

Applied Data Sciences / PCDR11® Software Specifications

2.1 Subroutines

There are six (6) C compatible subroutines to provide device level software support for the PCDR11 interface card. These subroutines should be used in conjunction with the Include File PCDR11.H which provide definitions for C programs. These subroutines are compatible and have been tested with applications developed under Microsoft C (c) Version 5.0. Additional information is presented in the following sections.

2.2 PCDR11.H File - Device Control Block

The Device Control Block hereafter known as the DCB is a "C" data structure which provides all parameters and temporary storage necessary for any of the Device Driver Subroutines. The source for this structure is provided in the include file PCDR11.H on the distribution disk (File Part Number 0950019) and is also presented in more detail in APPENDIX A of this manual. Each element is used as follows:

```
struct dcb_struct
{
union
{
unsigned far * pnt; /* Address of data buffer. */
unsigned long bin;
} buf_adr;
unsigned int xfr_cnt; /* Buffer length in 16 bit words. */

/* Device Control/Status Flags */

union {
struct
{
unsigned : 7; /* Reserve */
unsigned nowait : 1; /* Nowait for transfer complete = 1. */
unsigned dev_err : 1; /* Device opened */
unsigned dev_att : 1; /* Attention Interrupt Flag */
```

```

unsigned : 4; /* Reserve */
unsigned tim_err : 1; /* Time-out error status = 1. */
unsigned xfr_fin : 1; /* Device Busy Flag = 1. */
} bit;
unsigned int bin;
} sts_flg;

/* DR11 Control Word */

union {
struct dr_cntl_struct
{
unsigned go_puls : 1; /* Go Pulse Enable Bit */
unsigned funcbit : 3; /* DRV11 Function Bits */
unsigned byt_swp : 1; /* Byte swap select 0 = No, 1 = Yes. */
unsigned wrd_swp : 1; /* Word swap select 0 = No, 1 = Yes. */
unsigned int_enb : 1; /* Interrupt Enable */
unsigned : 1; /* Reserve */
unsigned cyc_rqs : 1; /* Cycle Request Prime */
unsigned pc_clr : 1; /* Reserve */
unsigned xfr_dir : 1; /* Transfer direction 0=To PC, 1=To Device */
unsigned : 1; /* Reserve */
unsigned mnt_sel : 1; /* Maintenance select for diagnostics. */
unsigned : 1; /* Reserve */
unsigned dma_enb : 1; /* DMA Enable */
unsigned hld_rfr : 1; /* Hold off refresh. */
} bit;
unsigned int bin;
} dr11ctl;

/* PC Control/Status Word */

union {
struct pc_cntl_struct
{
unsigned : 1; /* Reserve */
unsigned bsy_pol : 1; /* busy output polarity 0 = Low true */
/* else 1 = High true */
unsigned f2l_sel : 1; /* function 2 level/pulse select 1 = pulse */
unsigned iv2_pls : 1; /* InitV2 pulses with Init if = 1 */

```

```

/* else does not track init pulse */
unsigned eoc_str : 1; /* EOC start on end of busy */
/* else EOC ends on end of busy */
unsigned eoc_rst : 1; /* EOC resets ready */
/* else ready is reset at end of busy */
unsigned ost_tim : 2; /* Output set-up Time select */
unsigned mon_sel : 1; /* Monitor Mode operation = 1 */
unsigned tst_sel : 1; /* Test mode select = 1 */
unsigned skw_tim : 2; /* Input data skew */
unsigned cyc_pol : 1; /* Cycle request polarity 0=Low 1=High */
unsigned cyc_dis : 1; /* Disable cycle request B input */
unsigned dma_stp : 1; /* Stop DMA transfers if interrupt. */
unsigned pls_lng : 1; /* Pulse functions length 0=300 1=500 */
} bit;
unsigned int bin;
} ctl_reg;

/* DR11 Status Word */

union {
struct dr_stat_struct
{
unsigned rdy_int : 1; /* Ready Interrupt Status */
unsigned funcbit : 3; /* DRV11 Function Bits */
unsigned out_fif_emp : 1; /* Output Fifo Empty Status */
unsigned in_fif_emp : 1; /* Input Fifo Empty Status */
unsigned ien_sta : 1; /* Interrupt Enable */
unsigned dr11rdy : 1; /* DR11 Ready Signal State */
unsigned cyc_rqs : 1; /* Cycle Request Prime */
unsigned dev_sts : 3; /* DRV11 Device Status */
unsigned man_sta : 1; /* Maintenance select for diagnostics. */
unsigned att_sta : 1; /* Attention signal state from device */
unsigned dma_act : 1; /* PCDR11 DMA active */
unsigned att_int : 1; /* Attention Interrupt bit */
} bit;
unsigned int bin;
} sts_xfr;
unsigned int sts_cnt; /* End of Xfer Count */
unsigned int io_adr; /* Base address of PCDR11 card */
unsigned char irq_lev; /* Hardware Interrupt level for PCDR11 */

