University of the West of England
BRISTOL

Coursework Submission Coversheet

(individual coursework only)

Student Number **1100 4840**

11004840 2230110 27

BRISTOL INSTITUTE OF TECHNOLOGY
COMPUTER SCIENCE AND CREATIVE TECHNOLOGIES

For checking by the student:

Please ensure all information is complete and correct and attach this form securely to the front of your work before handing it in to the Help Desk reception **located in the Project Room (2Q30)**.

Your Award name:	**BSC (HONS) COMPUTER SYSTEMS INTEGRATION**
Module code:	**UFCETS-20-1**
Module name:	**PROGRAMMING IN C**
Module run:	**11SEP/1 AY**
Coursework code:	**CW1**
Coursework title:	**PRACTICAL COURSEWORK 1**

	Date:	Time:
Submission deadline:	**08 December 2011**	**14:00**

In submitting this form with your assignment you make the following declaration:

I declare that the coursework submitted is my own work and has not (either in whole or part) been submitted towards the award of any other qualification either at UWE or elsewhere. I have fully attributed/referenced all sources of information used during the completion of my assignment, and I am aware that failure to do so constitutes an assessment offence.

You are strongly advised to retain a second copy of your work in case of any query about the assignment

For completion by the student:

Total number of pages including this front sheet.

For completion by the school:

Mark

This mark is provisional and subject to moderation and approval by the relevant examining board.

Duckshoot

Programming in C

By James Johns • ID: 11004840 • University of the West of England • 4 December 2011

Contents

University of the West of England
BRISTOL

MODULAR PROGRAMME

ASSESSMENT SPECIFICATION

Module Details

Module Code UFCETS-20-1	**Run** 11SEP/1 AY	**Module Title** Programming in C
Module Leader Ian Johnson	**Module Tutors** Ian Johnson	
Component and Element Number B1		**Weighting: (% of the Module's assessment)** 13%
Element Description Practical Coursework 1		**Total Assignment time** 5 hours + Lab Time

Dates

Date Issued to Students w/b 24/10/2011	**Date to be Returned to Students** 20th January 2012
Submission Place **PROJECT ROOM - 2Q30** **(Help Desk open 9.00 - 6.00pm)**	**Submission Date** 8th December 2011
	Submission Time **2.00 pm**

Deliverables

As per the attached specification.

Module Leader Signature

Ian Johnson

UFCETS-20-1 – C Programming
First Coursework - October 2011

Duckshoot

Your assignment, due at 2pm on Thursday 8th December is to implement the duckshoot program as specified in the Introduction to Digital IO worksheet available on the module web page (http://www.cems.uwe.ac.uk/~irjohnso/coursenotes/ufs001/ufs001c1-w06-NEW.pdf).
As per the description on the worksheet, you should start the program with alternate LEDs lit (10101010) on the MARCO RACK, and rotate this bit pattern, inverting LEDs with a trigger up, then down, then up sequence on the bottom switch.
Not "machine gunning" implies that starting with the switch up, down to fire, return to up again sequence has to be repeated in full to fire again.
Penalty for leaving the switch down means that unless the switch is returned to up in a given time, the duck reappears (or fails to disappear) to penalize the player.
In all circumstances, a single action by the user (e.g. leaving a switch up) will only cause a single change (e.g. 1 duck reappears)

The marks shown are the maximum available in each category.

Deliverables:

A diagrammatic design of your program using any method with which you are familiar	(10)

Printed source code:

