An Open-Source Anaesthesia Workstation (Linux)

Richard W. D. Nickalls
Simon Dales
Adrian K. Nice
The single biggest problem we face is that of visualisation

Richard P. Feynman (1918–1988)
An open source Anaesthesia Workstation

Richard W. D. Nickalls,
*formerly:* Department of Anaesthesia, Nottingham University Hospitals, City Hospital Campus, Nottingham, UK.

dick@nickalls.org
http://www.nickalls.org/

Simon Dales,
Purrsoft, Oxford, UK
simond@purrsoft.co.uk

Adrian K. Nice,
Department of Information and Computing Technology, City Hospital, Nottingham, UK.
anice@ncht.co.uk
# Contents

## Contents

I Background 1

1 An anaesthesia workstation 2

  1.1 Introduction 2
  1.2 Difficulty with funding and R&D 2
  1.3 The Linux project 3
  1.4 Modules 4
    1.4.1 Printing/archiving module 4
    1.4.2 Data acquisition and display module 4
    1.4.3 MAC display widget 5
    1.4.4 Decision-support module 5
    1.4.5 A diabetes alert module 5
    1.4.6 Drug-menu module 5
  1.5 Screen-shots 8

2 Data processing in anaesthesia 16

  2.1 Introduction 16
  2.2 History of the anaesthesia record 16
    2.2.1 Background 16
    2.2.2 Automation 17
    2.2.3 Guidelines 17
  2.3 The anaesthesia workstation 18
    2.3.1 Databases 19
    2.3.2 The future 19

3 \TeX{} in the Operating Theatre 23

4 The Datex AS/3 anaesthesia monitor 26

  4.1 Introduction 26
    4.1.1 Software version 27
    4.1.2 Available software 27
  4.2 Serial interface 27
    4.2.1 Serial port connector 27
    4.2.2 Cable connections 28
    4.2.3 Protocol 28
## CONTENTS

4.3 Transmission Request string ........................................ 29
  4.3.1 Format ..................................................... 29
  4.3.2 Transmission request command ............................. 30
4.4 Output data-string format ......................................... 32
4.5 List of parameter names ............................................ 37
4.6 Example of data output ............................................ 40
4.7 Correspondence .................................................... 48

5 Interfacing the serial port—using Perl ............................ 49
  5.1 Introduction .................................................... 49
  5.2 Device::SerialPort.pm ......................................... 49
  5.3 Sending program (as3sim.pl) .................................. 50
  5.4 Receiving program (dn-getfile2.pl) ........................... 52

6 Age corrected MAC ..................................................... 56
  6.1 Introduction .................................................... 56
  6.1.1 MAC subroutine (MS-DOS) .................................. 57
  6.2 Age corrected MAC charts ....................................... 59
  6.3 Generating the charts .......................................... 60
    6.3.1 A data file for a single iso-MAC curve .................. 61
    6.3.2 mathsPIC script for drawing the whole graph .......... 62
    6.3.3 Final mathsPIC program for making the version of BJA charts for the Oxford Handbook of Anaesthesia .......... 75
    6.3.4 Output mac-iso8t.mt code from the previous mathsPIC program 79
  6.4 References ..................................................... 89

7 Age & temperature corrected MAC-charts ........................... 92
  7.1 Introduction .................................................... 92
  7.2 Program: macatdes2.pl ......................................... 93
  7.3 Program: macatdes2.m ......................................... 102
  7.4 TeX file macdes034.mt ........................................ 107
  7.5 Program macATdes2batch.pl for automating the pdf output .... 115
  7.6 Test program MACsolver.pl .................................... 124
    7.6.1 Output datafile (macsolver-out.txt) .................... 130
  7.7 References ..................................................... 132

II The front-end coordinating program .................................. 134

8 The Perl/Tk front-end ................................................ 135
  8.1 Introduction .................................................... 135
  8.2 The BASH script runcamomile ................................ 136
  8.3 Pressing the “RUN” button .................................... 137
    8.3.1 Program: tklaunch2.pl ................................ 137
  8.4 Useful Linux tools to use with the launcher ................. 140

9 The launchcam12.pl program ......................................... 141
  9.1 Introduction .................................................... 141
  9.2 The program launchcam12.pl .................................. 143
CONTENTS

III The data program—Camomile 147

10 System overview 148
  10.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 148
  10.2 Modules . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 148
    10.2.1 Graphical front-end module . . . . . . . . . . . . . . . . . . 148
    10.2.2 Data collection and display module . . . . . . . . . . . . . 149
    10.2.3 Printing module . . . . . . . . . . . . . . . . . . . . . . . . . 149
    10.2.4 Epidural database . . . . . . . . . . . . . . . . . . . . . . . . 149
    10.2.5 Help files . . . . . . . . . . . . . . . . . . . . . . . . . . . . 149
  10.3 Directory structure . . . . . . . . . . . . . . . . . . . . . . . . . . . 149

11 The Camomile program 151
  11.1 Directory listing of camomile.v.0.1_040413b . . . . . . . . . . 151

12 Configuration files 156
  12.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 156
  12.2 c.as3rn.conf . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 157
  12.3 x.configrn.conf . . . . . . . . . . . . . . . . . . . . . . . . . . . . 159
  12.4 projectdir.conf . . . . . . . . . . . . . . . . . . . . . . . . . . . . 160
  12.5 w-monitor-datexas3.conf . . . . . . . . . . . . . . . . . . . . . . 160
  12.6 People.conf . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 162
  12.7 Drugs.conf . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 162
  12.8 x-widgets.conf . . . . . . . . . . . . . . . . . . . . . . . . . . . . 163

13 Drug dictionary 167
  13.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 167
  13.2 Initial drug list . . . . . . . . . . . . . . . . . . . . . . . . . . . . 168
  13.3 Download bundle . . . . . . . . . . . . . . . . . . . . . . . . . . . . 169
  13.4 VTM File format . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 169
  13.5 Perl program dn-dmd5.pl . . . . . . . . . . . . . . . . . . . . . . . 170
  13.6 Perl program reverse.pl . . . . . . . . . . . . . . . . . . . . . . . . 178
  13.7 Initial data listing . . . . . . . . . . . . . . . . . . . . . . . . . . . 179
  13.8 The ordered list . . . . . . . . . . . . . . . . . . . . . . . . . . . . 179
  13.9 Adding drugs to the list . . . . . . . . . . . . . . . . . . . . . . . . 180
  13.10 Perl program add2list.pl . . . . . . . . . . . . . . . . . . . . . . 181
  13.11 Logfile generated by add2list.pl . . . . . . . . . . . . . . . . . . 182
  13.12 Final list for pull-down menu . . . . . . . . . . . . . . . . . . . . 184

14 Diabetes decision-support system 185
  14.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 185
    14.1.1 Kalar...
Part I

Background
Chapter 1

An anaesthesia workstation

1.1 Introduction

Since 1994 I have run an on-going research-project to develop an open-source anaesthesia workstation for free use by the NHS in the operating theatre. What started as a small project to automate the production of the anaesthetic record, has since developed into a clinically-useful support tool for anaesthetists.

During the period 1996–2001 I developed a working theatre-based prototype MS-DOS program, which was used in the thoracic operating theatre (City Hospital). A paper anaesthesia record (for the patient notes) was output using the open-source programs GNUplot (for graphic trends) and LATEX (for typesetting). Much of the initial work relating to interfacing medical devices via the serial port was published as a book by Cambridge University Press (Nickalls and Ramasubramanian, 1995).

During 2002-2004 the program was ported to Linux (see Section 1.3) and expanded to include alarms, some basic decision-support, as well as the calculation of various useful so-called value-added real-time parameters, for example, age-dependent MAC (Nickalls and Mapleson 2003).

1.2 Difficulty with funding and R&D

During the past eight years or so I have tried to collaborate with various university departments with a view to R&D. Discussions with the Nottingham University Departments of Computing and Department of Electrical and Electronic Engineering in 2005 did not lead anywhere owing to lack of funding. Unfortunately funding has not been forthcoming (an EPSRC grant application in conjunction with Dept Med Physics, Liverpool Univ Hosp was rejected—see details below), and therefore serious development has stalled. These ventures are summarised below.

---

1 My original version was in QuickBasic 4.5. It was later ported to PowerBasic 3.5
2 Minimum Alveolar Concentration (MAC) of an anaesthetic agent is an index of anaesthetic potency. A typical anaesthetic is associated with approximately 1–1.2 MAC.
Collaboration with Leicester University—2001

During the academic year 2001–2002 I formed a collaboration with the Department of Electronic & Software Engineering, University of Leicester, UK, with a view to porting the existing program to the Linux operating system and making several enhancements, including a module for accessing existing patient data via the City Hospital Information System (HIS) (in conjunction with Andy Smith, Department of Information and Computing Technology, City Hospital). During this period four engineering students worked on parts of the program for their final year practical modules. Unfortunately however, the relatively short time allowed the students for their project was insufficient for a prototype to be developed, and the project terminated after one year.

Collaboration with Liverpool University — 2002

Significant interest in this project was shown by the Department of Clinical Engineering at the Royal Liverpool University Hospital. Unfortunately, however, a joint grant application (2004) to the EPSRC (Engineering and Physical Sciences Research Council) in conjunction with the Department of Clinical Engineering (RLUH) to fund research and development was not successful.

Collaboration with Nottingham Trent University — 2005

In December 2005 we explored a collaboration with Department of Computing and Informatics, Nottingham Trent University with a view to rewriting the software and implementing a more robust and scalable architecture. However, funding was unsuccessful.

Collaboration with Nottingham University Hospitals — 2008

In December 2008 we embarked (in conjunction with Professor Mahajan, Department of Anaesthesia) on a collaboration with the Department of Medical Physics at the Nottingham University Hospitals, City Hospital Campus, with a view to developing a stand-alone real-time MAC-monitor for use by anaesthetists. This project failed to progress owing to insufficient resources.

1.3 The Linux project

Towards the end of 2002 I formed an ‘open-source’ collaboration with Simon Dales (Software engineer, Oxford, UK). During 2003–2004 the original program was rewritten from scratch for the Linux operating system—the data acquisition and display module in C/C++ by SD, and the printing & processing modules in Perl, GNUplot and LATEX by RWDN.

The resulting working ‘stand-alone’ Linux prototype has been ‘up-and-running’ in the ‘thoracic’ operating theatre at the City Hospital, Nottingham since 2004, used by both consultant and trainee anaesthetists, and has been very successful (see illustrations at the end). The program gives a continuous trend display of a variety of measured and derived parameters, as well as ‘help’ and other general information, allows inputting of drug and other information, and automatically prints out the Anaesthesia Record at the end of the operation in a form suitable to be placed directly into the patient’s notes as a final record. In time we would like to incorporate a suitable database, develop
smart-alarm and decision-support software, extend the on-line help facility, and to explore connectivity with the hospital information system (HISS).

Support is ‘in-house’ by the Group members (see below). Electrical safety issues relating to the hardware are overseen by Ged Dean (Medical Physics, City Hospital); Linux support is by Adrian Nice (Department of Information and Computing Technology, City Hospital).

This project has been the subject of many lecture presentations over the last few years, and these are listed in the References.

**Group members**

The project team consists of the following members.

- **Richard WD Nickalls**, Consultant in Anaesthesia & Intensive Care, Department of Anaesthesia, Nottingham University Hospitals, City Hospital Campus, Nottingham, UK.
- **Simon Dales**, Software Engineer, PurrSoft, Oxford, UK.
- **Adrian K Nice**, Senior Systems Developer, Department of Information and Computing Technology, Nottingham University Hospitals, City Hospital Campus, Nottingham, UK.
- **Ged Dean**, Clinical Engineer, Department of Medical Physics, Nottingham University Hospitals, City Hospital Campus, Nottingham, UK.

**1.4 Modules**

The Anaesthesia Workstation project currently consists of four software components as follows (see screenshots at the end).

**1.4.1 Printing/archiving module**

This is written in the Perl language (Perl5) and coordinates data manipulation, graph plotting (using GNUplot), and typesetting (using \LaTeX\ 2e). An electronic form of the *Anaesthesia Record* and associated data and programs is made available for easy viewing via a HTML front-end.

A paper version of the *Anaesthesia Record* in a format suitable for placing directly into the patient notes generated and is printed in the operating theatre at the end of anaesthesia. This consists of (a) the graphic trends (a series of 1-hour graphic records of measured parameters), and (b) the data log and keyboard entries (events, procedures, drugs given, blood lost etc.).

**1.4.2 Data acquisition and display module**

This is written in C/C++ and uses the Qt library (standard with Linux systems). The program accesses serial data from the Datex AS/3 anaesthesia monitor and displays the data in trend and tabular formats on the screen. The operating theatre PC runs Mandriva-Linux on a Dell Pentium PC.
1.4.3 MAC display widget

A key invention/development by our group is the real-time MAC display widget (Figure 1.1), which is positioned in the lower right part of the main display screen (Figure 6.1). This widget displays the current MAC value, and implements an alerting colour change (to red) to warn the anaesthetists of an out of range value, and hence greatly facilitates the avoidance of inadvertent awareness of the patient under anaesthesia.

![Example of the real-time age-corrected MAC-widget displayed by the anaesthesia workstation software (c) Nickalls RWD and Dales S (1996–2009)) interfaced to the Datex S/5 monitor. If the corrected MAC is too low or too high (as shown in this case—total MAC 1.87) then, in addition to sounding an audible alarm, the dial of the MAC-widget turns red.](image)

The development of this displayed real-time MAC data follows from our earlier work on developing charts facilitating the determination of age-corrected MAC for anaesthetists (Nickalls and Mapleson 2003). These charts are also included in a widely used anaesthesia handbook (Nickalls 2006).

1.4.4 Decision-support module

This is an HTML information system offering decision-support, information on relevant drugs, medical conditions, etc. for anaesthetists in the operating theatre. The emphasis is on an intuitive well structured menuing system to enable items to be found easily and quickly. We hope to include suitable commercially available HTML texts as they come available.

1.4.5 A diabetes alert module

This is a program (in Perl) which makes use of the Linux Kalarm utility. Tk widgets are used to present a menu which allows the user to quickly set special alerts to prompt regular monitoring of blood glucose. A ‘help’ system allows the user to access protocols for the insulin management of diabetic patients during major surgery.

1.4.6 A drug-menu module

This is a pull-down drug menu from which the anaesthetists can select a drugname for addition to the drug record. This database is the standard DM+D EU drug-list database (downloaded from the NHS DM+D website) which is updated weekly. The list currently consists of about 2500 drugs.
Figure 1.2: Screenshot showing the MAC widget in a red-alert state. Note that the main display screen (pushed to the LHS) is designed so that all the important minute-to-minute data and alarm data is positioned on the RHS of the main display screen, and so allows the main display screen to be moved towards the left in order to view other data, files, or images as required. In this example a file is opened on the RHS of the PC screen.

References


- Nickalls RWD (2005a). Interfacing the PC to medical equipment. Invited talk to the Nottingham & East Midlands Society of Anaesthesia (NEMSA) (Queen’s Medical Centre; April 8, 2005). [mini-symposium on Information Technology]

Invited presentation to the Nottingham Linux Users Group, Nottingham, UK; March 16, 2005.


- Nickalls RWD (2004b). *Age corrected MAC.* Invited talk to the Nottingham & East Midlands Society of Anaesthesia (NEMSA) (Queen’s Medical Centre; October 8, 2004). [mini-symposium on MAC, elderly patients and confusion]

- Nickalls RWD (2004c). An open-source anaesthesia workstation for the NHS. [Presentation to the Patient Safety Network meeting; IBIS Hotel, Birmingham, UK; April 27, 2004]


- Nickalls RWD (1998b). *\LaTeX* in the operating theatre: an Anaesthesia application. [Invited presentation to the Annual UK \LaTeX Users Group meeting, Cambridge, UK. (September 21–22, 1998)]


---

3 A meeting concerned with the IEEE-1073 Standard regarding computer interfacing to Medical Devices.

1.5 Screen-shots

Figure 1.3: Program running in Theatre 1
Figure 1.4: Screen showing full width option for the lower half of the screen. Top half shows saturation (red), blood pressure (dark blue), ecg heart rate (green); oximeter heart rate (black), inspired oxygen (red), central venous pressure (pale blue)—current values are shown in top right window. Bottom half of the screen shows expired CO$_2$ (blue), inspired CO$_2$ (red), tidal volume TV (blue), respiratory rate (green), expired anes agent (sevoflurane, red) and age corrected MAC (blue)

Figure 1.5: Anaesthetic record — HTML front-end
Figure 1.6: Anaesthetic record — graphic record

Figure 1.7: Anaesthetic record — drug record
Figure 1.8: Screen showing the initial graphic front-end (right) which allows the user to either start the program, or access other utilities. For example, clicking on the <epidural> button runs the Epidural and Double-lumen tube database program (shown on the left of the screen) which predicts epidural depth and tube length for a given height and weight.

Figure 1.9: Screen showing the log, alarm, MAC and trend windows. The blood pressure (BP) is highlighted in yellow in the alarms window, indicating a minor departure from the ‘normal’ range.
Figure 1.10: Screen showing use of the Patient Data widget

Figure 1.11: Screen showing the Datex controller (bottom left of screen)
Figure 1.12: Screen showing a 'help' file viewed using the KDE web browser

Figure 1.13: Screen showing real-time data plus preview of printout
Figure 1.14: Screen showing help desk home page.

Figure 1.15: Help desk showing the drug info for Calcium.
Figure 1.16: Screen showing preview of the Anaesthetic Record about to be printed
Chapter 2

Data processing in anaesthesia

2.1 Introduction

The next significant change in anaesthesia practice will very likely be related to data processing, particularly in the areas of smart alarms and decision support. While development and take-up in the operating theatre is almost imperceptible just now, the future surely lies in computers offering anaesthetists seriously useful facilities and real-time information. The initial motivation with regard to data handling lay in automating the anaesthesia record. However, while this technology has been effectively solved for over 15 years (see Kenny 1990), the take-up by anaesthetists remains almost zero.

2.2 History of the anaesthesia record

The documentation of events, procedures undertaken, physiological parameters (vital signs) which are associated with the process of anaesthesia (for example, in conjunction with surgery or an intensive care setting) is known as the Anaesthesia Record. This record serves two main functions, namely (a) medical (the moment-to-moment drug history and vital-signs serves as a useful practical aid), and (b) medico-legal (the anaesthesia record is a legal document in its own right, setting out the facts as they unfold during an anaesthetic).

2.2.1 Background

Effective surgical anaesthesia was established in 1846 following the discovery of the effects of inhaled diethyl-ether (“ether”). Although John Snow (1813–1858), Joseph Clover (1825–1882), and Mounier (1855) demonstrated the importance of monitoring the pulse and respiration during anaesthesia (Ellis, 1995; Rushman, Davies and Atkinson, 1996) it was not until 1894, at the Massachusetts General Hospital, Boston, that surgeons Ernst A Codman (1869–1940) and Harvey Cushing (1869–1939) established the practice of keeping a careful written record (on graph paper) of the patient's pulse and respiration rate during operations—known as the ‘ether chart’ (Beecher, 1940; Hirsch and Smith, 1986). Apparently this was prompted by a death under anaesthesia in 1893 (Rushman,
Davies and Atkinson 1996, p 128). In 1901 they started including measurements of the arterial blood pressure using the newly described apparatus of Scipione Riva-Rocci (1863–1937) of Turin (Cushing 1902; Cushing 1903; Rushman, Davies and Atkinson, 1996, p 157).

Ralph Waters (1936; 1942) championed and emphasised the importance of written anaesthetic records, and later Noseworthy (1945) produced special cards on which to record anaesthetic details (see Rushman, Davies and Atkinson (1996), p 111, for an illustration).

2.2.2 Automation

The first mechanical device capable of printing an anaesthetic record was the Nargraf machine of 1930 developed by EI McKessons (Westhorpe 1989), which generated a semi-automated record of inspired oxygen, tidal volume and inspiratory gas pressure.

Since then little of real technological significance was developed in the area of anaesthesia monitoring until the 1970s, when advances in chip technology gave rise to clinically useful portable electronic devices for measuring such things as arterial and central venous blood pressure, breath-by-breath concentrations of oxygen, carbon dioxide and inhalational anaesthetics, pulse oximetry, and of course, small computers.

From an interfacing point of view, a very significant and far reaching feature was incorporated into virtually all early medical monitoring devices, namely a specialised serial communications interface known as the RS-232 port. Equally significant, therefore, was the decision by IBM to incorporate the same RS-232 port into the IBM Personal Computer which appeared in 1981. Fortunately all IBM-compatible PCs since then have also incorporated the RS-232 serial port.

Owing to the widespread use of the RS-232 interface in medical equipment it soon became a relatively easy matter to use a PC to access the numerous measured parameters output by patient monitoring devices, and consequently anaesthetists increasingly explored methods for automating data collection and processing, with a view to developing useful trend displays of measured data, real-time calculation of derived parameters, and hard-copy data printouts.

The RS-232 interface is set to be replaced in the relatively near future by the Medical Interface Bus (MIB; IEEE-1073). This a high-tech high-speed medical plug-and-play version of the familiar domestic USB interface, and will greatly facilitate medical device inter-connectivity, largely by allowing the relevant interface software to be more easily standardised.

An automated anaesthesia record is significantly superior to the usual hand-written record, since it samples data more frequently and more accurately, and hence it has significant medico-legal advantages regarding the documentation of patient care, particularly during complicated and/or unstable cases.

2.2.3 Guidelines

The Royal College of Anaesthetists has published a summary of what data ought to be collected (in addition to the electronic data from the anaesthesia monitors) as part of the Anaesthesia Record (Adams 1996), building on the work of Lack et al. (1994). The extent to which these guidelines are actually being met has also been looked at (Smith, 1997). The required record set which appears to be emerging, consists of a number of fields within the following general categories: pre-, per- and post-operative information, untoward events and hazard flags.
2.3 The anaesthesia workstation

Much work has gone into studying the anaesthetists’s workload (Weinger et al. 1997; Byrne, Sellen and Jones 1998; Leedal and Smith 2005), and it is clear that computerisation would free anaesthetists and nurses from much of the work of documentation (e.g. drug doses, procedures, measured parameters etc.), releasing significant amounts of time which could be better spent on direct patient care and vigilance. Anaesthesia/ITU information and record-keeping systems clearly offer the advantage of allowing the anaesthetists and nursing staff to concentrate fully on the patient, leading to enhanced vigilance and improved patient care and safety.

For example, Kennedy et al. (1976) showed that anaesthetists commonly spend 10–15% of their time producing the handwritten record. Similarly, Smith (1997) pointed out that about 10% of the anaesthetists’ time was related to record keeping, and that if this were to increase then this would likely be to the patient’s detriment. A similar study by Wong et al. (2003) showed that an ICU information system reduced the time spent by nurses on documentation by 31%, with the significant benefit being that almost half of the time saved was transferred to patient assessment and direct patient care.

Secondary data processing by anaesthetists in the UK is well behind other countries in this regard, with electronic data collection being actively supported by foreign health organisations. For example, in 2001 a special newsletter issue of the Anesthesia Patient Safety Foundation (APSF) was devoted to Information systems in anaesthesia (APSF, 2001). In 2002 the APSF formally endorsed the use of automated anesthesia information management systems (AIMS) as the following quote indicates (see also www.gasnet.org/societies/apsf/).

In this context it is heartening that the . . . APSF has recently endorsed the use of automated anesthesia information management systems (AIMS):

“The Anesthesia Patient Safety Foundation endorses and advocates the use of automated record keeping in the perioperative period and the subsequent retrieval and analysis of that data to improve patient safety.”

Gage, 2002.

Anaesthetists urgently need to harness the power of computing technology in a way which can help them both in the operating theatre and in the clinic, most likely via some form of anaesthesia workstation. While such systems will probably be commercial, this is not necessarily the only route. Providing anaesthetists take some interest in the details, it is not impossible to imagine useful systems being developed along the Open Source model (cf. the immensely successful Linux operating system).

The emphasis for such a workstation needs to be on helping the anaesthetist give a safe anaesthetic during difficult circumstances. It would access data from various sources via the Medical Interface Bus (e.g. anaesthesia monitors, HIS) and then process the data in various ways; for example, data storage, making the anaesthesia record, smart alarms, decision support, data export, emergency communications. It is important that such workstations are developed separately from the commercial anaesthesia monitors and anaesthesia machines, rather than being integrated with them.

Even at a basic level computers in the operating theatre already offer significant advantages over and above creating good anaesthesia records. For a long time now it has been relatively straightforward to access data from anaesthesia monitors (Nickalls and Ramasubranian 1995; Nickalls 1998) and display warnings, information and value-
added parameters; for example, real-time age-corrected MAC (Nickalls and Mapleson, 2003).

Of course commercial information and anaesthesia record systems are available (e.g. the NarKoData system (IMESO, GmbH, Huttenberg, Germany)—see Benson et al. 2000), but they are generally far from ideal. For example, these systems tend to be extremely expensive and are generally machine specific (e.g. the Datex AS/3 system), and are quite awkward to use. The existing commercial systems tend to be most useful in collecting what one might loosely call "hospital/theatre management" information, while being relatively unhelpful in facilitating anaesthesia-related activities, or even generating good quality records. These latter failings largely account for the poor take-up of commercial systems by anaesthetists.

Computerisation also offers a significant research benefit. For example, in a study by Muller et al. (2002) anaesthetists were able to search the database of their automated anaesthesia record-keeper and establish useful risk factors predictive of subsequent inotropic support requirement following cardio-pulmonary bypass.

2.3.1 Databases

Extracting data from big databases requires a good data dictionary (Sanderson and Monk 2003) as, for example, the currently well advanced SNOMED Clinical Terms program (SNOMED-CT) (http://www.snomed.org/snomedct/), which is a dynamic health care terminology infrastructure being developed as part of the NHS National Program for Information Technology (NPfIT). A demonstration program can be accessed from the SNOMED-CT home page.

Another NPfIT dictionary database of interest to anaesthetists is the Dictionary of Medicines and Devices (DM+d) (http://www.dmd.nhs.uk/). This consists of a number of coordinated XML-encoded pharmaceutical-related databases, which also incorporate the associated SNOMED encoding. Of particular interest to anaesthetists is the Virtual Therapeutic Moiety (VTM) database of approximately 2000 official drug names which are to be used henceforth in all computer interactions relating to drugs. This list is updated weekly and can be downloaded from the website (password required). This list is currently incorporated into the experimental program used in the thoracic theatre.

2.3.2 The future

The future holds the exciting prospect of developing sophisticated (and possibly Open Source) anaesthesia workstations giving anaesthetists access to good data displays and trends, sophisticated alarms (smart-alarms), real-time (and predictive) modelling for drugs and physiological parameters, information management and decision-support systems (Sanderson, Watson and Russell 2005). A good overview of what might be possible (in a USA office setting) was presented recently by Gage (2002).

References


- Fulton JF (1946). Harvey Cushing—a biography. (Charles C Thomas, Springfield, IL, USA).


• Middleton H (1958b). *Brit Med Bull*; 14, 42. [from Middleton (1958a)]

• Mounier CCR (1855). *Acad. Sci. Paris*; vol(40), p 530. [from Rushman, Davies and Atkinson (1996)]


• Noseworthy M (1937). *St. Thomas’s Hospital Reports (London)*; 2, 54. [from Rushman, Davies and Atkinson (1996)]

• Noseworthy M (1943). *British Journal of Anaesthesia*; 18, 4 (? p 160). [from Oldham (1963); Middleton (1958)]

• Noseworthy M (1945). *Anesthesia and Analgesia*; 24, 221. [from Rushman, Davies and Atkinson (1996)]

• Noseworthy M (1953). *Anaesthesia*; 8, 43. [from Noseworthy (1963)]


• Sanderson IC and Monk TG (2003). Standard anesthesia terminologies: how can we avoid wasting the data we collect? *ASA Newsletter*; 67, November. [http://www.asahq.org/Newsletters/2003/11_03/sanderson.html](http://www.asahq.org/Newsletters/2003/11_03/sanderson.html) [The November ASA Newsletter was a special issue on “Performance and outcome measures”]


Chapter 3

\TeX in the Operating Theatre: an Anaesthesia application.

R. W. D. Nickalls BSc, PhD, MBBS, FRCA.
Consultant in Anaesthesia & Intensive Care,
Department of Anaesthesia,
City Hospital, Nottingham, UK.
dick@nickalls.org

Abstract

This article describes the author’s experience of using \TeX for typesetting the Anaesthesia Record as part of an automated data-collection system developed for use in the operating theatre.


Introduction

Since the theme of this year’s conference is “Integrating \TeX with the surrounding world” I would like to describe my integration of \TeX with the world of the operating theatre—specifically with the domain of anaesthesia.

One of the many things that occupies anaesthetists during an operation is documentation. This takes the form of a log of various physiological parameters (see Figure 1), drugs used, blood lost, fluids administered, procedures performed etc., otherwise known as the Anaesthesia Record. Since this is generally a handwritten record, the documentation side of things can become rather neglected during busy periods, and consequently, anaesthetists are increasingly using computers to automate the collection of such data. This has many advantages including allowing real-time processing of data, generation of various derived parameters, and greatly enhanced information display facilities.

Collecting and processing the data

Since most monitoring equipment used in Critical Care environments has an RS-232 serial interface the process of data-collection, construction of trend graphics, formatting and typesetting can be automated reasonably easily.

My own system is a menu-driven research application which uses compiled QuickBASIC programs to coordinate the
access, display and printing of both real-time physiological data and keyboard inputs. The printing module uses \LaTeX{} to typset the text and graphics to create the Anaesthesia Record in a format suited to the hospital notes.

The data from the various anaesthesia monitors is accessed via the serial port using a multiplexing device. Individual parameters are then extracted using the relevant software for each of the various monitors—see [1] for interfacing details relating to particular anaesthesia monitors. Unfortunately there is currently no standardisation with regard to data formats for medical monitoring devices, but this may well soon change with the development of the new international Medical Information Bus (MIB) standard (IEEE 1073).

During anaesthesia the program accesses and displays all the data in real-time as graphic trends, as well as deriving a number of so-called ‘value-added’ parameters and processing keyboard entries. At the end of the operation the program typesets the text and graphics to form the Anaesthesia Record.

The graphics are created using the excellent freeware program GNUPLOT\footnote{http://www.cs.dartmouth.edu/gnuplot-info.html} which allows batch processing and will output graphics in \LaTeX{} picture format.

Armed with the maximum and minimum values for each of the measured parameters, the program writes the GNUPLOT input files, and then calls GNUPLOT, outputting the graphics in \LaTeX{} picture format, and placing them into the appropriate directories. The program then writes the \LaTeX{} input .tex file, and then calls \LaTeX{} to typset the text and graphics. Finally the .dvi file is printed and put into the hospital notes. In practice all this is performed locally within the operating theatre, such that the Anaesthesia Record is printed and placed in the patient notes just as the patient is returned to the recovery area. Figure 1 shows the graphics page of a typical Anaesthesia Record.

Advantages of ASCII-based systems

The fact that both \LaTeX{} and GNUPLOT use inputs which are ASCII-based has the great advantage that their input files can be written on-the-fly by the coordinating computer program. Such flexibility allows the final text and graphics of the document to be tailored to the data. For example, this allows the axes of graphs to be automatically adjusted depending on maximum and minimum values. Similarly, text layout can be made to vary depending on the particular keyboard entries made during the operation.

Small is beautiful

An automated system for data collection, display and printing has clear advantages over the usual hand-written method; it is certainly a more accurate record, and physiological data can be sampled much more frequently. Furthermore, keyboard entry of drugs and other information can be made simple and fast by careful design of the interface.

Since this is a specific stand-alone application, it is possible to use a much cut-down version of \LaTeX{} consisting only of the essential files, fonts and style options required for the application, with the effect that the size of the printing module can be made extremely small. A not insignificant bonus, therefore, of using \LaTeX{} as the typesetting engine is that I am able to make use of old 386 PCs having relatively small hard-drives, which have been discarded by my memory-hungry colleagues!

References

ANAESTHETIC SHEET

Theatre 1, City Hospital, Nottingham, UK.

DATE: 18 August 2000
OPERATION: Laparotomy
ANAESTHETISTS: RWD Nickalls et al.
SURGEONS: An Other et al.

JOHN DOE
dob 24/01/1925
Hosp No: 123456789
Nightingale Ward
Age: 75

Figure 3.1: Example of the graphics section of a typical Anaesthesia Record. The six graphs are output by GNUPLOT in \LaTeX{} picture format. The record shows blood pressure (BP), heart rate (HR), central venous pressure (CVP), oxygen saturation of haemoglobin (Sat), inspired oxygen (O$_2$), inspired nitrous oxide (N$_2$O), expired carbon dioxide (CO$_2$), tidal volume, respiration rate, isoflurane and MAC.
Chapter 4

The Datex AS/3 anaesthesia monitor

4.1 Introduction

The Datex-Ohmeda\textsuperscript{1} AS/3 and CS/3 monitors are versatile modular anaesthesia monitoring systems, which have an asynchronous serial interface for data acquisition. The various modules access a comprehensive range of physiological parameters. Note that the technical latest manual regarding the serial interface is \textit{AS/3 and CS/3 Monitor Product specification—computer information.} v.3.4 March 1999 (G-version update by Rene Coffeng, 23/Nov/1998).

The electrical safety Type classifications of the various Applied Parts (e.g. NIBP cuff, temperature probe) are shown in Table 4.1.

<table>
<thead>
<tr>
<th>Applied Part</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECG</td>
<td>CF</td>
</tr>
<tr>
<td>NIBP</td>
<td>BF</td>
</tr>
<tr>
<td>Invasive BP/CVP/PA</td>
<td>CF</td>
</tr>
<tr>
<td>Temperature probe</td>
<td>CF</td>
</tr>
<tr>
<td>Cardiac output</td>
<td>?</td>
</tr>
</tbody>
</table>

Table 4.1: Applied Parts and their Types.

\textsuperscript{1}Datex-Ohmeda Division, Instrumentarium Corp., P. O. Box 900, FIN-00031 Datex-Engstrom, Finland. Tel: +358–9–39411; FAX: +358–9–146–3310.

Datex-Ohmeda, 71, Great North Road, Hatfield, Hertfordshire, AL9-5EN, UK; Tel: 01707-263-570, FAX 01707-260-065.
4.1.1 Software version

Software is frequently revised, and different monitors may have different software versions. The software version is displayed on the screen when the monitor is switched on, and is also indicated as a 1-byte code (the $5^{th}$ byte) in the 40-byte ‘header’ which precedes all data output via the serial port. The 1-byte software version codes are shown in Table 4.2.

Table 4.2: Software versions and their Datex Read Interface codes ($r_{\text{dr}_i_{\text{level}}}$).

<table>
<thead>
<tr>
<th>Software version</th>
<th>code</th>
</tr>
</thead>
<tbody>
<tr>
<td>S-STD93</td>
<td>0</td>
</tr>
<tr>
<td>S-STD94, S-ARK94</td>
<td>1</td>
</tr>
<tr>
<td>S-STD95, S-ARK95, S-STD96, S-ARK96</td>
<td>2</td>
</tr>
<tr>
<td>S-ANE97, S-ARK97, S-ICU97</td>
<td>3</td>
</tr>
</tbody>
</table>

4.1.2 Available software

A program for PCs called COLLECT.EXE, which saves data from the Datex AS/3 monitor, is available from Datex. This program is known as the AS/3 PC Data Collection Software. The program collects the data-strings output by the monitor and saves them to the hard disk of the PC either as an ASCII file, a binary file, or in a form compatible with LOTUS 1-2-3. The package consists of three program files as follows.

COLLECT.EXE
COLLECT.CFR (used for storing setup information)
AUTOFILE.CFR (used for writing an automatic date-dependent filename)

The Collect program is also available for the S/5 monitor—see Datex-Ohmeda S/5 Collect Users Reference Manual (February 2001).

4.2 Serial interface


4.2.1 Serial port connector

The monitors have a MALE 9-pin D-type serial port which conforms to the RS-232-E standard. The serial port is located at the back of the monitor. On the S/5 anaesthesia monitor the RS-232 serial data connector is designated as X8 (see: § 2.3.3 in: Datex-Ohmeda Document No. 8001022)—the pin-outs are as shown in Table 4.3.

Caution: The S/5 monitor also has a 9-pin female D-connector (designated as X4) which is a ‘network interface’.

The serial port allows commands to be sent to the monitor, and also allows CTS/RTS flow control (hardware handshaking) via pins 7 and 8 of the serial port.
CHAPTER 4. THE DATEX AS/3 ANAESTHESIA MONITOR

Table 4.3: Datex AS/3, CS/3, S/5 RS-232 serial port (9-pin MALE D-connector).

<table>
<thead>
<tr>
<th>Pin No.</th>
<th>Name</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>RxD</td>
<td>Receives data</td>
</tr>
<tr>
<td>3</td>
<td>TxD</td>
<td>Transmits data (LOW on power-up)</td>
</tr>
<tr>
<td>5</td>
<td>GND</td>
<td>Signal ground</td>
</tr>
<tr>
<td>7</td>
<td>RTS</td>
<td>Set HIGH when powered up</td>
</tr>
<tr>
<td>8</td>
<td>CTS</td>
<td>Can be used to control data flow</td>
</tr>
</tbody>
</table>

4.2.2 Cable connections

The wiring configuration for interfacing the Datex AS/3 monitor to a PC is shown in Figure 4.1.

- **CTS** [NB: not fully checked out for AS/3] Data output from the Datex monitor is usually controlled by influencing the voltage status of the Datex-AS3 CTS line. Data output is enabled only if its CTS is held HIGH (positive). [BUT in my experience data output was only stopped by setting the datex RTS line LOW !!]

However, if it is necessary to use hardware handshaking to control data output, then it is probably best to connect the Datex monitor’s CTS line to the computer’s RTS line, which can then be used to control data output by setting the status of the computer’s RTS line HIGH or LOW as necessary (see Section 5.16 in Nickalls & Ramasubramanian, 1995).

- **RTS** The Datex-AS3’s RTS line is held HIGH on power up. Holding the Datex RTS line LOW will stop all data output until it is pulled HIGH again.

Consequently it is usual to connect this line to the computer’s CTS line, to enable the computer program to control data output from the Datex monitor.

Figure 4.1: Wiring configuration for the Datex AS/3 & CS/3 monitors.

4.2.3 Protocol

The serial protocol is shown in Table 4.4. Note that this protocol is slightly unusual in that it uses an 11-bit character-frame (1 start-bit, 8 data-bits, EVEN parity-bit, 1 stop-bit). Consequently some older software which uses a ten-bit character frame
(e.g. QuickBASIC 4.5, QBASIC 1.1) cannot be used to program the Datex-AS3 serial interface. PowerBASIC 3.5, FirstBASIC (PB1.0) and VisualBASIC can all handle 11-bit character-frames. Note that the recent 3.15 version of KERMIT (1998) also accommodates the 11-bit character frame (see the SET PARITY HARDWARE command), and so can be used to access data from the Datex AS/3 monitor.

Table 4.4: Serial protocol for the Datex AS/3 & CS/3 monitors.

<table>
<thead>
<tr>
<th>Bit rate</th>
<th>19200</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start bits</td>
<td>1</td>
</tr>
<tr>
<td>Data bits</td>
<td>8</td>
</tr>
<tr>
<td>Parity</td>
<td>Even</td>
</tr>
<tr>
<td>Stop bits</td>
<td>1</td>
</tr>
</tbody>
</table>

### 4.3 Transmission Request string

Unlike the earlier Ultima or Capnomac monitors the AS/3 series of monitors (on startup) do not by default output any data via the serial port on startup. In order to get the monitor to output data (digital or waveform data) one has to first send a ‘Transmission Request’ message, in which one specifies, among other things, the required data frequency, say, every 10 seconds. The complete output data-string is 321 bytes, and is very comprehensive. It is well documented in Datex-Ohmeda S/5 Computer Interface: Record Specification, Document No. 8005313-1, October 2003).

The monitor is able to output data in a number of modes; either (a) only the current displayed measurement values; (b) values averaged over the last 10 seconds, (c) values averaged over the last 60 seconds. See the Datex manual for full details (Datex-Ohmeda S/5 Computer Interface Specification, Document No. M1017617-01, March 2004).

#### 4.3.1 Format

Unfortunately the AS/3 & S/5 monitors use a rather complicated and somewhat confusing ‘transmission request’ command (a string of 52 bytes) to instruct the monitor to output data. The frequency of data output (every 10 seconds, 60 seconds etc) is set using bytes 43 and 44. In practice we require data output every 10 seconds, for which is encoded using byte 43 \( \rightarrow 0A\text{hex} \), and byte 44 \( \rightarrow 00\text{hex} \) (see below).

The Transmission Request string which is the one currently used is described below.

