**Freescale Semiconductor**
**User's Guide**

Document Number: PTKIT711xUG
Rev. 1, 04/2006

# MSC711x Packet Telephony Farm Card (SPT711xPFCE)

The MSC711x packet telephony farm card (SPT711xPFCE) accelerates time to market of MSC711x DSP voice over internet protocol (VoIP) designs. This card contains a farm of four StarCore™-based MSC711x DSP devices aggregated through a ten-port Ethernet switch and interfacing with a computer telephony (CT) bus carrying up to four TDM streams. The board meets the common mezzanine card (CMC) specification for a single-width form factor. Two separate CMC connector sites, one on each side, provide the pin connections for two different development platforms.

The board is designed to connect to the Freescale Packet Telephony Development Kit (PDK) and the Freescale Modular Development System (MDS). Both development platforms are designed with Freescale PowerQUICC™ processors. One side has three CMC connectors placed as described in the CMC specification, plus a fourth non-standard CMC connector for additional signalling. The other side has all four standard CMC connectors. All power/ground and most signal connections match the PCI telecom mezzanine card (PTMC) specification (type 3).

The devices can boot standalone through an on-board $I^2C$ EEPROM or under external host control through the host processor bus interface. Typical data flow converts VoIP traffic from the Ethernet switch to the TDM bus and generates VoIP traffic using data from the TDM bus.

**Contents**





freescale™ semiconductor

Software support and examples for the SPT711xPFCE are available in the PDK. The latest release of the `libpdk` software for Linux supports the SPT711xPFCE.

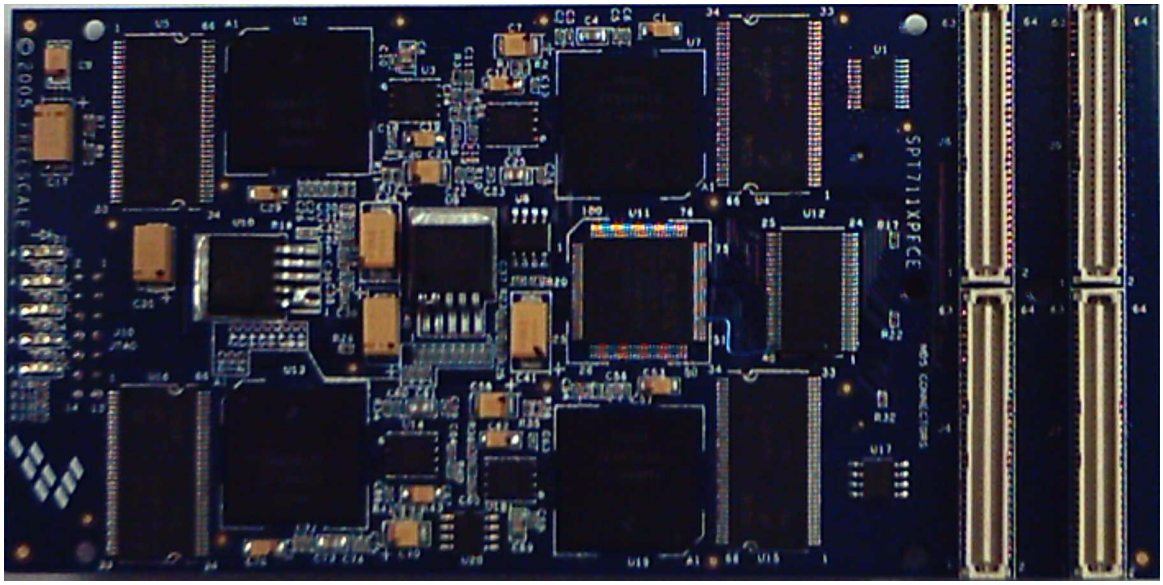



**Figure 1. MSC711x Packet Telephony Farm Card (SPT711xPFCE)**

Features of the SPT711xPFCE are as follows:

- CMC/PMC form factor.
- Two sets of CMC connector sites:
  - Compatible with the Packet Telephony Development Kit (PDK).
  - Compatible with the PowerQUICC™ modular design system (MDS).
- 4 MSC711x DSP devices running at up to 300 MHz.
- 16 Mbyte of 16-bit double data rate (DDR) SDRAM for each MSC711x DSP device.
- 32 Kbyte of $I^2C$ EEPROM for MSC711x boot memory.
- 32 Kbyte of $I^2C$ EEPROM for Ethernet switch configuration.
- On-board 10-port Ethernet switch.
- Four TDM streams connected to all MSC711x DSP devices.
- 72 Macrocell complex programmable logic device (CPLD) for I/O and reset control.
- **JTAG™** connection for in-system debugging of all MSC711x DSP devices and CPLD.
- CPLD control of boot time mode selection.
- Requires 3.3 VDC and 5 VDC supplied from an external source:
  - 3.3 VDC generates 2.5 VDC for DDR interfaces.
  - 5 VDC generates 1.2 VDC for MSC711X DSP device core voltage.

# 1 Packet Telephony Development Kit

The *Packet Telephony Development Kit (PDK)* is a Freescale platform for evaluating and developing voice-over packet applications. The PDK has an MPC8260 host network processor that runs Linux, StarCore DSP resource cards that run DSP code, and a public switched telephone network (PSTN) card with interfaces such as E1/T1 and analog telephone lines (see **Figure 2**).

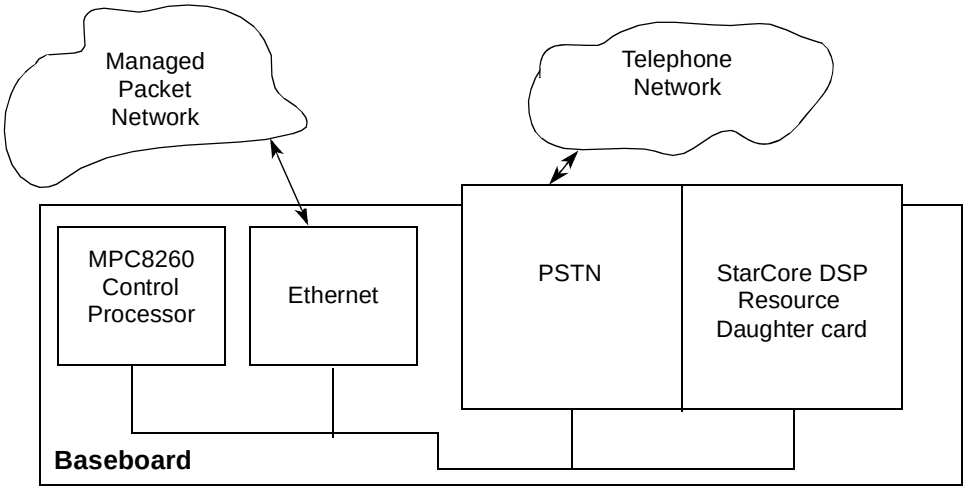



**Figure 2. Components of the PDK**

The documentation for the kit components is listed in **Table 1**.

**Table 1. PTK Components and Their Associated Documents**

| Component | Document | Document ID |
|---|---|---|
| Baseboard | *Packet Development Kit Baseboard Hardware User's Guide* | PTKITBASEUG |
| MPC8260 control processor | *MPC8260 PowerQUICC II™ Family Reference Manual*<br>(Available at the website listed on the back page of this user's guide.) | MPC8260UM |
| PSTN card | *Packet Development Kit PSTN Mezzanine User's Guide* | PTKITPSTNUG |
| StarCore DSP resource daughter card | • *MSC8102 Packet Telephony Farm Card (MSC8102PFC) User's Guide*<br>• *MSC8101 Packet Telephony Farm Card (MSC8101PFC) User's Guide*<br>• *MSC8122 Packet Telephony Farm Card (MSC8122PFC-HV) User's Guide*<br>• *MSC711x Packet Telephony Farm Card (SPT711xPFCE) User's Guide* | PTKIT8102UG<br>PTKIT8101UG<br>PTKIT8122UG<br>PTKIT711xUG |
| MSC8122 processor | *MSC8122 Reference Manual* and other MSC8122 documentation are located at the web site listed on the back cover of this user's guide. | MSC8122RM |
| MSC8103 processor | *MSC8103 Reference Manual* and other MSC8103 documentation are located at the web site listed on the back cover of this user's guide. | MSC8103RM |
| Software | *Packet Telephony Development Kit Software User's Guide* | PTKITSOFTUG |

**CAUTION**

The packet telephony development kit includes open-construction printed circuit boards that contain static-sensitive components. These boards are subject to damage from electrostatic discharge (ESD). To prevent such damage, you must use static-safe work surfaces and grounding straps, as defined in ANSI/EOS/ESD S6.1 and ANSI/EOS/ESD S4.1. All handling of these boards must be in accordance with ANSI/EAI 625.

**Table 2. Reference Documents**

| Document | Revision | Date | Document ID |
|---|---|---|---|
| Standard Physical and Environmental layers for PCI Mezzanine Cards: PMC | Draft 2.4 | January 12, 2001 | IEEE: P1386.1 |
| Standard for a Common Mezzanine Card Family: CMC | Draft 2.4a | March 21, 2001 | IEEE: P1386 |
| CompactPCI PCI Telecom Mezzanine Card Specification | R1.0 | April 11, 2001 | PICMG 2.15 |
| H.100 Hardware Compatibility Specification: CT Bus | 1.0 | | H.100 |

# 2 System Interfaces

To support VoIP applications development, the SPT711xPFCE provides four time-division multiplex (TDM) and four Ethernet interfaces at the CMC connectors (see **Figure 3**). It includes interfaces for program storage, download, and debug. These interfaces can also be used to transfer control messages with a system controller.

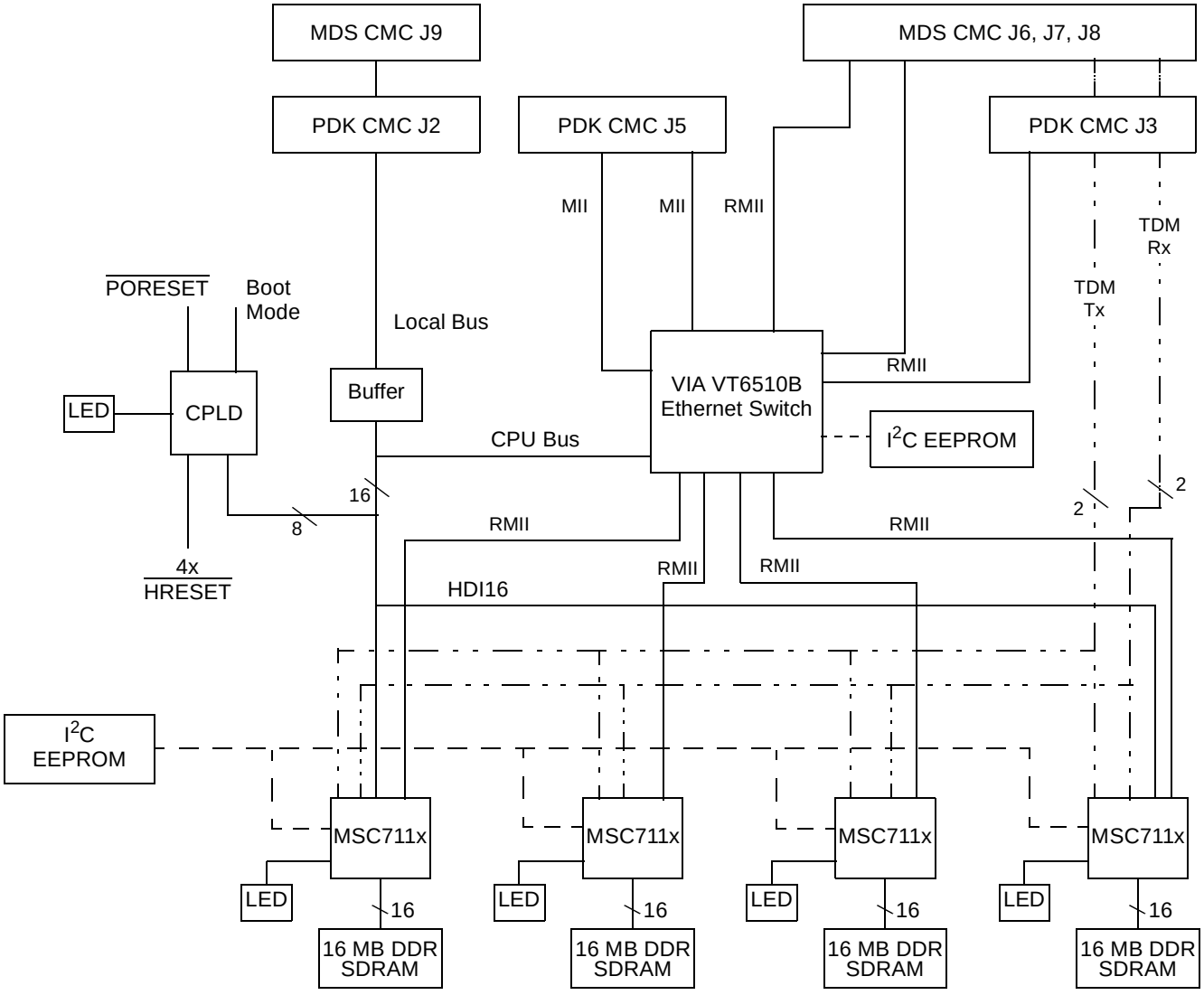



**Figure 3. Block Diagram of SPT711xPFCE**

## 2.1 External Bus Through Programmable Logic

A complex programmable logic device (CPLD) converts bus transfers on the CMC connectors to HDI16, Ethernet switch, or CPLD register transfers. This logic decodes the address and asserts the proper chip-select signal when the PFC chip-select signal is asserted. The host device must meet the timing requirements for the MSC711x HDI16, Ethernet switch, and CPLD register transfers, assuming a 10 ns delay through the CPLD. **Figure 4** shows the timing requirements for the Ethernet switch, which is the slowest interface and therefore the driving influence on timing considerations. The actual value of timing requirements is implementation-specific. Review all device specifications to determine the values.

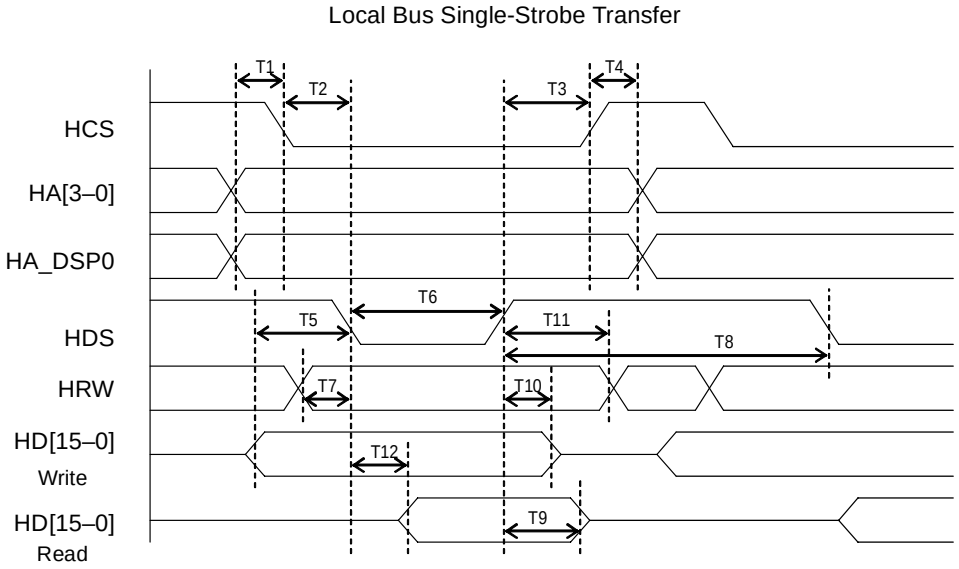



**Figure 4. Bus Transfer Timing Diagram**

The CPLD contains registers for controlling and/or sampling several board signals. These signals include reset, power and LED control, and interrupts. Consult **Section 7**, *CPLD Registers*, on page 25 and the board schematics for information on signals controlled or sampled by the CPLD registers.

**Table 3. Bus Transfer Timing Intervals**

| Description | Key | Minimum |
|---|---|---|
| Address set-up before the chip select | T1 | 10 ns |
| Chip-select set-up before the data strobe | T2 | 10 ns |
| Chip-select hold after the data strobe | T3 | 10 ns |
| Address hold after the chip select | T4 | 0 ns |
| Write data set-up before the data strobe assertion | T5 | 10 ns |
| Data strobe width | T6 | 40 ns |
| HRW set-up before the data strobe | T7 | 10 ns |
| Data strobe after the data strobe | T8 | 20 ns |
| Read data hold after the data strobe deassertion | T9 | 10 ns |
| Write data hold after the data strobe deassertion | T10 | 5 ns |
| HRW hold after the data strobe | T11 | 10 ns |
| Read data valid after the data strobe assertion | T12 | 30 ns |

## 2.2 Ethernet (MII/RMII)

Two media-independent interfaces (MII) and three reduced MII (RMII) are available for connection to an external host board at the CMC connectors. All four MSC711x devices connect to separate full duplex RMII ports of the on-board ten-port Ethernet switch. One full duplex RMII port from the switch connects to J3 of the PDK CMC site. Two full duplex RMII ports for the switch connect to J8 of the MDS CMC site. **Figure 5** shows a diagram of the Ethernet switch port connections on the board. The switch is configured through its CPU bus interface (see **Figure 3**).

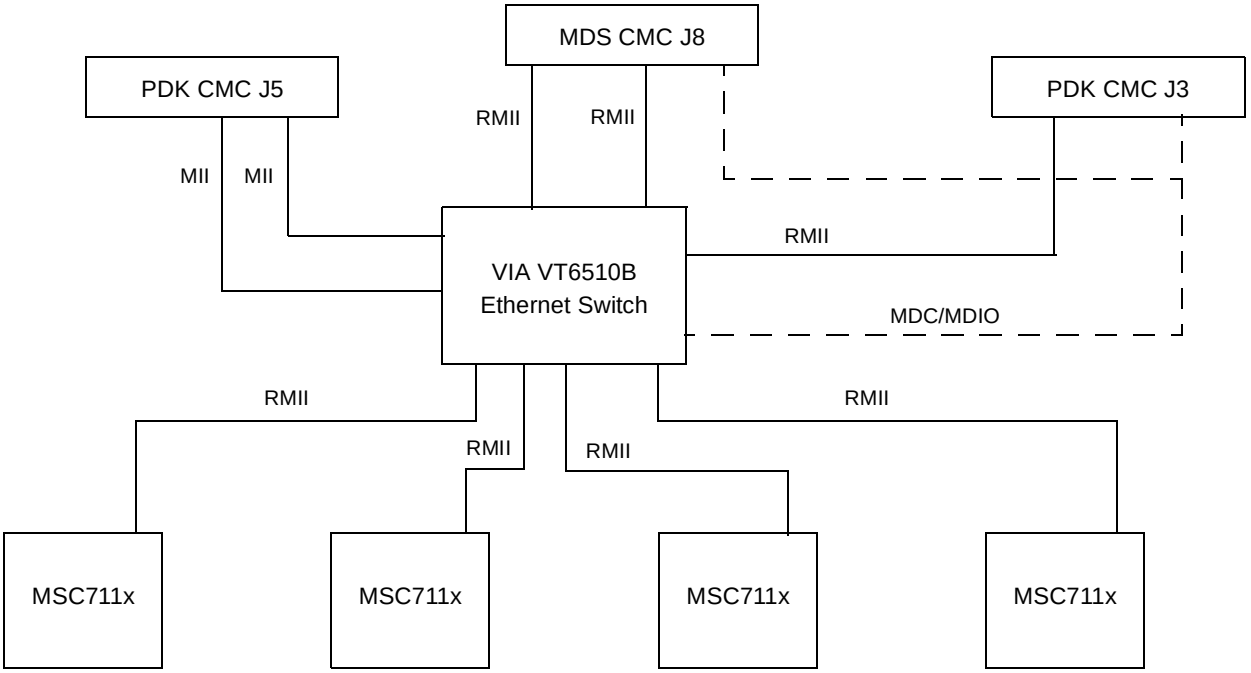



**Figure 5. Ethernet Connections**

**Table 4. VIA Ethernet Switch Port Connections**

| Port | Type | Connection |
|---|---|---|
| 0 | RMII | MSC711x Device 0 |
| 1 | RMII | MSC711x Device 1 |
| 2 | RMII | MSC711x Device 2 |
| 3 | RMII | MSC711x Device 3 |
| 4 | RMII | MDS CMC |
| 5 | RMII | MDS CMC P3 |
| 6 | RMII | PDK CMC |
| 7 | RMII | No Connection |
| 8 | MII | PDK CMC P5 |
| 9 | MII | PDK CMC P5 |

## 2.3 Time-Division Multiplex (TDM) Bus

Four streams of TDM data are available for connection to an external host board and are connected to all MSC711x DSP devices. The streams connect to J3 of the PDK CMC site and J6, J7, and J8 of the MDS CMC site. One pair of frame and clock signalling is provided for all TDM streams. Streams CT_D1 and CT_D5 must be used for TDM streams from the external host to the DSP farm. Streams CT_D0 and CT_D4 must be used for TDM streams from the DSP farm to the external host. The frame and clock signals are buffered and then connected to each DSP device. The CT_D0, CT_D1, CT_D5, and CT_D4 signals are connected directly from the DSP devices to the CMC connectors. **Figure 6** shows the TDM stream connections on the board.

