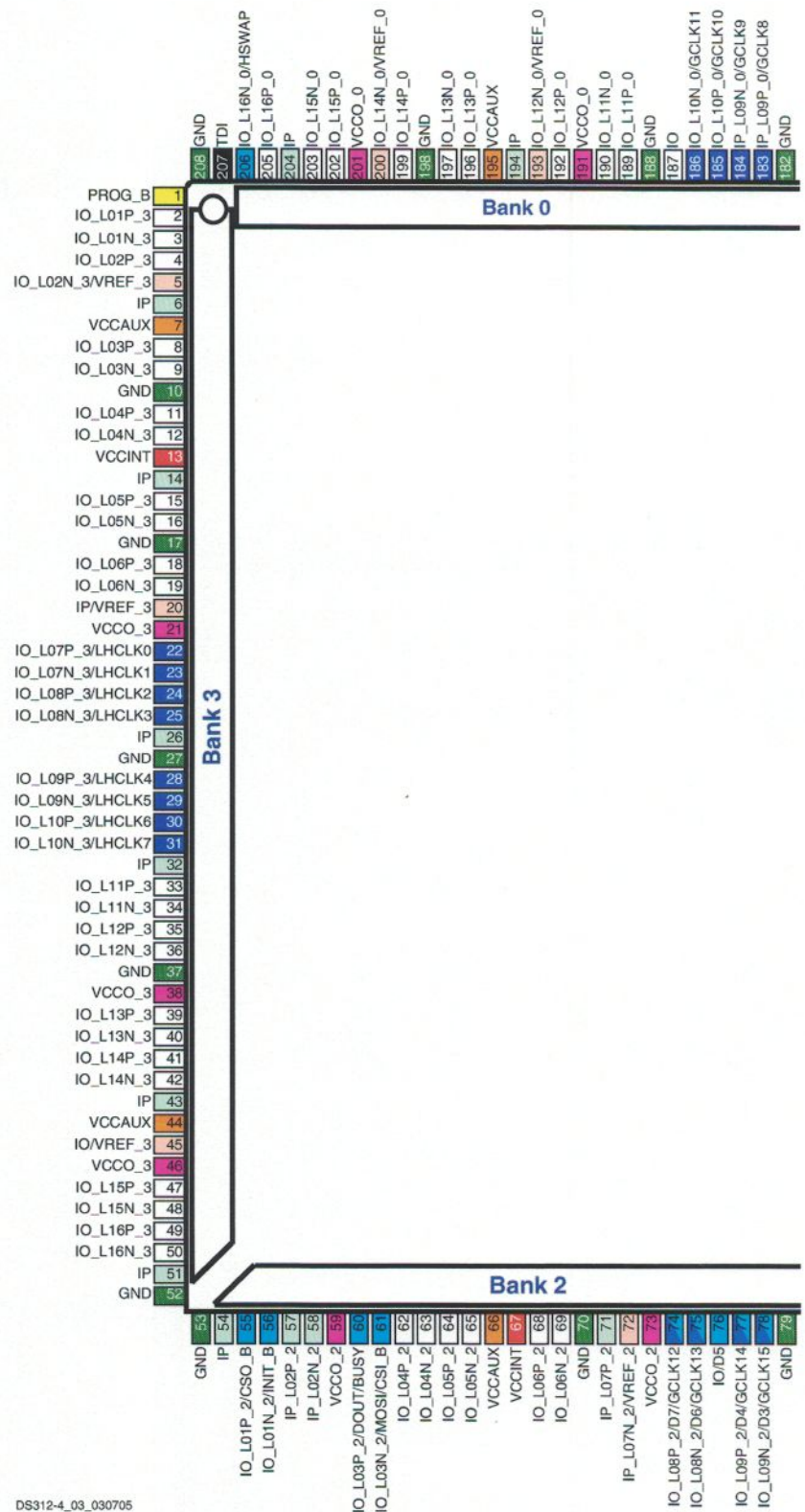
Table 141: User I/Os Per Bank for the XC3S250E and XC3S500E in the PQ208 Package

Package Edge	I/O Bank	Maximum I/O	All Possible I/O Pins by Type				
			I/O	INPUT	DUAL	VREF	CLK
Top	0	38	18	6	1	5	8
Right	1	40	9	7	21	3	0 ⁽¹⁾
Bottom	2	40	8	6	24	2	0 ⁽¹⁾
Left	3	40	23	6	0	3	8
TOTAL		158	58	25	46	13	16

Notes:

- The eight global clock pins in this bank have optional functionality during configuration and are counted in the DUAL column.