For example, in the following PowerBasic code (used in my original DOS program)\(^2\) the Transmission Request string is assembled by the SUB requeststring, which is part of the Datex module. This string is sent only once (by the Main Module) soon after system initialisation; it sends the string and then waits a maximum of 5 seconds for the first incoming data-string before timing-out—note that the LOC() function returns the number of bytes in the ‘receive’ buffer.

```plaintext
... REM now trigger data output (every 10 sec) from Datex AS/3 monitor
```

\(^2\)see Chapter 12 in [http://www.nickalls.org/dick/xenon/ARS5aDOC2014a.pdf](http://www.nickalls.org/dick/xenon/ARS5aDOC2014a.pdf)
CALL RequestString :REM in DatexAS3 module
REM start timer and wait max 5 sec for data to arrive
thistime=timer
DO
  IF TIMER > thistime + 5 THEN
    BEEP
    PRINT " No data --- quitting program"
    SLEEP 1
  END
END IF
REM if data in buffer, then continue
IF LOC(datexAS3comportfilenumber%) > 0 THEN
  PRINT " data output OK"
  SLEEP 1
  EXIT DO
END IF
SLEEP 1
REM print dots ... while waiting
PRINT ".";
LOOP
...

4.3.2 Transmission request command

The structure of the Transmission request command-string used in this project is that of type-2 (see the correspondence at the end of this chapter), and triggers data-outout every 10 seconds. The following few points are relevant here.

- The string starts and ends with a 7Eh byte).
- I have numbered the bytes (decimal) starting with 1 (1–52)
- The byte values are given in Hexadecimal (h) and Decimal (d).
- The bytes are divided up into their functional groups (1, 2 4 bytes etc)

<table>
<thead>
<tr>
<th>Byte</th>
<th>Hex</th>
<th>Decimal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td>value</td>
<td>value</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>7E</td>
<td>126</td>
<td>Start flag</td>
</tr>
<tr>
<td>2</td>
<td>31</td>
<td>49</td>
<td>(start of header)</td>
</tr>
<tr>
<td>3</td>
<td>00</td>
<td>0</td>
<td>Total length = 0031h = 49d bytes (word r_len)</td>
</tr>
<tr>
<td>4</td>
<td>00</td>
<td>0</td>
<td>Reserved, set to zero (byte res1)</td>
</tr>
<tr>
<td>5</td>
<td>00</td>
<td>0</td>
<td>Ignored by monitor, set to zero (byte r_dri_level)</td>
</tr>
<tr>
<td>6</td>
<td>00</td>
<td>0</td>
<td>Reserved = 0000H (byte res2[2])</td>
</tr>
<tr>
<td>7</td>
<td>00</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Transmission time = 0x00000000, ignored by monitor when sending transmission request (dword r_time). However, time can be meaningful in outputted messages, which use the header of the same structure (dword r_time).

Reserved = 00000000H (dword res3)

Main type of record = DRI_MT_PHDB = 0 (r_maintype)

Offset to the first subrecord = 0000H (sr_desc[0].offset)

Type of first subrecord, DRI_PH_XMIT_REQ = 0 (sr_desc[0].sr_type)

Offset to the second subrecord = 0000H, calculated from the beginning of the data area after the header part. Value is not meaningful, since there is only one subrecord in the request (sr_desc[1].offset).

"No more subrecords" (sr_desc[1].sr_type)

sr_desc[2].offset = 0x0000, no meaning since only one subrecord transmitted...

sr_desc[2].sr_type, no meaning

sr_desc[3].offset = 0x0000, no meaning

sr_desc[3].sr_type, no meaning

sr_desc[4].offset = 0x0000, no meaning

sr_desc[4].sr_type, no meaning

sr_desc[5].offset = 0x0000, no meaning

sr_desc[5].sr_type, no meaning

sr_desc[6].offset = 0x0000, no meaning

sr_desc[6].sr_type, no meaning
4.4 **Output data-string format**

The data format for the Datex AS/3 & CS/3 monitors is described in the Datex document *AS/3 & CS/3 Computer Interface Specification, Revision 3.1, 15/5/1997*. This is a 37-page A4 document available from Datex, and covers the software versions listed in Table 4.2.

After the computer sends the above Transmission Request command-string, the Datex AS/3 monitor responds by outputting the following 321-byte string every 10 seconds, which consists of

- (a) a 1-byte ‘start’ flag `<7Eh>`,
- (b) a 40-byte ‘header’,
- (c) a number of so-called ‘sub-records’, and finally
- (d) a checksum (1 byte) followed by a 1-byte ‘stop flag’ `<7Eh>`.

The following few points are relevant.

- The string starts and ends with a `7Eh` byte = 126d.
- I have numbered the bytes (decimal) starting with 1 (1–321). Note that byte-1 is the FIRST byte to be received by the PC.
- The byte values are given in Hexadecimal (h)
• In the following listing the bytes are divided up into their functional groups (1, 2, 3, 4 bytes etc). Note that the order of the bytes within a group is shown in the left-hand column.

• When decoding the groups of bytes, note that UNIX rules apply, and the within-group byte-order needs to be ‘reversed’ (i.e. in order to have the highest byte number to the left-hand side, and lowest byte number to the right-hand side). For example, the four-byte double-word group \( \text{<CBh><CFh><F2h><33h>} \) (bytes 8–11) encodes for the time in seconds since 1970.00 yrs. Reversing the the byte order (i.e. having bit-0 on the right-hand side) gives the double-word \( \text{<33F2CFBCBh>} \) which is 871550923 seconds \( \rightarrow 10087 \) days \( \rightarrow 27 \) July, 1997.

• Each functional grouping (what Datex calles a ‘sub-record’) has a group of status-bytes (usually 4 bytes) and a group of label-bytes (usually two bytes). These status and label bytes are mostly bit-encoded to indicate such things as the source of the particular measurement, or the existence of an error state etc—some of the encodings for the more important parameters are included in this list, but it is not comprehensive just now (see the Datex Record Specification manual for full details).

---

1 <7E> Start Flag

---

START OF HEADER

2, 3 <3E> <01> Total no of bytes in transmission

\(<01h><3eh> = 318d = 318 \text{ bytes (header + data)}\>

Total bytes=321= 1 start + 318 + 1checksum + 1 stop

---

4 <94>

5 <03> Interface version supported by device (0-3)

see version code Table

6 <00>

7 <00>

8-11 <CB> <CF> <F2> <33> Time in secs since 1970.00 yrs

\( = <33><f2><cf><cb>=871550923 \text{ secs} \)

\( = 10087 \text{ days} = 27\text{yrs 7 months 22 days} \)

\( = 27 \text{ July 1997} \)

12-15 <00> <00> <00> <00>

16-17 <00> <00>

18-20 <00> <00> <01> The <1> here is sr_type for output data (1-4, p 10)

21-23 <BD> <BD> <ff> ?? the <ff> indicating no more subrecords??

24-26 <BD> <BD> <BD> ?? why are these fields filled with <BD> ????

27-29 <BD> <BD> <BD> Note <bdh> = 189d

30-32 <BD> <BD> <BD>

33-35 <BD> <BD> <BD>

36-38 <BD> <BD> <BD>

39-41 <BD> <BD> <BD>

-------end of header, always 40 bytes---------
-----start of the data area-----------------------
42-45  <CB>  <CF>  <F2>  <33>  Time in secs since 1/1/1970 = 8.7155092 E08

---------------
ECG subrecord
46-49  <0B>  <3A>  <00>  <00>  ECG group header (status- 4 bytes)
50-51  <74>  <32>  (label - 2 bytes)
52-53  <02>  <80>  HR
54-55  <00>  <81>  st1 (mm/100)
56-57  <05>  <81>  st2
58-59  <08>  <81>  st3
60-61  <01>  <80>  rr (resp rate/min)

---------------
INV Press(1) subrecord
62-65  <1>  <0>  <0>  <0>  Inv Press 1 header (status)
66-67  <1>  <0>  (label)
68-69  <2>  <80>  sys x100
70-71  <2>  <80>  diast x100
72-73  <2>  <80>  mean x100
74-75  <1>  <80>  heart rate/min

---------------
INV Press(2) subrecord
76-79  <1>  <0>  <0>  <0>  status
80-81  <2>  <0>  label
82-83  <2>  <80>  sys x100
84-85  <2>  <80>  diast x100
86-87  <2>  <80>  mean x100
88-89  <1>  <80>  heart rate/min

---------------
INV Press(3) subrecord
90-93  <1>  <0>  <0>  <0>  status
94-95  <B>  <0>  label
96-97  <2>  <80>  sys x100
98-99  <2>  <80>  diast x100
100-101 <2>  <80>  mean x100
102-103 <1>  <80>  heart rate/min

---------------
INV Press(4) subrecord
104-107 <1>  <0>  <0>  <0>  status
108-109 <3>  <0>  label
110-111 <2>  <80>  sys x100
112-113 <2>  <80>  diast x100
114-115 <2>  <80>  mean x100
116-117 <1>  <80>  heart rate/min

---------------
NIBP subrecord
118-121 <3>  <0>  <0>  <0>  status
122-123 <3>  <1>  label (bit-8 --> 1 after 60 secs)
124-125 <1>  <80>  sys x100
126-127 <1>  <80>  diast x100
128-129 <1>  <80>  mean x100
130-131 <1> <80> HR /min

-------------------

Temp (1) subrecord
132-135 <3> <0> <0> <0> status
136-137 <B> <0> label
138-139 <1> <80> deg C x100

-------------------

Temp (2) subrecord
140-143 <3> <0> <0> <0> status
144-145 <C> <0> label
146-147 <1> <80> deg C x100

-------------------

Temp (3) subrecord
148-151 <0> <0> <0> <0> status
152-153 <D> <0> label
154-155 <1> <80> deg C x100

-------------------

Temp (4) subrecord
156-159 <0> <0> <0> <0> status
160-161 <E> <0> label
162-163 <1> <80> deg C x100

-------------------

Saturation (SpO2) subrecord
164-167 <3> <0> <0> <0> status
168-169 <0> <0> label (00=SaO2 01=SvO2 10=error)
170-171 <1> <80> (SAT% * 100)
172-173 <1> <80> HR
174-175 <2> <80> IR-amp (infra red amplitude)
176-177 <1> <80> label for SaO2 = 1 /SvO2 = 2 /SO2= 0 / 3 not used

-------------------

Carbon dioxide (CO2) subrecord
178-181 <47> <0> <0> <0> status
182-183 <0> <0> label (source: 01=CO2 10=ECG)
184-185 <1> <80> ET (% x100)
186-187 <1> <80> FI (% x100)
188-189 <1> <80> RR
190-191 <28> <1D> amb_P (x10 mmHg ambient pressure)

-------------------

Oxygen (O2) subrecord
192-195 <3> <0> <0> <0> status
196-197 <0> <0> label
198-199 <1> <80> ET O2 (% x100)
200-201 <1> <80> FI O2 (% x100)

-------------------

Nitrous Oxide (N2O) subrecord
202-205 <3> <0> <0> <0> status
206-207 <0> <0> label
208-209 <1> <80> ET N2O (% x100)
210-211 <1> <80> FI N2O (% x100)
CHAPTER 4. THE DATEX AS/3 ANAESTHESIA MONITOR

Anaesthetic agent
212-215 <3> <0> <0> status
216-217 <2> <0> label
218-219 <1> <80> ET AA (% x100)
220-221 <1> <80> FI AA (% x100)
222-223 <0> <0> MAC sum (x100)

----------------------

Flow & Volume
224-227 <3> <0> <0> status
228-229 <0> <0> label
230-231 <0> <0> RR (resp rate)
232-233 <1> <80> pPeak x100
234-235 <1> <80> peep x100
236-237 <1> <80> pPlat x100
238-239 <1> <80> TV-insp x10
240-241 <1> <80> TV-exp x10
242-243 <1> <80> compliance x100 cms H2O
244-245 <1> <80> MV exp x100/min

----------------------

Cardiac Output & Wedge press
246-249 <3> <0> <0> status
250-251 <7> <0> label
252-253 <1> <80> CO
254-255 <1> <80> Blood Temp
256-257 <1> <80> Ref
258-259 <1> <80> pcwp

----------------------

Neuro-Muscular J (NMJ)
260-263 <20> <0> <0> status
264-265 <0> <0> label
266-267 <1> <80>
268-269 <1> <80>
270-271 <FF> <8d>

----------------------

ECG (2) (no header)
272-273 <2> <80>
274-275 <1> <80>
276-277 <1> <80>

----------------------

Reserved-1 (8 bytes)
278-285 <0> <0> <0> <D3> <0> <1> <80>

----------------------

Invas Press (5) subrecord
286-289 <0> <0> <0>
290-291 <D> <0>
292-293 <2> <80>
294-295 <2> <80>
296-297 <2> <80>
298-299 <1> <80>

--------
4.5 List of parameter names

This is a provisional list (namelst.tex) of parameter variable and string names (made in November 2001) for use in the Linux Anaesthesia AS/3 project.

Each parameter name has an associated number of Bytes, Unit, Divisor, and Description. The parameters have a fixed byte location (Position) in the output data string. Note that the first location is numbered 1, and hence this numbering corresponds exactly with the numbers used in the SUB Decode (Section ??, page ??).

The parameter names I have suggested in the following table consist mostly of combined upper-case and lower-case characters.

Use of the Divisor (DIV): we first extract the numeric value, and then divide it by the DIV value to obtain the true value. (see example on Table 3.5 on page 21, for invas BP, for which DIV = 100). Use the numeric example given in brackets at the end of each line as a guide to the number of decimal places to use for the final parameter value (always either only the integer or 1 decimal place).

Note (a) that bytes 1–41 are part of the data header, (b) bytes 278–317 are not allocated at present, (c) the FirstWord (bytes 2-3) and LastWord (bytes 318–319) are always ?? the same number. (d) the checksum is an 8-bit unsigned integer of the sum of all bytes including start AND stop flags (need to check this).
Table 4.5: Parameter definitions for Datex AS/3

<table>
<thead>
<tr>
<th>Position</th>
<th>bytes</th>
<th>Name</th>
<th>Unit</th>
<th>Div</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2–3</td>
<td>2</td>
<td>FirstWord</td>
<td></td>
<td></td>
<td>total bytes excluding &lt;start&gt; &amp; &lt;stop&gt; &amp; &lt;checksum&gt; (318)</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>stringNumber</td>
<td></td>
<td></td>
<td>gives the no of strings since start</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>AS3version</td>
<td></td>
<td></td>
<td>software interface version (2)</td>
</tr>
<tr>
<td>42–45</td>
<td>4</td>
<td>TIMEabsolute</td>
<td>secs</td>
<td>1</td>
<td>Time in secs since 1/1/1970 (871550923)</td>
</tr>
<tr>
<td>46–49</td>
<td>4</td>
<td>ECGstatus</td>
<td>—</td>
<td>—</td>
<td>ECG status</td>
</tr>
<tr>
<td>50–51</td>
<td>2</td>
<td>ECGlabel</td>
<td>—</td>
<td>—</td>
<td>ECG label</td>
</tr>
<tr>
<td>52–53</td>
<td>2</td>
<td>HRecg</td>
<td>/min</td>
<td>1</td>
<td>ECG Heart Rate (63)</td>
</tr>
<tr>
<td>54–55</td>
<td>2</td>
<td>ST1</td>
<td>mm</td>
<td>100</td>
<td>ST depression 1 (1.6)</td>
</tr>
<tr>
<td>56–57</td>
<td>2</td>
<td>ST2</td>
<td>mm</td>
<td>100</td>
<td>ST depression 2 (1.6)</td>
</tr>
<tr>
<td>58–59</td>
<td>2</td>
<td>ST3</td>
<td>mm</td>
<td>100</td>
<td>ST depression 3 (1.6)</td>
</tr>
<tr>
<td>60–61</td>
<td>2</td>
<td>RRecg</td>
<td>/min</td>
<td>1</td>
<td>Respiratory rate (15)</td>
</tr>
<tr>
<td>62–65</td>
<td>4</td>
<td>BP1status</td>
<td>—</td>
<td>—</td>
<td>Inv Press 1 status</td>
</tr>
<tr>
<td>66–67</td>
<td>2</td>
<td>BP1label</td>
<td>—</td>
<td>—</td>
<td>Inv Press 1 label</td>
</tr>
<tr>
<td>68–69</td>
<td>2</td>
<td>BP1s</td>
<td>mmHg</td>
<td>100</td>
<td>Inv Press 1 systolic (120)</td>
</tr>
<tr>
<td>70–71</td>
<td>2</td>
<td>BP1d</td>
<td>mmHg</td>
<td>100</td>
<td>Inv Press 1 diastolic (80)</td>
</tr>
<tr>
<td>72–73</td>
<td>2</td>
<td>BP1m</td>
<td>mmHg</td>
<td>100</td>
<td>Inv Press 1 mean (90)</td>
</tr>
<tr>
<td>74–75</td>
<td>2</td>
<td>HRbp1</td>
<td>/min</td>
<td>1</td>
<td>Inv Press 1 heart rate (63)</td>
</tr>
<tr>
<td>76–79</td>
<td>4</td>
<td>BP2status</td>
<td>—</td>
<td>—</td>
<td>Inv Press 2 status</td>
</tr>
<tr>
<td>80–81</td>
<td>2</td>
<td>BP2label</td>
<td>—</td>
<td>—</td>
<td>Inv Press 2 label</td>
</tr>
<tr>
<td>82–83</td>
<td>2</td>
<td>BP2s</td>
<td>mmHg</td>
<td>100</td>
<td>Inv Press 2 systolic (120)</td>
</tr>
<tr>
<td>84–85</td>
<td>2</td>
<td>BP2d</td>
<td>mmHg</td>
<td>100</td>
<td>Inv Press 2 diastolic (80)</td>
</tr>
<tr>
<td>86–87</td>
<td>2</td>
<td>BP2m</td>
<td>mmHg</td>
<td>100</td>
<td>Inv Press 2 mean (90)</td>
</tr>
<tr>
<td>88–89</td>
<td>2</td>
<td>HRbp2</td>
<td>/min</td>
<td>1</td>
<td>Inv Press 2 heart rate (63)</td>
</tr>
<tr>
<td>90–93</td>
<td>4</td>
<td>BP3status</td>
<td>—</td>
<td>—</td>
<td>Inv Press 3 status</td>
</tr>
<tr>
<td>94–95</td>
<td>2</td>
<td>BP3label</td>
<td>—</td>
<td>—</td>
<td>Inv Press 3 label</td>
</tr>
<tr>
<td>96–97</td>
<td>2</td>
<td>BP3s</td>
<td>mmHg</td>
<td>100</td>
<td>Inv Press 3 systolic (120)</td>
</tr>
<tr>
<td>98–99</td>
<td>2</td>
<td>BP3d</td>
<td>mmHg</td>
<td>100</td>
<td>Inv Press 3 diastolic (80)</td>
</tr>
<tr>
<td>100–101</td>
<td>2</td>
<td>BP3m</td>
<td>mmHg</td>
<td>100</td>
<td>Inv Press 3 mean (90)</td>
</tr>
<tr>
<td>102–103</td>
<td>2</td>
<td>HRbp3</td>
<td>/min</td>
<td>1</td>
<td>Inv Press 3 heart rate (63)</td>
</tr>
<tr>
<td>104–107</td>
<td>4</td>
<td>BP4status</td>
<td>—</td>
<td>—</td>
<td>Inv Press 4 status</td>
</tr>
<tr>
<td>108–109</td>
<td>2</td>
<td>BP4label</td>
<td>—</td>
<td>—</td>
<td>Inv Press 4 label</td>
</tr>
<tr>
<td>110–111</td>
<td>2</td>
<td>BP4s</td>
<td>mmHg</td>
<td>100</td>
<td>Inv Press 4 systolic (120)</td>
</tr>
<tr>
<td>112–113</td>
<td>2</td>
<td>BP4d</td>
<td>mmHg</td>
<td>100</td>
<td>Inv Press 4 diastolic (80)</td>
</tr>
<tr>
<td>114–115</td>
<td>2</td>
<td>BP4m</td>
<td>mmHg</td>
<td>100</td>
<td>Inv Press 4 mean (90)</td>
</tr>
<tr>
<td>116–117</td>
<td>2</td>
<td>HRbp4</td>
<td>/min</td>
<td>1</td>
<td>Inv Press 4 heart rate (63)</td>
</tr>
<tr>
<td>Position</td>
<td>bytes</td>
<td>Name</td>
<td>Unit</td>
<td>Div</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------</td>
<td>---------------</td>
<td>--------</td>
<td>-----</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td>118–121</td>
<td>4</td>
<td>NIBPstatus</td>
<td>—</td>
<td>—</td>
<td>NIBP status</td>
</tr>
<tr>
<td>122–125</td>
<td>2</td>
<td>NIBPlabel</td>
<td>—</td>
<td>—</td>
<td>NIBP label</td>
</tr>
<tr>
<td>124–125</td>
<td>2</td>
<td>NIBPs</td>
<td>mmHg</td>
<td>100</td>
<td>NIBP systolic (120)</td>
</tr>
<tr>
<td>126–127</td>
<td>2</td>
<td>NIBPd</td>
<td>mmHg</td>
<td>100</td>
<td>NIBP diastolic (80)</td>
</tr>
<tr>
<td>128–129</td>
<td>2</td>
<td>NIBPm</td>
<td>mmHg</td>
<td>100</td>
<td>NIBP mean (90)</td>
</tr>
<tr>
<td>130–131</td>
<td>2</td>
<td>HRnibp</td>
<td>/min</td>
<td>1</td>
<td>NIBP heart rate (63)</td>
</tr>
<tr>
<td>132–135</td>
<td>4</td>
<td>TEMP1status</td>
<td>—</td>
<td>—</td>
<td>Temp1 status</td>
</tr>
<tr>
<td>136–137</td>
<td>2</td>
<td>TEMP1label</td>
<td>—</td>
<td>—</td>
<td>Temp1 label</td>
</tr>
<tr>
<td>138–139</td>
<td>2</td>
<td>TEMP1</td>
<td>degC</td>
<td>100</td>
<td>Temp1 (37.5)</td>
</tr>
<tr>
<td>140–143</td>
<td>4</td>
<td>TEMP2status</td>
<td>—</td>
<td>—</td>
<td>Temp2 status</td>
</tr>
<tr>
<td>144–145</td>
<td>2</td>
<td>TEMP2label</td>
<td>—</td>
<td>—</td>
<td>Temp2 label</td>
</tr>
<tr>
<td>146–147</td>
<td>2</td>
<td>TEMP2</td>
<td>degC</td>
<td>100</td>
<td>Temp2 (37.5)</td>
</tr>
<tr>
<td>148–151</td>
<td>4</td>
<td>TEMP3status</td>
<td>—</td>
<td>—</td>
<td>Temp3 status</td>
</tr>
<tr>
<td>152–153</td>
<td>2</td>
<td>TEMP3label</td>
<td>—</td>
<td>—</td>
<td>Temp3 label</td>
</tr>
<tr>
<td>154–155</td>
<td>2</td>
<td>TEMP3</td>
<td>degC</td>
<td>100</td>
<td>Temp3 (37.5)</td>
</tr>
<tr>
<td>156–159</td>
<td>4</td>
<td>TEMP4status</td>
<td>—</td>
<td>—</td>
<td>Temp4 status</td>
</tr>
<tr>
<td>160–161</td>
<td>2</td>
<td>TEMP4label</td>
<td>—</td>
<td>—</td>
<td>Temp4 label</td>
</tr>
<tr>
<td>162–163</td>
<td>2</td>
<td>TEMP4</td>
<td>degC</td>
<td>100</td>
<td>Temp4 (37.5)</td>
</tr>
<tr>
<td>164–167</td>
<td>4</td>
<td>SATstatus</td>
<td>—</td>
<td>—</td>
<td>Saturation status</td>
</tr>
<tr>
<td>168–169</td>
<td>2</td>
<td>SATlabel</td>
<td>—</td>
<td>—</td>
<td>Saturation label</td>
</tr>
<tr>
<td>170–171</td>
<td>2</td>
<td>SAT</td>
<td>%</td>
<td>100</td>
<td>Saturation (95)</td>
</tr>
<tr>
<td>172–173</td>
<td>2</td>
<td>HRsat</td>
<td>/min</td>
<td>1</td>
<td>Saturation heart rate</td>
</tr>
<tr>
<td>174–175</td>
<td>2</td>
<td>SATamp</td>
<td>%</td>
<td>1</td>
<td>Plethysmograph IR-amplitude</td>
</tr>
<tr>
<td>176–177</td>
<td>2</td>
<td>SATloc</td>
<td>—</td>
<td>100</td>
<td>Sat probe location (ven/art)</td>
</tr>
<tr>
<td>178–181</td>
<td>4</td>
<td>CO2status</td>
<td>—</td>
<td>—</td>
<td>CO2 status</td>
</tr>
<tr>
<td>182–183</td>
<td>2</td>
<td>CO2label</td>
<td>—</td>
<td>—</td>
<td>CO2 label</td>
</tr>
<tr>
<td>184–185</td>
<td>2</td>
<td>ETCO2</td>
<td>%</td>
<td>100</td>
<td>End Tidal CO2 conc (4.6)</td>
</tr>
<tr>
<td>186–187</td>
<td>2</td>
<td>FICO2</td>
<td>%</td>
<td>100</td>
<td>Frac Insp CO2 conc (1.3)</td>
</tr>
<tr>
<td>188–189</td>
<td>2</td>
<td>RRCO2</td>
<td>/min</td>
<td>1</td>
<td>Resp Rate CO2 (12)</td>
</tr>
<tr>
<td>190–191</td>
<td>2</td>
<td>AMBPCO2</td>
<td>mmHg</td>
<td>10</td>
<td>Ambient pressure</td>
</tr>
<tr>
<td>192–195</td>
<td>4</td>
<td>O2status</td>
<td>—</td>
<td>—</td>
<td>Oxygen status</td>
</tr>
<tr>
<td>196–197</td>
<td>2</td>
<td>O2label</td>
<td>—</td>
<td>—</td>
<td>Oxygen label</td>
</tr>
<tr>
<td>198–199</td>
<td>2</td>
<td>ETO2</td>
<td>%</td>
<td>100</td>
<td>End Tidal oxygen conc (37)</td>
</tr>
<tr>
<td>200–201</td>
<td>2</td>
<td>FIO2</td>
<td>%</td>
<td>100</td>
<td>Frac Insp oxygen conc (28)</td>
</tr>
<tr>
<td>202–205</td>
<td>4</td>
<td>N2Ostatus</td>
<td>—</td>
<td>—</td>
<td>N2O status</td>
</tr>
<tr>
<td>206–207</td>
<td>2</td>
<td>N2Olabel</td>
<td>—</td>
<td>—</td>
<td>N2O label</td>
</tr>
<tr>
<td>208–209</td>
<td>2</td>
<td>ETN2O</td>
<td>%</td>
<td>100</td>
<td>End Tidal N2O conc (66)</td>
</tr>
<tr>
<td>210–211</td>
<td>2</td>
<td>FIN2O</td>
<td>%</td>
<td>100</td>
<td>Frac Insp N2O conc (63)</td>
</tr>
<tr>
<td>212–215</td>
<td>4</td>
<td>AAsstatus</td>
<td>—</td>
<td>—</td>
<td>Anaesthetic agent status</td>
</tr>
<tr>
<td>216–217</td>
<td>2</td>
<td>AAlabel</td>
<td>—</td>
<td>—</td>
<td>Anaesthetic agent label</td>
</tr>
<tr>
<td>218–219</td>
<td>2</td>
<td>ETAA</td>
<td>%</td>
<td>100</td>
<td>End Tidal Anaes agent (1.65)</td>
</tr>
<tr>
<td>220–221</td>
<td>2</td>
<td>FIAA</td>
<td>%</td>
<td>100</td>
<td>Frac Insp Anaes agent (1.65)</td>
</tr>
<tr>
<td>222–223</td>
<td>2</td>
<td>MAC</td>
<td>%</td>
<td>100</td>
<td>MAC total (1.25)</td>
</tr>
</tbody>
</table>
### Table 4.7: Parameter definitions for Datex AS/3

<table>
<thead>
<tr>
<th>Position</th>
<th>bytes</th>
<th>Name</th>
<th>Unit</th>
<th>Div</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>224–227</td>
<td>4</td>
<td>FLOWVOLstatus</td>
<td>—</td>
<td>—</td>
<td>Flow/volume status</td>
</tr>
<tr>
<td>228–229</td>
<td>2</td>
<td>FLOWVOLlabel</td>
<td>—</td>
<td>—</td>
<td>Flow/volume label</td>
</tr>
<tr>
<td>230–231</td>
<td>2</td>
<td>RR</td>
<td>/min</td>
<td>1</td>
<td>Resp rate (12)</td>
</tr>
<tr>
<td>232–233</td>
<td>2</td>
<td>Ppeak</td>
<td>cmH2O</td>
<td>100</td>
<td>Peak Pressure (30)</td>
</tr>
<tr>
<td>234–235</td>
<td>2</td>
<td>PEEP</td>
<td>cmH2O</td>
<td>100</td>
<td>PEEP (5)</td>
</tr>
<tr>
<td>236–237</td>
<td>2</td>
<td>Plat</td>
<td>cmH2O</td>
<td>100</td>
<td>Plateau pressure (12)</td>
</tr>
<tr>
<td>238–239</td>
<td>2</td>
<td>TVinsp</td>
<td>ml</td>
<td>10</td>
<td>Inspiratory Tidal volume (653)</td>
</tr>
<tr>
<td>240–241</td>
<td>2</td>
<td>TVexp</td>
<td>ml</td>
<td>10</td>
<td>Expiratory Tidal volume (653)</td>
</tr>
<tr>
<td>242–243</td>
<td>2</td>
<td>compliance</td>
<td>ml/cmH2O</td>
<td>100</td>
<td>compliance (23)</td>
</tr>
<tr>
<td>244–245</td>
<td>2</td>
<td>MVexp</td>
<td>ml</td>
<td>100</td>
<td>Inspiratory Tidal volume (653)</td>
</tr>
<tr>
<td>246–249</td>
<td>4</td>
<td>COstatus</td>
<td>—</td>
<td>—</td>
<td>Cardiac output status</td>
</tr>
<tr>
<td>250–251</td>
<td>4</td>
<td>COlabel</td>
<td>—</td>
<td>—</td>
<td>Cardiac output label</td>
</tr>
<tr>
<td>252–253</td>
<td>2</td>
<td>CO</td>
<td>ml/min</td>
<td>1</td>
<td>cardiac output (5000)</td>
</tr>
<tr>
<td>254–255</td>
<td>2</td>
<td>BLOODtemp</td>
<td>degC</td>
<td>100</td>
<td>Blood temperature (37.4)</td>
</tr>
<tr>
<td>256–257</td>
<td>2</td>
<td>EJfrac</td>
<td>%</td>
<td>1</td>
<td>Right heart ejection fraction (54)</td>
</tr>
<tr>
<td>258–259</td>
<td>2</td>
<td>pcwp</td>
<td>mmHg</td>
<td>100</td>
<td>Pulm cap wedge pressure (12)</td>
</tr>
<tr>
<td>260–263</td>
<td>4</td>
<td>NMJstatus</td>
<td>—</td>
<td>—</td>
<td>Neuromuscular J status</td>
</tr>
<tr>
<td>264–265</td>
<td>4</td>
<td>NMJlabel</td>
<td>—</td>
<td>—</td>
<td>Neuromuscular J label</td>
</tr>
<tr>
<td>266–267</td>
<td>2</td>
<td>NMJt1</td>
<td>%</td>
<td>10</td>
<td>Train of 4 T1 (34)</td>
</tr>
<tr>
<td>268–269</td>
<td>2</td>
<td>NMJratio</td>
<td>%</td>
<td>10</td>
<td>Train of 4 ratio (34)</td>
</tr>
<tr>
<td>270–271</td>
<td>2</td>
<td>NMJptc</td>
<td>—</td>
<td>—</td>
<td>Post tetanic count (bit encoded)</td>
</tr>
<tr>
<td>272–273</td>
<td>2</td>
<td>HRecg2</td>
<td>/min</td>
<td>1</td>
<td>ECG Heart Rate (63)</td>
</tr>
<tr>
<td>274–275</td>
<td>2</td>
<td>HRmax2</td>
<td>/min</td>
<td>1</td>
<td>ECG Heart Rate (max) (63)</td>
</tr>
<tr>
<td>276–277</td>
<td>2</td>
<td>HRmin2</td>
<td>/min</td>
<td>1</td>
<td>ECG Heart Rate (min) (63)</td>
</tr>
<tr>
<td>318–319</td>
<td>4</td>
<td>LastWord</td>
<td>—</td>
<td>—</td>
<td>total bytes excluding &lt;start&gt; &amp; &lt;stop&gt; &amp; &lt;checksum&gt; (318)</td>
</tr>
<tr>
<td>320</td>
<td>1</td>
<td>&lt;checksum&gt;</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
</tbody>
</table>

#### 4.6 Example of data output

The following Datex AS/3, S/5 output data-string was received during an operation and saved in the following D-data format (decimal) of 18 lines of decimal codes, each with a leading 'AS3nn' header. For details of the D-data format see Chapter 15 (page 219).

```
AS301,126,062,001,111,005,000,000,166,052,241,058,000,000,000,000,000,000,000
AS302,000,001,000,074,255,097,220,044,000,000,044,000,000,189,189,032
AS303,000,189,189,032,000,166,052,241,058,019,048,000,000,000,034,067,000,021
AS304,000,001,128,001,128,001,128,003,000,000,000,001,000,062,058,231,028,049
```

...
For example, one of the Datex AS/3 invasive blood pressure ‘sub-records’ is encoded in
bytes 62–75, as shown in the following Table.

Table 4.8: Decoding part of the invasive blood pressure 1 record (bytes 62–75).

The systolic, diastolic and mean blood pressure values ×100 are encoded as a
sequence of three pairs of Hex words (ie six bytes) in the positions 68–73 (see
Table 4.5). The following table shows the decimal and hex forms of each byte,
and the hex and decimal values of the respective words (1 word consists of two
bytes). Note the ordering of the bytes when combining them into words. Note
also that the decimal value of each word has to be divided by 100 to obtain the
physiological value, and in this particular case the decoded values are: systolic
BP 149.1, diastolic BP 73.99, mean BP 105.45. In practice we would only pass
on the integer values for blood pressure.

<table>
<thead>
<tr>
<th>Byte number</th>
<th>Dec value</th>
<th>Hex value</th>
<th>Hex word</th>
<th>mean</th>
<th>diastolic</th>
<th>systolic</th>
</tr>
</thead>
<tbody>
<tr>
<td>68-73</td>
<td>0041 049</td>
<td>0028 231</td>
<td>0058 062</td>
<td>73.70</td>
<td>71.049</td>
<td>058.062</td>
</tr>
</tbody>
</table>

The following is the same data but placed in byte order (1–321), together with the
Dec and Hex equivalent.

<table>
<thead>
<tr>
<th>byte,Hex,Dec</th>
</tr>
</thead>
<tbody>
<tr>
<td>001,7E,126</td>
</tr>
<tr>
<td>002,3E,062</td>
</tr>
<tr>
<td>003,01,001</td>
</tr>
<tr>
<td>004,6F,111</td>
</tr>
<tr>
<td>005,05,005</td>
</tr>
<tr>
<td>006,00,000</td>
</tr>
<tr>
<td>007,00,000</td>
</tr>
<tr>
<td>008,66,166</td>
</tr>
<tr>
<td>009,34,052</td>
</tr>
<tr>
<td>010,1F,241</td>
</tr>
<tr>
<td>011,3A,258</td>
</tr>
<tr>
<td>012,00,000</td>
</tr>
<tr>
<td>013,00,000</td>
</tr>
<tr>
<td>014,00,000</td>
</tr>
<tr>
<td>015,00,000</td>
</tr>
<tr>
<td>016,00,000</td>
</tr>
<tr>
<td>017,00,000</td>
</tr>
<tr>
<td>018,00,000</td>
</tr>
<tr>
<td>Hexadecimal Code</td>
</tr>
<tr>
<td>------------------</td>
</tr>
<tr>
<td>019</td>
</tr>
<tr>
<td>020</td>
</tr>
<tr>
<td>021</td>
</tr>
<tr>
<td>022</td>
</tr>
<tr>
<td>023</td>
</tr>
<tr>
<td>024</td>
</tr>
<tr>
<td>025</td>
</tr>
<tr>
<td>026</td>
</tr>
<tr>
<td>027</td>
</tr>
<tr>
<td>028</td>
</tr>
<tr>
<td>029</td>
</tr>
<tr>
<td>030</td>
</tr>
<tr>
<td>031</td>
</tr>
<tr>
<td>032</td>
</tr>
<tr>
<td>033</td>
</tr>
<tr>
<td>034</td>
</tr>
<tr>
<td>035</td>
</tr>
<tr>
<td>036</td>
</tr>
<tr>
<td>037</td>
</tr>
<tr>
<td>038</td>
</tr>
<tr>
<td>039</td>
</tr>
<tr>
<td>040</td>
</tr>
<tr>
<td>041</td>
</tr>
<tr>
<td>042</td>
</tr>
<tr>
<td>043</td>
</tr>
<tr>
<td>044</td>
</tr>
<tr>
<td>045</td>
</tr>
<tr>
<td>046</td>
</tr>
<tr>
<td>047</td>
</tr>
<tr>
<td>048</td>
</tr>
<tr>
<td>049</td>
</tr>
<tr>
<td>050</td>
</tr>
<tr>
<td>051</td>
</tr>
<tr>
<td>052</td>
</tr>
<tr>
<td>053</td>
</tr>
<tr>
<td>054</td>
</tr>
<tr>
<td>055</td>
</tr>
<tr>
<td>056</td>
</tr>
<tr>
<td>057</td>
</tr>
<tr>
<td>058</td>
</tr>
<tr>
<td>059</td>
</tr>
<tr>
<td>060</td>
</tr>
<tr>
<td>061</td>
</tr>
<tr>
<td>062</td>
</tr>
<tr>
<td>063</td>
</tr>
<tr>
<td>064</td>
</tr>
<tr>
<td>065</td>
</tr>
<tr>
<td>066</td>
</tr>
<tr>
<td>067</td>
</tr>
<tr>
<td>068</td>
</tr>
</tbody>
</table>
069,3A,058
070,E7,231
071,1C,028
072,31,049
073,29,041
074,43,067
075,00,000
076,03,003
077,00,000
078,00,000
079,00,000
080,02,002
081,00,000
082,F7,247
083,08,008
084,F4,244
085,05,005
086,2C,044
087,07,007
088,43,067
089,00,000
090,00,000
091,00,000
092,00,000
093,00,000
094,0B,011
095,00,000
096,02,002
097,80,128
098,02,002
099,80,128
100,02,002
101,80,128
102,01,001
103,80,128
104,00,000
105,00,000
106,00,000
107,00,000
108,03,003
109,00,000
110,02,002
111,80,128
112,02,002
113,80,128
114,02,002
115,80,128
116,01,001
117,80,128
118,03,003
119,00,000
120,00,000
121,00,000
122,03,003
123,01,001
124,01,001
125,80,128
126,01,001
127,80,128
128,01,001
129,80,128
130,01,001
131,80,128
132,03,003
133,00,000
134,00,000
135,00,000
136,0B,011
137,00,000
138,D2,210
139,0D,013
140,03,003
141,00,000
142,00,000
143,00,000
144,0C,012
145,00,000
146,04,004
147,80,128
148,00,000
149,00,000
150,00,000
151,00,000
152,0D,013
153,00,000
154,01,001
155,80,128
156,00,000
157,00,000
158,00,000
159,00,000
160,0E,014
161,00,000
162,01,001
163,80,128
164,03,003
165,00,000
166,00,000
167,00,000
168,00,000
169,00,000
170.DE,222
171.26,038
172.44,068
173.00,000
174.6C,108
175.00,000
176.01,001
177.80,128
178.03,003
179.00,000
180.00,000
181.00,000
182.09,009
183.00,000
184.8A,138
185.01,001
186.00,000
187.00,000
188.0C,012
189.00,000
190.66,102
191.1D,029
192.03,003
193.00,000
194.00,000
195.00,000
196.00,000
197.00,000
198.71,113
199.0E,014
200.A5,165
201.0F,015
202.03,003
203.00,000
204.00,000
205.00,000
206.00,000
207.00,000
208.07,007
209.17,023
210.F1,241
211.16,022
212.03,003
213.00,000
214.00,000
215.00,000
216.04,004
217.00,000
218.00,000
<table>
<thead>
<tr>
<th>Number</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>219</td>
<td>00,000</td>
</tr>
<tr>
<td>220</td>
<td>00,000</td>
</tr>
<tr>
<td>221</td>
<td>00,000</td>
</tr>
<tr>
<td>222</td>
<td>3A,058</td>
</tr>
<tr>
<td>223</td>
<td>00,000</td>
</tr>
<tr>
<td>224</td>
<td>03,003</td>
</tr>
<tr>
<td>225</td>
<td>00,000</td>
</tr>
<tr>
<td>226</td>
<td>00,000</td>
</tr>
<tr>
<td>227</td>
<td>00,000</td>
</tr>
<tr>
<td>228</td>
<td>00,000</td>
</tr>
<tr>
<td>229</td>
<td>00,000</td>
</tr>
<tr>
<td>230</td>
<td>0C,012</td>
</tr>
<tr>
<td>231</td>
<td>00,000</td>
</tr>
<tr>
<td>232</td>
<td>0A,010</td>
</tr>
<tr>
<td>233</td>
<td>0F,015</td>
</tr>
<tr>
<td>234</td>
<td>08,008</td>
</tr>
<tr>
<td>235</td>
<td>02,002</td>
</tr>
<tr>
<td>236</td>
<td>CO,192</td>
</tr>
<tr>
<td>237</td>
<td>0D,013</td>
</tr>
<tr>
<td>238</td>
<td>82,130</td>
</tr>
<tr>
<td>239</td>
<td>16,022</td>
</tr>
<tr>
<td>240</td>
<td>E5,229</td>
</tr>
<tr>
<td>241</td>
<td>14,020</td>
</tr>
<tr>
<td>242</td>
<td>F4,244</td>
</tr>
<tr>
<td>243</td>
<td>06,006</td>
</tr>
<tr>
<td>244</td>
<td>7E,126</td>
</tr>
<tr>
<td>245</td>
<td>02,002</td>
</tr>
<tr>
<td>246</td>
<td>00,000</td>
</tr>
<tr>
<td>247</td>
<td>00,000</td>
</tr>
<tr>
<td>248</td>
<td>00,000</td>
</tr>
<tr>
<td>249</td>
<td>00,000</td>
</tr>
<tr>
<td>250</td>
<td>07,007</td>
</tr>
<tr>
<td>251</td>
<td>00,000</td>
</tr>
<tr>
<td>252</td>
<td>01,001</td>
</tr>
<tr>
<td>253</td>
<td>80,128</td>
</tr>
<tr>
<td>254</td>
<td>01,001</td>
</tr>
<tr>
<td>255</td>
<td>80,128</td>
</tr>
<tr>
<td>256</td>
<td>01,001</td>
</tr>
<tr>
<td>257</td>
<td>80,128</td>
</tr>
<tr>
<td>258</td>
<td>01,001</td>
</tr>
<tr>
<td>259</td>
<td>80,128</td>
</tr>
<tr>
<td>260</td>
<td>20,032</td>
</tr>
<tr>
<td>261</td>
<td>00,000</td>
</tr>
<tr>
<td>262</td>
<td>00,000</td>
</tr>
<tr>
<td>263</td>
<td>00,000</td>
</tr>
<tr>
<td>264</td>
<td>00,000</td>
</tr>
<tr>
<td>265</td>
<td>00,000</td>
</tr>
<tr>
<td>266</td>
<td>01,001</td>
</tr>
<tr>
<td>267</td>
<td>80,128</td>
</tr>
<tr>
<td>268</td>
<td>01,001</td>
</tr>
</tbody>
</table>
269, 80, 128
270, FF, 255
271, 8D, 141
272, 01, 001
273, 80, 128
274, 43, 067
275, 00, 000
276, 42, 066
277, 00, 000
278, 00, 000
279, 00, 000
280, 00, 000
281, 00, 000
282, BD, 189
283, BD, 189
284, 01, 001
285, 80, 128
286, 00, 000
287, 00, 000
288, 00, 000
289, 00, 000
290, 0D, 013
291, 00, 000
292, 02, 002
293, 80, 128
294, 02, 002
295, 80, 128
296, 02, 002
297, 80, 128
298, 01, 001
299, 80, 128
300, 00, 000
301, 00, 000
302, 00, 000
303, 00, 000
304, 0E, 014
305, 00, 000
306, 02, 002
307, 80, 128
308, 02, 002
309, 80, 128
310, 02, 002
311, 80, 128
312, 01, 001
313, 80, 128
314, 00, 000
315, 00, 000
316, 00, 000
317, 40, 064
318, 51, 081
4.7 Correspondence

The subrecord types are intended to be used in the sr_type field of the sr_desc -struct:s (see section 3.2, page 8 of the specification) and 0 (=DRI_PH_XMIT_REQ) is the correct value for that field. However, the phdb_rcrd_type field in the data structure "struct phdb_req" is used for a different purpose, though the used enumeration is the same.

