



CITY *of* BRISTOL
COLLEGE

College Green Centre

Faculty of Computing and Digital Communication

Documentation Standards:

**Guidelines for Writing a Software Report
for Programs written in 'C'**

Document No: DS01/SWR	Date: 1 September 2003
Issue: 02	Author(s): Ian Shere

Contents

Summary	3
1 Title Page	3
2 Contents List	3
3 Program Description / Introduction	3
4 Function Breakdown Chart	3
5 Program Variables	4
5.1 Global Variables	4
5.2 Defined Constants	4
6 Externals	4
6.1 Header Files Required	4
6.2 External Functions Called	4
7 Function Blocks	5
7.1 Function 'Name'	5
7.1.1 Description of the Functions Operation	5
7.1.2 Calling function and Called Functions	5
7.1.3 Parameters and Return Values	5
7.1.4 Local Variables	5
7.1.5 Global Variables	5
7.1.6 Reference to Process Specification for Function.	5
9 Process Specifications	6
10 Sample Software Report	6

Software Reports for a 'C' Program

Summary

The purpose of these guidelines are to provide an aid for writing reports on programs written in the High Level Language of 'C'. It describes the format required, what each section is for and how it should be written. References are made to an example "Software Report" on an application for validating VAT Registration Numbers. This report is attached to these guidelines.

1 Title Page

This page should contain the following:

Faculty of Computing and Digital Communication
Course Name and Year
Program Title
Name of Author(s)
Date

2 Contents List

This is prepared after the rest of the report. It is a numbered list showing the section and appendices numbers on the left and starting page number on the right (remember to number all pages). It allows the reader to see the structure of the report and where to find what interests them.

3 Program Description / Introduction

This section should describe what the program is intended to do, possibly why it was chosen and the tasks it is designed to carry out. This is a top level description of what the program does to meet the objective, not an in depth description of how the program works. For example the attached report the description describes that the program is designed to validate a VAT Registration Number.

4 Function Breakdown Chart

The functional breakdown chart (or hierarchy chart or structure chart) shows graphically the 'C' functions that are used in the program and from where they are called. For example the attached report the functional breakdown chart shows that the function `main()` calls four other user functions, `Copyright()`, `Instructions()`, `VATcheck()` and `Again()`.

Functions that are not defined within your program are External Functions and need not be shown on the chart. Also if a function calls a function more than once, than that

function is only shown once.

5 Program Variables

Under this heading all global variables, defined constants and defined data types (e.g. structures) declared in your program should be listed and described. These appear before the function `main()` in the source code. Any variable listed outside of any function is global. Global variables, constants declared using `#define` and defined data types are accessible by all functions.

Local variables, those declared within functions (including `main()`) are not shown in this section but under the appropriate Functional Block.

5.1 Global Variables

Here you list the data type, identifier, any default value and a short description of each variable used. The reason for this is to provide an understanding of why global variables are used by the functions and why.

5.2 Defined Constants

Under this heading, list all the names of the constants created using `#define` preprocessor command together with a short description of what they represent.

6 Externals

As mentioned earlier, External Functions are those not defined by the programmer and therefore defined in a header file. In this section those header files (see 6.1), declared for use by the preprocessor command `#include`, are listed. This can be followed by a short description of the header files purpose. Also each external function (see 6.2) used should be listed together with a description.

6.1 Header Files Required

A listing of header files included in your source code, for example:

<code>stdio.h</code>	standard input / output functions
<code>math.h</code>	mathematical functions

6.2 External Functions Called

Each external function used should be listed with a short description. This listing should take the format of the function prototype as given in the header file which should also be shown.

7 Function Blocks

This section lists and describes each user defined function in the program (i.e. those shown in the functional breakdown chart). This, therefore excludes any external functions. For each function the following should be given

7.1 Function 'Name'

The full prototype declaration of the function

7.1.1 Description of the Functions Operation

A short description of the function purpose.

7.1.2 Calling function and Called Functions

List the functions that call this function and list function that are called from within it.