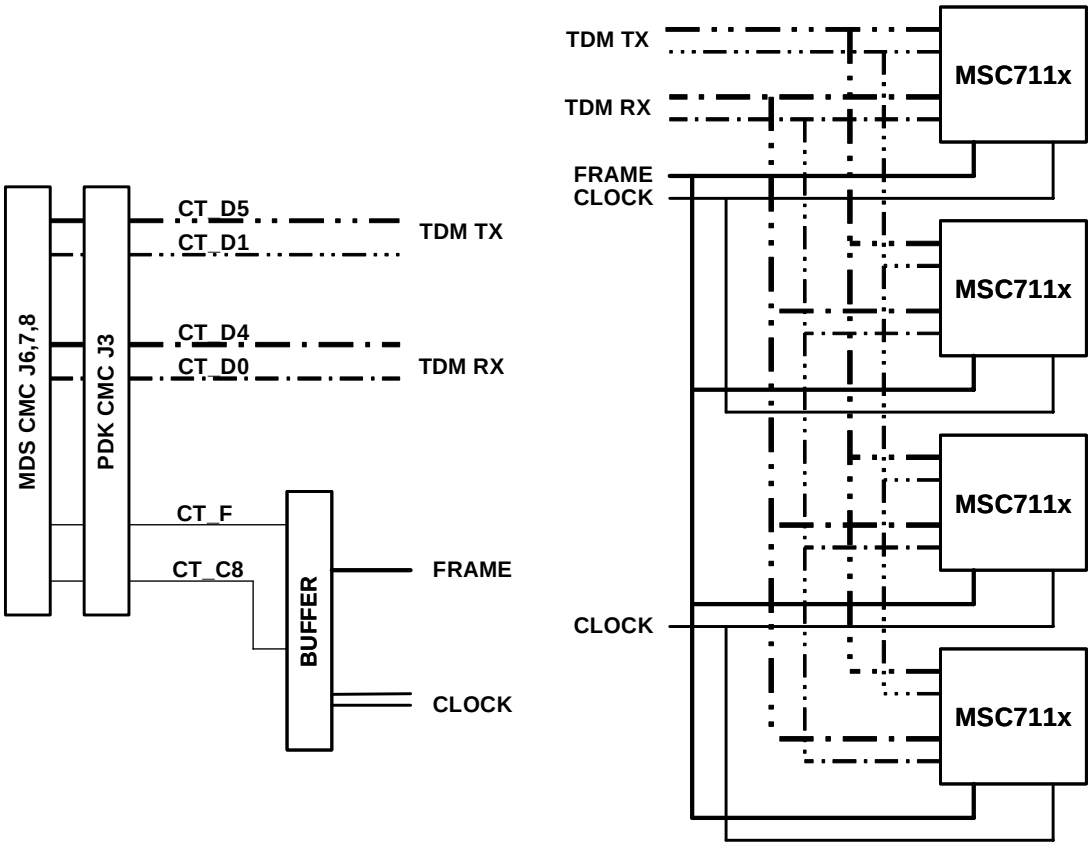



**Figure 6. TDM Connections**

## 2.4 JTAG Connector

Connector J10 is a 14-pin 0.100 inch pitch header that provides JTAG debugger access to the board. For details on the connector signals, refer to the schematics in Appendix A. One JTAG chain on the board connects all four MSC711x devices and the CPLD. The chain is connected as follows:


# 3 Hardware

**Figure 7** depicts the main components of the SPT711xPFCE board, which are the MSC711x DSP devices, CPLD, Ethernet switch, $I^2C$ bus devices, and voltage regulators.

## 3.1 MSC711x DSPs

The MSC711x family of highly integrated DSPs targets high-bandwidth computationally intensive DSP applications and is optimized for Enterprise class packet telephony applications. These processors deliver enhanced performance while maintaining low power dissipation and greatly reducing system cost.

At the heart of an MSC711x DSP device is the StarCore™ SC1400 core, providing the processing power for intensive numeric processing. The four ALUs in the SC1400 core work together to deliver 800 million multiply and add commands per second (MMACS) performance with an internal 200 MHz clock at 1.2 V. An extended core services the SC1400 bandwidth requirements, with high speed zero wait state memory elements for both program and data accesses. In the extended core are a multi-ported 64 or 192 Kbyte level 1 internal memory (M1) for both high speed program and data storage. A 16 Kbyte, 16-way instruction cache (ICache) provides an instruction stream to the core with no wait states on cache hits. The efficiency of the ICache is greatly enhanced by an intelligent fetch unit with advanced features for real-time processing. A 4-entry write buffer allows the SC1400 core to continue processing while the write buffer writes to locations outside the platform. A 192 Kbyte level 2 memory (M2) is also available on some MSC711x family devices for bursting to the ICache and for accesses from the extended core.

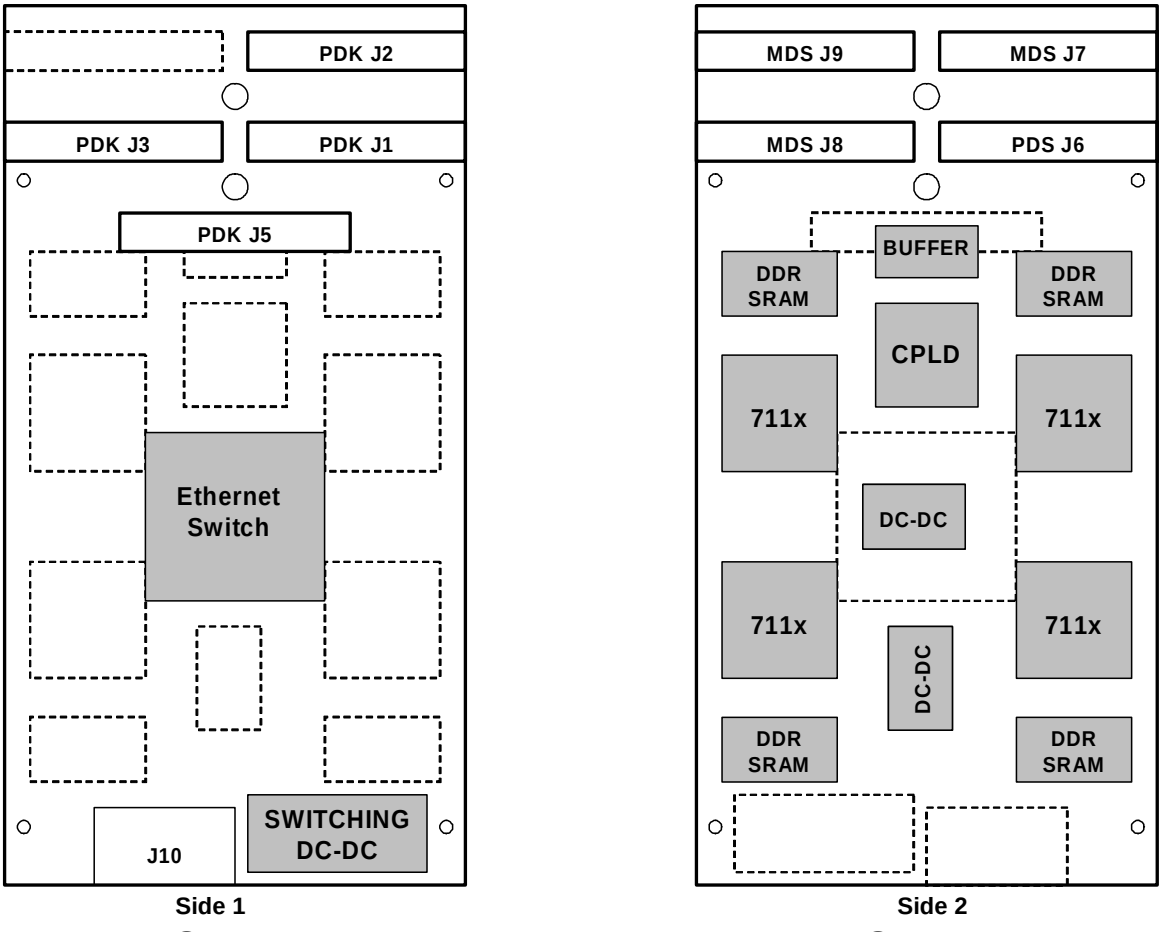



**Figure 7. Major Components of the SPT711xPFCE**

Each device in the MSC711x family is designed for optimal data flow to/from the SC1400 core. Wide full-speed buses exactly match the busing requirements of the SC1400 core. Data is transferred to the DSP from either the external memory interface, the Ethernet controller on some devices, the host interface, or the TDM serial interfaces. The flexible DMA controller transfers data through the bus switch from any of these ports to buffers in the internal memories. The SC1400 cores and other modules interconnect via a crossbar switch that manages rapid data transfer and storage between the MSC711x device, its internal components, and external devices. This multi-port crossbar switch allows multiple data transfers to occur

in parallel outside the extended core. The SC1400 core processes the data in the buffers and the result is transferred back to one of the ports. **Figure 8** shows a block diagram of the MSC7116 device. Unlike some devices in the MSC711x family, the MSC7116 has an Ethernet port.

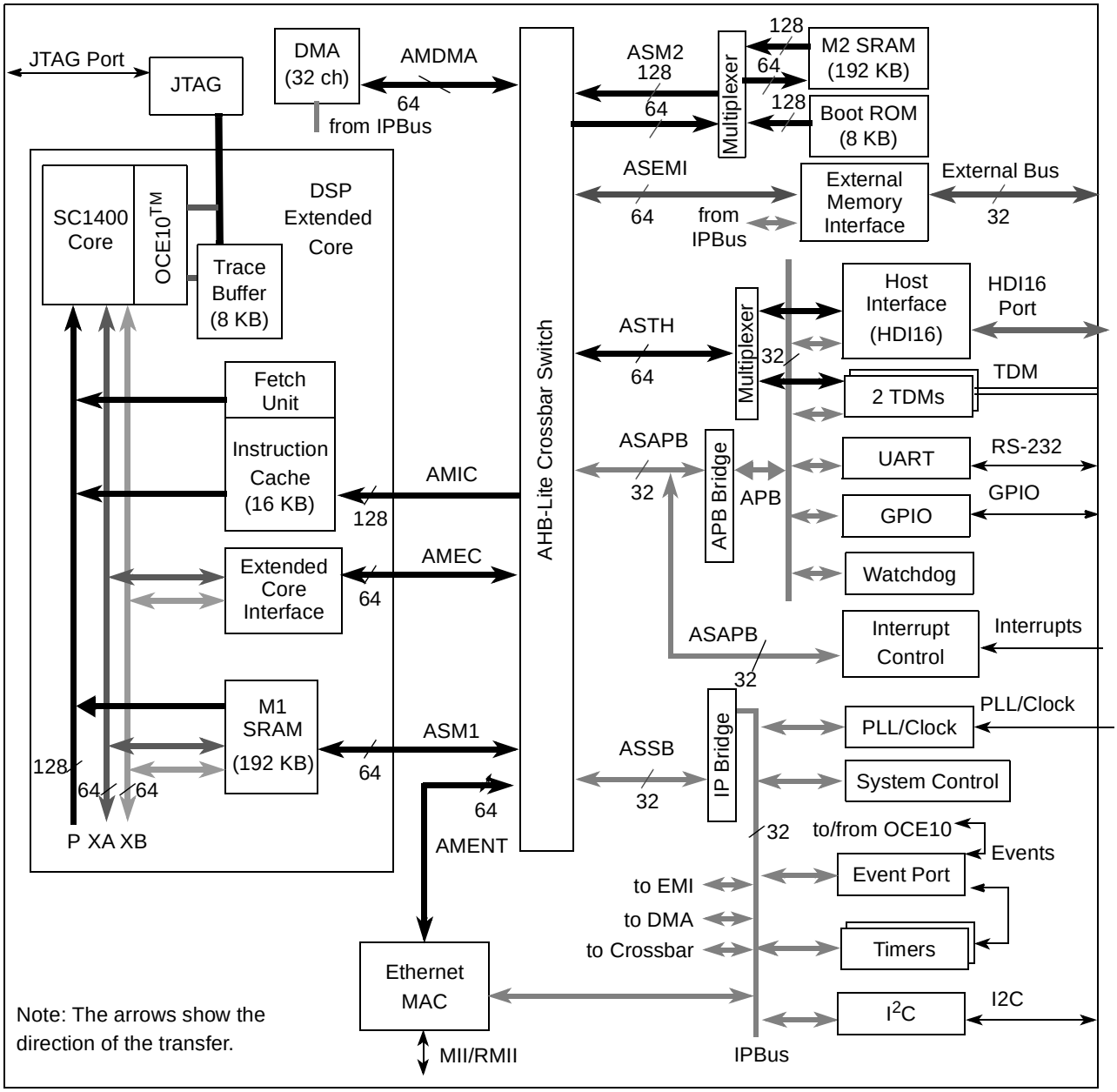



**Figure 8. MSC7116 Block Diagram.**

The flexible 32-channel DMA controller transfers data to and from internal memory, external memory, peripherals, the host data interface, and the TDM interfaces. The DDR-RAM memory controller enables the SC1400 cores to access external memory devices with glueless accesses to DDR-RAM memory devices on the system bus. For more information on the MSC711x family of DSP devices, visit the Freescale Semiconductor web site listed on the back cover of this document.

## 3.2 DDR SDRAM

Each MSC711x device has a dedicated 16-bit (DDR synchronous dynamic random access memory (SDRAM) device for external memory storage. These devices provide 16 Mbyte of external memory to each device. The memory devices connect to the dedicated DDR SDRAM interface on each MSC711x device. Example 1 shows the basic initialization code for the Micron DDR SRAM memory as connected on the SPT711xPFCE. These devices are Micron part number MT46V8M16P-6T. For information on these devices, refer to the Micron web site at `http:// www.micron.com`.

**Example 1. Micron DDR Initialization Code**

```
typedef struct

{
   unsigned int UP;
   unsigned int START;
   unsigned int BYTES;
   unsigned int WIDTH32;
   unsigned int CS0CFG;
   unsigned int TCFG1;
   unsigned int TCFG2;
   unsigned int SMCFG;
   unsigned int SICFG;
   unsigned int SCFG;
}  t_DDR_PARAMETERS;

t_DDR_PARAMETERS DDR_PARAMETERS;


#define DDR_DEFAULT_UP 0
#define DDR_DEFAULT_START 0x20000000
#define DDR_DEFAULT_BYTES 0x01000000
#define DDR_DEFAULT_32BIT 0
#define DDR_DEFAULT_CS0CFG 0x80000001
#define DDR_DEFAULT_TCFG1 0x38345331
#define DDR_DEFAULT_TCFG2 0x00000400
#define DDR_DEFAULT_SMCFG 0x10000062
#define DDR_DEFAULT_SICFG 0x0490007F
#define DDR_DEFAULT_SCFG 0x42000000

// Initialize Global Variables
DDR_PARAMETERS.UP = DDR_DEFAULT_UP;
DDR_PARAMETERS.START = DDR_DEFAULT_START;
DDR_PARAMETERS.BYTES = DDR_DEFAULT_BYTES;
DDR_PARAMETERS.WIDTH32 = DDR_DEFAULT_32BIT;
DDR_PARAMETERS.CS0CFG = DDR_DEFAULT_CS0CFG;
DDR_PARAMETERS.TCFG1 = DDR_DEFAULT_TCFG1;
DDR_PARAMETERS.TCFG2 = DDR_DEFAULT_TCFG2;
DDR_PARAMETERS.SMCFG = DDR_DEFAULT_SMCFG;
DDR_PARAMETERS.SICFG = DDR_DEFAULT_SICFG;
DDR_PARAMETERS.SCFG = DDR_DEFAULT_SCFG;

#if (DDR_DEFAULT_UP)
   // Default is up, so insert the code to bring it up.
   DDR_UP(&DDR_PARAMETERS);
#endif
```

```
void DDR_UP(t_DDR_PARAMETERS* params)
{
   unsigned int end;
   volatile ddr_memc_map_t* ddr = (volatile ddr_memc_map_t*) DDR_BASE;
   volatile clk_map_t* clk = (volatile clk_map_t*) CLK_BASE;
   volatile btm_map_t* btm = (volatile btm_map_t*) BTM_BASE;

   // STOPCTL - enable DDR clk
   clk->STOPCTRL &= ~0x00003000;

   // DEVCFG
   if (params->WIDTH32 == 1)
   { btm->CHPCFG |= 0x00000020; } // 32-bit DDR
   else
   {                             // 16-bit
       btm->CHPCFG &= ~0x00000020;

       if (btm->CHPCFG & 0x00000020) // Cannot be cleared
       { params->WIDTH32 = 1; }
   }

   // CSBR0
   end = (params->START) + (params->BYTES - 1);
   if (params->WIDTH32 == 1)
   { ddr->CS0_BNDS = (params->START >> 7) | (end >> 23); }
   else
   { ddr->CS0_BNDS = (params->START >> 6) | (end >> 22); }

   // CS0CFG
   ddr->CS0_CONFIG = params->CS0CFG;

   // TCFG1
   ddr->TIMING_CFG_1 = params->TCFG1;

   // TCFG2
   ddr->TIMING_CFG_2 = params->TCFG2;

   // SCFG (First Time)
   ddr->DDR_SDRAM_CFG = (params->SCFG);

   // SMCFG
   ddr->DDR_SDRAM_MODE = params->SMCFG;

   // SICFG
   ddr->DDR_SDRAM_INTERVAL = params->SICFG;

   // SCFG (Last Time)
   ddr->DDR_SDRAM_CFG = (params->SCFG) | 0x80000000;

   return;
}
```

## 3.3 Complex Programmable Logic Device (CPLD)

A Xilinx XC9572XL CPLD device implements chip-select logic and logic registers on the SPT711xPFCE board. This device contains 72 macrocells and is programmed through JTAG signals on J10. The CPLD self configures when reset is deasserted and does not require external configuration memory. For information on this device, refer to the Xilinx web site (http://www.xilinx.com) .

## 3.4 VIA VT6510B Ethernet Switch

The SPT711xPFCE has an on-board 10 port (2 × MII, 8 × RMII) Ethernet switch connected to both CMC sites and all four DSP devices. The Ethernet switch is a VIA VT6510B (http://www.vntek.com). Example 1 provides basic initialization code for the switch through a host controller connected to the CPLD host interface and using the memory map described in **Section 6**, *CPLD Memory Map*, on page 24.

**Code Listing 1.** VIA Switch Initialization Code

```
/*****************************************************************************
 * PFC Ethernet Switch Initialization Routines
 *
 *****************************************************************************/
void writeSwitchReg(unsigned int _addr, unsigned char _data)
{
  *pdk_bd_regs.u711xpfc.sw_addr_l = (unsigned short)(_addr%256);
  *pdk_bd_regs.u711xpfc.sw_addr_h = (unsigned short)(_addr/256);
  *pdk_bd_regs.u711xpfc.sw_reg_data = (unsigned short)(_data);
  pdk_waste_clks_10(1000);
}

unsigned char readSwitchReg(unsigned int _addr)
{
  unsigned char readData = 0;
  *pdk_bd_regs.u711xpfc.sw_addr_l = (unsigned short)(_addr%256);
  *pdk_bd_regs.u711xpfc.sw_addr_h = (unsigned short)(_addr/256);
  readData = (unsigned char) *pdk_bd_regs.u711xpfc.sw_reg_data;
  pdk_waste_clks_10(1000);
  return readData;
}

void enableSwitchPhyAutoPolling()
{
  while ((readSwitchReg(0x0405) & 1) != 0);
  writeSwitchReg(0x0404, 0x08); // Enable Auto Polling;
  while ((readSwitchReg(0x0405) & 1) != 0);
}

void disableSwitchPhyAutoPolling()
{
  while ((readSwitchReg(0x0405) & 1) != 0);
  writeSwitchReg(0x0404, 0x04); // Disable Auto Polling;
  while ((readSwitchReg(0x0405) & 1) != 0);
}

// _addr = {4-bit-port-num, 3'b000, 5-bit-reg-num}
void writeSwitchPhy(unsigned short _addr, unsigned short _data)
{
  // Note - Autopolling mode must be disabled
  while ((readSwitchReg(0x0405) & 1) != 0);
  writeSwitchReg(0x0400, (unsigned char)((_addr >> 8) & 0x000f));
  writeSwitchReg(0x0401, (unsigned char)(_addr & 0x1f));
  writeSwitchReg(0x0402, (unsigned char)(_data%256));
  writeSwitchReg(0x0403, (unsigned char)(_data/256));
  writeSwitchReg(0x0404, 0x01); // Write to Phy;
```

```
  while ((readSwitchReg(0x0405) & 1) != 0);
}

unsigned short readSwitchPhy(unsigned short _addr)
{
  unsigned short rtn;
  unsigned char vals[2];
  // Note - Autopolling mode must be disabled
  while ((readSwitchReg(0x0405) & 1) != 0);
  writeSwitchReg(0x0400, (unsigned char)((_addr >> 8) & 0x000f));
  writeSwitchReg(0x0401, (unsigned char)(_addr & 0x1f));
  writeSwitchReg(0x0404, 0x02); // Read from Phy;
  while ((readSwitchReg(0x0405) & 1) != 0);
  vals[0] = readSwitchReg(0x0402);
  vals[1] = readSwitchReg(0x0403);
  rtn = (unsigned short) vals[0];
  rtn += (((unsigned short) vals[1]) << 8);
  return rtn;
}

int configVIAEthernetSwitch(void)
{

  // Reset Switch (will PORESET dsps due to "strapping pins" shared)
  *pdk_bd_regs.u711xpfc.cpld_gcr |= 0x0200;
  pdk_waste_clks_10(100000);
  *pdk_bd_regs.u711xpfc.cpld_gcr &= ~0x0200;
  pdk_waste_clks_10(100000);

  // Confirm Communication with Switch
  // Check default MAC for Switch
  if (readSwitchReg(0x0619) != 0x00) { return 1; }
  if (readSwitchReg(0x061a) != 0x40) { return 1; }
  if (readSwitchReg(0x061b) != 0x63) { return 1; }
  if (readSwitchReg(0x061c) != 0x80) { return 1; }
  if (readSwitchReg(0x061d) != 0x00) { return 1; }
  if (readSwitchReg(0x061e) != 0x00) { return 1; }

  // port configuration:
  // 100Mbps, Full Duplex, Output and Input Enabled: 0x03
  // Phy addresses are 0 and 1
  writeSwitchReg(0x0418, 0x00);
  writeSwitchReg(0x0419, 0x01);
  writeSwitchReg(0x0420, 0x03);
  if (readSwitchReg(0x0419) != 0x01) { return 1; } // Check write success
  writeSwitchReg(0x0421, 0x03);
  writeSwitchReg(0x0422, 0x03);
  writeSwitchReg(0x0423, 0x03);
  writeSwitchReg(0x0424, 0x03);
  writeSwitchReg(0x0425, 0x03);
  writeSwitchReg(0x0426, 0x03);
  writeSwitchReg(0x0427, 0x03);
  writeSwitchReg(0x0428, 0x03);
  writeSwitchReg(0x0429, 0x03);
// PDK Phy Auto-Negotiation
```

```
        disableSwitchPhyAutoPolling();
        if (readSwitchPhy(0x0802) == 0x0013)
        {
          writeSwitchPhy(0x0800, 0x8000);
          while ((readSwitchPhy(0x0800) & 0x8000) != 0);
          writeSwitchPhy(0x0804, 0x0DE1); // Capabilities
          writeSwitchPhy(0x0800, 0x1000); // Enable Auto-Negotiation
          if (readSwitchPhy(0x0804) != 0x0DE1)
          { return 1; }
        }
        if (readSwitchPhy(0x0902) == 0x0013)
        {
          writeSwitchPhy(0x0900, 0x8000);
          while ((readSwitchPhy(0x0900) & 0x8000) != 0);
          writeSwitchPhy(0x0904, 0x0DE1); // Capabilities
          writeSwitchPhy(0x0900, 0x1000); // Enable Auto-Negotiation
          if (readSwitchPhy(0x0904) != 0x0DE1)
          { return 1; }
        }

        return 0;
      }
```

## 3.5 I²C Bus Devices

There are two I²C EEPROM devices on the SPT711XPFCE. One connects only to the VIA Ethernet switch, and the other connects to all four MSC711x devices. The I²C address for the EEPROM connected to the switch is 0b1010001x. The I²C address for the EEPROM connected to MSC711x devices is 0b1010000x. The EEPROM devices are part number M24256-BWMN6G from ST Microelectronics. For more information, refer to the manufacturer website (`http://www.st.com`).