The phdb_rcrd_type field indicates what kind of physiological data you are requesting, for example:

\[
\begin{align*}
\text{sr_type} = 0, \text{phdb_rcrd_type} = 1 & \quad \Rightarrow \text{Send current values of the physiological database.} \\
\text{sr_type} = 0, \text{phdb_rcrd_type} = 2 & \quad \Rightarrow \text{Send 10 s trended values} \\
\text{sr_type} = 0, \text{phdb_rcrd_type} = 3 & \quad \Rightarrow \text{Send 60 s trended values} \\
\text{sr_type} = 0, \text{phdb_rcrd_type} = 4 & \quad \Rightarrow \text{Send auxiliary phys. information values 1, 2, 3 and 4 for field sr_type are reserved for output values, as you suggested.}
\end{align*}
\]

So DRI_PH_DISPL = 1, DRI_PH_10S_TREND = 2, DRI_PH_60S_TREND = 3 and DRI_PH_AUX_INFO = 4. The values correspond to subrecord type listed on page 10 of the specification, although the constant names are not explicitly defined in the table.

In addition, the texts in the "Value" field of the table on page 11 of the specification (related to tx_interval) are incorrect: Instead of texts "Any positive value together with subrecord type ..." the texts should be "Any positive value together with physiological record type ..." referencing to field phdb_rcrd_type of struct phdb_req rather than to the sr_type field of struct sr_desc.

... the tx_interval field specifies the transmission interval for the physiological data records, the type of which is specified by the field phdb_rcrd_type ("... together with subrecord type ..."). For 10s and 60s trends the transmission interval is, however, always fixed (10s and 60s). In addition, the special values -1 and 0 have special side effects as documented in the table on page 11 of the specification.
Chapter 5

Interfacing the serial port—using *Perl*

5.1 Introduction

Currently using the Perl programs `as3sim.pl` and `dn-getfile2.pl`. Both in the dir
`~/aHOUSE/perl/serial-port/serial-port-code/testing/`

5.2 Device::SerialPort.pm

This is a Perl program which allows control of the serial port in Linux. I originally used
version 1.002_000, and have needed to modify it by adding a new CTS subroutine in
order to allow hardware handshaking control via the CTS line. This was done simply by
copying and modifying the existing sub `rts_active` subroutine.

```perl
sub cts_active { ## RWDN Jan 2 / 2006
    return unless (@_ == 2);
    my $self = shift;
    ### return unless ($self->can_cts());
    my $on = yes_true( shift );
    # returns ioctl result
    my $value=$IOCTL_VALUE_CTS;
    my $rc=$self->ioctl($on ? 'TIOCMBIS' : 'TIOCMBIC', $value);
    #my $rc=ioctl($self->{HANDLE}, $on ? $bitset : $bitclear, $rtsout);
    warn "cts_active($on) ioctl: $!
    return $rc;
}
```

The current version is `Device-SerialPort-1.04.tar.gz` available from CPAN.
5.3 Sending program (as3sim.pl)

```perl
#!/ perl
##----------------
# sends data out
##----------------
# AS3sim.pl sends data (from dxdemo3c.pl)
# RWD Nickalls Nov 27, 2005
use Device::SerialPort qw(:STAT); # for MS_RTS_ON functions etc
#use POSIX;
use strict;
use warnings;
use Fatal;
use Carp;
use IO::Handle; ## for autoflush() page 224-226
## use prompt module
## use commandline stuff
##-----------------------------
my $COM1 = "/dev/ttyS0";
my $ob = Device::SerialPort->new ($COM1) || croak "Can’t open COM1: $!";
##-------------------
$ob->error_msg(1); # use built-in error messages
$ob->user_msg(1);
#---------------
## setup the COM port
$ob->baudrate(19200) || croak "fail setting baudrate"; ## 19200
$ob->parity("none") || croak "fail setting parity";
$ob->databits(8) || croak "fail setting databits";
$ob->stopbits(1) || croak "fail setting stopbits";
$ob->handshake("none") || croak "fail setting handshake";
$ob->write_settings || croak "no settings";
##-------------------
my $pass;
# use a while() loop to send output data via the serial port

#---------test pulses--
## works OK
# print "testing RTS on/off\n";
# $ob->pulse_rts_on(1000); # 100 ms
# $ob->pulse_rts_off(1000);

#----------------
#if (MS_RTS_ON() == 1){print "RTS-ON\n"}
# else {print "RTS-OFF\n"};

#if (MS_CTS_ON() ==0){print "CTS-ON\n"}
# else {print "CTS-OFF\n"};
```
my $rtsval=0;
$rtsval = MS_RTS_ON();
print "RTSval = ", $rtsval, "\n"

my $ctsval=0;
$ctsval = MS_CTS_ON();
print "CTSval = ", $ctsval, "\n"

my $ringval=0;
$ringval = MS_RING_ON();
print "RIval = ", $ringval, "\n"

#$ob->dtr_active(\'F\'); # 0=red, 1=green OK
#$ob->rts_active(0); # 0=red, 1=green OK

sleep 2;
#---------------
#
#----------------------
## send the file
sendfile();
goto LASTLINE;
#-----------------------

my $crlf="\r\n";
my $outstring1="abcdefg12345\r\n";
my $outstring2="***123***\r\n";
## write the strings to the port

while (1) {
    print $outstring1;
    $pass=$ob->write($outstring1);
sleep 3;
    print $outstring2;
    $pass=$ob->write($outstring2);
sleep 3
}
LASTLINE:
close ; # close any open files
$ob->close || croak "can't close SERIAL PORT";
undef $ob; ## returns memory back to Perl

###---SUB------------------------
## to send a file line by line

sub sendfile{
    ## works OK
    ## always send EOF character to signify the end
my $ifile = './drugs.txt';
local *outfile; ## make it local if in SUB (best practices p=?)

if (-e $ifile) {
    open (*outfile, '<', $ifile)||croak "ERROR: can't open file $ifile\n";
}

## now read each line in the file, and place parameters into an array
print "...reading the fields file < $ifile > line-by-line\n";
my $dataline;
my $outstring;
my $Len; # length of string
my $total_len=0;

LINE:
while (<outfile>){
    # next LINE if /^#/; #skip # comments
    # next LINE if /^%/; #skip % comments
    # next LINE if /^$/; #skip blank lines
    #--------
    # grab the whole line as a string
    $dataline = $_;
    $outstring = $dataline;
    # determine the Byte size of the file
    $Len=length $outstring; $total_len=$total_len + $Len;
    ##chomp($dataline); # remove the line-ending
    print $outstring;
    ## need a small delay to work properly - why exactly
    for (my $j=1;$j<15000;$j++){}; ## seems to be OK
};

## now send EOF character ASCII(26) = ^Z
my $EOF=chr 26;
$pass=$ob->write($EOF);
for (my $j=1;$j<15000;$j++){}; ## seems to be OK

print "\n----end of file-----\n";
print "total length of file = ", $total_len, "\n";
print "waiting 5 secs before closing the file\n";
sleep 5; # ? include slight pause here before closing the file
close (*outfile); # need to keep the *

5.4 Receiving program (dn-getfile2.pl)

#! perl
##--------------
# receives data file
##---------------
# dn-getfile.pl (from dxdemo3c.pl)
## (receives a file & prints to the log file)

# RWD Nickalls Dec 31, 2005

```perl
use Device::SerialPort qw(:STAT);
use strict;
use warnings;
use Fatal;
use Carp;
use IO::Handle; ## for autoflush() page 224-226

my $pass; ## used when writing output to the port ?

my $COM2 = "/dev/ttyS1";
my $ob = Device::SerialPort->new ($COM2) || croak "Can't open COM2: $!";

open my $LOG, ">", "logfile.log" ||croak "can't open logfile file \n";

$LOG->autoflush(); # to the log file
STDOUT->autoflush(); # to the screen
$LOG->prompt(); # see book p 224-226 for better autoflush using IO::Handle module

my $timenow=localtime();
print {$LOG} "the time is:- ", $timenow, "\n";

$ob->error_msg(1); # use built-in error messages
$ob->user_msg(1);

my $dump;
my $portbuffer="";
my $ld;
my $lb;

$ob->baudrate(19200) || croak "fail setting baudrate"; # 19200
$ob->parity("none") || croak "fail setting parity";
$ob->databits(8) || croak "fail setting databits";
$ob->stopbits(1) || croak "fail setting stopbits";
$ob->handshake("none") || croak "fail setting handshake";
$ob->write_settings || croak "no settings";

$ob->lookclear; ## flush buffers

while (($portbuffer=$ob->input) ne "") {  
```
$dump=$dump.$portbuffer;
$Lb=length $portbuffer;
$Ld=length $dump;
print {$LOG} "UART buffer length = ", $Lb," ", "software-buffer length = ", $Ld," ");

JUMP:

print {$LOG} "\n----flush done-- \n";
print {$LOG} "\n===starting collecting data====\n";

#---------------------------
my $EOF=chr 26; # EOF character
my $Leof = -1;
my $Lcr; # char length to the CR
my $buffer=""; ## the string buffer
my $data="";
my $sumpb=0;
## use a while{} loop to read the input data from the serial port
##my $crlf="\r\n";
my $lf="\n";

my $j=0;
INPUT:
while (1) {
    print "waiting for data.....<CTRL-C> to quit\n"
    print " total chars = ",$sumpb, "\n"
    while (($portbuffer=$ob->input) ne ") {
        $buffer.=$portbuffer; # ie $buffer=$buffer.$portbuffer;
        $Lcr= index ($buffer, $lf); ## length to next LF
        $Leof= index ($buffer, $EOF); ## detect EOF character
        $sumpb=$sumpb+ (length $portbuffer);
        if ($Lcr > -1) {
            # detects LF character and prints line
            $data= substr($buffer, 0, $Lcr);
            print {$LOG} $data,"\n"
            $buffer = substr($buffer, $Lcr + 1 ); ## +1 remove the LF as well as CR
            # print {$LOG} "remaining buffer = ", $buffer,"\n"
            # print {$LOG} "total portbuffer chars = ", $sumpb,"\n"
            # print {$LOG} "-------------\n"
        }
        elsif ($Leof > -1){
            ## detects EOF char and prints out last line
            $data= substr($buffer, 0, $Leof);
            print {$LOG} $data,"\n"
            # print {$LOG} "total portbuffer chars = ", $sumpb,"\n"
            # print {$LOG} "-----eof--------\n"
            $pass=$ob->write("thank you"); # works OK
            last INPUT;
        }
        else {## no LF or EOF found
            next;## skip the printing to the file
            ## use this for diagnostics
        }
    }
print {$LOG} "-----NO LF, NO EOF ------\n";
print {$LOG} "buffer = ", $buffer,"\n";
print {$LOG} "portbuffer = ", $portbuffer,"\n";
print {$LOG} "Lcr = ",$Lcr,"\n";
print {$LOG} "Leof = ",$Leof,"\n";
print {$LOG} "---------------\n";
}; ## end of while2
}; ## end of while1

##-----------
close ($LOG);
## now close the serial port
# $ob->close ||croak "failed to close";
undef $ob; ## frees memory back to Perl (but no error message)
#-------------end ----------------
Chapter 6

Age corrected MAC

6.1 Introduction

I first implemented the real-time age-corrected MAC output on the anaesthesia worksta-
tion towards the end of 1996, quite soon after reading Mapleson’s MAC paper (Mapleson
1996). The workstation program at that time was an MS-DOS application (written in
QuickBasic 4.5) and already running in the thoracic theatre at the City Hospital.

In practice this application was greatly facilitated by the excellent serial-port data
stream output by the Datex Cardiocap and Capnomac Ultima series of anaesthesia
monitors we then used (detailed in: Nickalls and Ramasubramanian 1995), since the data

Figure 6.1: Screenshot (November 1997) of the MS-DOS anaesthe-
sia workstation program (version D2c), showing the age-corrected
MAC (“bigMAC”) value in a red-alert state (only 0.74) on the lower
RHS of the screen. Other ‘red-alert’ states also indicated are for Bp
(blood pressure—too low), and alarm sound OFF.
included agent name and inspired and expired vapour concentrations. Consequently, a practical real-time age-corrected MAC output display on the screen was straightforward and simple to implement, since all I had to do was to write a small subroutine to calculate the value and display the numeric value continuously, and arrange for the value to be displayed in red and also trigger an audible alarm when less than a critical value (initially I chose the critical value to be 0.86—see the program below).

An interesting problem regarding the administration of anaesthesia at that time was the fact that with no less than four inhalational anaesthetic vapours in common use (halothane, isoflurane, desflurane, sevoflurane), and it was therefore virtually impossible to remember the appropriate settings for each combination of agent and age. Consequently the prospect of inadvertent awareness was ever present, and anaesthetists tended, therefore, to learn how to use one particular agent for most things even though other agents may well be more suitable in certain circumstances.

In view of this problem, the display of age-corrected MAC was an immediate success since one could now use any agent for any patient irrespective of age, simply by administering the agent in terms of MAC, and with the great benefit of essentially eliminating the possibility of inadvertent awareness simply by ensuring the age-corrected MAC was greater than a certain critical value—now taken to be 1 MAC (Hardman JG and Aitkenhead AR 2005). In fact we now had a working practical way of giving anaesthetics in terms of MAC units, as originally foreseen by Mapleson many years earlier in his insightful Clover lecture (Mapleson 1979). Our system at that time was therefore almost certainly the only such system in the UK, and possibly in the world.

The theatre program was later rewritten for the Linux operating system using the new Datex-Ohmeda AS3 monitors, having a much better data-stream (detailed in the Datex chapter).

### 6.1.1 MAC subroutine (MS-DOS)

The agent name and the end-tidal concentration (output by the Datex monitor) were used as inputs for the calculation, the \( MAC_{\text{age}=40} \) values for each agent being stored in simple look-up table in the following subroutine (written in QuickBASIC 4.5).

```basic
REM MS−DOS program
REM 1996 QuickBASIC 4.5
SUB mac (n2opercent, vapourname$, etvapour, ageofpatient%, bmac)
REM −−−−−−−−−−−−−−
REM Determines the current value of MAC
REM using the recent paper by Mapleson (BJA, 1996, vol 76, p 179−185)
REM Effect of age on MAC in humans: a meta−analysis
REM −−−−−−−−−−−−−−
REM new MAC sub using etn2o
REM returns the value of BIGMAC (bmac)
REM this is the newMAC which works correctly
REM − − − − − − − −
IF etvapour < 0 THEN etvapour = .001
n2o = n2opercent
v$ = vapourname$
vap = etvapour
A% = ageofpatient%
deltage% = A% − 40
BB = −.00269
```
REM this MAC sub is called from the end of PLOTVAPOUR sub
REM vapour is on Datex Ultima BOO and C04 (13,3) data strings
REM vapourcode$= ISO, HAL etc = ” ” when not selected
IF v$ = "" THEN mac40 = 0
IF v$ = "HAL" THEN mac40 = .75
IF v$ = "ISO" THEN mac40 = 1.17
IF v$ = "ENF" THEN mac40 = 1.63
IF v$ = "SEV" THEN mac40 = 1.8
IF v$ = "DES" THEN mac40 = 6.6
REM mac40 for N2O = 104
REM
do N2O calculation first
REM restrict n2o to zero or above
IF n2o < 0 THEN n2o = 0
REM eqn mac=(mac40)*10^(-0.00269* deltaage%)
macn2o = 104 * 10^ (BB * deltaage%)
IF macn2o <= 0 THEN
    Fmacn2o = .01: REM changed from 0 to .01 check
ELSE
    Fmacn2o = n2o / macn2o
END IF

do VAPOUR calc next
REM eqn mac=(mac40)*10^(-0.00269* deltaage%)
macvapour = mac40 * 10^ (BB * deltaage%)
IF macvapour <= 0 THEN
    totalFmac = Fmacn2o
ELSE
    Fmacvapour = (vap / macvapour)
    totalFmac = Fmacvapour + Fmacn2o
END IF

do not print to screen if printing last 20 mins fast data
IF pl20mf$ = "on" THEN GOTO MAClastline

A = Fmacn2o
B = Fmacvapour
C = totalFmac

COLOR green, screenbackcolour
REM cannot print digits with PRINT USING and
REM strings in same PRINT statement, so therefore
REM we have to print them separately (red if vap mac=0)
LOCATE 18, 68: PRINT SPACE$(11)
LOCATE 18, 68: PRINT "MAC ";
IF B <= 0 THEN
    COLOR red, screenbackcolour
    PRINT USING "#.##"; B;
    COLOR green, screenbackcolour
ELSE
    PRINT USING "#.##"; B;
END IF
PRINT "/";
6.2 Age corrected MAC charts

Sometime during the next couple of years I started wondering how I could create a paper nomogram-type chart for determining age-corrected MAC for use when I did lists at the QMC, since I was unable then to use my computer program (based in the thoracic theatre at the City Hospital).

The main problem was figuring out how best to incorporate the optional and flexible use of nitrous oxide, since the charts would not be particularly useful clinically unless they easily allowed for the effect of nitrous oxide. The design of such a chart was not at all straightforward and it was quite a long time before I formulated a suitable design which allowed nitrous-oxide use (see Figure 6.3). Eventually a single chart for each inhalational agent was generated using Perl and mathsPIC (Nickalls 1999, 2000; Syropoulos and Nickalls 2000), and this was then tested clinically over a period of time.

Figure 6.2: One of the first age-corrected iso-MAC charts, drawn using mathsPIC.
Encouraged by colleagues who tested these charts (one for each of the three main inhalational agents), I eventually submitted a paper to the British Journal of Anaesthesia (November 2001). However, it met with opposition from some of the reviewers who felt that my new paper and these charts failed to offer anything over and above Mapleson’s original 1996 paper. Well, at one level this is true—the raw data was the same in each paper. However, it was impossible to use the data presented in the Mapleson 1996 paper in a clinical setting to guide at all accurately the appropriate choice of end-tidal agent concentration for a particular patient. The reviewers simply failed to appreciate that by reformulating Mapleson’s data and supplementing it with nitrous-oxide data one could present the information in a way that would be a useful guide for practising anaesthetists, and essentially eliminate the guesswork which was otherwise involved.

I therefore telephoned Bill Mapleson and asked if he would agree to be a co-author, as together we would stand a better chance of getting the paper published. Bill Mapleson agreed immediately, and we submitted a new version of the paper (February 2003) which was then accepted and published in the *British Journal of Anaesthesia* (Nickalls and Mapleson 2003). We were fortunate in that the article was also the subject of an editorial (White, 2003).

Since then the use of these age-corrected iso-MAC charts have become generally used, and have also been included in the widely used *Oxford handbook of anaesthesia* (Allman and Wilson 2006).

### 6.3 Generating the charts

The charts were generated using QuickBasic 4.5 (MS-DOS), Perl and mathsPIC. I originally used a QuickBasic program (e.g. *iso-mac.dat*; see below) to generate the agent-specific data-files (for isoflurane, sevoflurane, desflurane) containing the data points for each of the iso-MAC curve (i.e. for the curves associated with the MAC values 0.6, 0.8, 1.0, 1.2, 1.4, 1.6). These data-files were coded with the letters j, k, m, n, p, q. For example the following program *iso-mac.bas* generated the isoflurane data-file *isoqdata.dat* (i.e. the data-file for the ‘q’ (iso-MAC 1.6) curve for isoflurane). In order to generate all the different data-files (a total of 3 × 6 different data-files) the program was run many times, each run having different values enabled for agent and MAC etc.

```basic
REM new iso-mac.bas
COLOR 15, 1
CLS
REM IF ageofpatient% < 1 THEN ageofpatient% = 1
REM ---------------------
REM this MAC sub is called from the end of PLOTVAPOUR sub
REM vapour is on BOO and C04 (13,3) data strings
REM vapourcode$= ISO, HAL etc = " " when not selected
REM IF v$ = " " THEN mac40 = 0
REM IF v$ = "HAL" THEN mac40 = .75
REM IF v$ = "ISO" THEN mac40 = 1.17
REM IF v$ = "ENF" THEN mac40 = 1.63
REM IF v$ = "SEV" THEN mac40 = 1.8
REM IF v$ = "DES" THEN mac40 = 6.6
REM mac40 for N2O = 104
REM ---------------------
```
REM ethNO = 100 – (etO2 + etCO2 + etvap)
REM
REM do N2O calculation first
REM restrict n2o to zero or above
REM IF n2o < 0 THEN n2o = 0
REM eqn  mac=(mac40)*10^(-0.00269* deltaage%)
REM
REM q = 1.6 mac = 1.17
REM p = 1.4 mac = 1.17
REM n = 1.2 mac = 1.17
REM m = 1 mac = 1.17
REM k = 0.8 mac = 1.17
REM j = 0.6 mac = 1.17

OPEN “isoqdata.dat” FOR OUTPUT AS #1
n = 1.6
code$ = “q”
mac40 = 1.17: REM isoflurane

REM
PRINT #1, “%% “ + code$ + “= mac40(iso) * “; n
FOR j = 5 TO 95 STEP 5
REM j = age
deltaage = j – 40
BB = –.00269
mac = (n * mac40) * 10 ^ (BB * deltaage)
PRINT j, mac
PRINT #1, “point(“ + code$; j; “){“; j; “,”; mac; “}”
s$ = s$ + code$ + STR$(j) + “ “
NEXT j
PRINT #1,
PRINT #1, “drawline(“ + s$ + “)“
REM $--------------------------

6.3.1 A data file for a single iso-MAC curve

The following output data-file (isoqdata.dat) was generated by the above program. This data-file contained the mathsPIC code for drawing the iso-MAC 1.6 curve (‘q’) for the agent isoflurane. This file was then one of the input files for another mathsPIC program which drew the whole graph.

% isoqdata.dat
% q = mac40(iso) * 1.6
point(q 5) { 5, 2.325176 } % manual
point(q 10) { 10, 2.25427 }
point(q 15) { 15, 2.185525 }
point(q 20) { 20, 2.118877 }
point(q 25) { 25, 2.054262 }
point(q 30) { 30, 1.991617 }
6.3.2 mathsPIC script for drawing the whole graph

Once having generated all the different data-files (above), a mathsPIC script was written to draw the axes, and to draw the graph by inputting all the relevant data-files. For example, the following mathsPIC script (mac–iso7.m) inputs each of the various data-files (one for each iso-MAC curve) and draws the complete isoflurane graph, outputting the \LaTeX form of the graph.

For those not familiar with \TeX and \LaTeX the complete process to be run through is roughly as follows: we first process the mathsPIC script via the mathsPIC program (a Perl program) to generate the \LaTeX (\texttt{.mt}) output file, and then we \LaTeX this file to generate the (\texttt{.dvi}) output file. Next we generate a PostScript version (using the \texttt{dvips} utility, and then define the Bounding Box (BB) (using GhostScript) and form the EPS version (i.e. by including the BB coordinates and then renaming the file). Finally we generate the associated (\texttt{.pdf}) files using the \texttt{epstopdf} utility.

Note that the particular mathsPIC program used at that time was actually an early $\beta$ version of the final mathsPIC program (Syropoulos A and Nickalls RWD 2005), so that the following mathsPIC script contains instances of the old \texttt{\variable{}} commands which were still being used (eventually changed to the Perl-like format \texttt{\var{}}).
ISOflurane Delta for N₂O = 0.75 = (66.6666/104) * 1.17

\begin{center}
\begin{tabular}{cccc}
0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 \\
2 & 2 & 2 & 2 \\
3 & 3 & 3 & 3 \\
4 & 4 & 4 & 4 \\
5 & 5 & 5 & 5 \\
6 & 6 & 6 & 6 \\
7 & 7 & 7 & 7 \\
8 & 8 & 8 & 8 \\
9 & 9 & 9 & 9 \\
10 & 10 & 10 & 10 \\
11 & 11 & 11 & 11 \\
12 & 12 & 12 & 12 \\
13 & 13 & 13 & 13 \\
14 & 14 & 14 & 14 \\
15 & 15 & 15 & 15 \\
16 & 16 & 16 & 16 \\
17 & 17 & 17 & 17 \\
18 & 18 & 18 & 18 \\
19 & 19 & 19 & 19 \\
20 & 20 & 20 & 20 \\
21 & 21 & 21 & 21 \\
22 & 22 & 22 & 22 \\
23 & 23 & 23 & 23 \\
24 & 24 & 24 & 24 \\
25 & 25 & 25 & 25 \\
26 & 26 & 26 & 26 \\
27 & 27 & 27 & 27 \\
28 & 28 & 28 & 28 \\
29 & 29 & 29 & 29 \\
30 & 30 & 30 & 30 \\
31 & 31 & 31 & 31 \\
32 & 32 & 32 & 32 \\
33 & 33 & 33 & 33 \\
34 & 34 & 34 & 34 \\
35 & 35 & 35 & 35 \\
36 & 36 & 36 & 36 \\
37 & 37 & 37 & 37 \\
38 & 38 & 38 & 38 \\
39 & 39 & 39 & 39 \\
40 & 40 & 40 & 40 \\
41 & 41 & 41 & 41 \\
42 & 42 & 42 & 42 \\
43 & 43 & 43 & 43 \\
44 & 44 & 44 & 44 \\
45 & 45 & 45 & 45 \\
46 & 46 & 46 & 46 \\
47 & 47 & 47 & 47 \\
48 & 48 & 48 & 48 \\
49 & 49 & 49 & 49 \\
50 & 50 & 50 & 50 \\
51 & 51 & 51 & 51 \\
52 & 52 & 52 & 52 \\
53 & 53 & 53 & 53 \\
54 & 54 & 54 & 54 \\
55 & 55 & 55 & 55 \\
56 & 56 & 56 & 56 \\
57 & 57 & 57 & 57 \\
58 & 58 & 58 & 58 \\
59 & 59 & 59 & 59 \\
60 & 60 & 60 & 60 \\
61 & 61 & 61 & 61 \\
62 & 62 & 62 & 62 \\
63 & 63 & 63 & 63 \\
64 & 64 & 64 & 64 \\
65 & 65 & 65 & 65 \\
66 & 66 & 66 & 66 \\
67 & 67 & 67 & 67 \\
68 & 68 & 68 & 68 \\
69 & 69 & 69 & 69 \\
70 & 70 & 70 & 70 \\
71 & 71 & 71 & 71 \\
72 & 72 & 72 & 72 \\
73 & 73 & 73 & 73 \\
74 & 74 & 74 & 74 \\
75 & 75 & 75 & 75 \\
76 & 76 & 76 & 76 \\
77 & 77 & 77 & 77 \\
78 & 78 & 78 & 78 \\
79 & 79 & 79 & 79 \\
80 & 80 & 80 & 80 \\
81 & 81 & 81 & 81 \\
82 & 82 & 82 & 82 \\
83 & 83 & 83 & 83 \\
84 & 84 & 84 & 84 \\
85 & 85 & 85 & 85 \\
86 & 86 & 86 & 86 \\
87 & 87 & 87 & 87 \\
88 & 88 & 88 & 88 \\
89 & 89 & 89 & 89 \\
90 & 90 & 90 & 90 \\
91 & 91 & 91 & 91 \\
92 & 92 & 92 & 92 \\
93 & 93 & 93 & 93 \\
94 & 94 & 94 & 94 \\
95 & 95 & 95 & 95 \\
96 & 96 & 96 & 96 \\
97 & 97 & 97 & 97 \\
98 & 98 & 98 & 98 \\
99 & 99 & 99 & 99 \\
100 & 100 & 100 & 100 \\
\end{tabular}
\end{center}
\begin{verbatim}
inputfile(isopdata.dat) %1.4
 \thickline%
inputfile(isondata.dat) % 1.2
 \thinline%
inputfile(isomdata.dat) % 1
 \thickline%
inputfile(isokdata.dat) % 0.8
 \thinline%
inputfile(isojdata.dat) %0.6
\end{verbatim}

---

---

\textbf{from mac.des.m}

\begin{enumerate}
\item variable(x){\{-1\}}
\item variable(x2){[x, advance(2)]}
\item point(h){[x2,2.475]}
\item text(MAC){[h]}
\end{enumerate}

\begin{enumerate}
\item variable(d){\{0.29\}}
\item variable(h6){\{0.88\}}\%0.9
\item text\{\begin{minipage}{29mm}\centering
\end{minipage}\}\{x,h6\}
\item variable(h8){[h6, advance(d)]}
\item text\{\begin{minipage}{29mm}\centering
\end{minipage}\}\{x,h8\}
\item variable(h10){[h8, advance(d)]}
\item text\{\begin{minipage}{29mm}\centering
\end{minipage}\}\{x,h10\}
\item variable(h12){[h10, advance(d)]}
\item text\{\begin{minipage}{29mm}\centering
\end{minipage}\}\{x,h12\}
\item variable(h14){[h12, advance(d)]}
\item text\{\begin{minipage}{29mm}\centering
\end{minipage}\}\{x,h14\}
\item variable(h16){[h14, advance(d)]}
\item text\{\begin{minipage}{29mm}\centering
\end{minipage}\}\{x,h16\}
\end{enumerate}

---

\begin{enumerate}
\item newcommand{\myleft} \%
\item framebox \%
\item begin{minipage}{29mm}\centering \%
\item End--expired (\%)
\item in 100\\%
\item oxygen \%
\item end{minipage} \%
\item \%
\item \%
\end{enumerate}

\text{\myleft}{\{-45, 2.0\}}

---

\begin{enumerate}
\item newcommand{\myrightb} \%
\item framebox \%
\item begin{minipage}{4cm} \%
\end{enumerate}
The following example is the \TeX{} code output by the above mathsPIC program.
\%\% want to print only some of the L axis scale (0.6--2.4), so do it manually
\axis left
  ticks with-values 0{\$\cdot$}6 0{\$\cdot$}8 1{\$\cdot$}0 1{\$\cdot$}2
  1{\$\cdot$}4 1{\$\cdot$}6
  1{\$\cdot$}8 2{\$\cdot$}0 2{\$\cdot$}2 2{\$\cdot$}4 /
  at 0.6 0.8 1.0 1.2 1.4
  1.6 1.8 2.0 2.2 2.4 / /
\axis bottom
  ticks with-values 0 10 20 30 40 50 60 70 80 90 100 /
  at 0 10 20 30 40 50 60 70 80 90 100 / /
\axis right
\%\%
  \{using N2O 67\%\} shift = 0.7523
  ticks with-values 0 0 0 0 0{\$\cdot$}6 0{\$\cdot$}8 1{\$\cdot$}0 1{\$\cdot$}2
  1{\$\cdot$}4 1{\$\cdot$}6 /
  at 0.7523 0.9523 1.1523 1.3523 1.5523 1.7523 1.9523 2.1523
  2.3523 / /
\%\% extra 50\% right axis shift = 0.5614
\% since this axis is off the graph then need new paper command
\% but do not use axis() option
\% paper\{units(0.7mm,3.818181cm)\range(-8,121)\range(0.5614,2.3614)\} 
\setcoordinatesystem units <.7mm,3.818181cm> 
\% ... note: xunits & yunits are different
\setplotarea x from -8 to 121, y from .5614 to 2.3614
\axis right \%\% seconds right axis for 50\% oxygen shift = 0.5614
  ticks with-values 0 0 0 0 0{\$\cdot$}6 0{\$\cdot$}8
  1{\$\cdot$}0 1{\$\cdot$}2 1{\$\cdot$}4 1{\$\cdot$}6 1{\$\cdot$}8 /
  at 0.5614 0.7614 0.9614 1.1614 1.3614
  1.5614 1.7614 1.9614 2.1614 2.3614 / /
\%\% beginSKIP
\newcommand\{\thickline\} {\setplotsymbol(\{\Large .\})}\%
\newcommand\{\thinline\} {\setplotsymbol(\{\tiny .\})}\%
\thickline
\%\% inputfile(isoqdata.dat) \%1.6 
\%\% ... start of file <isoqdata.dat>
\% q = macd(iso) + 1.6
\% point(q5){5,2.325176} (5 , 2.325176) \% manual
\% point(q10){10,2.25427} (10 , 2.25427 )
\% point(q15){15,2.185525} (15 , 2.185525 )
\% point(q20){20,2.118877} (20 , 2.118877 )
\% point(q25){25,2.054262} (25 , 2.054262 )
\% point(q30){30,1.991617} (30 , 1.991617 )
\% point(q35){35,1.930882} (35 , 1.930882 )
\% point(q40){40,1.872} (40 , 1.872 )
\% point(q45){45,1.814913} (45 , 1.814913 )
\% point(q50){50,1.759567} (50 , 1.759567 )
\% point(q55){55,1.705909} (55 , 1.705909 )
\% point(q60){60,1.653887} (60 , 1.653887 )
\% point(q65){65,1.603451} (65 , 1.603451 )
\plot 30 1.244761 35 1.206802 / \% m30m35
\plot 35 1.206802 40 1.17 / \% m35m40
\plot 40 1.17 45 1.134321 / \% m40m45
\plot 45 1.134321 50 1.099729 / \% m45m50
\plot 50 1.099729 55 1.066193 / \% m50m55
\plot 55 1.066193 60 1.033679 / \% m55m60
\plot 60 1.033679 65 1.002157 / \% m60m65
\plot 65 1.002157 70 .9715963 / \% m65m70
\plot 70 .9715963 75 .9419674 / \% m70m75
\plot 75 .9419674 80 .9132419 / \% m75m80
\plot 80 .9132419 85 .8853925 / \% m80m85
\plot 85 .8853925 90 .8583924 / \% m85m90
\plot 90 .8583924 95 .8322156 / \% m90m95

%% \plot 5 1.162588 10 1.127135 / \% k5k10
%% \plot 10 1.127135 15 1.092763 / \% k10k15
%% \plot 15 1.092763 20 1.059439 / \% k15k20
%% \plot 20 1.059439 25 1.027131 / \% k20k25
%% \plot 25 1.027131 30 .9958085 / \% k25k30
%% \plot 30 .9958085 35 .9654412 / \% k30k35
%% \plot 35 .9654412 40 .936 / \% k35k40
%% \plot 40 .936 45 9074566 / \% k40k45
%% \plot 45 9074566 50 .8797836 / \% k45k50
%% \plot 50 .8797836 55 .8529544 / \% k50k55
%% \plot 55 .8529544 60 .8269435 / \% k55k60
%% \plot 60 .8269435 65 .8017257 / \% k60k65
%% \plot 65 .8017257 70 .7772771 / \% k65k70
%% \plot 70 .7772771 75 .7535739 / \% k70k75
%% \plot 75 .7535739 80 .7305936 / \% k75k80
%% \plot 80 .7305936 85 .708314 / \% k80k85

%% \plot 85 .708314 90 .6867139 / \% k85k90
%% \plot 90 .6867139 95 .6657725 / \% k90k95