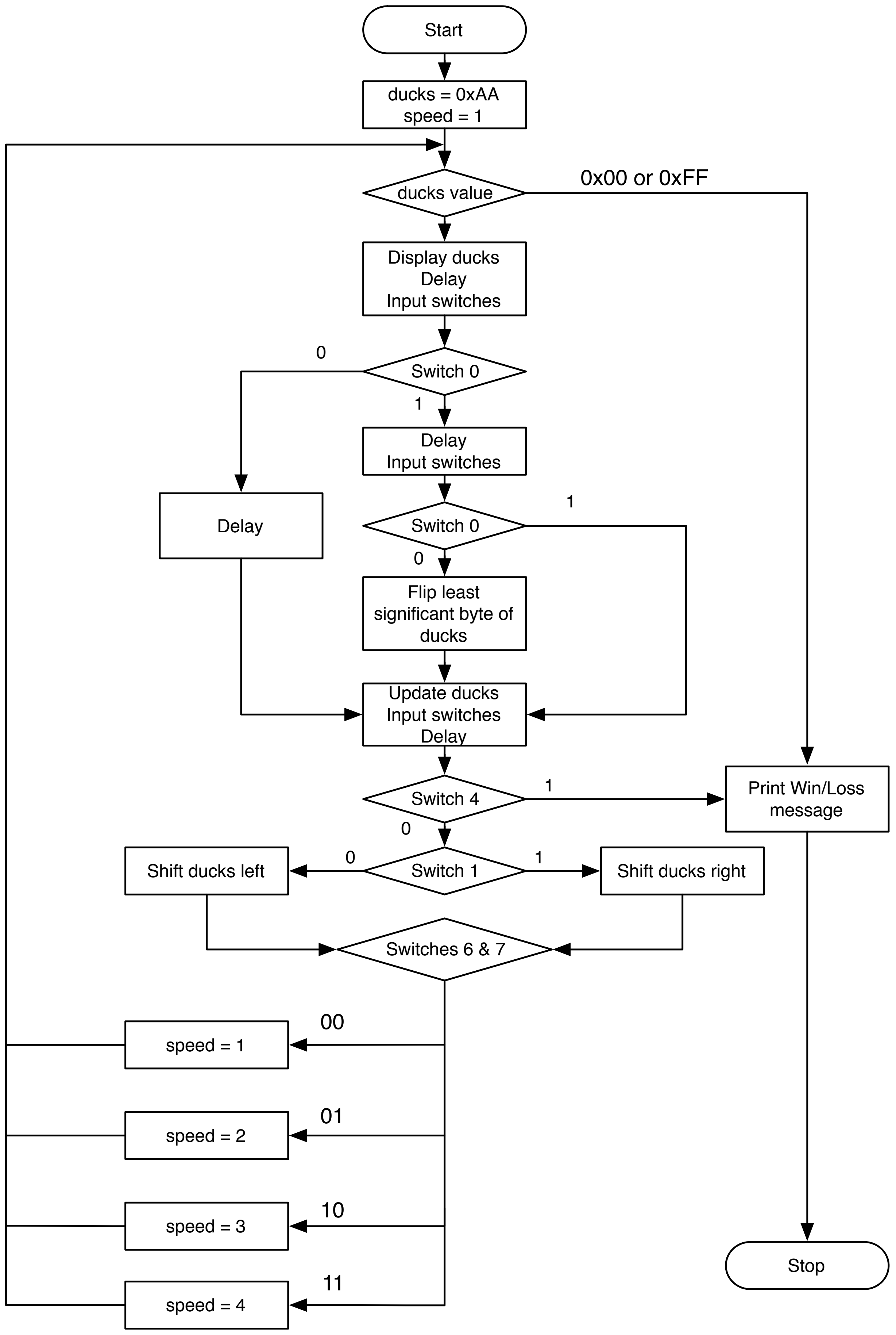
Well commented (-5% for C++ comments)	(10)
Program Banner	(5)
Program uses functions	(5)
Function banners	(5)
Functions use arguments	(15)
No Global variables	(10)
Program is laid out in either classic C or Pascal style	(10)

Together with this page, signed off by your lab tutor, to indicate you have demonstrated your work:

Test	Marks	Signature & Date
Continuously rotating bit pattern on a suitable timebase, **not using a software delay loop**	(5)	
Player penalty for leaving switch down (duck reappears)	(5)	
Not machine-gunning	(10)	
Direction variable by switch within game	(5)	
Using 2 switches to implement 4 speeds or levels of difficulty within game	(5)	

All the above should be submitted together with a cover sheet to the Project Room.