Example 2 shows how to initialize the interface, write a test pattern to the EERPOM at address 0b1010000x, read the test pattern back, and then erase the EEPROM. To use this code to test the I²C interface, simply write the test pattern using one of the four MSC711x devices and then read the test pattern back from all four devices. Finally, use one of the four devices to erase the test pattern.

**Example 2. I²C EEPROM Access Example Code for MSC711x Devices**

```
#define I2C_BUF_SIZE 1024
unsigned char I2C_BUFFER[I2C_BUF_SIZE];

/*****************************************************************************
   I2C_UP()
       Initialize I2C Interface - no interrupts, polling only, master only
       ARGS:
          d_addr      : 7-bit address of this device
       RTNS:
          0 = Success
          1 = Failure
*****************************************************************************/

int I2C_UP(unsigned char d_addr)
{
   gpio_map_t *pstGPIO = (gpio_map_t *)GPIO_BASE;
   i2c_map_t* I2C = (i2c_map_t*) I2C_BASE;
```

```
   // Configure Pins
   pstGPIO->gp[0].GP_CTL |=  0x0000C000;

   // Disable Control Register
   I2C->I2CR = 0x00;

   // Set slowest I2C bus speed possible
   I2C->IFDR = 0x3f;

   // Set address
   I2C->IADR = (d_addr << 1) | 0x08;

   // Enable Control Register
   I2C->I2CR = 0x80;

   return 0;
}

/*****************************************************************************

  I2C_STATUS()
   Check if I2C is initialized
   ARGS:
       (None.)
   RTNS:
       0 = Down
       1 = Up
*****************************************************************************/

int I2C_STATUS()
{
   i2c_map_t* I2C = (i2c_map_t*) I2C_BASE;

   if (I2C->I2CR & 0x80)
   { return 1; }

   return 0;
}

/*****************************************************************************

  I2C_RX()
       Receive number of bytes from the bus.
       ARGS:
          *data       : Data buffer for received
          cnt         : Number of bytes to read
       RTNS:
          Number of bytes received successfully.
*****************************************************************************/

int I2C_RX(unsigned char *data, int cnt)
{
   i2c_map_t* I2C = (i2c_map_t*) I2C_BASE;
   int x = 0;

   // Receive mode
   I2C->I2CR &= ~0x0010;

   // Set or clear Ack control before dummy read
   if (cnt > 1)
   { I2C->I2CR &= ~0x0008; }
   else
   { I2C->I2CR |= 0x0008; }
```

```
   // Read data (dummy read to start transfer)
   data[x] = (unsigned char) I2C->I2DR;
   cnt--;

   while (cnt > 0)
   {
       // Wait for transfer to start
       while (I2C->I2SR & 0x0080);

       // Wait for transfer to end
       while ((I2C->I2SR & 0x0080) == 0);

       // Set or clear Ack control before last read
       if (cnt > 1)
       { I2C->I2CR &= ~0x0008; }
       else
       { I2C->I2CR |= 0x0008; }

       // Read data
       data[x++] = (unsigned char) I2C->I2DR;
       cnt--;
   }

   // Wait for transfer to start
   while (I2C->I2SR & 0x0080);

   // Wait for transfer to end
   while ((I2C->I2SR & 0x0080) == 0);

   // Signal stop before last read
   I2C->I2CR &= ~0x0030;

   // Read Read for last data (post stop condition)
   data[x++] = (unsigned char) I2C->I2DR;

   return x;
}

/*****************************************************************************

  I2C_TX()
   Transmit byte on bus
   ARGS:
       data        : Data to send
   RTNS:
       0 = Success
       1 = Failure
*****************************************************************************/

int I2C_TX(unsigned char data)
{
   i2c_map_t* I2C = (i2c_map_t*) I2C_BASE;

   // Send byte
   I2C->I2CR |= 0x0010;          // Tx

   I2C->I2DR = (unsigned short) (data);

   // Wait for transfer to start
   while (I2C->I2SR & 0x0080)
   {
       if (I2C->I2SR & 0x0050)
```

```
        { return 1; }
    }

    // Wait for transfer to end
    while ((I2C->I2SR & 0x0080) == 0)
    {
        if (I2C->I2SR & 0x0050)
        { return 1; }

   }

    // Check for Ack
    if (I2C->I2SR & 0x0001)
    { return 1; }

    return 0;
}

/*****************************************************************************

I2C_WR()
    Write a byte to an I2C EEPROM Device.  This performs one byte write at a
    time to avoid page burst boundary requirements.
    ARGS:
        t_addr      : LSB 3-bit address of target device from base address 0xA0.
        b_addr      : Byte address in device
        *data       : Data to write
    RTNS:
        0 = Success
        1 = Failure
*****************************************************************************/

int I2C_WR(unsigned char t_addr, unsigned short b_addr, unsigned char *data)
{
    i2c_map_t* I2C = (i2c_map_t*) I2C_BASE;

    // Arbitrate for Bus
    while (I2C->I2SR & 0x0020);   // Bus Busy
    I2C->I2CR |= 0x0020;          // Master

    // Send device address
    if (I2C_TX(0xA0 | ((t_addr & 7) << 1)))
    { I2C->I2CR &= ~0x0030; return 1; }

    // Send byte address (MSB first)
    if (I2C_TX((unsigned char) (b_addr >> 8)))
    { I2C->I2CR &= ~0x0030; return 1; }
    if (I2C_TX((unsigned char) (b_addr & 0x00ff)))
    { I2C->I2CR &= ~0x0030; return 1; }

    // Send byte data
    if (I2C_TX(*data))
    { I2C->I2CR &= ~0x0030; return 1; }

    // Release Bus
    I2C->I2CR &= ~0x0030;         // Slave Rx

    return 0;
}

/*****************************************************************************
    I2C_RD()
```

```
       Read data from an I2C EEPROM Device
       ARGS:
          t_addr      : LSB 3-bit address of target device from base address 0xA0.
          b_addr      : Byte address in device
          *data       : Data to write
          cnt         : number of bytes to read
       RTNS:
          Number of bytes read.
*****************************************************************************/

int I2C_RD(unsigned char t_addr, unsigned short b_addr, unsigned char *data,
          int cnt)
{
   i2c_map_t* I2C = (i2c_map_t*) I2C_BASE;
   int x;

   if (cnt < 0)
   { return 0; }

   // Arbitrate for Bus
   while (I2C->I2SR & 0x0020);   // Bus Busy
   I2C->I2CR |= 0x0020;          // Master

   // Send device address (Keep action a "Write" to set Random Access address)
   if (I2C_TX(0xA0 | ((t_addr & 7) << 1)))
   { I2C->I2CR &= ~0x0030; return 1; }

   // Send byte address (MSB first)
   if (I2C_TX((unsigned char) (b_addr >> 8)))
   { I2C->I2CR &= ~0x0030; return 1; }
   if (I2C_TX((unsigned char) (b_addr & 0x00ff)))
   { I2C->I2CR &= ~0x0030; return 1; }

   I2C->I2CR |= 0x0004;          // Restart

   // Send device address (Now send "Read")
   if (I2C_TX(0xA1 | (t_addr << 1)))
   { I2C->I2CR &= ~0x0030; return 1; }

   // Receive a byte data (bus is released by subroutine)
   x = I2C_RX(data, cnt);

   return x;
}

/*****************************************************************************
   I2C_EEPROM_INIT()
       Write Test Vector Into I2C EEPROM
       ARGS:
          (None.)
       RTNS:
          0 = Success
          1 = Failure
*****************************************************************************/

int I2C_EEPROM_INIT()
{
   unsigned short x;
   unsigned char y;
   unsigned int retry;
```

```
    // Init first 1k of EEPROM with sequntial numbers (+5, will roll-over)
    y = 0;
    for (x = 0; x < 1024; x++)
    {
        retry = 0;
        while (I2C_WR(0, x, &y))
        {
            retry++;
            if (retry > 150)
            { return 1; }
        }
        y += 5;
    }
    return 0;
}

/*****************************************************************************

    I2C_EEPROM_CHECK()
        Check Test Vector In I2C EEPROM
        ARGS:
            (None.)
        RTNS:
            0 = Success
            1 = Failure
*****************************************************************************/

int I2C_EEPROM_CHECK()
{
    unsigned short x;
    unsigned char y;

    // Read data into buffer
    if (I2C_RD(0, 0, I2C_BUFFER, I2C_BUF_SIZE) != I2C_BUF_SIZE)
    { return 1; }

    // Check first part of EEPROM for sequntial numbers (+5, will roll-over)
    y = 0;

    for (x = 0; x < I2C_BUF_SIZE; x++)
    {
        if (I2C_BUFFER[x] != y)
        { return 1; }
        y += 5;
    }
    return 0;
}

/*****************************************************************************

    I2C_EEPROM_ERASE()
        ERASE I2C EEPROM
        ARGS:
            (None.)
        RTNS:
            0 = Success
            1 = Failure
*****************************************************************************/

int I2C_EEPROM_ERASE()
{
```

```
    unsigned short x;
    unsigned char y;
    unsigned int retry;

    // Init first 1k of EEPROM with 0xFF
    y = 0xff;
    for (x = 0; x < 1024; x++)
    {
        retry = 0;
        while (I2C_WR(0, x, &y))
        {
            retry++;
            if (retry > 150)
            { return 1; }
        }
    }
    return 0;
}
```

## 3.6 Voltage Regulators

Two linear regulator modules on the SPT711xPFCE generate the DDR SDRAM (2.5 VDC) and Ethernet switch core (2.5 V) voltages from the externally supplied 3.3 VDC. A switching regulator generates the MSC711x core (1.2 VDC) voltage from the externally-supplied 5 VDC. **Figure 9** shows how to probe the board voltages.

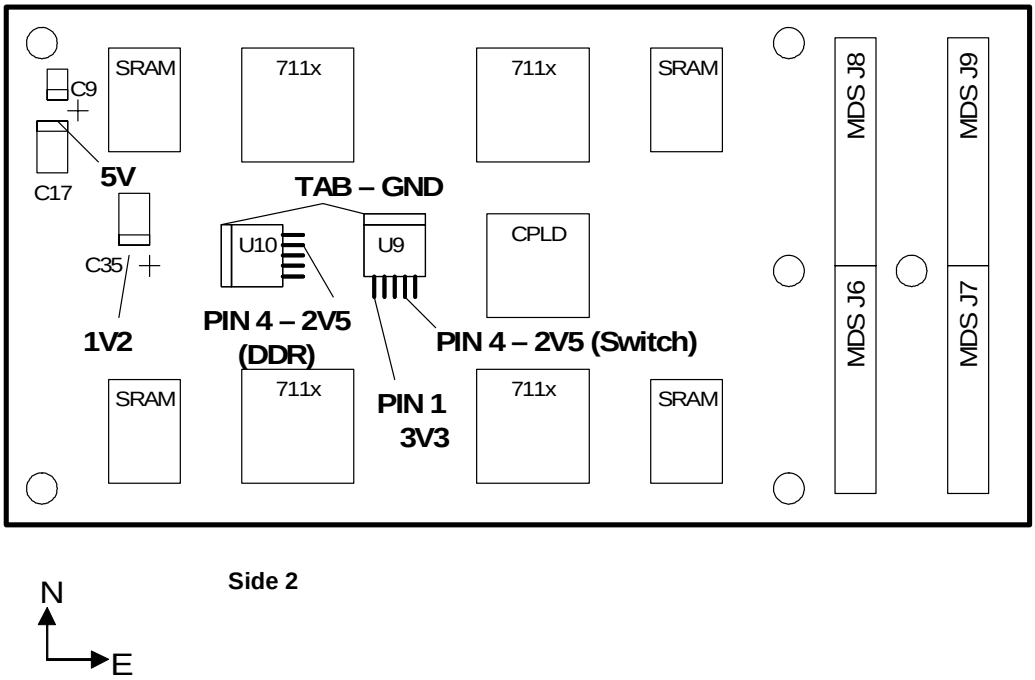



**Figure 9. Voltage Probing Points**

# 4 Board Configuration

Some SPT711xPFCE features are not enabled when the board is shipped from the factory. Carefully evaluate the features and their conflicts before enabling them.

## 4.1 Reset and Boot Mode Selection

The complex programmable logic device (CPLD) controls reset and boot mode selection. It contains the logic necessary to assert $\overline{\text{PORESET}}$ to all four MSC711x devices and individual $\overline{\text{HRESET}}$ to each MSC711x device. The host can also change the boot mode selection signals to all four MSC711x devices. By default, all devices reset to boot mode 0b0000 with SWTE and DBREQ deasserted. For details, see **Section 7**, *CPLD Registers*, on page 25.

The on-board reset circuitry has been minimized and handles only initial powerup and final powerdown. The circuitry requires a rise/fall time of at least 1 V per 25 ms for both external voltage sources. Also, the external 5 VDC supply must begin rising within 100 ms of the time when the external 3.3 VDC source reaches 2.90 VDC. The reset circuitry issues a $\overline{\text{PORESET}}$ when the external 3.3 VDC source drops below 2.95 VDC or when the 5 VDC source drops below 0.8 VDC. These requirements are reasonable for the board's intended use.

If the external 5 VDC supply is between 0.8 VDC and 3.0 VDC after 200 ms of the external 3.3 VDC reaches 2.95 VDC, the core voltage may "brown out" without triggering a $\overline{\text{PORESET}}$. The host processor must issue a $\overline{\text{PORESET}}$ through the CPLD registers to reset the cores.

## 4.2 Optional Resistors

Several optional features can be enabled or disabled by installing and/or removing resistors, as detailed in **Table 5**.

**Table 5. Optional Resistors**

| Resistor | Option |
|---|---|
| R181 (R179) | JTAG Chain Bypass CPLD |
| R2 (R1) | JTAG Chain Bypass SL0 |
| R8 (R5) | JTAG Chain Bypass SL1 |
| R285 (R54) | JTAG Chain Bypass SL2 |
| R40 (R41) | JTAG Chain Bypass SL3 |
| R121 (R78) | TPSEL = 0 SL0 |
| R124 (R15) | TPSEL = 0 SL1 |
| R56 (R55) | TPSEL = 0 SL2 |
| R51 (R52) | TPSEL = 0 SL3 |
| U6 (R148, R135) | SPI EEPROM SL0 |
| U3 (R150, R97) | SPI EEPROM SL1 |
| U14 (R221, R244) | SPI EEPROM SL2 |

**Table 5. Optional Resistors (continued)**

| Resistor | Option |
|---|---|
| U18 (R301, R295) | SPI EEPROM SL3 |
| R161, R26, R233, R236 | Via Switch Boot Mode |
| **Note:** Components in () should be removed when the others are installed, and *vice versa*. For details, see the schematics. | |

# 5 Board Specifications

**Figure 10** shows a detailed mechanical drawing of the SPT711xPFCE. These dimensions conform to the CMC standard length and width dimensions. Board height varies according to the components on the board and choice of CMC connector height. Estimated board height is 13.5 mm unless both sets of CMC connectors are populated. If CMC connectors are populated on both sides of the board, the estimated board height is 24 mm.

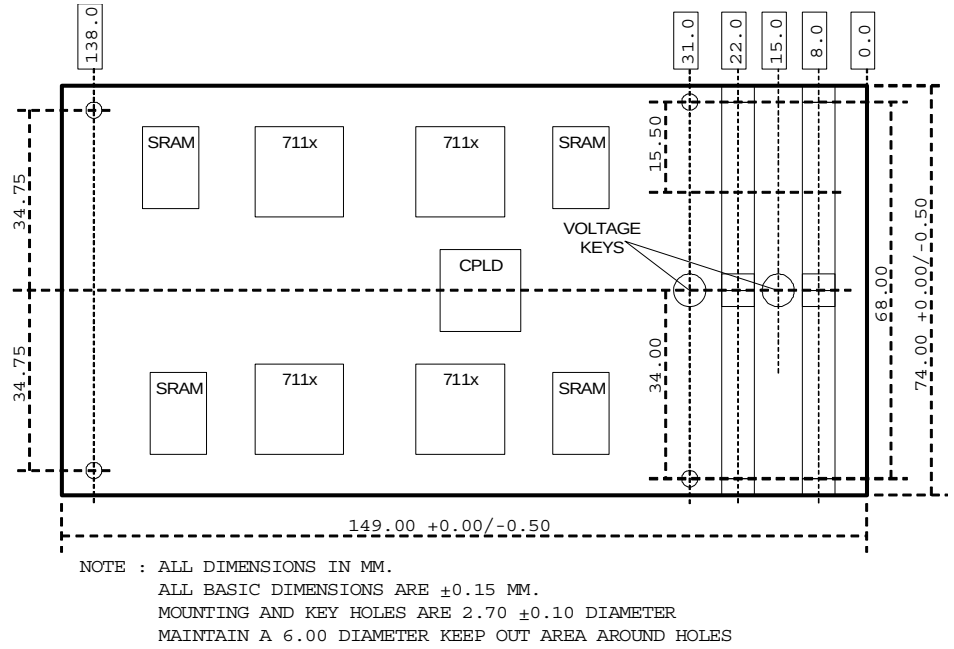



**Figure 10. SPT711xPFCE Board Mechanical Dimensions**

**Table 6** shows the results of SPT711xPFCE power measurements.

**Table 6. Power Consumption**

| Description | 3.3 VDC | 5.0 VDC | Units |
|---|---|---|---|
| Idle: | | | |
| Approximate current | 0.56 | 0.23 | Amps |
| Approximate power | 1.8 | 1.1 | Watts |
| Total approximate power | 3.0 | | Watts |

**Table 6. Power Consumption (continued)**

| Description | 3.3 VDC | 5.0 VDC | Units |
|---|---|---|---|
| One core DMA:<br>Approximate current<br>Approximate power<br>Total approximate power | <br>0.58<br>1.9<br>3.2 | <br>0.26<br>1.3 | <br>Amps<br>Watts<br>Watts |
| One core flood ping:<br>Approximate current<br>Approximate power<br>Total approximate power | <br>0.78<br>2.5<br>3.7 | <br>0.23<br>1.1 | <br>Amps<br>Watts<br>Watts |
| All cores DMA and ping:<br>Approximate current<br>Approximate power<br>Total approximate power | <br>1.0<br>3.4<br>5.3 | <br>0.39<br>1.9 | <br>Amps<br>Watts<br>Watts |
| **Note:** Power measurements were taken on a PDK base board using external power supplies. The test environment was room temperature (approximately 25× C). The values provided include a 10 percent guard band above the test results. The DMA test runs at approximately 80 Mbps (read and write transfers together) and includes a 1 Kbyte block 32-bit checksum calculation and check. The flood ping test is performed by rapidly sending ICMP echo requests from a host processor to each DSP core. The cores respond with ICMP echo replies. | | | |

# 6 CPLD Memory Map

The system memory map is determined by the CPLD on the SPT711xPFCE, as shown in **Table 7**.

**Table 7. SPT711xPFCE System Memory Map**

| Register | Offset | Description |
|---|---|---|
| **DSP1** | | First digital signal processor |
| ICR | 0x00000 | HDI16 interface control register |
| CVR | 0x00002 | HDI16 command vector register |
| ISR | 0x00004 | HDI16 interface status register |
| Tx/Rx[0–3] | 0x02000 | HDI16 transmit/receive data registers |
| **DSP2** | | Second digital signal processor |
| ICR | 0x08000 | HDI16 interface control register |
| CVR | 0x08002 | HDI16 command vector register |
| ISR | 0x08004 | HDI16 interface status register |
| Tx/Rx[0–3] | 0x0A000 | HDI16 transmit/receive data registers |
| **DSP3** | | Third digital signal processor |
| ICR | 0x10000 | HDI16 interface control register |
| CVR | 0x10002 | HDI16 command vector register |
| ISR | 0x10004 | HDI16 interface status register |
| Tx/Rx[0–3] | 0x12000 | HDI16 transmit/receive data registers |
| **DSP4** | | Fourth digital signal processor |
| ICR | 0x18000 | HDI16 interface control register |
| CVR | 0x18002 | HDI16 command vector register |

**Table 7. SPT711xPFCE System Memory Map (continued)**

| Register | Offset | Description |
|---|---|---|
| ISR | 0x18004 | HDI16 interface status register |
| Tx/Rx[0–3] | 0x1A000 | HDI16 transmit/receive data registers |
| **DSP broadcast** | | All four digital signal processors |
| ICR | 0x20000 | HDI16 interface control register |
| CVR | 0x20002 | HDI16 command vector register |
| ISR | 0x20004 | HDI16 interface status register |
| Tx/Rx[0–3] | 0x22000 | HDI16 transmit/receive data registers |
| **Ethernet switch** | | Ethernet switch control registers |
| PMDATA | 0x28000 | 16-bit packet or memory data |
| RDATA | 0x28002 | 8-bit register data |
| ADDRL | 0x28004 | Low byte of 16-bit address |
| ADDRH | 0x28006 | High byte of 16-bit address |
| **CPLD Registers** | | Internal 8-bit CPLD control registers |
| RCSR | 0x38000 | Reset control/status register |
| BMCR | 0x38002 | Boot mode control register |
| HSR | 0x38004 | HREQ status register |
| HER | 0x38006 | HREQ enable register |
| ISR | 0x3A000 | IRQ from DSP status register |
| IER | 0x3A002 | IRQ from DSP enable register |
| — | 0x3A004 | Reserved |
| FVR | 0x3A006 | Firmware version register |

# 7 CPLD Registers

The CPLD registers are implemented inside the CPLD firmware and control many functions of the board, including DSP reset and interrupt generation. The registers discussed in this section are as follows:

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | HRS3 | HRS2 | HRS1 | HRS0 | HRC3 | HRC2 | HRC1 | HRC0 |
| W | | | | | | | | |
| Reset: | — | — | — | — | 0 | 0 | 0 | 0 |

**Figure 7-11. Reset Control and Status Register (RCSR)**

**Table 7-8. RCSR Bit Descriptions**

| Bits | Name | Description | Settings |
|---|---|---|---|
| 7–4 | HRS[3–0] | $\overline{\text{HRESET}}$ status from DSP[3–0]. Indicates the state of the $\overline{\text{HRESET}}$ signal from each of the MSC711x devices. | 0 $\overline{\text{HRESET}}$ is asserted.<br>1 $\overline{\text{HRESET}}$ is deasserted. |
| 3–0 | HRC[3–0] | $\overline{\text{HRESET}}$ command to DSP[3–0]. Controls the value driven by the CPLD onto each $\overline{\text{HRESET}}$ signal to each MSC711x device. | 0 CPLD does not drive signal.<br>1 CPLD asserts signal. |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | LED | PRST | SWTE | DBRQ | BM3 | BM2 | BM1 | BM0 |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 7-12. Boot Mode Control Register (BMCR)**