7.1.3 Parameters and Return Values

Any variables that are passed or returned by the function.

7.1.4 Local Variables

A listing of local variables declared in the function with their data type and description.

7.1.5 Global Variables

A listing of global variables used by the function.

7.1.6 Reference to Process Specification for Function.

Each function must also be described using one of the following methods of a Process Specification and a clear reference as to the page showing it should be made. Process specifications can be in the form of either a flowchart, pseudo code or decisions table for each function. In the example attached flowcharts are used.

8 Code

This section include the source code of your program. The format of the program should adhere to the 'C Programming Documentation Standard'.

Functions should be shown in alphabetical order or that in which they are declared under the Function Prototype heading. For clarity it is often useful to have each function start on a new page (also gives an indication if the function is too long if it goes over a single page).

9 Process Specifications

In this section a program flowchart, pseudo code or decision table of each user defined function is show. The flowcharts should be drawn in accordance with the standards set. Remember that a flowchart for a function should not be more than one A4 page. If you go over the page then ensure that there is only one point of entry to the function and one point of exit. The off-page connector should clearing indicate the continuing process.

Notwithstanding, all efforts should be made to keep to one page, it is often an indication that if more is needed then the function itself may be too large and contain more than one concept, hence increasing coupling.

Pseudo code should be structured and non-program dependent.

Decision tables can be either limited or extended but all stage of consolidation should be shown.

The following is an example of the referenced software report and is provided to give guidance of depth of content rather than a full example.

10 Sample Software Report

Faculty of Computing and Digital Communication

HND Computing (Software Engineering)

1999

VAT Number Verification Program

Author: A. N. Other

21 February 1999

Contents

1. Task Description	1
2. Functional Breakdown Chart	1
3. Task Variables	1
4. Externals	2
5. Functional Blocks	2
6. Program Source Code	5
7. Flowcharts	9

Software Report

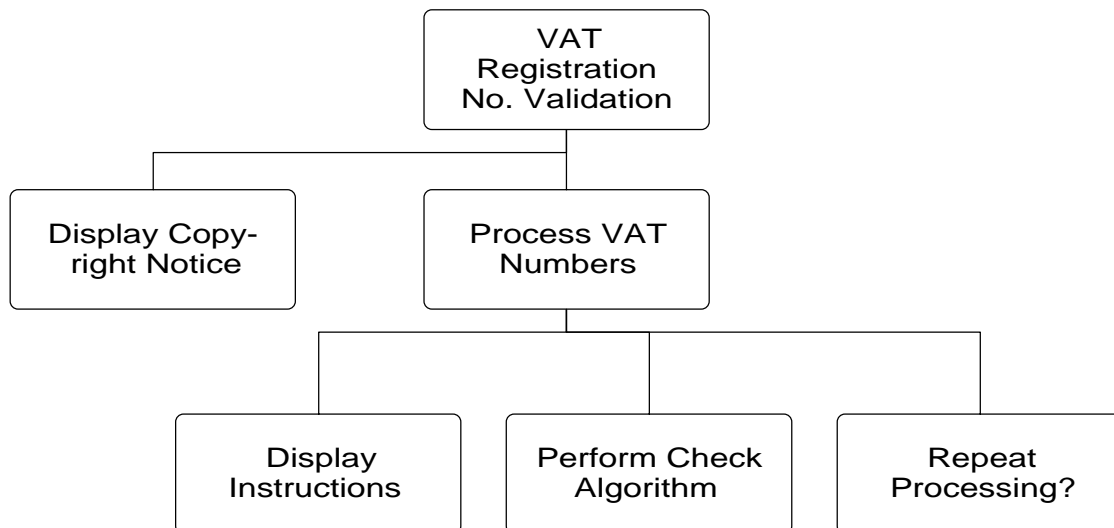
1. Task Description

This program validates a UK VAT Registration number.

The program displays a copyright message and then on pressing a key a set of instructions.

First time through the program the user is automatically requested to enter a VAT Registration number, after it has been validated (or rejected as not valid), an option appears at the bottom of the screen to enter another, 'again Y/N'. The user presses Y to repeat the validation of a new number or N to exit to the operating system.