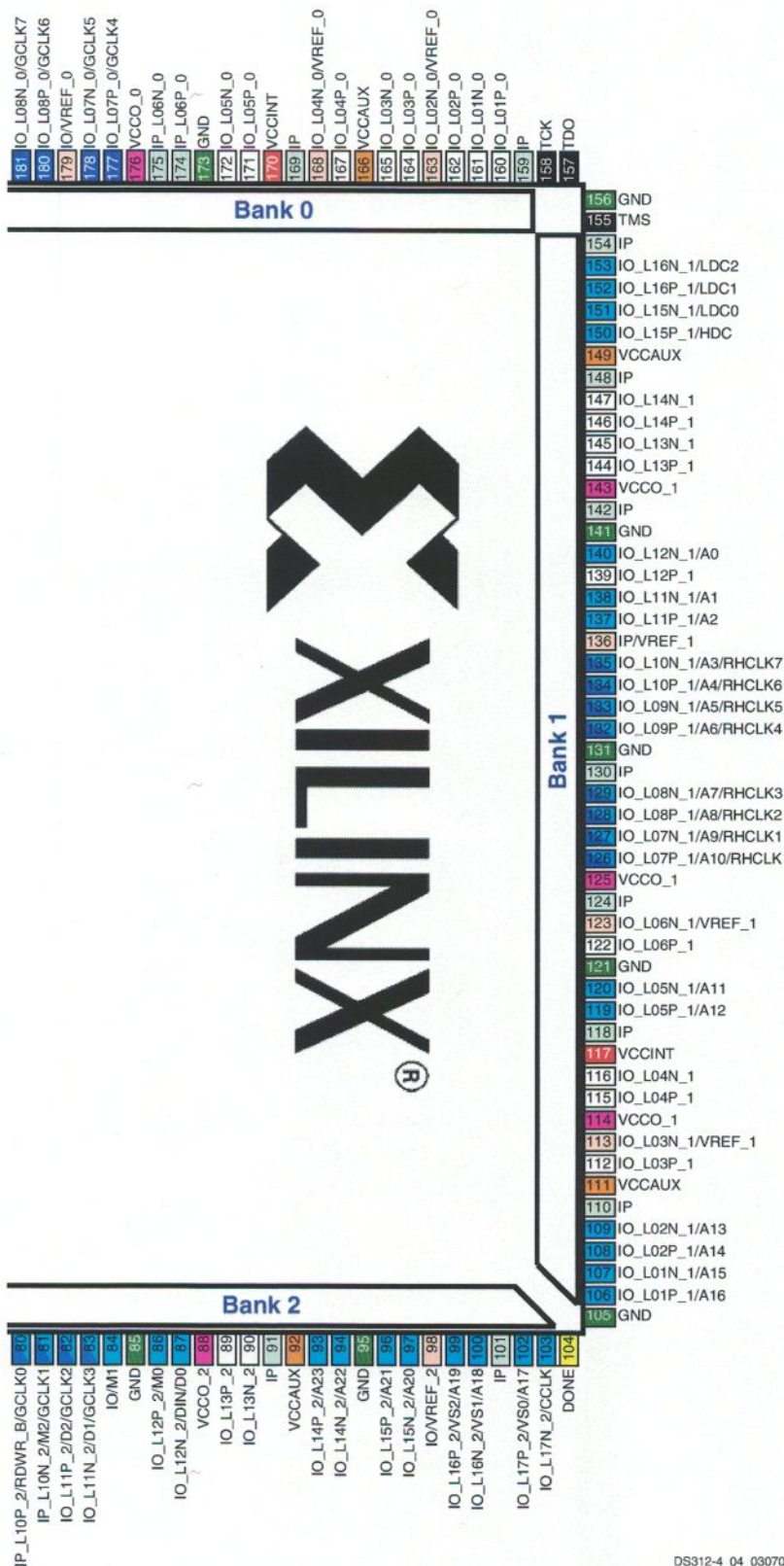
PQ208 Footprint (Left)



