Application Note

AN2347/D
Rev. 0, 9/2002

Using an MPC8260 and an MPC7410 with Shared Memory

Dave McCartney
NCSG Applications
East Kilbride,
Scotland

The MPC8260 PowerQUICC II™ is a versatile communications processor that integrates on one chip a high-performance MPC603e™ microprocessor, a very flexible system integration unit, and many communications peripheral controllers that can be used in a variety of applications, particularly in communications and networking systems. The MPC8260 can be operated with the on-chip RISC microprocessor core disabled and allows an external processor device to use all of the MPC8260's system integration and communication features.

This application note describes the design of a system containing a high-performance MPC7410 RISC microprocessor and an MPC8260.

When used with an external processor, the MPC8260 does not support a full ECC memory system. Therefore, it is necessary to use an external memory controller to support an ECC memory system. In this application, a Tundra Semiconductors PowerPro CA91L750 is used as the memory controller.

1.1 Block Diagram of System

A block diagram of the main components in the system is shown in Figure 1-1.

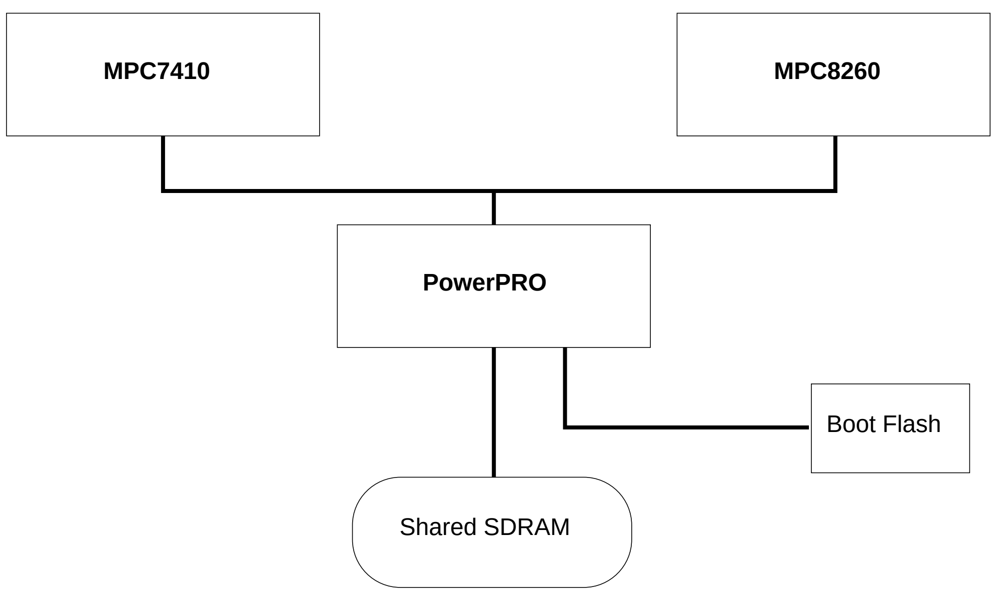

Figure 1-1. System Block Diagram

The main components consist of the MPC7410, the MPC8260 and the Tundra CA91L750 PowerPRO memory controller. These devices are connected using a standard 60x bus operating at 83MHz. The PowerPRO arbitrates control of the bus between the MPC7410 and the CPM within the MPC8260.

1.2 Features

The key features of the three main devices used in the application are as follows:

- MPC7410
  - High performance, superscaler microprocessor
  - Eight independent execution units and three register files
  - Rename buffers
  - Completion uit
  - Separate on-chip L1 instruction and data caches (Harvard architecture)
  - Level 2 (L2) cache interface
  - Separate memory management units (MMUs) for instructions and data
  - Efficient data flow
  - Multiprocessing support features
  - Power and thermal management
  - Performance monitor can be used to help debug system designs and improve software efficiency
  - In-system testability and debugging features through JTAG boundary-scan capability
- MPC8260
  - G2 dual-issue integer core—disabled in this application
  - Lower power
  - Separate power supply for internal logic and for I/O logic
  - Separate PLLs for G2core and for CPM
  - 64-bit data and 32-bit address 60x bus
  - 32-bit data and 18-bit address local bus
  - System interface unit (SIU)
  - Twelve-bank memory controller
  - Communications processor module (CPM)
- Tundra PowerPRO (CA91L750)
  - Processor interface
  - SDRAM interface
  - FLASH/ROM Interface
  - Two high-speed UARTs
  - I^2C interface
  - Programmable general purpose timer
  - System watchdog timer
  - 32 channel interrupt controller
  - 50 general purpose I/O pins—multiplexed with other functions
  - JTAG support for board level testing

1.3 Interface Schematics

A full set of schematics is provided in “Appendix A—Interface Schematics” on page 4. Figure 1-2 shows a top-level interconnect of the major system components. It is followed by three top-level schematics that show each major component’s functional blocks—Figure 1-3 shows the MPC8260, Figure 1-4 shows the MPC7410, and Figure 1-5 shows the Tundra PowerPRO CA91L750. Figure 1-6 shows the shared SDRAM that is controlled by the Tundra device and Figure 1-7 shows the boot FLASH.

Subsequent figures show the detailed interconnection to each of the functional blocks.

1.4 Initialization Software

A complete listing of the initialization code and required header files is provided in “Appendix B—Initialization Software” on page 25. This code should be held in the boot FLASH device and provides all the initialization for both the PowerPRO and the MPC7410 and MPC8260 devices. The MPC8260 is configured in hardware to be a configuration slave.