**Table 7-9. BMCR Bit Descriptions**

| Bits | Name | Description | Settings |
|---|---|---|---|
| 7 | LED | LED on. Illuminates the LED on the SPT711xPFCE connected to the CPLD. | 0 CPLD LED is not on.<br>1 CPLD LED is on. |
| 6 | PRST | Power reset control and status. When read, this bit indicates the state of the $\overline{\text{PORESET}}$ signal to all four MSC711x devices. When written, this bit controls the value driven by the CPLD onto the $\overline{\text{PORESET}}$ signal. | Read:<br>0 $\overline{\text{PORESET}}$ is deasserted.<br>1 $\overline{\text{PORESET}}$ is asserted.<br><br>Write:<br>0 CPLD does not drive signal.<br>1 CPLD asserts signal. |
| 5 | SWTE | SWTE control. Controls the value driven by the CPLD onto the SWTE signal to the MSC711x devices when $\overline{\text{PORESET}}$ is asserted. | 0 SWTE is logic low.<br>1 SWTE is logic high. |
| 4 | DBRQ | DBREQ control. Controls the value driven by the CPLD onto the DBREQ signal to the MSC711x devices when $\overline{\text{PORESET}}$ is asserted. | 0 DBREQ is logic low.<br>1 DBREQ is logic high. |
| 3–0 | BM[3–0] | BM[3–0] control. Controls the value driven by the CPLD onto the BM[3–0] signals to the MSC711x devices when $\overline{\text{PORESET}}$ is asserted. | |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | HRRQ3 | HRRQ2 | HRRQ1 | HRRQ0 | HTRQ3 | HTRQ2 | HTRQ1 | HTRQ0 |
| W | | | | | | | | |
| Reset: | — | — | — | — | — | — | — | — |

**Figure 7-13. HREQ Status Register (HSR)**

**Table 7-10. HSR Bit Descriptions**

| Bits | Name | Description | Settings |
|---|---|---|---|
| 7–4 | HRRQ[3–0] | HRRQ status from DSP[3–0]. Indicates the state of the HRRQ signal from each of the MSC711x devices. | 0 HRRQ is logic low.<br>1 HRRQ is logic high. |
| 3–0 | HTRQ[3–0] | HTRQ status from DSP[3–0]. Indicates the state of the HTRQ signal from each of the MSC711x devices. | 0 HTRQ is logic low.<br>1 HTRQ is logic high. |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | HRRE3 | HRRE2 | HRRE1 | HRRE0 | HTRE3 | HTRE2 | HTRE1 | HTRE0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 7-14. HREQ Enable Register (HER)**

**Table 7-11. HER Bit Descriptions**

| Bits | Name | Description | Settings |
|---|---|---|---|
| 7–4 | HRRE[3–0] | DSP[3–0] HRRQ enable. Determines whether an HRRQ signal from an MSC711x device is used to signal the global active low HRRQ signal to the host. | 0 HRRQ is not enabled.<br>1 HRRQ is enabled. |
| 3–0 | HTRE[3–0] | DSP[3–0] HTRQ enable. Determines whether an HTRQ signal from an MSC711x device is used to signal the global active low HTRQ signal to the host. | 0 HTRQ is not enabled.<br>1 HTRQ is enabled. |
| **Note:** The CPLD firmware is written to enable/disable the HRRQ and HTRQ signals properly when they are configured as active low out of the MSC711x devices. If only one device is enabled at a time, the signal polarity does not matter. | | | |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | IRQ3 | IRQ2 | IRQ1 | IRQ0 | — | — | — | — |
| W | | | | | | | | |
| Reset: | — | — | — | — | 0 | 0 | 0 | 0 |

**Figure 7-15. Interrupt from DSP Status Register (ISR)**

**Table 7-12. ISR Bit Descriptions**

| Bits | Name | Description | Settings |
|---|---|---|---|
| 7–4 | IRQ[3–0] | Interrupt request from DSP[3–0]. Shows the interrupt status from each MSC711x device. The Event2 pin from each MSC711x device signals interrupts to the host. | 0 Requesting interrupt.<br>1 Not requesting interrupt. |
| 3–0 | — | Reserved. Cleared to zero for future compatibility. | |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | IE3 | IE2 | IE1 | IE0 | — | — | — | — |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 7-16. Interrupt from DSP Enable Register (IER)**

**Table 7-13. IER Bit Descriptions**

| Bits | Name | Description | Settings |
|---|---|---|---|
| 7–4 | IE[3–0] | Interrupt request from DSP[3–0] enable. Determines whether an IRQ signal from an MSC711x device is used to signal the global active low $\overline{\text{IRQ}}$ signal to the host. | 0 Request is disabled.<br>1 Request is enabled. |
| 3–0 | — | Reserved. Cleared to zero for future compatibility. | |
| **Note:** The CPLD firmware is written to enable/disable the IRQ signals properly when they are configured as active low out of the MSC711x devices. If only one device is enabled at a time, the signal polarity does not matter. | | | |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | BV1 | BV0 | MV2 | MV1 | MV0 | mV2 | mV1 | mV0 |
| W | | | | | | | | |
| Reset: | — | — | — | — | — | — | — | — |

**Figure 7-17. Firmware Version Register (FVR)**

**Table 7-14. FVR Bit Descriptions**

| Bits | Name | Description | Settings |
|---|---|---|---|
| 7–6 | BV[1–0] | Board version. Indicates the board revision level as programmed into the CPLD firmware. | |
| 5–3 | MV[2–0] | Major version. Indicates the firmware major revision level. | |
| 2–0 | mV[2–0] | Minor version. Indicates the firmware minor revision level. | |

# Appendix A CPLD Firmware

```
/*   Copyright 2005, All Rights Reserved, Freescale Semiconductor, Inc.    */
/*************************************************************************
  PROJECT   : MSC711xPFC Revision 2 CPLD (Xilinx XC9572XL-10TQG100C)
  MODULE    : TOPLEVEL

  NOTES     : All bus control signals are active-low (CS, DS, RW, etc.).

  AUTHOR    : David M. Koltak  10/17/2005

VER   DATE        AUTHOR            -DESCRIPTION
-------------------------------------------------------------------------
0.1   10/17/2005  David M. Koltak   -Initial Design
1.0   10/18/2005  David M. Koltak   -Compiles without warnings or errors
1.1   10/19/2005  David M. Koltak   -IDMA Mode, BM Reset I2C, Clock Loop
1.2   10/19/2005  David M. Koltak   -Changed BM0 so default=0, output=1
1.3   11/07/2005  David M. Koltak   -Changed default BM back to 0x0
                                    -Changed clock assignments
2.0   01/30/2006  David M. Koltak   -Changed HDREQ equations
                                    -Added Firmware Version Register
                                    -Removed IDMA Mode
3.0   04/19/2006  David M. Koltak   -Updated for production boards (V2.7)
*************************************************************************/

‘define BOARD_VERSION 2’d1
‘define MAJOR_VERSION 3’d3
‘define MINOR_VERSION 3’d0

module TOPLEVEL
(
  CLKIN,
  DSP_CLKIN,
  PORESET,

  GOOD_3V3,
  GOOD_5V,
  LED,

  EONCE_HRESET,
  DSP_HRESET,
  DSP_BM0,
  DSP_BM,
  DSP_SWTE,
  DSP_DBREQ,

  DSP_HTRQ,
  DSP_HRRQ,
  HDREQ,

  DSP_HA,
  HA_DSP,
```

```
  DSP_HD,
  HCS,
  HRW,
  HDS,

  DSP_HDS,
  DSP_HRW,
  DSP_BHCS,
  DSP_HCS,
  SWCH_HCS,

  MDS_IRQ_OUT,
  PDK_IRQ_OUT,
  SWCH_IRQ,
  TP
);

  // External Pin Definitions
  input                 CLKIN;
  output    [3:0]       DSP_CLKIN;
  output                PORESET;
  input                 GOOD_3V3;
  input                 GOOD_5V;
  output                LED;

  input                 EONCE_HRESET;
  inout     [3:0]       DSP_HRESET;
  output    [3:0]       DSP_BM0;
  output    [3:1]       DSP_BM;
  output                DSP_SWTE;
  output                DSP_DBREQ;

  input     [3:0]       DSP_HTRQ;
  input     [3:0]       DSP_HRRQ;
  output    [2:1]       HDREQ;

  input     [2:0]       DSP_HA;
  input     [0:2]       HA_DSP;
  inout     [7:0]       DSP_HD;
  input                 HCS;
  input                 HRW;
  input                 HDS;
  output                DSP_HDS;
  output                DSP_HRW;
  output                DSP_BHCS;
  output    [3:0]       DSP_HCS;
  output                SWCH_HCS;

  output                MDS_IRQ_OUT;
  output                PDK_IRQ_OUT;
  output                SWCH_IRQ;
  output    [1:0]       TP;
```

```
// Internal Module Signals/Registers
reg                   m_int_cs;
reg                   m_swch_hcs;
reg                   m_dsp_bhcs;
reg       [3:0]       m_dsp_hcs;
wire                  m_int_we;
wire                  m_int_oe;
reg                   m_dsp_hds;
reg                   m_dsp_hrw;

reg                   m_led_out;
reg                   m_poreset_out;
wire                  m_poreset;
reg       [3:0]       m_hreset_out;
wire      [3:0]       m_hreset;
reg                   m_swte_out;
reg                   m_dbreq_out;
reg       [3:0]       m_bm_out;
wire      [3:0]       m_bm0;
reg       [7:0]       m_hreq_en;
reg       [3:0]       m_dsp_irq_en;
wire                  m_dsp_irq;
wire      [3:0]       m_dsp_irq_in;
reg       [7:0]       m_data_out;

// Assign Internal to/from External Signals
assign DSP_CLKIN[0] = CLKIN;
assign DSP_CLKIN[1] = !CLKIN;
assign DSP_CLKIN[2] = CLKIN;
assign DSP_CLKIN[3] = !CLKIN;

assign TP[1:0] = 2'b00;

assign LED = m_led_out;

assign m_poreset = (GOOD_3V3 | (!GOOD_5V) | m_poreset_out) ? 1'b0 : 1'b1;
assign PORESET = m_poreset;
assign m_bm0[0] = (m_poreset) ? 1'bz : m_bm_out[0];
assign m_bm0[1] = (m_poreset) ? 1'bz : m_bm_out[0];
assign m_bm0[2] = (m_poreset) ? 1'bz : m_bm_out[0];
assign m_bm0[3] = (m_poreset) ? 1'bz : m_bm_out[0];
assign DSP_BM0[0] = m_bm0[0];
assign DSP_BM0[1] = m_bm0[1];
assign DSP_BM0[2] = m_bm0[2];
assign DSP_BM0[3] = m_bm0[3];
assign DSP_BM[1] = m_bm_out[1];
assign DSP_BM[2] = m_bm_out[2];
assign DSP_BM[3] = m_bm_out[3];
assign DSP_SWTE = m_swte_out;
assign DSP_DBREQ = m_dbreq_out;

assign m_hreset[0] = (m_hreset_out[0] | (!EONCE_HRESET)) ? 1'b0 : 1'bz;
assign m_hreset[1] = (m_hreset_out[1] | (!EONCE_HRESET)) ? 1'b0 : 1'bz;
```

```
assign m_hreset[2] = (m_hreset_out[2] | (!EONCE_HRESET)) ? 1'b0 : 1'bz;
assign m_hreset[3] = (m_hreset_out[3] | (!EONCE_HRESET)) ? 1'b0 : 1'bz;
assign DSP_HRESET = m_hreset;

assign HDREQ[2] = (DSP_HRRQ[3] | (!m_hreq_en[7])) &
                  (DSP_HRRQ[2] | (!m_hreq_en[6])) &
                  (DSP_HRRQ[1] | (!m_hreq_en[5])) &
                  (DSP_HRRQ[0] | (!m_hreq_en[4]));

assign HDREQ[1] = (DSP_HTRQ[3] | (!m_hreq_en[3])) &
                  (DSP_HTRQ[2] | (!m_hreq_en[2])) &
                  (DSP_HTRQ[1] | (!m_hreq_en[1])) &
                  (DSP_HTRQ[0] | (!m_hreq_en[0]));

assign m_dsp_irq_in[3:0] = ~(DSP_BM0[3:0]); // Bitwise Not
assign m_dsp_irq = (m_dsp_irq_in[3] | (!m_dsp_irq_en[3])) &
                        (m_dsp_irq_in[2] | (!m_dsp_irq_en[2])) &
                        (m_dsp_irq_in[1] | (!m_dsp_irq_en[1])) &
                        (m_dsp_irq_in[0] | (!m_dsp_irq_en[0]));

assign MDS_IRQ_OUT = (m_dsp_irq) ? 1'b0 : 1'bz;
assign PDK_IRQ_OUT = (m_dsp_irq) ? 1'b0 : 1'bz;
assign SWCH_IRQ = 1'b1;

// DSP/Switch Bus Control (Single or Dual Strobe)
assign DSP_HDS = m_dsp_hds;
assign DSP_HRW = m_dsp_hrw;
always @ *
begin
  if (!m_swch_hcs)  // Dual Strobe for Switch
  begin
    m_dsp_hds = HDS | HRW;     // #IOW
    m_dsp_hrw = HDS | (!HRW);  // #IOR
  end
  else             // Single Strobe for DSPs
  begin
    m_dsp_hds = HDS;
    m_dsp_hrw = HRW;
  end
end

// Chip Select Logic
// 0-3 : DSP 0-3
// 4   : DSP Broadcast
// 5   : Ethernet Switch
// 7   : CPLD Registers
assign SWCH_HCS = m_swch_hcs;
assign DSP_BHCS = m_dsp_bhcs;
assign DSP_HCS = m_dsp_hcs;
always @ *
begin
  case (HA_DSP)
  3'd0: {m_int_cs, m_swch_hcs, m_dsp_bhcs, m_dsp_hcs} <= {6'b111111, HCS};
```

```
    3'd1: {m_int_cs, m_swch_hcs, m_dsp_bhcs, m_dsp_hcs} <= {5'b11111, HCS, 1'b1};
    3'd2: {m_int_cs, m_swch_hcs, m_dsp_bhcs, m_dsp_hcs} <= {4'b1111, HCS, 2'b11};
    3'd3: {m_int_cs, m_swch_hcs, m_dsp_bhcs, m_dsp_hcs} <= {3'b111, HCS, 3'b111};
    3'd4: {m_int_cs, m_swch_hcs, m_dsp_bhcs, m_dsp_hcs} <= {2'b11, HCS, 4'b1111};
    3'd5: {m_int_cs, m_swch_hcs, m_dsp_bhcs, m_dsp_hcs} <= {1'b1, HCS, 5'b11111};
    // No 6, default.
    3'd7: {m_int_cs, m_swch_hcs, m_dsp_bhcs, m_dsp_hcs} <= {HCS, 6'b111111};
    default: {m_int_cs, m_swch_hcs, m_dsp_bhcs, m_dsp_hcs} <= 7'b1111111;
    endcase
  end

  // Internal Registers (Write Logic)
  assign m_int_we = (m_int_cs | HDS | HRW);
  always @ (negedge m_int_we or negedge GOOD_3V3)
  begin
    if (!GOOD_3V3)
    begin
      m_led_out <= 1'b0;
      m_poreset_out <= 1'd0;
      m_hreset_out <= 4'd0;
      m_swte_out <= 1'd0;
      m_dbreq_out <= 1'd0;
      m_bm_out <= 4'd0;
      m_hreq_en <= 8'd0;
      m_dsp_irq_en <= 4'd0;
    end
    else
    begin
      case (DSP_HA)
      // RESET CONTROL/STATUS REGISTER
      3'd0: {m_hreset_out} <= DSP_HD[3:0];
      // BOOT MODE CONTROL REGISTER
      3'd1: {m_led_out, m_poreset_out, m_swte_out,
                                                m_dbreq_out, m_bm_out} <=
{DSP_HD[7:0]};
      // HREQ ENABLE REGISTER
      3'd3: m_hreq_en <= DSP_HD[7:0];
      // IRQ FROM DSP ENABLE REGISTER
      3'd5: m_dsp_irq_en <= DSP_HD[7:4];
      default: ;
      endcase
    end
  end

  // Internal Registers and Status (Read Logic)
  assign DSP_HD = m_data_out;
  assign m_int_oe = (m_int_cs | HDS | (!HRW));
  always @ *
  begin
    if (m_int_oe)
    begin
      m_data_out <= 8'hzz;
    end
```

```
    else
    begin
      case (DSP_HA)
      // RESET CONTROL/STATUS REGISTER
      3'd0: m_data_out <= {DSP_HRESET, m_hreset_out};
      // BOOT MODE CONTROL REGISTER
      3'd1: m_data_out <= {m_led_out, (!PORESET), m_swte_out,
                                          m_dbreq_out, m_bm_out[3:0]};
      // HREQ STATUS REGISTER
      3'd2: m_data_out <= {DSP_HRRQ, DSP_HTRQ};
      // HREQ ENABLE REGISTER
      3'd3: m_data_out <= m_hreq_en;
      // IRQ FROM DSP STATUS REGISTER
      3'd4: m_data_out <= {m_dsp_irq_in, 4'd0};
      // IRQ FROM DSP ENABLE REGISTER
      3'd5: m_data_out <= {m_dsp_irq_en, 4'd0};
      // FIRMWARE VERSION REGISTER
      3'd7: m_data_out <= {`BOARD_VERSION, `MAJOR_VERSION, `MINOR_VERSION};
      default: m_data_out <= 8'd0;
      endcase
    end
  end

endmodule
```

# Appendix B SPT711xPFCE Schematics

# MSC711xPFC
# Revision 2.x

The MSC711xPFC is a Packet-Telephony Farm card based on the MSC711x family of devices from Freescale Semiconductor. It combines four MSC7116 devices sharing an HDI16 bus and aggregated with an on board Ethernet switch.

PAGES

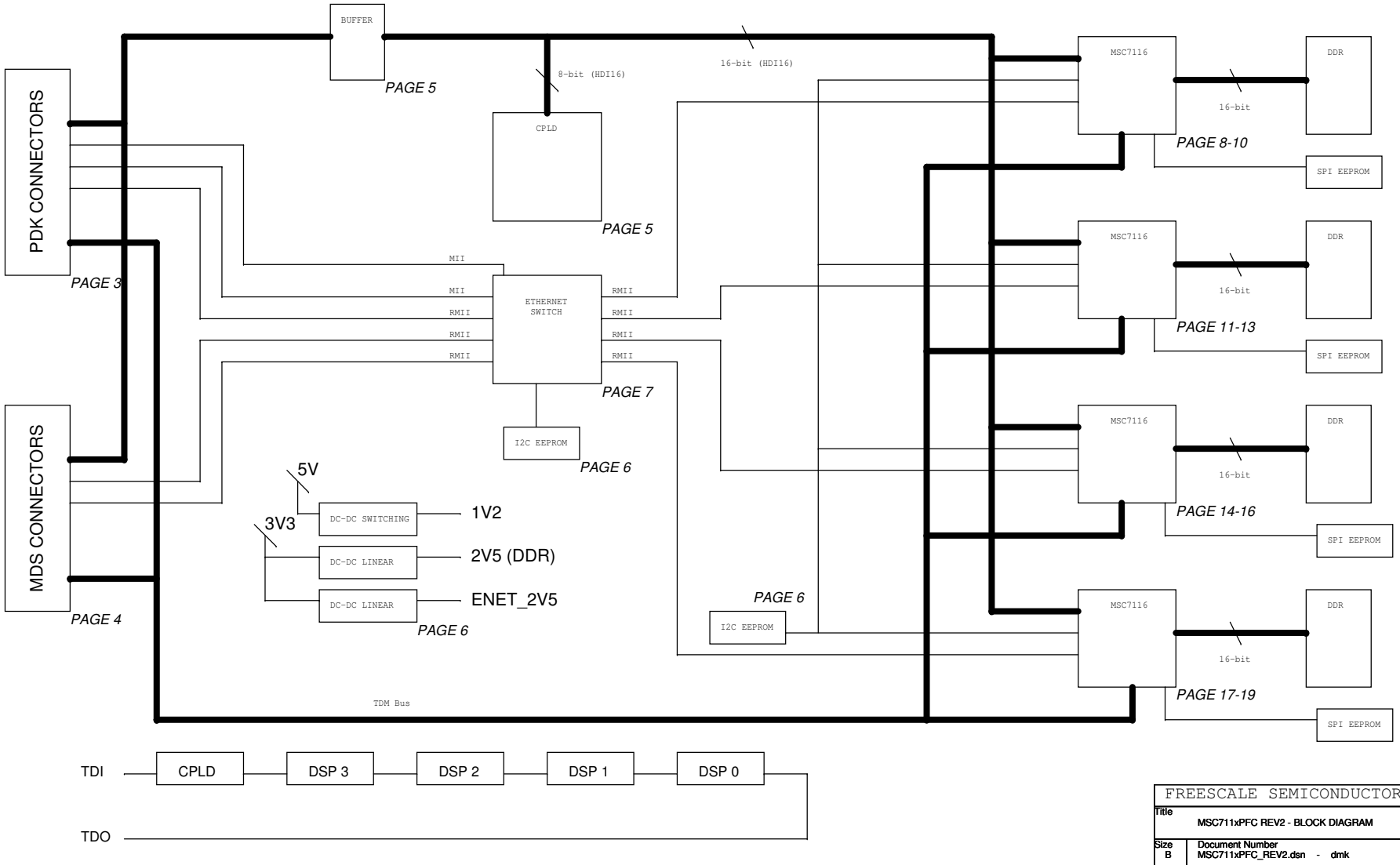


VERSION HISTORY

| VER | DATE | AUTHOR | -COMMENTS |
|---|---|---|---|
| 2.0 | 07/25/2005 | David M. Koltak | -Initial Design |
| 2.1 | 08/05/2005 | David M. Koltak | -Zero-Delay Buffer |
| 2.2 | 08/12/2005 | David M. Koltak | -Enet2V5 |
| 2.3 | 08/18/2005 | David M. Koltak | -J10 Part Changed & DNPs |
| 2.4 | 09/13/2005 | David M. Koltak | -Enet Boot Strap, TPs, nTRST P/D, PMC Molex |
| 2.5 | 09/19/2005 | David M, Koltak | -Agile 0.1uF Caps, Ref's |
| ---------- | PROTOTYPES | ---------- | |
| 2.6 | 11/14/2005 | David M. Koltak | -Reset, MII, LED, J10 |
| 2.7 | 01/27/2006 | David M. Koltak | -I2C Pull Ups |