2. Functional Breakdown Chart



3. Task Variables

a) Global Variables
None

b) Defined Values (Constants)

Name	Value	Description
LENGTH	9	Length of VAT number

c) Defined Data Types
None

Function VATcheck

void VATcheck(void)

Function description: This function requests the user to enter the VAT Registration number to be validated and then press enter. The number is checked against the algorithm and the appropriate message displayed.

Calling Function: main

Called Functions: none

Variables**Local Variables:**

Type	Identifier	Description / use
int	vat[9]	Array to express VAT number as single integers
int	vat2[9]	temporary variable for holding weighted value
int	vmod	first part of modulus calculation
int	vmod2	calculated check digit
int	chkdigit	value of check digits
int	vatchksum	value of weighed VAT number
int	i	loop counter in for loop
int	weight	initial weighting value, reduced by 1 for each position
char	vat3[9]	character array to capture VAT number

Global Variables: None

Values Passed: None

Values returned: None

Flowchart Reference: FC/vatchk page 10

Function Instructions

void Instructions(void)

Function description: This function displays the users instructions for the program.

Calling Function: main

Called Functions: none

Variables

Local Variables: None

Global Variables: None

Values Passed: None

Values returned: None

Flowchart Reference: FC/instr page 11

Function Copyright

void Copyright(void)

Function description: This function displays a short copywriter notice and requests that the user presses any key to continue to the rest of the program.

Calling Function: main

Called Functions: none

Variables

Local Variables: None

Global Variables: None

Values Passed: None

Values returned: None

Flowchart Reference: FC/copyright page 11

Function Again

char Again(void)

Function description: This function displays the message 'again y/n' indicating that the user should press 'y' to enter another number for validation or 'n' to exit to the operating system. It checks that only 'y' or 'n' is accepted in a loop.

Calling Function: main

Called Functions: none

Variables

Local Variables:

Type	Identifier	Description / use
char	KeyPress	key press value

Global Variables: None

Values Passed: None

Values returned: KeyPress

Flowchart Reference: FC/again page 12

6. Program Source Code

```
/*===== Program Information =====  
NAME OF SOURCE FILE:    VATCHK02.C  
AUTHORS:                A N Other  
PROGRAM DESCRIPTION:    Validate a VAT Registration Number  
DATE:                   24/07/93  
  
LIBRARIES*/  
#include <stdio.h>  
#include <conio.h>  
#include <ctype.h>  
#include <math.h>  
  
/*DEFINITIONS*/  
#define LENGTH 9        /*Length of VAT number*/  
  
/*GLOBAL VARIABLES*/  
    /* none */  
  
/*FUNCTION PROTOTYPE*/  
void VATcheck(void);  
void Instructions(void);  
void Copyright(void);  
char Again(void);  
  
/*===== MAIN =====*/  
void main(void)  
{  
    char Run_Again = 'y';  
  
    Copyright();  
  
    do  
    {  
        Instructions();  
        VATcheck();  
        Run_Again=Again();  
    }while(Run_Again == 'y');  
}
```

```

/* ***** FUNCTION *****
* Function Name:      VATcheck
* Operation:         performs validation of VAT registration number
* Parameters Passed: none
* Return Values:     none
***** */
void VATcheck(void)
{
    int vat[LENGTH];
    int vat2[LENGTH];
    int vmod=0;
    int vmod2=0;
    int chkdigit;
    int vatchksum=0;
    int i;
    int weight = 8;
    char vat3[LENGTH];

/* Get VAT number*/
gotoxy(26,12);
printf("Enter VAT number: ");
scanf("%s",vat3);

/* Convert VAT number to single int values*/
for(i=0;i<LENGTH;i++)
{
    vat[i]=(int)vat3[i];
}

/* Calculate or get check digits - last two numbers*/
chkdigit = (vat[7]-48)*10+(vat[8]-48);

/* Calculate checksum on weighting*/
for (i=0;i<(LENGTH-2);i++)
{
    weight=8-i;
    vat2[i]=(vat[i]-48)*weight;
    vatchksum+=vat2[i];
}

/* Calculate check digit with modulus*/
vmod=vatchksum%97;
vmod2=abs(vmod-97);

/* Print result to screen*/
gotoxy(20,15);
if(vmod2==chkdigit)
{
    printf("%s is a VALID VAT Number",vat3);
}
else
{
    printf(" * %s is NOT a VALID VAT Number *",vat3);
}
}