Appendix A—Interface Schematics

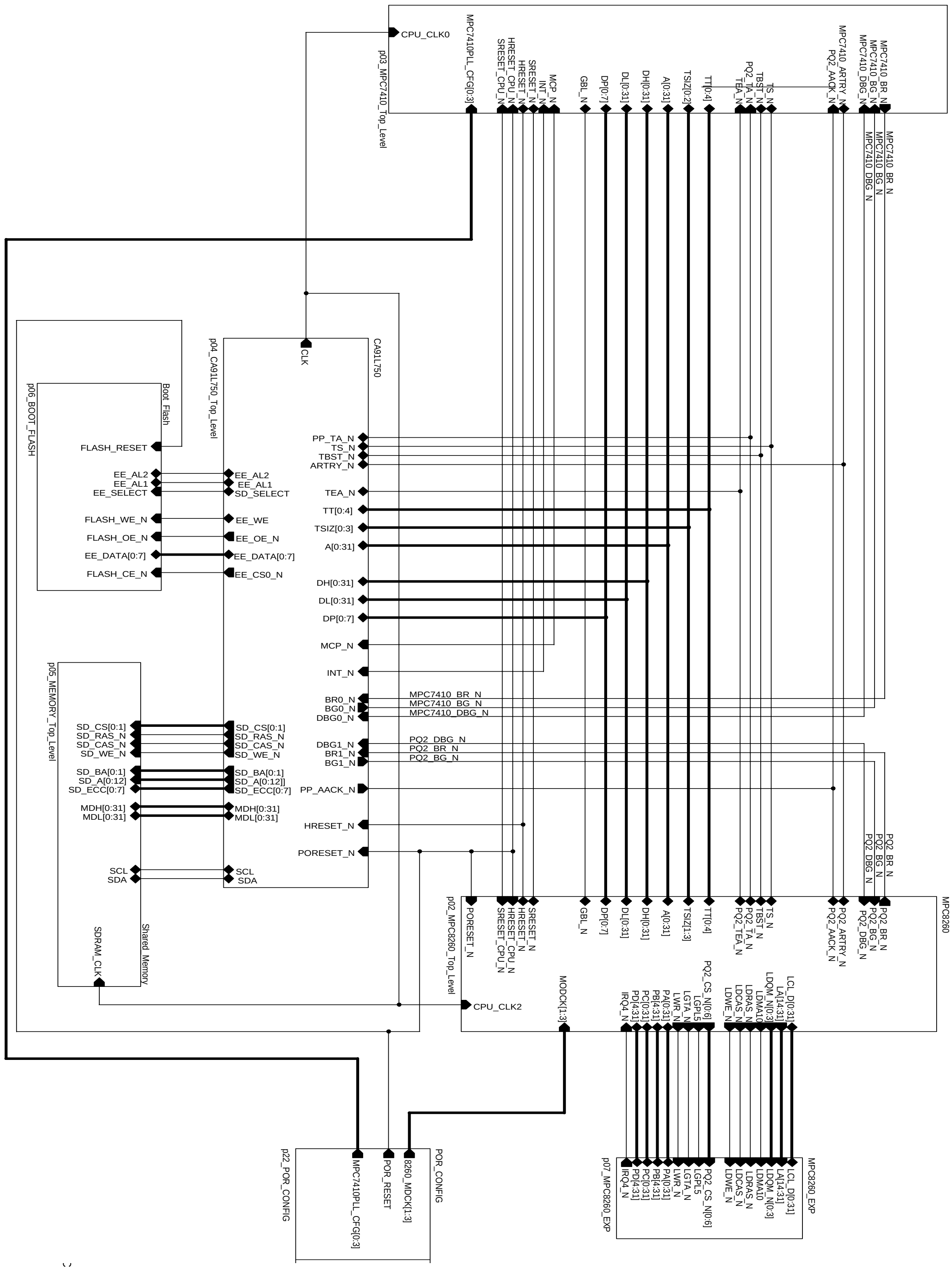

Figure 1-2. MPC8260/MPC7410 Interface

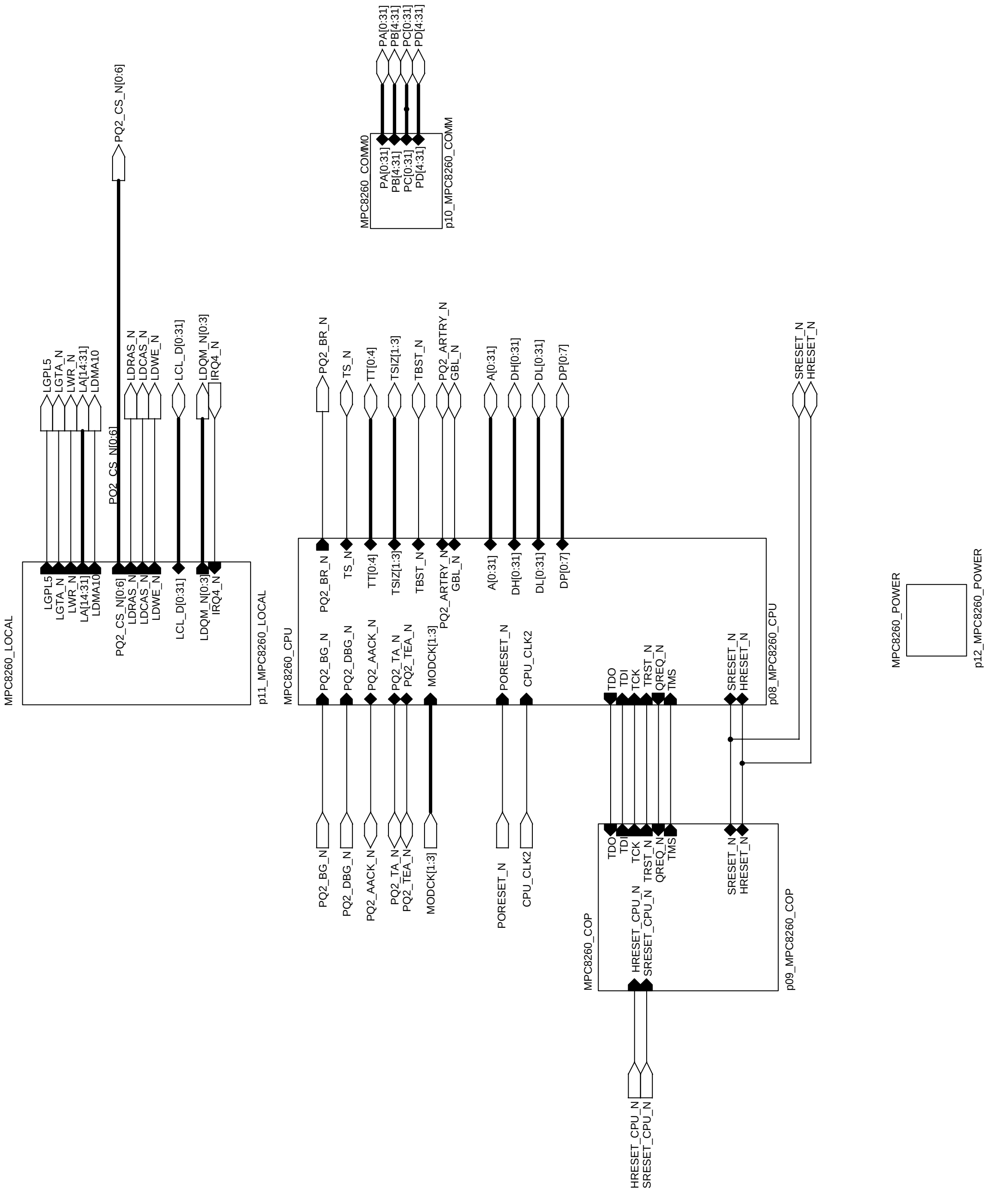

Figure 1-3. MPC8260 Top Level

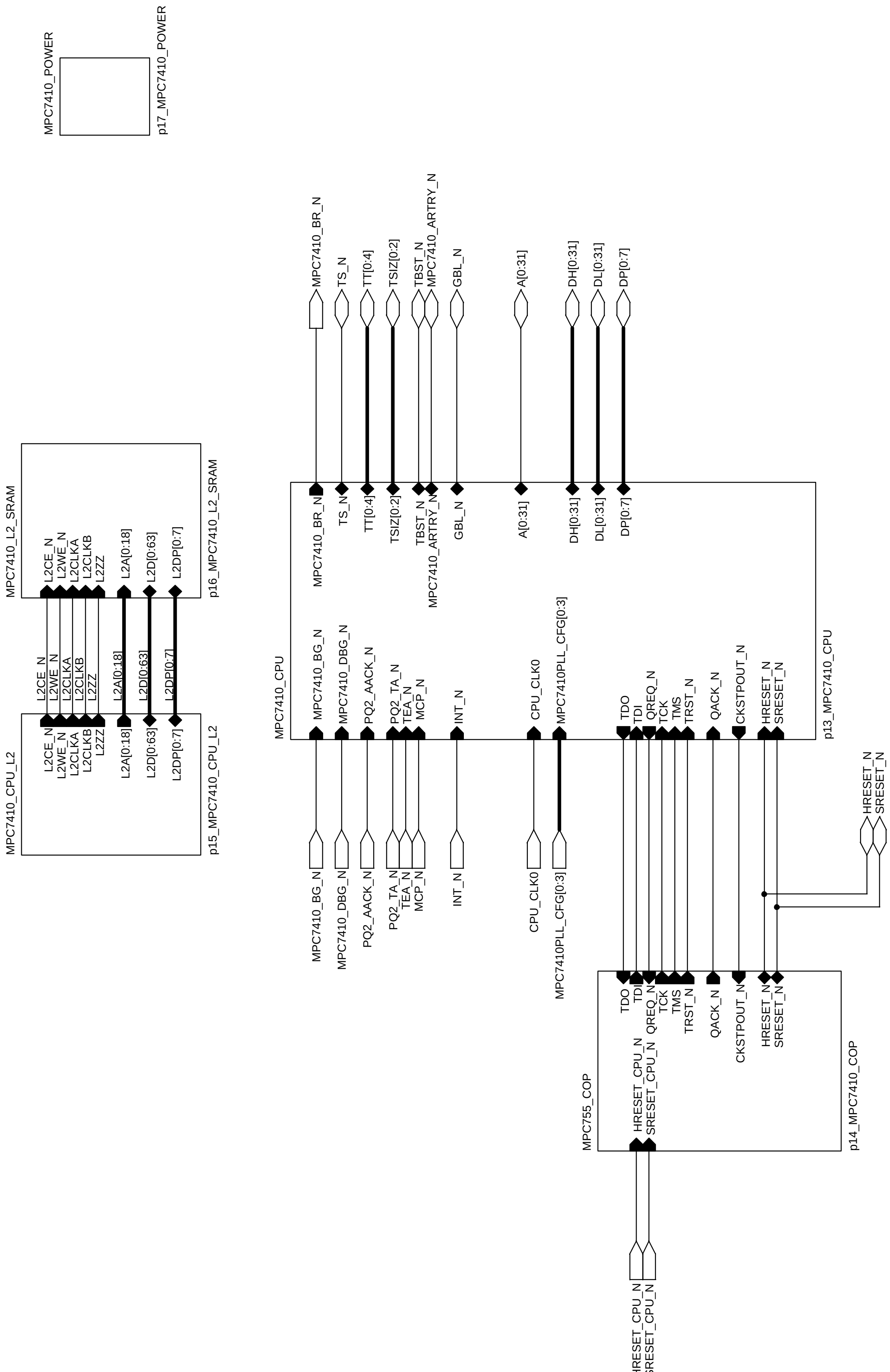

Figure 1-4. MPC7410 Top Level

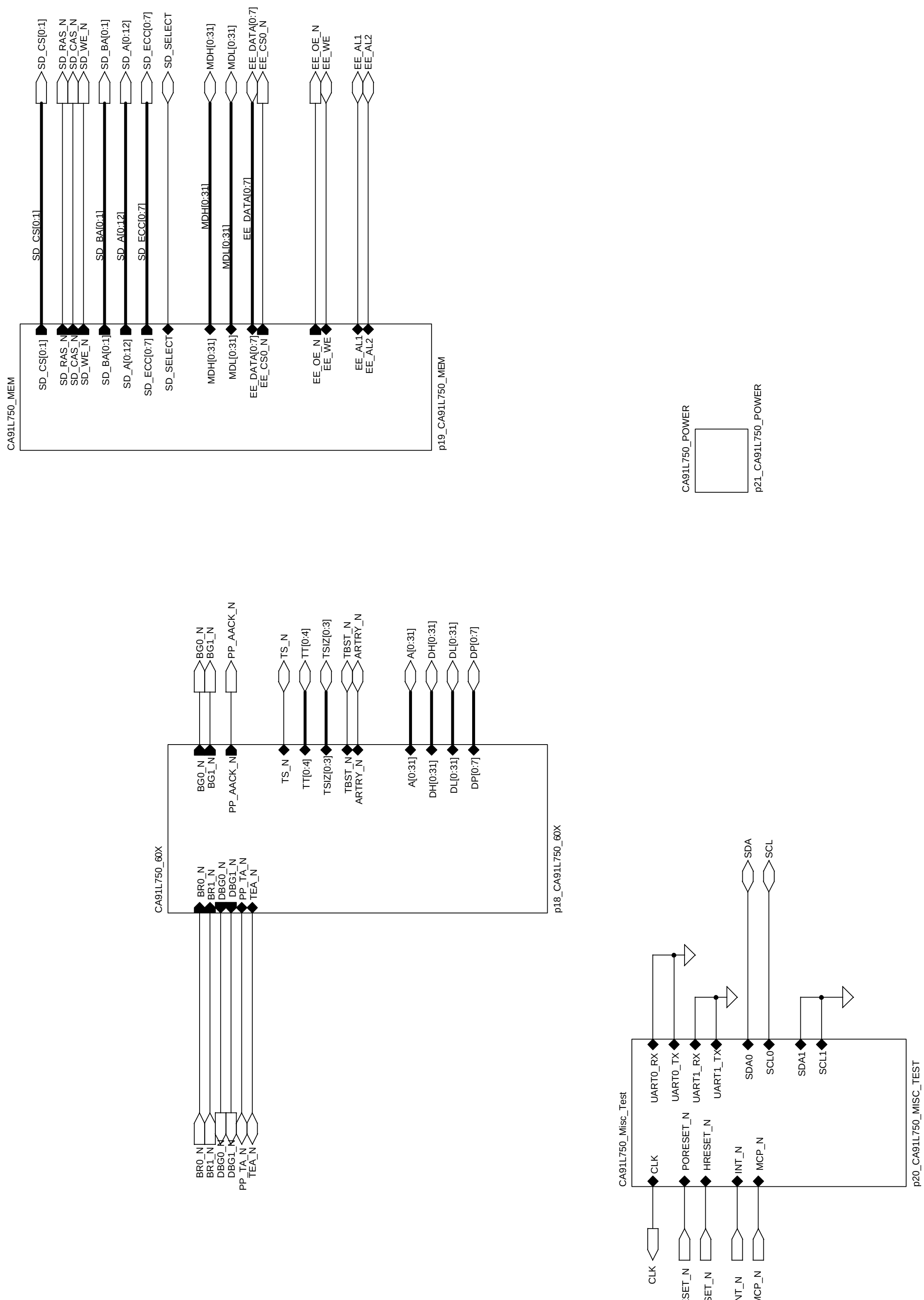

Figure 1-5. Tundra CA91L750 Top Level

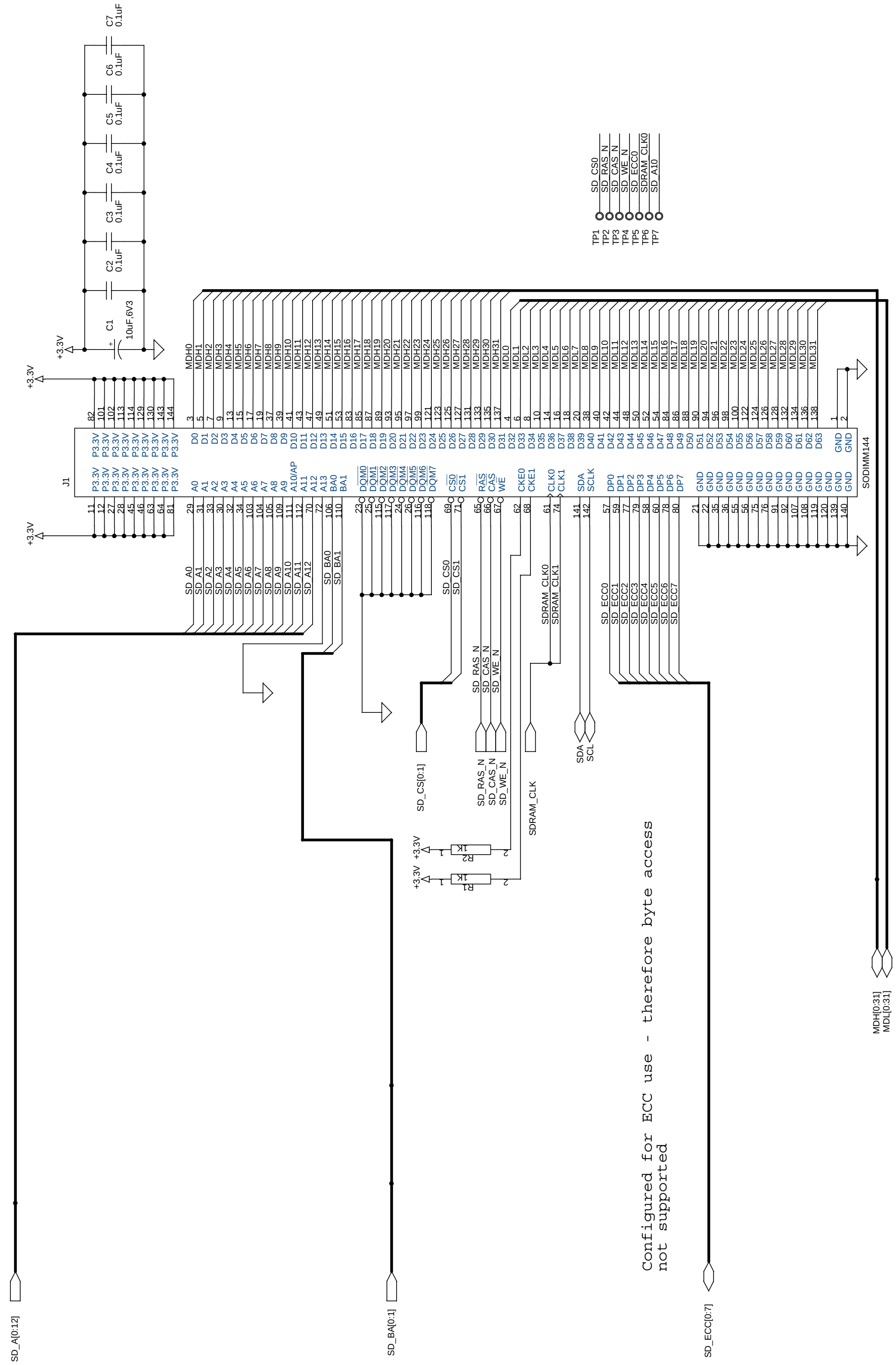

Figure 1-6. Shared Memory

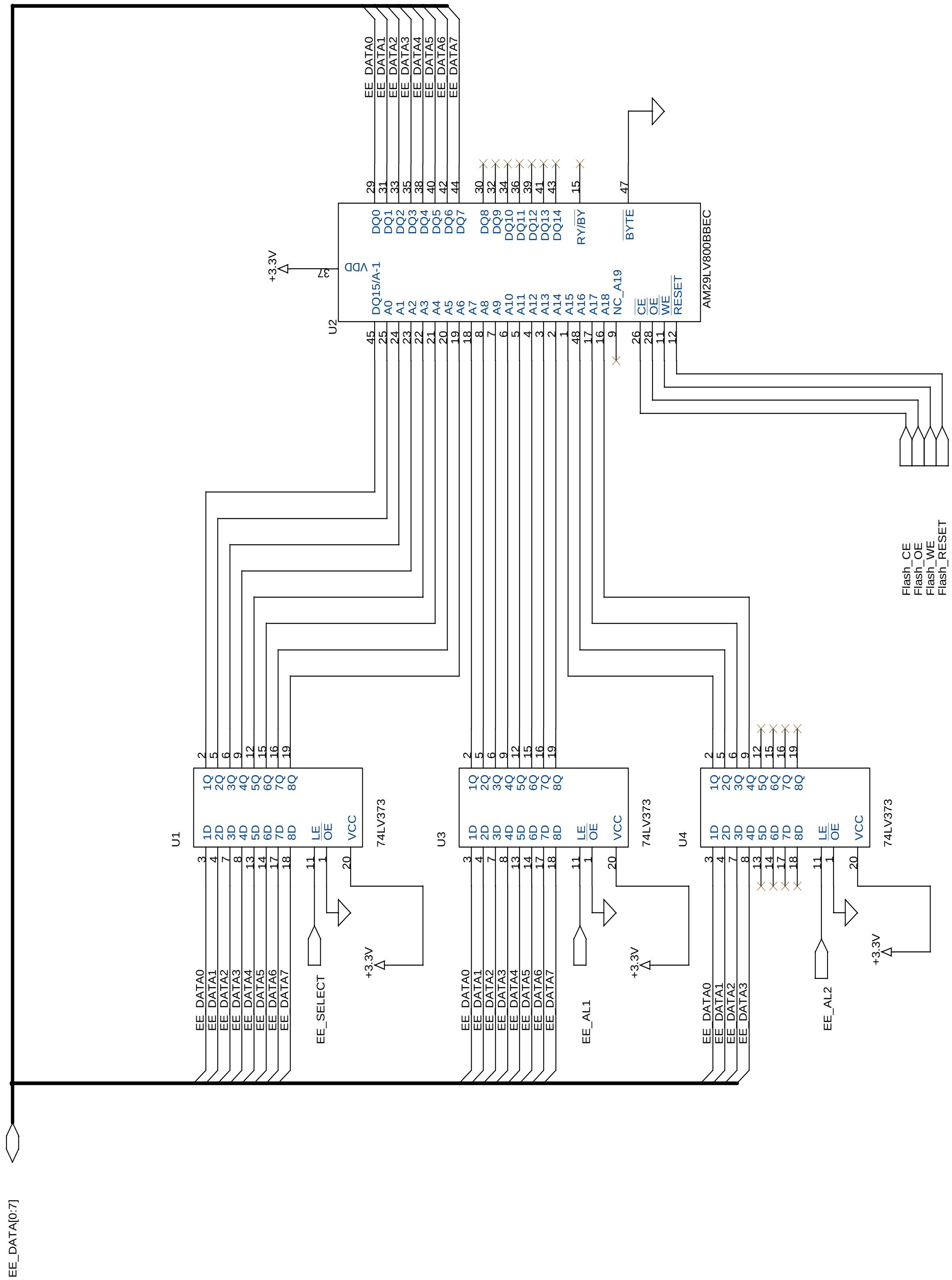

Figure 1-7. Boot Flash

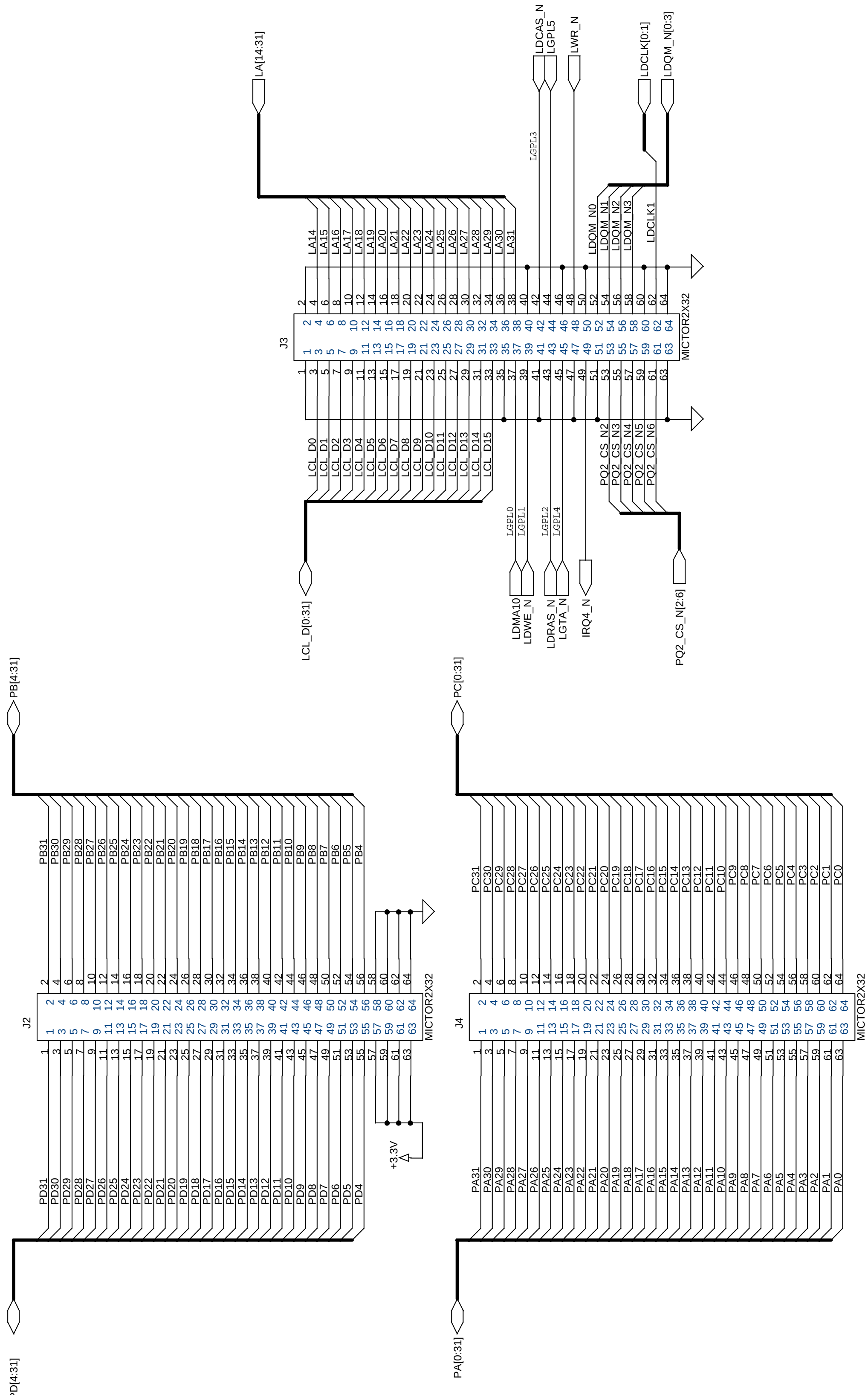

Figure 1-8. MPC8260 Expansion

Note:
TSIZ0 should be pulled down to allow for when TSIZ0 is an input.

