Ms Console Development Manual – September 2009

Mal Haysom
m.haysom@cartography.id.au

Abstract

This manual contains information to assist developers to refine the Ms Console code. To work on the code a developer will require a platform to compile and run command line C programs, some knowledge of the C programming language, and a more substantial knowledge of physiology. Interested parties are invited to experiment with refining Ms Console with the broader aim of improving Ms Resi.

1. Introduction

The Ms Resi (the graphical program) source code consists of several modules. The modules can be broadly classified as graphical interface code or physiological core code. The graphical interface modules have been written in C++ using Borland Builder 6. The physiological core module contains functions written in C. These functions will compile with a C++ or C compiler.

The physiological core functions are duplicated in Ms Console (the command line program). The Ms Console source code consists of two files `ms_console.c` and `ms_console.h`. These files can be compiled with a C compiler to produce a command line executable file. This executable code is similar in operation to that in Appendix IV of Dickinson's work. [1].

For example, under Unix (after loading the files into an appropriate directory) the steps to run Ms Console might be:

```
[ironbark:~/bin] user% cc ms_console.c                          (compile)
[ironbark:~/bin] user% ./a.out                                  (run executable)
            -- MacPuf -- Version 76.4 --- 1 November 1976 --    (Ms Console output)
                    -- C rework -- Mar  5 2009 --
   --- New Subject ---
   To proceed key 1, enter.
... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ...
```

Unix (with a native C compiler) is standard on many "mainframes" and is available on recent Macintosh operating systems. C compilers are available (sometimes without charge [2]) for all Microsoft operating systems. Since the structure of the code is already established, a developer could do considerable tuning of the model with only a rudimentary knowledge of C.

2. Coding Information

2.1 General

On the whole the MS Console C code is a direct translation from the Fortran code given in Appendix IV of Dickinson's work [1]. *It is strongly recommend that a developer has this work to hand.*

The designers of the Appendix IV model are Dr. C. J. Dickinson, Dr. E. J .M. Campbell, Dr. A. S. Rebuck, Dr. N. L. Jones, Dr. D. Ingram and Dr. K. Ahmed.

In 1981-2 Professor George Havenith [3], then at the Theoretical Biology Group, State University, Utrecht made some refinements to the Appendix IV code. He has made two reports on his work and the modified code available on the Chime Contributor's web site [4]. *These items are strongly recommended to developers.* The changes made by Havenith will be termed GH code.

The quality of the Ms Console code varies (especially for conditional branches) as the author became more familiar with the Fortran syntax. The original text was scanned using optical character recognition – errors occurred. Some of these errors the author may have missed and some errors he may have introduced in the translation process. The author is increasingly confident that transcription and translation errors have been eliminated, however checking against the original code is advised.

2.2 Variations from Appendix IV

Significant changes the author has made in the code are:

2.2.1 The iteration period is fixed at 1 second (Appendix IV iteration rate is variable from 2 to 10 seconds). A second is a suitable value for the graphical display process and is laughed at by modern computers.

2.2.2 The pressure units are not selectable, but fixed as mmHg. This arrangement is more convenient for Ms Resi who likes to make her own changes. As a consequence `SIMLT` is not present in `model()`.

2.2.3 The bulk of the main iteration loop code is in a separate function, `model()`. This arrangement permits the same function to be used by Ms Resi.

2.2.4 The `delay()` function uses a different technique to the Appendix IV code. The changes are a consequence of the fixed and shorter iteration period and the method used for the storage of variables in Ms Resi. The principal change is that Ms Console uses a look-up table, `dly_dt`, and an interpolation function, `interpol()`, introduced in GH, to determine the delay period rather than the algorithm used in Appendix IV. The look-up table will permit developers to make changes to the delay/cardiac output relationship more readily.

2.2.5 As a result of the differences between Fortran and C I/O formats, there are extensive changes to the I/0 code.

2.2.6 Havenith's physiological code changes have been incorporated (including the introduction of work load and heart rate) but his user interface modifications have not been implemented.

