How to use the PIO port C of the DNP/5280

With the help of the DNP/5280 Hardware Access Driver (SSVHWA) you can access direct the MCF5280 special function registers with a user space program.

- **1. Step**: Use the following C source code as sample. This code implements a simple end-less loop with PIO access to port C. Within this loop the programs reads current state of the PIO port C of the DNP/5280 PIO.

```
// PIO port C Demo for DIL/NetPC DNP/5280-3V
// Author  : mha@st1.de
// Date    : 26.08.2003
// Version : 1.00
// License : GPL

#include <unistd.h>
#include <stdio.h>

#include "ssvhwa.h"

#define MCFBAR   0x40000000

#define PORTAS    (MCFBAR + 0x0010000C)      //  8 bit
#define DDRAS     (MCFBAR + 0x00100020)      //  8 bit
#define PORTASP   (MCFBAR + 0x00100034)      //  8 bit
#define SETAS     (MCFBAR + 0x00100034)      //  8 bit
#define CLRAS     (MCFBAR + 0x00100048)      //  8 bit
#define PASPAR    (MCFBAR + 0x00100056)      // 16 bit

#define GPTAPORT  (MCFBAR + 0x001A001D)      // 8 bit
#define GPTADDR   (MCFBAR + 0x001A001E)      // 8 bit

#define PORTQS    (MCFBAR + 0x0010000D)      // 8 bit
#define DDRQS     (MCFBAR + 0x00100021)      // 8 bit
#define PORTQSP   (MCFBAR + 0x00100035)      // 8 bit
#define SETQS     (MCFBAR + 0x00100035)      // 8 bit
#define CLRQS     (MCFBAR + 0x00100049)      // 8 bit
#define PQSPAR    (MCFBAR + 0x00100059)      // 8 bit

void write_portC (unsigned char data)
{
    // write data Port C

    ssvhwa_write8 (PORTQS, data);
}

unsigned char read_portB (void)
{
    unsigned char data_bl, data_bh;

    // read data Port B[0-3]

    data_bl = ssvhwa_read8 (GPTAPORT) & 0xf;
```

```
    // read data Port B[4-7]

    data_bh = (ssvhwa_read8 (PORTASP) & 0xf) << 4;
    return (data_bh | data_bl);
}

int main (void)
{
    unsigned char iDIPsw, iDIPswMirror;

    // check user identity

    if (geteuid () != 0) {
       printf ("No root access rights !\n");
       exit (1);
    }

    if (ssvhwa_open() < 0) {           // open ssvhwa driver
       perror ("ssvhwa open");
       exit (-1);
    }

    // set DIL/NetPC PIO Port C = output

    ssvhwa_write8 (PQSPAR, 0x00);      // Port C to GPIO
    ssvhwa_write8 (DDRQS, 0x0f);       // Port C to output

    // set DIL/NetPC PIO Port B = input

    ssvhwa_write8 (GPTADDR, 0x00);     // Port B[0-3] input
    ssvhwa_write8 (DDRAS, 0x00);       // Port B[7-4] input

    // build DIP-Switch status mirror (read Port B - only
    // low nibble)

    iDIPswMirror = read_portB () & 0xf;

    // show current DIP-Switch status (message and LEDs)...

    printf ("\n  DIP-Switch (low nibble)= 0x%02x", iDIPswMirror);
    fflush (stdout);

    // update LEDs

    write_portC (iDIPswMirror);        // write Port C

    // read DIP-Switch and show status until user break...

    while (1) {

       // Read current DIP-Switch Status (read Port B - only
       // low nibble).

       iDIPsw = read_portB () & 0xf;
```

```
            // compare current status with last status

            if (iDIPsw != iDIPswMirror) {

               // DIP-Switch status change...

               iDIPswMirror= iDIPsw;
               printf ("\b\b%02x", iDIPsw); // update message
               fflush (stdout);

               // update LEDs

               write_portC (iDIPsw);         // write Port C
            }
            usleep (1000);
         }
         ssvhwa_close ();
         return (0);
      }
```

- **2. Step**: Translate the C source code with the GNU C cross compiler to a executable for the DNP/5280.

- **3. Step**: Transfer the executable within a TFTP session from your development PC to the DNP/5280.

- **4. Step**: Make sure that the executable owns executable rights (`chmod +x prg-name`).

- **5. Step**: Run the executable on the DNP/5280. Use a Telnet session for user I/O.