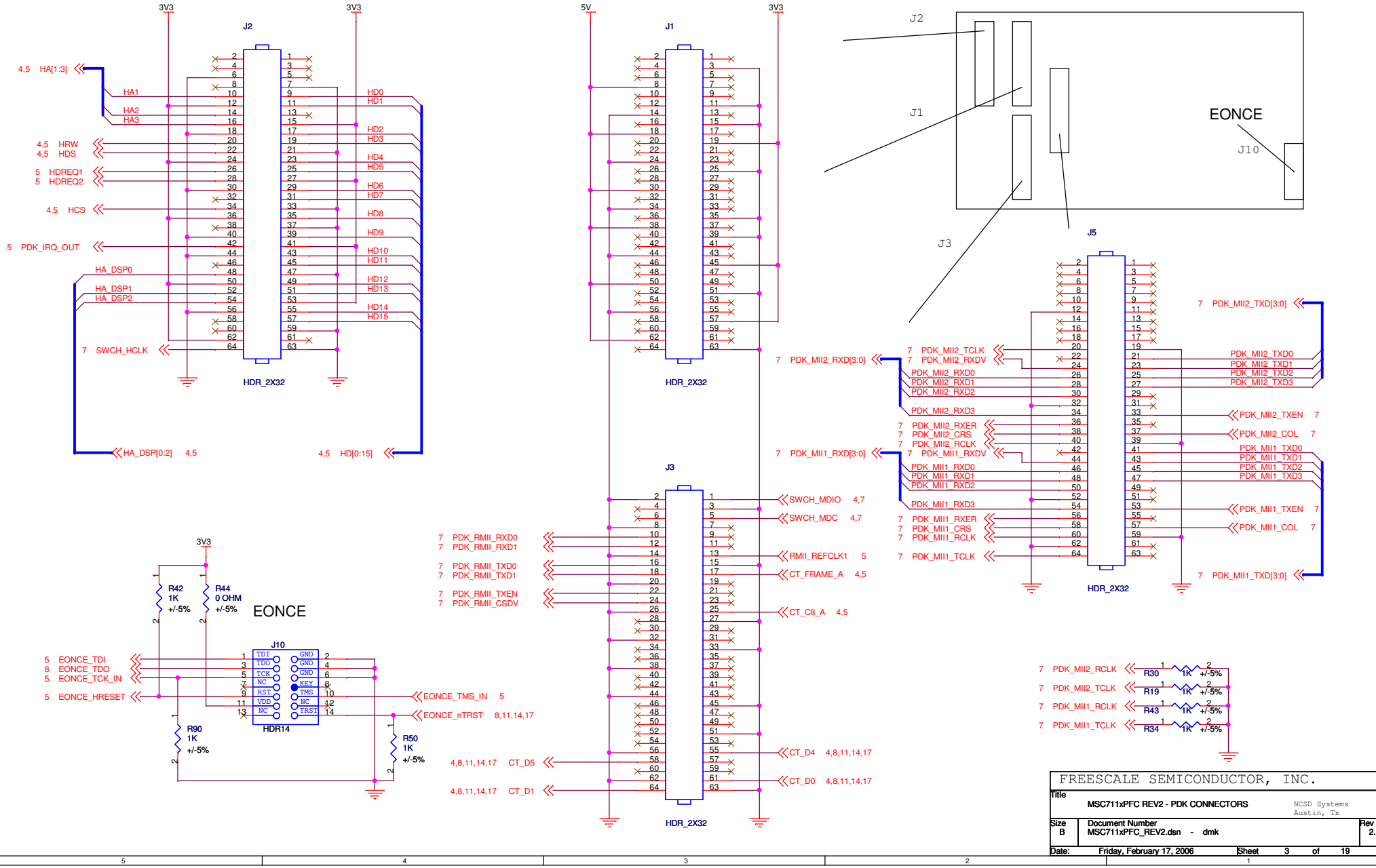
FREESCALE SEMICONDUCTOR, INC.

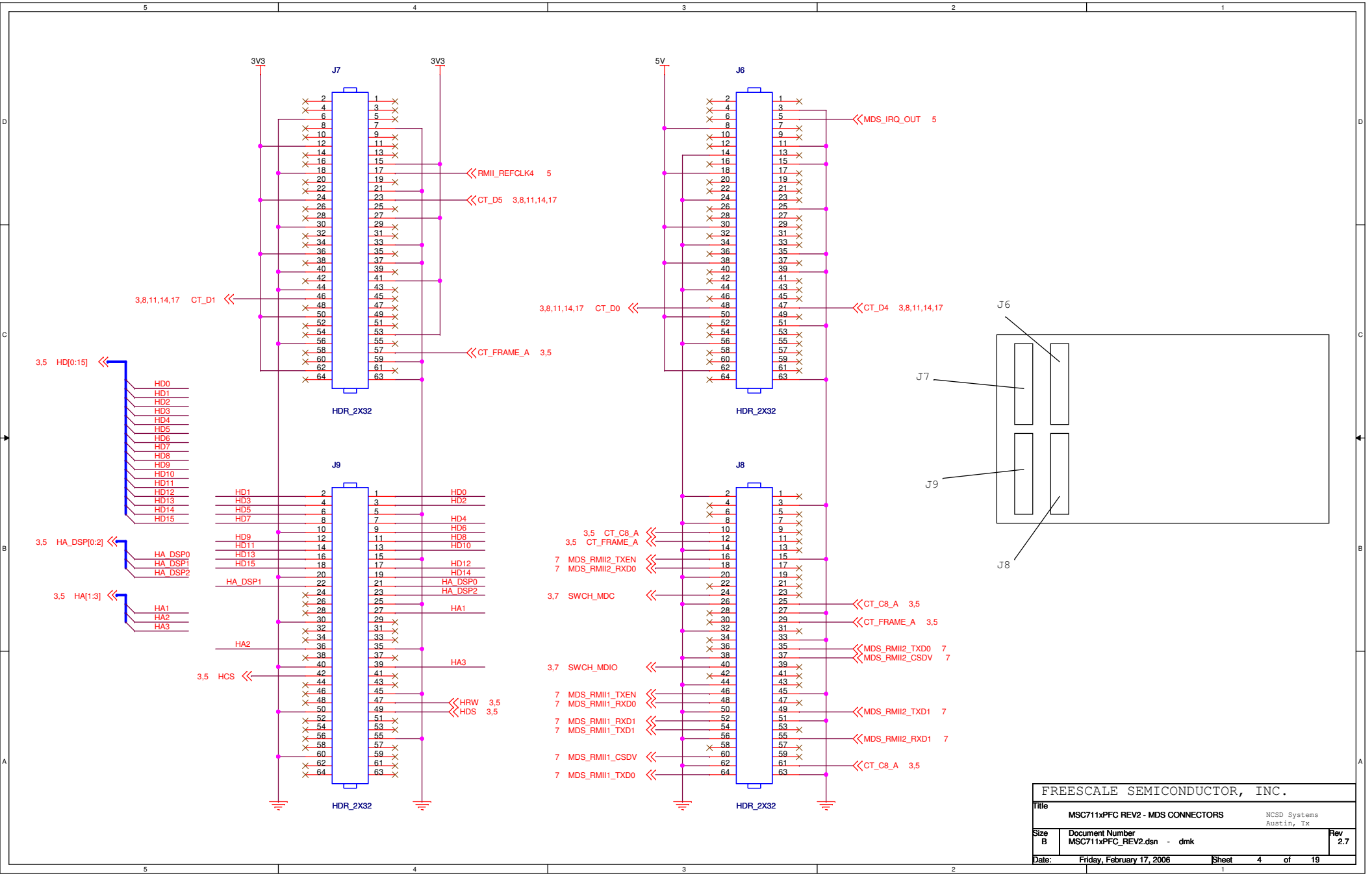
| | | |
|---|---|---|
| Title: MSC711xPFC REV2 - TITLE | NCSD Systems Austin, Tx | |
| Size: B | Document Number: MSC711xPFC_REV2.dsn - dmk | Rev: 2.7 |
| Date: Friday, February 17, 2006 | Sheet 1 of 19 | |




FREESCALE SEMICONDUCTOR, INC.

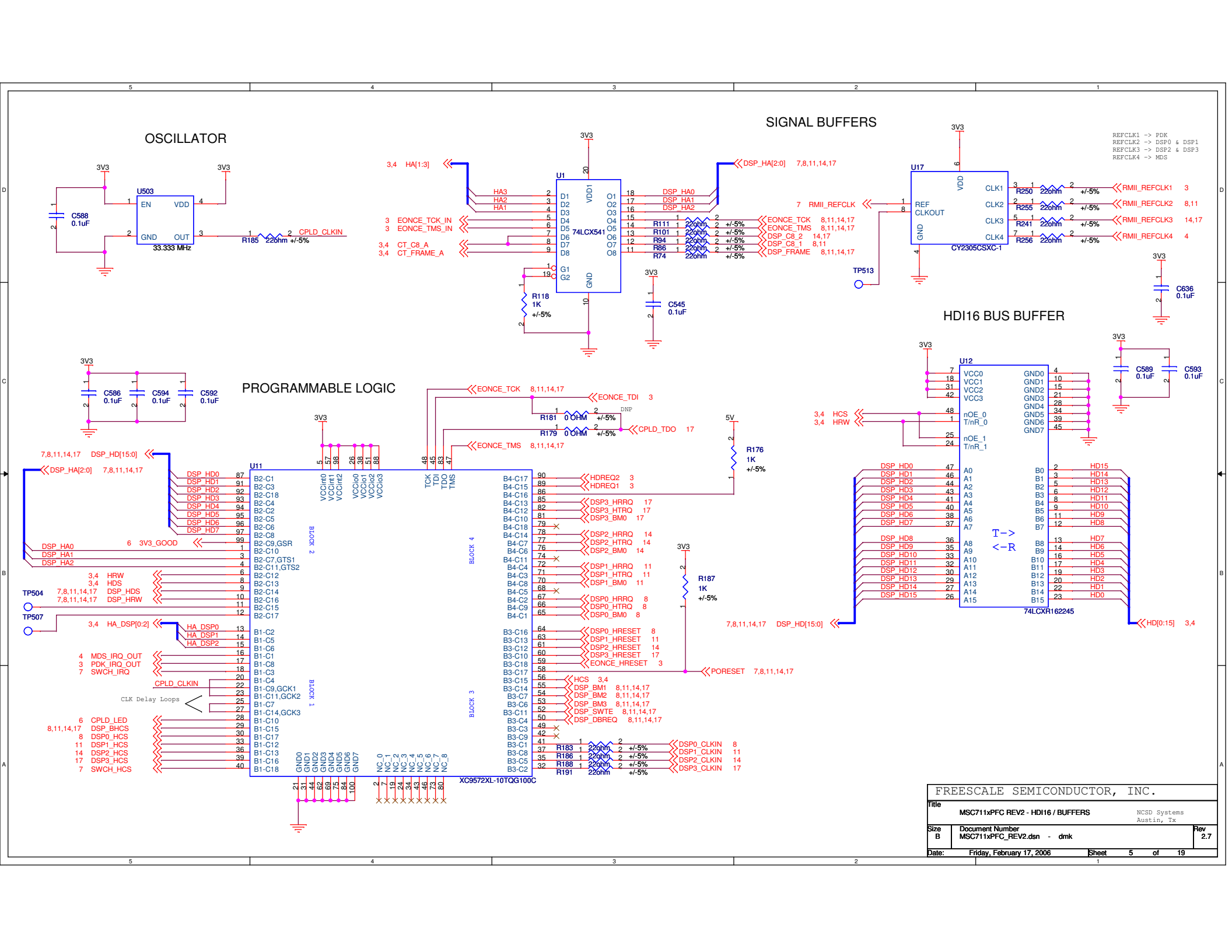
| | | |
|---|---|---|
| Title: MSC711xPFC REV2 - BLOCK DIAGRAM | | NCSD Systems Austin, Tx |
| Size: B | Document Number: MSC711xPFC_REV2.dsn - dmk | Rev: 2.7 |
| Date: Friday, February 17, 2006 | Sheet 2 of 19 | |


J2
3V3
3V3
4,5 HA[1:3]
HA1
HA2
HA3
4,5 HRW
4,5 HDS
5 HDREQ1
5 HDREQ2
4,5 HCS
5 PDK_IRQ_OUT
HA_DSP0
HA_DSP1
HA_DSP2
7 SWCH_HCLK
HD0
HD1
HD2
HD3
HD4
HD5
HD6
HD7
HD8
HD9
HD10
HD11
HD12
HD13
HD14
HD15
HA_DSP[0:2] 4,5
4,5 HD[0:15]
HDR_2X32
J1
5V
3V3
HDR_2X32
J2
J1
J3
EONCE
J10
J5
7 PDK_MII2_TXD[3:0]
7 PDK_MII2_TCLK
7 PDK_MII2_RXDV
7 PDK_MII2_RXD[3:0]
PDK_MII2_RXD0
PDK_MII2_RXD1
PDK_MII2_RXD2
PDK_MII2_RXD3
7 PDK_MII2_RXER
7 PDK_MII2_CRS
7 PDK_MII2_RCLK
7 PDK_MII1_RXDV
7 PDK_MII1_RXD[3:0]
PDK_MII1_RXD0
PDK_MII1_RXD1
PDK_MII1_RXD2
PDK_MII1_RXD3
7 PDK_MII1_RXER
7 PDK_MII1_CRS
7 PDK_MII1_RCLK
7 PDK_MII1_TCLK
PDK_MII2_TXD0
PDK_MII2_TXD1
PDK_MII2_TXD2
PDK_MII2_TXD3
PDK_MII2_TXEN 7
PDK_MII2_COL 7
PDK_MII1_TXD0
PDK_MII1_TXD1
PDK_MII1_TXD2
PDK_MII1_TXD3
PDK_MII1_TXEN 7
PDK_MII1_COL 7
7 PDK_MII1_TXD[3:0]
HDR_2X32
7 PDK_RMII_RXD0
7 PDK_RMII_RXD1
7 PDK_RMII_TXD0
7 PDK_RMII_TXD1
7 PDK_RMII_TXEN
7 PDK_RMII_CSDV
SWCH_MDIO 4,7
SWCH_MDC 4,7
RMII_REFCLK1 5
CT_FRAME_A 4,5
CT_C8_A 4,5
CT_D4 4,8,11,14,17
CT_D0 4,8,11,14,17
4,8,11,14,17 CT_D5
4,8,11,14,17 CT_D1
HDR_2X32
3V3
R42
1K
+/-5%
R44
0 OHM
+/-5%
EONCE
J10
5 EONCE_TDI
8 EONCE_TDO
5 EONCE_TCK_IN
5 EONCE_HRESET
TDI
TDO
TCK
NC
RST
VDD
NC
GND
GND
GND
KEY
TMS
NC
TRST
EONCE_TMS_IN 5
EONCE_nTRST 8,11,14,17
HDR14
R90
1K
+/-5%
R50
1K
+/-5%
7 PDK_MII2_RCLK
7 PDK_MII2_TCLK
7 PDK_MII1_RCLK
7 PDK_MII1_TCLK
R30
R19
R43
R34
1K
+/-5%
FREESCALE SEMICONDUCTOR, INC.
Title
MSC711xPFC REV2 - PDK CONNECTORS
NCSD Systems
Austin, Tx
Size
B
Document Number
MSC711xPFC_REV2.dsn - dmk
Rev
2.7
Date: Friday, February 17, 2006
Sheet 3 of 19

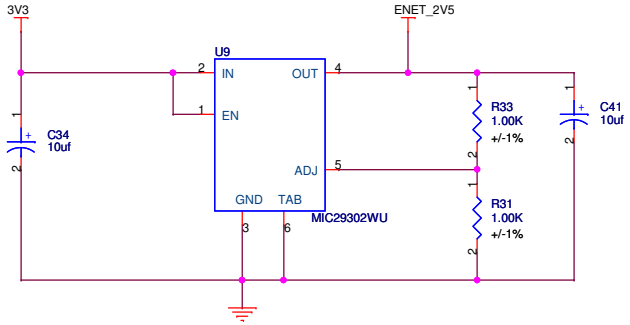



FREESCALE SEMICONDUCTOR, INC.

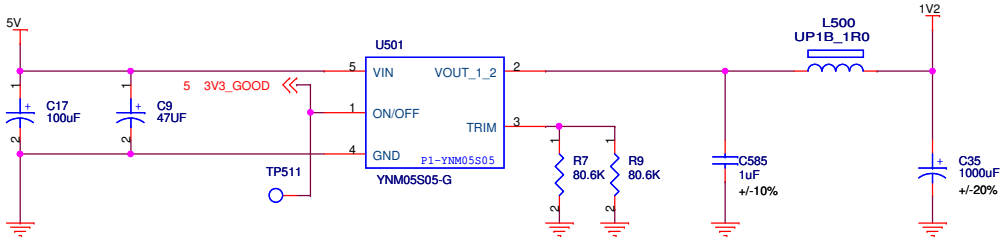
| | | |
|---|---|---|
| Title | MSC711xPFC REV2 - MDS CONNECTORS | NCSD Systems Austin, Tx |
| Size B | Document Number MSC711xPFC_REV2.dsn - dmk | Rev 2.7 |
| Date: Friday, February 17, 2006 | Sheet 4 of 19 | |


OSCILLATOR
SIGNAL BUFFERS
HDI16 BUS BUFFER
PROGRAMMABLE LOGIC
U503
33.333 MHz
U1
74LCX541
U17
CY2305CSXC-1
U12
74LCXR162245
U11
XC9572XL-10TQG100C
REFCLK1 -> PDK
REFCLK2 -> DSP0 & DSP1
REFCLK3 -> DSP2 & DSP3
REFCLK4 -> MDS
CLK Delay Loops
FREESCALE SEMICONDUCTOR, INC.
MSC711xPFC REV2 - HDI16 / BUFFERS
NCSD Systems
Austin, Tx
Document Number
MSC711xPFC_REV2.dsn - dmk
Rev 2.7
Date: Friday, February 17, 2006
Sheet 5 of 19

## ENET SWITCH 2.5 V POWER (3A)

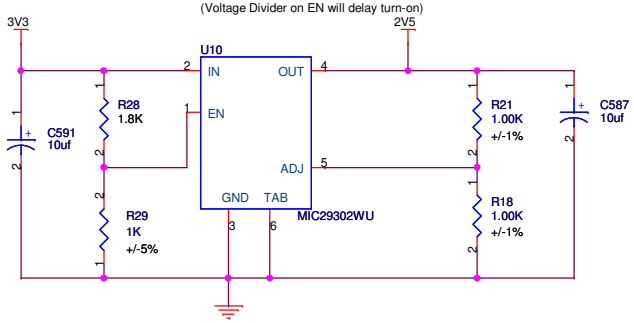



## MSC711x 1.2V CORE POWER (5A)



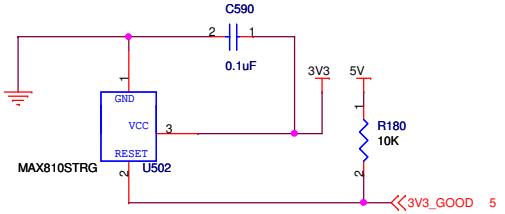

## DDR 2.5 V POWER (3A)

(Voltage Divider on EN will delay turn-on)

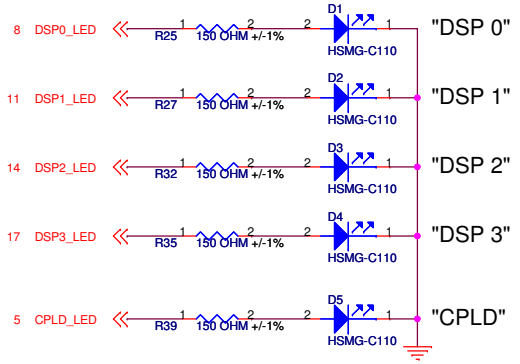



## 3V3 VOLTAGE DETECTOR

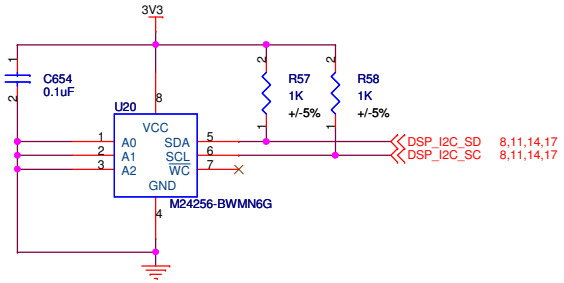
(Turn on 1V2 240ms after 3V3 reaches 2.93V)