DS312-4_03_030705

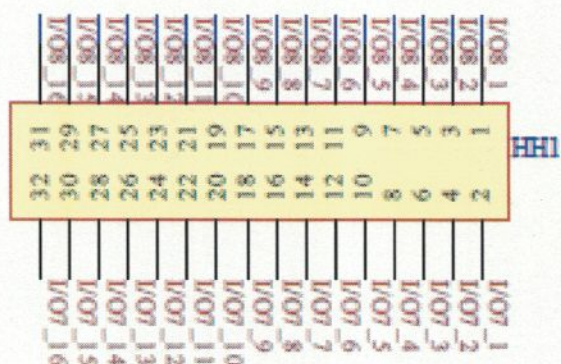
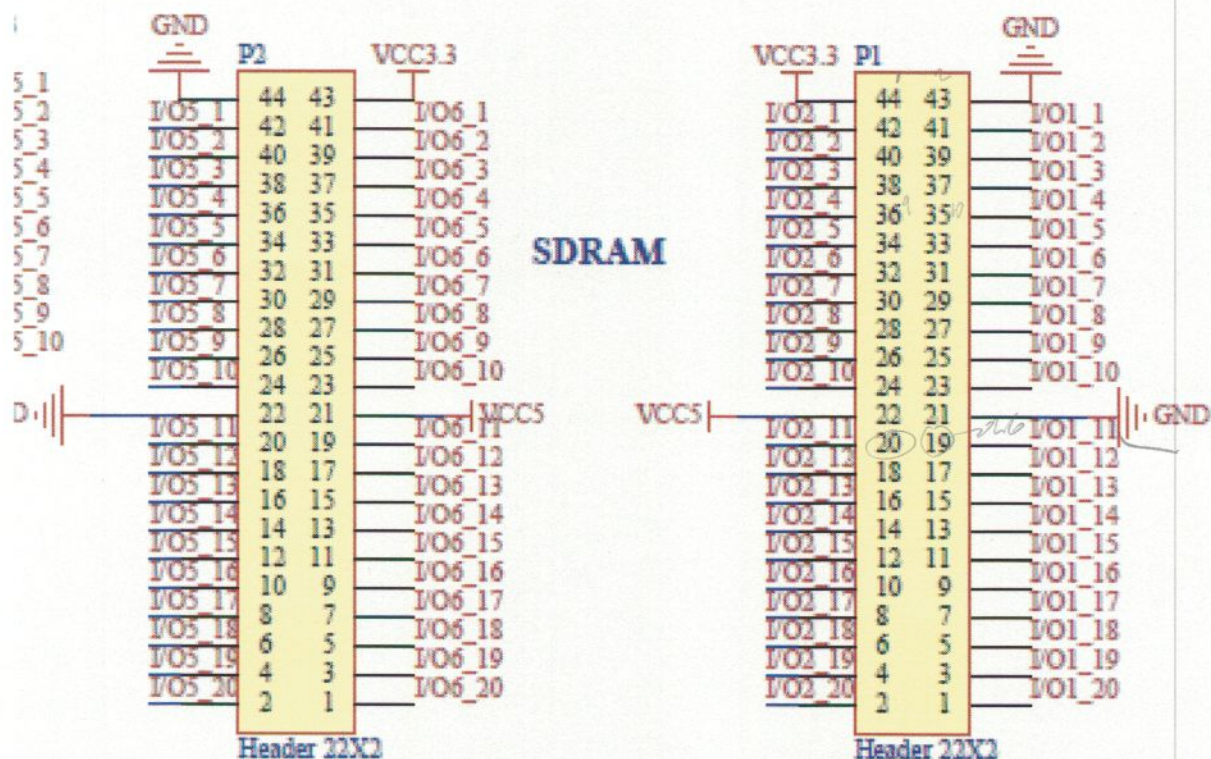
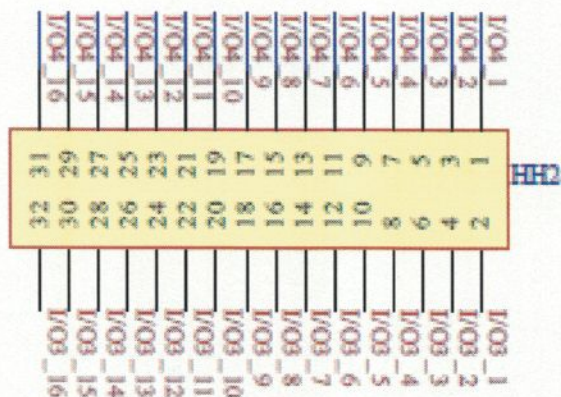
Figure 84: PQ208 Footprint (Left)