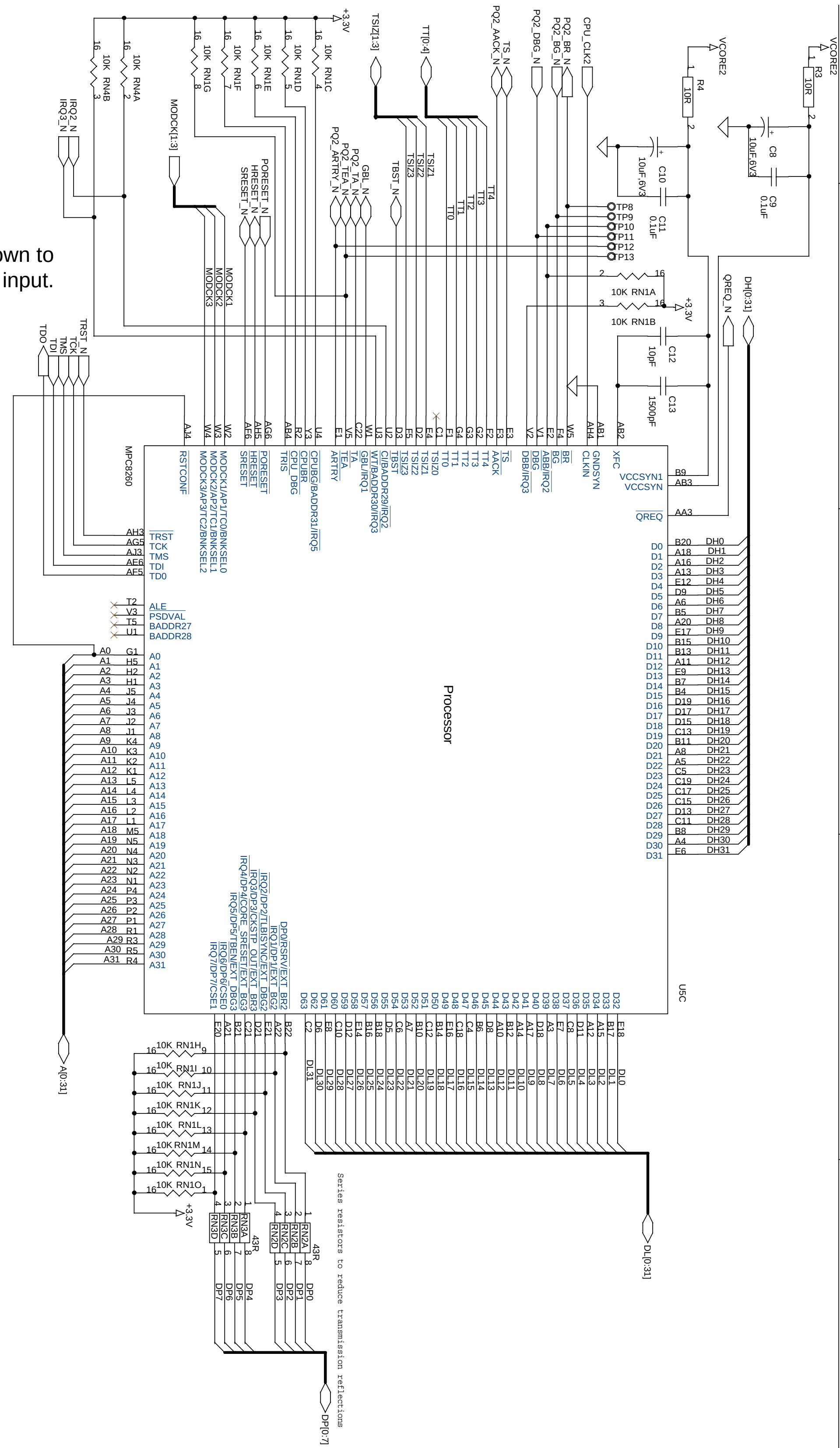

Figure 1-9. MPC8260 CPU

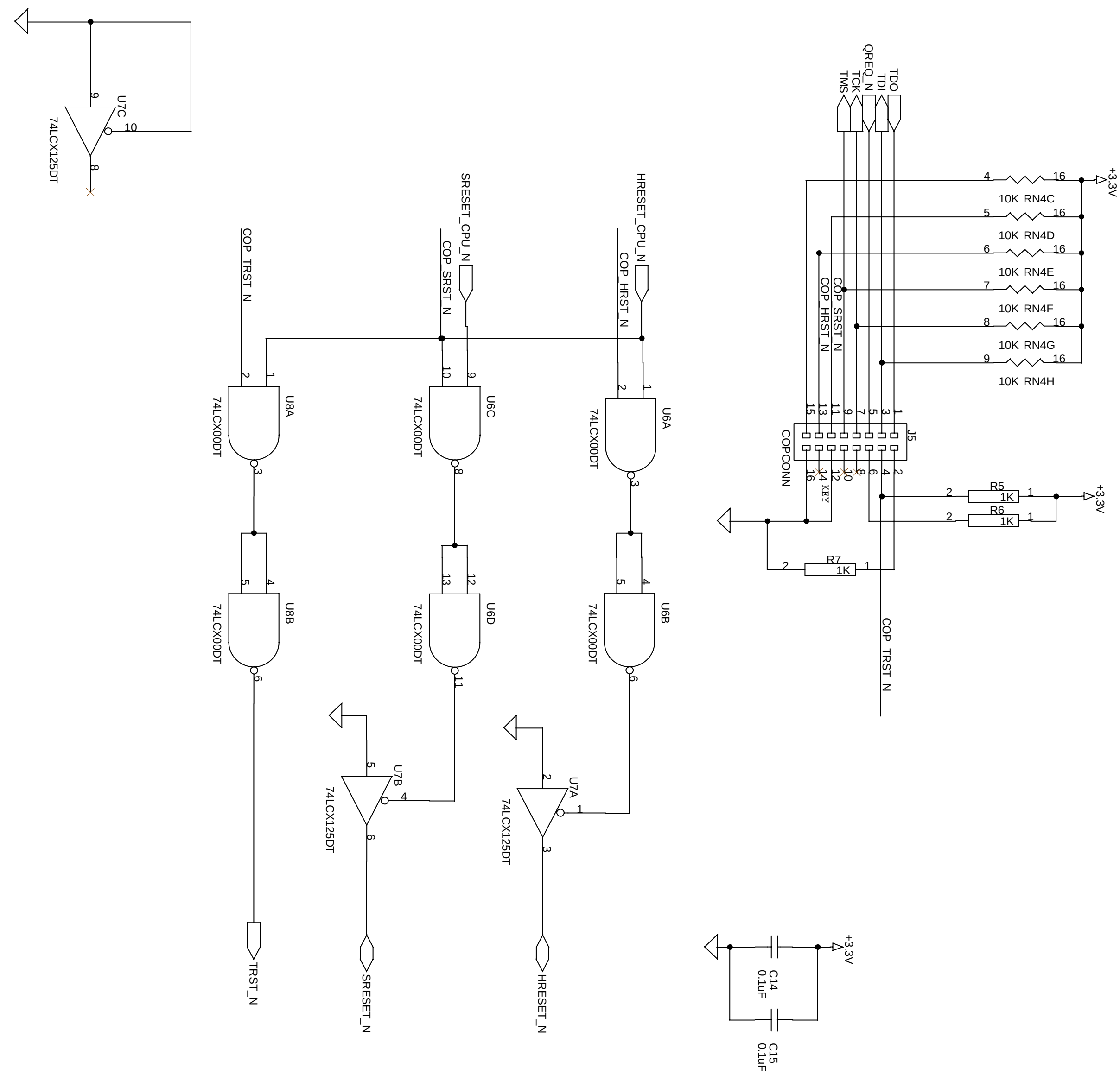

Figure 1-10. MPC8260 COP Interface

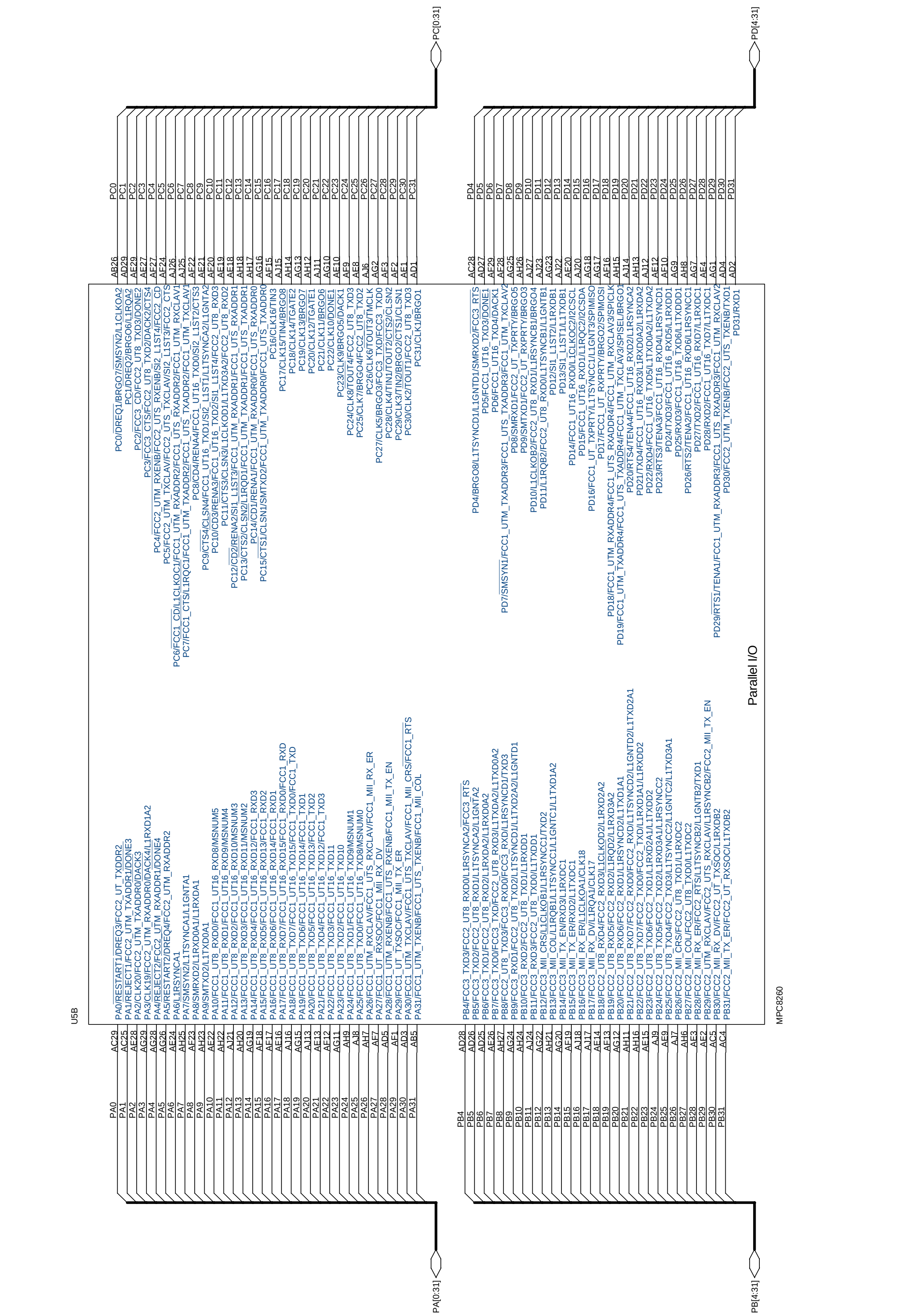

Figure 1-11. MPC8260 Parallel I/O Ports

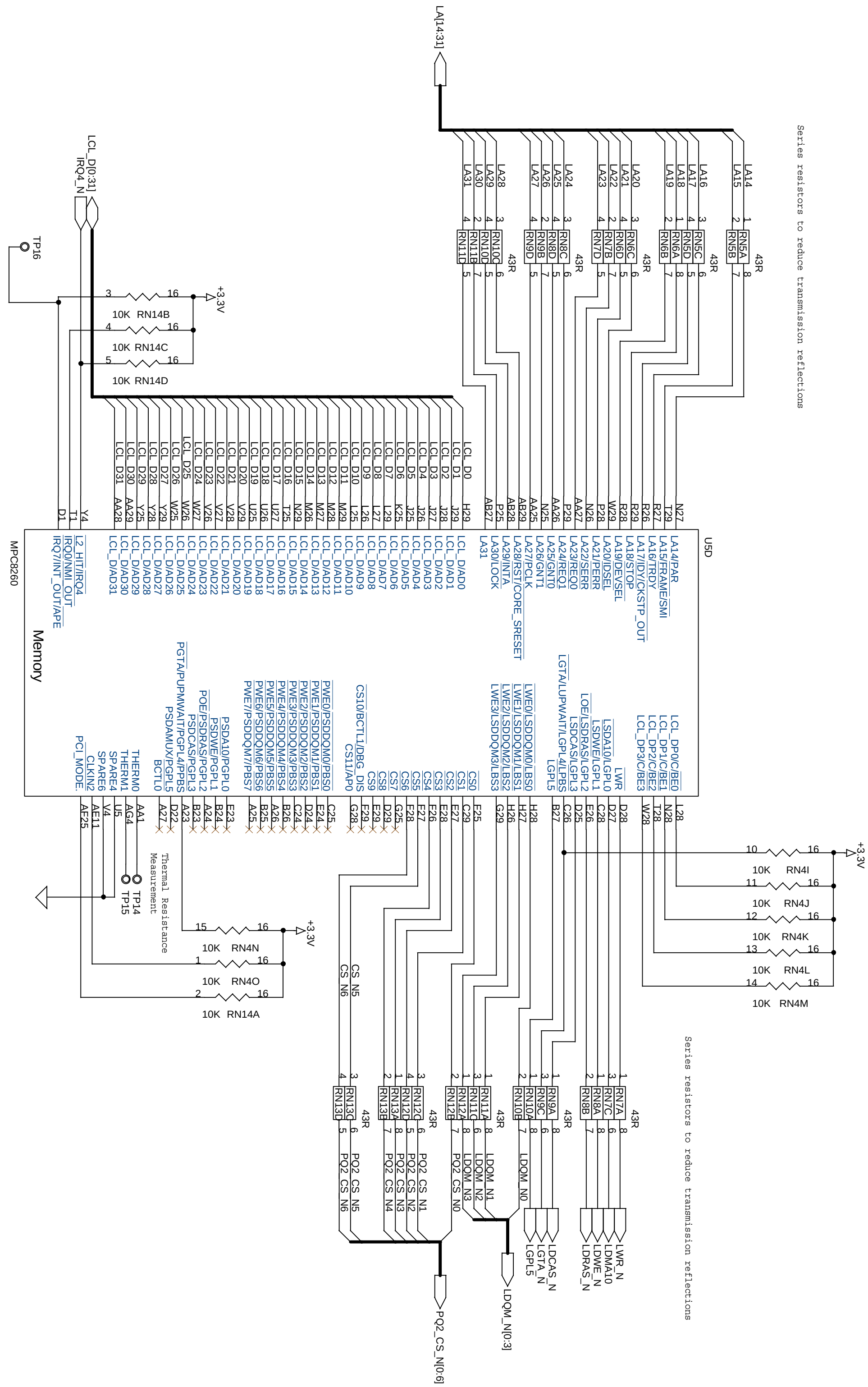

Figure 1-12. MPC8260 Local Memory

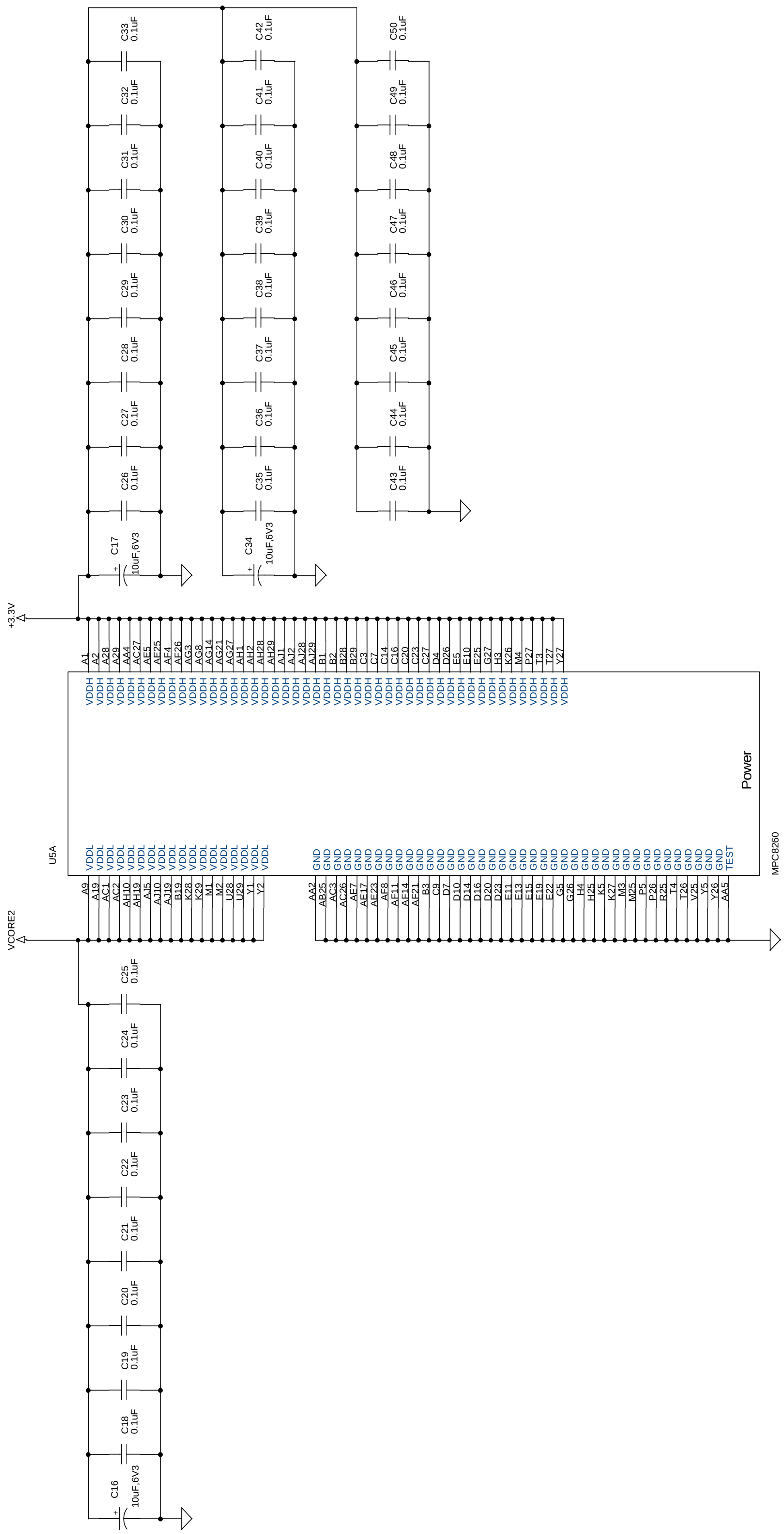

Figure 1-13. MPC8260 Power Connections

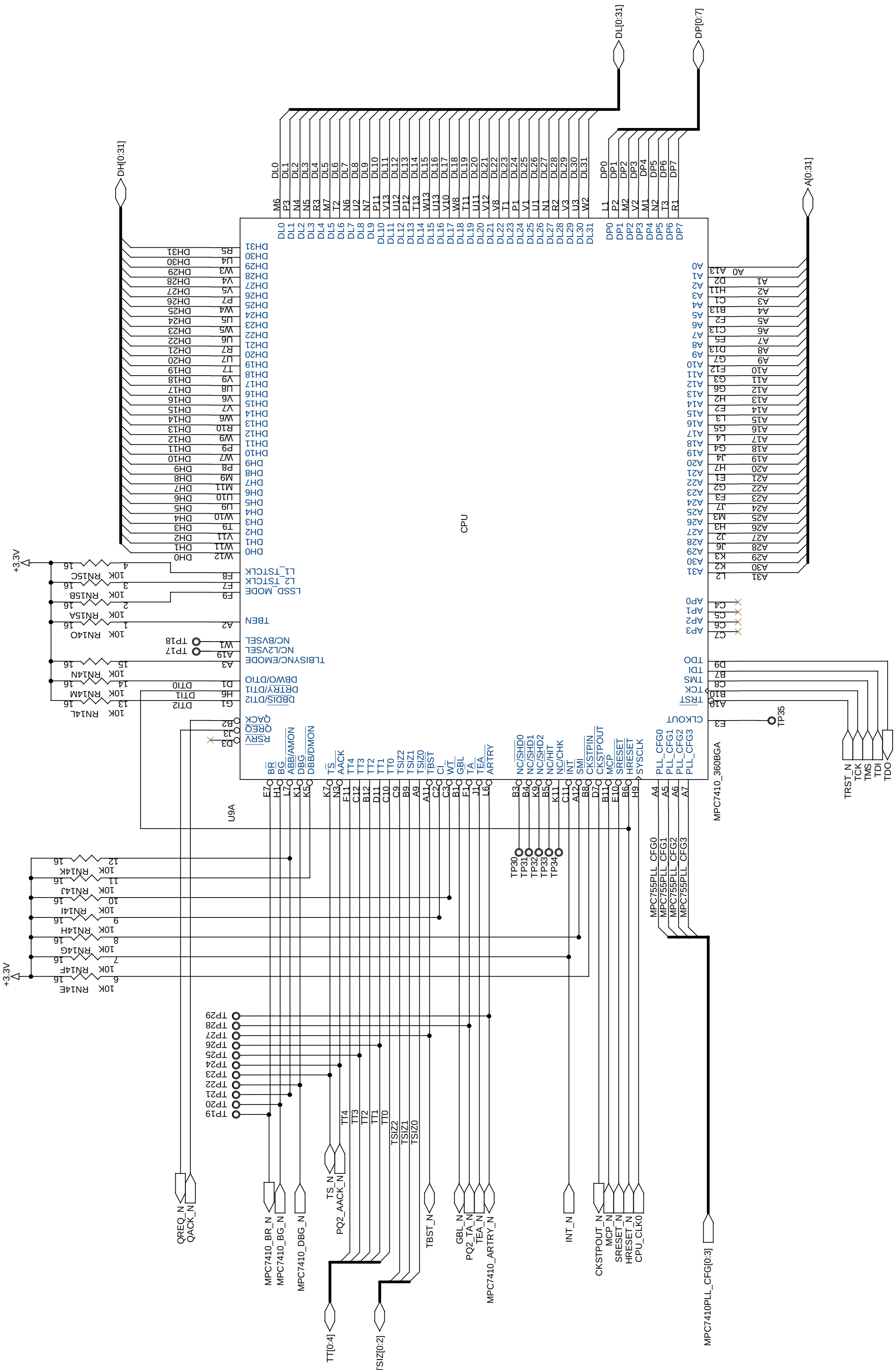

Figure 1-14. MPC7410 CPU

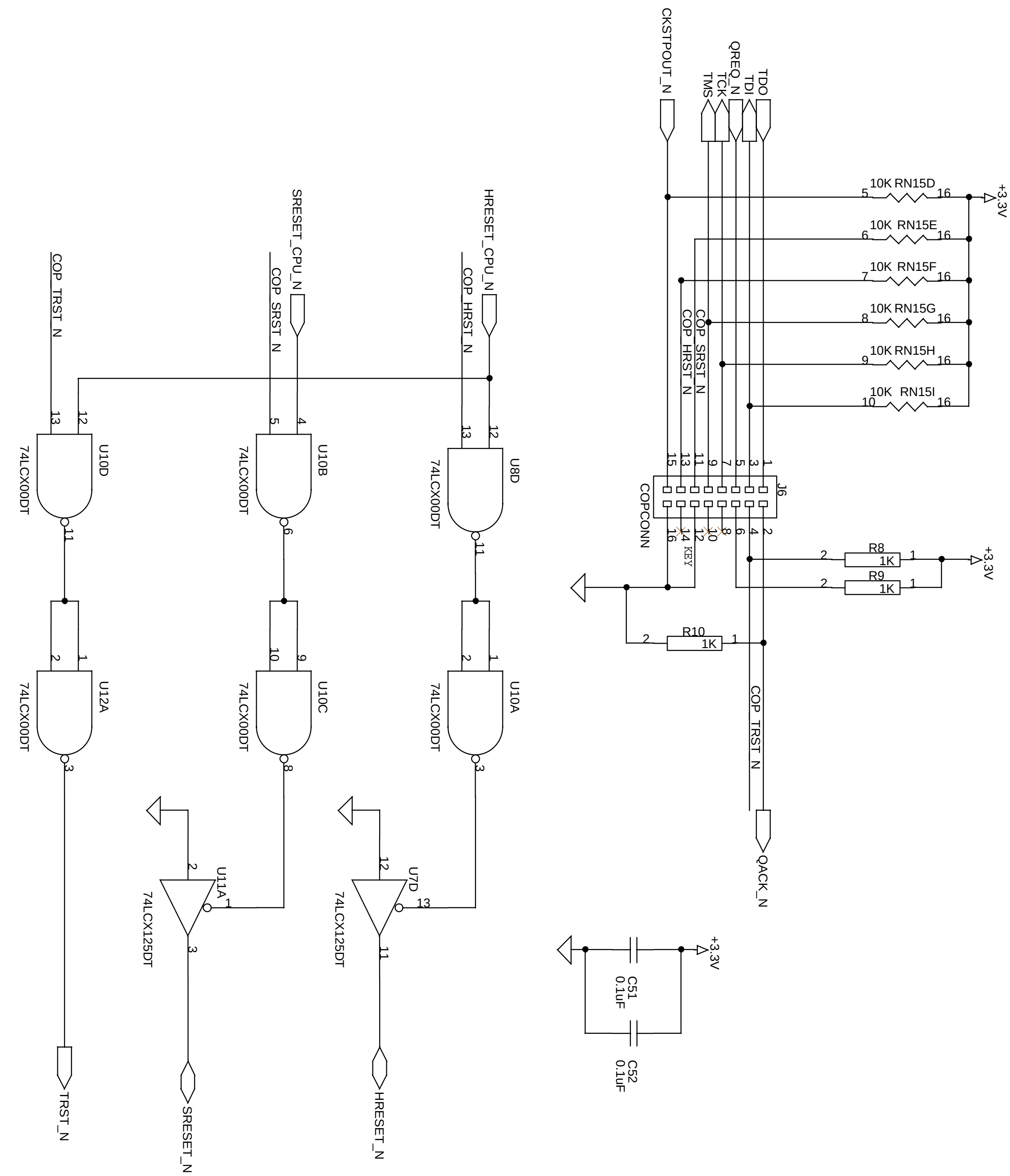

Figure 1-15. MPC7410 COP Interface

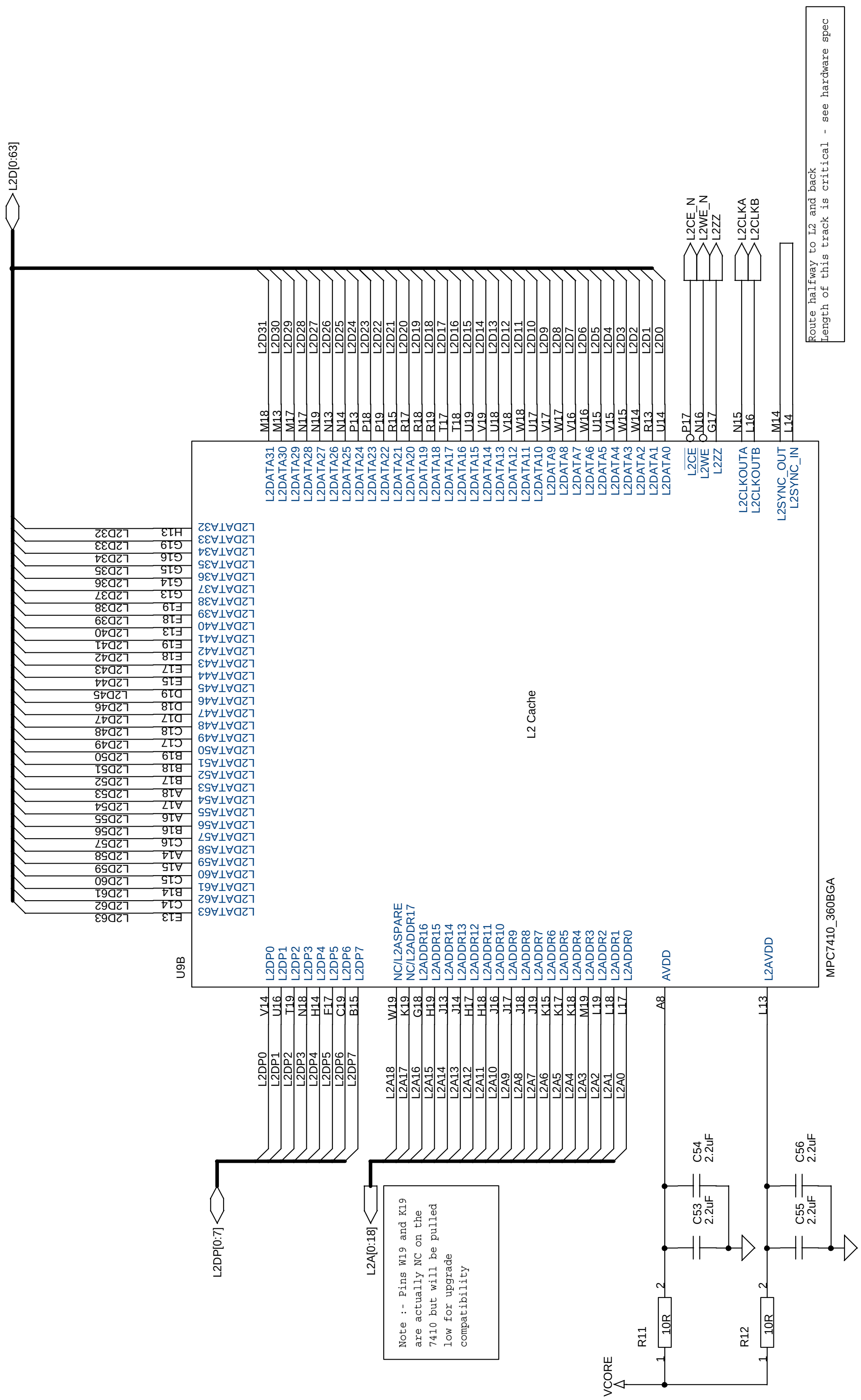

Figure 1-16. MPC7410 L2 Interface

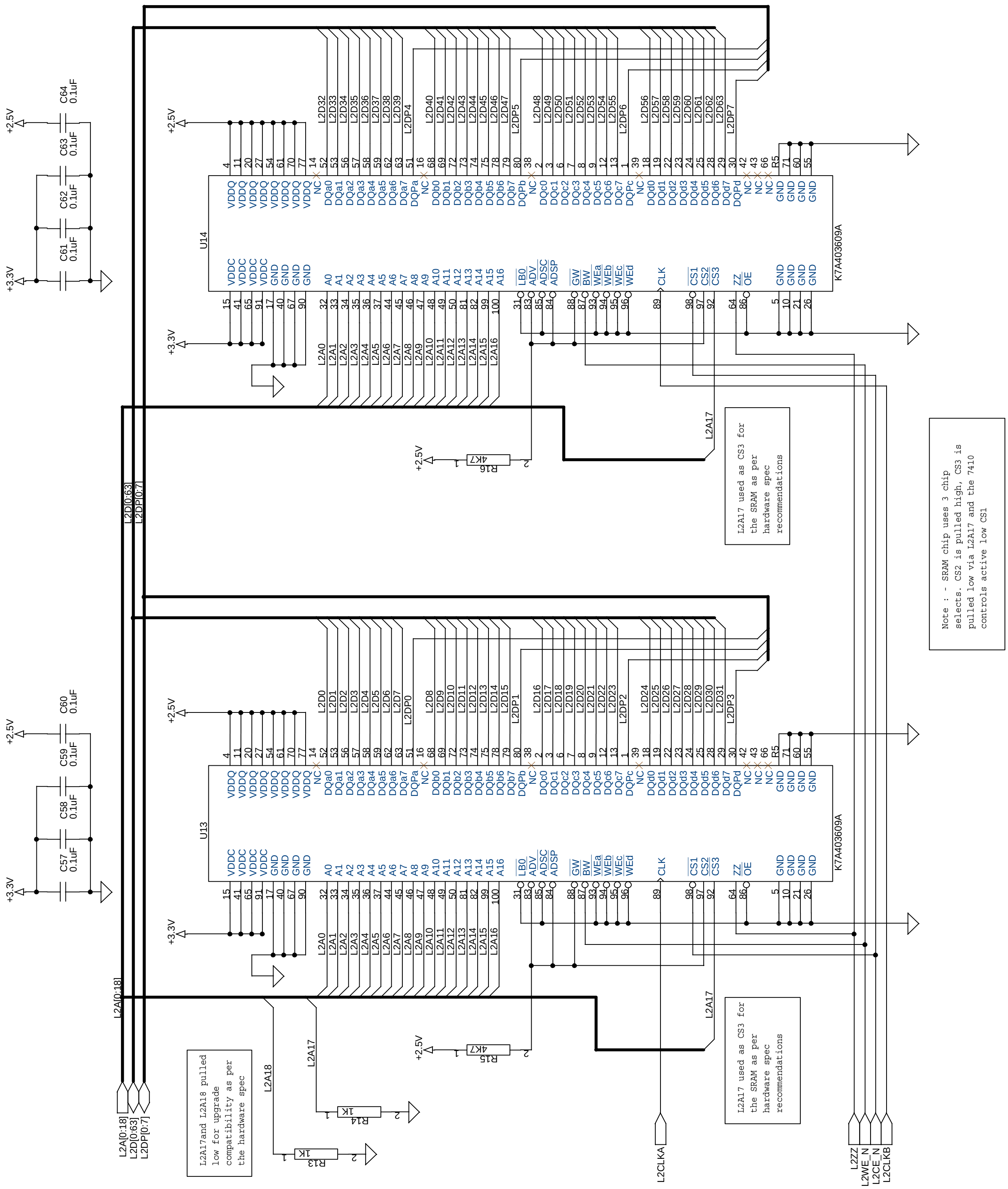

Figure 1-17. MPC7410 L2 Cache SRAM

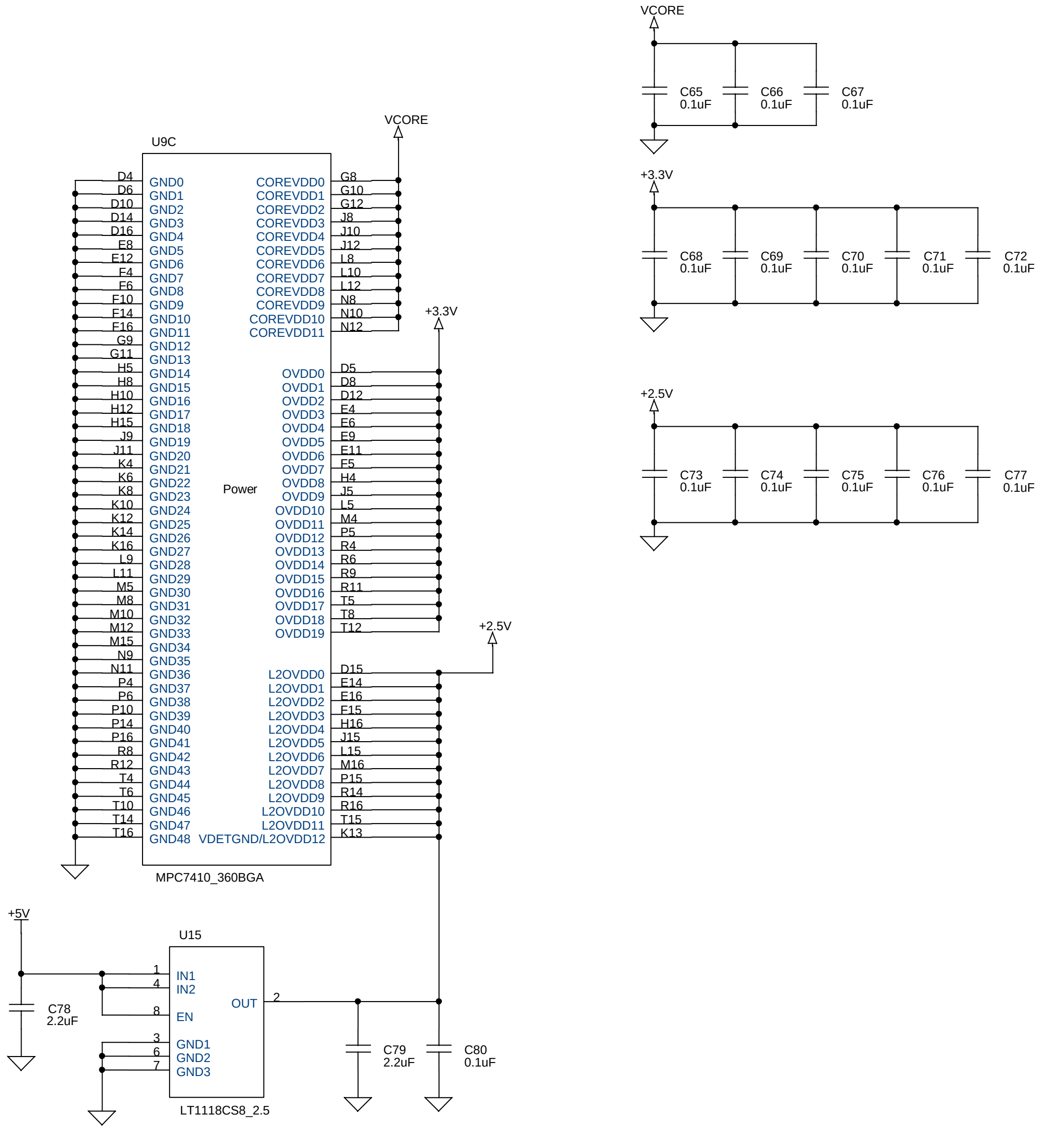

Figure 1-18. MPC7410 Power Connections

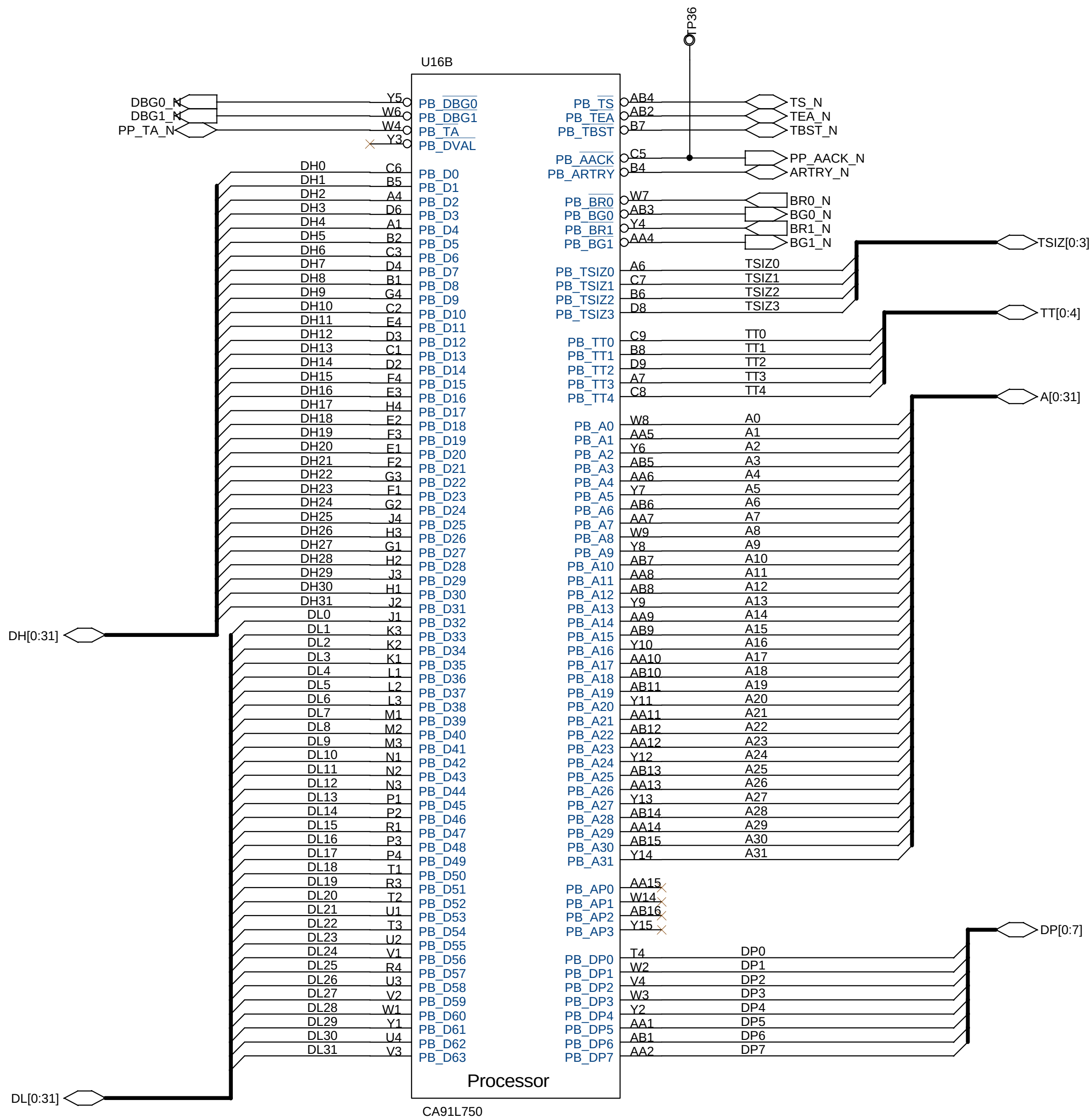

Figure 1-19. Tundra CA91L750 60x Bus Interface

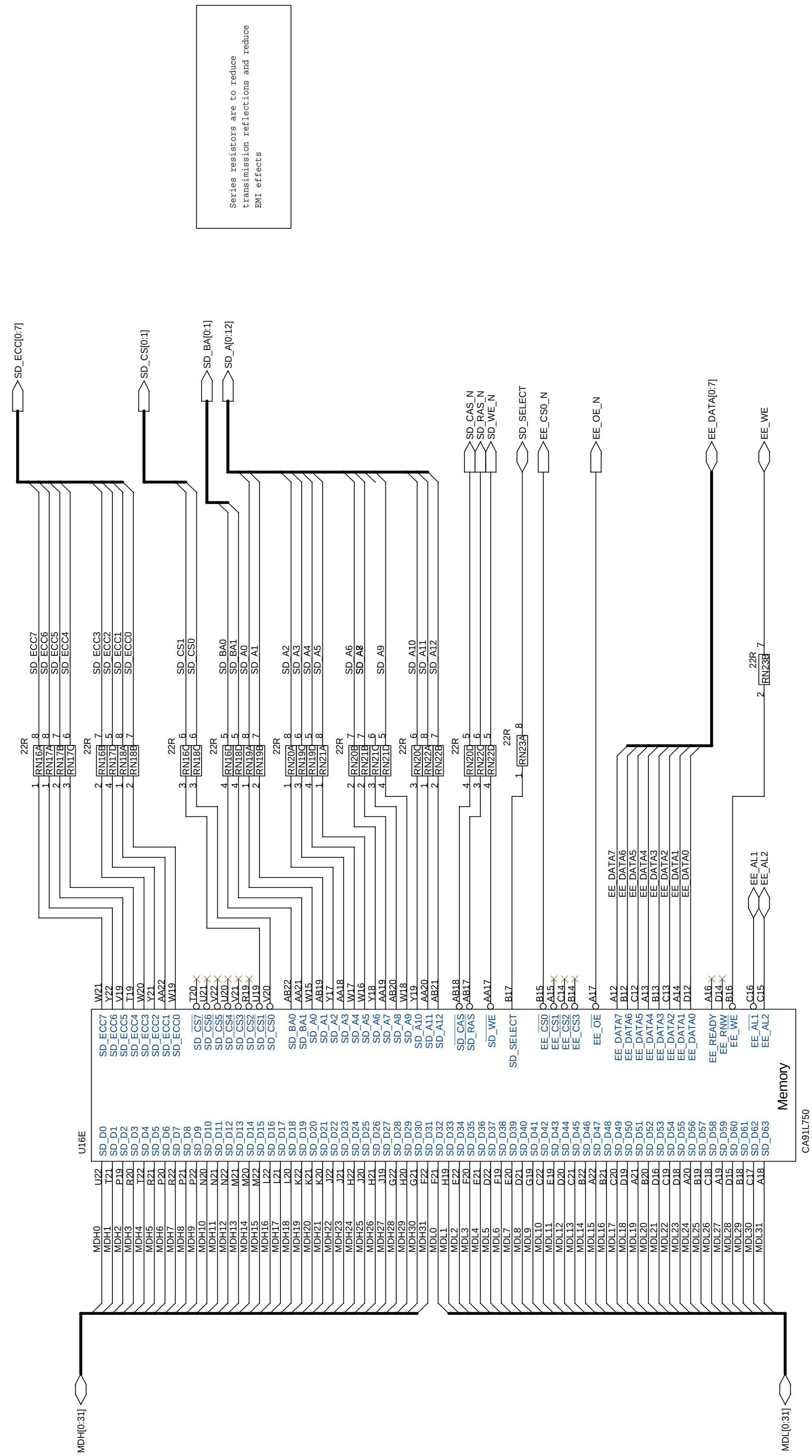

Figure 1-20. Tundra CA91L750 Memory Interface

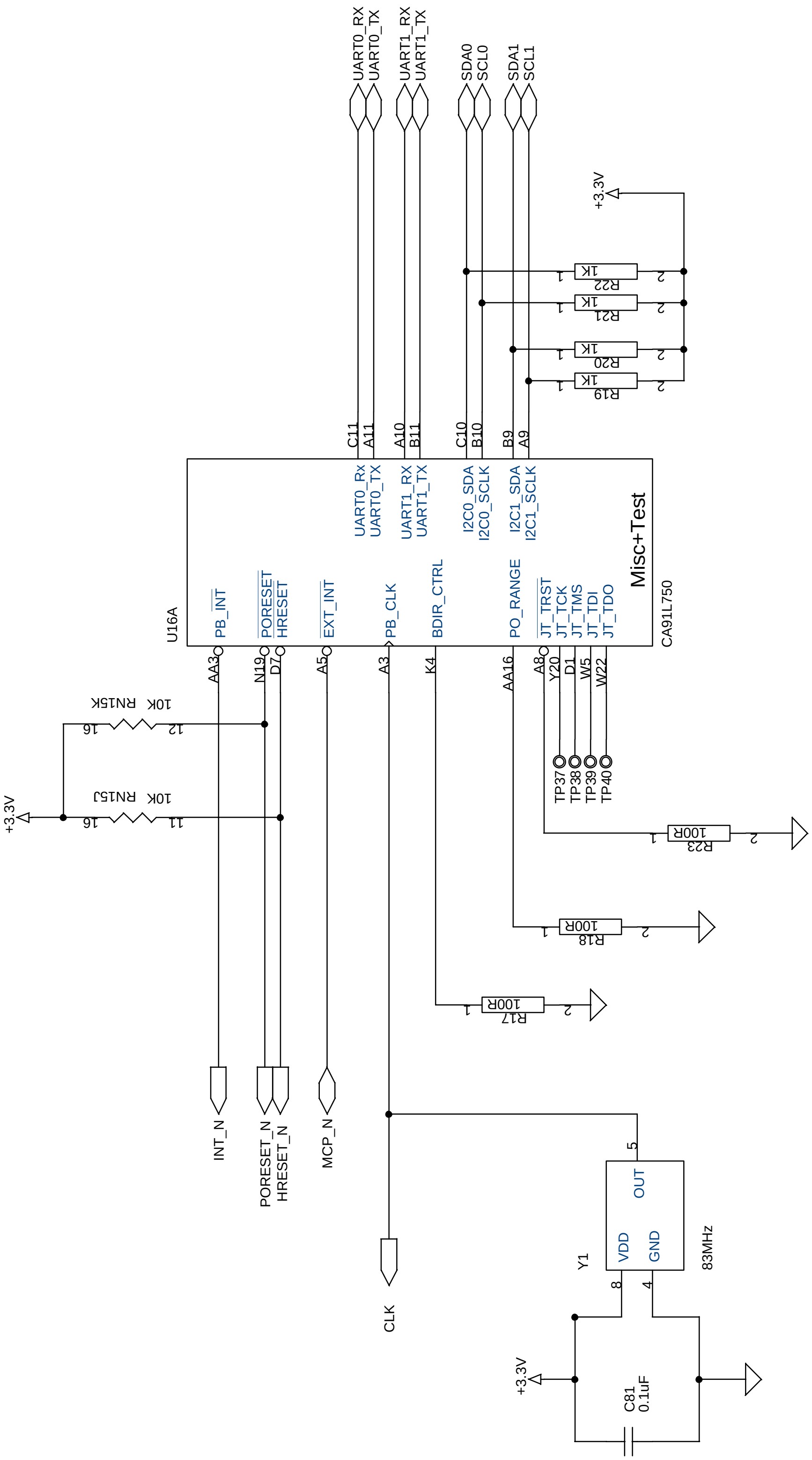