```

```

/* ***** FUNCTION *****
* Function Name:      Instructions
*
* Operation:         displays instruction for use on screen
*
* Parameters Passed: none
*
* Return Values:     none
***** */
void Instructions(void)
{
    clrscr();

    printf("\t\t\tV A T Registration Validation System\n\n\n");
    printf("\tThis program is designed to validate VAT registration numbers.");
    printf("\n\tEnter the VAT number to be validated (without space) and press ");
    printf("\n\tthe [ENTER] key.  ");
}

/* *****FUNCTION*****
* Function Name:      Copyright
*
* Operation:         Displays copyright notice
*
* Parameters Passed:  none
*
* Return Values:     none
***** */
void Copyright(void)
{
    clrscr();
    gotoxy(15,15);
    printf("V A T  Registration Number Validation System");
    gotoxy(15,18);
    printf("          (c) Ian Shere 1993");
    gotoxy(15,23);
    printf("          press any key to continue");
    getch();
}

```

```

/* *****FUNCTION*****
* Function Name:      Again
*
* Operation:         Asks user to repeat or exit system
*
* Parameters Passed:  none
*
* Return Values:     KeyPress
***** */
char Again(void)
{
    char KeyPress = 'x';

    gotoxy(30,20);
    printf("again (y/n)?");
    do
    {
        KeyPress=tolower(getch());
    }while(KeyPress != 'y' && KeyPress != 'n');

    return(KeyPress);
}
/* ===== end of program =====*/