PQ208 Footprint (Right)

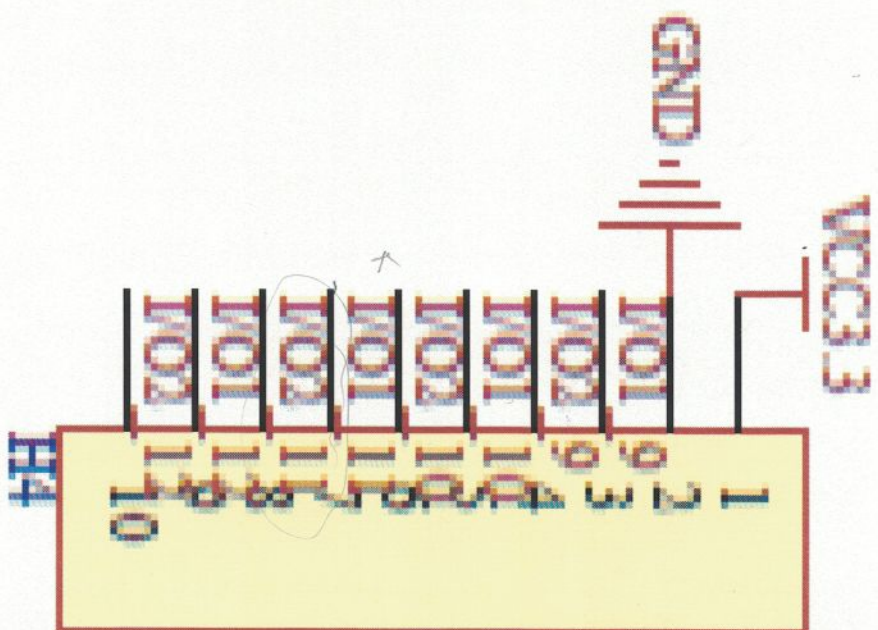


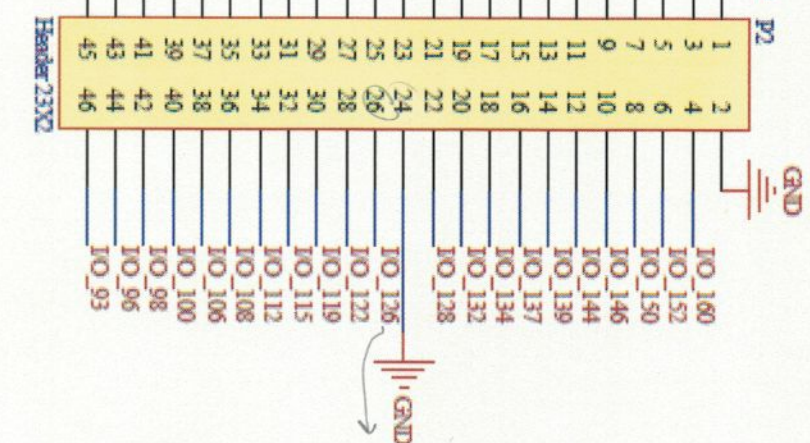
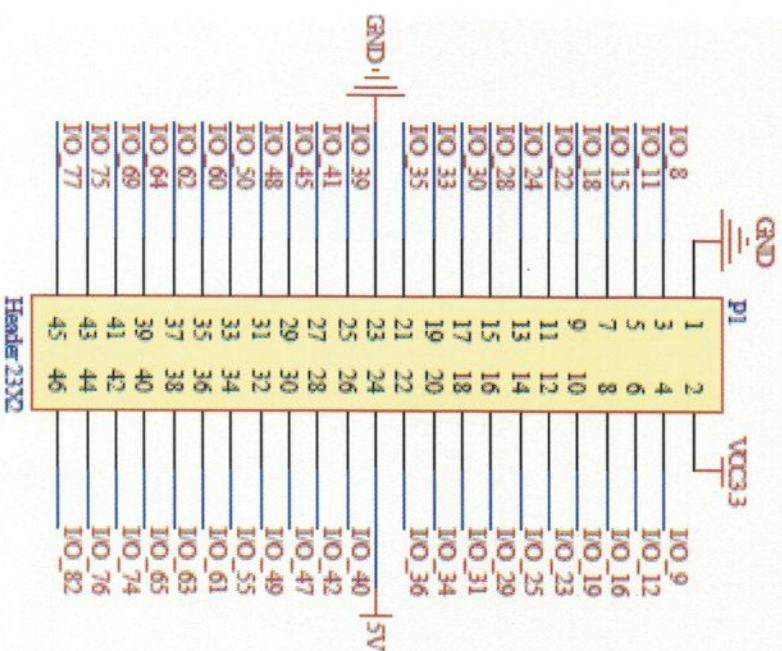
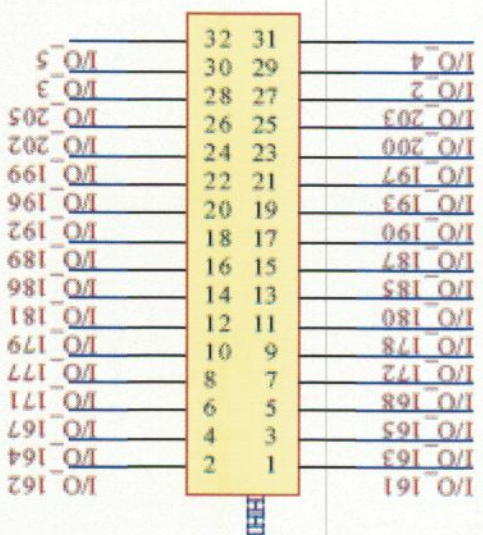
DS312-4_04_030705