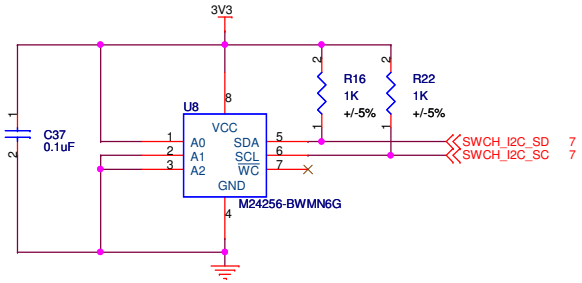

## INDICATORS

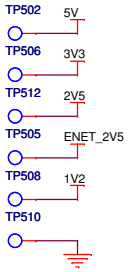



## DSP I2C BOOT EEPROM

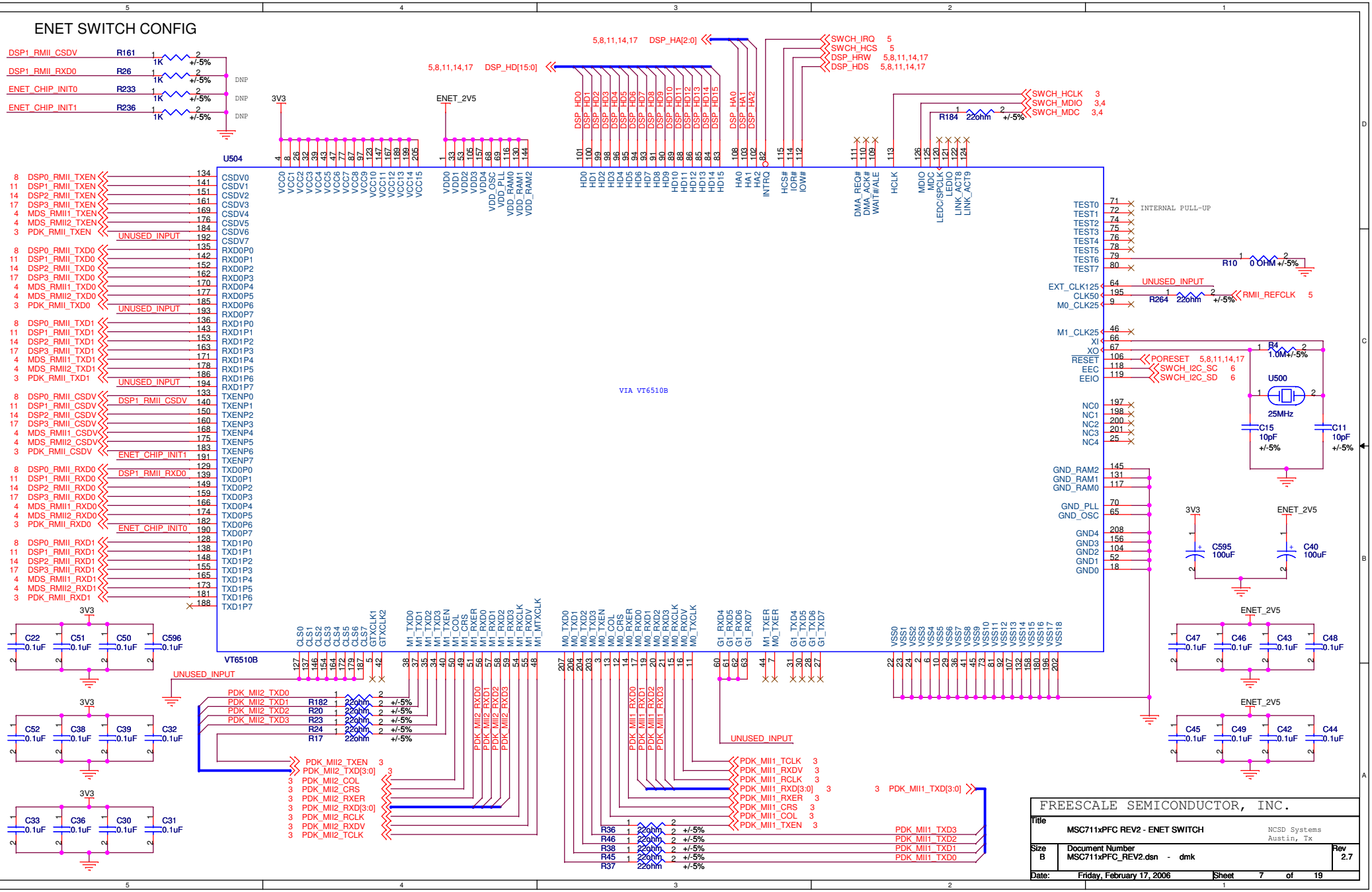


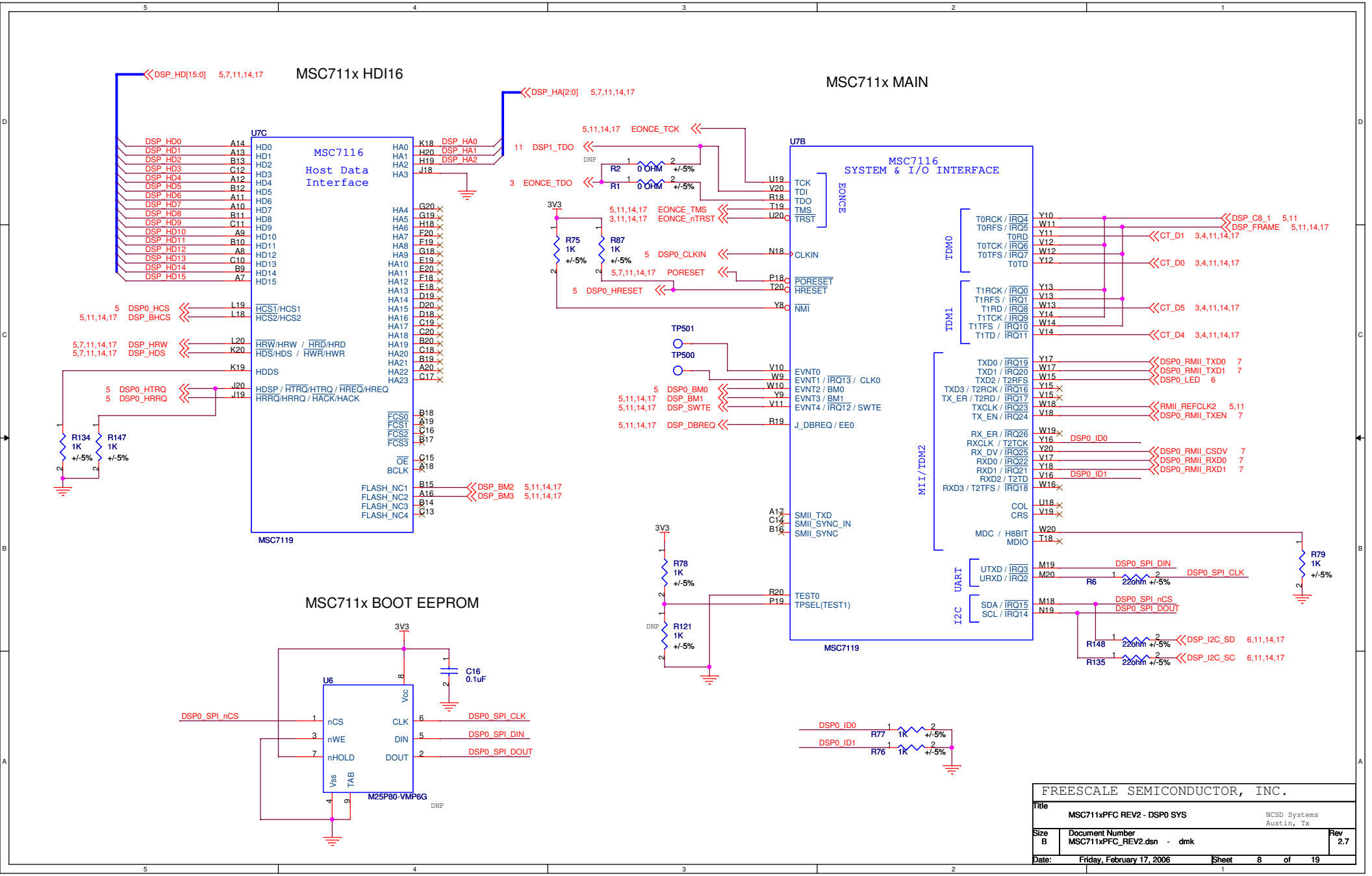

## ENET SWITCH BOOT EEPROM

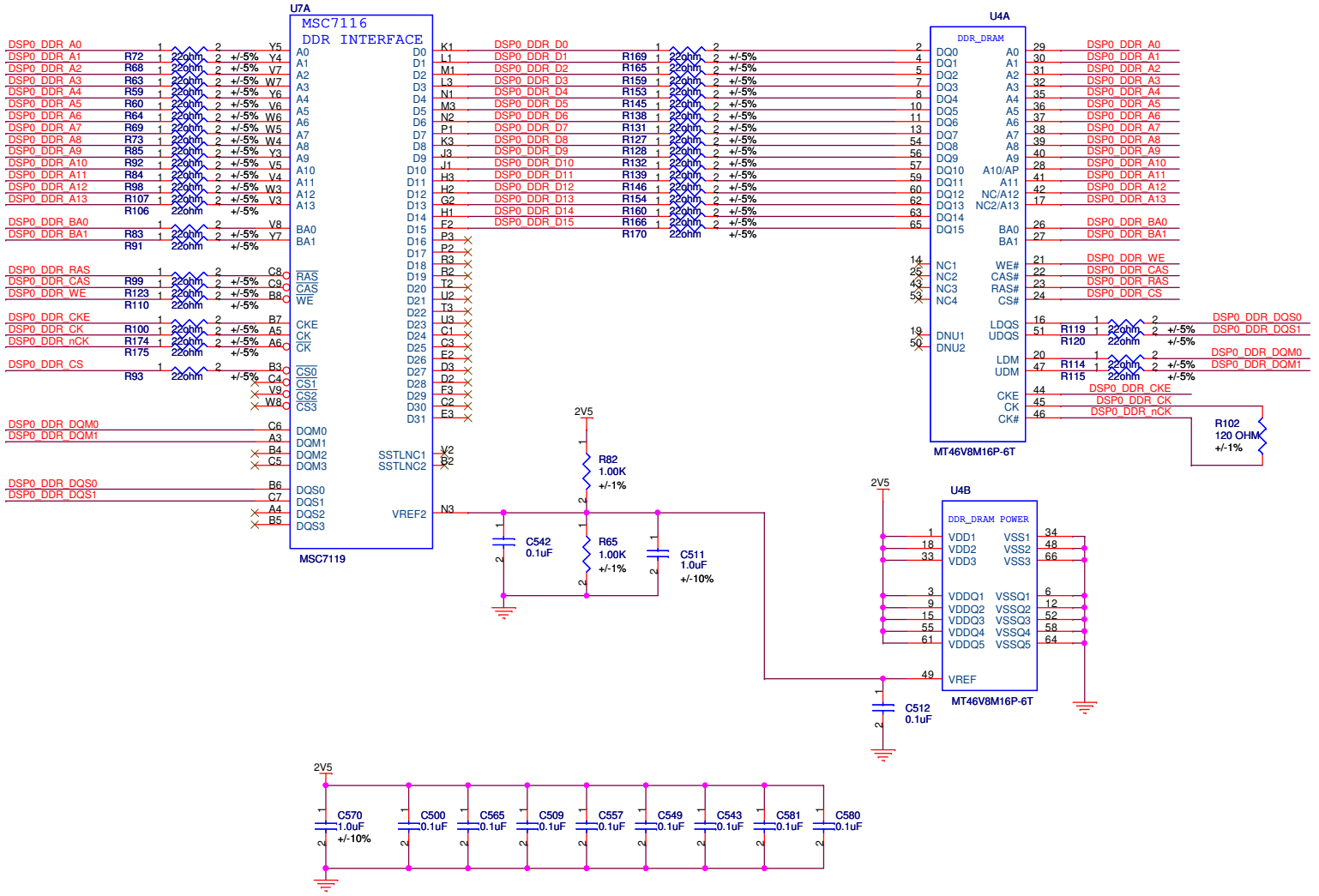



| | | |
|---|---|---|
| FREESCALE SEMICONDUCTOR, INC. | | |
| Title: MSC711xPFC REV2 - POWER / RESET | NCSD Systems Austin, Tx | |
| Size: B | Document Number: MSC711xPFC_REV2.dsn - dmk | Rev: 2.7 |
| Date: Friday, February 17, 2006 | Sheet 6 of 19 | |

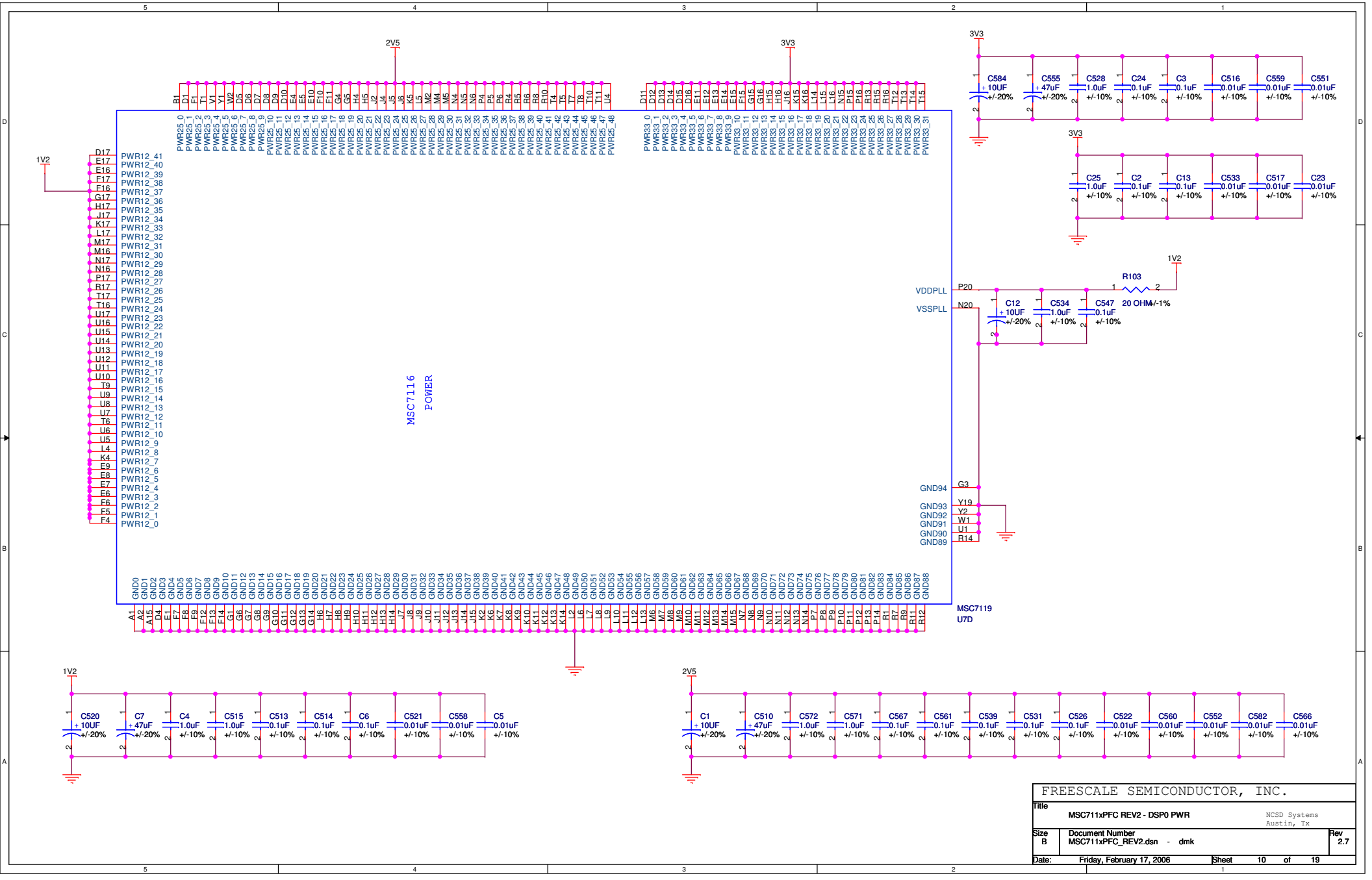

ENET SWITCH CONFIG
U504
VT6510B
VIA VT6510B
U500
25MHz
R10 0 OHM +/-5%
R4 1.0M+/-5%
R264 22ohm +/-5%
R184 22ohm +/-5%
C15 10pF +/-5%
C11 10pF +/-5%
C595 100uF
C40 100uF
INTERNAL PULL-UP
UNUSED_INPUT
FREESCALE SEMICONDUCTOR, INC.
MSC711xPFC REV2 - ENET SWITCH
NCSD Systems Austin, Tx
Document Number MSC711xPFC_REV2.dsn - dmk
Size B
Rev 2.7
Date: Friday, February 17, 2006
Sheet 7 of 19


MSC711x HDI16
MSC711x MAIN
MSC711x BOOT EEPROM
U7C
MSC7116
Host Data
Interface
MSC7119
U7B
MSC7116
SYSTEM & I/O INTERFACE
MSC7119
U6
M25P80-VMP6G
DNP
FREESCALE SEMICONDUCTOR, INC.
Title
MSC711xPFC REV2 - DSP0 SYS
NCSD Systems
Austin, Tx
Size
B
Document Number
MSC711xPFC_REV2.dsn - dmk
Rev
2.7
Date:
Friday, February 17, 2006
Sheet
8
of
19



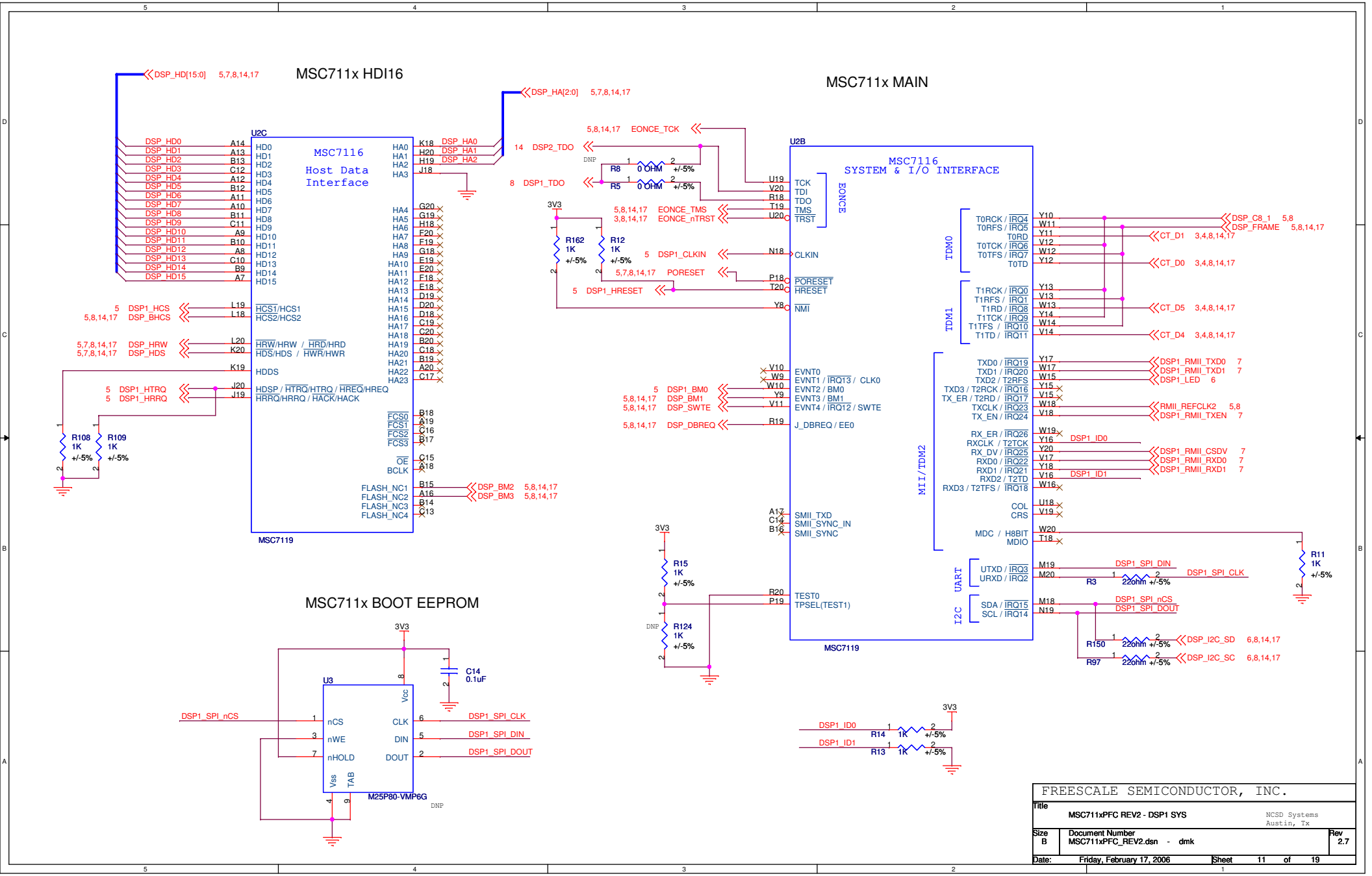

FREESCALE SEMICONDUCTOR, INC.

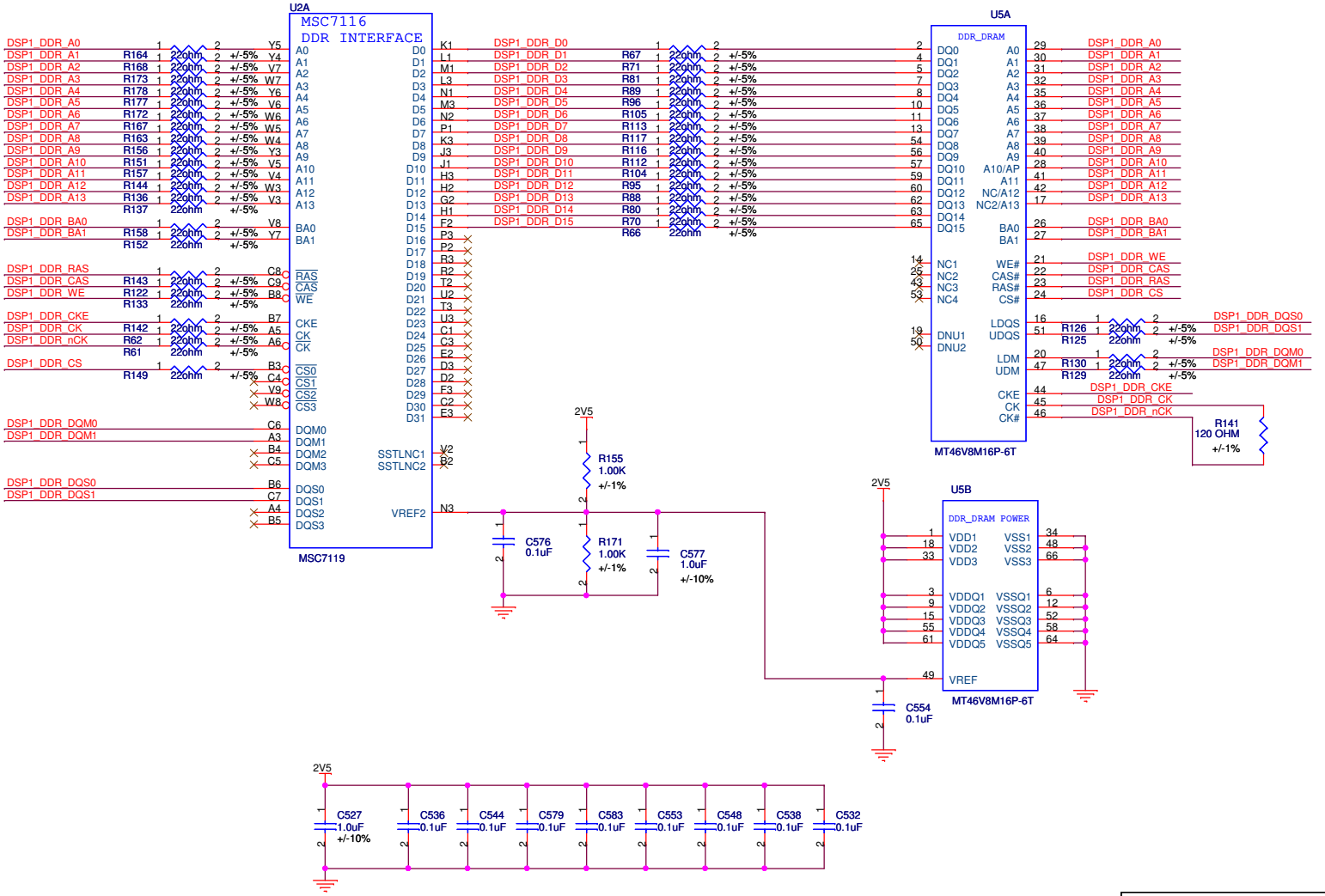
| | | |
|---|---|---|
| Title | MSC711xPFC REV2 - DSP0 DDR | NCSD Systems Austin, Tx |
| Size B | Document Number MSC711xPFC_REV2.dsn - dmk | Rev 2.7 |
| Date: Friday, February 17, 2006 | Sheet 9 of 19 | |




FREESCALE SEMICONDUCTOR, INC.