2.3 Duplicated functions

The functions in Ms Console that are duplicated in Ms Resi are:

`model()`	identical
`cnst()`	(`const()` in Appendix IV and GH) identical except for some initial code
`death()`	minor syntax differences
`delay()`	access to buffer differs
`gases()`	identical
`gsinv()`	identical
`sympt()`	identical
`interpol()`	identical

2.4 Notation:

2.4.1 Appendix IV page numbers are shown – `/* P 198 */` Code due to Havenith is noted "GH".

2.4.2 C language format labels have been constructed from the Fortran line numbers by prefixing with alpha characters – `G130:`

2.4.3 Symbols for the major variables such as `PC2CT, TC2CT, DVENT`, etc, etc have been preserved. This has been achieved by a set of macro definitions in `ms_console.h`.

2.4.4 At the user interface the Appendix IV factor numbers are preserved. For example, factor 2 is inspired CO_2 in both the MS Console model and the Appendix IV model.

4. The damp function

The damp function is used in MacPuff and Ms Resi code to model the rate of change of concentration of a solute in a fluid pool with an incoming fluid of a different concentration of solute. It is also used as means of limiting the rate of change of other variables for stability and control.

In Appendix IV code, and in Ms Resi versions prior to V1.23, the damping process was implemented as a macro. Post Ms Resi and Ms Console V1.23 the damp process is implemented as a function. The function version permits range checking of variables (only partially implemented.) The damping variable used in the Appendix IV code and the earlier Ms Resi and Ms Console versions was a 'liveliness' parameter – the larger the value the less the damping. In the later versions the damping variable is a 'damping' parameter – the larger the value the more the damping. The author believes that this arrangement is more intuitive. The units of the damping factor are seconds.

There is more detail on the damp function in Appendix B.

3. References

[1] *A Computer Model of Human Respiration*, C. J. Dickinson, MT Press, 1977
[2] (i) *Free Borland C++ Compiler*, http://www.codegear.com/downloads/free/cppbuilder
(ii) *Miracle C Compiler*, http://www.c-compiler.com/
(The author has not tested either of the above compilers.)
(iii) *Bloodshed Software Dev-C++ 5* http://www.bloodshed.net/
(A Mac Models subscriber has reported favourably on V 4.9.9.2 under Win XP Professional.)
[3] Loughborough University, http://www.lboro.ac.uk/departments/hu/staff/hugh.html
[4] UCL Medical School, http://www.chime.ucl.ac.uk/resources/Models/contributions/index.shtml

Appendix A – Factor Dump

Ms Console has a dump function `tdump()`. This function is accessed via the first bracketed option on some menu lines and provides current values of the 'T' variables.

```
  Do you want to..
          1: change, 2: continue, 3: restart, 4: inspect, 5: stop? (6, 7)
?_6
T dump 0       1      2      3       4       5       6       7      8       9
 0  20.93    0.03  100.0    0.0    0.00     0.0  3000.0     5.0    3.0   100.0
 1  100.0   100.0  760.0   37.0    0.80   13.40    12.0    14.8   45.0  3000.0
 2    0.0    -0.0    3.7  100.0    31.0     0.4    35.0     0.0    5.0     0.0
 3  14.63   100.0    7.4   7.37    4.35    7.33     0.0     0.0  343.5   146.2
 4  101.0   40.23   7.40    0.0   29.61    52.9   468.2    12.8  19.53  1544.2
 5   6.01    6.01  47.48  19.64   14.53    51.6   10.34    56.5   7.37    23.9
 6  51.47  195.25  475.8  10.99  1988.7   18.51   677.7    53.6   0.79  129.60
 7 3000.0   93.04  240.0   40.3     0.0   97.03   180.0    47.6    3.3    2.38
 8  100.0     4.7    0.0  45.71   317.8   71.42    25.6   25.52   34.1    0.98
 9   22.7     1.0   5.10  19.53   179.6   40.42    45.7   439.0   72.5     0.0
10  47.58   76.14  571.1    4.0   565.1    0.73     0.0     7.3   63.7   0.080
11  -0.55   968.7    4.2    0.0     0.0    0.00     1.0   178.0   70.0    40.0
```

Thus (in this instance) the value of `T[48]` is 19.53. T[] matches the Appendix IV variables `T1, T2, T3,` … but is the index is displaced by 1. That is, `T[0]` is equivalent to `T1` etc. The chart below may assist in identifying variables.