Figure 85: PQ208 Footprint (Right)



8I/Os_1






```

1  -- sig01
2  -- Started 05/13/2022
3  -- This is a signal pulse generator.
4
5  -- VHDL library Declarations
6  LIBRARY IEEE;
7  USE IEEE.STD_LOGIC_1164.ALL;
8  USE IEEE.STD_LOGIC_UNSIGNED.ALL;
9  USE IEEE.STD_LOGIC_ARITH.ALL;
10
11 -- The Entity Declarations
12 ENTITY sig01 IS
13     PORT
14     (
15         -- rst & clk
16         reset_0:          IN STD_LOGIC;
17         clk:              IN STD_LOGIC;    -- 50 MHz
18
19         -- Outputs for the signal generator.
20         freq_10000_hz_output: OUT STD_LOGIC; -- 10000 Hz clock - 50% duty cycle
21         freq_1000_hz_output:  OUT STD_LOGIC; -- 1000 Hz clock - 50% duty cycle
22         freq_100_hz_output:   OUT STD_LOGIC; -- 100 Hz clock - 50% duty cycle
23         freq_10_hz_output:    OUT STD_LOGIC; -- 10 Hz clock - 50% duty cycle
24         freq_1_hz_output:     OUT STD_LOGIC; -- 1 Hz clock - 50% duty cycle
25         freq_p1_hz_output:    OUT STD_LOGIC; -- 0.1 Hz clock - 50% duty cycle
26         led01:               OUT STD_LOGIC; -- LED01 - 100 Hz flasher
27         led02:               OUT STD_LOGIC; -- LED02 - 10 Hz flasher
28         led03:               OUT STD_LOGIC; -- LED03 - 1 Hz flasher
29         led04:               OUT STD_LOGIC; -- LED04 - 0.1 Hz flasher
30 END sig01;
31
32 -- The Architecture of Entity Declarations
33 ARCHITECTURE Behavioral OF sig01 IS
34     SIGNAL freq_10000_hz: STD_LOGIC:= '0'; -- 10000 Hz - pulse
35     SIGNAL freq_1000_hz:  STD_LOGIC:= '0'; -- 1000 Hz - pulse
36     SIGNAL freq_100_hz:   STD_LOGIC:= '0'; -- 100 Hz - pulse
37     SIGNAL freq_10_hz:    STD_LOGIC:= '0'; -- 10 Hz - pulse
38     SIGNAL freq_1_hz:     STD_LOGIC:= '0'; -- 1 Hz - pulse
39     SIGNAL counter_10000_hz: INTEGER:= 4999; -- Countdown
40     SIGNAL counter_1000_hz:  INTEGER:= 9; -- Countdown
41     SIGNAL counter_100_hz:   INTEGER:= 9; -- Countdown
42     SIGNAL counter_10_hz:    INTEGER:= 9; -- Countdown
43     SIGNAL counter_1_hz:     INTEGER:= 9; -- Countdown
44     SIGNAL counter_p1_hz:    INTEGER:= 9; -- Countdown
45     SIGNAL freq_10000_hz_dc: STD_LOGIC:= '0'; -- 10000 Hz - pulse 50% Duty Cycle
46     SIGNAL freq_1000_hz_dc:  STD_LOGIC:= '0'; -- 1000 Hz - pulse 50% Duty Cycle
47     SIGNAL freq_100_hz_dc:   STD_LOGIC:= '0'; -- 100 Hz - pulse 50% Duty Cycle
48     SIGNAL freq_10_hz_dc:    STD_LOGIC:= '0'; -- 10 Hz - pulse 50% Duty Cycle
49     SIGNAL freq_1_hz_dc:     STD_LOGIC:= '0'; -- 1 Hz - pulse 50% Duty Cycle
50     SIGNAL freq_p1_hz_dc:    STD_LOGIC:= '0'; -- 0.1 Hz - pulse 50% Duty Cycle
51 BEGIN
52
53 -- Service the leds
54 PROCESS(reset_0,clk)
55 BEGIN
56     IF reset_0 = '0' THEN
57         led01 <= '1';
58         led02 <= '1';
59         led03 <= '1';
60         led04 <= '1';
61     ELSIF RISING_EDGE(clk) THEN
62         IF freq_100_hz_dc = '1' THEN
63             led01 <= '0';
64         ELSE
65             led01 <= '1';
66         END IF;
67         IF freq_10_hz_dc = '1' THEN