Design - Algorithmic State Machine



Source code

Makefile

```
all: main.c
	cc -std=c99 -Wall -lc -lcomedi -lm -o duckshoot main.c

clean:
	rm duckshoot
```

main.c

```c
/*********************************************************************************
 * Filename          : main.c                                                   *
 * Author            : James Johns                                              *
 *                                                                              *
 * Copyright © 2011, James Johns. All rights reserved.                          *
 *                                                                              *
 * First written on 31/10/2011 by James Johns.                                  *
 *                                                                              *
 * Description:                                                                 *
 *  Program to play a game of duckshoot on a comedilib compatible device.       *
 * Compile with:                                                                *
 *  gcc -std=c99 -Wall -lc -lcomedi -lm -o duckshoot main.c                     *
 *                                                                              *
 ********************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>
#include <getopt.h>
#include <ctype.h>
#include <comedi.h>
#include <comedilib.h>

#define DIRECTION_UP 0
#define DIRECTION_DOWN 1

/* usleep was not defined in <unistd.h> on some machines, so manually define it ourselves */
int usleep(unsigned int usec);

int checkGun(unsigned int numval);
void showWinMessage(unsigned char ducks);
void shiftDucks(unsigned char *ducks, unsigned int shouldFlip, unsigned int direction);


/*********************************************************************************
 * Function name   : int main(int argc, const char *argv[])                     *
 *     returns     : 0 on success, otherwise error code.                        *
 *          argc       : Number of elements in argv                             *
 *          argv       : Array of command line arguments                        *
 *                                                                              *
 * Created by      : James Johns                                                *
 * Date created    : 15/11/2011                                                 *
 * Description     : Main entry point. Opens and configures Comedi device,      *
 *                   initialises game variables and begins main execution loop. *
 * NOTES           : BUG - can flip trigger switch quickly without any consequence.  *
 ********************************************************************************/
int main(int argc, const char *argv[])
{
    comedi_t *device;
    unsigned int subdevice = 2;
    char *filename = "/dev/comedi0";
    unsigned char ducks = 0xAA;     /* Value to hold Duck layout. Initially 0xAA for alternating ducks */
    unsigned int outputMask = 0xFF; /* Bits 0-7 are our output mask for LEDs */
    unsigned char switches = 0x00;
    unsigned int speedMul = 0x01;   /* Speed multiplier, used to slow down/speed up the main
                                     * loop by changing the length of usleep */

        /* Open a device to the Control Rack */
    device = comedi_open(filename);
    if (!device) {
        comedi_perror(filename);
        exit(1);
    }

        /* Make sure we have input and output available */
    int stype = comedi_get_subdevice_type(device, subdevice);
    if (stype != COMEDI_SUBD_DIO) {
        printf("%d is not a digital I/O subdevice", subdevice);
        comedi_close(device);
        exit(2);
    }

        /* Configure ports; 0-7 outputs, 8-15 inputs */
    for (int i = 0; i < 8; i++)
        comedi_dio_config(device, subdevice, i, COMEDI_OUTPUT);
    for (int i = 8; i < 16; i++)
        comedi_dio_config(device, subdevice, i, COMEDI_INPUT);

        /* Initialise the default direction and LEDs as
         * alternate on and off before starting the game */
    unsigned int direction = DIRECTION_DOWN;
    unsigned int numval = ducks;
    comedi_dio_bitfield(device, subdevice, outputMask, &numval);

        /* win/loss conditions are all ducks on/off,
         * so we loop until one of these conditions are met */
    while ((ducks != 0x00) && (ducks != 0xFF)) {

            /* temporary variables required for loop algorithm. */
        char fire = 0;
        char gun = 0;
        char lastGun = 1;

        numval = ducks;
        comedi_dio_bitfield(device, subdevice, outputMask, &numval);

            /* wait a while to let user react to shifted pattern of LEDs */
        usleep(200000);                 /* wait long enough for user to react to shift */
        comedi_dio_bitfield(device, subdevice, outputMask, &numval);
        lastGun = checkGun(numval);     /* preliminary check. must return 1 to enable firing */
        usleep(100000 * speedMul);      /* wait for a short while to allow for user
                                         * to possibly trigger switch back up again. increase
                                         * with speed multiplier */

        comedi_dio_bitfield(device, subdevice, outputMask, &numval);
        gun = checkGun(numval);         /* final checking. must return 0 to fire */

        if ((lastGun == 1) && (gun == 0)){
            fire = 1;
        }


        shiftDucks(&ducks, fire, direction);

        numval = ducks;
        comedi_dio_bitfield(device, subdevice, outputMask, &numval);
        usleep(100000);


        switches = ~(numval >> 8) & 0xFF;