%% \drawline(k5k10k15k20k25k30k35k40k45k50k55k60k65k70k75k80k85k90k95)
\plot 85 .708314 90 .6867139 / \%\% k85k90
\plot 90 .6867139 95 .6657725 / \%\% k90k95
\%\% drawpoint(k10k20k30k40k50k60k70k80k90k)
\put {\$bullet\$} at 10 1.127135 \%\% k10
\put {\$bullet\$} at 20 1.059439 \%\% k20
\put {\$bullet\$} at 30 .9958085 \%\% k30
\put {\$bullet\$} at 40 .936 \%\% k40
\put {\$bullet\$} at 50 .8797836 \%\% k50
\put {\$bullet\$} at 60 .8269435 \%\% k60
\put {\$bullet\$} at 70 .7772771 \%\% k70
\put {\$bullet\$} at 80 .7305936 \%\% k80
\put {\$bullet\$} at 90 .6867139 \%\% k90
\%\% ... end of file  <isokdata.dat>
\thinline%
\%\% inputfile(isojdata.dat) \%0.6
\%\% ... start of file  <isojdata.dat>
\%\% j= mac40(iso) * .6
\%\% point(j5)[5,.871941] (5 , .871941 ) \%\% manual
\%\% point(j10)[10,.8453511] (10 , .8453511 )
\%\% point(j15)[15,.819572] (15 , .819572 )
\%\% point(j20)[20,.794579] (20 , .794579 )
\%\% point(j25)[25,.7703483] (25 , .7703483 )
\%\% point(j30)[30,.7468564] (30 , .7468564 )
\%\% point(j35)[35,.7240809] (35 , .7240809 )
\%\% point(j40)[40,.702 ] (40 , .702 )
\%\% point(j45)[45,.6805924] (45 , .6805924 )
\%\% point(j50)[50,.6598377] (50 , .6598377 )
\%\% point(j55)[55,.6397159] (55 , .6397159 )
\%\% point(j60)[60,.6202077] (60 , .6202077 )
\%\% point(j65)[65,.6012943] (65 , .6012943 )
\%\% point(j70)[70,.5829578] (70 , .5829578 )
\%\% point(j75)[75,.5651804] (75 , .5651804 )
\%\% point(j80)[80,.5479452] (80 , .5479452 )
\%\% point(j85)[85,.5312335] (85 , .5312335 )
\%\% point(j90)[90,.5150355] (90 , .5150355 )
\%\% point(j95)[95,.4993294] (95 , .4993294 )
\%\% drawline(j5j10j15j20j25j30j35j40j45j50j55j60j65j70j75j80j85j90j95)
\plot 5 .871941 10 .8453511 / \%\% j5j10
\plot 10 .8453511 15 .819572 / \%\% j10j15
\plot 15 .819572 20 .794579 / \%\% j15j20
\plot 20 .794579 25 .7703483 / \%\% j20j25
\plot 25 .7703483 30 .7468564 / \%\% j25j30
\plot 30 .7468564 35 .7240809 / \%\% j30j35
\plot 35 .7240809 40 .702 / \%\% j35j40
\plot 40 .702 45 .6805924 / \%\% j40j45
\plot 45 .6805924 50 .6598377 / \%\% j45j50
\plot 50 .6598377 55 .6397159 / \%\% j50j55
\plot 55 .6397159 60 .6202077 / \%\% j55j60
\plot 60 .6202077 65 .6012943 / \%\% j60j65
\plot 65 .6012943 70 .5829578 / \%\% j65j70
\plot 70 .5829578 75 .5651804 / \%\% j70j75
\plot 75 .5651804 80 .5479452 / \%\% j75j80
\plot 80 .5479452 85 .5312335 / \%\% j80j85
\plot 85 .5312335 90 .5150355 / \%\% j85j90
\plot 90 .5150355 95 .4993294 / \%|j90\j95
% ... end of file <isojdata.dat>
%\endSKIP

%\from mac--des.m
%\variable(x){-1} (-1)
%\variable(x2){x,\advance(2)} (1)
%\point(h){x2,2.475} (1, 2.475)
%\text(MAC){h}
\put{MAC}{1 2.475}
% vertical diff = 0.29 units
%\variable(d){0.29} (.29)
%\variable(h6){0.88} (.88)
%\text(\fbox{$0 \cdot 6$}){x,h6}
\put{\fbox{$0 \cdot 6$}}{-1 .88}
%\variable(h8){h6,\advance(d)} (1.17)
%\text(\fbox{$0 \cdot 8$}){x,h8}
\put{\fbox{$0 \cdot 8$}}{-1 1.17}
%\variable(h10){h8,\advance(d)} (1.46)
%\text(\fbox{$1 \cdot 0$}){x,h10}
\put{\fbox{$1 \cdot 0$}}{-1 1.46}
%\variable(h12){h10,\advance(d)} (1.75)
%\text(\fbox{$1 \cdot 2$}){x,h12}
\put{\fbox{$1 \cdot 2$}}{-1 1.75}
%\variable(h14){h12,\advance(d)} (2.04)
%\text(\fbox{$1 \cdot 4$}){x,h14}
\put{\fbox{$1 \cdot 4$}}{-1 2.04}
%\variable(h16){h14,\advance(d)} (2.33)
%\text(\fbox{$1 \cdot 6$}){x,h16}
\put{\fbox{$1 \cdot 6$}}{-1 2.33}

%\newcommand{\myleft}{\%
%\framebox{
%\begin{minipage}{29mm}\centering
%\text{oxygen}\%
%\end{minipage} \%
} %\}%
%\text(\myleft){-45, 2.0}
\put{\myleft}{-45 2}

%\newcommand{\myrightb}{\%
%\begin{minipage}{4cm}
%\text{N$_2$O}\hspace{7.5mm}N$_2$O
%\end{minipage}
%\%
%\}% end of newcommand
%\text(\myrightb){102, 2.657}[l] %\% was 2.6
\put{\myrightb}[l]{102 2.657}
\newcommand{\mybottom}{\text{(years)}}
\text{\mybottom}{46, 0.15}
\put{\mybottom}{46 .15}
\text{\copyright \ RWD Nickalls\ 2001}{22,0.5}
\text{\large ISOFLURANE}{46, 2.7} \% \% 80
\put{\large ISOFLURANE}{46 2.7}

\linethickness=0.4pt \% equivalent to \text{\tiny .}
\linethickness=0.6pt \% half way between \text{\tiny} and \text{\normalsize}
\setdashes
\variable{x5}{5} \% Left X value
\variable{x6}{100} \% Right X value
\variable{y16}{2.3523} \% 2.3523
\variable{y14}{2.1523} \% 2.1523
\variable{y12}{1.9523} \% 1.9523
\variable{y10}{1.7523} \% 1.7523
\variable{y08}{1.5523} \% 1.5523
\variable{y06}{1.3523} \% 1.3523
\variable{y04}{1.1523} \% 1.1523
\variable{y02}{0.9523} \% 0.9523
\variable{y00}{0.7523} \% 0.7523
\point{L16}{5,2.3523}
\point{R16}{100,2.3523}
\point{L14}{5,2.1523}
\point{R14}{100,2.1523}
\point{L12}{5,1.9523}
\point{R12}{100,1.9523}
\point{L10}{5,1.7523}
\point{R10}{100,1.7523}
\point{L08}{5,1.5523}
\point{R08}{100,1.5523}
\point{L06}{5,1.3523}
\point{R06}{100,1.3523}
\point{L04}{5,1.1523}
\point{R04}{100,1.1523}
\point{L02}{5,0.9523}
\point{R02}{100,0.9523}
\point{L00}{5,0.7523}
\point{R00}{100,0.7523}
\drawline{L16R16,L14R14,L12R12,L10R10,L08R08,L06R06,L04R04,L02R02,L00R00}
\putrule from 5 2.3523 to 100 2.3523 \% L16R16
\putrule from 5 2.1523 to 100 2.1523 \% L14R14
\putrule from 5 1.9523 to 100 1.9523 \% L12R12
\putrule from 5 1.7523 to 100 1.7523 \% L10R10
\putrule from 5 1.5523 to 100 1.5523 \% L08R08
\putrule from 5 1.3523 to 100 1.3523 \% L06R06
\putrule from 5 1.1523 to 100 1.1523 \% L04R04
\putrule from 5 .9523 to 100 .9523 \% L02R02
\putrule from 5 .7523 to 100 .7523 \% L00R00
6.3.3 Final mathsPIC program for making the version of BJA charts for the Oxford Handbook of Anaesthesia

This version of the mathsPIC program (mac–iso8t.m) incorporated axis legend rotation (using \LaTeX{} and PostScript).

```
%% mac–iso8T.m (TEST version modified from mac–iso8.m)
%% Jan 10, 2006
%% mathsPICperl version
%% final graph/chart for the bja
%% with decimals ($\cdot$) and \fbox{}
%% new curves for anaesthesia
%% mathsPIC

%% to test rotation legend on axes
%% $\rightarrow$
%% \% for percent
%% enter the Y2 Y1 values in ET units
%% adjust \oddsidemargin
%% adjust linethickness
%% adjust minipage $\rightarrow$ 3.6cm
%% adjust possn of MAC
%% remove isoflurane word from ylegend
%% push Isoflutane title up
%% push age down

\documentclass[a4paper,12pt]{article}
\usepackage{mathspic}
\usepackage{decimal,rotating}

\begin{document}
\% \oddsidemargin=-17mm
\% \framebox{\% 
\begin{picture}
\% structure copied from mac–des.m
```
ISOflurane Delta for N2O = 0.75 = (66.6666/104) * 1.17

\[
\text{y units} = \frac{12\text{cm}}{2.2} = 5.454545
\]

\[
\text{units(mm,5.454545cm) xrange(-5,100) yrange(0.4,2.6) axes(L) ticks(10,0.2)}
\]

\[
\text{paper{units(0.7mm,3.818181cm) xrange(-8,100) yrange(-8,100) yrange(0.4,2.6)}}
\]

\% want to print only some of the L axis scale (0.6–2.4), so do it manually

\[
\text{axis left}
\]

\[
\text{ticks withvalues 0}\{\text{\cdot}\} 6 0\{\text{\cdot}\} 8 1\{\text{\cdot}\} 0 1\{\text{\cdot}\} 2 1\{\text{\cdot}\} 4 1\{\text{\cdot}\} 6 1\{\text{\cdot}\} 8 2\{\text{\cdot}\} 0 2\{\text{\cdot}\} 2 2\{\text{\cdot}\} 4 / \at 0.60 0.80 1.00 1.20 1.40 1.60 1.80 2.00 2.20 2.40 / /}
\]

\[
\text{axis bottom}
\]

\[
\text{ticks withvalues 0 10 20 30 40 50 60 70 80 90 100 / at 0 10 20 30 40 50 60 70 80 90 100 / /}
\]

\[
\text{axis right}
\]

\[
\text{ticks withvalues 0 0\{\text{\cdot}\} 2 0\{\text{\cdot}\} 4 0\{\text{\cdot}\} 6 0\{\text{\cdot}\} 8 1\{\text{\cdot}\} 0 1\{\text{\cdot}\} 2 1\{\text{\cdot}\} 4 1\{\text{\cdot}\} 6 1\{\text{\cdot}\} 8 / at 0.7523 0.9523 1.1523 1.3523 1.5523 1.7523 1.9523 2.1523 2.3523 / /}
\]

\% extra 50% right axis shift = 0.5614
\%
\% since this axis is off the graph then need new paper command
\%
\% but do not use axis() option
\%
\[
\text{paper{units(0.7mm,3.818181cm) xrange(-8,121) yrange(0.5614,2.3614)}}
\]

\[
\text{axis right} \%\% seconds right axis for 50% oxygen shift = 0.5614
\]

\[
\text{ticks withvalues 0 0\{\text{\cdot}\} 2 0\{\text{\cdot}\} 4 0\{\text{\cdot}\} 6 0\{\text{\cdot}\} 8 1\{\text{\cdot}\} 0 1\{\text{\cdot}\} 2 1\{\text{\cdot}\} 4 1\{\text{\cdot}\} 6 1\{\text{\cdot}\} 8 / at 0.5614 0.7614 0.9614 1.1614 1.3614 1.5614 1.7614 1.9614 2.1614 2.3614 / /}
\]

\% beginSKIP
\%
\% newcommand{\thickline}{\setplotsymbol{\Large .})}%
\%
\% newcommand{\thinline}{\setplotsymbol{\tiny .})}% = BJA graphs
\%
\% make thin line a bit thicker for the OUP graphs
\%
\% newcommand{\thinline}{\setplotsymbol{\large .})}%
\%
\text{\thickline% inputfile(isoqdata8.dat) %1.6}
\%
\text{\thinline% inputfile(isopdata8.dat) %1.4}
\%
\text{\thickline% inputfile(isondata8.dat) %1.2}
\%
\text{\thinline% inputfile(isomdata8.dat) %1}
inputfile(isokdata8.dat) % 0.8
\thinline%
inputfile(isojdata8.dat) %0.6
\%endSKIP

\%-- from mac - des.m
var x=-1
var x2=x + 2
point(h){x2,2.55}\% 2.475
text(MAC){h}
\% vertical diff = 0.29 units \% 0.28
var d=0.29
var h6=0.88
text(\fbox{\$\cdot 6\$}){x,h6}

var h8=h6 + d
text(\fbox{\$\cdot 8\$}){x,h8}

var h10=h8 + d
text(\fbox{\$\cdot 0\$}){x,h10}

var h12=h10 +d
text(\fbox{\$\cdot 2\$}){x,h12}

var h14 = h12+d
text(\fbox{\$\cdot 4\$}){x,h14}

var h16=h14 +d
text(\fbox{\$\cdot 6\$}){x,h16}

\%\%======new rotated legends from macATdes2.pl=====================
var y2=2.6
var y1=0.4
\%
\%--- newcommand{\ylegend}{\sf End tidal (\%) in 100\%, \% oxygen/air}\%
\%--- - determine string length ----> Yunits etc---------
\newlength{\ylength}\%
\settowidth{\ylength}{\ylegend}\%
\text(answer ylength = \number\ylength){37,−0.4}
\% halflength/3.818=0.777 y units \%
text(\turnbox{90}{\ylegend}){−25, y1+((y2−y1)/2) − 0.777}
\%
\%
\begin{SKIP}
\%
\%
\%\% newcommand{\rightylegend}{\sf End - tidal (\%) in N_2$O_2$}\%
\newlength{\rylength}\%
\settowidth{\rylength}{\rightylegend}\%
text(answerylength = \number\rylength){37,−1.0}
\% halflength/3.818=0.7188 y units \%
text(\turnbox{270}{\rightylegend}){(140, y1+((y2−y1)/2) + 0.7188)}
\%
\begin{minipage}{29mm}
\centering
End--expired (\% in 100\% oxygen)
\end{minipage}
\text{\(-45, 2.0\)}
\end{minipage}
\newcommand{\myrightb}{%\fbox{\begin{minipage}{3.5cm} 3.8cm \hspace{9mm} 67 \hspace{8mm} 50 \hspace{7.5mm} N_2O \hspace{7.5mm} N_2O \hspace{7.5mm} N_2O\end{minipage}}% end of newcommand
\text{\(89.143, 2.657\)}[l]
\newcommand{\mybottom}{Age (years)}%\text{\(46, 0.12\) % 0.15
\text{\(46, 2.8\) % 80
\text{\(46, 0.12\) % 0.15
\text{\(46, 2.8\) % 80
% draw horizontal dashed lines %\linethickness=0.4pt %\text{\(\small\)} %\linethickness=0.6pt %\text{\(\scriptsize\)} %\setdashes %\text{\(\footnotesize\)} %\copyright \text{\(RWD Nickalls\ 2003\)}}%
% draw the dashes from Left to Right
% (so have small gap at right axis)
drawline(L16R16, L14R14, L12R12, L10R10, L08R08, L06R06, L04R04, L02R02, L00R00)
\endpicture
\begin{document}
\begin{picture}
\usepackage{mathspic}
\usepackage{decimal,rotating}
\begin{document}
\oddsidemargin=-17mm
\framebox{\begin{document}
\usepackage{sf font}
\usepackage{mathspic}
\begin{document}
\oddsidemargin=-17mm
\framebox{\begin{document}
\texttt{\% Command line: /usr/local/bin/mpic100.pl mac–iso8t.m}
\texttt{\% Input filename : mac–iso8t.m}
\texttt{\% Output filename: mac–iso8t.mt}
\texttt{\% Date & time: 2006/01/13  09:19:33}
\texttt{\%}
\texttt{\% mac–iso8T.m (TEST version modified from mac–iso8.m)}
\texttt{\% Jan 10, 2006}
\texttt{\% mathsPICperl version}
\texttt{\% final graph/chart for the bja}
\texttt{\% wih decimals ($\cdot$) and \fbox{}}
\texttt{\% new curves for anaesthesia}
\texttt{\% mathsPIC}
\texttt{\% to test rotation legend on axes}
\texttt{\%}
\texttt{\% ISOflurane Delta for N2O = 0.75 = (66.6666/104)*1.17}
\texttt{\% y units = 12cm/2.2 = 5.454545}
\texttt{\% paper{units(mm,5.454545cm) xrange(−5,100) yrange(0,4,2.6) axes(L) ticks(10,0.2)}}
\texttt{\% paper{units(0.7mm,3.818181cm) xrange(−8,100) yrange(0,4,2.6)}}
\texttt{\% setcoordinatesystem units <0.7mm,3.818181cm>}
\texttt{\% setplotarea x from −8.00000 to 100.00000, y from 0.400000 to 2.600000}
\texttt{\%}
\texttt{\% \% ISOflurane Delta for N2O = 0.75 = (66.6666/104)*1.17}
\texttt{\% y units = 12cm/2.2 = 5.454545}
\texttt{\% paper{units(mm,5.454545cm) xrange(−5,100) yrange(0,4,2.6) axes(L) ticks(10,0.2)}}
\texttt{\% paper{units(0.7mm,3.818181cm) xrange(−8,100) yrange(0,4,2.6)}}
\texttt{\% setcoordinatesystem units <0.7mm,3.818181cm>}
\texttt{\% setplotarea x from −8.00000 to 100.00000, y from 0.400000 to 2.600000}
\texttt{\%}
\texttt{\% \% ISOflurane Delta for N2O = 0.75 = (66.6666/104)*1.17}
\texttt{\% y units = 12cm/2.2 = 5.454545}
\texttt{\% paper{units(mm,5.454545cm) xrange(−5,100) yrange(0,4,2.6) axes(L) ticks(10,0.2)}}
\texttt{\% paper{units(0.7mm,3.818181cm) xrange(−8,100) yrange(0,4,2.6)}}
\texttt{\% setcoordinatesystem units <0.7mm,3.818181cm>}
\texttt{\% setplotarea x from −8.00000 to 100.00000, y from 0.400000 to 2.600000}
\texttt{\%}
\texttt{\% want to print only some of the L axis scale (0.6–2.4), so do it manually}
\texttt{\% axis left}
\texttt{\% ticks withvalues 0\{\$\cdot\}6 0\{\$\cdot\}8 1\{\$\cdot\}0 1\{\$\cdot\}2}
\texttt{\% 1\{\$\cdot\}4 1\{\$\cdot\}6}
\texttt{\% 1\{\$\cdot\}8 2\{\$\cdot\}0 2\{\$\cdot\}2 2\{\$\cdot\}4 /}
\texttt{\% at 0.60 0.80 1.00 1.20 1.40}
\texttt{\% 1.60 1.80 2.00 2.20 2.40 / /}
\end{document}
\end{picture}
\end{document}
CHAPTER 6. AGE CORRECTED MAC RWD Nickalls

\% drawline(q5 q10 q15 q20 q25 q30 q35 q40 q45 q50 q55 q60 q65 q70 q75 q80 q85 q90 q95)
\plot 5.00000 2.32518 10.00000 2.25427 / \% q5q10
\plot 10.00000 2.25427 15.00000 2.18553 / \% q10q15
\plot 15.00000 2.18553 20.00000 2.11888 / \% q15q20
\plot 20.00000 2.11888 25.00000 2.05426 / \% q20q25
\plot 25.00000 2.05426 30.00000 1.99162 / \% q25q30
\plot 30.00000 1.99162 35.00000 1.93088 / \% q30q35
\plot 35.00000 1.93088 40.00000 1.87200 / \% q35q40
\plot 40.00000 1.87200 45.00000 1.81491 / \% q40q45
\plot 45.00000 1.81491 50.00000 1.75957 / \% q45q50
\plot 50.00000 1.75957 55.00000 1.70591 / \% q50q55
\plot 55.00000 1.70591 60.00000 1.65389 / \% q55q60
\plot 60.00000 1.65389 65.00000 1.60345 / \% q60q65
\plot 65.00000 1.60345 70.00000 1.55455 / \% q65q70
\plot 70.00000 1.55455 75.00000 1.50715 / \% q70q75
\plot 75.00000 1.50715 80.00000 1.46119 / \% q75q80
\plot 80.00000 1.46119 85.00000 1.41663 / \% q80q85
\plot 85.00000 1.41663 90.00000 1.37343 / \% q85q90
\plot 90.00000 1.37343 95.00000 1.33154 / \% q90q95
\% drawpoint(q10 q20 q30 q40 q50 q60 q70 q80 q90)
\put \{$\bullet$\} at 10.00000 2.25427 \% q10
\put \{$\bullet$\} at 20.00000 2.11888 \% q20
\put \{$\bullet$\} at 30.00000 1.99162 \% q30
\put \{$\bullet$\} at 40.00000 1.87200 \% q40
\put \{$\bullet$\} at 50.00000 1.75957 \% q50
\put \{$\bullet$\} at 60.00000 1.65389 \% q60
\put \{$\bullet$\} at 70.00000 1.55455 \% q70
\put \{$\bullet$\} at 80.00000 1.46119 \% q80
\put \{$\bullet$\} at 90.00000 1.37343 \% q90
\% ... end of file <isoqdata8.dat> loop [1]
\thinline%
\% inputfile(isopdata8.dat) %1.4
\% ... start of file <isopdata8.dat> loop [1]
\% \% Iteration number: 1
\% \% p = mac40(iso) + 1.4
\% \% p0 = mac40(iso) + 1.4
\% \% point(p5)\{(5.0,0.034529 \% manual \ p5 = (5.00000, 0.03453)
\% \% point(p10)\{(10.0,1.972486) \ p10 = (10.00000, 1.97249)
\% \% point(p15)\{(15.0,1.912335) \ p15 = (15.00000, 1.91233)
\% \% point(p20)\{(20.0,1.854018) \ p20 = (20.00000, 1.85402)
\% \% point(p25)\{(25.0,1.797479) \ p25 = (25.00000, 1.79748)
\% \% point(p30)\{(30.0,1.742665) \ p30 = (30.00000, 1.74266)
\% \% point(p35)\{(35.0,1.689522) \ p35 = (35.00000, 1.68952)
\% \% point(p40)\{(40.0,1.638) \ p40 = (40.00000, 1.63800)
\% \% point(p45)\{(45.0,1.588049) \ p45 = (45.00000, 1.58805)
\% \% point(p50)\{(50.0,1.539621) \ p50 = (50.00000, 1.53962)
\% \% point(p55)\{(55.0,1.49267) \ p55 = (55.00000, 1.49267)
\% \% point(p60)\{(60.0,1.447151) \ p60 = (60.00000, 1.44715)
\% \% point(p65)\{(65.0,1.403020) \ p65 = (65.00000, 1.40302)
\% \% point(p70)\{(70.0,1.360235) \ p70 = (70.00000, 1.36024)
\% \% point(p75)\{(75.0,1.318754) \ p75 = (75.00000, 1.31875)
\% \% point(p80)\{(80.0,1.278539) \ p80 = (80.00000, 1.27854)
\% \% point(p85)\{(85.0,1.23955) \ p85 = (85.00000, 1.23955)
\% \% point(p90)\{(90.0,1.201749) \ p90 = (90.00000, 1.20175)
CHAPTER 6. AGE CORRECTED MAC RWD Nickalls

\%\% point(p95) \{95,1.165102\} \hspace{1em} p95 = (95.00000, 1.16510)
\%\% drawline(p5 p10 p15 p20 p25 p30 p35 p40 p45 p50 p55 p60 p65 p70 p75 p80 p85 p90 p95)
\plot 5.00000 2.03453 10.00000 1.97249 / \%\% p5p10
\plot 10.00000 1.97249 15.00000 1.91233 / \%\% p10p15
\plot 15.00000 1.91233 20.00000 1.85402 / \%\% p15p20
\plot 20.00000 1.85402 25.00000 1.79748 / \%\% p20p25
\plot 25.00000 1.79748 30.00000 1.74266 / \%\% p25p30
\plot 30.00000 1.74266 35.00000 1.68952 / \%\% p30p35
\plot 35.00000 1.68952 40.00000 1.63800 / \%\% p35p40
\plot 40.00000 1.63800 45.00000 1.58805 / \%\% p40p45
\plot 45.00000 1.58805 50.00000 1.53962 / \%\% p45p50
\plot 50.00000 1.53962 55.00000 1.49267 / \%\% p50p55
\plot 55.00000 1.49267 60.00000 1.44715 / \%\% p55p60
\plot 60.00000 1.44715 65.00000 1.40302 / \%\% p60p65
\plot 65.00000 1.40302 70.00000 1.36024 / \%\% p65p70
\plot 70.00000 1.36024 75.00000 1.31875 / \%\% p70p75
\plot 75.00000 1.31875 80.00000 1.27854 / \%\% p75p80
\plot 80.00000 1.27854 85.00000 1.23955 / \%\% p80p85
\plot 85.00000 1.23955 90.00000 1.20175 / \%\% p85p90
\plot 90.00000 1.20175 95.00000 1.16510 / \%\% p90p95
\%\% ... end of file <isopdata8.dat> loop [1]
\thickline%
\%\% inputfile(isondata8.dat) % 1.2
\%\% ... start of file <isondata8.dat> loop [1]
\%\% Iteration number: 1
\%\% n= mac40(iso) * 1.2
\%\% point(n5) \{5,1.743882\} \%\% manual n5 = (5.00000, 1.74388)
\%\% point(n10) \{10,1.690702\} \hspace{1em} n10 = (10.00000, 1.69070)
\%\% point(n15) \{15,1.639144\} \hspace{1em} n15 = (15.00000, 1.63914)
\%\% point(n20) \{20,1.589158\} \hspace{1em} n20 = (20.00000, 1.58916)
\%\% point(n25) \{25,1.540697\} \hspace{1em} n25 = (25.00000, 1.54070)
\%\% point(n30) \{30,1.493713\} \hspace{1em} n30 = (30.00000, 1.49371)
\%\% point(n35) \{35,1.448162\} \hspace{1em} n35 = (35.00000, 1.44816)
\%\% point(n40) \{40,1.404\} \hspace{1em} n40 = (40.00000, 1.40400)
\%\% point(n45) \{45,1.361185\} \hspace{1em} n45 = (45.00000, 1.36119)
\%\% point(n50) \{50,1.319675\} \hspace{1em} n50 = (50.00000, 1.31967)
\%\% point(n55) \{55,1.279432\} \hspace{1em} n55 = (55.00000, 1.27943)
\%\% point(n60) \{60,1.240415\} \hspace{1em} n60 = (60.00000, 1.24042)
\%\% point(n65) \{65,1.202589\} \hspace{1em} n65 = (65.00000, 1.20259)
\%\% point(n70) \{70,1.165916\} \hspace{1em} n70 = (70.00000, 1.16592)
\%\% point(n75) \{75,1.130361\} \hspace{1em} n75 = (75.00000, 1.13036)
\%\% point(n80) \{80,1.09589\} \hspace{1em} n80 = (80.00000, 1.09589)
\%\% point(n85) \{85,1.062471\} \hspace{1em} n85 = (85.00000, 1.06247)
\%\% point(n90) \{90,1.030071\} \hspace{1em} n90 = (90.00000, 1.03007)
\%\% point(n95) \{95,0.9986387\} \hspace{1em} n95 = (95.00000, 0.99866)
\%\% drawline(n5 n10 n15 n20 n25 n30 n35 n40 n45 n50 n55 n60 n65 n70 n75 n80 n85 n90 n95)
\plot 5.00000 1.74388 10.00000 1.69070 / \%\% n5n10
\plot 10.00000 1.69070 15.00000 1.63914 / \%\% n10n15
\plot 15.00000 1.63914 20.00000 1.58916 / \%\% n15n20
\plot 20.00000 1.58916 25.00000 1.54070 / \%\% n20n25
\plot 25.00000 1.54070 30.00000 1.49371 / \%\% n25n30
\plot 30.00000 1.49371 35.00000 1.44816 / \%\% n30n35
\plot 35.00000 1.44816 40.00000 1.40400 / \% n35n40
\plot 40.00000 1.40400 45.00000 1.36119 / \% n40n45
\plot 45.00000 1.36119 50.00000 1.31967 / \% n45n50
\plot 50.00000 1.31967 55.00000 1.27943 / \% n50n55
\plot 55.00000 1.27943 60.00000 1.24042 / \% n55n60
\plot 60.00000 1.24042 65.00000 1.20259 / \% n60n65
\plot 65.00000 1.20259 70.00000 1.16592 / \% n65n70
\plot 70.00000 1.16592 75.00000 1.13036 / \% n70n75
\plot 75.00000 1.13036 80.00000 1.09589 / \% n75n80
\plot 80.00000 1.09589 85.00000 1.06247 / \% n80n85
\plot 85.00000 1.06247 90.00000 1.03007 / \% n85n90
\plot 90.00000 1.03007 95.00000 0.99866 / \% n90n95
\% drawpoint(n10 n20 n30 n40 n50 n60 n70 n80 n90)
\put {$\bullet$} at 10.00000 1.69070 \% n10
\put {$\bullet$} at 20.00000 1.58916 \% n20
\put {$\bullet$} at 30.00000 1.49371 \% n30
\put {$\bullet$} at 40.00000 1.40400 \% n40
\put {$\bullet$} at 50.00000 1.31967 \% n50
\put {$\bullet$} at 60.00000 1.24042 \% n60
\put {$\bullet$} at 70.00000 1.16592 \% n70
\put {$\bullet$} at 80.00000 1.09589 \% n80
\put {$\bullet$} at 90.00000 1.03007 \% n90
\% ... end of file <isondat8.dat> loop [1]
\thinline%
\% ... start of file <isondat8.dat> loop [1]
\% Iteration number: 1
\% m = mac40(iso) * 1
\% point(m5){5, 1.453235} m5 = (5.00000, 1.45324)
\% point(m10){10,1.408918} m10 = (10.00000, 1.40892)
\% point(m15){15,1.365953} m15 = (15.00000, 1.36595)
\% point(m20){20,1.324298} m20 = (20.00000, 1.32430)
\% point(m25){25,1.283914} m25 = (25.00000, 1.28391)
\% point(m30){30,1.244761} m30 = (30.00000, 1.24476)
\% point(m35){35,1.206802} m35 = (35.00000, 1.20680)
\% point(m40){40,1.177} m40 = (40.00000, 1.17000)
\% point(m45){45,1.134321} m45 = (45.00000, 1.13432)
\% point(m50){50,1.099729} m50 = (50.00000, 1.09973)
\% point(m55){55,1.066193} m55 = (55.00000, 1.06619)
\% point(m60){60,1.033679} m60 = (60.00000, 1.03368)
\% point(m65){65,1.002157} m65 = (65.00000, 1.00216)
\% point(m70){70,0.971963} m70 = (70.00000, 0.97160)
\% point(m75){75,0.9419674} m75 = (75.00000, 0.94197)
\% point(m80){80,0.9132419} m80 = (80.00000, 0.91324)
\% point(m85){85,0.8853925} m85 = (85.00000, 0.88539)
\% point(m90){90,0.8583924} m90 = (90.00000, 0.85839)
\% point(m95){95,0.8322156} m95 = (95.00000, 0.83222)
\% drawline(m5 m10 m15 m20 m25 m30 m35 m40 m45 m50 m55 m60 m65 m70 m75 m80 m85 m90 m95)
\plot 5.00000 1.45324 10.00000 1.40892 / \% m5m10
\plot 10.00000 1.40892 15.00000 1.36595 / \% m10m15
\plot 15.00000 1.36595 20.00000 1.32430 / \% m15m20
\plot 20.00000 1.32430 25.00000 1.28391 / \% m20m25
\plot 25.00000 1.28391 30.00000 1.24476 / \% m25m30
CHAPTER 6. AGE CORRECTED MAC RWD Nickalls

\plot 30.00000 1.24476 35.00000 1.20680 / \% m30m35
\plot 35.00000 1.20680 40.00000 1.17000 / \% m35m40
\plot 40.00000 1.17000 45.00000 1.13432 / \% m40m45
\plot 45.00000 1.13432 50.00000 1.09973 / \% m45m50
\plot 50.00000 1.09973 55.00000 1.06619 / \% m50m55
\plot 55.00000 1.06619 60.00000 1.03368 / \% m55m60
\plot 60.00000 1.03368 65.00000 1.00216 / \% m60m65
\plot 65.00000 1.00216 70.00000 0.97160 / \% m65m70
\plot 70.00000 0.97160 75.00000 0.94197 / \% m70m75
\plot 75.00000 0.94197 80.00000 0.91324 / \% m75m80
\plot 80.00000 0.91324 85.00000 0.88539 / \% m80m85
\plot 85.00000 0.88539 90.00000 0.85839 / \% m85m90
\plot 90.00000 0.85839 95.00000 0.83222 / \% m90m95
\%... end of file <isomdata8.dat>

\thickline%
\% inputfile(isomatdata8.dat) \% 0.8
\%... start of file <isomatdata8.dat> loop [1]
\% \% Iteration number: 1
\% \% k= mac40(iso)
\% \% point(k5){5,1.162588} \% manual \hspace{0.5cm} k5 = (5.00000, 1.16259)
\% \% point(k10){10,1.127135} \hspace{0.5cm} k10 = (10.00000, 1.12713)
\% \% point(k15){15,1.092763} \hspace{0.5cm} k15 = (15.00000, 1.09276)
\% \% point(k20){20,1.059439} \hspace{0.5cm} k20 = (20.00000, 1.05944)
\% \% point(k25){25,1.027131} \hspace{0.5cm} k25 = (25.00000, 1.02713)
\% \% point(k30){30,0.9958085} \hspace{0.5cm} k30 = (30.00000, 0.99581)
\% \% point(k35){35,0.9654412} \hspace{0.5cm} k35 = (35.00000, 0.96544)
\% \% point(k40){40,0.93600} \hspace{0.5cm} k40 = (40.00000, 0.93600)
\% \% point(k45){45,0.9074566} \hspace{0.5cm} k45 = (45.00000, 0.90746)
\% \% point(k50){50,0.8797836} \hspace{0.5cm} k50 = (50.00000, 0.87978)
\% \% point(k55){55,0.8529544} \hspace{0.5cm} k55 = (55.00000, 0.85295)
\% \% point(k60){60,0.8269435} \hspace{0.5cm} k60 = (60.00000, 0.82694)
\% \% point(k65){65,0.8017257} \hspace{0.5cm} k65 = (65.00000, 0.80173)
\% \% point(k70){70,0.7772771} \hspace{0.5cm} k70 = (70.00000, 0.77728)
\% \% point(k75){75,0.7535739} \hspace{0.5cm} k75 = (75.00000, 0.75357)
\% \% point(k80){80,0.7305936} \hspace{0.5cm} k80 = (80.00000, 0.73059)
\% \% point(k85){85,0.708314} \hspace{0.5cm} k85 = (85.00000, 0.70831)
\% \% point(k90){90,0.6867139} \hspace{0.5cm} k90 = (90.00000, 0.68671)
\% \% point(k95){95,0.6657725} \hspace{0.5cm} k95 = (95.00000, 0.66577)
\% \% drawline(k5 k10 k15 k20 k25 k30 k35 k40 k45 k50 k55 k60 k65 k70 k75 k80 k85 k90 k95)
\plot 5.00000 1.16259 10.00000 1.12713 / \% k5k10
\plot 10.00000 1.12713 15.00000 1.09276 / \% k10k15
\plot 15.00000 1.09276 20.00000 1.05944 / \% k15k20
\plot 20.00000 1.05944 25.00000 1.02713 / \% k20k25
\plot 25.00000 1.02713 30.00000 0.99581 / \% k25k30
\plot 30.00000 0.99581 35.00000 0.96544 / \% k30k35
\plot 35.00000 0.96544 40.00000 0.93600 / \% k35k40
\plot 40.00000 0.93600 45.00000 0.90746 / \% k40k45
\plot 45.00000 0.90746 50.00000 0.87978 / \% k45k50
\plot 50.00000 0.87978 55.00000 0.85295 / \% k50k55
\plot 55.00000 0.85295 60.00000 0.82694 / \% k55k60
\plot 60.00000 0.82694 65.00000 0.80173 / \% k60k65
\plot 65.00000 0.80173 70.00000 0.77728 / \% k65k70
\plot 70.00000 0.77728 75.00000 0.75357 / \% k70k75
\plot 75.00000 0.56518 80.00000 0.54795 / \%
\plot 80.00000 0.54795 85.00000 0.53124 / \%
\plot 85.00000 0.53124 90.00000 0.51504 / \%
\plot 90.00000 0.51504 95.00000 0.49933 / \%
\% ... end of file <isojdata8.dat> loop [1]
\%endSKIP

\verb|\%from mac-des.m|
\% var x=-1
\% var x2=x + 2
\% point(h)\{x2,2.55\} \%
\% put \{MAC\} at 1.000000 2.550000
\% vertical diff = 0.29 units \%
\% var d=0.29
\% h6 = 0.88
\% text(fbox{$0 \cdot 6$})\{x,h6\}
\% put \{fbox{$0 \cdot 6$}\} at -1.000000 0.880000
\% var h8=h6+d
\% h8 = 1.17
\% text(fbox{$0 \cdot 8$})\{x,h8\}
\% put \{fbox{$0 \cdot 8$}\} at -1.000000 1.170000
\% var h10=h8 + d
\% h10 = 1.46
\% text(fbox{$1 \cdot 0$})\{x,h10\}
\% put \{fbox{$1 \cdot 0$}\} at -1.000000 1.460000
\% var h12=h10 + d
\% h12 = 1.75
\% text(fbox{$1 \cdot 2$})\{x,h12\}
\% put \{fbox{$1 \cdot 2$}\} at -1.000000 1.750000
\% var h14 = h12 + d
\% h14 = 2.04
\% text(fbox{$1 \cdot 4$})\{x,h14\}
\% put \{fbox{$1 \cdot 4$}\} at -1.000000 2.040000
\% var h16=h14 + d
\% h16 = 2.33
\% text(fbox{$1 \cdot 6$})\{x,h16\}
\% put \{fbox{$1 \cdot 6$}\} at -1.000000 2.330000

\%führen new rotated legends from macATdes2.pl=================================
\% var y2=2.6
\% y2 = 2.6
\% var y1=0.4
\% y1 = 0.4

\newcommand{\ylegend}{\\sf End-tidal (\%) in 100, \% oxygen/air} \%
\% \% determine string length \% \% Yunits etc \% \%
\newlength{\ylength} \%
\settowidth{\ylength}{\ylegend} \%
\%\%text(answer ylength = \number\ylength{37,-0.4} \%
\% halflength/3.818=0.777 y units \%
\begin{align*}
\text{Age Corrected MAC} & \quad \text{RWD Nickalls} \quad 88
\end{align*}
% y04 = 1.1523
% var y02=0.9523  % y00 = 0.7523 + 0.2
% y02 = 0.9523
% var y00=0.7523  % y00 = 0.7523
% point(L16){x5, y16} L16 = (5.00000, 2.35230)
% point(R16){x6, y16} R16 = (100.00000, 2.35230)
% point(L14){x5, y14} L14 = (5.00000, 2.15230)
% point(R14){x6, y14} R14 = (100.00000, 2.15230)
% point(L12){x5, y12} L12 = (5.00000, 1.95230)
% point(R12){x6, y12} R12 = (100.00000, 1.95230)
% point(L10){x5, y10} L10 = (5.00000, 1.75230)
% point(R10){x6, y10} R10 = (100.00000, 1.75230)
% point(L08){x5, y08} L08 = (5.00000, 1.55230)
% point(R08){x6, y08} R08 = (100.00000, 1.55230)
% point(L06){x5, y06} L06 = (5.00000, 1.35230)
% point(R06){x6, y06} R06 = (100.00000, 1.35230)
% point(L04){x5, y04} L04 = (5.00000, 1.15230)
% point(R04){x6, y04} R04 = (100.00000, 1.15230)
% point(L02){x5, y02} L02 = (5.000000, 0.95230)
% point(R02){x6, y02} R02 = (100.00000, 0.95230)
% point(L00){x5, y00} L00 = (5.000000, 0.75230)
% point(R00){x6, y00} R00 = (100.00000, 0.75230)
% draw the dashes from Left to Right
% (so have small gap at right axis)
% drawline(L16R16, L14R14, L12R12, L10R10, L08R08, L06R06, L04R04, L02R02,
% L00R00)
\putrule from 5.00000 2.35230 to 100.00000 2.35230 % L16R16
\putrule from 5.00000 2.15230 to 100.00000 2.15230 % L14R14
\putrule from 5.00000 1.95230 to 100.00000 1.95230 % L12R12
\putrule from 5.00000 1.75230 to 100.00000 1.75230 % L10R10
\putrule from 5.00000 1.55230 to 100.00000 1.55230 % L08R08
\putrule from 5.00000 1.35230 to 100.00000 1.35230 % L06R06
\putrule from 5.00000 1.15230 to 100.00000 1.15230 % L04R04
\putrule from 5.00000 0.95230 to 100.00000 0.95230 % L02R02
\putrule from 5.00000 0.75230 to 100.00000 0.75230 % L00R00
\endpicture
%} %framebox
\end{document}

6.4 References

- Eger EI (2001). Age, minimum alveolar anesthetic concentration, and minimum alveolar anesthetic concentration-awake. Anesthesia and Analgesia; 93, 947–953. [has an appendix on temperature correction]
Figure 6.3: The isoflurane version (mac-iso8t.m) generated for the *Oxford Handbook of Anaesthesia* with rotated LHS-axis legend.