```

```

68         led02 <= '0';
69     ELSE
70         led02 <= '1';
71     END IF;
72     IF freq_1_hz_dc = '1' THEN
73         led03 <= '0';
74     ELSE
75         led03 <= '1';
76     END IF;
77     IF freq_p1_hz_dc = '1' THEN
78         led04 <= '0';
79     ELSE
80         led04 <= '1';
81     END IF;
82 END IF;
83 END PROCESS;
84
85 -- 10000 Hz signal generator
86 PROCESS(reset_0,clk)
87 BEGIN
88     IF reset_0 = '0' THEN
89         counter_10000_hz <= 4999;
90         freq_10000_hz <= '0';
91         freq_10000_hz_dc <= '0';
92     ELSIF RISING_EDGE(clk) THEN
93         IF counter_10000_hz /= 0 THEN
94             counter_10000_hz <= counter_10000_hz - 1;
95         ELSE
96             counter_10000_hz <= 4999;
97         END IF;
98         IF counter_10000_hz = 0 THEN
99             freq_10000_hz <= '1';
100        ELSE
101            freq_10000_hz <= '0';
102        END IF;
103        IF counter_10000_hz < 2500 THEN
104            freq_10000_hz_dc <= '1';
105        ELSE
106            freq_10000_hz_dc <= '0';
107        END IF;
108    END IF;
109 END PROCESS;
110
111 freq_10000_hz_output <= freq_10000_hz_dc;
112
113 -- 1000 Hz signal generator
114 PROCESS(reset_0,freq_10000_hz)
115 BEGIN
116     IF reset_0 = '0' THEN
117         counter_1000_hz <= 9;
118         freq_1000_hz <= '0';
119         freq_1000_hz_dc <= '0';
120     ELSIF RISING_EDGE(freq_10000_hz) THEN
121         IF counter_1000_hz /= 0 THEN
122             counter_1000_hz <= counter_1000_hz - 1;
123         ELSE
124             counter_1000_hz <= 9;
125         END IF;
126         IF counter_1000_hz = 0 THEN
127             freq_1000_hz <= '1';
128         ELSE
129             freq_1000_hz <= '0';
130         END IF;
131         IF counter_1000_hz < 5 THEN
132             freq_1000_hz_dc <= '1';
133         ELSE
134             freq_1000_hz_dc <= '0';