Figure 1-21. Tundra CA91L750 MSC + Test Interface

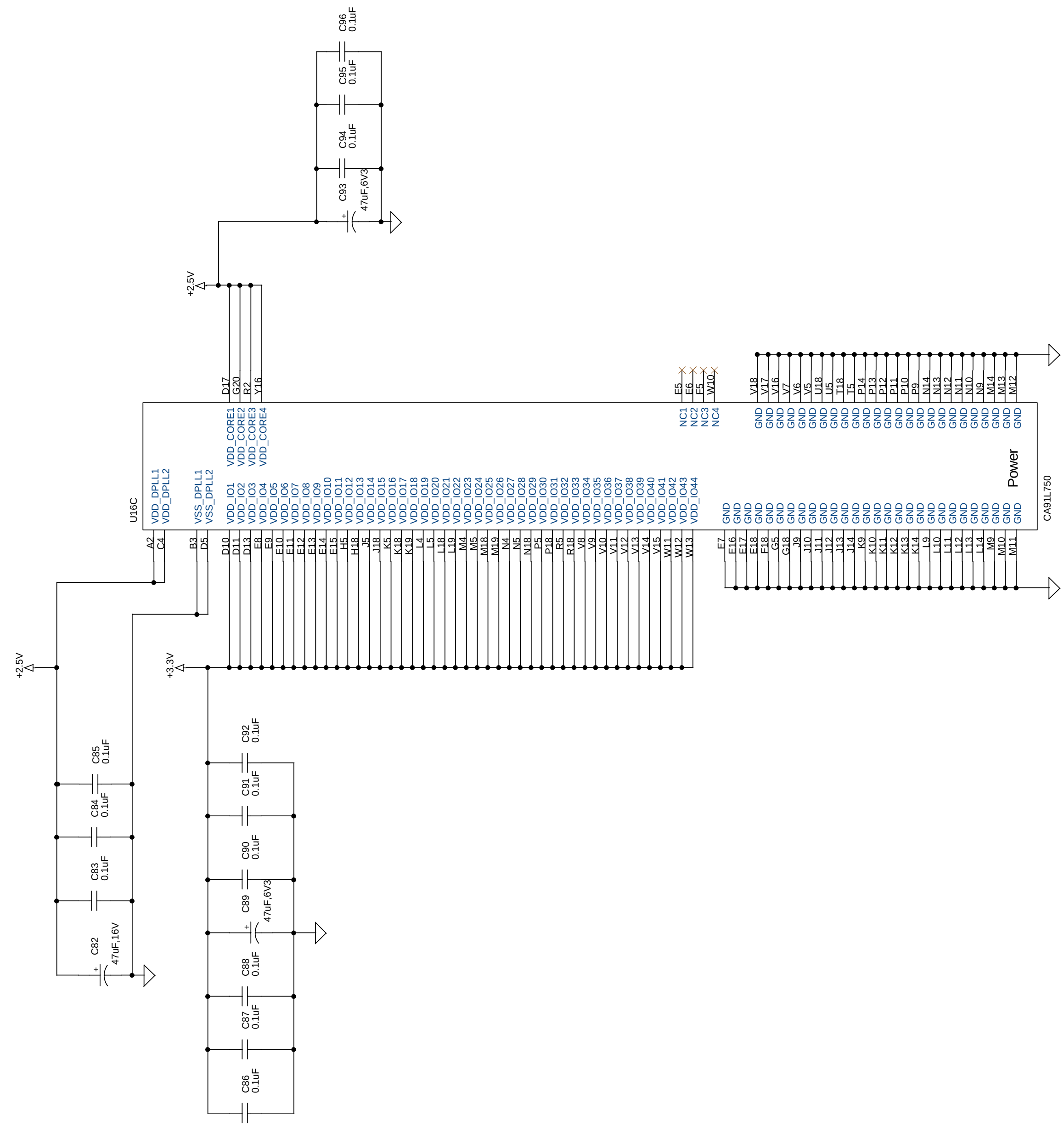

Figure 1-22. Tundra CA91L750 Power Connections

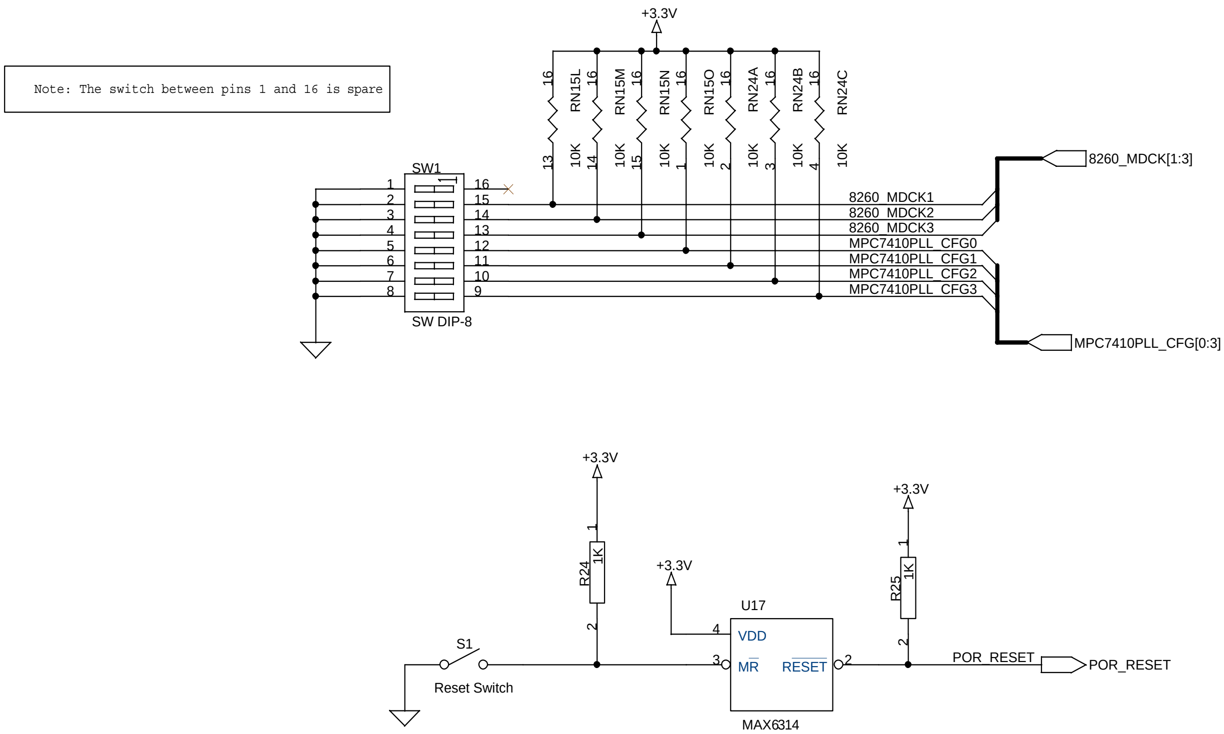

Figure 1-23. POR Reset and Configuration

Appendix B—Initialization Software

This is a complete listing of the initialization code used for the application.

Main Initialization Routine—P2N.2

Start-up Function for an Embedded Environment

```
        .text

        .globl          _start

        .globl          _stack_addr

        .globl          _SDA_BASE_

        .globl          _SDA2_BASE_

        .globl          main

        .globl          mpc8260_main

        .align          2

 .include    "init_cn.h"       # Expanded version of Init8260.h
```

```
 .include    "initPowerPro.h"  # PowerPro register definitions
```

Shared SDRAM Parameters

```
REFINT          .equ    984             # units of Processor bus clocks

_start:

#

# insert other init code here

#

        andis.  r0,r0,0