- Nickalls RWD (2000). mathsPIC DOS 2·1 (http://www.tex.ac.uk/tex-archive/graphics/mathspic/dos/)


  [A new bug-fix version: February 2007]

Chapter 7

Age & temperature corrected MAC-charts

7.1 Introduction

The age & temperature corrected MAC charts (see Figure below) enable anaesthetists to readily determine the appropriate End-tidal (ET) concentration of the commonly used volatile inhalational anaesthetic agents (isoflurane, sevoflurane, desflurane) when used either with or without nitrous-oxide. The temperature-correction data is derived from Eger (2001), and the age-correction data from Mapleson (1996) and Nickalls & Mapleson (2003).

Figure 7.1:
View of a typical age & temperature corrected MAC chart for desflurane, generated by the Perl program macatdes2.pl
It is envisaged that such charts would be printed out on-demand by anaesthetists in the operating theatre by selecting the patient age from a simple menu. This chart could easily be included in the patient notes along with the rest of the electronic anaesthesia record. Alternatively (and far better) the corrected MAC can be displayed in real-time on a computer.

It is the purpose of this note to detail the programs and methodology for generating these charts. The primary generating programs, which are written in Perl (version 5), generate as output a \textsf{mathsPIC} script (for drawing the graph). This script is then run through \textsf{mathsPIC} itself (Nickalls 1999, Syropoulos and Nickalls 2000, Syropoulos and Nickalls 2005).

7.2 Program: \texttt{macatdes2.pl}

This Perl program asks the user to select the age of the patient, and then generates the desflurane chart for the given age.

```perl
#!/usr/bin/perl
# LINUX version
# Desflurane
# macATdes2.pl (from macATdes1.pl)
# to fix the decimal problem
# Dick Nickalls
# May 6,2005
#

use Math::Trig;

## program name is $O (see black book p 211
$timenowgmt = localtime;
##===============================
## from Programming Perl p 191
$sign='[-+]?';
$digits = '\d+';
$decimal = '\.?';
$more_digits = '\d*';
$number = "$sign$digits$decimal$more_digits";## works OK for here but rejects .2
```

```
## Number" - (?\(d+(?:\(d+)?(?:\(d+))")); # page 190 does not work OK
## Number" = "(?-1)?(?:\(d\(d+)(?:(Ee[+-]1)?(?:\(d+))")")); # page 190 does not work OK

```perl
print " enter AGE yrs  [q to quit]: ";
$age = <STDIN>;
chomp $age;
$age =~ s/\^\+//, $age =~ s/\-/\+/

if (lc($age) eq "q") {
  print " quitting program now....\n\n";
  exit
}
else {
  if ($age =~ /\^\+$/o) {}
  else {
    print " ERROR: value must be a decimal number (1−120)\n";
    print " -- must be in range 1−120 \n";
    print " -- quitting program now\n\n";
    exit
  }

## final check that all input parameters were OK -- else quit at this stage
if ($age >= 1 and $age <= 120) { }
else {
  print " ERROR: entered inappropriate age ($age)\n";
  print " -- must be in range 1−120 \n";
  print " -- quitting program now\n\n";
  exit
}

## use DVI filename to show that it has been printed on the fly
$filename = $dvifilename;
```

## do some initial calculation of some required variables
## FIRST calculate the slopes of all lines for mAC1.6

```perl
$pi = 4 * atan2(1,1);
$s1634 = etdes(1.6, 34);
$s1640 = etdes(1.6, 40);
$s16deg = $r2d * atan(($s1640 - $s1634) * 5 / (6 * 2));
$s1434 = etdes(1.4, 34);
$s1440 = etdes(1.4, 40);
$s14deg = $r2d * atan(($s1440 - $s1434) * 5 / (6 * 2));
```
$s1234=etdes(1.2, 34);
$s1240=etdes(1.2, 40);
$s12deg = \text{r2d} \times \text{atan}((s1240 - s1234) \times 5/(6 \times 2));
$s1034=etdes(1.0, 34);
$s1040=etdes(1.0, 40);
$s10deg = \text{r2d} \times \text{atan}((s1040 - s1034) \times 5/(6 \times 2));
$s0834=etdes(0.8, 34);
$s0840=etdes(0.8, 40);
$s08deg = \text{r2d} \times \text{atan}((s0840 - s0834) \times 5/(6 \times 2));
$s0634=etdes(0.6, 34);
$s0640=etdes(0.6, 40);
$s06deg = \text{r2d} \times \text{atan}((s0640 - s0634) \times 5/(6 \times 2));

print "etdes1634 = $s1634\n";
print "etdes1640 = $s1640\n";

print "slope s16Deg = $s16deg\n";
print "slope s14Deg = $s14deg\n";
print "slope s12Deg = $s12deg\n";
print "slope s10Deg = $s10deg\n";
print "slope s08Deg = $s08deg\n";

## now to do 67% axis scale
## $age = 12;
## $m40=6.6;
## $agecorr=(10^{*}(-0.00269*(\text{age}-40)));
## first need to know what Y2 is
$k2=(1.6*\text{m40}*\text{agecorr}(1 - 0.05*(37 - 40)));
# increment scales by 0.5%
$j2=int((k2+0.25)/0.5);
$y2=(j2*0.5) +0.5;
#----------------

print "k2= $k2\n";
print "j2= $j2\n";
print "y2= $y2\n";

## for left scale 100% o2
## $k1 is ycoord of KHS of isoMAC 0.6 line
$sk1=(0.6*\text{m40}*\text{agecorr}(1 - 0.05*(37 - 34)));
$sj1=int((sk1-0.25)/0.5);
$sy1=(sj1*0.5)-0.5;
$k=int($y2/0.5);
# make string A
$\text{string100a} = " \backslash \text{ ticks with values } \"$

$\text{string100b} = " \backslash \text{ at } \"$

for ($j = \$y1; j < \$y2 + 0.15; j = j + 0.5) {
## problem with j < \$y2 looses last cycle
## hence use fix j \leq \$y2 + 0.15
print "j = \$j \n";
## format values to 1 decimal place
$\text{vv} = \text{sprintf } "%.1f", \$j;
if ($\text{vv} == "0.0") { \$vv = "0" };
$\text{string100a} = $\text{string100a} . " \$vv \"$
$\text{string100b} = $\text{string100b} . \$j \"$
};
$\text{string100a} = $\text{string100a} . " \slash \"
$\text{string100b} = $\text{string100b} . " / / \"

####
#### desflurane
#### for scale 67% get offset from file mac\_des7.m
$\text{offset67} = 4.217;
$\text{y67} = \$y2 - \text{offset67};
$\text{k} = \text{int}(\text{y67}/0.5);
# make string A
$\text{string67a} = " \backslash \text{ ticks with values } \"
$\text{string67b} = " \backslash \text{ at } \"
$\text{value67a} = $j * 0.5;
$\text{value67b} = \text{value67b} + 0.5;
if ($j eq 0) { \text{value67b} = \text{offset67} };
## format values to 1 decimal place
$\text{vv} = \text{sprintf } "%.1f", \text{value67a};
if ($\text{vv} == "0.0") { \$vv = "0" };
$\text{string67a} = $\text{string67a} . " \$vv \"
$\text{string67b} = $\text{string67b} . \$value67b \"$
};
$\text{string67a} = $\text{string67a} . " \slash \"
$\text{string67b} = $\text{string67b} . " / / \"

###
### for scale 50% get offset from file mac\_des7.m
$\text{offset50} = 3.147;
$\text{y50} = \$y2 - \text{offset50};
$\text{k} = \text{int}(\text{y50}/0.5);
# make strings A and B
$\text{string50a} = " \backslash \text{ ticks with values } \"
$\text{string50b} = " \backslash \text{ at } \"
$\text{value50a} = $j * 0.5;
$\text{value50b} = \text{value50b} + 0.5;
if ($j eq 0) { \text{value50b} = \text{offset50} };
## format values to 1 decimal place
$\text{vv} = \text{sprintf } "%.1f", \text{value50a};
if ($\text{vv} == "0.0") { \$vv = "0" };

CHAPTER 7. AGE & TEMPERATURE CORRECTED MAC-CHARTS  RWD Nickalls

\begin{verbatim}
199    $string50a=$string50a." \$vv\$ ";
200    $string50b=$string50b." $value50b ";
201    
202    $string50a=$string50a." /";
203    $string50b=$string50b." / /";
204    #==========================================================
205    #==========================================================
206    print (outfile " EOF);
207    \%\% mathsPIC
208    \% macDes1.m \% for Desflurane
209    \% to create temp and age MAC chart corrections
210    \% April 13, 2005
211    \%-----------------------------------------------
212    \{documentclass[a4paper]{article}
213    \usepackage{mathspic,color}
214    \usepackage{decimal,rotating}
215    \setlength{\textwidth}{30cm}
216    \setlength{\oddsidemargin}{0cm}
217    \begin{document}
218    \begin{picture}
219    \%-----------------------------------------------
220    \% DES MAC40 = 6.6
221    \% agent specific adjustments
222    \% need to adjust Xunits, Yunits, MAC, zero offset for N2O 50% and 67%
223    \% the Ytick interval (0.5 for des),
224    \% as well as the space above and below the line space
225    \%-----------------------------------------------
226    \%def age(\%age\% AGE
227    var m40=6.6\%desflurane MAC40
228    \%def agecorr(10**(-0.00269*(\%age-40)))\% agecorr
229    \%def etdes(t(m=m40+&agecorr*(1 - 0.05*(37 - t)))/% etdes
230    \%-----------------------------------------------
231    \% determine the X and Y axes lengths (age dependent)
232    \% want to make the Y axis just longer than the line space
233    \% for the ticks to work nicely we want left Yaxis to be an
234    \% exact multiple of tick interval
235    \% therefore we add some extra and adjust with 0.2 increments for sevo
236    \% top right - use t=40 and MAC=1.6
237    var k2=(1.6+m40+&agecorr*(1 - 0.05*(37 - 40))), j2=int((k2+0.25)/0.5),
238    var y2=(2 +0.5) +0.5,%
239    \% bottom left - use t=34 and MAC=0.6
240    var k1=(0.6+m40+&agecorr*(1 - 0.05*(37 - 34))), j1=int((k1-0.25)/0.5),
241    var y1=(j1+0.5) -0.5,%
242    \% now do the paper command (only for axes(B) here (as no decimals)--
243    \% paper(units(2cm,1.3cm), xrange(34,40), yrange(y1,y2), axes(B), ticks(1,0.5)) \%
244    \% left axis 100\% oxygen
245    \% left axis \% left axis for 100\% oxygen
246    $string100a
247    $string100b
248    \%-----------------------------------------------modified from
249    \% mac--des7.m \%-----------------------------------------------
250    \% offset for 67\% N2O scale
251    $\%o
\end{verbatim}
\begin{verbatim}
\%-- - - - - - - - - modified from mac\-des7.m-- - - - - - - - - - - - -
\% extra right axis 50\% N2O
\% as axis off graph then need new paper command
\%\% paper\{units(cm,3.5294) xrange(-5,117) yrange(0.8653, 3.465) \}
\% paper\{units(cm,3.5294) xrange(34,40.75) yrange(3.142, 4.16) \}
\%\% use offset 50 \% second right axis for 50\% oxygen Offset = 3.142
\% initial conditions
\linethickness(1pt)\%
\drawline(T137 B137)\%
% vertical \text{T}=37 line
% make top end half way between top and MAC1.6 line
var y22= y2 - (y2 - k2)/2
point(T137)\{37,y22\}
point(B137)\{37,y1\}
\drawline(T137 B137)\%
% initial conditions
%\text{Age} = jj
% define the polar (rr,tt) for the \text{turnbox} \{}
var r9=0.2
var t9=40 \%
% Temperature $^\circ\text{C}$
% Temperature corrected iso MAC chart
% define the polar (rr,tt) for the \text{turnbox} \{}
\begin{verbatim}
Eger EI (2001). Anesthesiology, \underline{93}, 947--953
Nickalls RWD and Mapleson WW (2003). Br. J. Anaesthesia,
\underline{91}, 170 -- 174
text(\{sf Eger EI (2001)\})\%
\end{verbatim}
\end{verbatim}
CHAPTER 7. AGE & TEMPERATURE CORRECTED MAC-CHARTS RWD Nickalls

```latex
\begin{verbatim}
\input{code}
\end{verbatim}
```
464  $t = $array[1];  # temp
465  ## use 1.8 instead of m40 for now – fix later
466  $etdes = ($mac * 1.8 * $agecorr * (1 - 0.05 * (37 - $t)));
467  return $etdes;
468
469  _END_;

7.3 Program: macatdes2.m

This mathsPIC program is output by the above program (macatdes2.pl) Perl, and it this program which draws the chart.
CHAPTER 7. AGE & TEMPERATURE CORRECTED MAC-CHARTS  RWD Nickalls

\begin{center}
% axis left  \% left axis for 100% oxygen
% ticks with values $2.5\% \ 3.0\% \ 3.5\% \ 4.0\% \ 4.5\% \ 5.0\% \ 5.5\% \ 6.0\% \ 6.5\% \ 7.0\%$
% $7.5\% \ 8.0\% \ 8.5\% \ 9.0\% \ 9.5\% \ 10.0\% \ 10.5\% \ 11.0\% \ 11.5\% \ 12.0\% \ 12.5\%$
% $\%
% % modified from mac-des7.m %
\end{center}

\begin{center}
% axis right \% 67% N2O scale
% offset for 67% = 4.22
% ticks with values $0\% \ 0.5\% \ 1.0\% \ 1.5\% \ 2.0\% \ 2.5\% \ 3.0\% \ 3.5\% \ 4.0\% \ 4.5\%$
% $5.0\% \ 5.5\% \ 6.0\% \ 6.5\% \ 7.0\% \ 7.5\% \ 8.0\% \ 8.5\% \ 9.0\% \ 9.5\% \ 10.0\% \ 10.5\% \ 11.0\% \ 11.5\% \ 12.0\% \ 12.5\% \ 13.0\%
% % modified from mac-des7.m
\end{center}

% extra right axis 50% N2O
% as axis off graph then need new paper command
% paper(units(mm,3.5294cm) xrange(-5,117) yrange(0.8653, 3.465) )
% paper(units(2cm,1.3cm) xrange(34,40.75) yrange(3.142, y2))
% % offset 50%
% ticks with values $0\% \ 0.5\% \ 1.0\% \ 1.5\% \ 2.0\% \ 2.5\% \ 3.0\% \ 3.5\% \ 4.0\% \ 4.5\%$
% $5.0\% \ 5.5\% \ 6.0\% \ 6.5\% \ 7.0\% \ 7.5\% \ 8.0\% \ 8.5\% \ 9.0\% \ 9.5\% \ 10.0\% \ 10.5\% \ 11.0\% \ 11.5\% \ 12.0\% \ 12.5\% \ 13.0
% % modified from mac-des7.m
\end{center}

% initial conditions
line(thickness(1pt))
% % draw vertical T37 line
% make top end half way between top and MAC1.6 line
var y22 = y2 - (y2 - k2)/2
point(T137){37,y22}
point(B137){37,y1}
drawline(T137 B137) % vertical T=37 line
% % define the polar (rr,tt)
% for the urnbox { }
var r9=0.2
var t9 = 40 \ degrees
% % Temperature $\hat{\circ}C$
% Large Age \& temperature corrected iso–MAC chart
\end{center}
\section*{Conclusion}

In summary, the study aimed to compare the effects of age and temperature on MAC values and tidal volumes during desflurane anesthesia. The results showed that:

-There was a significant difference in MAC values between different age groups, with older patients requiring higher concentrations of desflurane.
-There was also a significant difference in tidal volumes, with younger patients having higher tidal volumes.

These findings have implications for the clinical management of patients undergoing desflurane anesthesia, particularly in regards to adjusting the anesthetic protocol based on age and temperature. Further research is needed to confirm these findings and to explore the underlying mechanisms.

\textbf{Future directions:}

The study highlights the importance of individualizing anesthesia protocols based on patient age and temperature. Future research could focus on:

-Developing more accurate models for predicting MAC values and tidal volumes in different age groups.
-Evaluating the impact of temperature on MAC and tidal volume during desflurane anesthesia in various clinical settings.

These studies could help improve the safety and efficacy of desflurane anesthesia in clinical practice.
```
point(B39) {39, etdes(39)}
point(B40) {39.75, etdes(39.75)}
\color{blue}
drawline(B34 B37)
\color{red}
drawline(B37 B40)
\color{black}
drawpoint(B35 B36 B37 B38 B39)%
text(\turnbox{18.1039569311426}}{{\boldmath$1.4$}}{B37, polar(r9, (18.1039569311426 + t9)deg)}
\%

\var m=1.2 \% MAC

point(C34) {34.25, etdes(34.25)}
point(C35) {35, etdes(35)}
point(C36) {36, etdes(36)}
point(C37) {37, etdes(37)}
point(C38) {38, etdes(38)}
point(C39) {39, etdes(39)}
point(C40) {39.75, etdes(39.75)}
\color{blue}
drawline(C34 C37)
\color{red}
drawline(C37 C40)
\color{black}
drawpoint(C35 C36 C37 C38 C39)
text(\turnbox{15.654093594647}}{{\boldmath$1.2$}}{C37, polar(r9, (15.654093594647 + t9)deg)}
\%

\var m=1.0 \% MAC

point(D34) {34.25, etdes(34.25)}
point(D35) {35, etdes(35)}
point(D36) {36, etdes(36)}
point(D37) {37, etdes(37)}
point(D38) {38, etdes(38)}
point(D39) {39, etdes(39)}
point(D40) {39.75, etdes(39.75)}
\color{blue}
drawline(D34 D37)
\color{red}
drawline(D37 D40)
\color{black}
drawpoint(D35 D36 D37 D38 D39)
text(\turnbox{13.1441191824082}}{{\boldmath$1.0$}}{D37, polar(r9, (13.1441191824082 + t9)deg)}
\%

\var r14 = \atan((5/2) + \tan(s))
% get slope of line
\var r14 = r14 * \pi/180
\var d14=r14 + &r2d
\%

\var m=0.8 \% MAC

point(E34) {34.25, etdes(34.25)}
point(E35) {35, etdes(35)}
point(E36) {36, etdes(36)}
```
\begin{center}

\textbf{CHAPTER 7. AGE & TEMPERATURE CORRECTED MAC-CHARTS}

\textbf{RWD Nickalls}

\end{center}

\begin{verbatim}
point(E37) {37, etdes(37)}
point(E38) {38, etdes(38)}
point(E39) {39, etdes(39)}
point(E40) {39.75, etdes(39.75)}
\color{blue}
drawline(E34 E37)
\color{red}
drawline(E37 E40)
\color{black}
drawpoint(E35 E36 E37 E38 E39)
text('\text{\textbackslash turnbox\{10.5817540253587\} \{\text{\textbackslash boldmath}$0.8$\}\{E37, polar(r9,10.5817540253587 + t9)deg}\)\}\\%

var \textbf{m} = 0.6 \ % MAC
point(F34) {34.25, etdes(34.25)}
point(F35) {35, etdes(35)}
point(F36) {36, etdes(36)}
point(F37) {37, etdes(37)}
point(F38) {38, etdes(38)}
point(F39) {39, etdes(39)}
point(F40) {39.75, etdes(39.75)}
\color{blue}
drawline(F34 F37)
\color{red}
drawline(F37 F40)
\color{black}
drawpoint(F35 F36 F37 F38 F39)
text('\text{\textbackslash turnbox\{7.97587571582933\} \{\text{\textbackslash boldmath}$0.6$\}\{F37, polar(r9, 7.97587571582933 + t9)deg\}\)\}\\%

\% now draw all the horizontal dashed lines
\setdashes
linethickness(0.4pt)
var w1 = 34.1 \ % left end of dashed lines
var w2 = 39.85 \ % right end of dashed lines
point(L16) {w1, ycoord(A37)}
point(R16) {w2, ycoord(A37)}
point(L14) {w1, ycoord(B37)}
point(R14) {w2, ycoord(B37)}
point(L12) {w1, ycoord(C37)}
point(R12) {w2, ycoord(C37)}
point(L10) {w1, ycoord(D37)}
point(R10) {w2, ycoord(D37)}
point(L08) {w1, ycoord(E37)}
point(R08) {w2, ycoord(E37)}
point(L06) {w1, ycoord(F37)}
point(R06) {w2, ycoord(F37)}
drawline(L16 R16, L14 R14, L12 R12, L10 R10, L08 R08, L06 R06)
linethickness(1pt)
\setsolid \%

\%----------------------------------
\%----------------------------------
\normalcolor
\end{picture}
\end{document}
\end{verbatim}
7.4 \TeX{} file \texttt{macdes034.mt}

The following \TeX{} file, generated by the above mathsPIC script, is processed by \texttt{pdflatex} to generate the final pdf version for printing.

\begin{verbatim}
\documentclass[a4paper]{article}
\usepackage{mathspic, color}
\usepackage{decimal, rotating}
\setlength{\textheight}{30cm}
\setlength{\oddsidemargin}{0cm}
\begin{document}
\begin{picture}
% define age()34 % AGE
% var m40=6.6%desflurane MAC40
% m40 = 6.6
% def agecorr()(10**(-0.00269*(\&age-40)))% agecorr
% def etdes(t)(m*m40*\&agecorr*(1 - 0.05*(37 -t)))% etdes
%------------------------------------
% determine the X and Y axes lengths (age dependent)
% want to make the Y axis just longer than the line space
% for the ticks to work nicely we want left Yaxis to be an
% exact multiple of tick interval
% therefore we add some extra and adjust with 0.2 increments for sevo
% top right - use t=40 and MAC=1.6
% var k2=(1.6*m40*10**(-0.00269*(34-40)))*(1 - 0.05*(37 -40)),
% j2=int((k2+0.25)/0.5)  %
% k2 = 12.6038074143575
% j2 = 25
% var y2=(j2 *0.5) +0.5
% y2 = 13
% bottom left - use t=34 and MAC=0.6
% var k1=(0.6*m40*10**(-0.00269*(34-40)))*(1 - 0.05*(37 -34)),
% j1=int((k1-0.25)/0.5)  
% k1 = 1.70414149914286
% j1 = 1
% etica = 10**(-0.00269*(34-40))
% etica = 0.598672040831748
% (37 -t))
% j2=int((k2+0.25)/0.5)  
% k2 = 12.6038074143575
% j2 = 25
% var y2=(j2 *0.5) +0.5
% y2 = 13
% bottom left - use t=34 and MAC=0.6
% var k1=(0.6*m40*10**(-0.00269*(34-40)))*(1 - 0.05*(37 -34)),
% j1=int((k1-0.25)/0.5)  
\end{picture}
\end{document}
\end{verbatim}
% k1 = 3.49344662028388
% j1 = 6
% var y1=(j1*0.5)-0.5
% y1 = 2.5
%now do the paper command (only for azes(B) here (as no decimals))---
% paper(units(2cm,1.3cm), xrange(34,40), yrange(y1,y2), azes(B),
ticks(1,0.5)) %
\setcoordinatesystem units <2cm,1.3cm>
\setplotarea x from 34.00000 to 40.00000, y from 2.50000 to 13.00000
\axis bottom ticks numbered from 34 to 40 by 1 /
%------now do LEFT axes ticks etc------
% left left % left axis 100% oxygen
\axis left % left left axis for 100% oxygen
ticks withvalues $2.5$ $3.0$ $3.5$ $4.0$ $4.5$ $5.0$ $5.5$
$6.0$ $6.5$ $7.0$ $7.5$ $8.0$ $8.5$ $9.0$ $9.5$ $10.0$
$10.5$ $11.0$ $11.5$ $12$ $12.5$ $13$ / 
at 2.5 3 3.5 4 4.5 5 5.5 6 6.5 7 7.5 8 8.5 9 9.5 10
10.5 11.5 12 12.5 13 / /

%--------modified from mac-des7.m------------------------
\axis right % 67% N2O scale
% offset for 67% = 4.22
ticks withvalues $0$ $0.5$ $1.0$ $1.5$ $2.0$ $2.5$ $3.0$
$3.5$ $4.0$ $4.5$ $5.0$ $5.5$ $6.0$ $6.5$ $7.0$ $7.5$
$8.0$ $8.5$ / 
12.717 / /
%----------modified from mac-des7.m---------------------
%% extra right axis 50% N2O
% as axis off graph then need new paper command
% paper(units(mm,3.5294cm) xrange(-5,117) yrange(0.8653, 3.465) )
% paper(units(2cm,1.3cm) xrange(34,40.75) yrange(3.142, y2))%% use
offset 50%
\setcoordinatesystem units <2cm,1.3cm>
\setplotarea x from 34.00000 to 40.75000, y from 3.14200 to 13.00000
\axis right % second right axis for 50% oxygen Offset = 3.142
ticks withvalues $0$ $0.5$ $1.0$ $1.5$ $2.0$ $2.5$ $3.0$
$3.5$ $4.0$ $4.5$ $5.0$ $5.5$ $6.0$ $6.5$ $7.0$ $7.5$
$8.0$ $8.5$ $9.0$ $9.5$ / 
at 3.147 3.647 4.147 4.647 5.147 5.647 6.147 6.647
11.147 11.647 / /
%--------------------------------
%-------------------------------
%% initial conditions
% linethickness(1pt)%
\linethickness=1.00000pt\Linethickness{1.00000pt} %
\font\CM=cmr10 at 9.97226pt%\font\CM=cmr10 %
\setplotsymbol ({\CM .})%
%----
%% draw vertical T37 line
%% make top end half way between top and MAC1.6 line
% var y22= y2 - (y2-k2)/2
% y22 = 12.8019037071788
% point(T137){37,y22} T137 = (37.00000, 12.80190)
% point(B137){37,y1} B137 = (37.00000, 2.50000)
% drawline(T137 B137) % vertical T=37 line
\putrule from 37.00000 12.80190 to 37.00000 2.50000 % T137B137
%------------
% point(Q1){34,y2} % top left corner Q1 = (34.00000, 13.00000)
% test({\sf\Huge DESFLURANE}{37, y2 + 0.7} % text at 37.00000 13.700000
% text\{
% \sf \Large AGE = jj\}
% at Q1, shift(0.2, -0.7) % left
% text\{
% \sf \Large AGE = 34\}
% at Q1, shift(0.2, -0.7) % left
% \put \{
% \sf \Large AGE = 34\}
% at 34.200000 12.300000 % left
% point(Q2){40,y1} % bottom right corner Q2 = (40.00000, 2.50000)
% text\{
% \sf \copyright\ RWD Nickalls 2005\}
% at Q2, shift(-0.2, 0.35) % right
% text\{
% \sf \copyright\ RWD Nickalls 2005\}
% at Q2, shift(-0.2, 0.7) % right
% text\{
% \sf macdes034.dvi\}
% at Q3, shift(0, 0.35) % left
% \put \{
% \sf macdes034.dvi\}
% at 35.000000 2.850000 % left
%------------------
% define the polar (rr,tt) for the \urnbox{}{} MAC labels
% var r9=0.2 %
% var r9 = 0.2
% var t9=40\% degrees %
% t9 = 40
%------------
% text\{
% \sf Temperature $^{\circ}$C\}
% at 37.000000 1.800000
% text\{
% \sf \Large Age \& temperature corrected iso-MAC chart\}
% at 37.000000 1.100000
%---------refs-------
% newcommand\{
% \refs\}{
% \begin{minipage}{\textwidth}
% \sf Eger EI (2001). Anesthesiology, \underline{93}, 947--953
% \newline\sf Nickalls RWD and Mapleson WW (2003). Br. J. Anaesthesia,
% \underline{91}, 170--174
% \end{minipage} %
% \\% text\{
% \refs\}\{34, y1 - 2.35\}[1] %
% \put \{
% \refs\}\{1\} at 34.000000 0.150000 %
% \---determine string length \longrightarrow \Yunits etc------
% \\% newcommand\{
% \ylegend\}{
% \sf End-tidal (\% Desflurane in 100\%, \% oxygen/air)\%
% \% \setlength\{\ylegend\}{1.5}\Yunits
% \% text\{
% answer = \number\{\ylegend\}\{37,-1\}\%
% \% half\length/1.5=2.2223 y units
% \% text\{
% \turnbox{90}{\ylegend}\}\{33.3, yf=((y2-y1)/2) - 2.2223\}%
% \put \{
% \turnbox{90}{\ylegend}\}\{33.300000 5.527700 %
% \---
% labels at top
%% test({\sf N}_2O)\{40.25, y2 + 0.7\}
\put {\sf N}_2O at 40.250000 13.700000
%% test({\sf 67\,%})\{40.25, y2 + 0.7 - 0.35\}
\put {\sf 67\,%} at 40.250000 13.350000
%% test({\sf 50\,%})\{40.98, y2 + 0.7\}
\put {\sf 50\,%} at 40.980000 13.700000
%--------------
\newcommand{\rightylegend}{\sf End-tidal (\%) Desflurane in N}_2O% 
%% text(\turnbox{270}{\rightylegend})\{41.4, y1+(y2-y1)/2 + 1.66\}
\put {\turnbox{270}{\rightylegend}} at 41.400000 9.410000
% MACage label
%def macage(jj)test(\sf large MAC$_{jj}$)\{A37, shift(0.12,0.8)\}[l]
% test(\sf large MAC$_{34}$)\{A37, shift(0.12,0.8)\}[l]
\put {\sf large MAC$_{34}$} [l] at 37.120000 11.759830
%------------
\%
\text{var m=1.4 \%MAC}
\%
\text{m = 1.4}
\%
\text{point(B34)\{34.25,(m*m40*(10**(-0.00269*(34-40))))*(1 - 0.05*(37 -34.25))\}) \quad B34 = (34.25000, 8.27125)
\%
\text{point(B35)\{35,(m*m40*(10**(-0.00269*(34-40))))*(1 - 0.05*(37 -35))\}) \quad B35 = (35.00000, 8.63087)
\%
\text{point(B36)\{36,(m*m40*(10**(-0.00269*(34-40))))*(1 - 0.05*(37 -36))\}) \quad B36 = (36.00000, 9.11036)
\%
\text{point(B37)\{37,(m*m40*(10**(-0.00269*(34-40))))*(1 - 0.05*(37 -37))\}) \quad B37 = (37.00000, 9.58985)
\%
\text{point(B38)\{38,(m*m40*(10**(-0.00269*(34-40))))*(1 - 0.05*(37 -38))\}) \quad B38 = (38.00000, 10.06935)
\%
\text{point(B39)\{39,(m*m40*(10**(-0.00269*(34-40))))*(1 - 0.05*(37 -39))\}) \quad B39 = (39.00000, 10.54884)
\%
\text{point(B40)\{39.75,(m*m40*(10**(-0.00269*(34-40))))*(1 - 0.05*(37 -39.75))\}) \quad B40 = (39.75000, 10.90846)

\\%
\text{\textcolor{blue}{drawline(B34 B37)}}
\%
\text{\plot 34.25000 8.27125 37.00000 9.58985 / \% B34B37}
\%
\text{\color{red}{drawline(B37 B40)}}
\%
\text{\plot 37.00000 9.58985 39.75000 10.90846 / \% B37B40}
\%
\text{\color{black}{drawpoint(B35 B36 B37 B38 B39)}}
\%
\text{\put \{\textbullet\} at 35.00000 8.63087 \% B35}
\%
\text{\put \{\textbullet\} at 36.00000 9.11036 \% B36}
\%
\text{\put \{\textbullet\} at 37.00000 9.58985 \% B37}
\%
\text{\put \{\textbullet\} at 38.00000 10.06935 \% B38}
\%
\text{\put \{\textbullet\} at 39.00000 10.54884 \% B39}
\%
\text{\textcolor{blue}{text(\textbf{18.1039569311426})\{\textcolor{boldmath}{\textbf{1.4}}\}} at 37.105676 9.759652}
\%
\text{\textcolor{red}{text(\textbf{18.1039569311426})\{\textbf{1.4}\}} at 37.105676 9.759652}

\%
\text{\textcolor{blue}{var m=1.2 \%MAC}}
\%
\text{\textcolor{red}{m = 1.2}}
\%
\text{\textcolor{blue}{point(C34)\{34.25,(m*m40*(10**(-0.00269*(34-40))))*(1 - 0.05*(37 -34.25))\}) \quad C34 = (34.25000, 7.08964)}
\%
\text{\textcolor{blue}{point(C35)\{35,(m*m40*(10**(-0.00269*(34-40))))*(1 - 0.05*(37 -35))\}) \quad C35 = (35.00000, 7.39799)}
\%
\text{\textcolor{blue}{point(C36)\{36,(m*m40*(10**(-0.00269*(34-40))))*(1 - 0.05*(37 -36))\}) \quad C36 = (36.00000, 7.80888)}
\%
\text{\textcolor{blue}{point(C37)\{37,(m*m40*(10**(-0.00269*(34-40))))*(1 - 0.05*(37 -37))\}) \quad C37 = (37.00000, 8.21987)}
\%
\text{\textcolor{blue}{point(C38)\{38,(m*m40*(10**(-0.00269*(34-40))))*(1 - 0.05*(37 -38))\}) \quad C38 = (38.00000, 8.63087)}
\%
\text{\textcolor{blue}{point(C39)\{39,(m*m40*(10**(-0.00269*(34-40))))*(1 - 0.05*(37 -39))\}) \quad C39 = (39.00000, 9.04186)}
\%
\text{\textcolor{blue}{point(C40)\{39.75,(m*m40*(10**(-0.00269*(34-40))))*(1 - 0.05*(37 -39.75))\}) \quad C40 = (39.75000, 9.35011)}

\color{blue}
\%
\text{\textcolor{blue}{drawline(C34 C37)}}
\%
\text{\plot 34.25000 7.08964 37.00000 8.21987 / \% C34C37}
\%
\text{\color{red}{drawline(C37 C40)}}

\plot 37.00000 8.21987 39.75000 9.35011 / \%
\color{black}
\%
\drawpoint(C35 C36 C37 C38 C39 )
\put {$\bullet$} at 35.00000 7.39789 \%
\put {$\bullet$} at 36.00000 7.80888 \%
\put {$\bullet$} at 37.00000 8.21987 \%
\put {$\bullet$} at 38.00000 8.63087 \%
\put {$\bullet$} at 39.00000 9.04186 \%
\%
text(\turnbox{15.6540933594647}{\boldmath $1.2$}){C37, polar(r9,(15.6540933594647 + t9)deg)}
\put {\turnbox{15.6540933594647}{\boldmath$1.2$}} at 37.112838 8.384999
\
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\[ m = 0.8 \]
\[ \text{point}\(E34\)\{34.25,(m*m40*(10**(-0.00269*(34-40)))*(1 - 0.05*(37 -34.25)))\} \quad E34 = (34.25000, 4.72643) \]
\[ \text{point}\(E35\)\{35,(m*m40*(10**(-0.00269*(34-40)))*(1 - 0.05*(37 -35)))\} \quad E35 = (35.00000, 4.93192) \]
\[ \text{point}\(E36\)\{36,(m*m40*(10**(-0.00269*(34-40)))*(1 - 0.05*(37 -36)))\} \quad E36 = (36.00000, 5.05292) \]
\[ \text{point}\(E37\)\{37,(m*m40*(10**(-0.00269*(34-40)))*(1 - 0.05*(37 -37)))\} \quad E37 = (37.00000, 5.17992) \]
\[ \text{point}\(E38\)\{38,(m*m40*(10**(-0.00269*(34-40)))*(1 - 0.05*(37 -38)))\} \quad E38 = (38.00000, 5.36511) \]
\[ \text{point}\(E39\)\{39,(m*m40*(10**(-0.00269*(34-40)))*(1 - 0.05*(37 -39)))\} \quad E39 = (39.00000, 5.62791) \]
\[ \text{point}\(E40\)\{39.75,(m*m40*(10**(-0.00269*(34-40)))*(1 - 0.05*(37 -39.75)))\} \quad E40 = (39.75000, 6.23340) \]

\textbf{MAC}

\[ m = 0.6 \]
\[ \text{point}\(F34\)\{34.25,(m*m40*(10**(-0.00269*(34-40)))*(1 - 0.05*(37 -34.25)))\} \quad F34 = (34.25000, 3.54482) \]
\[ \text{point}\(F35\)\{35,(m*m40*(10**(-0.00269*(34-40)))*(1 - 0.05*(37 -35)))\} \quad F35 = (35.00000, 3.69894) \]
\[ \text{point}\(F36\)\{36,(m*m40*(10**(-0.00269*(34-40)))*(1 - 0.05*(37 -36)))\} \quad F36 = (36.00000, 3.89044) \]
\[ \text{point}\(F37\)\{37,(m*m40*(10**(-0.00269*(34-40)))*(1 - 0.05*(37 -37)))\} \quad F37 = (37.00000, 4.10994) \]
\[ \text{point}\(F38\)\{38,(m*m40*(10**(-0.00269*(34-40)))*(1 - 0.05*(37 -38)))\} \quad F38 = (38.00000, 4.31543) \]
\[ \text{point}\(F39\)\{39,(m*m40*(10**(-0.00269*(34-40)))*(1 - 0.05*(37 -39)))\} \quad F39 = (39.00000, 4.52083) \]
\[ \text{point}\(F40\)\{39.75,(m*m40*(10**(-0.00269*(34-40)))*(1 - 0.05*(37 -39.75)))\} \quad F40 = (39.75000, 4.67505) \]
\color{black}

% drawpoint(F35 F36 F37 F38 F39 )
\put {$\bullet$} at 35.00000 3.69894 $\%$ F35 
\put {$\bullet$} at 36.00000 3.90444 $\%$ F36 
\put {$\bullet$} at 37.00000 4.10994 $\%$ F37 
\put {$\bullet$} at 38.00000 4.31543 $\%$ F38 
\put {$\bullet$} at 39.00000 4.52093 $\%$ F39

%% test(\turnbox{7.97587571582933}{\boldmath$0.6$}){F37, polar(r9, 
\(7.97587571582933 + t9\) deg)})
\put \turnbox{7.97587571582933}{\boldmath$0.6$} at 37.133889 4.258513

%%%=======
%%% now draw all the horizontal dashed lines
\setsolid
\linethickness=0.40000pt
\setplotsymbol ({\CM .})
%% var w1=34.1 $\%$ left end of dashed lines
%% w1 = 34.1
%% var w2=39.85 $\%$ right end of dashed lines
%% w2 = 39.85
%% point(L16){w1, ycoord(A37)} L16 = (34.10000, 10.95983)
%% point(R16){w2, ycoord(A37)} R16 = (39.85000, 10.95983)
%% point(L14){w1, ycoord(B37)} L14 = (34.10000, 9.58985)
%% point(R14){w2, ycoord(B37)} R14 = (39.85000, 9.58985)
%% point(L12){w1, ycoord(C37)} L12 = (34.10000, 8.21987)
%% point(R12){w2, ycoord(C37)} R12 = (39.85000, 8.21987)
%% point(L10){w1, ycoord(D37)} L10 = (34.10000, 6.84990)
%% point(R10){w2, ycoord(D37)} R10 = (39.85000, 6.84990)
%% point(L08){w1, ycoord(E37)} L08 = (34.10000, 5.47992)
%% point(R08){w2, ycoord(E37)} R08 = (39.85000, 5.47992)
%% point(L06){w1, ycoord(F37)} L06 = (34.10000, 4.10994)
%% point(R06){w2, ycoord(F37)} R06 = (39.85000, 4.10994)
%% drawline(L16 R16, L14 R14, L12 R12, L10 R10, L08 R08, L06 R06)
\putrule from 34.10000 10.95983 to 39.85000 10.95983 $\%$ L16R16
\putrule from 34.10000 9.58985 to 39.85000 9.58985 $\%$ L14R14
\putrule from 34.10000 8.21987 to 39.85000 8.21987 $\%$ L12R12
\putrule from 34.10000 6.84990 to 39.85000 6.84990 $\%$ L10R10
\putrule from 34.10000 5.47992 to 39.85000 5.47992 $\%$ L08R08
\putrule from 34.10000 4.10994 to 39.85000 4.10994 $\%$ L06R06

%%% linethickness(1pt)
\linethickness=1.00000pt
\setplotsymbol ({\CM .})
%%%------------------
%%%-------------
\normalcolor
\endpicture
\end{document}
7.5 Program \texttt{macATdes2batch.pl} for automating the pdf output

Since most anaesthetists do not have Perl installed on their (Microsoft) PCs, they are unable to run the above interactive program (for generating a single chart for a specific agent and age), and so it was decided to distribute a CD with all the charts for all possible ages. The following batch program generates all the desflurane charts (one for each age), placing all in the same directory. The filenames are age encoded for convenience.

