C Universal library

C.1 Introduction

The universal library provides functionality to program the digital board from your Windows/C++ project. This appendix gives an overview of the most important function calls and contains a small sample program to help you get started. A more detailed help file should be somewhere in D: Software.
\

C.2 Adding the library to your C/C++ projects

To use the universal library in your software project, you need to include the header file "cbw.h" and the file "cbw32bc.lib" to your project. Your C program should have #include "cbw.h" at the top. Also make sure "cbw32.dll" is located in the same folder as all your project files, though it does not need to be added to your project.

All files should be available on the course website

C.3 Board and ports numbers

The universal library can manage several digital boards simultaneously, therefore you always need to specify a board number for your I/O. The boards in the Digilab have board number 0.

The digital board emulates two Intel 8255 chips; each chip provides four ports: port A and port B are 8-bits wide, port C is split into two 4-bit ports (lower and upper nibble). When referring to these ports, you can use the following constants:
FIRSTPORTA, FIRSTPORTB, FIRSTPORTCL/FIRSTPORTCH for the four ports on the first chip and SECONDPORTA, SECONDPORTB, SECONDPORTCL/SECONDPORTCH for the second chip.

C.4 Functions

This sections describes the most important functions in the universal library and their parameters. For shortness of notation, let *USHORT* be an unsigned 16-bit integer, i.e. *typedef unsigned short USHORT*.

int cbDeclareRevision(float* RevNum)

Must be the first universal library function to be called by your program. With this function you declare the version of the library that you used when you wrote your program (parameter RevNum). It checks if the currently installed library is compatible with the version that you used. The return value indicates whether you can proceed (0) or if an error occurred (non-zero value).
int cbErrHandling(int ErrReporting, int ErrHandling)

Specifies the error handling for all subsequent library function calls. Most functions return error codes, but you can other features like how errors are reported and treated.

- *ErrReporting*: controls when the library prints on the screen:
  - DONTPRINT (default): errors will not generate a message on the screen. Your program must do all error checking and handling.
  - PRINTWARNINGS: only warning errors generate a message on the screen. Your program will have to check for fatal errors.
  - PRINTFATAL: only fatal errors will generate a screen message. Your program must check for warning errors.
  - PRINTALL: all errors will generate a message to the screen.
- *ErrHandling*: specifies what class of error will cause the program to halt:
  - DONTSTOP (default): the program will continue executing when an error occurs.
  - STOPFATAL: halt on fatal errors.
  - STOPALL: halt on any error.
- *Return value*: always returns 0.

int cbDConfigPort(int BoardNum, int PortNum, int Direction)

Configures a digital port for input or output. Using this function will reset all ports on a chip configured for output to a zero state. For this reason, this function is usually called at the beginning of the program for each port to be configured.

- *BoardNum*: number of the installed board. Boards in the digilab computers should have number 0.
- *PortNum*: the port you want to configure.
- *Direction*: DIGITALIN for input and DIGITALOUT for output.

Return value: error code or 0 if no error occurred.

int cbDBitIn(int BoardNum, int PortType, int BitNum, USHORT *BitValue)

Reads the state of a single digital input bit. This function treats all of the DIO ports on a board as a single port and lets your read any individual bit on ports configured for input.

BoardNum: board number

PortType: set to FIRSTPORTA even if you access another port (see below).

BitNum: specifies the bit number within the single large port. The specified bit should be in a port that is currently configured for input.

port	bit number
FIRSTPORTA	0–7
FIRSTPORTB	8–15
FIRSTPORTCL	16–19
FIRSTPORTCH	20–23
SECONDPORTA	24–31
SECONDPORTB	32–39
SECONDPORTCL	40–43
SECONDPORTCH	44–47

Table C.1: Port bit numbering of the unified port.

BitValue: address of the variable to store the bit value. Note that a logic high reading (1) does not necessarily mean +5V.

Return value: error code or 0 if no error occurred.

int cbDBitOut (int BoardNum, int PortType, int BitNum, USHORT BitValue)

Sets the state of a single digital output bit. Note that that port must be configured for output. The parameters and the return value are the same as for *cbdBitIn*, only that *BitValue* is a value parameter and not a reference.

int cbDIn (int BoardNum, int PortNum, USHORT *DataValue)

Read a digital input port. If the port is configured for output, this function will provide readback of the last output value. Note that the size of the ports vary — 4-bit ports use the 4 LSBs and 8-bit ports use the 8 LSBs.

- *BoardNum, PortNum*: board and port number
- *DataValue*: reference to the variable that stores the digital input value.

Return value: error code or 0 if no error occurred.

int cbDOut (int BoardNum, int PortNum, USHORT DataValue)

Write a byte to a digital output port.

BoardNum, PortNum: board and port number

DataValue: value to be written — for 4-bit ports the 4 LSBs and for 8-bit ports the 8 LSBs.

Return value: error code or 0 if no error occurred.

C.5 Example program

```
/* Include files */
#include <stdio.h> #include <conio.h> #include
"cbw.h"

/* Prototypes */
void ClearScreen (void); void
GetTextCursor (int *x, int *y); void
MoveCursor (int x, int y);

void main ()
    /* Variable Declarations */
{
    int Row, Col, I; int BoardNum = 0; int ULStat, PortNum, Direction, BitValue; USHORT DataValue,
    PowerVal; float RevLevel =(float) CURRENTREVNUM;

    /* Declare UL Revision Level */
    ULStat = cbDeclareRevision(&RevLevel);

    /* Initiate error handling */
    cbErrHandling(PRINTALL, STOPALL);

    /* Set up the display screen */
    ClearScreen();
    printf("Demonstration of cbDIn()\n\n");
    printf("Press any key to quit.\n\n");
    printf ("The bits on FIRSTPORTA are: ");
    printf ("76 5 4 3 2 1 0 \n");
    GetTextCursor (&Col, &Row);


    /* Configure FIRSTPORTA for digital input */
    PortNum = FIRSTPORTA;
    Direction = DIGITALIN;
    ULStat = cbDConfigPort (BoardNum, PortNum, Direction);


    while (!kbhit())
        /* Read the 7 bits digital input and display */
    {
        ULStat = cbDIn(BoardNum, PortNum, &DataValue);

        /* Display the value collected from the port */
        MoveCursor (Col, Row);
        printf ("Port Value: %u ", DataValue);


        /* Parse DataValue into bit values to indicate on/off status */
        MoveCursor (Col + 21, Row);
        for (I = 7; I >= 0; I    )
                               --
```

```
            PowerVal =1 << I;
      { if (DataValue & PowerVal)
      BitValue = 1;

                  else
                        BitValue = 0;

                  printf (" %u ", BitValue);

            }
            MoveCursor (1,20);
            printf ("\n");


      }
}
```