        andi.   r0,r0,0 # make sure r0 is zero

        mfmsr   r3

        andi.   r3,r3,0xffbf# clear ip bit

        mtmsr   r3

        isync

        sync

        mtspr   hid0,r0 # clear hid0

        isync

        sync
```

Load the following physical address into the Link Register. Jump to that address so that LR equals Program Counter (PC).

```
        addis    r3,0,_sync_jump@h      # load the address

        ori      r3,r3,_sync_jump@l

        mtspr    LR,r3

        bclr     20,0                   # jump unconditionally to address in
                                         Link Register (LR)

_sync_jump:
```

Call PowerPro_init to initialize the PowerPro device. This must be done first to enable the HIT* logic.

```
        bl PowerPro_init         # Initialization code for the PowerPro

cpu1:

        bl mpc7410_init

        /*

         *      initialize stack pointer

         */
```

```
        lis     r1, _stack_addr@h/* _stack_addr is generated by linker */
        ori     r1, r1, _stack_addr@l
        /*
         *      initialize small data area pointers (EABI)
         */
        lis     r2, _SDA2_BASE_@h/* __SDA2_BASE_ is generated by linker */
        ori     r2, r2, _SDA2_BASE_@l
        lis     r13, _SDA_BASE_@h/* _SDA_BASE_ is generated by linker */
        ori     r13, r13, _SDA_BASE_@l
        b  main                   # Jump to test code
mpc8260:
   bl mpc8260_init
        /*
         *      initialize stack pointer
         */
        lis     r1, _stack_addr@h/* _stack_addr is generated by linker */
        ori     r1, r1, _stack_addr@l
        /*
         *      initialize small data area pointers (EABI)
         */
        lis     r2, _SDA2_BASE_@h/* __SDA2_BASE_ is generated by linker */
        ori     r2, r2, _SDA2_BASE_@l
        lis     r13, _SDA_BASE_@h/* _SDA_BASE_ is generated by linker */
        ori     r13, r13, _SDA_BASE_@l
        b  mpc8260_main           # Jump to test code
```