```perl
#!/usr/bin/perl
## Desflurane
## \texttt{macATdes2batch.pl} (from \texttt{macatsdes1batch.pl})
## Dick Nickalls
## May 01,2005
##
## use Math::Trig;
##
## program name is $O (see black book p 211
$timenowgmt = localtime;
#======================
## define filename to be printed on chart (pdf or ps)
$filename = $psfilename;
```

```bash
for ($age = 1; $age <= 120; $age = $age + 1) {

## need the filenames to have 3 digit integers
## force age --> 3 integer digits eg 002, 055, etc
## see black book p 233
## $agef is the formatted age string (only for filenames)
$agef = sprintf "%.3u", $age;
$basename="macdes";
$filename=$basename.$agef.".m";
$basename=$basename.$agef.".mt";
$basename=$basename.$agef.".ps";
$basename=$basename.$agef.".pdf";
$basename=$basename.$agef.".dvi";
```

```
print "\n- - - - - - - - - - - - - - - - - -\n";
print " \ making files for age = \$age\n";
print " \n";

##==================================================================
# create the output file
open (outfile, ">".$mfilename)| die "ERROR: can’t create file $mfilename\n";
##==================================================================
# do some initial calculation of some required variables
## FIRST calculate the slopes of all lines for mAC1.6 -- > 0.6

$pi = 4*atan2(1,1);
$r2d=180/$pi;
$d2r=$pi/180;
$m40=6.6; # desflurane MAC40
$agecorr= (10**(-0.00269*($age-40))); # agecorr
## \%def etdes(t)(m\*m40\&agecorr\{1 - 0.05+(37 -t)\})\% etdes
$s1634=etdes(1.6, 34);
$s1640=etdes(1.6, 40);
$s16deg= $r2d*atan(($s1640-$s1634)*5/(6*$pi/2));
$s1434=etdes(1.4, 34);
$s1440=etdes(1.4, 40);
$s14deg= $r2d*atan(($s1440-$s1434)*5/(6*$pi/2));
$s1234=etdes(1.2, 34);
$s1240=etdes(1.2, 40);
$s12deg= $r2d*atan(($s1240-$s1234)*5/(6*$pi/2));
$s1034=etdes(1.0, 34);
$s1040=etdes(1.0, 40);
$s10deg= $r2d*atan(($s1040-$s1034)*5/(6*$pi/2));
$s0834=etdes(0.8, 34);
$s0840=etdes(0.8, 40);
$s08deg= $r2d*atan(($s0840-$s0834)*5/(6*$pi/2));
$s0634=etdes(0.6, 34);
$s0640=etdes(0.6, 40);
$s06deg= $r2d*atan(($s0640-$s0634)*5/(6*$pi/2));
print "etdes1634 = \$s1634\n";
print "etdes1640 = \$s1640\n";
print "slope s16Deg = \$s16deg\n";
print "slope s14Deg = \$s14deg\n";
print "slope s12Deg = \$s12deg\n";
print "slope s10Deg = \$s10deg\n";
print "slope s08Deg = \$s08deg\n";
print "slope s06Deg = \$s06deg\n";
## now to do 67% axis scale
##\$age = 12;
##\$m40=6.6;
# $agecorr = 10^*(-0.00269*($age - 40));
## first need to know what Y2 is
$k2 = (1.6 * $m40 + $agecorr * (1 - 0.05 * (37 - 40)));
## increment scales by 0.5%
$y2 = int((k2 + 0.25) / 0.5);
print "k2 = $k2 "n;
print "j2 = $y2 "n;
print "y2 = $y2 "n;
## for left scale 100% o2
## k1 is ycoord of KHS of isoMAC 0.6 line
$k1 = (0.6 * $m40 + $agecorr * (1 - 0.05 * (37 - 34)));
$y1 = int(($k1 - 0.25) / 0.5);
print "k1 = $k1 "n;
print "j1 = $y1 "n;
print "y1 = $y1 "n;
##==============
## 100 % oxygen LHS scale
$K = int($y2 / 0.5);
# make string A
$string100a = "\ ticks with values ";
$string100b = "\ at ";
for ($j = $y1; $j < $y2 + 0.15; $j = $j + 0.5) {
## problem with $j < $y2 loses last cycle
## hence use fix $j < $y2 + 0.15
# print "$j = $j "n;
## format values to 1 decimal place
$vv = sprintf "%.1f", $j;
if ($vv == "0.0") {$vv = "0";)
$string100a = $string100a. "$vv "$;
$string100b = $string100b. "$ $j ";
}
$string100a = $string100a. " / / ";
$string100b = $string100b. " / / ";
###=============
## desflurane
## for scale 67% get offset from file mac – des7.m
$offset67 = 4.217;
$y67 = $y2 - $offset67;
$K = int($y67 / 0.5);
# make string A
$string67a = "\ ticks with values ";
$string67b = "\ at ";
$value67b = $offset67;
for ($j = 0; $j <= $K; $j++) {
$value67a = $j + 0.5;
$value67b = $value67b + 0.5;
if ($j \text{ eq 0})
{ \$value67b = \$offset67; }
## format values to 1 decimal place
\$vv = \text{sprintf} \ "%\.1f\"\", \$value67a;
if (\$vv == "0.0"){\$vv = "0"};
\$string67a=\$string67a." \$$vv\$ ";
\$string67b=\$string67b." \$value67b ";
};
\$string67a=\$string67a." / ";
\$string67b=\$string67b." / / ";
#===============================================
#===============================================
## for scale 50% get offset from file mac−des7.m
\$offset50= 3.147;
\$y50=\$y2 − \$offset50;
\$k=\text{int}(\$y50/0.5);#
## make strings A and B
\$string50a="
\$\$ 
\$ticks withvalues ";
\$string50b="
\$\$ 
\$at ";
\$value50b = \$offset50;
for ($j=0; $j <\$k; $j++) 
{ 
\$value50a=$j \times 0.5;
\$value50b=\$value50b + 0.5;
if ($j \text{ eq 0}){\$value50b = \$offset50};
## format values to 1 decimal place
\$vv = \text{sprintf} \ "%\.1f\"\", \$value50a;
if (\$vv == "0.0"){\$vv = "0"};
\$string50a=\$string50a." \$$vv\$ ";
\$string50b=\$string50b." \$value50b ";
};
\$string50a=\$string50a." / ";
\$string50b=\$string50b." / / ";
#===============================================
#===============================================
\text{print}(\text{outfile} << \text{EOF});
\text{\% mathsPic}
\text{\% macDes1.m \% for Desflurane}
\text{\% to create temp and age MAC chart corrections}
\text{\% May 01, 2005}
\text{\%}-------------------------------
\documentclass[a4paper]{article}
\usepackage{mathpict,color}
\usepackage{decimal,rotating}
\setlength{\textwidth}{30cm}
\setlength{\oddsidemargin}{0cm}
\begin{document}
\begin{picture}
\text{\% DES MAC40 = 6.6}
\text{\% agent specific adjustments}
\text{\% need to adjust Xunits, Yunits, MAC, zero offset for N2O 50% and 67%}
\text{\% the Ytick interval (0.5 for des),}
\text{\% as well as the space above and below the line space}
\text{\%}
\text{\% def age()\$age \% AGE}
\text{\%}-------------------------------
\begin{verbatim}
var m40=6.6 \% desflurane MAC40
\% def agecorr(10+(-0.00269*(\&age-40))) \% agecorr
\% def etdes(t)(m*m40*agecorr*(1 - 0.05*(37 - t))) \% etdes
\%------------------------
\% determine the X and Y axes lengths (age dependent)
\% \% want to make the Y axis just longer than the line space
\% \% for the ticks to work nicely we want left Yaxis to be an
\% \% exact multiple of tick interval
\% \% therefore we add some extra and adjust with 0.2 increments for sevo
\% \% top right – use t=40 and MAC=1.6
var k2=(1.6*m40*agecorr*(1 - 0.05*(37 - 40))), j2=int((k2+0.25)/0.5)
var y2=(j2*0.5)+0.5
\% \% bottom left – use t=34 and MAC=0.6
var k1=(0.6*m40*agecorr*(1 - 0.05*(37 - 34))), j1=int((k1-0.25)/0.5)
var y1=(j1*0.5)-0.5
\% now do the paper command (only for axes(B) here (as no decimals))
\% paper{units(2cm,1.3cm), xrange(34,40), yrange(y1,y2), axes(B), ticks(1,0.5)}
\% \% left axis 100 % oxygen
\% \% right axis 67% N2O scale
\% label {lines{using N2O}}
\% \% extra right axis 50% N2O
\% \% initial conditions
linethickness(1pt) \%
\% \% draw vertical T37 line
\% \% make top end half way between top and MAC1.6 line
var y22=y2-(y2-k2)/2
point(T137)[37,y22]
point(B137)[37,y1]
drawline(T137 B137) \% vertical T=37 line
\% point(Q1){34,y2} \% top left corner
\% text({\sf \large NAME} {Q1, shift(0.2,-0.35)})[I]
text({\sf \large DESFLURANE} {37, y2 + 0.7})
\% def theage(jj)text({\sf \large AGE = jj} {Q1, shift(0.2,-0.7)})[I]
\end{verbatim}
\drawpoint(D35 D36 D37 D38 D39)
{text(\\turnbox{$s10\text{deg}$}{\\boldmath{$1.0$}})(D37, polar(r9,($s10\text{deg} + t9)\text{deg})}
\%\% get slope of line
var s=direction(D35 D40) \%\% radians
var r14 = atan((5/2) * tan(s))
var d14=r14 * &r2d
\%\%============
var m=0.8 \%\%MAC
point(E34)\{34.25,&etdes(34.25)\}
point(E35)\{35,&etdes(35)\}
point(E36)\{36,&etdes(36)\}
point(E37)\{37,&etdes(37)\}
point(E38)\{38,&etdes(38)\}
point(E39)\{39,&etdes(39)\}
point(E40)\{39.75,&etdes(39.75)\}
\color{blue}
drawline(E34 E37)
\color{red}
drawline(E37 E40)
\color{black}
drawpoint(E35 E36 E37 E38 E39)
\%\%============
\%\% now draw all the horizontal dashed lines
\setdashes
lineThickness(0.4pt)
var w1=34.1 \%\% left end of dashed lines
var w2=39.85 \%\% right end of dashed lines
point(L16)\{w1, ycoord(A37)\}
point(R16)\{w2, ycoord(A37)\}
point(L14)\{w1, ycoord(B37)\}
point(R14)\{w2, ycoord(B37)\}
point(L12)\{w1, ycoord(C37)\}
point(R12)\{w2, ycoord(C37)\}
point(L10)\{w1, ycoord(D37)\}
point(R10)\{w2, ycoord(D37)\}
point(L08)\{w1, ycoord(E37)\}
Note that in the above program we used the utility ps2pdf (line 435) to generate the PDF file without actually determining the Bounding Box of the image, as follows:

```perl
system("mathspic $mfilename");
system("latex $mtfilename");
```
system("dvips -o $psfilename $dvifilename");
# system("gv $psfilename");
system("ps2pdf $psfilename");

In fact it would have been better to have generated the PDF via the EPS version (has
the Bounding Box specified), by using DVIPS with the -E switch (which extracts
the Bounding Box information from the PostScript file), as the following BASH file
(makepdf.sh) shows.

-----makepdf.sh-----
echo ...running makepdf on file $1.tex
mathspic $1.m
latex $1.mt
dvips $1.dvi -o $1.ps
dvips -E $1.dvi -o $1.eps ## determines the BBox -->eps
epstopdf $1.eps
echo ... end of run

7.6 Test program MACsolver.pl

The following Perl test program was written to generate test data for checking the charts
against. The calculation of the corrected MAC\_age\_temp of desflurane for a given age,
temperature, and MAC\_fraction, uses the following age correction factor (Mapleson
1996; Nickalls and Mapleson 2003)

\[
\text{agecorr} = 10^{-0.00269(age-40)}
\]

and the 5\% linear temperature correction factor (Eger 2001)

\[
(1 - 0.05(37 - temp))
\]

to correct the MAC\_40 for desflurane (6\%-6\%). Thus for a given MAC\_fraction of 1.6 we
have

\[ET\text{desflurane} = MAC\text{frac} \times MAC\_40,\text{des} \times \text{agecorr}(age) \times (1 - 0.05(37 - temp));\]

For example, if age = 40yrs and temp = 37 then this reduces to

\[ET\text{desflurane} = MAC\text{frac} \times MAC\_40,\text{des} \times 1 \times 1\]

i.e. we obtain, as expected,

\[MAC\text{fraction} = \frac{ET\text{desflurane}}{6.6}\]

In order to generate the chart for a given agent and age, we select particular MAC\_fractions (0.6--1.6) and calculate the associated ET concentrations for the temperature range 34--40 for the given age. These results are then plotted as a series of straight lines (colour coded) centered on the normal temperature 37.

As a final check we then use the above program to calculate the values associated
with the extremes of the lines for 1.6 and 0.6.

For example, the program generates the following output for age 34, temp 37, mac\_fraction1.6:
ET hal (age=34, temp=37, macFrac=1.6) = 1.24543551525272
ET iso (age=34, temp=37, macFrac=1.6) = 1.94287940379424
ET sevo (age=34, temp=37, macFrac=1.6) = 2.98904523660653
ET des (age=34, temp=37, macFrac=1.6) = 10.9598325342239

and for age 34, temp 40, macFraction1.6 we get

ET hal (age=34, temp=40, macFrac=1.6) = 1.43225084254063
ET iso (age=34, temp=40, macFrac=1.6) = 2.23431131436338
ET sevo (age=34, temp=40, macFrac=1.6) = 3.43740202209751
ET des (age=34, temp=40, macFrac=1.6) = 12.6038074143575

The test program is as follows:

#!/usr/bin/perl
## LINUX version
## /book−xenon/macAT/testing/MACsolver.pl
##
## Dick Nickalls
## Jan 5, 2007
##
## numerical test prog to
## calculate correct age & temp corr mac
## to serve as a computational check on my MAC,age,temp graphs
##-----------------------------------------------
# warning switch
use warnings;
use strict;
use Carp;
##use Perl6::Builtins qw( system );
#use Getopt::Long; ## for commandline stuff
#use version;
#use Cwd; # grab this dir

#-------------------
$age=0;

#---
$m40=0.75; #halothane
$agecorr = (10**(−0.00269*(thisage−40)));
$ethal = ($mf*$m40*$agecorr*(1−0.05*(37−$t)));

print "EThal = $ethal \n";
#-------------------

print "test program (Perl5, Linux) for MAC age & temp correction charts\n";
print "c RWD Nickalls, October, 2008\n";
print "filename = MACsolver.pl\n";
print "Uses MAC and temp correction data from: Nickalls & Mapleson (2003), and Eger (2001)\n";
print "agecorr = (10**((−0.00269*(thisage−40)))\n";
print "ETdesflurane = (macfrac+m40des*agecorr(myage)*(1 − 0.05*(37 − temp)) + nitrousoffset)\n";

my $age=0;
$age=22;
print "="-'-"n;";
ETisoflurane($age, 34, 0, 0.6);
ETisoflurane($age, 37, 0, 0.6);
ETisoflurane($age, 40, 0, 0.6);
ETisoflurane($age, 34, 50, 1.6);
ETisoflurane($age, 37, 50, 1.6);
ETisoflurane($age, 40, 50, 1.6);
ETisoflurane($age, 34, 67, 1.6);
ETisoflurane($age, 37, 67, 1.6);
ETisoflurane($age, 40, 67, 1.6);
print "-''-'-"n;";
ETsevoflurane($age, 34, 0, 0.6);
ETsevoflurane($age, 37, 0, 0.6);
ETsevoflurane($age, 40, 0, 0.6);
ETsevoflurane($age, 34, 50, 1.6);
ETsevoflurane($age, 37, 50, 1.6);
ETsevoflurane($age, 40, 50, 1.6);
ETsevoflurane($age, 34, 67, 1.6);
ETsevoflurane($age, 37, 67, 1.6);
ETsevoflurane($age, 40, 67, 1.6);
print "-''-'-"n;";
ETdesflurane($age, 34, 0, 0.6);
ETdesflurane($age, 37, 0, 0.6);
ETdesflurane($age, 40, 0, 0.6);
ETdesflurane($age, 34, 50, 1.6);
ETdesflurane($age, 37, 50, 1.6);
ETdesflurane($age, 40, 50, 1.6);
ETdesflurane($age, 34, 67, 1.6);
ETdesflurane($age, 37, 67, 1.6);
ETdesflurane($age, 40, 67, 1.6);

$age=40;
print "="-'-"n;";
ETisoflurane($age, 34, 0, 0.6);
ETisoflurane($age, 37, 0, 0.6);
ETisoflurane($age, 40, 0, 0.6);
ETisoflurane($age, 34, 50, 1.6);
ETisoflurane($age, 37, 50, 1.6);
ETisoflurane($age, 40, 50, 1.6);
ETisoflurane($age, 34, 67, 1.6);
ETisoflurane($age, 37, 67, 1.6);
ETisoflurane($age, 40, 1.6);

print "−−−−−−−−−−−−−−−−−−−−−−\n";
ETsevoflurane($age, 34, 0, 0.6);
ETsevoflurane($age, 37, 0, 0.6);
ETsevoflurane($age, 40, 0, 0.6);

ETsevoflurane($age, 34, 50, 1.6);
ETsevoflurane($age, 37, 50, 1.6);
ETsevoflurane($age, 40, 50, 1.6);

ETsevoflurane($age, 34, 67, 1.6);
ETsevoflurane($age, 37, 67, 1.6);
ETsevoflurane($age, 40, 67, 1.6);

ETdesflurane($age, 34, 0, 0.6);
ETdesflurane($age, 37, 0, 0.6);
ETdesflurane($age, 40, 0, 0.6);

ETdesflurane($age, 34, 50, 1.6);
ETdesflurane($age, 37, 50, 1.6);
ETdesflurane($age, 40, 50, 1.6);

ETdesflurane($age, 34, 67, 1.6);
ETdesflurane($age, 37, 67, 1.6);
ETdesflurane($age, 40, 67, 1.6);

$sage=80;

print "−−−−−−−−−−−−−−−−−−−−−−\n";
ETisoflurane($age, 34, 0, 0.6);
ETisoflurane($age, 37, 0, 0.6);
ETisoflurane($age, 40, 0, 0.6);

ETisoflurane($age, 34, 50, 1.6);
ETisoflurane($age, 37, 50, 1.6);
ETisoflurane($age, 40, 50, 1.6);

ETisoflurane($age, 34, 67, 1.6);
ETisoflurane($age, 37, 67, 1.6);
ETisoflurane($age, 40, 67, 1.6);

print "−−−−−−−−−−−−−−−−−−−−−−\n";
ETsevoflurane($age, 34, 0, 0.6);
ETsevoflurane($age, 37, 0, 0.6);
ETsevoflurane($age, 40, 0, 0.6);

ETsevoflurane($age, 34, 50, 1.6);
ETsevoflurane($age, 37, 50, 1.6);
ETsevoflurane($age, 40, 50, 1.6);
CHAPTER 7. AGE & TEMPERATURE CORRECTED MAC-CHARTS  RWD Nickalls

ETSevoflurane($age, 34, 67, 1.6);
ETSevoflurane($age, 37, 67, 1.6);
ETSevoflurane($age, 40, 67, 1.6);

print "−−−−−−−−−−−−−−−−−−−−−\n";
ETDesflurane($age, 34, 0, 0.6);
ETDesflurane($age, 37, 0, 0.6);
ETDesflurane($age, 40, 0, 0.6);

ETDesflurane($age, 34, 50, 1.6);
ETDesflurane($age, 37, 50, 1.6);
ETDesflurane($age, 40, 50, 1.6);

ETDesflurane($age, 34, 67, 1.6);
ETDesflurane($age, 37, 67, 1.6);
ETDesflurane($age, 40, 67, 1.6);

print "====================\n";
#

sub EThalothane {
  ## calculate et—halothane (from age, macfrac, temp)
  my ($myage, $temp, $macfrac,) = @_
  #print "mac, temp = $mac, $temp\n";
  my $m40hal=0.75;
  my (EThalothane, $string, $ethal4dp);
  $EThalothane = ($macfrac * $m40hal * agecorr($myage) * (1 − 0.05 * (37 − $temp)));

my $ethal4dp=sprintf "%.4f", $EThalothane;
$string="(age=$myage, temp=$temp, macFrac=$macfrac)";
print "ET hal $string, $ethal4dp\n";
#return $EThalothane;
}

sub ETIsoflurane {
  ## calculate et—isoflurane (from age, macfrac, temp)
  my ($myage, $temp, $nitrous, $macfrac,) = @_
  #print "mac, temp = $mac, $temp\n";
  my $m40iso=1.17;
  my $nitrousoffset50iso=−0.56;
  my $nitrousoffset67iso=−0.75;
  my ($ETIsoflurane, $string, $etiso4dp);
  if ($nitrous==50) {
    $ETIsoflurane = ($macfrac * $m40iso * agecorr($myage) * (1 − 0.05 * (37 − $temp))) + $nitrousoffset50iso
  } elsif ($nitrous==67) {
    $ETIsoflurane = ($macfrac * $m40iso * agecorr($myage) * (1 − 0.05 * (37 − $temp))) + $nitrousoffset67iso
  } else {
    $ETIsoflurane = ($macfrac * $m40iso * agecorr($myage) * (1 − 0.05 * (37 − $temp)))
  }

my $etiso4dp=sprintf "%.4f", $ETIsoflurane;
$string="(age=$myage, temp=$temp, macFrac=$macfrac, n2o=$nitrous)";
print "ET iso $string, $etiso4dp\n";
#return $ETIsoflurane;
}
sub ETsevoflurane {
    ## calculate et—sevoflurane (from age, macfrac, temp)
    my ($myage, $temp, $nitrous, $macfrac) = @_
    #print "mac, temp = "$mac, "$temp\n";
    my $m40sevo=1.8;
    my $nitrousoffset50sevo=−0.86;
    my $nitrousoffset67sevo=−1.16;
    my ($ETsevoflurane, $string, $etsevo4dp);
    if ($nitrous==50){$ETsevoflurane = ($macfrac*$m40sevo*agecorr($myage)*(1 − 0.05*(37 −$temp))+ $nitrousoffset50sevo}
    elsif ($nitrous==67){$ETsevoflurane = ($macfrac*$m40sevo*agecorr($myage)*(1 − 0.05*(37 −$temp))+ $nitrousoffset67sevo}
    else {$ETsevoflurane = ($macfrac*$m40sevo*agecorr($myage)*(1 − 0.05*(37 −$temp)))};
    $etsevo4dp=sprintf "%.4f", $ETsevoflurane;
    $string="(age=$myage, temp=$temp, macFrac=$macfrac, n2o=$nitrous)";
    print "ET sevo $string, $etsevo4dp \n";
    #return $ETsevoflurane;
}

sub ETdesflurane {
    ## calculate et—desflurane (from age, macfrac, temp)
    my ($myage, $temp, $nitrous, $macfrac) = @_
    #print "mac, temp = "$mac, "$temp\n";
    my $m40des=6.6;
    my $nitrousoffset50des=−3.15;
    my $nitrousoffset67des=−4.22;
    my ($ETdesflurane, $string, $etdes4dp);
    if ($nitrous==50){$ETdesflurane = ($macfrac*$m40des*agecorr($myage)*(1 − 0.05*(37 −$temp))+ $nitrousoffset50des}
    elsif ($nitrous==67){$ETdesflurane = ($macfrac*$m40des*agecorr($myage)*(1 − 0.05*(37 −$temp))+ $nitrousoffset67des}
    else {$ETdesflurane = ($macfrac*$m40des*agecorr($myage)*(1 − 0.05*(37 −$temp)))};
    $etdes4dp=sprintf "%.4f", $ETdesflurane;
    $string="(age=$myage, temp=$temp, macFrac=$macfrac, n2o=$nitrous)";
    print "ET des $string, $etdes4dp \n";
    #return $ETdesflurane;
}

sub agecorr {
    ## calculate age—correction factor (from age)
    my ($thisage) = @_
    my ($agecorr);
    $agecorr = (10**(-0.00269*($thisage−40)));
    return $agecorr;
}

##$
### 7.6.1 Output datafile (macsolver-out.txt)

Test program (Perl5, Linux) for MAC age & temp correction charts

c RWD Nickalls, October, 2008

data generated by prog <MACsolver.pl>

Uses MAC and temp correction data from: Nickalls & Mapleson (2003), and Eger (2001)

\[
\text{agecorr} = (10^{0.00269 \times (\text{thisage}-40)})
\]

\[
\text{ETdesflurane} = (\text{macfrac} \times 40\text{des} \times \text{agecorr} \times (\text{myage}) \times (1 - 0.05 \times (37 - \text{temp}))) + \text{nitrousoffset}
\]

<table>
<thead>
<tr>
<th>Condition</th>
<th>ET (Age 22)</th>
<th>ET (Age 40)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Age=22, Temp=34, MAC=0.6, N2O=0)</td>
<td>0.6671</td>
<td>0.5967</td>
</tr>
<tr>
<td>(Age=22, Temp=37, MAC=0.6, N2O=0)</td>
<td>0.7848</td>
<td>0.7020</td>
</tr>
<tr>
<td>(Age=22, Temp=40, MAC=0.6, N2O=0)</td>
<td>0.9025</td>
<td>0.8073</td>
</tr>
<tr>
<td>(Age=22, Temp=34, MAC=1.6, N2O=50)</td>
<td>1.2189</td>
<td>1.0312</td>
</tr>
<tr>
<td>(Age=22, Temp=37, MAC=1.6, N2O=50)</td>
<td>1.5328</td>
<td>1.3120</td>
</tr>
<tr>
<td>(Age=22, Temp=40, MAC=1.6, N2O=50)</td>
<td>1.8467</td>
<td>1.5928</td>
</tr>
<tr>
<td>(Age=22, Temp=34, MAC=1.6, N2O=67)</td>
<td>1.0289</td>
<td>0.8412</td>
</tr>
<tr>
<td>(Age=22, Temp=37, MAC=1.6, N2O=67)</td>
<td>1.3428</td>
<td>1.1220</td>
</tr>
<tr>
<td>(Age=22, Temp=40, MAC=1.6, N2O=67)</td>
<td>1.6567</td>
<td>1.4028</td>
</tr>
</tbody>
</table>

Sevo

<table>
<thead>
<tr>
<th>Condition</th>
<th>ET (Age 22)</th>
<th>ET (Age 40)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Age=22, Temp=34, MAC=0.6, N2O=0)</td>
<td>1.0263</td>
<td>0.9180</td>
</tr>
<tr>
<td>(Age=22, Temp=37, MAC=0.6, N2O=0)</td>
<td>1.2074</td>
<td></td>
</tr>
<tr>
<td>(Age=22, Temp=40, MAC=0.6, N2O=0)</td>
<td>1.3885</td>
<td></td>
</tr>
<tr>
<td>(Age=22, Temp=34, MAC=1.6, N2O=50)</td>
<td>1.8767</td>
<td></td>
</tr>
<tr>
<td>(Age=22, Temp=37, MAC=1.6, N2O=50)</td>
<td>2.3597</td>
<td></td>
</tr>
<tr>
<td>(Age=22, Temp=40, MAC=1.6, N2O=50)</td>
<td>2.8426</td>
<td></td>
</tr>
<tr>
<td>(Age=22, Temp=34, MAC=1.6, N2O=67)</td>
<td>1.5767</td>
<td></td>
</tr>
<tr>
<td>(Age=22, Temp=37, MAC=1.6, N2O=67)</td>
<td>2.0597</td>
<td></td>
</tr>
<tr>
<td>(Age=22, Temp=40, MAC=1.6, N2O=67)</td>
<td>2.5426</td>
<td></td>
</tr>
</tbody>
</table>

Des

<table>
<thead>
<tr>
<th>Condition</th>
<th>ET (Age 22)</th>
<th>ET (Age 40)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Age=22, Temp=34, MAC=0.6, N2O=0)</td>
<td>3.7630</td>
<td></td>
</tr>
<tr>
<td>(Age=22, Temp=37, MAC=0.6, N2O=0)</td>
<td>4.4271</td>
<td></td>
</tr>
<tr>
<td>(Age=22, Temp=40, MAC=0.6, N2O=0)</td>
<td>5.0911</td>
<td></td>
</tr>
<tr>
<td>(Age=22, Temp=34, MAC=1.6, N2O=50)</td>
<td>6.8847</td>
<td></td>
</tr>
<tr>
<td>(Age=22, Temp=37, MAC=1.6, N2O=50)</td>
<td>8.6555</td>
<td></td>
</tr>
<tr>
<td>(Age=22, Temp=40, MAC=1.6, N2O=50)</td>
<td>10.4263</td>
<td></td>
</tr>
<tr>
<td>(Age=22, Temp=34, MAC=1.6, N2O=67)</td>
<td>5.8147</td>
<td></td>
</tr>
<tr>
<td>(Age=22, Temp=37, MAC=1.6, N2O=67)</td>
<td>7.5855</td>
<td></td>
</tr>
<tr>
<td>(Age=22, Temp=40, MAC=1.6, N2O=67)</td>
<td>9.3563</td>
<td></td>
</tr>
</tbody>
</table>

ISO

<table>
<thead>
<tr>
<th>Condition</th>
<th>ET (Age 40)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Age=40, Temp=34, MAC=0.6, N2O=0)</td>
<td>0.5967</td>
</tr>
<tr>
<td>(Age=40, Temp=37, MAC=0.6, N2O=0)</td>
<td>0.7020</td>
</tr>
<tr>
<td>(Age=40, Temp=40, MAC=0.6, N2O=0)</td>
<td>0.8073</td>
</tr>
<tr>
<td>(Age=40, Temp=34, MAC=1.6, N2O=50)</td>
<td>1.0312</td>
</tr>
<tr>
<td>(Age=40, Temp=37, MAC=1.6, N2O=50)</td>
<td>1.3120</td>
</tr>
<tr>
<td>(Age=40, Temp=40, MAC=1.6, N2O=50)</td>
<td>1.5928</td>
</tr>
<tr>
<td>(Age=40, Temp=34, MAC=1.6, N2O=67)</td>
<td>0.8412</td>
</tr>
<tr>
<td>(Age=40, Temp=37, MAC=1.6, N2O=67)</td>
<td>1.1220</td>
</tr>
<tr>
<td>(Age=40, Temp=40, MAC=1.6, N2O=67)</td>
<td>1.4028</td>
</tr>
</tbody>
</table>

Sevo

<table>
<thead>
<tr>
<th>Condition</th>
<th>ET (Age 40)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Age=40, Temp=34, MAC=0.6, N2O=0)</td>
<td>0.9180</td>
</tr>
<tr>
<td>Method</td>
<td>Age</td>
</tr>
<tr>
<td>--------</td>
<td>-----</td>
</tr>
<tr>
<td>Sevoflurane</td>
<td>40</td>
</tr>
<tr>
<td>Sevoflurane</td>
<td>40</td>
</tr>
<tr>
<td>Sevoflurane</td>
<td>40</td>
</tr>
<tr>
<td>Sevoflurane</td>
<td>40</td>
</tr>
<tr>
<td>Sevoflurane</td>
<td>40</td>
</tr>
<tr>
<td>Sevoflurane</td>
<td>40</td>
</tr>
<tr>
<td>Sevoflurane</td>
<td>40</td>
</tr>
<tr>
<td>Sevoflurane</td>
<td>40</td>
</tr>
<tr>
<td>Desflurane</td>
<td>40</td>
</tr>
<tr>
<td>Desflurane</td>
<td>40</td>
</tr>
<tr>
<td>Desflurane</td>
<td>40</td>
</tr>
<tr>
<td>Desflurane</td>
<td>40</td>
</tr>
<tr>
<td>Desflurane</td>
<td>40</td>
</tr>
<tr>
<td>Desflurane</td>
<td>40</td>
</tr>
<tr>
<td>Desflurane</td>
<td>40</td>
</tr>
<tr>
<td>Desflurane</td>
<td>40</td>
</tr>
<tr>
<td>Isoflurane</td>
<td>80</td>
</tr>
<tr>
<td>Isoflurane</td>
<td>80</td>
</tr>
<tr>
<td>Isoflurane</td>
<td>80</td>
</tr>
<tr>
<td>Isoflurane</td>
<td>80</td>
</tr>
<tr>
<td>Isoflurane</td>
<td>80</td>
</tr>
<tr>
<td>Isoflurane</td>
<td>80</td>
</tr>
<tr>
<td>Isoflurane</td>
<td>80</td>
</tr>
<tr>
<td>Isoflurane</td>
<td>80</td>
</tr>
<tr>
<td>Isoflurane</td>
<td>80</td>
</tr>
<tr>
<td>Sevoflurane</td>
<td>80</td>
</tr>
<tr>
<td>Sevoflurane</td>
<td>80</td>
</tr>
<tr>
<td>Sevoflurane</td>
<td>80</td>
</tr>
<tr>
<td>Sevoflurane</td>
<td>80</td>
</tr>
<tr>
<td>Sevoflurane</td>
<td>80</td>
</tr>
<tr>
<td>Sevoflurane</td>
<td>80</td>
</tr>
<tr>
<td>Sevoflurane</td>
<td>80</td>
</tr>
<tr>
<td>Sevoflurane</td>
<td>80</td>
</tr>
<tr>
<td>Sevoflurane</td>
<td>80</td>
</tr>
<tr>
<td>Desflurane</td>
<td>80</td>
</tr>
<tr>
<td>Desflurane</td>
<td>80</td>
</tr>
<tr>
<td>Desflurane</td>
<td>80</td>
</tr>
<tr>
<td>Desflurane</td>
<td>80</td>
</tr>
<tr>
<td>Desflurane</td>
<td>80</td>
</tr>
<tr>
<td>Desflurane</td>
<td>80</td>
</tr>
<tr>
<td>Desflurane</td>
<td>80</td>
</tr>
<tr>
<td>Desflurane</td>
<td>80</td>
</tr>
<tr>
<td>Desflurane</td>
<td>80</td>
</tr>
</tbody>
</table>
7.7 References

- Eger EI (2001). Age, minimum alveolar anesthetic concentration, and minimum alveolar anesthetic concentration-awake. *Anesthesia and Analgesia; 93*, 947–953. [has an appendix on temperature correction]


Age & temperature corrected iso-MAC chart

Eger EI (2001). Anesthesia and Analgesia, 93, 947–953

Figure 7.2:
View of a typical age & temperature corrected MAC chart for desflurane, generated by the Perl program macatdes2.pl
Part II

The front-end coordinating program
Chapter 8

The Perl/Tk front-end

8.1 Introduction

The camomile program is currently launched by a Perl/Tk program which allows the user to launch the main camomile program, as well as the other associated components of the system (e.g. access the epidural and double-lumen tube database, print out the anaesthesia record etc). Clicking on the ‘run camomile’ button launches the coordinating program launchcam12.pl which launches the Camomile program itself.

bash runcamomile.sh (generates the widget <tklaunch2.pl>)
--- click on "RUN" button
--- perl launchcam12.pl (runs the Camomile program)
--- at end of operation terminate program (click on "QUIT" menu option)
   --- closes down screen and generates the widget again
--- click on "PRINT LAST CASE" button
   (generates the paper and HTML Anaesthetic Record)

After the anaesthetic/operation we terminate the launchcamXX.pl program and control reverts to the launching widget, from which we can then start the post-processing of the collected data and hence generate the printed Anaesthesia Record. More recently, the Anaesthesia Record data and graphs have been conveniently coordinated via a HTML frontend which allows all the data, programs and graphs to be viewed easily. The buttons are mapped to programs as follows:

- RUN (camomile) → launchcam12.pl
- EPIDURAL (database) → epidural.pl
- PROJECT TEAM → camteana5dvi.dvi
- QUIT → exit()
- PRINT LAST CASE (not active; just gives help message)
Figure 8.1:
Screen showing the initial graphic front-end (loader widget; <tklaunch2.pl>) which allows the user to either start the Camomile program, or access other utilities (e.g., process the data from last case, or run the Tube & EPIDural database program—TEPID). Note that the program <tklaunch2.pl> is itself launched by the bash script <runcamomile>.

Allow the user to search the TEPID database to determine the predicted tube length/size and epidural depth for a given patient, by inputting age, gender, height, weight.

8.2 The BASH script runcamomile

In practice, the graphic front-end is itself launched by the small BASH script runcamomile. The reason for using a preliminary script to launch the Perl/Tk program is because this allows the initial start-up size and position of the Tk widget to be easily controlled using the `geometry` commandline option.

```bash
#!/usr/bin/bash
# runcamomile.sh
## BASH script to change dir to --> /datexsim
##-------
echo "changing directory to ~/allfiles/camomiletop/datexsim"
cd /home/dick/allfiles/camomiletop/datexsim/
perl ./tklaunch2.pl -geometry 300x400-50-300
```

Note that the opening size and position in the screen is set using the `geometry` switch and its various options width, height, x-shift, y-shift (see Lidie and Walsh, 2002, p 409). The format for the `geometry` switch is as follows

```
.... -geometry [width]x[height]{+|-}[x-shift]{+|-}[y-shift]
```

1See the book: *Mastering Perl/Tk* by Lidie S and Walsh N (O'Reilly).
The sign option \{+\-\} determines the location of the origin of the screen coordinates. The \- sign is associated with the position of the bottom right-hand corner of the widget relative to the bottom right-hand corner of the screen, and the \+ sign is associated with the top left-hand corner of the widget relative to the top left-hand corner of the screen.

In order to make the script function ‘globally’ (i.e. much as a DOS batch-file would), it first has its mode set to ‘executable’ using the Linux command

```
$ chmod u+x runcamomile
```

(which adds the ‘executable’ permission for the user), and then the script (which must have no file extension) is placed in the $PATH, which in the case of a Linux ‘user’s’ batch-file means that it is placed in the standard directory `/usr/local/bin/` (which is always in the Linux $PATH), i.e.

```
/usr/local/bin/runcamomile
```

Now, whichever directory the user types the command `runcamomile` in, then Linux will move to the `.../datexsim` directory and run the `tklaunch2.pl` program.

### 8.3 Pressing the “RUN” button

The subroutine and code which starts the Camomile program is as follows: Clicking on one of the button first deletes the screen widget (to prevent another button being pressed), calls the associated program or message widget, and finally restores the screen widget when the launched program terminates. For example pressing the ‘RUN’ button launches the perl program `launchcam12.pl` by calling the subroutine `launch()` as follows.

```
sub launch {  
  if (−e "launchcam12.pl")  
    # first remove the Tk screen  
    $topwindow −>destroy if Tk::Exists($topwindow);  
    # now launch the program  
    system("perl ./launchcam12.pl");  
    # reinstate the widget when the program terminates  
    system("perl ./tklaunch2.pl −geometry 300x400−50−300");  
  else{print "....ERROR: \n    print "....can’t find program <launchcam12.pl>\n\n";exit();  
  }

8.3.1 Program: tklaunch2.pl

The widget program uses the perl Tk module, and the associated Tk::DialogBox. Note that the nice Perl5/Tk logo is the image anim.gif which can be found at the following directory. `/usr/lib/perl5/vendor_perl/5.8.1/i386-linux-thread-multi/Tk/`
#!/usr/bin/perl

use Tk;
use Tk::DialogBox;

$topwindow = MainWindow -> new();

$dialog1 = $topwindow -> DialogBox( title => "STATUS", buttons => ["OK"]);

$dialog1 -> add("Label",
              text => "The PRINT option is not enabled just now. However, in due course the PRINT button will coordinate printing out of all the sheets from the last operation",
              wraplength => 400)
              -> pack();

$topwindow -> title("Launch CAMOMILE");
$topwindow -> Label(text => "Click on the <RUN> button to start the CAMOMILE anaesthesia program",
                   wraplength => 130,
                   padx => 250,
                   height => 10 )
                   -> pack();

$topwindow -> title("Launch CAMOMILE");
$topwindow -> Label(text => "(c) The CAMOMILE project team 2004",
                   padx => 30,
                   pady => 20,
                   relief => 'flat',
                   background => 'LightGrey',
                   activebackground => 'Grey',
                   foreground => 'Blue',
                   command => '\&projectteam')
                   -> pack( side => 'bottom', expand => 1);

$topwindow -> title("Launch CAMOMILE");
$topwindow -> Label(text => "RUN",
                   padx => 50,
                   pady => 90,
                   relief => 'raised',
                   background => 'SeaGreen1',
                   activebackground => 'SeaGreen2',
                   command => '\&launch')
                   -> pack( side => 'left', expand => 1);
# QUIT button
$topwindow → Button (−text => "QUIT",  
−padx => 20, −pady => 20,  
−relief => 'raised',  
−background => 'LightBlue1',  
−activebackground => 'LightBlue2',  
−command => \&quit )  
−> place(−relx=>0, −rely=>0.1);  
#−> pack(−side =>'left', −expand => 1);

# EPIDURAL button
$topwindow → Button (−text => "EPIDURAL and DOUBLE−LUMEN TUBE  
database",  
−wraplength => 110,  
−padx => 30, −pady => 50,  
−relief => 'raised',  
−background => 'DarkSeaGreen2',  
−activebackground => 'DarkSeaGreen3',  
−command => \&epidural )  
−> pack(−side =>'bottom', −expand => 1); ##right

# PRINT button
$topwindow → Button (−text => "PRINT LAST CASE",  
−padx => 60, −pady => 60,  
−relief => 'raised',  
−background => 'LightBlue3',  
−activebackground => 'LightBlue4',  
−command => \&printout )  
−> pack(−side =>'right', −expand => 1);

MainLoop;

sub launch {
  if (−e "launchcam12.pl")  
    {  
      ## first remove the Tk screen
      $topwindow −> destroy if $topwindow;  
      ## $topwindow−> bell; # beeps if click window (p 296)
      system("perl ./launchcam12.pl");  
      system("perl ./tklaunch2.pl −geometry 300x400 −50−300");  
      $result = $dialog1 −> Show;  
      if ($result eq "OK") {};  
    }  
  else{  
    print "....ERROR: \n\n....can’t find program <launchcam12.pl>\n\n\n\nexit()  
  }  
}

sub quit {exit()}

sub printout {
  #$topwindow −> bell;  
  $result = $dialog1 −> Show;  
  if ($result eq "OK") {};  
}

sub projectteam {
  #$topwindow −> bell;  
  $result = $dialog2 −> Show;
7.4 Useful Linux tools to use with the launcher

In practice it may be easier to use many of the existing Xwindows utilities for displaying manual pages, examples, info and warnings etc. Note that the widget size and screen location can be easily controlled from the commandline using the \-geometry option. Check the relevant options by viewing the manpages for each of these utilities. Note there is a FullScreen option for Tk

\texttt{xclock}
\texttt{xman}
\texttt{xmessage}
\texttt{xdvi} (for viewing \texttt{.dvi} information pages)
\texttt{xdvi}
\texttt{xpdf}
\texttt{xghostscript} (for \texttt{.ps} files and \texttt{.pdf})
Chapter 9

The *launchcam12.pl* program

9.1 Introduction

This perl program is currently used to launch and coordinate the camomile system. It is launched from the perl/Tk widget. Note that currently the program coordinates the printing process by copying a lot of printing utility files into the `/project/pdata/` directory—this will change soon to keep all the printing tools (files) in a separate directory. The program currently performs the following actions.

**A** Create a time-encoded project directory name `$projdir` for the operation. This is achieved by passing the current `$localtime` to the subroutine `tedname()`. This directory name is also passed to the camomile program as a command-line option (to force camomile to create this particular base directory name for the operation). We add the forward slash to the end of the directory name in order to allow the camomile program to create the `fields` subdirectory (for its output of `.binlog` data files).