	0	1	2	3	4	5	6	7	8	9
0	FIO2	FIC2	CO	WRATE	FADM	BULLA	VLUNG	ELAST	VADM	AZ
1	BZ	CZ	BARPR	TEMP	TRQ	TC2MT	TVOL	HB	PCV	VBLVL
2	ADDC3	BC3AJ	DPG	PR	FITNS	SPACE	COMAX	SHUNT	VC	PEEP
3	VO2CT	PA	RPH	VPH	FVENT	BPH	BAGO	BAGC	AO2MT	AC2MT
4	AO2PR	AC2PR	DPH	XLACT	BO2PR	BC2PR	TIDVL	RRATE	RO2CT	VC2MT
5	DVENT	SVENT	PC2CT	PO2CT	TO2CT	TC2CT	BO2CT	BC2CT	TPH	RC3CT
6	VC2CT	RO2MT	RC2MT	XRESP	AN2MT	BO2MT	BC2MT	CBF	PC	DSPAC
7	REFLV	RO2PR	CONSO	RC2PR	PG	PJ	TND	RC2CT	QB	PW
8	PD	CONOM	BUBBL	TC2RF	TC3MT	VC3MT	TC3CT	VC3CT	TLAMT	RLACT
9	BC3CT	BO2AD	COADJ	EO2CT	TO2MT	TO2PR	TC2PR	VO2MT	AVENT	PL
10	EC2CT	TN2MT	TN2PR	FEV	SN2PR	EN2CT	UN2MT	RN2MT	HRATE	STRVL
11	TC3AJ	SN2MT	QA	RVADM	XDSPA	BAG	XMALE	HT	WT	AGE

Ms Console also has a dump function `cdump()`. This function is accessed via the second bracketed option on some menu lines and provides current values of the 'C' variables. There is a direct correspondence between `C[]` and `C1, C2, C3,` … . That is, `C[1]` is equivalent to `C1` etc.

```
  Do you want to..
          1: change, 2: continue, 3: restart, 4: inspect, 5: stop? (6, 7)
?_7
C dump 0       1      2      3       4       5       6       7      8       9
 0    ---    0.00    0.3    0.0    0.30     1.1     0.5   129.0  241.1     5.6
 1    0.0     0.0  713.0    0.0    0.08    1.00     0.0     1.0    0.2   240.0
 2    0.0  3250.0    1.0  270.0     0.0     1.4     0.8     0.6    0.0     7.5
 3   0.02    43.3    0.2    0.0  -245.5   29.05     1.2     0.6   24.0     0.0
 4    0.0    1.65   0.38    0.0    0.30     1.3     0.8    11.0   0.48     2.2
 5  11.50  137.50   5.95  70.00   48.00     0.0    6.00     0.1   1.00     0.0
 6   0.00    0.00    1.5  10.00     0.5    3.30     7.3     7.3  70.00    0.08
 7    0.0    0.02   14.3    0.0    24.0  184.00
```

The two dumps above were obtained with the default subject in normal conditions.