PowerPro_init

This function initializes the PowerPro's bus interface and memories.

```
        mfspr    r31,LR         # Save the Link Register value. The link
                                register's value will be restored so that this
                                function can return to the calling address.


        lis     r7, PowerPro_Base@h     # Set r7 as a pointer to the PowerPro
```

```
  ori      r7, r7, PowerPro_Base@l       # internal register block
addi      r4, r7, PB_GEN_CTRL # Set r4 as pointer to required register
lwarx    r3, 0, r4        # get current contents of register
ori      r3, r3, 0x0005   # turn on even parity
stw      r3, 0, (r4)      # restore register
addi     r4, r7, SD_REFRESH# Set r4 as pointer to required register
addis    r3, 0, 0x0000
ori      r3, r3, REFINT
stw      r3, 0, (r4)      # Initialise register
addi     r4, r7, SD_TIMING# Set r4 as pointer to required register
addis    r3, 0, 0x0002
ori      r3, r3, 0x0000
stw      r3, 0, (r4)      # Initialise register
addi     r4, r7, SD_B0_ADDR# Set r4 as pointer to required register
addis    r3, 0, 0x0000
ori      r3, r3, 0x0001
stw      r3, 0, (r4)      # Initialise register
addi     r4, r7, SD_B0_MASK# Set r4 as pointer to required register
addis    r3, 0, 0xF800
ori      r3, r3, 0x0000
stw      r3, 0, (r4)      # Initialise register
addi     r4, r7, SD_B0_CTRL# Set r4 as pointer to required register
addis    r3, 0, 0xD811
ori      r3, r3, 0x0000
stw      r3, 0, (r4)      # Initialise register
addi      r4, r7, SD_TIMING # Set r4 as pointer to required register
lwarx    r3, 0, r4        # get current contents of register
oris     r3, r3, 0x8000   # enable the SDRAM
stw      r3, 0, (r4)      # restore register
addi     r4, r7, EE_B0_ADDR# Set r4 as pointer to required register
addis    r3, 0, 0xFFF0
ori      r3, r3, 0x0101
```

```
stw      r3, 0, (r4)             # Initialise register
addi     r4, r7, EE_B0_MASK# Set r4 as pointer to required register
addis    r3, 0, 0xFFF0
ori      r3, r3, 0x0000
stw      r3, 0, (r4)      # Initialise register
addi     r4, r7, EE_B0_CTRL# Set r4 as pointer to required register
addis    r3, 0, 0b0001010001100101
ori      r3, r3, 0b0100010010110000
stw      r3, 0, (r4)      # Initialise register
addi     r4, r7, I2C0_CSR # Set r4 as pointer to required register
addis    r3, 0, 0x0000
ori      r3, r3, 0xA000
stw      r3, 0, (r4)      # Initialise register
mtspr    LR, r31          # restore Link Register
bclr     20, 0            # Jump unconditionally to address in LR
```

mpc7410_init

This function initialize the MPC7410.

```
mfspr    r31,LR        # Save the Link Register value. The link register's
                                value will be restored so that this function can
                                return to the calling address.
```

Set up the BATs

```
addis          r3, 0, 0x0000   # Cacheable global 256M from 0
ori            r3, r3, 0x1fff
addis          r4, 0, 0x0000
ori            r4, r4, 0x0012
mtspr          dbat3l, r4
mtspr          dbat3u, r3
isync
addis          r3, 0, 0xf000   # Cacheable 256M from 0xf000_0000
ori            r3, r3, 0x1fff
addis          r4, 0, 0xf000
ori            r4, r4, 0x0002
```

```
        mtspr       ibat0l, r4
        mtspr       ibat0u, r3
        isync
        addis       r3, 0, 0x0000   # Cacheable 256M from 0x0000_0000
        ori         r3, r3, 0x1fff
        addis       r4, 0, 0x0000
        ori         r4, r4, 0x0002
        mtspr       ibat1l, r4
        mtspr       ibat1u, r3
        isync
        addis       r3, 0, 0        # clear unused bats
        mtspr       ibat2u, r3
        mtspr       ibat3u, r3
        isync
        mfmsr   r3              # Switch on translation
        ori     r3, r3, 0x30
        mtmsr   r3
        isync
```

Setup L2Cache

```
        addis   r10, r0, 0
        mtspr   l2cr, r10       ; Clear L2CR
        sync
        addis   r10,r0,(SIZ1M|CLK20|RAMSB|OH05|L2I|L2DO)@ha     ; 1Mb sync
                                ; burst, divide by 2.0, 1ns output hold
        mtspr   l2cr, r10
        isync
L2inv:                          ; Wait on invalidate in progress clear
        mfspr   r10, l2cr
        andi.   r10, r10, L2IP@l
        bne     L2inv
        isync
```

```
addis   r10, r0,(SIZ1M|CLK20|RAMSB|OH05|L2E|L2DO)@ha   ; Set enable bit
mtspr   l2cr, r10
sync
mfspr   r5, hid0        # turn on the D cache.
ori     r6, r5, 0x4400# Data cache only 0x4400
andi.   r5, r6, 0xfbff# clear the invalidate bit 0x4000
mtspr   hid0, r6
isync
sync
mtspr   hid0, r5
isync
sync
mfspr   r5, hid0                # turn on the I cache.
ori     r6, r5, 0x8800          # Instruction cache only! 0x8800
andi.   r5, r6, 0xf7ff          # clear the invalidate bit 0x8000
mtspr   hid0, r6
isync
sync
mtspr   hid0, r5
isync
sync
mtspr   LR, r31         # restore original Link Register value
bclr    20, 0           # jump unconditionally to effective address
                                in Link register
```

mpc8260_init

This function initialize the MPC8260's bus interfaces and memories.

```
mfspr    r31, LR        # Save the Link Register value. The link
                        register's value will be restored so that this
                        function can return to the calling address.
```

Set up the BATs

```
addis   r3, 0, 0xf000   # non-cacheable 256M from 0xf000_0000
ori     r3, r3, 0x1fff
```

```
addis   r4, 0, 0xf000
ori     r4, r4, 0x0022
mtspr   dbat0l, r4
mtspr   dbat0u, r3
isync
addis   r3, 0, 0xc000    # non-cacheable global 256M from 0xc000_0000
ori     r3, r3, 0x1fff
addis   r4, 0, 0xc000
ori     r4, r4,0x0022
mtspr   dbat1l, r4
mtspr   dbat1u, r3
isync
addis   r3, 0, 0x2000    # Cacheable global 256M from 0x2000_0000
ori     r3, r3, 0x1fff
addis   r4, 0, 0x2000
ori     r4, r4, 0x0012
mtspr   dbat2l, r4
mtspr   dbat2u, r3
isync
addis   r3, 0, 0x0000    # Cacheable global 256M from 0
ori     r3, r3, 0x1fff
addis   r4, 0, 0x0000
ori     r4, r4, 0x0012
mtspr   dbat3l, r4
mtspr   dbat3u, r3
isync
addis   r3, 0, 0xf000            # Cacheable 256M from 0xf000_0000
ori     r3, r3, 0x1fff
addis   r4, 0, 0xf000
ori     r4, r4, 0x0002
mtspr   ibat0l, r4
mtspr   ibat0u, r3
```

```
isync
addis   r3, 0,0x0000            # Cacheable global 256M from 0
ori     r3, r3, 0x1fff
addis   r4, 0, 0x0000
ori     r4, r4, 0x0002
mtspr   ibat1l, r4
mtspr   ibat1u, r3
isync
addis   r3, 0, 0                # clear unused bats
mtspr   ibat2u, r3
mtspr   ibat3u, r3
isync
mfmsr   r3                      # Switch on translation
ori     r3, r3, 0x30
mtmsr   r3
isync
mfspr   r5, hid0                # turn on the D cache.
ori     r6, r5, 0x4400          # Data cache only 0x4400
andi.   r5, r6, 0xfbff          # clear the invalidate bit 0x4000
mtspr   hid0, r6
mtspr   hid0, r5
isync
sync
mfspr   r5, hid0                # turn on the I cache.
ori     r6, r5, 0x8800          # Instruction cache only! 0x8800
andi.   r5, r6, 0xf7ff          # clear the invalidate bit 0x8000
mtspr   hid0, r6
mtspr   hid0, r5
isync
sync
```

Load the IMMR register with the base address

```
        addis   r4, 0, 0xF000           # IMMR base addr = 0xF0000000
        sync
        addis   r4, 0, 0xF001   # IMMR base addr = 0xF0000000, therefore
                                        registers start at 0xF0010000
#       bl init_siu
```

Configure EPIC

```
        addis   r4, r0, 0xf804
        ori     r4, r4, 0x1020
        addis   r5, r0, 0x2000
        stwbrx  r5, r0, r4              # set mixed mode

        mtspr   LR, r31                 # restore original Link Register value

        bclr    20, 0                   # jump unconditionally to effective
                                                address in Link register
```

init_siu

This function initialize the MPC8260's SIU.

Leave BCR as defined by hard reset configuration (which programs it for 60x mode)

Go ahead and disable the s/w watchdog just to ease my mind (although it shouldn't have time to go off in a simulation.)

```
        addis   r3, 0, 0xFFFF
        ori     r3, r3, 0xFFC0          # SYPCR = 0xFFFFFFC0
        stw     r3, SYPCR(r4)
        sync
```

For SIUMCR: ESE=1 (enable GBL), DPPC=01, L2CPC=01 (IRQ function), CS10PC=01 (BCTL1 function), BCTLC=01 (W/R buffers)

```
        addis   r3, 0, 0x4505
        stw     r3, SIUMCR(r4)
        sync
        bclr    20, 0           # jump unconditionally to effective address in
                                        Link register
```

Required Header File—init_cn.h

On-Chip Core Registers

These values represent the special purpose registers used in this example. Refer to *PowerPC Microprocessor Family: The Programming Environments for 32-Bit Microprocessors* manual for the complete set and the *MPC8260 Users Manual*.

L2CR Bit Settings

```
# high word

L2E:            .equ    0x80000000

L2PE:           .equ    0x40000000

SIZ256K:        .equ    0x10000000

SIZ512K:        .equ    0x20000000

SIZ1M:          .equ    0x30000000

CLK10:          .equ    0x02000000

CLK15:          .equ    0x04000000

CLK20:          .equ    0x08000000

CLK25:          .equ    0x0a000000

CLK30:          .equ    0x0c000000

RAMFT:          .equ    0x00000000

RAMSB:          .equ    0x01000000

RAMLW:          .equ    0x01800000

L2DO:           .equ    0x00400000

L2I:            .equ    0x00200000

L2CTL:          .equ    0x00100000

L2WT:           .equ    0x00080000

L2TS:           .equ    0x00040000

OH05:           .equ    0x00000000

OH10:           .equ    0x00010000

OH12:           .equ    0x00020000

OH14:           .equ    0x00030000

# low word

L2SL:           .equ    0x8000
```

```
L2DF:           .equ    0x4000
L2BYP:          .equ    0x2000
L2IP:           .equ    0x0001
```

MSR Bit Settings

```
# High word
POW:            .equ    0x00040000
ILE:            .equ    0x00010000
# Low word
EE:             .equ    0x8000
PR:             .equ    0x4000
FP:             .equ    0x2000
ME:             .equ    0x1000
FE0:            .equ    0x0800
SE:             .equ    0x0400
BE:             .equ    0x0200
FE1:            .equ    0x0100
IP:             .equ    0x0040
IR:             .equ    0x0020
DR:             .equ    0x0010
PM:             .equ    0x0004
RI:             .equ    0x0002
LE:             .equ    0x0001
```

HID0 Bit Settings

```
# high word
EMCP:           .equ    0x80000000
DBP:            .equ    0x40000000
EBA:            .equ    0x20000000
EBD:            .equ    0x10000000
BCLK:           .equ    0x08000000
ECLK:           .equ    0x02000000
PAR:            .equ    0x01000000
```

```
DOZE:           .equ    0x00800000
NAP:            .equ    0x00400000
SLEEP:          .equ    0x00200000
DPM:            .equ    0x00100000
NHR:            .equ    0x00010000
# low word
ICE:            .equ    0x8000
DCE:            .equ    0x4000
ILOCK:          .equ    0x2000
DLOCK:          .equ    0x1000
ICFI:           .equ    0x0800
DCFI:           .equ    0x0400
SPD:            .equ    0x0200
IFEM:           .equ    0x0100
SGE:            .equ    0x0080
DCFA:           .equ    0x0040
BTIC:           .equ    0x0020
ABE:            .equ    0x0008
BHT:            .equ    0x0004
NOOPTI:         .equ    0x0001

 C0:      .equ     0
 XER:     .equ     1     #
 LR:      .equ     8     # Link Register
 CTR:     .equ     9     # Counter Register
 DSISR:   .equ     18    # Cause of Data Access and Alignment Exceptions
 DAR:     .equ     19    # Data Address Register
 DEC_R:   .equ     22    # Decrementer Register
 SDR1:    .equ     25    # Page Table Format Register
 SRR0:    .equ     26    # Save/Restore Register 0
 SRR1:    .equ     27    # Save/Restore Register 1
 EAR:     .equ     282   # External Address Register
```

```
PVR:      .equ      287   # Processor Version Register

SPRG0:    .equ      272   # Special Purpose Register General 0
SPRG1:    .equ      273   # Special Purpose Register General 1
SPRG2:    .equ      274   # Special Purpose Register General 2
SPRG3:    .equ      275   # Special Purpose Register General 3

TBL_W:    .equ      284   # Time Base Write Register Lower
TBU_W:    .equ      285   # Time Base Write Register Upper
TBL_R:    .equ      284   # Time Base Read Register Lower
TBU_R:    .equ      285   # Time Base Read Register Upper
IBAT0U:   .equ      528   # Instruction BAT Register Upper 0
IBAT0L:   .equ      529   # Instruction BAT Register Lower 0
IBAT1U:   .equ      530   # Instruction BAT Register Upper 1
IBAT1L:   .equ      531   # Instruction BAT Register Lower 1
IBAT2U:   .equ      532   # Instruction BAT Register Upper 2
IBAT2L:   .equ      533   # Instruction BAT Register Lower 2
IBAT3U:   .equ      534   # Instruction BAT Register Upper 3
IBAT3L:   .equ      535   # Instruction BAT Register Lower 3
DBAT0U:   .equ      536   # Data BAT Register Upper 0
DBAT0L:   .equ      537   # Data BAT Register Lower 0
DBAT1U:   .equ      538   # Data BAT Register Upper 1
DBAT1L:   .equ      539   # Data BAT Register Lower 1
DBAT2U:   .equ      540   # Data BAT Register Upper 2
DBAT2L:   .equ      541   # Data BAT Register Lower 2
DBAT3U:   .equ      542   # Data BAT Register Upper 3
DBAT3L:   .equ      543   # Data BAT Register Lower 3

DMISS:    .equ      976   # Data TLB Miss Register
DCMP:     .equ      977   # Data PTE Compare Register
HASH1:    .equ      978   # Primary Hash Address Register
HASH2:    .equ      979   # Secondary Hash Address Register
```

```
IMISS:    .equ     980   # Instruction TLB Miss Register
ICMP:     .equ     981 # Instruction PTE Compare Register
RPA:      .equ     982 # Required Physical Address Register
HID0:     .equ     1008 # Hardware Implementation Register 0
HID1:     .equ     1009 # Hardware Implementation Register 1
```

Register Offset Definitions

All these values are offsets from the Internal Memory Map Register (IMMR) base pointer. The base value is determined by the ISB bits in the Hard Reset Configuration Word. See the reset section of the user's manual. For a complete set of all the IMMR registers, refer to the *MPC8260 Users Manua*l. As an additional note, only those register values used in this example are listed below. Therefore this list does not comprise all possible Internal Memory Mapped Registers.

```
SIUMCR:   .equ     0x0000   # SIU Module Configuration Register
SYPCR:    .equ     0x0004   # System Protection Control Register
SCCR:     .equ     0x0C80   # System Clock Control Register
BR0:      .equ     0x0100   # Base Register Bank 0
OR0:      .equ     0x0104   # Option Register Bank 0
BR1:      .equ     0x0108   # Base Register Bank 1
OR1:      .equ     0x010C   # Option Register Bank 1
BR2:      .equ     0x0110   # Base Register Bank 2
OR2:      .equ     0x0114   # Option Register Bank 2
BR3:      .equ     0x0118   # Base Register Bank 3
OR3:      .equ     0x011C   # Option Register Bank 3
BR4:      .equ     0x0120   # Base Register Bank 4
OR4:      .equ     0x0124   # Option Register Bank 4
BR8:      .equ     0x0140   # Base Register Bank 8
OR8:      .equ     0x0144   # Option Register Bank 8
MPTPR:    .equ     0x0184   # Memory Periodic Timer Prescaler Register
PSDMR:    .equ     0x0190   # PowerPC Bus SDRAM Machine Mode Register
LSDMR:    .equ     0x0194   # Local Bus SDRAM Machine Mode Register
PSRT:     .equ     0x019C   # 60x Bus Assigned SDRAM Refresh Timer
LSRT:     .equ     0x01A4   # Local Bus Assigned SDRAM Refresh Timer
IMMR:     .equ     0x01A8   # Internal I/O base Register offset
BCR:      .equ     0x0024   # Bus Configuration Register
```

```
PPC_ACR: .equ      0x0028   # 60x Bus Arbiter Configuration Register
MBMR:    .equ      0x0174   # Machine B Mode Register
MDR:     .equ      0x0188   # Memory Data Register
IDMR1:   .equ      0x1024   # IDMA1 mask register
RCCR:    .equ      0x19C4   # CP configuration register
SIMR_H:  .equ      0x0C1C   # SIU interrupt mask register (high)
SIMR_L:  .equ      0x0C20   # SIU interrupt mask register (low)
CPCR:    .equ      0x19C0   # CP command register
```

IDMA Parameter RAM Definitions

```
IBASE:   .equ      0x0000   # IDMA BD Base
DCM:     .equ      0x0002   # DMA channel mode
IBDPTR:  .equ      0x0004   # IDMA BD pointer
DPR_BUF: .equ      0x0006   # IDMA buffer in DPRAM
SS_MAX:  .equ      0x000A   # Steady-state maximum IDMA transfer size
STS:     .equ      0x000E   # Source transfer size
DTS:     .equ      0x0016   # Destination transfer size
ISTATE:  .equ      0x0028   # Internal IDMA state
```

Instruction and Data Cache Definition

Note: must load into bits 0–15.