```

```

unsigned char dma_chn; /* Hardware DMA level for PCDR11 */
unsigned char dma_mod; /* DMA Mode of operation */
unsigned char tim_out; /* Time Out max time in seconds. */
unsigned int spr_cnt; /* Spurious Interrupt Count */
unsigned int reserve[9]; /* Reserve workong storage space. */
};

```

2.3 Device Subroutines

The six (6) device callable subroutines are presented in detail in the following sections. The object code for these subroutines are provided in the file DR11DEV.OBJ on the distribution disk (File Part Number 0950018). A tested example of the use of these subroutines is presented in APPENDIX B of this manual.

2.3.1 DRVOPEN - Device Open

This subroutine must be called prior to calling any other interface subroutine. It resets the PCDR11, links the interrupt service routine into the interrupt structure and initializes any data in the Device Control Block. The Device Control Block is passed to this subroutine via a pointer in the parameter list and the subroutine returns an "open" status. A return status of anything other zero indicates an error which should be corrected before any further subroutine calls.

NOTE: The Device Control Block must have valid set-up information prior to calling DRVOPEN, except for the transfer address and count. These are not used at this time and need not be correct.

int DRVOPEN(dcb) Open device channel

```

struct dcb_struct *dcb; /* Pointer to device control block for all transfers. */

```

Return status codes are:

0 = successful completion

2 = invalid interrupt level

3 = invalid DMA level

4 = invalid DMA mode

5 = invalid I/O port address

6 = device not present or malfunctioning

2.3.2 DRV_CLOS - Device Close

This subroutine must be called before exiting the application. It restores the interrupt vectors to the state they were in prior to calling DRVOPEN. The subroutine will not close the device if a transfer is currently executing, but will return with an error code.

NOTE: Call DRVTERM to stop unwanted transfers prior to calling DRVCLOS.

int DRVCLOS(dcb) Close device channel

struct dcb_struct *dcb; /* Pointer to device control block for all transfers. */

Returns status codes are:

0 = successful completion

1 = transfer not complete

2.3.3 DRVTERM - Terminate Transfer

A call to this subroutine will cause any existing DMA transfer to be ceased.

void DRVTERM(&dcb, reset) Terminate device channel

struct dcb_struct *dcb; /* Pointer to device control block for all transfers. */

int reset Additional bits to set in PCDR11 control/status register. Reset (bit 9) is set unconditionally.

2.3.4 DRVXFER - Start DMA Transfer

This subroutine is used to start a DMA transfer on the PCDR11 interface. All control parameters are set-up in the DCB. The subroutine can be called in either wait or nowait mode. In wait mode the subroutine waits until the transfer is complete or a time-out occurs.

NOTE: A time-out value of zero causes an infinite time-out length.

int DRVXFER(dcb) Transfer data block

struct dcb_struct *dcb /* Pointer to device control block for all transfers. */

Returns status codes are:

0 = successful completion

1 = device not open error

2 = device busy error

5 = time-out error

2.3.5 DRVWREG - Write PCDR11 Registers

The user would call this subroutine to directly write to the PCDR11 control registers. The DCB address, register and 16 bit value are passed to the subroutine. The register parameter is a name defined in the PCDR11.H include file.

The valid names are:

PC_CNTL = PC Control register

XFR_CNT = Transfer Count

DR_CNTL = DRV11 Control Register

DAT_REG = Data I/O Register

void DRVWREG(dcb,register,value) Set PCDR11 Register

struct dcb_struct *dcb; /* Pointer to device control block for all transfers. */

int register; /* register name

PC_STAT = PC Status register

XFR_CNT = Transfer Count

DR_STAT = DRV11 Status Register

DAT_REG = Data I/O Register */

int value; /* 16 bit write value. */

NOTE: If the register is PC_CNTL, the contents of the ctl_reg element of the DCB is or'd with value. If the register is DR_CNTL, the contents of the drlctl element of the DCB, except the dma_enb, pc_clr, cyc_rqs, and reserved bits, is or'd with value.

2.3.6 DRVRREG - Read PCDR11 Register

This subroutine is called to read the contents of the PCDR11 status registers. Like the register write routine it takes as parameters the DCB address and register name. It returns a 16 bit value as the contents of the corresponding status register.

void DRVRREG(dcb,register,value) Read PCDRV11 register

struct dcb_struct *dcb; /* Pointer to device control block for all transfers. */

```
int register; /* register name
PC_STAT = PC Status register
XFR_CNT = Transfer Count
DR_STAT = DRV11 Status Register
DAT_REG = Data I/O Register */
int *value; /* address of 16 bit return value. */
```