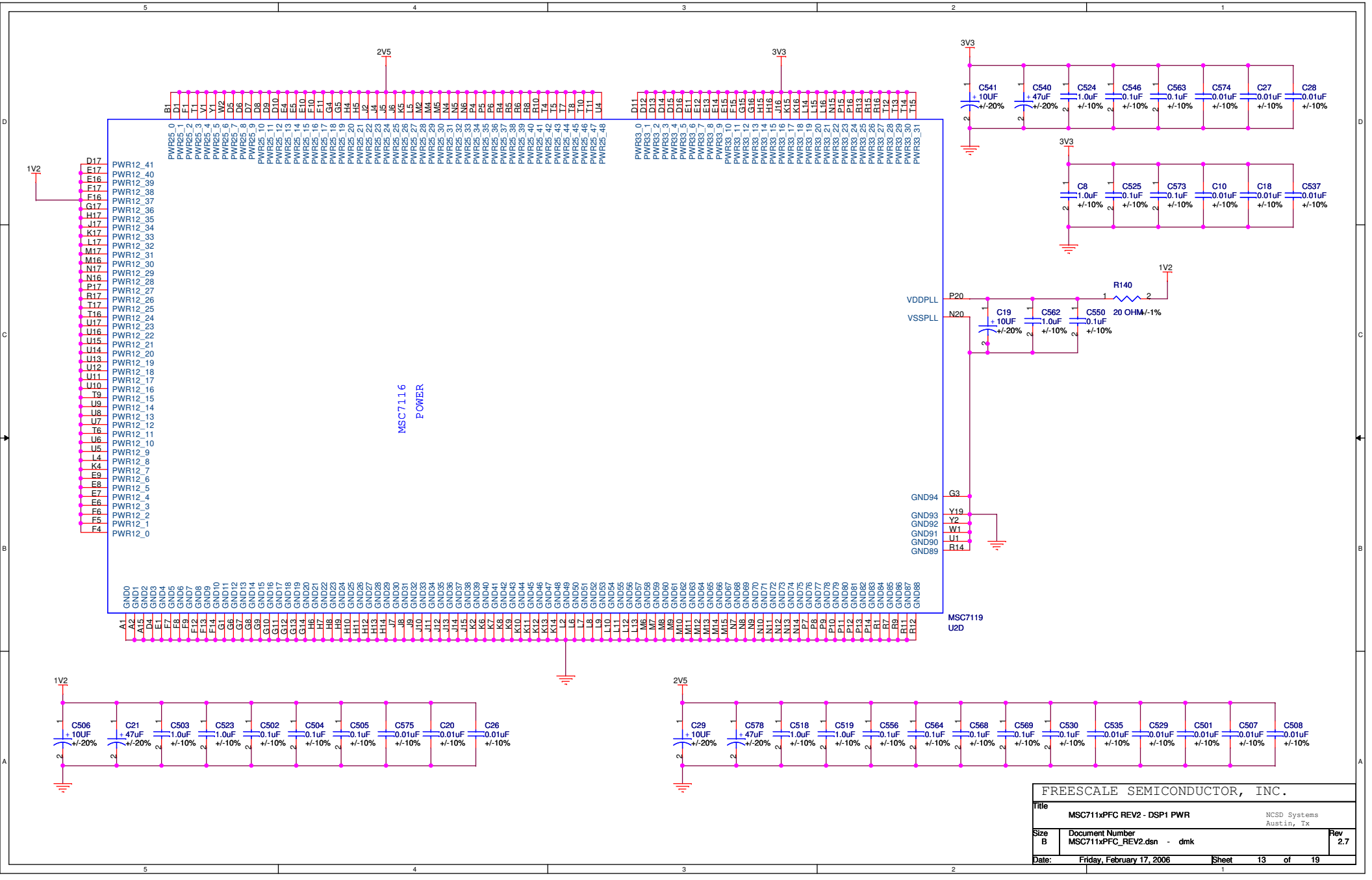
| | | |
|---|---|---|
| Title | MSC711xPFC REV2 - DSP0 PWR | NCSD Systems Austin, Tx |
| Size B | Document Number MSC711xPFC_REV2.dsn - dmk | Rev 2.7 |
| Date: Friday, February 17, 2006 | Sheet 10 of 19 | |


MSC711x HDI16
MSC711x MAIN
MSC711x BOOT EEPROM
U2C
MSC7116
Host Data
Interface
MSC7119
U2B
MSC7116
SYSTEM & I/O INTERFACE
MSC7119
U3
M25P80-VMP6G
DNP
FREESCALE SEMICONDUCTOR, INC.
Title
MSC711xPFC REV2 - DSP1 SYS
NCSD Systems
Austin, Tx
Size
B
Document Number
MSC711xPFC_REV2.dsn - dmk
Rev
2.7
Date:
Friday, February 17, 2006
Sheet
11
of
19



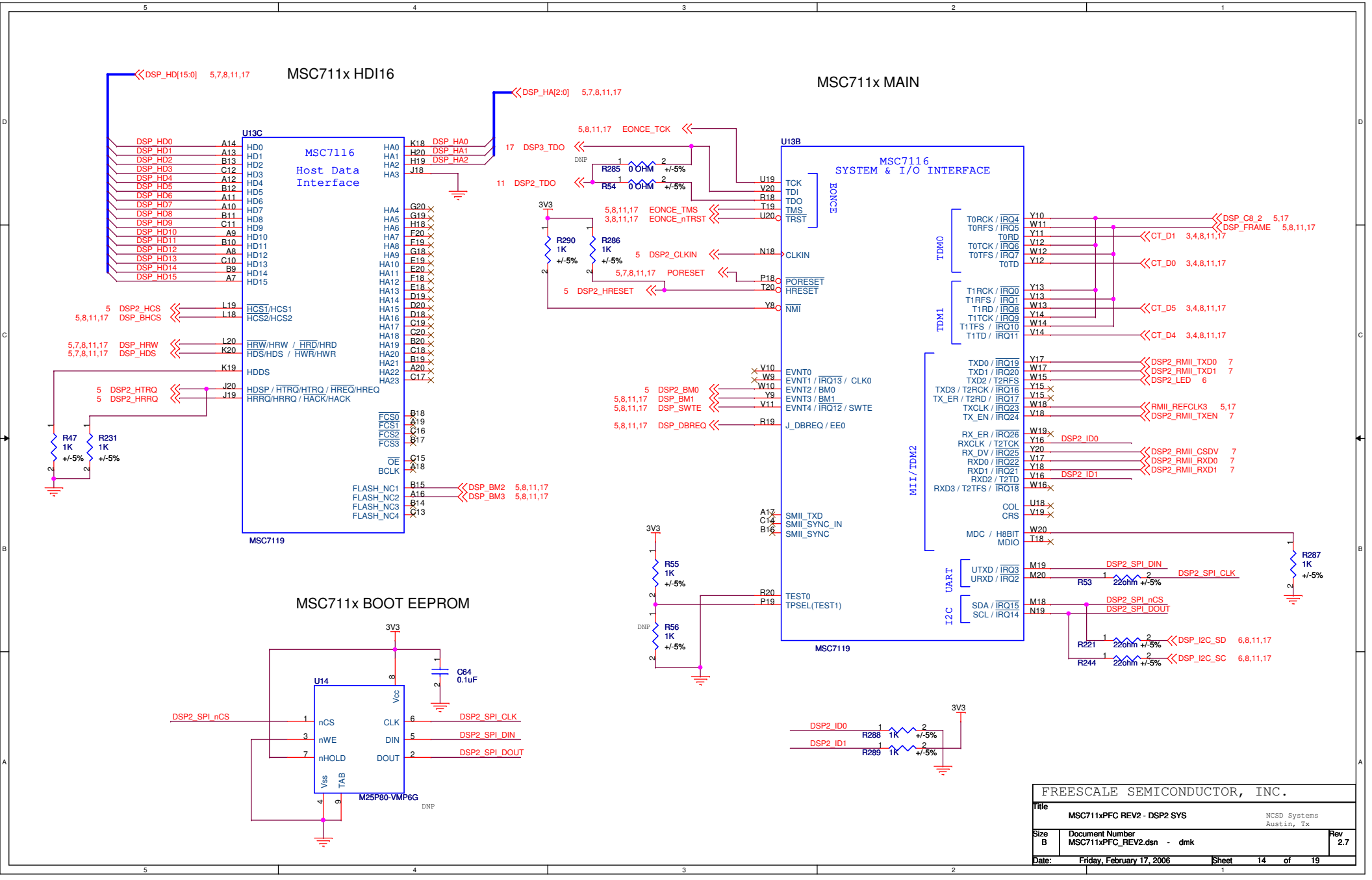

FREESCALE SEMICONDUCTOR, INC.

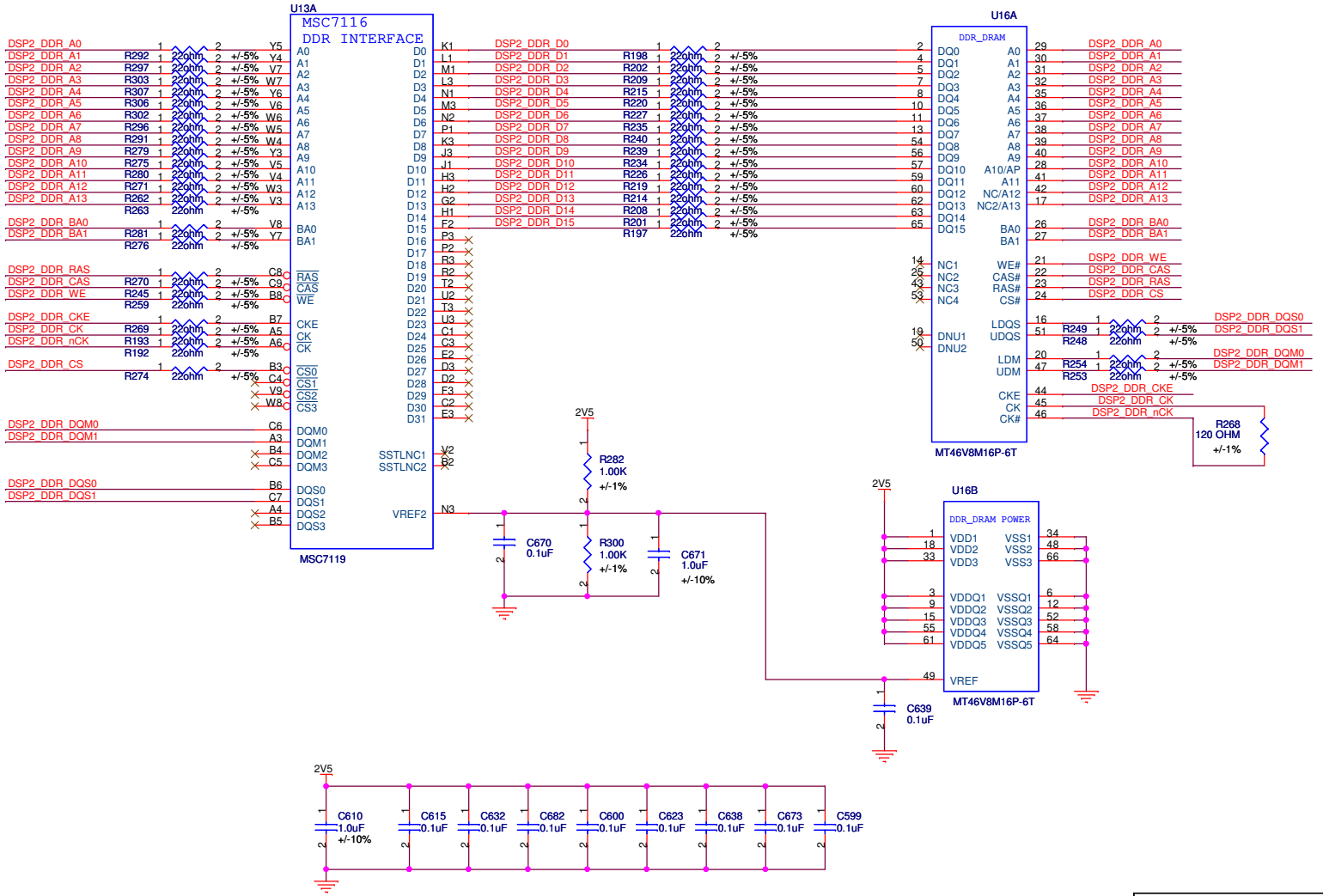
| | | |
|---|---|---|
| Title | MSC711xPFC REV2 - DSP1 DDR | NCSD Systems Austin, Tx |
| Size B | Document Number MSC711xPFC_REV2.dsn - dmk | Rev 2.7 |
| Date: Friday, February 17, 2006 | Sheet 12 of 19 | |




FREESCALE SEMICONDUCTOR, INC.

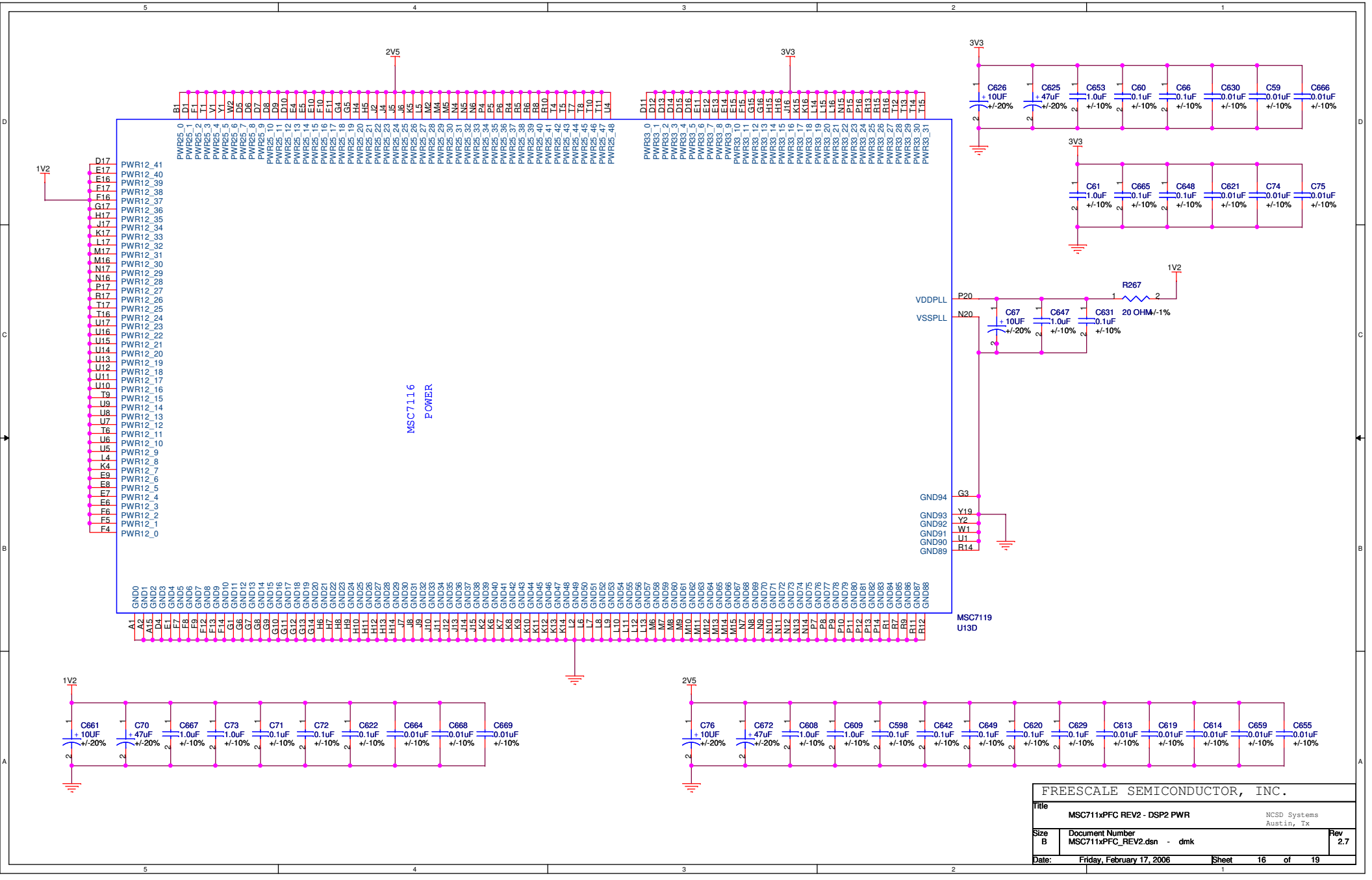
| | | |
|---|---|---|
| Title | MSC711xPFC REV2 - DSP1 PWR | NCSD Systems Austin, Tx |
| Size B | Document Number MSC711xPFC_REV2.dsn - dmk | Rev 2.7 |
| Date: Friday, February 17, 2006 | Sheet 13 of 19 | |


MSC711x HDI16
MSC711x MAIN
MSC711x BOOT EEPROM
U13C
MSC7116
Host Data
Interface
MSC7119
U13B
MSC7116
SYSTEM & I/O INTERFACE
MSC7119
U14
M25P80-VMP6G
DSP_HD[15:0] 5,7,8,11,17
DSP_HA[2:0] 5,7,8,11,17
DSP_HD0
DSP_HD1
DSP_HD2
DSP_HD3
DSP_HD4
DSP_HD5
DSP_HD6
DSP_HD7
DSP_HD8
DSP_HD9
DSP_HD10
DSP_HD11
DSP_HD12
DSP_HD13
DSP_HD14
DSP_HD15
DSP_HA0
DSP_HA1
DSP_HA2
5 DSP2_HCS
5,8,11,17 DSP_BHCS
5,7,8,11,17 DSP_HRW
5,7,8,11,17 DSP_HDS
5 DSP2_HTRQ
5 DSP2_HRRQ
R47 1K +/-5%
R231 1K +/-5%
17 DSP3_TDO
11 DSP2_TDO
R285 0 OHM +/-5%
R54 0 OHM +/-5%
5,8,11,17 EONCE_TCK
5,8,11,17 EONCE_TMS
3,8,11,17 EONCE_nTRST
5 DSP2_CLKIN
5,7,8,11,17 PORESET
5 DSP2_HRESET
R290 1K +/-5%
R286 1K +/-5%
5 DSP2_BM0
5,8,11,17 DSP_BM1
5,8,11,17 DSP_SWTE
5,8,11,17 DSP_DBREQ
DSP_BM2 5,8,11,17
DSP_BM3 5,8,11,17
R55 1K +/-5%
R56 1K +/-5%
R20
R19
DSP2_ID0
DSP2_ID1
R288 1K +/-5%
R289 1K +/-5%
C64 0.1uF
DSP2_SPI_nCS
DSP2_SPI_CLK
DSP2_SPI_DIN
DSP2_SPI_DOUT
DSP_C8_2 5,17
DSP_FRAME 5,8,11,17
CT_D1 3,4,8,11,17
CT_D0 3,4,8,11,17
CT_D5 3,4,8,11,17
CT_D4 3,4,8,11,17
DSP2_RMII_TXD0 7
DSP2_RMII_TXD1 7
DSP2_LED 6
RMII_REFCLK3 5,17
DSP2_RMII_TXEN 7
DSP2_RMII_CSDV 7
DSP2_RMII_RXD0 7
DSP2_RMII_RXD1 7
DSP2_SPI_DIN
DSP2_SPI_CLK
R53 22ohm +/-5%
DSP2_SPI_nCS
DSP2_SPI_DOUT
DSP_I2C_SD 6,8,11,17
DSP_I2C_SC 6,8,11,17
R221 22ohm +/-5%
R244 22ohm +/-5%
R287 1K +/-5%
TDM0
TDM1
MII/TDM2
UART
I2C
JTAG
EONCE
FREESCALE SEMICONDUCTOR, INC.
Title
MSC711xPFC REV2 - DSP2 SYS
NCSD Systems
Austin, Tx
Size B
Document Number MSC711xPFC_REV2.dsn - dmk
Rev 2.7
Date: Friday, February 17, 2006
Sheet 14 of 19