Interrupt StackFrame Definitions

```
R0_OFFSET:     .equ     (0*4)    # R0 Stack Offset
R1_OFFSET:     .equ     (1*4)    # R1 Stack Offset
R2_OFFSET:     .equ     (2*4)    # R2 Stack Offset
R3_OFFSET:     .equ     (3*4)    # R3 Stack Offset
R4_OFFSET:     .equ     (4*4)    # R4 Stack Offset
R5_OFFSET:     .equ     (5*4)    # R5 Stack Offset
R6_OFFSET:     .equ     (6*4)    # R6 Stack Offset
R7_OFFSET:     .equ     (7*4)    # R7 Stack Offset
R8_OFFSET:     .equ     (8*4)    # R8 Stack Offset
R9_OFFSET:     .equ     (9*4)    # R9 Stack Offset
R10_OFFSET:    .equ     (10*4)   # R10 Stack Offset
```

```
R11_OFFSET:    .equ    (11*4)  # R11 Stack Offset
R12_OFFSET:    .equ    (12*4)  # R12 Stack Offset
SRR0_OFFSET:   .equ    (13*4)  # SRR0 Stack Offset
SRR1_OFFSET:   .equ    (14*4)  # SRR1 Stack Offset
LR_OFFSET:     .equ    (15*4)  # LR Stack Offset
CTR_OFFSET:    .equ    (16*4)  # CTR Stack Offset
XER_OFFSET:    .equ    (17*4)  # XER Stack Offset
CR_OFFSET:     .equ    (18*4)  # CR Stack Offset
STACK_SZ:      .equ    (20*4)   # Quad-Word Aligned Interrupt Stack Frame
C_FRAME_SZ:    .equ    (4*4)    # Quad-Word Aligned C Frame Size
```

Buffers Section

```
IDMABD1:           .equ   0
IDMABD2:           .equ   16
IDMABD3:           .equ   32
IDMABD4:           .equ   48
IDMABD5:           .equ   64
IDMABD6:           .equ   80
IDMABD7:           .equ   96
IDMABD8:           .equ   112
IDMABD9:           .equ   128
IDMABD10:          .equ   144
IDMABD11:          .equ   160
```

Required Header File—initPowerPro.h

Default Value of PowerPro Internal Registers after Power on Reset

```
PowerPro_Base   .equ    0xFFFFFE00#This value is placed in PB_REG_ADDR at reset
```

PowerPro Register Map

Offsets of registers relative to value in PB_REG_ADDR

Processor Bus registers

```
PB_REG_ADDR:   .equ    0x000     # PB Base Address
```

```
PB_GEN_CTRL:   .equ    0x004     # PB General Control and Configuration
PB_ARB_CTRL:   .equ    0x008     # PB Arbiter Control
PB_ERR_ATTR:   .equ    0x00C     # PB Error Log - Attributes
PB_ERR_ADDR:   .equ    0x010     # PB Error Log - Address
PB_AM:         .equ    0x014     # PB Address Match - Address
PB_AM_MASK:    .equ    0x018     # PB Address Match - Mask
VERSION_REG:   .equ    0x01C     # PowerPro Version Register
TEST_MODE_SELECT:.equ  0x01F     # Test Mode Select
```

SDRAM Registers

```
SD_REFRESH:    .equ    0x020     # SDRAM Refresh interval
SD_TIMING:     .equ    0x024     # SDRAM Timing Adjustment
PLL_FB_TUNE:   .equ    0x028     # PLL Feedback
                       0x032-03C PowerPro reserved

SD_B0_ADDR:    .equ    0x040     # SDRAM Bank 0 Base Address
SD_B0_MASK:    .equ    0x044     # SDRAM Bank 0 Base Address Compare Mask
SD_B0_CTRL:    .equ    0x048     # SDRAM Bank 0 Control
#                      0x04C   Reserved

SD_B1_ADDR:    .equ    0x050     # SDRAM Bank 1 Base Address
SD_B1_MASK:    .equ    0x054     # SDRAM Bank 1 Base Address Compare Mask
SD_B1_CTRL:    .equ    0x058     # SDRAM Bank 1 Control
#                      0x05C   Reserved

SD_B2_ADDR:    .equ    0x060     # SDRAM Bank 2 Base Address
SD_B2_MASK:    .equ    0x064     # SDRAM Bank 2 Base Address Compare Mask
SD_B2_CTRL:    .equ    0x068     # SDRAM Bank 2 Control
#                      0x06C   Reserved

SD_B3_ADDR:    .equ    0x070     # SDRAM Bank 3 Base Address
SD_B3_MASK:    .equ    0x074     # SDRAM Bank 3 Base Address Compare Mask
SD_B3_CTRL:    .equ    0x078     # SDRAM Bank 3 Control
```

```
#                       0x07C   Reserved
```

FLASH/ROM Registers

```
EE_B0_ADDR:    .equu    0x080     # ROM Bank 0 Base Address
EE_B0_MASK:    .equ     0x084     # ROM Bank 0 Base Address Compare Mask
EE_B0_CTRL:    .equ     0x088     # ROM Bank 0 Control
#                       0x08C   Reserved

EE_B1_ADDR:    .equ     0x090     # ROM Bank 1 Base Address
EE_B1_MASK:    .equ     0x094     # ROM Bank 1 Base Address Compare Mask
EE_B1_CTRL:    .equ     0x098     # ROM Bank 1 Control
#                       0x09C   Reserved

EE_B2_ADDR:    .equ     0x0A0     # ROM Bank 2 Base Address
EE_B2_MASK:    .equ     0x0A4     # ROM Bank 2 Base Address Compare Mask
EE_B2_CTRL:    .equ     0x0A8     # ROM Bank 2 Control
#                       0x0AC   Reserved

EE_B3_ADDR:    .equ     0x0B0     # ROM Bank 3 Base Address
EE_B3_MASK:    .equ     0x0B4     # ROM Bank 3 Base Address Compare Mask
EE_B3_CTRL:    .equ     0x0B8     # ROM Bank 3 Control
#                       0x0BC   Reserved

I2C0_CSR:      .equ     0x0C0     #I2C interface 0 (primary) Control and Status
I2C1_CSR:      .equ     0x0C4   #I2C interface 1 (secondary) Control and Status

#                       0x0C8-0x0EC PowerPro reserved
```

Watchdog Timer Registers

```
WD_CTRL:       .equ     0x0F0     # Watchdog Timer Control
WD_TIMEOUT:    .equ     0x0F4     # Watchdog Timer Time-out Value
WD_COUNT:      .equ     0x0F8     # Watchdog Timer Current Count
WD_BUS:        .equ     0x0FC     # Bus Watchdog Timer
```

General Purpose Timer Registers

```
GPT0_COUNT:    .equ    0x100   # GPT 0 Timer Base Count
GPT0_CAPTURE:  .equ    0x104   # GPT 0 Capture Events
GPT1_COUNT:    .equ    0x108   # GPT 1 Timer base Count
GPT0_INT:      .equ    0x10C   # GPT 0 Timer Capture/Compare Interrupt Control
GPT0_ISTATUS:  .equ    0x110   # GPT 0 Timer Capture/Compare Interrupt Status
GPT1_CAPTURE:  .equ    0x114   # GPT 1 Capture Events
GPT1_INT:      .equ    0x118   # GPT 1 Interrupt Control
GPT1_ISTATUS:  .equ    0x11C   # GPT 1 Interrupt Status
GPT0_T0:       .equ    0x120   # GPT 0 Timer Capture Trigger 0
GPT0_T1:       .equ    0x124   # GPT 0 Timer Capture Trigger 1
GPT0_T2:       .equ    0x128   # GPT 0 Timer Capture Trigger 2
GPT0_T3:       .equ    0x12C   # GPT 0 Timer Capture Trigger 3
GPT1_T0:       .equ    0x130   # GPT 1 Timer Capture Trigger 0
GPT1_T1:       .equ    0x134   # GPT 1 Timer Capture Trigger 1
GPT1_T2:       .equ    0x138   # GPT 1 Timer Capture Trigger 2
GPT1_T3:       .equ    0x13C   # GPT 1 Timer Capture Trigger 3
GPT0_C0:       .equ    0x140   # GPT 0 Timer Compare 0
GPT0_C1:       .equ    0x144   # GPT 0 Timer Compare 1
GPT0_C2:       .equ    0x148   # GPT 0 Timer Compare 2
GPT0_C3:       .equ    0x14C   # GPT 0 Timer Compare 3
GPT1_C0:       .equ    0x150   # GPT 1 Timer Compare 0
GPT1_C1:       .equ    0x154   # GPT 1 Timer Compare 1
GPT1_C2:                .equ    0x158   # GPT 1 Timer Compare 2
GPT1_C2:       .equ    0x15C   # GPT 1 Timer Compare 3
GPT0_M0:       .equ    0x160   # GPT 0 Timer Compare Mask 0
GPT0_M1:       .equ    0x164   # GPT 0 Timer Compare Mask 1
GPT0_M2:       .equ    0x168   # GPT 0 Timer Compare Mask 2
GPT0_M3:       .equ    0x16C   # GPT 0 Timer Compare Mask 3
GPT1_M0:       .equ    0x170   # GPT 1 Timer Compare Mask 0
GPT1_M1:       .equ    0x174   # GPT 1 Timer Compare Mask 1
GPT1_M2:       .equ    0x178   # GPT 1 Timer Compare Mask 2
```

```
GPT1_M3:        .equ    0x17C   # GPT 1 Timer Compare Mask 3
```

Interrupt Controller Registers

```
INT_STATUS:     .equ    0x180   # Interrupt Status
INT_MSTATUS:    .equ    0x184   # Interrupt Masked status
INT_ENABLE:     .equ    0x188   # Interrupt Enable
INT_GENERATE:   .equ    0x18C   # Interrupt Generation Type
INT_POLARITY:   .equ    0x190   # Interrupt Polarity
INT_TRIGGER:    .equ    0x194   # Interrupt Trigger Type
INT_VBADDR:     .equ    0x198   # Interrupt Vector Base Address
INT_VINC:       .equ    0x19C   # Interrupt Vector Incement
INT_VECTOR:     .equ    0x1A0   # Interrupt Vector Address
INT_SOFTSET:    .equ    0x1A4   # Interrupt Software Set
INT_SOFTSRC:    .equ    0x1A8   # Interrupt Controller Software Source
#                       0x1AC   PowerPro Reserved
```

UART Registers

```
UART0_RX_TX:    .equ    0x1B0   # UART 0 Receive / Transmit Data
UART0_IER:      .equ    0x1B1   # UART 0 Interrupt Enable
UART0_ISTAT_FIFO: .equ 0x1B2    # UART 0 Interrupt Status / FIFO Control
UART0_LCR:      .equ    0x1B3   # UART 0 Line Control
UART0_MCR:      .equ    0x1B4   # UART 0 Modem Control
UART0_LSR:      .equ    0x1B5   # UART 0 Line Status
UART0_MSR:      .equ    0x1B6   # UART 0 Modem Status
UART0_SCR:      .equ    0x1B7   # UART 0 Scratchpad
#                       0x1B8   PowerPro Reserved
UART1_RX_TX:    .equ    0x1C0   # UART 1 Receive / Transmit Data
UART1_IER:      .equ    0x1C1   # UART 1 Interrupt Enable
UART1_ISTAT_FIFO: .equ 0x1C2    # UART 1 Interrupt Status / FIFO Control
UART1_LCR:      .equ    0x1C3   # UART 1 Line Control
UART1_MCR:      .equ    0x1C4   # UART 1 Modem Control
UART1_LSR:      .equ    0x1C5   # UART 1 Line Status
UART1_MSR:      .equ    0x1C6   # UART 1 Modem Status
```

```
 UART1_SCR:     .equu    0x1C7   # UART 1 Scratchpad
 #                       0x1C8   PowerPro Reserved


 #                       0x1CC - 0x1DC PowerPro reserved
```

General Purpose I/O Registers

```
GPIO_A:         .equ    0x1E0   # GPIO Enable, Mask, Direction, Data[0:7]
 GPIO_B:        .equ    0x1E4   # GPIO Enable, Mask, Direction, Data[8:15]
 GPIO_C:        .equ    0x1E8   # GPIO Enable, Mask, Direction, Data[16:23]
 GPIO_D:        .equ    0x1EC   # GPIO Enable, Mask, Direction, Data[24:31]
 GPIO_E:        .equ    0x1F0   # GPIO Enable, Mask, Direction, Data[32:39]
 GPIO_F:        .equ    0x1F4   # GPIO Enable, Mask, Direction, Data[40:47]
 GPIO_G:        .equ    0x1F8   # GPIO Enable, Mask, Direction, Data[48:49]
 #                      0x1FC     PowerPro Reserved
```

HOW TO REACH US:

USA/EUROPE/LOCATIONS NOT LISTED:

Motorola Literature Distribution
P.O. Box 5405, Denver, Colorado 80217
1-303-675-2140
(800) 441-2447

JAPAN:

Motorola Japan Ltd.
SPS, Technical Information Center
3-20-1, Minami-Azabu Minato-ku
Tokyo 106-8573 Japan
81-3-3440-3569

ASIA/PACIFIC:

Motorola Semiconductors H.K. Ltd.
Silicon Harbour Centre, 2 Dai King Street
Tai Po Industrial Estate, Tai Po, N.T., Hong Kong
852-26668334

TECHNICAL INFORMATION CENTER:

(800) 521-6274

HOME PAGE:

www.motorola.com/semiconductors

MOTOROLA

AN2347/D