```perl
$timenowgmt = localtime;
$projdir=tedname($timenowgmt);
$projdir=$projdir."/";
```

**B** Call the camomile program using command-line switches for automatic startup (`-A 1`), Path (`-P`), and configuration file (`-c`) respectively, as follows (need to make sure that everything is all on one line). Note that we also pass the string `$projdir` to the camomile startup command and make camomile itself create the new project directory. Camomile then places all its output data files into the directory `/$projdir/fields/`

```bash
...../camomile -A 1 -P $projdir -c ../conf2/c_as3rn.conf
```

**C** Now write the start-time (in unixtime and localgmttime formats) to a new specially created file `<starttime.dat>`, which we write to a new data directory `/projdir/pdata/`, and is used to facilitate data processing and printing.
Note that we have to wait until Camomile terminates since the starttime and project directory name are determined immediately before starting Cammomile (see A). The time written to the <starttime.dat> file then indicates the “zero” time reference for all subsequent data processing and graphs.

```
open (outfile1, ">$destinationfilename1")
    ||die "ERROR: can't create file <starttime.dat>
";
print (outfile1 "%% file name: startfile.dat: created $timenowgmt"
);
print (outfile1 "%% file generated by <launchcam.pl> RWD Nickalls"
);
print (outfile1 "%% file read by <plotgnnk2.pl> \n"
);
print (outfile1 "projectdir,$projdir"
);##use commas no spaces
print (outfile1 "starttime,$timenowunix,$timenowgmt"
);##no spaces
close (outfile1);
```

We now copy all the print-tools utility files to the /project/pdata/ directory in preparation for data processing and printing.

```
system ("cp -v ./printfiles/*.* $projpdatadir"
);
```

We now process all the output files from camomile by calling the utility program plotgnnk.pl.

```
chdir $projpdatadir;
system ("perl ./plotgnnk2.pl"
);
```

We now print out all the .dvi files in reverse order by calling the utility printall.pl. (these constitute the printed Anaesthesia Record).

```
if (-e "printall.pl")
{print "... sending data to the printer now.....\n";
system ("perl printall.pl"
);
print "... done\n\n"}
else
{print "ERROR...can't find program <printall.pl>\n"
}
else
{print " returning to original dir now.....\n\n"};
```

Finally, we return to the original directory

```
$returndir="/home/dick/allfiles/camomile/top/datexsim"
chdir $returndir;
print "\n******************************************************************\n\nFINISHED\n\n******************************************************************\n\n";
```
9.2 The program launchcam12.pl

```perl
#!/usr/bin/perl
## launchcam12.pl
## CALLed by the Tk frontend widget (tklaunch2.pl)
##
## April 10, 2004
## for launching camomile and the printing program
##
## RWD Nickalls
## works well
##
## 1. new version to use Simon’s new camomilefields2tex version
##
## 2. new version to use Simon’s new camomilefields2tex version
##
## now create the projdir as a timerelated filename
## call the SUB tedname to generate the projdirname
## format of tedname = /home/dick/allfiles/camomiletop/theatredata/$date
## we pass the timenowGMT value to the tedname
##
print "calling [sub tedname] for time encoded dirname
";
$projdir=tedname($timenowgmt);
## remember to add the / at the end of the dir (so Camomile makes the /fields dir
## as a subdirectory
$projdir=$projdir="/";
print " ... time encoded directory made OK (=$projdir)
";

#### for testing========================
#### use this for testing with the dir cam1404fields
#### for 1240 test --
####
#### now create the projdir as a timerelated filename
#### call the SUB tedname to generate the projdirname
#### format of tedname = /home/dick/allfiles/camomiletop/theatredata/$date
####
print "... making a time encoded base directory \n"
# grab the start time
$timenowgmt = localtime;
$timenowunix= time();

# run camomile here from /camomiletop/datexsim/
# keep everything on single line
$campath = ”/tarballs/camomile–0.1.040411/camomile/camomile”;
```
system("$campath -A 1 -P $projdir -c ../conf2/c.as3rn.conf");

#####Camomile has terminated===============
#####so we tidy up, process all the data (make new directory etc),
#####and return control to launch widget

print "\n- - - - - - - - - - - - - - - - - - -\n\n... end of camomile program\n";

## flush the buffers after Camomile just to be sure
system ("sync");

## return to <launchcam>
print " ....... returning to <launchcam.pl>\n\n"

## now create and write the <starttime.dat> file
## since the base dir (project dir) for output is created by Camomile
## we have to wait until camomile terminates before sending
## the <starttime.dat> file to the new /projdir/pdata/ dir
## which will contain all the NEW processed data
## (all the original collected data is in the /projdir/fields/ directory)
## <starttime.dat> file only needed for the printing, ie after running Camomile
## write the starttime file to the /projdir/pdata/ dir

## first need to create the new /pdata/ dir
$projpdatadir=$projdir."pdata/";
mkdir $projpdatadir;

## now write the starttime.dat file into the /pdata/ directory
print "writing the <starttime.dat> file to pdata dir ....\n";

$destinationfilename1=$projpdatadir."starttime.dat";##
print " <starttime> destinationfilename1 = $destinationfilename1\n";
open (outfile1,">$destinationfilename1")||die "ERROR: can't create file
<starttime.dat>\n";

##
print (outfile1 "%% file name: startfile.dat: created $timenowgmt\n");
print (outfile1 "%% file generated by <launchcam.pl> RWD Nickalls\n");
print (outfile1 "%% file read by <plotgnnk2.pl>\n");
print (outfile1 "projectdir,$projdir,$timenowunix,$timenowgmt\n");##no spaces
close (outfile1);

##
print " ......<starttime.dat>.... done\n";
#

## now copy all the <printfiles> tools to the /projdir/pdata/ dir
print "copying files from ../datexsim/printfiles/ to ../project/pdata/ directory\n";
system ("cp -v ./printfiles/ to ../project/pdata/ directory\n";
print "...... done\n";

###===============================================
### now start the (optional) printing process
## now move to the project/pdata/ dir to CALL the print prog <plotgnnk2.pl>
print " moving dir -- > $projpdatadir\n";
cd $projpdatadir;
print "the new dir is: ...\n";
system ("pwd");

## now start running the printing process by running <plotgnnk.pl>
print "... now calling perl ./plotgnnk2.pl \n";
system ("perl ./plotgnnk2.pl");

##==================
## finally copy the starttime file to the base dir for safekeeping
print "... now copying file <starttime.dat> to /project/ dir \n";
system ("cp −v starttime.dat ..");
goto jump;
##==========print OPTION=====================
print " \n−−−−−−−−−−−−−−−−−\n\n Press P to PRINT results [q to quit]: ";
$p = <STDIN>, chomp $p;
##imortant here to remove the <CRLF>
if (lc($p) eq "p") {
  ## check program exists
  if (−e "printall.pl") {
    print "... sending data to the printer now....\n";
    system ("perl printall.pl");
    print "... done\n\n"}
  else {
    print "ERROR...can’t find program <printall.pl>\n"
  }
} else {
  print " returning to original dir now....\n\n";
##=====================================================
jump;
##===============SUB=================
## note that the <sub> keyword must be lowercase
sub subname{
  ## returns a date/time encoded filename -- > $projdir;
  ## using the GMT start−time string passed as a parameter
  my $starttmstring=$[0];
  my $n= $#_ + 1;
print " [SUB] starttimestring = $startgmtstring \n";
print " [SUB] number of args passed = \$n\n";
## note the main items are <space> separated except hh:mm:ss
## format is: Sun Jan 25 13:24:35 2004
## format is: Sun Jan 5 13:24:35 2004
## note get two spaces after the Month if days <10
# if two spaces in posn 8 and 9 then remove one
if (substr($startgmtstring,7,2) eq " ")
  {substr($startgmtstring,7,2," ");}
print " [SUB] new translated string = $startgmtstring\n";
## now replace spaces with commas
$startgmtstring =˜ tr / /,/; /*
## make an array
@stgmt=split (/[,]/, $startgmtstring);
$day=$stgmt[0];
$month=$stgmt[1];
$date=$stgmt[2];
$hms=$stgmt[3];
$year=$stgmt[4];
$noitems=$#stgmt+1;
print " [SUB] ....orig string = $startgmtstring\n";
print " [SUB] ....extracted gmt part is: $day,$month,$date,$hms,$year\n";
print " [SUB] ....extracted starttime hh:mm:ss $hms\n";
## now extract the hh:mm:ss part to get the hh:mm
@hhmmss=split (/[:]/, $hms);
$hour=$hhmmss[0];
$min=$hhmmss[1];
# $sec=$hhmmss[2];
## force two--digit for date (= day--of--month)
## as unix gmt uses only 1 char if less than 10
if ($date<10)
  {$date="0".$date};
## format the datestring as 2004--01--22--1341
$datestring="$year-$month-$date-$hour$min";
return "/home/dick/allfiles/camomiletop/theatredata/"."$datestring";
};
._END_;
Part III

The data program—Camomile
Chapter 10

System overview

10.1 Introduction

The anaesthesia work-station accesses data from both the keyboard and the Datex AS/3 anaesthesia monitor. This data is processed and made available to the anaesthetist in various ways; for example, as trend data on the screen, as a printed Anaesthesia Record, as age-corrected MAC, and alarm and warning information. Other aids for the anaesthetist are in the form of ‘help’ files for decision support, access to an epidural and double-lumen tube database, and timers (e.g. use with diabetic patients as reminders for determining blood sugars and adjustment of insulin/glucose therapy).

The software is ‘open source’ and designed and written for the Linux operating system. For the purposes of description, the software components fall into the following categories.

- a graphical ‘front-end’ module for launching the various systems.
- a data collection and display module
- a printing module
- an epidural and double-lumen tube database
- an HTML ‘help’ module

These are now described briefly in turn.

10.2 Modules

10.2.1 Graphical front-end module

The graphical front-end ‘launcher’ (tklaunch2.pl) is a Perl/Tk program, which is itself launched by typing the command runcamomile in a BASH terminal window. Once launched, the Tk widget shows a number of buttons, each of which will launch an application, for example, the Camomile anaesthesia program, an epidural database program, a collection of ‘help’ files, and an on-line ‘user’ manual.
10.2.2 Data collection and display module

This is the heart of the Camomile system. It accesses data from the keyboard, mouse and the Datex AS/3 anaesthesia monitor. Raw data is accessed every 5 seconds from the Datex monitor via the serial port, and saved to the hard drive. The data is displayed in trend format (one screen width shows 30 mins of data), and processed in the form of alarms, log entries, and age-corrected MAC.

At the end of the anaesthetic the program is terminated by clicking on the ‘exit’ option from a pull-down menu, whereupon the graphical front-end is returned.

10.2.3 Printing module

At the end of the anaesthetic all the relevant data (the Anaesthetic Record) is printed out in a form suitable for inclusion in the patient notes. The printing process is initiated by clicking on the relevant button on the graphical front-end.

10.2.4 Epidural database

This is accessed from the front-end by clicking on the relevant button. It is a database incorporating epidural and double-lumen tube collected since 1995, and allows the anaesthetist to estimate for a given height and weight of a patient (a) the midline epidural depth and (b) length of the double-lumen tube.

10.2.5 Help files

This is a collection of HTML ‘help’ files of information useful to the anaesthetist. Much of the information is in the form of City Hospital guidelines, but guidelines from other sources are included.

10.3 Directory structure

The directory structure for Camomile is as follows.

```
/home/.../camomile/
/home/.../camomile/docs/
/home/.../camomiletop/
/home/.../camomiletop/aneshelp/
/home/.../camomiletop/conf2/
/home/.../camomiletop/datexsim/
/home/.../camomiletop/datexsim/printfiles/
/home/.../camomiletop/tarballs/
/home/.../camomiletop/tarballs/camomile-0.1_040411/
/home/.../camomiletop/tarballs/camomile-0.1_040411/admin/
/home/.../camomiletop/tarballs/camomile-0.1_040411/camomile/
/home/.../camomiletop/tarballs/camomile-0.1_040411/camomile/docs/
/home/.../camomiletop/tarballs/camomile-0.1_040411/camomile/docs/en/
/home/.../camomiletop/tarballs/camomilefield2tex-0.1_040411/camomile/
/home/.../camomiletop/tarballs/camomilefield2tex-0.1_040411/camomile/docs/
```
/home/.../camomiletop/tarballs/camomilefield2tex-0.1_040411/camomile/docs/en/
/home/.../camomiletop/tarballs/inc/
/home/.../camomiletop/tarballs/inc/port_datex_as3.h
/home/.../camomiletop/theatredata/
/home/.../camomiletop/theatredata/2004-Mar-05-1027/
/usr/local/bin/runcamomile
/usr/local/bin/camomilefield2tex
Chapter 11

The Camomile program

11.1 Directory listing of camomile.v.0.1_040413b

This is the directory listing of the final working version of the Camomile program (written by Simon Dales; compiled April 15, 2004).

dir listing of camomile.v.0.1_040413b[c-Apr-15-2004]/camomile/

------------
1279 Nov 20 2003 bell_off.xpm
1263 Nov 20 2003 bell_on.xpm
408 Feb 17 2003 browser_back.xpm
411 Feb 17 2003 browser_exit.xpm
409 Feb 17 2003 browser_frwd.xpm
424 Feb 17 2003 browser_home.xpm
408 Jun 17 2003 browser_reload.xpm
443496 Apr 13 2004 camomile
40924 Apr 11 2004 camomile.cpp
1843 Apr 11 2004 camomiledoc.cpp
1646 Feb 17 2003 camomiledoc.h
7422 Dec 8 2003 camomile.h
1518 Apr 11 2004 camomileview.cpp
1472 Feb 17 2003 camomileview.h
1279 Feb 17 2003 camomile.xpm
4879 Apr 11 2004 dAboutBox.cpp
1349 Apr 11 2004 dAboutBox.h
12749 Jun 15 2003 dAboutBox.ui
3199 Apr 11 2004 dDisplayDial.cpp
929 Apr 11 2004 dDisplayDial.h
7631 May 28 2003 dDisplayDial.ui
1561 Apr 11 2004 dDisplayGraph.cpp
803 Apr 11 2004 dDisplayGraph.h
2066 May 29 2003 dDisplayGraph.ui
1776 Apr 11 2004 dDisplayNow.cpp
1226 Apr 13 2004 tguisetups.h
6220 Apr 10 2004 thelplrowser.cpp
1628 Jun 23 2003 thelplrowser.h
4840 Apr 10 2004 ticonfactory.cpp
1640 Mar 6 2003 ticonfactory.h
2401 Apr 13 2004 tlogevent_device_event.cpp
1704 Apr 5 2004 tlogevent_device_event.h
2592 Nov 21 2003 tlookup_vapour.cpp
1314 Nov 17 2003 tlookup_vapour.h
1072 Mar 11 2003 tport.cpp
1311 Mar 16 2003 tport.h
7129 Apr 10 2004 tportserial.cpp
5964 Apr 10 2004 tportserial_datex_as3.cpp
1810 Dec 3 2003 tportserial_datex_as3.h
4024 Apr 10 2004 tportserial_graseby_3400.cpp
1716 Nov 4 2003 tportserial_graseby_3400.h
2486 Aug 29 2003 tportserial.h
7358 Apr 10 2004 tproject.cpp
3066 Apr 10 2004 tprojectdialog.cpp
1497 Mar 6 2003 tprojectdialog.h
3242 Apr 10 2004 tproject.h
1940 Aug 20 2003 tsampler_displaybase.cpp
1882 Aug 20 2003 tsampler_displaybase.h
2255 Aug 20 2003 tsampler_display_clock.cpp
1488 Aug 20 2003 tsampler_display_clock.h
15716 Apr 10 2004 tsampler_display_dial.cpp
1833 Dec 15 2003 tsampler_display_dial.h
20149 Apr 13 2004 tsampler_display_graph.cpp
2681 Apr 13 2004 tsampler_display_graph.h
2529 Apr 10 2004 tsampler_display_lcd.cpp
1506 Aug 20 2003 tsampler_display_lcd.h
5717 Apr 13 2004 tsampler_display_log.cpp
1438 Aug 20 2003 tsampler_display_log.h
17799 Apr 13 2004 tsampler_display_nickallsalarm.cpp
2130 Nov 24 2003 tsampler_display_nickallsalarm.h
8086 Apr 11 2004 tsampler_display_nickallsmac.cpp
1945 Nov 20 2003 tsampler_display_nickallsmac.h
18135 Apr 10 2004 tsampler_display_nickallsnow.cpp
1886 Nov 24 2003 tsampler_display_nickallsnow.h
12174 Apr 11 2004 tsampler_display_pumpcontroller.cpp
2135 Aug 21 2003 tsampler_display_pumpcontroller.h
3846 Apr 10 2004 tsampler_display_relaxants.cpp
1522 Aug 20 2003 tsampler_display_relaxants.h
28736 Apr 10 2004 tsampler_portbase_datex_as3.cpp
5505 Dec 1 2003 tsampler_portbase_datex_as3.h
17744 Apr 10 2004 tsampler_portbase_graseby_3400.cpp
2417 Aug 28 2003 tsampler_portbase_graseby_3400.h
4717 Apr 10 2004 tsampler_portbasewidget.cpp
2355 Apr 10 2004 tsampler_portbasewidget.h
5643 Apr 10 2004 twaffle.cpp
<table>
<thead>
<tr>
<th>File Name</th>
<th>Date</th>
<th>File Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>twaffle.h</td>
<td>Apr 10</td>
<td>2004</td>
</tr>
<tr>
<td>widgetfactory.cpp</td>
<td>Apr 10</td>
<td>2004</td>
</tr>
<tr>
<td>widgetfactory.h</td>
<td>Mar 16</td>
<td>2003</td>
</tr>
<tr>
<td>widgetfactory_port.cpp</td>
<td>Apr 11</td>
<td>2004</td>
</tr>
<tr>
<td>widgetfactory_Widget.cpp</td>
<td>Apr 10</td>
<td>2004</td>
</tr>
<tr>
<td>widgetsampler.cpp</td>
<td>Apr 11</td>
<td>2004</td>
</tr>
<tr>
<td>widgetsampler.h</td>
<td>Mar 27</td>
<td>2003</td>
</tr>
<tr>
<td>widgetTimeEntry.cpp</td>
<td>Apr 11</td>
<td>2004</td>
</tr>
<tr>
<td>widgetTimeEntry.h</td>
<td>Apr 11</td>
<td>2004</td>
</tr>
<tr>
<td>widgetTimeEntry.ui</td>
<td>Feb 17</td>
<td>2003</td>
</tr>
<tr>
<td>wRelaxants.cpp</td>
<td>Apr 11</td>
<td>2004</td>
</tr>
<tr>
<td>wRelaxants.h</td>
<td>Apr 11</td>
<td>2004</td>
</tr>
<tr>
<td>wRelaxants.ui</td>
<td>Jun 19</td>
<td>2003</td>
</tr>
<tr>
<td>wRunClock.cpp</td>
<td>Apr 11</td>
<td>2004</td>
</tr>
<tr>
<td>wRunClock.h</td>
<td>Apr 11</td>
<td>2004</td>
</tr>
<tr>
<td>wRunClock.ui</td>
<td>Jun 19</td>
<td>2003</td>
</tr>
<tr>
<td>wToolsA.cpp</td>
<td>Apr 11</td>
<td>2004</td>
</tr>
<tr>
<td>wToolsA.h</td>
<td>Apr 11</td>
<td>2004</td>
</tr>
<tr>
<td>wToolsA.ui</td>
<td>Feb 17</td>
<td>2003</td>
</tr>
<tr>
<td>wTools.cpp</td>
<td>Apr 11</td>
<td>2004</td>
</tr>
<tr>
<td>wTools.h</td>
<td>Apr 11</td>
<td>2004</td>
</tr>
<tr>
<td>wTools.ui</td>
<td>Apr 9</td>
<td>2003</td>
</tr>
</tbody>
</table>
Chapter 12

Configuration files

12.1 Introduction

All the configuration files are placed in the directory /camomiletop/conf2/. At present the hospital program uses only the customised 'RN' configuration files, e.g. c_as3rn.conf. The order that the configuration files are input is as follows.

- c_as3rn.conf
- x-configrn.conf ← projectdir.conf
- x-widgets.conf
- w-monitor-datexas3.conf
- x-displays.conf

The list of configuration files is as follows.

camomile.sty
c_as3.conf
c_as3rn.conf
c_g3400_ro.conf
c_g3400_rw0.conf
projectdir.conf
u-drugs.conf
u-drugrn.conf
u-people.conf
u-peoplern.conf
u-pumpable.conf
w-display-relaxant.conf
w-monitor-datexas3.conf
w-pumpcontroller-bozo.conf
w-pumpcontroller-nickalls.conf
w-pump-graseby3400.conf
x-config.conf
x-configrn.conf
x-displays.conf
12.2  c—as3rn.conf

\begin{verbatim}
\%&LaTeX
\%!camomile
\%!OnOff: (beginCamomileConfig,endCamomileConfig)
\%!EndCamomileComments
\------------------------------
\documentclass[a4paper]{article}
\usepackage{geometry}
\geometry{hscale=0.8,vscale=0.85}
\nofiles
\voffset=-72bp
\oddsidemargin=30bp
\headheight=20bp
\headsep=5bp
\textwidth=450bp
\textheight=770bp
\oddsidemargin=-10bp
\usepackage{camomile}
\def\docName{Camomile Configuration file © 11/4/3}
\def\S#1{\section{#1}}
\def\SS#1{\subsection{#1}}
\def\SSS#1{\subsubsection{#1}}
\def\FN#1{{\tt #1}}
\def\fN#1{{\tt #1}}
\def\set#1#2{SET[#1][#2]}
\pagestyle{headings}
\makeindex
\begin{document}
\docName
\tableofcontents
\end{verbatim}
CHAPTER 12. CONFIGURATION FILES

\S{Introduction}

This is a configuration file for Camomile. It is laid out in \TeX{} so that we can do some form of literate programming. The alternative could be XML, or look at sendmail’s configuration file.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%\newpage
%S{Configuration}

Notes:

\begin{itemize}
\item screen dimensions in nominal units. On original screen measure off in whatever units you find convenient (mm, bp etc). When you port this configuration to another size of monitor, adjust the \fN{pixelsize} parameters.

It is probably best to set your \fN{pixelsize} parameters to an initially sensible value, say 1000, then adjust from there.
\end{itemize}

\newpage
\begin{CamomileConfig}
% \comment{\newpage
\SS{Configure Application}
}
% \SetCamomileIncludePath{/home/dick/allfiles/camomiletop/conf2/}
\newdict
% \set{path.config}{/home/dick/allfiles/camomiletop/conf2/}%
% \set{class}{main}%
% \newinstance%
% \popdict
% %\include{x-config.conf}
% \include{x-configrn.conf} %%% Nickalls
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% windows
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% \include{x-widgets.conf}
% % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% \% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% ::monitors
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\include{x.monitors.conf}
\include{w-monitor-datexas3.conf}
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% ::Pumps
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\include{x.pumps.conf}
\include{xx.bozo_controller.conf}
\include{xx.nickallscontroller.conf}
\include{xx.graseby3400.conf}
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% displays
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\include{x-displaysrn.conf}  %%% Nickalls
\include{x-displays.conf}
%
\endCamomileConfig
\newpage
\S{More Waffle}
\end{document}
%%%eof

12.3 x-configrn.conf

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% config paths
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\comment{\newpage
 \SS{Configure paths}
 }
\newdict
%
\set{path.config}{/home/dick/allfiles/camomiletop/conf2/}%
\set{path.help.base}{/home/dick/allfiles/camomiletop/docs/help/en/index.html}%
\set{path.help.cribsheet}{/home/dick/allfiles/camomiletop/aneshelp/index.html}%
\set{path.help.diabetescrib}{/home/dick/allfiles/camomiletop/aneshelp/diabetes.html}%
\set{path.project.wd}{/home/dick/allfiles/camomiletop/theatredata}%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%------------------
%%% rwdn  Feb 17 2004 now reads in both paths
%%%\set{path.project.format}{/home/dick/allfiles/camomiletop/theatredata/test/!Y-!M-!D-!h!m}
\include{projectdir.conf}  %%% has the new dirs from launchcam.pl
%------------------
\set{title.project.format}{Operation(\!Y-\!M-\!D\!h:\!m:\!s[\!S,\!W])} \%
\set{app.htmlbrowser}{konqueror \%s} \%
\set{class}{main} \%
\newinstance \%
\popdict \%
\comment{\newpage \SS{Configure Dialogs} \}
\newdict \%
\set{class}{lists} \%
\set{subclass}{people} \%
\include{u-peoplern.conf} \%
\% \%
\newinstance \%
\%
\popdict \%
\%
\newdict \%
\set{class}{lists} \%
\set{subclass}{drugs} \%
\include{u-drugsrn.conf} \%
\% \%
\newinstance \%
\%
\popdict \%
\%eof

12.4 projectdir.conf

%%% projectdir.conf: created Mon Mar 1 19:15:50 2004
%%% file generated by <launchcamX.pl> RWD Nickalls
%%% this file to be \input{} by /conf2/x-configRN.conf
\set{path.project.format}{/home/dick/allfiles/camomiletop/theatredata/2004-Mar-01-1915/}
%%% ------------

12.5 w-monitor-datexas3.conf

%%% widgets.conf
%%% mods:
%%% 11/4/3: initial
%%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Ports
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\comment{\newpage
SSS{Datex AS/3}
}
% 
\newdict
\set{widget.parent}{widget.monitors}
\set{class}{port}
%
\set{port.parity}{E}
\set{port.stopbits}{1}
\set{port.databits}{8}
\set{port.baud}{19200}
\set{reader.rate}{5000} % read at 5s/block
\set{widget.x}{2}
\set{widget.y}{2}
\set{widget.h}{250}
\pushdict
% 
\set{widget.parent}{widget.port.monitor.0}
% 
% \set{subclass}{TPort.Datex.AS3.v0.1}
% 
\set{device}{/dev/ttyS0}
% \set{device}{/dev/ttyS1}
% request start 0 = no, 1 = yes
\set{request.start.send}{0}
% request stop 0 = no, 1 = yes
\set{request.stop.send}{0}
% request stop period 0,-1, whatever
\set{request.stop.period}{0}
% 
\set{name}{First Datex}
\set{logfile}{datex0.dat}
% 
\set{param.sat.sat}{sat}
\set{param.inv[0].s}{bp.s}
\set{param.inv[0].d}{bp.d}
\set{param.ecg.hr}{ecg.hr}
\set{param.sat.hr}{sat.hr}
\set{param.ecg.rr}{ecg.rr}
\set{param.o2.insp}{o2.insp}
\set{param.inv[1].m}{cvp}
% 
\set{param.co2.exp}{co2.exp}
\set{param.co2.insp}{co2.insp}
12.6 People.conf

%% people
\add{anaesthetist}{Dick Nickalls}
\add{anaesthetist}{Ken Alagesan}
\add{anaesthetist}{Pam Wade}
\add{anaesthetist}{Ndu Okonkwo}
\add{anaesthetist}{Janet Latter}
%
\add{surgeon}{Ellis Morgan}
\add{surgeon}{David Beggs}
\add{surgeon}{John Duffy}
%
%%%eof

12.7 Drugs.conf

% drugs conf
\add{drugname}{Asprin}
\add{drugname}{Ephedrine}
\add{drugname}{Frusemide}
\add{drugname}{Morphine}
\add{drugname}{Propofol}
\add{drugname}{Remifentanil}
\add{drugname}{Vecuronium}
12.8 x-widgets.conf

% x-widgets.conf

\comment{
\newpage
\SSS{Widgets}
This file should be largely static for a site.
Draws the window widgets
}
%
%
%%%%%%%%%%%%%%%%%%%
%
\newdict
%
%\%x%\set{logfiles}{/projects/apple2/camomile/}%
%\%x%\set{app.name}{Camomile Data Display}%
%
%\set{display.period}{10001} % update every 10s%
%\set{display.period}{10000} % update every 10s%
%\set{display.period}{3000} % update every 10s%
%\set{display.period}{200} % update every 10s%
%\set{display.period}{2000} % update every 10s%
%\set{display.period}{1000} % update every 1s%
%\set{display.period}{100} % update every 100ms%
%
%\set{pixel.size.x}{3.1234}%
%\set{pixel.size.y}{2.418}%
%\set{pixel.offset.x}{0}%
%\set{pixel.offset.y}{-517}%
%
%\set{widget.x}{0}%
%\set{widget.y}{0}%
%\set{font.size}{10}%
%\set{widget.w}{1015}%
%\set{widget.h}{700}%
%
%\set{class}{main}%
%\newinstance%
%
\popdict
% setup some windows
\newdict
\set{widget.parent}{main}
\set{class}{window}
%
\pushdict
%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% top window
%%%%%%%%%%%%%%%%%%%%%%%%%%%
\set{widget.name}{widget.top}
\set{widget.x}{0}
\set{widget.y}{0}
\set{widget.h}{300}
\set{widget.w}{1015}
\set{fixed}{1}
\newinstance
\popdict
%
\pushdict
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% text window
%%%%%%%%%%%%%%%%%%%%%%%%%%%
% setup text window
\set{widget.name}{widget.text}
\set{widget.x}{900}
\set{widget.w}{115}
%
\set{widget.y}{300}
\set{widget.h}{322}
\newinstance
\popdict
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% bottom window
%%%%%%%%%%%%%%%%%%%%%%%%%%%
% setup bottom window
\pushdict
\set{subclass}{tabbedwindow}
\set{widget.w}{900}
%
\set{widget.x}{280}
\set{widget.h}{346}
\set{widget.name}{widget.bottom.big}
\newinstance
\popdict
% setup bottom window
\pushdict
  \set{widget.title}{\&Main}
  \set{widget.parent}{widget.bottom.big}
  \set{widget.name}{widget.bottom.frame}
  \newinstance
\popdict
%
\pushdict
  \set{widget.title}{ZZBottom}
  \set{widget.parent}{widget.bottom.frame}
  \set{subclass}{tabbedwindow}
  \set{widget.x}{100}
  \set{widget.w}{650}
  \set{widget.h}{550}
%
  \set{widget.y}{0}
  \set{widget.h}{322}
  \set{widget.name}{widget.bottom}
  \newinstance
\popdict
%
\pushdict
  % setup bottom tabbed window
  \set{widget.parent}{widget.bottom}
  \pushdict
    \set{widget.parent}{widget.bottom.big}
    %
    \set{widget.title}{\&Gases}
    \set{widget.name}{widget.gases}
    \newinstance
  \popdict
%
  % setup bottom tabbed window
  \set{widget.title}{\&Alarms}
  \set{widget.name}{widget.alarms}
  \newinstance
%
  % setup bottom tabbed window
  \set{widget.title}{\&Logs}
  \set{widget.name}{widget.logs}
  \newinstance
%
  % setup bottom tabbed window
  \set{widget.title}{Warning \&Robots}
  \set{widget.name}{widget.warningRobots}
  \newinstance
%
  % setup bottom tabbed window
\%\set{widget.title}{&Calculators}
\%\set{widget.name}{widget.calcs}
\%\newinstance
\%
% setup bottom tabbed window
\%pushdict
% \set{subclass}{tabbedwindow}
\set{widget.title}{Monitor\&s}
\set{widget.name}{widget.monitors}
\newinstance
%
\set{widget.title}{P\&umps}
\set{widget.name}{widget.pumps}
\newinstance
\%popdict
%
% setup bottom tabbed window
\set{widget.title}{\&Other Stuff}
\set{widget.name}{widget.otherstuff}
\newinstance
%
\popdict
\popdict
%
%
% more windows here
\popdict
\%\%eof
Chapter 13

Drug dictionary

13.1 Introduction

The drug dictionary listing used in the pull-down menu of drugs (and IV fluids) was derived from the NHS Dictionary of Medicines and Devices (DM+D) website (a username and password are required). The listing we used was the Virtual Therapeutic Moiety (VTM) database, and was downloaded every few weeks. This very comprehensive listing is added to periodically by the NHS, and is intended to be ultimately a list of all drugs and associated European-wide numeric codes for use in the NHS. In 2006 this list consisted of approximately 1800 drugs and drug combinations.

Figure 13.1: Screenshot showing the pull-down menu and the drug Bupivacaine selected.
13.2 Initial drug list

The drug list uploaded to the workstation was: *u-drugs.conf*, a typical example from June 2003 being as follows.

```plaintext
% canomile conf
% drugs01.cfg (15 June, 2003)
\add{drugname}{Adrenaline}
\add{drugname}{Alfentanil}
\add{drugname}{Atracurium}
\add{drugname}{Atropine}
\add{drugname}{Bicarbonate 8.4\%}
\add{drugname}{Blood (packed cells)}
\add{drugname}{Blood (whole)}
\add{drugname}{Cefuroxime}
\add{drugname}{Cisatracurium}
\add{drugname}{Dexamethasone}
\add{drugname}{Dextrose 5\%}
\add{drugname}{Diamorphine}
\add{drugname}{Digoxin}
\add{drugname}{Ephedrine}
\add{drugname}{Erythromycin}
\add{drugname}{Etomidate}
\add{drugname}{Fentanyl}
\add{drugname}{FFP}
\add{drugname}{Furosemide}
\add{drugname}{Gelofusin}
\add{drugname}{Glycopyrrolate}
\add{drugname}{GTN}
\add{drugname}{Hartmans solution}
\add{drugname}{Heparin}
\add{drugname}{HESPAN}
\add{drugname}{Hydrocortisone}
\add{drugname}{Isoprenaline}
\add{drugname}{Metaraminol}
\add{drugname}{Methoxamine}
\add{drugname}{Metronidazole}
\add{drugname}{Morphine}
\add{drugname}{Noradrenaline}
\add{drugname}{Normal Saline}
\add{drugname}{Phenylephrine}
\add{drugname}{Potassium}
\add{drugname}{Propofol}
\add{drugname}{Protamine}
\add{drugname}{Remifentanil}
\add{drugname}{Rocuronium}
\add{drugname}{Salbutamol}
\add{drugname}{Saline 0.9\%}
\add{drugname}{SNP}
\add{drugname}{Suxamethonium}
```
However, I started writing some Perl programs to extract and process the NHS listing which could be downloaded from the DM+D website.

### 13.3 Download bundle

Each download bundle had a filename something like `week192006-r2_3.zip` (i.e., the bundle for week 19, 2006), consisting of the following files.

- `amp_v2_3.xsd`
- `amp_v2_3.xsd`
- `BNF`
- `f_amp2_3110506.xml`
- `f_ampp2_3110506.xml`
- `f_ingredient2_3110506.xml`
- `f_lookup2_3110506.xml`
- `f_vmp2_3110506.xml`
- `f_vmpp2_3110506.xml`
- `f_vtm2_3110506.xml`
- `ingredient_v2_3.xsd`
- `letters`
- `lookup_v2_3.xsd`
- `vmpp_v2_3.xsd`
- `vmp_v2_3.xsd`
- `vtm_v2_3.xsd`

### 13.4 VTM File format

The `f_vtmXXX.xml` database (114 KB in this particular case) is an XML formatted database of about 1800 drugs and drug combinations (week 19, 2006).

