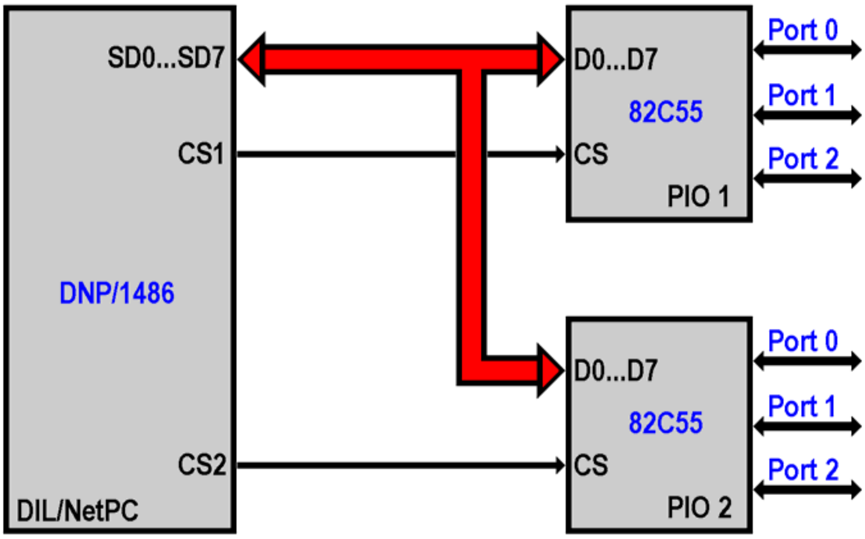
How to implement additional 48-Bit parallel I/O Lines

- **1. Step**: Connect two 82C55 PIOs to the (A)DNP/1486 data bus. Use CS1 and CS2 for generating the chip selects. Each 82C55 offers three 8-bit ports (Port 0, Port 1 and Port 3).



- **2. Step**: Use the following C source code for Linux to test your PIOs. This sample programs CS1 for a chip select range 0x200 to 0x207 and CS2 for 0x280 to 0x28f.

```
// Test for two 82C55 PIOs at 0x200 and 0x280
// Written by KDW (kdw@ist1.de) - 22.05.2002

// Includes

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <asm/io.h>

// Defines

#define CSCIR   0x22                      // SC410 CSC Index Register
#define CSCDR   0x23                      // SC410 CSC Data Register

////////////////////////////////////////////////////////////////////////
// SC410 Low Level Function: windex -- Write Byte to CSC Registerspace

void windex (unsigned char index, unsigned char data)
{
   outb (index, CSCIR);
   outb (data, CSCDR);
}

////////////////////////////////////////////////////////////////////////
// SC410 Low Level Function: rindex -- Read Byte from CSC Registerspace

unsigned char rindex (unsigned char index)
```

```
{
   outb (index, CSCIR);
   return (inb (CSCDR));
}

////////////////////////////////////////////////////////////////////////
// main -- The one and only main function ...

int main (int argc, char *argv[])
{
   int i;

   // We need I/O access to CSCIR, CSCDR, 0x200-0x203, 0x280-0x283...

   ioperm (CSCIR, 2, 1);                // CSCIR, CSCDR
   ioperm (0x200, 4, 1);                // 0x200-0x203
   ioperm (0x280, 4, 1);                // 0x280-0x283

   // ************** SETUP FOR SC410 CHIP UNIT **************
   // Set (A)DNP/1486 CS1 for I/O address space 0x200 - 0x207
   // Set (A)DNP/1486 CS2 for I/O address space 0x280 - 0x28f
   // =======================================================

   windex (0xa6, rindex (0xa6) | 0x03);           // Step 1
   windex (0xa0, rindex (0xa0) | 0x05);           // Step 2
   windex (0x3b, rindex (0x3b)| 0x03);            // Step 3
   windex (0xe5, (rindex (0xe5) & 0xfe) | 0x01);  // Step 4
   windex (0xb4, 0x00);                           // CSA Step 5.1
   windex (0xb5, 0x22);                           // CSA Step 5.2
   windex (0xb6, 0x80);                           // CSB Step 5.1
   windex (0xb7, 0x02);                           // CSB Step 5.2
   windex (0xb8, (rindex (0xb8) & 0x88) | 0x33);  // Step 6
   windex (0xb2, 0x10);                           // Step 7
   windex (0xa6, rindex (0xa6) & 0xfc);           // Step 8

   // Set all 82C55 ports for output...

   outb (0x80, 0x203);           // PIO1: Port0=Port1=Port2=Output
   outb (0x80, 0x283);           // PIO2: Port0=Port1=Port2=Output

   // Run counter for all ports...

   for (;;) {

      // Write 8-bit binary counter value to PIO 1, Port 0...

      printf ("\n");
      for (i= 0; (i < 256); i++) {

         outb (i & 0xff, 0x200);
         printf ("\r  PIO 1, Port 0: Current Counter Value= %3d", i);
         fflush (stdout);
         usleep (100000);
      }
```

```
// Write 8-bit binary counter value to PIO 1, Port 1...

printf ("\n");
for (i= 0; (i < 256); i++) {

   outb (i & 0xff, 0x201);
   printf ("\r  PIO 1, Port 1: Current Counter Value= %3d", i);
   fflush (stdout);
   usleep (100000);
}

// Write 8-bit binary counter value to PIO 1, Port 2...

printf ("\n");
for (i= 0; (i < 256); i++) {

   outb (i & 0xff, 0x202);
   printf ("\r  PIO 1, Port 2: Current Counter Value= %3d", i);
   fflush (stdout);
   usleep (100000);
}
outb (0x00, 0x200);
outb (0x00, 0x201);
outb (0x00, 0x202);

// Write 8-bit binary counter value to PIO 2, Port 0...

printf ("\n");
for (i= 0; (i < 256); i++) {

   outb (i & 0xff, 0x280);
   printf ("\r  PIO 2, Port 0: Current Counter Value= %3d", i);
   fflush (stdout);
   usleep (100000);
}

// Write 8-bit binary counter value to PIO 2, Port 1...

printf ("\n");
for (i= 0; (i < 256); i++) {

   outb (i & 0xff, 0x281);
   printf ("\r  PIO 2, Port 1: Current Counter Value= %3d", i);
   fflush (stdout);
   usleep (100000);
}

// Write 8-bit binary counter value to PIO 2, Port 2...

printf ("\n");
for (i= 0; (i < 256); i++) {

   outb (i & 0xff, 0x282);
   printf ("\r  PIO 2, Port 2: Current Counter Value= %3d", i);
   fflush (stdout);
```

```
         usleep (100000);
      }
      outb (0x00, 0x280);
      outb (0x00, 0x281);
      outb (0x00, 0x282);
   }

   // Exit...

   ioperm (CSCIR, 2, 0);
   ioperm (0x200, 4, 0);
   ioperm (0x280, 4, 0);
   return (EXIT_SUCCESS);
}
```