Appendix B – The damp function (a non-rigorous approach)

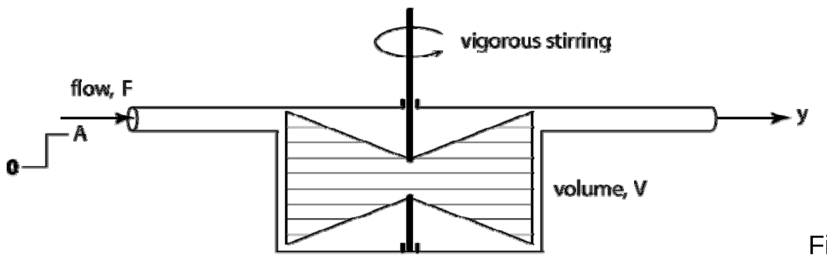

Fig 1.

A simple model of concentration changes within a fluid pool is shown in Fig 1. A step change in concentration from zero to A has occurred at the input at $t = 0$.

In an incremental time, dt, the amount of concentrate entering the pool will be $AFdt$; and amount exiting will be $yFdt$. Thus the change in concentration in the incremental time will be:

$$dy = \frac{Fdt(A-y)}{V} \qquad \text{equ. 1}$$

that is $$y + \frac{V}{F}\frac{dy}{dt} - A = 0 \qquad \text{equ. 2}$$

assuming the initial concentration of the pool to be zero then the differential equation, equ 2, has the solution:

$$y = A(1 - e^{\frac{-t}{\tau}}) \qquad \text{equ. 3}$$

where $\tau = V/F$. This parameter is known in some disciplines as the time constant of a single pole low pass filter.

The response of a low pass filter in the time domain to a step change in input is shown in Fig 2. After a period of one time constant the output has reached 63% of the span to the final value.

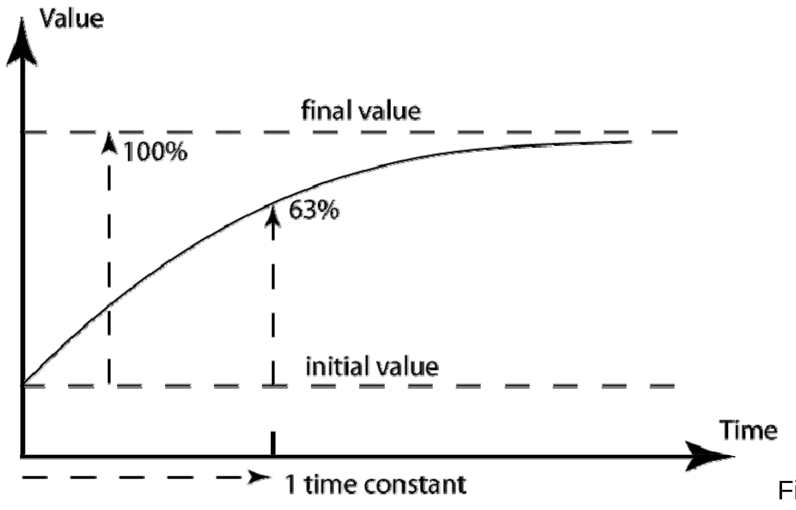

Fig 2

In a discrete time situation a low pass filter can be implemented by successive implementations of:

$$current\ output = \frac{(N-1)\,previous\ output}{N} + \frac{current\ input}{N} \qquad \text{equ. 4}$$

Where N is greater than 1, and N X (the iteration period) approximates to the time constant of the filter.

Equ 4 is of the same form as Dickinson's damping equation, equ. 5 [1, p27] – a proportional sharing of the current value and the input value.

$$new\ effective\ gas\ content = \frac{Z(ideal\ newcalculatedgascontent) + old\ \ gas\ content}{Z+1} \quad \text{equ. 5}$$

The relationship between N and Z is:

$$N = \frac{1}{Z} + 1 \qquad \text{equ. 5}$$

Conveniently the iteration period in Ms Resi is 1 second, so N is the time constant of the filter in seconds.

In the actual Ms Resi, Ms Console implementation,
`float damp(float current_input, float previous_value, float time_constant)`,
the value of the time constant is displaced by one (this simplifies conversion from the Appendix IV code) such that the minimum value for the time constant is zero, equating to no addition delay to the intrinsic delay of a finite sample period.