        if (switches & 0x02)            /* switch 1 is our direction switch */
            direction = DIRECTION_UP;
        else
            direction = DIRECTION_DOWN;

        if (switches & 0x10)            /* switch 4 is our exit switch */
            break;

            /* switches 6 and 7 are our speed selection */
        speedMul = ((switches >> 6) & 0x03) + 1;
    }

        /* make sure LEDs are up to date with ducks value */
    numval = ducks;
    comedi_dio_bitfield(device, subdevice, outputMask, &numval);

        /* print win/loss message then exit. */
    showWinMessage(ducks);
    comedi_close(device);
    return 0;
}

/*************************************************************************************
 * Function name  : void showWinMessage(unsigned char ducks)                         *
 *        ducks      : Value of ducks variable in main loop                          *
 *                                                                                   *
 * Created by     : James Johns                                                      *
 * Date created   : 15/11/2011                                                       *
 * Description    : Prints message to command line depending on value of ducks.      *
 *                  0x00 - Player wins,                                              *
 *                  0xFF - Player loss,                                              *
 *                  otherwise generic quiting message.                               *
 * NOTES          :                                                                  *
 *************************************************************************************/
void showWinMessage(unsigned char ducks) {
    if (ducks == 0x00) {
            /* player win */
        printf("You won! Congratulations!\n");
    }
    else if (ducks == 0xFF) {
            /* Duck win */
        printf("The Ducks won! Better luck next time!\n");
    }
    else {
        printf("Quiting\n");
    }
}
```

```c
/*******************************************************************************
 * Function name  : int checkGun(unsigned int numval)                           *
 *     returns    : 0 if trigger switch is off, else non-zero                   *
 *         numval     : Value recovered from comedi_dio_bitfield,               *
 *                      containing I/O statuses.                                *
 *                                                                              *
 * Created by     : James Johns                                                 *
 * Date created   : 10/11/2011                                                  *
 * Description    : Check status of trigger switch on rack (found in <numval>)  *
 *                  and return it's current value.                              *
 * NOTES          :                                                             *
 *******************************************************************************/
int checkGun(unsigned int numval) {

    unsigned int switches = ~((numval) >> 8) & 0xFF;
    return (switches & 0x01);
}

/*******************************************************************************
 * Function name  : void shiftDucks(unsigned char *ducks, unsigned int shouldFlip)  *
 *         ducks        : Pointer to ducks value in main loop. Is modified as per   *
 *                        description.                                          *
 *         shouldFlip   : non-zero if target duck should be shot at (i.e. flipped). *
 *                                                                              *
 * Created by     : James Johns                                                 *
 * Date created   : 15/11/2011                                                  *
 * Description    : Rotate value pointed to by ducks to the right by 1 inverting *
 *                  Least Significant Bit if shouldFlip is non-zero.            *
 * NOTES          :                                                             *
 *******************************************************************************/
void shiftDucks(unsigned char *ducks, unsigned int shouldFlip, unsigned int direction) {

    if (shouldFlip) {
        *ducks ^= 0x01;                 /* XOR flips bits regardless of initial value */
    }

    if (direction == DIRECTION_DOWN) {  /* shifting down the rack required right shift */
            if ((*ducks) & 0x01) {      /* if we're about to lose a duck over the edge of a byte */
                *ducks = (*ducks) >> 1;
                *ducks |= 0x80;         /* make sure it reappears on the other side */
            }
            else {                      /* otherwise just shift the ducks */
                *ducks = (*ducks) >> 1;
            }
    }
      else {                            /* shifting up the rack requires left shift */
            if ((*ducks) & 0x80) {      /* if we're about to lose a duck over the edge of a byte */
                *ducks = (*ducks) << 1;
                *ducks |= 0x01;         /* make sure it reappears on the other side */
            }
        else {                          /* otherwise just shift the ducks */
            *ducks = (*ducks) << 1;
        }
    }
}
```

References

Johnson, I., 2011. Introduction to Digital IO Worksheet. [E-Book]
Available at:
<http://www.cems.uwe.ac.uk/~irjohnso/courses/ets-20-1/>

Kernighan, BW. & Ritchie, DM., 1988. The C Programming Language, 2nd Edition.
Published by Prentice Hall P T R, 2011.

Comedi.org, 2008. Comedi Lib Reference Manual [Online]
Available at:
<http://www.comedi.org/doc/index.html>