```



```

135         END IF;
136     END IF;
137 END PROCESS;
138
139 freq_1000_hz_output <= freq_1000_hz_dc;
140
141 -- 100 Hz signal generator
142 PROCESS(reset_0,freq_1000_hz)
143 BEGIN
144     IF reset_0 = '0' THEN
145         counter_100_hz <= 9;
146         freq_100_hz <= '0';
147         freq_100_hz_dc <= '0';
148     ELSIF RISING_EDGE(freq_1000_hz) THEN
149         IF counter_100_hz /= 0 THEN
150             counter_100_hz <= counter_100_hz - 1;
151         ELSE
152             counter_100_hz <= 9;
153         END IF;
154         IF counter_100_hz = 0 THEN
155             freq_100_hz <= '1';
156         ELSE
157             freq_100_hz <= '0';
158         END IF;
159         IF counter_100_hz < 5 THEN
160             freq_100_hz_dc <= '1';
161         ELSE
162             freq_100_hz_dc <= '0';
163         END IF;
164     END IF;
165 END PROCESS;
166
167 freq_100_hz_output <= freq_100_hz_dc;
168
169 -- 10 Hz signal generator
170 PROCESS(reset_0,freq_100_hz)
171 BEGIN
172     IF reset_0 = '0' THEN
173         counter_10_hz <= 9;
174         freq_10_hz <= '0';
175         freq_10_hz_dc <= '0';
176     ELSIF RISING_EDGE(freq_100_hz) THEN
177         IF counter_10_hz /= 0 THEN
178             counter_10_hz <= counter_10_hz - 1;
179         ELSE
180             counter_10_hz <= 9;
181         END IF;
182         IF counter_10_hz = 0 THEN
183             freq_10_hz <= '1';
184         ELSE
185             freq_10_hz <= '0';
186         END IF;
187         IF counter_10_hz < 5 THEN
188             freq_10_hz_dc <= '1';
189         ELSE
190             freq_10_hz_dc <= '0';
191         END IF;
192     END IF;
193 END PROCESS;
194
195 freq_10_hz_output <= freq_10_hz_dc;
196
197 -- 1 Hz signal generator
198 PROCESS(reset_0,freq_10_hz)
199 BEGIN
200     IF reset_0 = '0' THEN
201         counter_1_hz <= 9;

```

```

202     freq_1_hz <= '0';
203     freq_1_hz_dc <= '0';
204     ELSIF RISING_EDGE(freq_10_hz) THEN
205         IF counter_1_hz /= 0 THEN
206             counter_1_hz <= counter_1_hz - 1;
207         ELSE
208             counter_1_hz <= 9;
209         END IF;
210         IF counter_1_hz = 0 THEN
211             freq_1_hz <= '1';
212         ELSE
213             freq_1_hz <= '0';
214         END IF;
215         IF counter_1_hz < 5 THEN
216             freq_1_hz_dc <= '1';
217         ELSE
218             freq_1_hz_dc <= '0';
219         END IF;
220     END IF;
221 END PROCESS;
222
223 freq_1_hz_output <= freq_1_hz_dc;
224
225 -- 0.1 Hz signal generator
226 PROCESS(reset_0, freq_1_hz)
227 BEGIN
228     IF reset_0 = '0' THEN
229         counter_p1_hz <= 9;
230         freq_p1_hz_dc <= '0';
231     ELSIF RISING_EDGE(freq_1_hz) THEN
232         IF counter_p1_hz /= 0 THEN
233             counter_p1_hz <= counter_p1_hz - 1;
234         ELSE
235             counter_p1_hz <= 9;
236         END IF;
237         IF counter_p1_hz < 5 THEN
238             freq_p1_hz_dc <= '1';
239         ELSE
240             freq_p1_hz_dc <= '0';
241         END IF;
242     END IF;
243 END PROCESS;
244
245 freq_p1_hz_output <= freq_p1_hz_dc;
246
247 END Behavioral;

```



```

1  # use the reset switch on the Core3S500E
2  NET "reset_0"    LOC = "P101";
3  NET "clk"        LOC = "P184";
4
5  # LEDs on the Core3S500E
6  NET "led01"      LOC = "p89";    # 10.0 Hz indicator
7  NET "led02"      LOC = "p83";    # 1.0 Hz indicator
8  NET "led03"      LOC = "p78";    # 0.1 Hz indicator
9  NET "led04"      LOC = "p68";    # 10,000 Hz counter zero count indicator
10
11 # Use 8I/Os_2
12 NET "freq_10000_hz_output"    LOC = "p132";    #8I/Os_2_1
13 NET "freq_1000_hz_output"     LOC = "p129";    #8I/Os_2_2
14 NET "freq_100_hz_output"      LOC = "p128";    #8I/Os_2_3
15 NET "freq_10_hz_output"       LOC = "p127";    #8I/Os_2_4
16 NET "freq_1_hz_output"        LOC = "p126";    #8I/Os_2_5
17 NET "freq_p1_hz_output"       LOC = "p123";    #8I/Os_2_6

```