```

7. Flowcharts

Programmer: A.N.Other

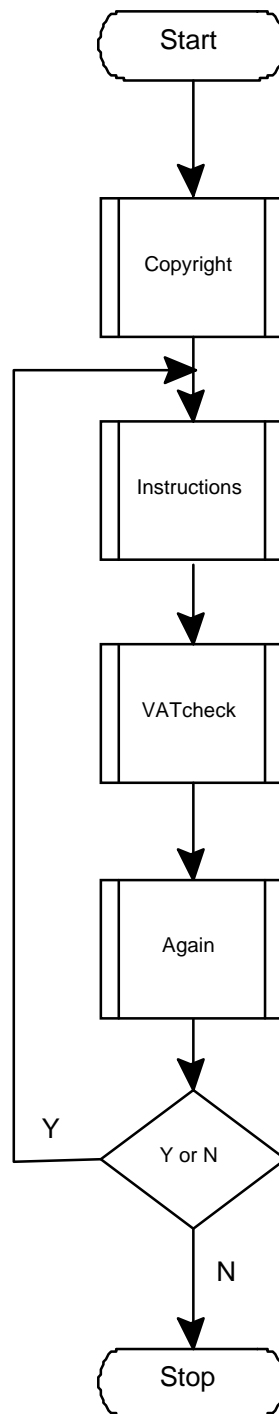
Date: 6 Feb 1999

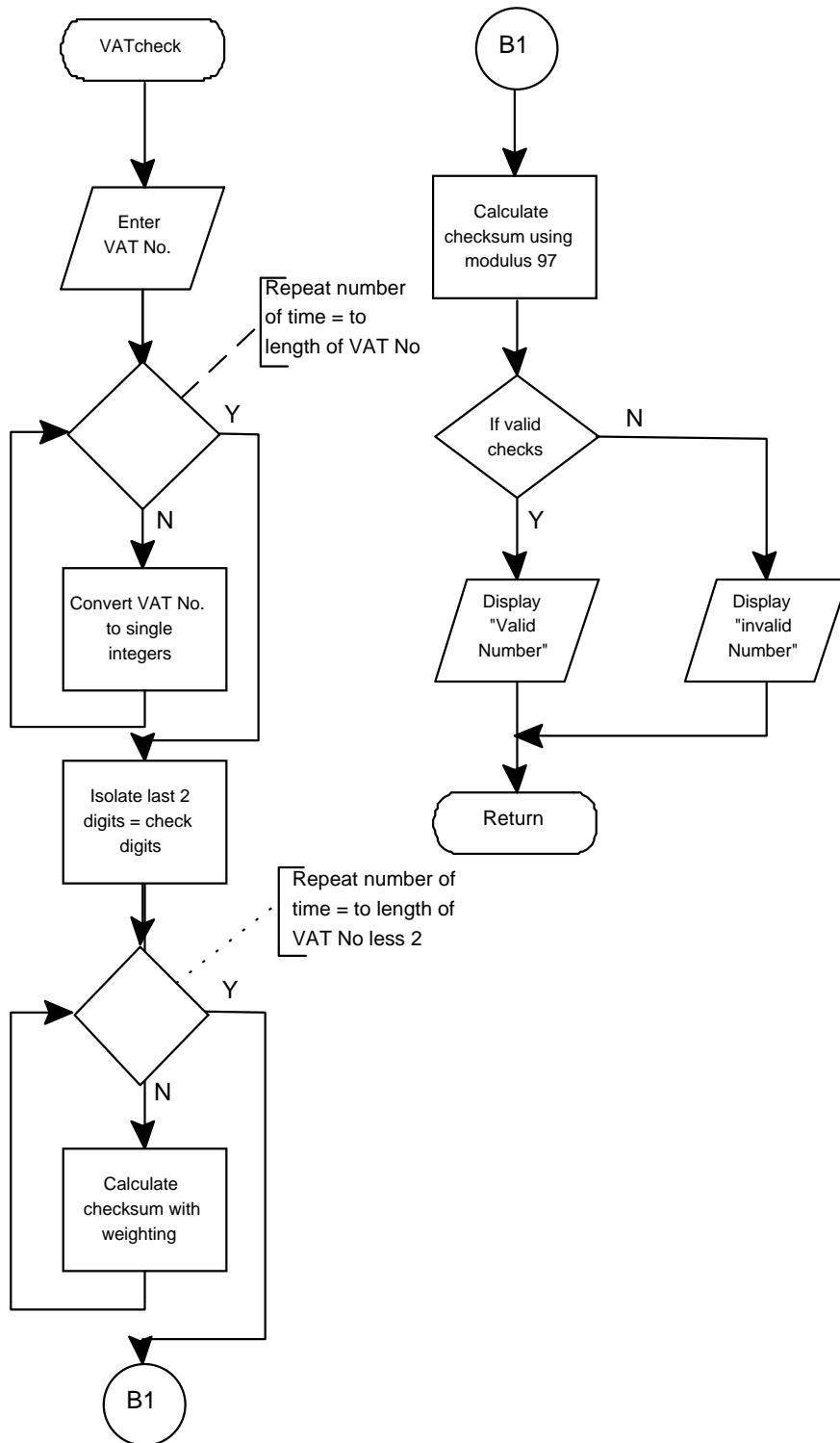
Module Ref: FC/main

Program Name: VATcheck02

Module Name: main

Function main() of VATCHK02.c

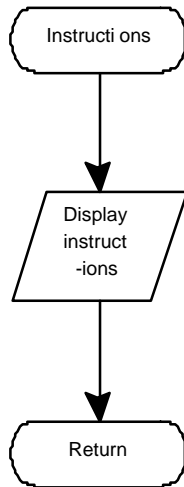




Programmer: A.N.Other
Program Name: VATcheck02

Date: 6 Feb 1999

Module Ref: FC/instr
Module Name: Instructions



Programmer: A.N.Other
Program Name: VATcheck02

Date: 6 Feb 1999

Module Ref: FC/copyright
Module Name: Copyright