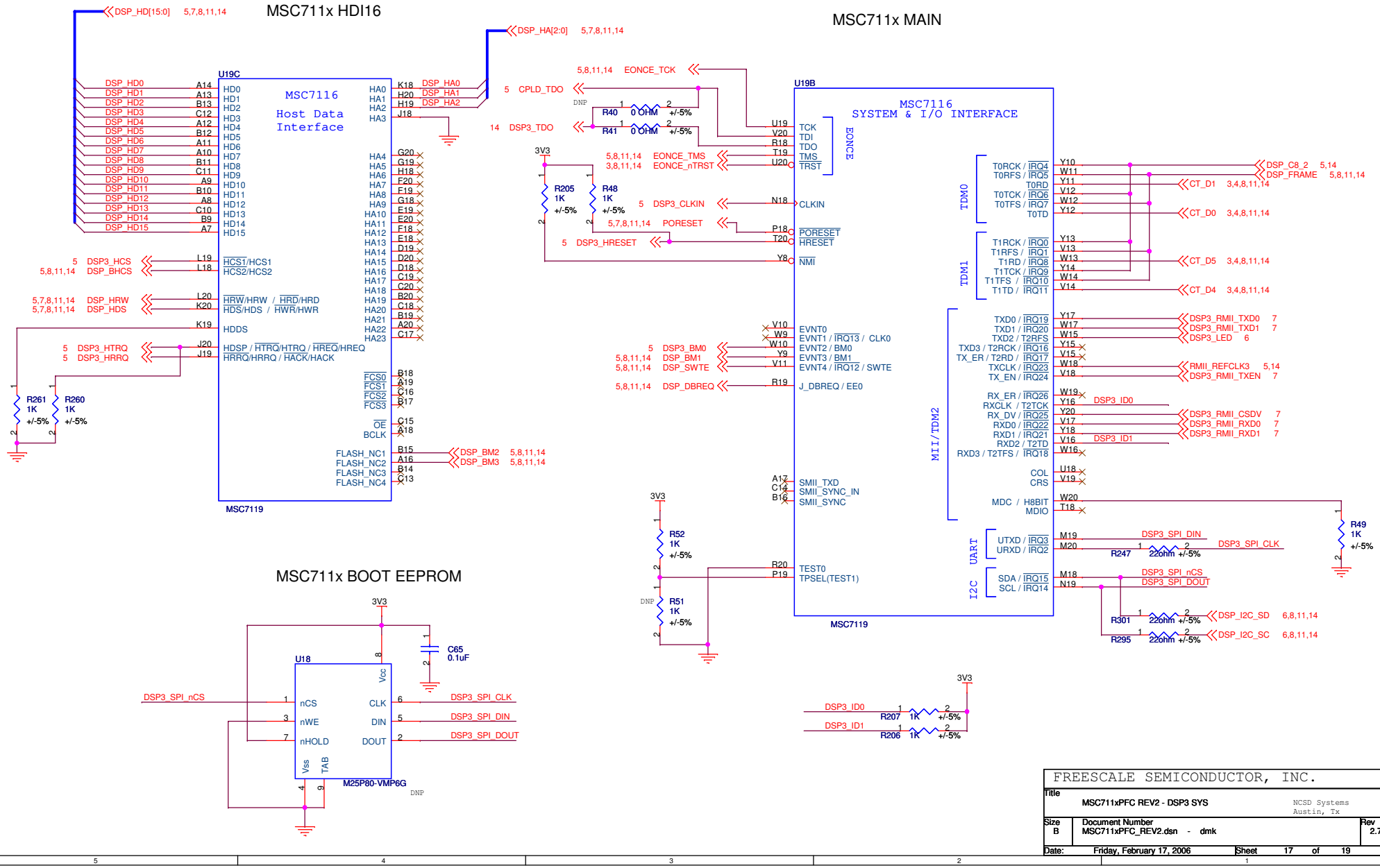


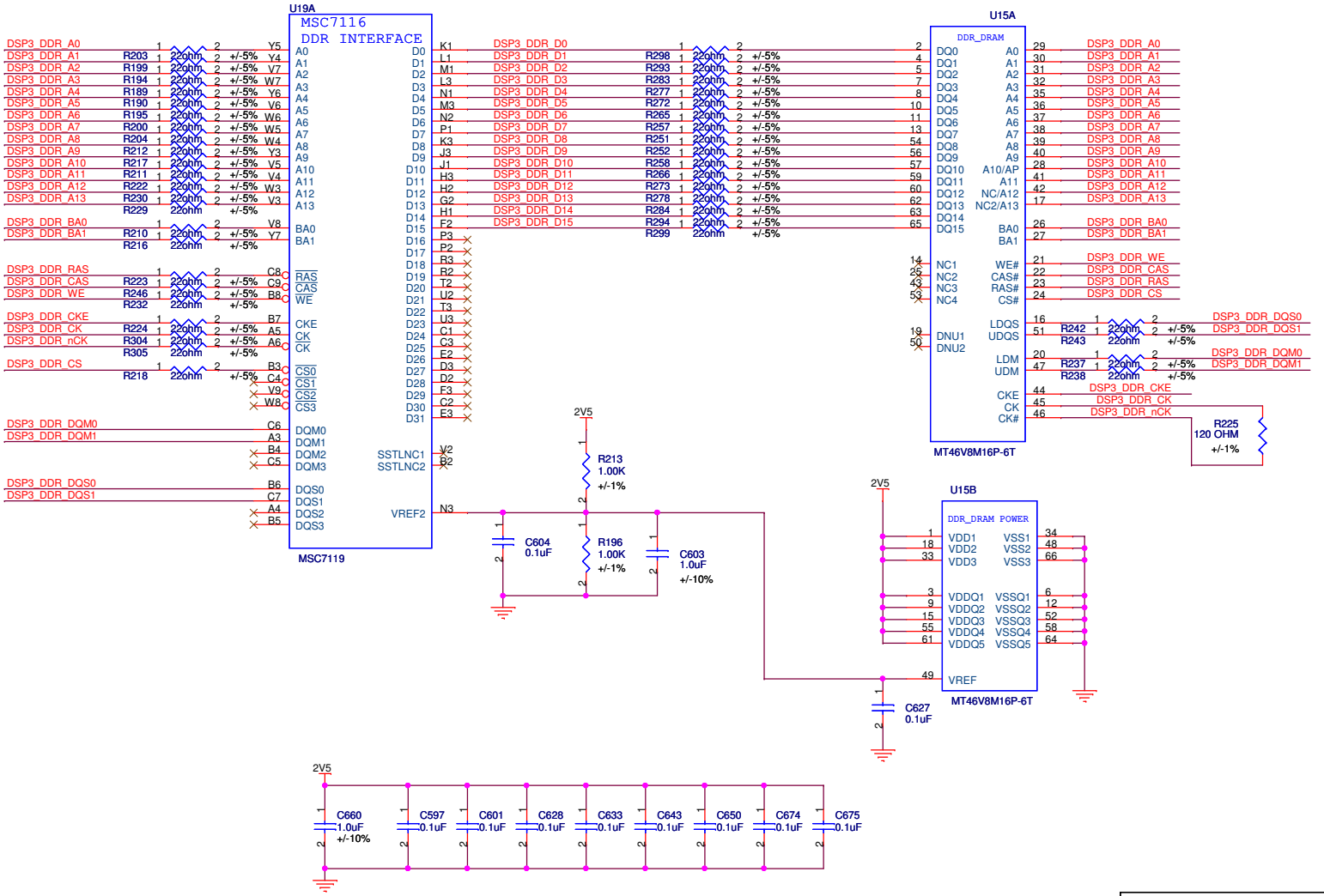

FREESCALE SEMICONDUCTOR, INC.

| | | |
|---|---|---|
| Title | MSC711xPFC REV2 - DSP2 DDR | NCSD Systems Austin, Tx |
| Size B | Document Number MSC711xPFC_REV2.dsn - dmk | Rev 2.7 |
| Date: Friday, February 17, 2006 | Sheet 15 of 19 | |



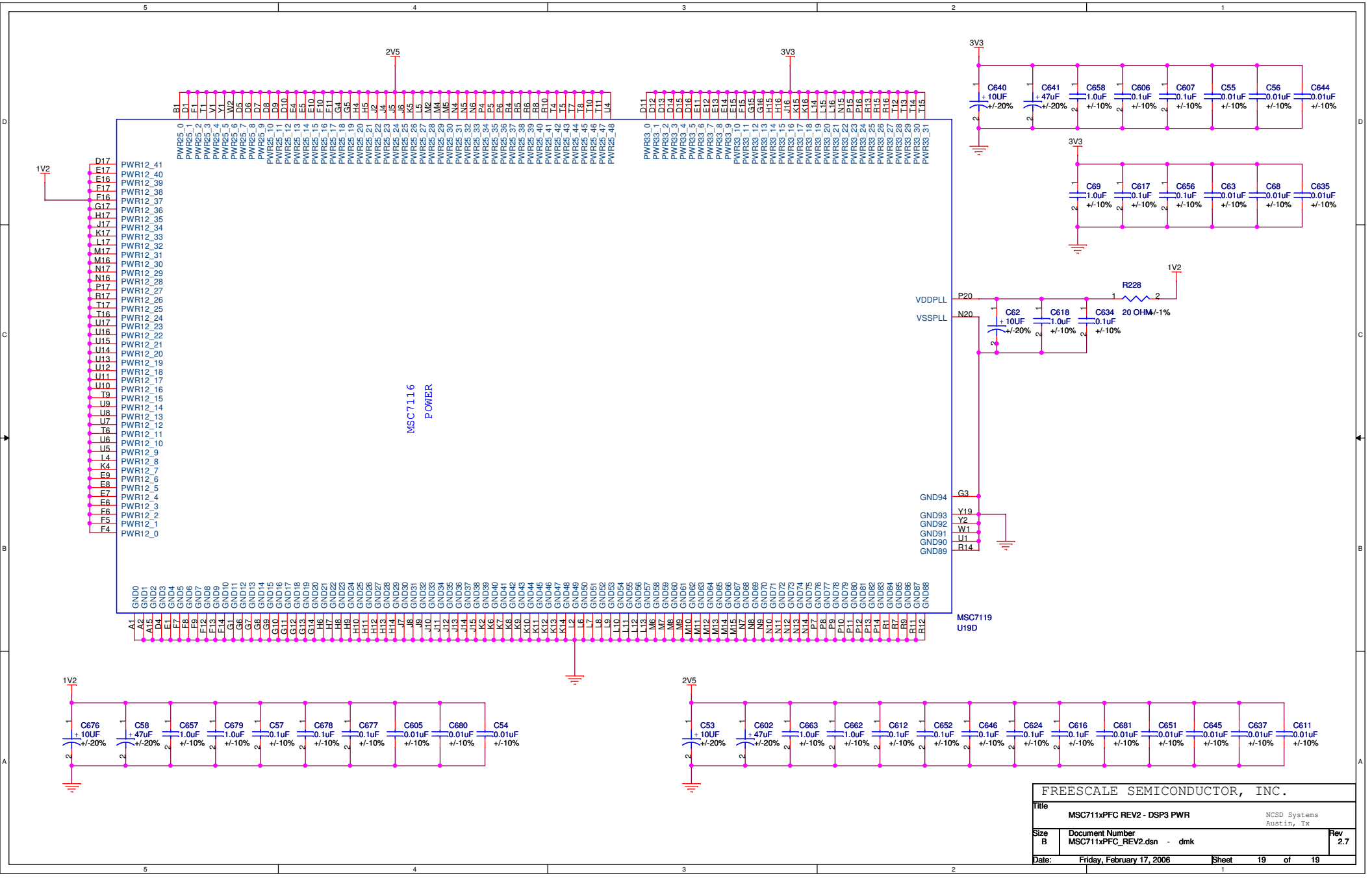

| | | |
|---|---|---|
| FREESCALE SEMICONDUCTOR, INC. | | |
| Title: MSC711xPFC REV2 - DSP2 PWR | NCSD Systems Austin, Tx | |
| Size: B | Document Number: MSC711xPFC_REV2.dsn - dmk | Rev: 2.7 |
| Date: Friday, February 17, 2006 | Sheet 16 of 19 | |


MSC711x HDI16
MSC711x MAIN
MSC711x BOOT EEPROM
U19C
MSC7116
Host Data
Interface
MSC7119
U19B
MSC7116
SYSTEM & I/O INTERFACE
MSC7119
U18
M25P80-VMP6G
DSP_HD[15:0] 5,7,8,11,14
DSP_HA[2:0] 5,7,8,11,14
HDSP / HTRQ/HTRQ / HREQ/HREQ
HRRQ/HRRQ / HACK/HACK
HDDS
HRW/HRW / HRD/HRD
HDS/HDS / HWR/HWR
HCS1/HCS1
HCS2/HCS2
FCS0
FCS1
FCS2
FCS3
OE
BCLK
FLASH_NC1
FLASH_NC2
FLASH_NC3
FLASH_NC4
TCK
TDI
TDO
TMS
TRST
CLKIN
PORESET
HRESET
NMI
EVNT0
EVNT1 / IRQ13 / CLK0
EVNT2 / BM0
EVNT3 / BM1
EVNT4 / IRQ12 / SWTE
J_DBREQ / EE0
SMII_TXD
SMII_SYNC_IN
SMII_SYNC
TEST0
TPSEL(TEST1)
T0RCK / IRQ4
T0RFS / IRQ5
T0RD
T0TCK / IRQ6
T0TFS / IRQ7
T0TD
T1RCK / IRQ0
T1RFS / IRQ1
T1RD / IRQ8
T1TCK / IRQ9
T1TFS / IRQ10
T1TD / IRQ11
TXD0 / IRQ19
TXD1 / IRQ20
TXD2 / T2RFS
TXD3 / T2RCK / IRQ16
TX_ER / T2RD / IRQ17
TXCLK / IRQ23
TX_EN / IRQ24
RX_ER / IRQ26
RXCLK / T2TCK
RX_DV / IRQ25
RXD0 / IRQ22
RXD1 / IRQ21
RXD2 / T2TD
RXD3 / T2TFS / IRQ18
COL
CRS
MDC / H8BIT
MDIO
UTXD / IRQ3
URXD / IRQ2
SDA / IRQ15
SCL / IRQ14
EONCE
TDM0
TDM1
MII/TDM2
UART
I2C
R40 0 OHM +/-5%
R41 0 OHM +/-5%
R205 1K +/-5%
R48 1K +/-5%
R261 1K +/-5%
R260 1K +/-5%
R52 1K +/-5%
R51 1K +/-5%
R49 1K +/-5%
R247 22ohm +/-5%
R301 22ohm +/-5%
R295 22ohm +/-5%
R207 1K +/-5%
R206 1K +/-5%
C65 0.1uF
DNP
3V3
FREESCALE SEMICONDUCTOR, INC.
Title
MSC711xPFC REV2 - DSP3 SYS
NCSD Systems
Austin, Tx
Size B
Document Number MSC711xPFC_REV2.dsn - dmk
Rev 2.7
Date: Friday, February 17, 2006
Sheet 17 of 19




FREESCALE SEMICONDUCTOR, INC.

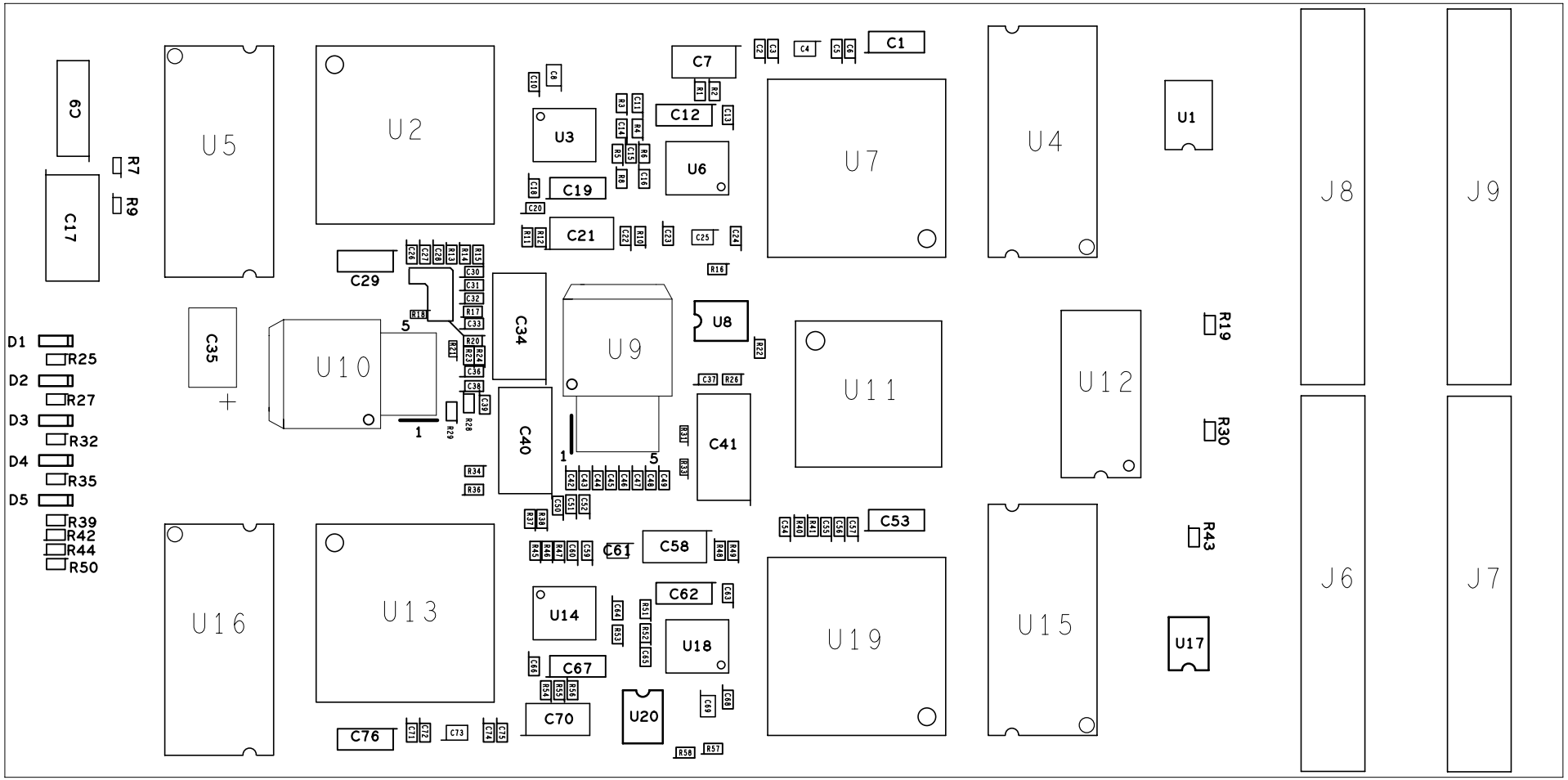
| | | |
|---|---|---|
| Title | MSC711xPFC REV2 - DSP3 DDR | NCSD Systems Austin, Tx |
| Size B | Document Number MSC711xPFC_REV2.dsn - dmk | Rev 2.7 |
| Date: Friday, February 17, 2006 | Sheet 18 of 19 | |

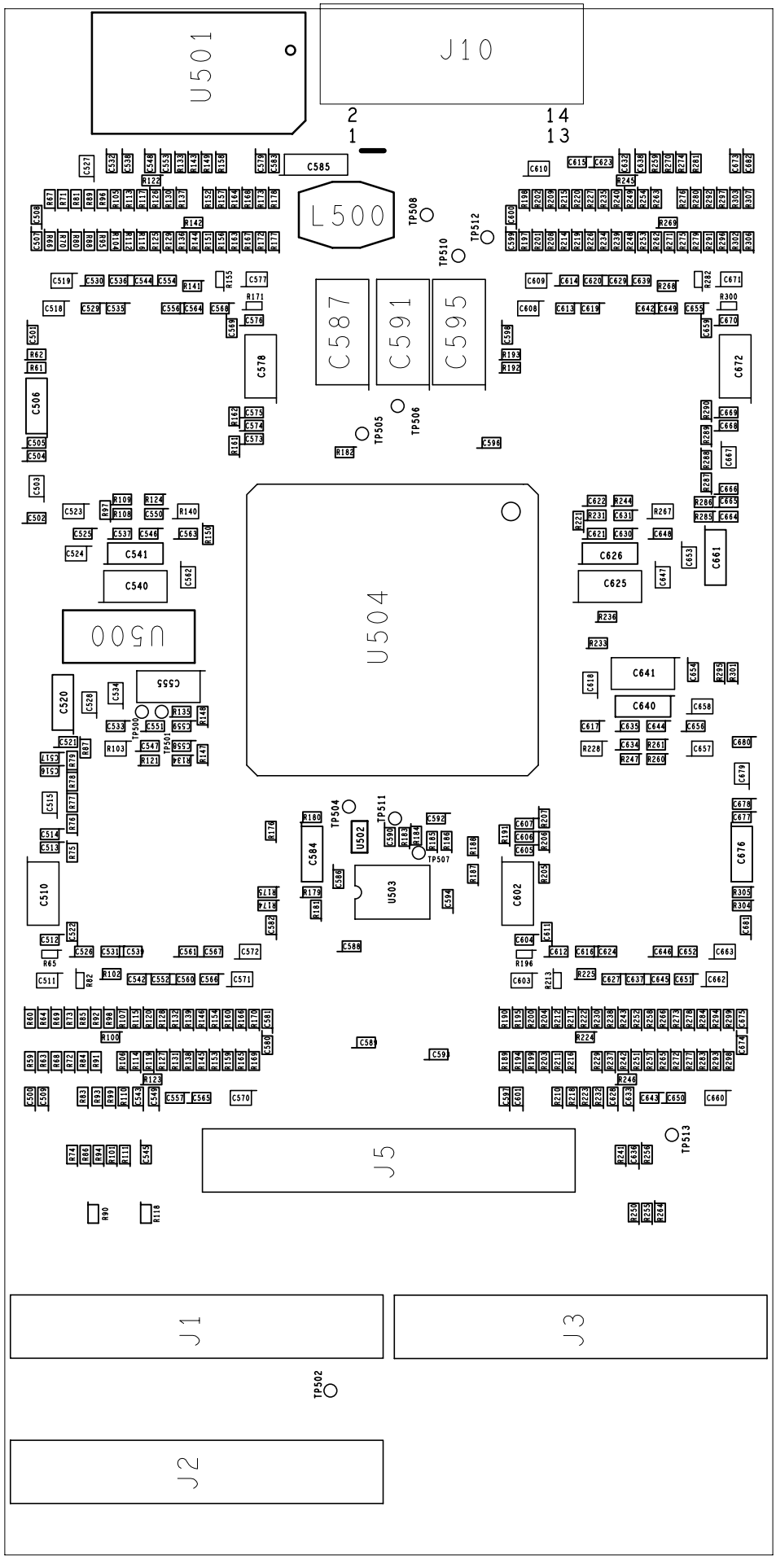



FREESCALE SEMICONDUCTOR, INC.

| | | |
|---|---|---|
| Title | MSC711xPFC REV2 - DSP3 PWR | NCSD Systems Austin, Tx |
| Size B | Document Number MSC711xPFC_REV2.dsn - dmk | Rev 2.7 |
| Date: Friday, February 17, 2006 | Sheet 19 of 19 | |

# Appendix C SPT711xPFCE Assembly Drawing


C3
C17
R7
R9
U5
U2
C29
C35
U10
C34
U9
C40
C41
U3
C7
C12
U6
C19
C21
U8
U7
U4
U1
C1
J8
J9
R19
U11
U12
R30
D1
R25
D2
R27
D3
R32
D4
R35
D5
R39
R42
R44
R50
U16
U13
C76
U14
C62
U18
C67
C70
U20
C58
C61
C53
U19
U15
U17
R43
J6
J7


U501
J10
2
1
14
13
L500
C585
C587
C591
C595
C578
C506
C672
C626
C625
C661
C541
C540
U500
U504
C555
C520
C641
C640
C510
C584
U502
U503
C602
C676
J5
J1
J3
J2
TP502
TP504
TP505
TP506
TP507
TP508
TP510
TP511
TP512
TP513

## THIS PAGE INTENTIONALLY LEFT BLANK

### *How to Reach Us:*

**Home Page:**
www.freescale.com

**email:**
support@freescale.com

**USA/Europe or Locations Not Listed:**
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
1-800-521-6274
480-768-2130
support@freescale.com

**Europe, Middle East, and Africa:**
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

**Japan:**
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku
Tokyo 153-0064, Japan
0120 191014
+81 3 5437 9125
support.japan@freescale.com

**Asia/Pacific:**
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate,
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

**For Literature Requests Only:**
Freescale Semiconductor
Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447
303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor
@hibbertgroup.com



Document Number: PTKIT711xUG
Rev. 1
04/2006

freescale™
semiconductor