```xml
<?xml version="1.0" encoding="utf-8" ?>
<VIRTUAL_THERAPEUTIC_MOIETIES xsi:noNamespaceSchemaLocation="vtm_v2_2.xsd"
 xmlns="" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
<!-- Generated by Prescription Pricing Authority -->
<VTM><VTMID>68088000</VTMID><NM>Acebutolol</NM></VTM>
<VTM><VTMID>90332006</VTMID><NM>Paracetamol</NM></VTM>
<VTM><VTMID>33664007</VTMID><NM>Acetazolamide</NM></VTM>
<VTM><VTMID>108974006</VTMID><NM>Abciximab</NM></VTM>
<VTM><VTMID>109077006</VTMID><NM>Acarbose</NM></VTM>
<VTM><VTMID>398910009</VTMID><NM>Acebutolol + Hydrochlorothiazide</NM></VTM>
<VTM><VTMID>329923004</VTMID><NM>Aceclofenac</NM></VTM>
<VTM><VTMID>116084008</VTMID><NM>Abacavir</NM></VTM>
....
....
```
CHAPTER 13. DRUG DICTIONARY RWD Nickalls

<VTM><VTMID>9835811000001101</VTMID><NM>Medium-chain triglycerides + Soyaoil</NM></VTM>
<VTM><VTMID>9835911000001106</VTMID><NM>Dornase alfa</NM></VTM>
<VTM><VTMID>9836011000001103</VTMID><NM>Drotrecogin alfa</NM></VTM>
<VTM><VTMID>9837611000001107</VTMID><NM>Homeopathic cocculus indicus</NM></VTM>
</VIRTUAL_THERAPEUTIC_MOIETIES>

13.5 Perl program dn-dmd5.pl

#!/usr/bin/perl
# dn−dmd5.pl (modified from dn−dmd4.pl)
# to accommodate the <INVALID> tag
# RWDN May 14, 2006
## to read the xml VTM files to extract the drug names and codes
## search the vtm.XSD file for all the key TAGS, and then extract these from the VTM.xml file
## search for the other possible key words
## search for the <INVALID> ... </INVALID> ** done
## search for the <ABBREVNM> ... </ABBREVNM>
## set up commandline flags —— in etc
## so user can specify input and output filenames etc
use strict;
use warnings;
use Carp; # allows croak "" and warn "" (warn always —> to STDERR)
use Fatal qw(open close); # for errors
use Perl6::Builtins qw(system );
use Getopt::Long; # for commandline stuff
use version;

#================================
# create a printer—log file
open my $logg, "">", "dn−dmdlog.log"||die "ERROR: can’t open dn−dmdlog.log file\n";
#--- grab current time
my $time_now_unix=time(); # seconds
my $time_now_string=localtime($time_now_unix);
print $logg "malarm.log, 
$time_now_string," Unix=",$time_now_unix,"\n";
print $logg "log of my Perl rnalarm3.pl program \\n";
#================================

#--- set up the in and out files---

#open my $in file," <", "test−vtm23.txt"||die "ERROR: can’t open INfile \n";
###open my $in file," <", "dn−ftom2xml.txt"||die "ERROR: can’t open INfile \n";
## w50−2005
open my $infile, "<", "vtm23-w19y2006xml.txt" || die "ERROR: can’t open INfile \n";
#
open my $outfile, ">", "test-out-vtm.txt" || die "ERROR: can’t open OUTfile \n";
open my $outfile2, ">", "test-outsorted.txt" || die "ERROR: can’t open OUTsorted file \n";

## now read each line in the file, and place parameters into an array
print "...reading the infile file line–by–line\n";
print ($logg) "...reading the infile file line–by–line\n";

## set the eventFLAG
my $eventnumber=0; # counts the number of drug/value pairs in the dictionary
my $eventFLAG="OFF";

# initialise variables
my $uid1 = 0;
my $uid2 = 0;
my $uid = "";
my $begincode="<VTM>";
my $endcode="</VTM>";
my $len = 0;
my $f1=0;
my $m1 = 0;
my $e1=0;
my $invalid=0;
my $invalidflag="OFF";
my $dataline="";
my $newline="";
my $p=" + ";
my $drugcode="";
my $drugname="";
my $delement;
my $REVstring;

# define the Unsorted drugname array
my @udrugname = ();

# define the hash (for drugname/drugcode pairs
my %dmd=();

#-------------------
LINE:
while (<$infile>){
    next LINE if /#/; #skip # comments
    next LINE if /%/; #skip % comments
    next LINE if /$/; #skip blank lines
    # grab the whole line as a string
    $newline = $;
    # append the newline string to any remaining dataline fragment
    # when we start a new line
}
$\text{chomp}($\text{dataline}); # removes the line-ending

# reset variables to zero
$suid1 = 0;
$suid2 = 0;
$suid = "";
$f1 = 0;
$m1 = 0;
$e1 = 0;

#### @value=split (/[,]/, $\text{dataline}); # print $\text{dataline};
## replace CR/LF/space/ with visible chars = newbuffer
# $\text{dataline} = \text{chomp}($\text{dataline});
# $\text{dataline} = s/\n/<\text{LF}>/;
# $\text{dataline} = s/<\text{SPACE}>/;
## print $\text{dataline}, "\n"

LINEA:
if ($\text{dataline} = m/<\text{VTM}>/) {
    if ($\text{eventFLAG} eq "ON") { print "FLAG is still ON\n"; }
    else { $\text{eventFLAG}="ON", print "FLAG=ON\n"; }
}

if ($\text{dataline} = m$/\text{endcode}/) {
    $\text{eventFLAG}="OFF", print "FLAG=OFF\n";
    ## now analyse the event string to find UID and TEXT
    print "NEW endcode found / starting to extract the name/SNOMEDcode pair --\n";
    ## increment event counter
    $\text{eventnumber} = $\text{eventnumber} + 1;
    $\text{len} = \text{length}($\text{dataline});
    print "len dataline = ", $\text{len}, "\n";
    # print "\n$\text{dataline} = \n";
    print "string number = ", $\text{eventnumber}, "\n";
    ## process the event string to locate begin and end codes
    ## get the index positions for UID and SEQUENCE
    $\text{suid1} = \text{index} $\text{dataline}, '<\text{VTM}>';
    $\text{suid2} = \text{index} $\text{dataline}, '</\text{VTM}>';
    print "uid1 = ", $\text{suid1}, "\n";
    print "uid2 = ", $\text{suid2}, "\n";
    $\text{suid} = \text{substr}($\text{dataline}, ($\text{suid1}), ($\text{suid2}+6)-$\text{suid1});
    ## print this string to outfile
    print "UID = ", $\text{suid}, "\n";
}

# dissect out the front, middle, end parts of the string $\text{suid}.
$s1 = \text{index} $\text{suid}, '<\text{VTM}><\text{VTMID}>';
$s2 = \text{index} $\text{suid}, '</\text{VTM}>';
$s3 = \text{index} $\text{suid}, '</\text{NM}>';
## detect the <INVALID> tag

```perl
$invalid = index $uid, '<INVALID>,'
# if find <INVALID> then remove the current string segment and get next line
if ($invalid > 1) {
    $invalidflag="ON";
    print "<INVALID> tag found\n"
    print "invalid FLAG = ON\n";
    goto REMOVE;
}
```

```perl
$drugcode = substr($uid, 12, ($m1-12)); #OK
print "drugcode = ",$drugcode,"\n";
$drugname = substr($uid, ($m1+12), ($e1-($m1+12))); #OK
print "drugname = ",$drugname,"\n";
```

```perl
# print new format to outfile
## this is actual Unordered contents of VTM file
print { $outfile }
"</eventnumber,"</drugname,"</drugcode>\n";
```

```perl
# collect all the drugname(s) into an Unsorted array (so we can sort it later)
push (@udrugname, $drugname);
```

```perl
# check drugname for + reverse, and add to listing
if ($drugnames=~m/[+]/) {
    print "YES the string has a +\n"
    ## make array of words separated by space [+]
    my @words = split (/\+[\-]+/; $drugname);
    # clean out/remove leading and trailing white space from each string
    my @clean_words = ();
    foreach $delement (@words) {
        $delement =~ s/^\+//; #remove leading white space
        $delement =~ s/\+$//; # trailing space
        push (@clean_words, $delement);
    }
    my ($w1,$w2,$w3,$w4,$w5,$w6,$w7) = @clean_words;
    my $n = $#clean_words+1;
    print "n= ",$n,"\n";
    print " string = ",$drugname,"\n";
    if ($n == 2) {
        ## reverse the order
        $REVstring = $w2.$p.$w1;
        print "REVstring = ",$REVstring,"\n"
        $drugname = $REVstring;
        push (@udrugname, $drugname); # add to the Unsorted drugname array
        %dmd = (%dmd, $drugname, $drugcode); # add new name/code pair to the hash
    }
    elsif ($n==3) {
```

```perl
```
```
```perl
# only need to have each item first once

$REVstring = $w2.$p.$w1.$p.$w3;
print "REVstring = ", $REVstring,"\n";
$drugname= $REVstring;
push ( @udrugname, $drugname); # add to the Unsorted drugname array
%dmd = (%dmd, $drugname, $drugcode); # add new name/code pair to the hash

$REVstring = $w3.$p.$w1.$p.$w2;
print "REVstring = ", $REVstring,"\n";
$drugname= $REVstring;
push ( @udrugname, $drugname); # add to the Unsorted drugname array
%dmd = (%dmd, $drugname, $drugcode); # add new name/code pair to the hash
}
elseif ($n==4){
## no strings with 3 + as yet
print "first= ",$w1,"\n";
print "second= ",$w2,"\n";
print "third= ",$w3,"\n";
print "4th= ",$w4,"\n";
}
else {croak "ERROR: string NOT processed as n+ = ",$n,"\n";
}
# else {print "NO the string has no + \"\n"};

#==============

REMOVE:
#---------------------
## remove last string from the current dataline
$dataline= substr($dataline, ($uid2+6), (length ($dataline) − length($uid)) );
## print $uid if invalid tag found
if ($invalidflag eq "ON") {print "string not processed\n";
_invalidflag="OFF";
print "invalid FLAG = OFF\n";
}
# print "**dataline = ",$dataline,"\n";
# sleep 1;
print "-----\n";
#--------now look for next string pair------------------------
print "looking for the next event\n";
goto LINEA;

## when fall off end of string, then look for next string
print "** ERROR looking for new line/string\n";
warn "ERROR must have a problem here as should not get here\n";
### must have a problem here as should not get here
next LINE;
```
## finally dump the event string and start again

```
# when fall of end of string, while still looking for the endcode then get another
# line/string
print "**looking for new line/string (can't find endcode)\n";' next LINE;
```

```
# print "***", $dataline, "\n";
} # end of the input loop reading the {FILE}
```

```
print "no more events found - terminating now\n";
print "---\n";
# print "event string = ", $event, "\n";
```

```
# now add missing drugs (if they do not already exist on the VTM list)
# make this be input from a file
# collect name/code pairs into a hash
## need to make this a subroutine which reads the names from a local list

### add drugs from the LOCAL file

```
print {FILE} "....adding drugs from the LOCAL list\n";
print {FILE} "---\n";
my $addname="";
my $addnamecode="---";

# open the input file
open my $datafile, "<", "dn-drugs2add.dat" || die "ERROR: can't open
drugs2add.dat file\n";

$newline="";
$dataline="";

LINE2:

while (<$datafile>){
    next LINE2 if /#/; # skip # comments
    next LINE2 if /%/; # skip % comments
    next LINE2 if /$/; # skip blank lines
    # grab the whole line as a string
    $newline = $.
    chomp($newline); # removes the line-ending
    # split up the line if = present
    my @drugs = split (/=/, $newline); #
    my @clean_drugs =();
    foreach $delement (@drugs) {
        $delement =~ s/\s+//; # remove leading white space
        $delement =~ s/\s+$//; # trailing space
        push ( @clean_drugs, $delement);
```
my ($drug1, $drug2) = @clean_drugs;
$saddname = ucfirst $drug1; ## force Uppercase first letter (ucfirst)

## if a synonym (drug1) is given for existing drug using = sign (= drug2) then
## grab the correct snomed code for drug2, and use it with the synonym
if ($#clean_drugs > 1) {
    $drug2 = ucfirst $drug2; ## force first letter to be Ucase
    # at least two drugs in the input line
    print $logg "drug1 = {"$drug1,"} drug2 ={"$drug2,"} \n;";
    # check we can actually find the snomed code
    if (exists($dmd{$drug2})) {
        $addnamecode = "−−−−\ $dmd{$drug2}."−−−;
    } else {
        print $logg "** can’t find synonym ",$drug2,"\n;"
        $addnamecode = "***ERROR***;"
    }
}

## if only single name given, then just add it to list without snomed code
## use code <−−−> so we can see which entries are added by us
if (exists($dmd{$saddname})) {
    print $logg "dmd{$saddname} = ",$dmd{$saddname}, "\n;"
    print $logg "dname, " *** is ALREADY on the VTM list\n;"
    # print "** = ",$dmd{$saddname},"\n;"
    # print "$logg "$plantimate, " *** is ALREADY on the VTM list\n;"
}
else {
    $dmd = (%dmd, $saddname, $addnamecode); # add new drugname/drugcode pair to the hash
    push (@udrugname, $saddname); # add new drug only to the Unsorted drugname array
    print $logg " ",$saddname, " has been put on the list just now\n;"
    print "$saddname, " has been put on the list just now\n;"
}

## reset the addnamecode to the default
$addnamecode = "−−−;"
print $logg "−−−−−−−−−−−−−−−−−−−\n;"

close ($datafile);

###------------------------
# now print out the arrays and hashes as a check
BBook p 74; works OK
my $key; # the drug name
my $value; # the Snomed code
while (( $key, $value ) = each (%dmd)) {
    print "$key => $value\n";
    # sleep 1;
    }
}

# now print the Unsorted name array
my $element;

foreach $element (@udrugname) {
    print "$element \n";
    # sleep 1;
}

# now sort the array alphabetically from the Unsorted list (@udrugname)
my @sdrugname;
@sdrugname = sort {
    $a cmp $b
} @udrugname;

# now print the sorted name array to the files
## s.. means SORTED
## u.. means UNsorted

my $n=0;
my $listnumber="";
my $dname="";
my $snomed_code="";

open my $camfile, ">", "u−drugsrn.conf=new" || die "ERROR: can't open CAMfile \n";

foreach $dname (@sdrugname) {
    print "$dname \n";
    $n=$n+1;
    $listnumber="0000".$n;
    $listnumber=substr($listnumber,−4);
    $snomed_code = $dmd{$dname};

    # print to a simple file
    print {
        $outfile2
    } "<"$listnumber,"">"$dname,"">"$snomed_code,"">\n";
    # print sorted order in format for Camomile
    ### \add{drugname}{...}
    print {
        $camfile
    } "\add{drugname} {"$dname,"} \n";
    ##print {
    # print {
        $camfile
    } "\add{drugname} {"$dname,"} \n";
}

###=$s
13.6 Perl program reverse.pl

#!/usr/bin/perl −w
## reverse.pl
## RWD Nickalls 2005
## to reverse a string of n names with +

my $instring = "A1A1 A2A2 + c1c1 c2c2 + R1R1 R2R2";
my $p=" + ";
# replace / + / with just +
# $instring =~ s/\$p/+/;

# put the words into an array

if ($instring =~ m/\[+\]/) {
    print "YES the string has a +\n";
    ## make array of words separated by space [+
    my @words = split (m/\[+\]/, $instring);
    my @clean_words = ();
    foreach $element (@words) {
        $element =~ s/^\s//; # remove leading white space
        $element =~ s/\s+$//; # trailing space
        push (@clean_words, $element);
    }
    my ($w1,$w2,$w3,$w4,$w5,$w6,$w7) = @clean_words;
    my $n=(#clean_words+1);
    print "n= ", $n,"\n";
    print " string = ", $instring,"\n";

    if ($n == 2) {
        print "REVstring = ", $w2.$p.$w1,"\n";
    }
    elsif ($n==3) {
        print "REVstring = ", $w1.$p.$w3.$p.$w2,"\n";
        print "REVstring = ", $w2.$p.$w1.$p.$w3,"\n";
        print "REVstring = ", $w3.$p.$w1.$p.$w2,"\n";
        print "REVstring = ", $w2.$p.$w3.$p.$w1,"\n";
        print "REVstring = ", $w3.$p.$w2.$p.$w1,"\n";
    }
    elsif ($n==4) {
        print "first= ", $w1,"\n";
        print "second= ", $w2,"\n";
        print "third= ", $w3,"\n";
        print "4th= ", $w4,"\n";
    }
    else { print "ERROR: string NOT processed as n = ", $n,"\n";
}
13.7 Initial data listing

The above program outputs the list in the existing order (as follows) showing that the list is not ordered alphabetically (this just reflects the fact that drugs are added to the list by the NHS simply in the order they are considered etc). The program then orders the list alphabetically to make it easier to find drugs in the pull-down menu (see below).

Where drugs are in combinations, then the program makes a new entry for each of the combined drugs (while including each of the other ones) so each drug combination appears several times, but each time with a different drug first. This naturally swells the drug listing (in this case from about 1842 entries to 2258—see below).

13.8 The ordered list

...
13.9 Adding drugs to the list

Since some of the anaesthesia drugs would be missing from the NHS list, then one had
to add these. In order to do this conveniently, a file containing the drugs we wanted to
add was created, as follows.

```bash
# dn-drugs2add.dat
# input file for the dn-dmd4.pl program
# Local drugname = official NHS drugname
#----------------------------------------
Adrenaline
Atracurium
Isoprenaline
Furosemide = Frusemide
Dextrose-saline = Normal saline 0.9% = Sodium chloride
Saline 0.9% = Sodium chloride
Bicarbonate 8.4% = Sodium bicarbonate
Sodium bicarbonate 8.4% = Sodium bicarbonate
HAS4.5 (Human-albumin-solution-4.5%)
HAS20 (Human-albumin-solution-20%)
```
Hespan (Hydroxy-ethyl-starch)
Gelofusin
Hartmans-solution = Sodium lactate
Blood (packed cells)
Blood (whole)
Magnesium = Magnesium sulphate
Insulin
Potassium = Potassium chloride
Thiopentone = Thiopental
Cryoprecipitate
FFP (Fresh-frozen-plasma)
PPF (Plasma-protein-fraction)
Esmolol

As time went by, some of these drugs would be added to the NHS list, and so the program indicated in the log file whether any of the drugs were found in the NHS list, and if so, did not add them.

13.10 Perl program add2list.pl

This program added to the NHS list the drugs in the missing list.

```perl
#!/usr/bin/perl

## add2list.pl
## RWDN Jan 13, 2006
##-------------------
use strict;
use warnings;
use Cwd;  # to get this PATH, eg $thisdir=cmd;
use Carp; # allows croak "" and warn "" (warn always --> to STDERR)
use Fatal qw(open close); # for errors
##use Perl6::Builtins qw(system);
#use Getopt::Long; ## for commandline stuff
#use version;

my @udrugname = ();
my %dmd = ();
my $drugname="";
my $drugcode="";
$drugname="Atropine", $drugcode="--";
push (@udrugname, $drugname); # collect name/code pairs into a hash
%dmd = (%dmd, $drugname, $drugcode);

$drugname="Bupivicaine", $drugcode="---";
push (@udrugname, $drugname); # collect name/code pairs into a hash
```
CHAPTER 13. DRUG DICTIONARY

13.11 Logfile generated by add2list.pl

rnalarm.log, Sun May 14 22:12:16 2006: Unix=1147641136
log of my Perl rnalarm3.pl program
...reading the infile file line-by-line
....adding drugs from the LOCAL list

--------------
dmd{addname} = 9885311000001102
Adrenaline *** is ALREADY on the VTM list
--------------
dmd{addname} = 9873211000001103
Atracurium *** is ALREADY on the VTM list
--------------
Isoprenaline has been put on the list just now
--------------
drug1 = {Frusemide} drug2 ={Furosemide}
Frusemide has been put on the list just now
----------------
drug1 = {Dextrose-saline} drug2 = {Glucose + Sodium chloride}
Dextrose-saline has been put on the list just now
----------------
drug1 = {Normal-Saline 0.9%} drug2 = {Sodium chloride}
Normal-Saline 0.9% has been put on the list just now
----------------
drug1 = {Saline 0.9%} drug2 = {Sodium chloride}
Saline 0.9% has been put on the list just now
----------------
drug1 = {Bicarbonate 8.4%} drug2 = {Sodium bicarbonate}
Bicarbonate 8.4% has been put on the list just now
----------------
drug1 = {Sodium bicarbonate 8.4%} drug2 = {Sodium bicarbonate}
Sodium bicarbonate 8.4% has been put on the list just now
----------------
HAS4.5 (Human-albumin-solution-4.5%) has been put on the list just now
----------------
HAS20 (Human-albumin-solution-20%) has been put on the list just now
----------------
Hespan (Hydroxy-ethyl-starch) has been put on the list just now
----------------
Gelofusin has been put on the list just now
----------------
drug1 = {Hartmans-solution} drug2 = {Sodium lactate}
Hartmans-solution has been put on the list just now
----------------
Blood (packed cells) has been put on the list just now
----------------
Blood (whole) has been put on the list just now
----------------
drug1 = {Magnesium} drug2 = {Magnesium sulphate}
Magnesium has been put on the list just now
----------------
Insulin has been put on the list just now
----------------
drug1 = {Potassium} drug2 = {Potassium chloride}
Potassium has been put on the list just now
----------------
drug1 = {Thiopentone} drug2 = {Thiopental}
Thiopentone has been put on the list just now
----------------
dmd\{addname\} = 10170311000001108
Cryoprecipitate *** is ALREADY on the VTM list
----------------
FFP (Fresh-frozen-plasma) has been put on the list just now
----------------
PPF (Plasma-protein-fraction) has been put on the list just now
----------------
13.12 Final list for pull-down menu

Finally, the program outputs a list suitable for the Workstation program, and which is input on startup. In practice we left the list as the complete list, and were intending to make a special anaesthesia subgroup for use with the workstation. Although this was not finished, in practice the pull-down menu was fast enough for us to simply leave the list as it was.

\add{drugname}{Abacavir}
\add{drugname}{Abacavir + Lamivudine}
\add{drugname}{Abciximab}
\add{drugname}{Acacia}
\add{drugname}{Acacia + Starch + Tragacanth}
\add{drugname}{Acamprosate}
\add{drugname}{Acarbose}
\add{drugname}{Acebutolol}
\add{drugname}{Acebutolol + Hydrochlorothiazide}
\add{drugname}{Acetarsol}
\add{drugname}{Acetazolamide}
\add{drugname}{Acetate acid}
...
...
\add{drugname}{Zinc sulphate + Lithium succinate}
\add{drugname}{Zinc undecenoate + Undecenoic acid}
\add{drugname}{Zoledronic acid}
\add{drugname}{Zolmitriptan}
\add{drugname}{Zolpidem}
\add{drugname}{Zonisamide}
\add{drugname}{Zopiclone}
\add{drugname}{Zotepine}
\add{drugname}{Zuclopenthixol}
\add{drugname}{Zuclopenthixol acetate}
\add{drugname}{Zuclopenthixol decanoate}
\add{drugname}{von Willebrand factor + Factor VIII}
Chapter 14

Diabetes decision-support system

14.1 Introduction

The Diabetes decision-support system consists of a diabetes widget which offers information and support as well as an alerting system to remind the anaesthetist to repeat blood sugars etc. This alert system uses the excellent Linux KDE Kalarm utility (see below).

The diabetes widget system described here is a working prototype which has proved to be extremely valuable for anaesthetists, and needs to be developed further at some stage. For example, it could usefully incorporate comprehensive algorithms similar to those described recently by Mraz et al. (2008).


Kalarm is a well established and sophisticated calendar/alarm system, and the version described here relates to Kalarm 1.8.10 (vers 2) as used with Mandriva-Linux 2006.0 (official). Kalarm is capable of sending emails, displaying text files, triggering an audible voice message, as well as displaying a coloured alert banner following a specified alarm interval, or at a specified date/time. The Kalarm system allows input either via a ‘form’ or via the command-line. However, the ‘form’ input method (mouse & keyboard) is too complicated and time consuming for reliable use in an anaesthesia setting. It was therefore decided to write a Perl-Tk program to generate a more convenient and intuitive widget system, which would allow a diabetes alert to be set quickly and reliably, simply by clicking on an appropriate widget button.

Diabetes widget. The buttons on the right offer preset time delays in the range 20–60 mins.
14.1.1 Kalarm and the iCalendar standard

Kalarm data is written to a text file encoded using the iCalendar Syntax Reference Standard 2445 (RFC 2445), which uses a number of nested so-called V-items, e.g. Valarm, Vevent etc. The following extract is from the Wikipedia entry for iCalendar (http://en.wikipedia.org/wiki/iCalendar).

iCalendar is a standard (RFC 2445 or RFC2445 Syntax Reference) for calendar data exchange. The standard is also known as “iCal”, which is the name of the Apple Computer calendar program that was the first software implementation of the standard.

iCalendar allows users to send meeting requests and tasks to other users through emails. Recipients of the iCalendar email (with supported software) can respond to the sender easily or counter propose another meeting date/time. It is implemented/supported by a large number of products, including 30Boxes, Google Calendar, Apple iCal application and iPod, Chandler, Lotus Notes, ScheduleWorld, KOrganizer, Lovento, Mozilla Calendar (including Mozilla Sunbird), Mulberry, Novell Evolution, Kronolith, Simple Groupware, Windows Calendar, Nuvvo, Upcoming.org and to some extent, Microsoft Outlook. . iCalendar data is typically exchanged using traditional email, but the standard is designed to be independent of the transport protocol. For example, it can also be shared and edited by using a WebDav server. Simple web servers (using just the HTTP protocol) are often used to distribute iCalendar data about an event and to publish busy times of an individual. Event sites on the web are embedding iCalendar data in web pages using hCalendar, a 1:1 representation of iCalendar in semantic XHTML.

14.1.2 VALARM specification from the RFC-2445 manual (v:2, Nov 1998)

Internet Calendaring and Scheduling Core Object Specification (iCalendar)
Copyright (C) The Internet Society (1998). All Rights Reserved.

4.6.6 Alarm Component

Component Name: VALARM

Purpose: Provide a grouping of component properties that define an alarm.

Formal Definition: A "VALARM" calendar component is defined by the following notation:

\[
\text{alarmc} = "\text{BEGIN" ":" "VALARM" CRLF} \\
\text{ (audioprop / dispprop / emailprop / procprop)} \\
\text{ "\text{END" ":" "VALARM" CRLF} \\
\text{audioprop = 2*} (} \\
; \text{’action’ and ’trigger’ are both REQUIRED,} \\
\text{but MUST NOT occur more than once} \\
\text{action / trigger/}
\]
; 'duration' and 'repeat' are both optional,
; and MUST NOT occur more than once each,
; but if one occurs, so MUST the other
duration / repeat /

; the following is optional,
; but MUST NOT occur more than once
attach /

; the following is optional,
; and MAY occur more than once
x-prop

) dispprop = 3*(

; the following are all REQUIRED,
; but MUST NOT occur more than once
action / description / trigger /

; 'duration' and 'repeat' are both optional,
; and MUST NOT occur more than once each,
; but if one occurs, so MUST the other
duration / repeat /

; the following is optional,
; and MAY occur more than once
*x-prop

) emailprop = 5*(

; the following are all REQUIRED,
; but MUST NOT occur more than once
action / description / trigger / summary

; the following is REQUIRED,
Description: A "VALARM" calendar component is a grouping of component properties that is a reminder or alarm for an event or a to-do. For example, it may be used to define a reminder for a pending event or an overdue to-do.

The "VALARM" calendar component MUST include the "ACTION" and
"TRIGGER" properties. The "ACTION" property further constrains the "VALARM" calendar component in the following ways:

When the action is "AUDIO", the alarm can also include one and only one "ATTACH" property, which MUST point to a sound resource, which is rendered when the alarm is triggered.

When the action is "DISPLAY", the alarm MUST also include a "DESCRIPTION" property, which contains the text to be displayed when the alarm is triggered.

When the action is "EMAIL", the alarm MUST include a "DESCRIPTION" property, which contains the text to be used as the message body, a "SUMMARY" property, which contains the text to be used as the message subject, and one or more "ATTENDEE" properties, which contain the email address of attendees to receive the message. It can also include one or more "ATTACH" properties, which are intended to be sent as message attachments. When the alarm is triggered, the email message is sent.

When the action is "PROCEDURE", the alarm MUST include one and only one "ATTACH" property, which MUST point to a procedure resource, which is invoked when the alarm is triggered.

The "VALARM" calendar component MUST only appear within either a "VEVENT" or "VTODO" calendar component. "VALARM" calendar components cannot be nested. Multiple mutually independent "VALARM" calendar components can be specified for a single "VEVENT" or "VTODO" calendar component.

The "TRIGGER" property specifies when the alarm will be triggered. The "TRIGGER" property specifies a duration prior to the start of an event or a to-do. The "TRIGGER" edge may be explicitly set to be relative to the "START" or "END" of the event or to-do with the "RELATED" parameter of the "TRIGGER" property. The "TRIGGER" property value type can alternatively be set to an absolute calendar date and time of day value.

In an alarm set to trigger on the "START" of an event or to-do, the "DTSTART" property MUST be present in the associated event or to-do. In an alarm in a "VEVENT" calendar component set to trigger on the "END" of the event, either the "DTEND" property MUST be present, or the "DTSTART" and "DURATION" properties MUST both be present. In an alarm in a "VTODO" calendar component set to trigger on the "END" of the to-do, either the "DUE" property MUST be present, or the "DTSTART" and "DURATION" properties MUST both be present.

The alarm can be defined such that it triggers repeatedly. A definition of an alarm with a repeating trigger MUST include both the "DURATION" and "REPEAT" properties. The "DURATION" property specifies
the delay period, after which the alarm will repeat. The "REPEAT" property specifies the number of additional repetitions that the alarm will triggered. This repetition count is in addition to the initial triggering of the alarm. Both of these properties MUST be present in order to specify a repeating alarm. If one of these two properties is absent, then the alarm will not repeat beyond the initial trigger.

The "ACTION" property is used within the "VALARM" calendar component to specify the type of action invoked when the alarm is triggered. The "VALARM" properties provide enough information for a specific action to be invoked. It is typically the responsibility of a "Calendar User Agent" (CUA) to deliver the alarm in the specified fashion. An "ACTION" property value of AUDIO specifies an alarm that causes a sound to be played to alert the user; DISPLAY specifies an alarm that causes a text message to be displayed to the user; EMAIL specifies an alarm that causes an electronic email message to be delivered to one or more email addresses; and PROCEDURE specifies an alarm that causes a procedure to be executed. The "ACTION" property MUST specify one and only one of these values.

In an AUDIO alarm, if the optional "ATTACH" property is included, it MUST specify an audio sound resource. The intention is that the sound will be played as the alarm effect. If an "ATTACH" property is specified that does not refer to a sound resource, or if the specified sound resource cannot be rendered (because its format is unsupported, or because it cannot be retrieved), then the CUA or other entity responsible for playing the sound may choose a fallback action, such as playing a built-in default sound, or playing no sound at all.

In a DISPLAY alarm, the intended alarm effect is for the text value of the "DESCRIPTION" property to be displayed to the user.

In an EMAIL alarm, the intended alarm effect is for an email message to be composed and delivered to all the addresses specified by the "ATTENDEE" properties in the "VALARM" calendar component. The "DESCRIPTION" property of the "VALARM" calendar component MUST be used as the body text of the message, and the "SUMMARY" property MUST be used as the subject text. Any "ATTACH" properties in the "VALARM" calendar component SHOULD be sent as attachments to the message.

In a PROCEDURE alarm, the "ATTACH" property in the "VALARM" calendar component MUST specify a procedure or program that is intended to be invoked as the alarm effect. If the procedure or program is in a format that cannot be rendered, then no procedure alarm will be invoked. If the "DESCRIPTION" property is present, its value specifies the argument string to be passed to the procedure or program. "Calendar User Agents" that receive an iCalendar object with this category of alarm, can disable or allow the "Calendar User" to
disable, or otherwise ignore this type of alarm. While a very useful alarm capability, the PROEDURE type of alarm SHOULD be treated by the "Calendar User Agent" as a potential security risk.

Example: The following example is for a "VALARM" calendar component that specifies an audio alarm that will sound at a precise time and repeat 4 more times at 15 minute intervals:

```
BEGIN:VALARM
TRIGGER;VALUE=DATE-TIME:19970317T133000Z
REPEAT:4
DURATION:PT15M
ACTION:AUDIO
ATTACH;FMTTYPE=audio/basic:ftp://host.com/pub/sounds/bell-01.aud
END:VALARM
```

The following example is for a "VALARM" calendar component that specifies a display alarm that will trigger 30 minutes before the scheduled start of the event or the due date/time of the to-do it is associated with and will repeat 2 more times at 15 minute intervals:

```
BEGIN:VALARM
TRIGGER:-PT30M
REPEAT:2
DURATION:PT15M
ACTION:DISPLAY
DESCRIPTION:Breakfast meeting with executive\n  team at 8:30 AM EST.
END:VALARM
```

The following example is for a "VALARM" calendar component that specifies an email alarm that will trigger 2 days before the scheduled due date/time of a to-do it is associated with. It does not repeat. The email has a subject, body and attachment link.

```
BEGIN:VALARM
TRIGGER:-P2D
ACTION:EMAIL
ATTENDEE:MAILTO:john_doe@host.com
SUMMARY:*** REMINDER: SEND AGENDA FOR WEEKLY STAFF MEETING ***
DESCRIPTION:A draft agenda needs to be sent out to the attendees to the weekly managers meeting (MGR-LIST). Attached is a pointer the document template for the agenda file.
ATTACH;FMTTYPE=application/binary:http://host.com/templates/agenda.doc
END:VALARM
```

The following example is for a "VALARM" calendar component that specifies a procedural alarm that will trigger at a precise date/time and will repeat 23 more times at one hour intervals. The alarm will
invoke a procedure file.

BEGIN:VALARM
TRIGGER;VALUE=DATE-TIME:19980101T050000Z
REPEAT:23
DURATION:PT1H
ACTION:PROCEDURE
ATTACH;FMTTYPE=application/binary:ftp://host.com/novo-
procs/felizano.exe
END:VALARM

Before describing the ‘diabetes alert’ widget and the associated Perl programs initiated by clicking on the various buttons, we first give a brief overview of the Kalarm system and its command structure, with illustrations linked to the diabetes alarm.

### 14.2 Kalarm

The Linux Kalarm utility is an established and versatile alarm tool which can be developed for use with the anaesthesia workstation. Kalarm is maintained by David Jarvie (software@astrojar.org.uk; http://www.astrojar.org.uk/linux/kalarm.html). The latest version is 1.4.0 (April 2006). Kalarm can be accessed either using a ‘form’ via the mouse from the taskbar icon, or via the command-line, and has good documentation via a standard kalarm --help command.

Alarms can be both initiated and cancelled using commands issued via the command-line.

#### 14.2.1 To show Kalarm icon

To generate the Kalarm icon just type

```
$ kalarm
```

at the command-line, and it will appear on the bottom-bar. The diabetes alarm depends on the Kalarm scheduling daemon running; this can be started using the [--reset] option, as follows (see also documentation section below).

```
$ kalarm --reset
```

#### 14.2.2 Documentation

Online help is available via the command kalarm --help-all as follows. Detailed information is also available from the Kalarm Handbook, which can be accessed via the alarm tray widget (click on ‘help’), and also from /usr/share/doc/HTML/en/kalarm/index.docbook

version 0.8.3

Usage: kalarm [Qt-options] [KDE-options] [options] [message]

        kalarm
    kalarm  [-bciLrstu] -f URL
    kalarm  [-bciLrstu] message
kalarm [-illrtu] -e commandline
kalarm --tray | --reset | --stop
kalarm --cancelEvent eventID [--calendarURL url]
kalarm --triggerEvent eventID [--calendarURL url]
kalarm --handleEvent eventID [--calendarURL url]
kalarm [generic_options]

KDE personal alarm message and command scheduler

Generic options:
--help Show help about options
--help-qt Show Qt specific options
--help-kde Show KDE specific options
--help-all Show all options
--author Show author information
-v, --version Show version information
--license Show license information
-- End of options

Qt options:
--display <displayname> Use the X-server display 'displayname'.
--session <sessionId> Restore the application for the given 'sessionId'.
--cmap Causes the application to install a private colour map on an 8-bit display.
--ncols <count> Limits the number of colours allocated in the colour cube on an 8-bit display, if the application is using the QApplication::ManyColor colour specification.
--nograb tells Qt to never grab the mouse or the keyboard.
--dograb running under a debugger can cause an implicit -nograb, use -dograb to override.
--sync switches to synchronous mode for debugging.
--fn, --font <fontname> defines the application font.
--bg, --background <color> sets the default background colour and an application palette (light and dark shades are calculated).
--fg, --foreground <color> sets the default foreground colour.
--btn, --button <color> sets the default button colour.
--name <name> sets the application name.
--title <title> sets the application title (caption).
--visual TrueColor forces the application to use a TrueColour visual on an 8-bit display.
--inputstyle <inputstyle> sets XIM (X Input Method) input style. Possible values are onthespot, overthespot, offthespot and root.
--im <XIM server> set XIM server.
--noxim disable XIM.
--reverse mirrors the whole layout of widgets.

KDE options:
CHAPTER 14. DIABETES DECISION-SUPPORT SYSTEM

14.2.3 Initiating a diabetes alarm

An example of the command-line (case sensitive) code for initiating a red alarm to prompt the user to repeat a blood-sugar measurement for a diabetic patient, with a pop-up window + beep repeating at 30 mins intervals is as follows (b=beep, c=colour, u=until-hh:mm, i=interval-mmmm).

In Mandrake-Linux the details of the alarm are written to the file /home/dick/.kde/share/apps/kalarm/calendar.ics. The default ‘empty’ file (ie with no alarms pending) is as follows.

```
BEGIN:VCALENDAR
PRODID:-//K Desktop Environment//NONSGML KAlarm 1.2.10//EN
VERSION:2.0
END:VCALENDAR
```
An example of the command-line (case sensitive) code for initiating a red alarm to prompt the user to repeat a blood-sugar measurement for a diabetic patient, with a pop-up window + beep repeating at 30 mins intervals is as follows (b=beep, c=colour, -t=trigger time yyyy-mm-dd:hh:mm, u=until-hh:mm, i=interval- mmmm).

    "DIABETES --- repeat blood sugar"

This command generates a new calendar.ics file, which encodes the alarm data. Note that a given alarm instance (VEVENT) may be associated with several alarms (VALARM) in different formats (eg text, displayed file, voice etc) There is one VALARM for the display of message, and another VALARM for the sound of the beep. Note the empty lines following the END: commands.

    BEGIN:VCALENDAR
    PRODID:-//K Desktop Environment//NONSGML KAlarm 1.2.10//EN
    VERSION:2.0
    BEGIN:VEVENT
    DTSTAMP:20080410T113102
    ORGANIZER:MAILTO:
    CREATED:20080410T113102
    UID:KAlarm-1412322138.966
    SEQUENCE:-1232236916
    LAST-MODIFIED:20080410T113102
    CLASS:PUBLIC
    PRIORITY:5
    RRULE:FREQ=MINUTELY;UNTIL=20080411T113100;INTERVAL=5
    DTSTART:20080410T115100
    TRANSP:TRANSPARENT
    BEGIN:VALARM
    DESCRIPTION: DIABETES --- repeat blood sugar
    ACTION:DISPLAY
    TRIGGER;VALUE=DURATION:PT0S
    X-KDE-KALARM-FONTCOLOR:#ff0000;#000000
    END:VALARM
    BEGIN:VALARM
    ACTION:AUDIO
    TRIGGER;VALUE=DURATION:PT0S
    END:VALARM
    END:VEVENT
    END:VCALENDAR

14.2.4 Displaying a file

Note that the “alarm” can be the display of a file. For example, the following code will immediately (since no -t option) display a HTML file in a window. Note that there must be NO display “message” argument with this command since in this case the file has taken the place of the message.
kalarm -b -c red -f "/home/dick/...../file.html"

### 14.2.5 Current alarm status

The list and status of all current outstanding alarms are displayed in the **Kalarm** tray - tabular listing seen by clicking on the **Kalarm** icon on the bottom-bar. The code to place the icon in the bottom-bar tray is as follows.

```
$ kalarm --tray
```

### 14.2.6 Cancelling an alarm

The **Kalarm** command (case sensitive) for cancelling an existing alarm having the UID **197659548.1073** as shown above is as follows, which has the effect of deleting the associated **VEVENT** environment from the **calendar.ics** file.

```
Kalarm -cancelEvent KAlarm-197659548.1073
```

Thus in order to delete an existing alarm ‘event’ it is necessary to parse the **calendar.ics** file and determine the **UID** associated with the particular alarm. Consequently, in order to facilitate identifying the correct **UID** for an alarm we simply arrange that (a) only a single alarm exists at any one time, and (b) we include a key word, say **DIABETES**, in the text message.

### 14.3 Alarm widget program (dn-tkalarm.pl)

![Figure 14.1: View of the pop-up diabetes support widget. Clicking one of the blue ‘diabetes’ buttons (20–60 mins) sets an alert for the associated time interval. The three grey buttons are for displaying help information; the two green buttons are for generating test displays.](image)

```bash
#!/usr/bin/perl
## dn−tkalarm.pl (modified from tklaunch2.pl)
## last modified April 24, 2006
```
CHAPTER 14. DIABETES DECISION-SUPPORT SYSTEM

RWD Nickalls

197

my $thisprog = ”[dn−tkalarm.pl]”; #to define this program−name in error messaages
## RWD Nickalls
## last change = Jan 22, 2006
## alarms for Xenon
## Useful books: page 301 Perl core languages (Little Black Book)
##−−−−−−−−−−−−−−−−−−−−−
## BOOK = Mastering Perl Tk (by: Lidie S and Walsh N (O’Reilly, 2002)
## to get FullScreen mode at startup (p 307)
## −geometry widthXheight+Xoffset+Yoffset (NO spaces**page 409)
## $ perl tklaunch2.pl −geometry 1028x768 −0−0 ## page 409
## system (”perl ./tklaunch2.pl −geometry 300x400−50−300”) }
##−−−−−−−−−−−−−−−−−−−−−
use warnings;
use strict;
use Carp;
use Fatal;
use Tk;
use Tk::Help;
use Cwd; # get this path
#−−−−−−−−macros−−−−−−−−−−
my $beep = ”∖a”; ##BEEP
my $OS ERROR =””; ## used in viewcal SUB
my $kalarm calendar path=”/home/dick/.kde/share/apps/kalarm/calendar.ics”;
#−−−−−−−−−−−−−−−−−−−
my $topwindow = MainWindow −> new();
#−−−−−−−−−−−−−−−−−−−−−−−−−−−−
$topwindow −> title(”XENON diabetes support”);
$topwindow −> Label(−text => ”Click on a diabetes button to set an alarm”,
−wraplength =>100,
−padx => 0.5, #250
−height => 10 )
−> pack();
## camel logo
if (−e ”./anim.gif”){
my $camelimage = $topwindow −> Photo(−file => ’./anim.gif’);
$topwindow −> Button(−relief => ’flat’, −image => $camelimage)
−> place(−relx=>0.005, −rely=>0);
}
#−−−−−−−−−−−−−−−−−−−
# QUIT button
$topwindow −> Button (−text => ”QUIT”,
−padx => 20, −pady => 20,
−relief => ’raised’,
−background => ’LightBlue1’,
−activebackground =>’LightBlue2’,
−command => ∖&quit )
−> place(−relx=>0.05, −rely=>0.115);
#−> pack(−side =>’left’, −expand => 1);


# (c) XENON project team

$topwindow  -->  Button (-text => "(c) The XENON project team",
   -pady => 10, -pady => 10,
   -relief => 'flat',
   -background => 'LightGrey',
   -foreground => 'Black',
   -activeforeground => 'Black',
)
   -> place(-relx=>0.35, -rely=>0.016);

# DIABETES 20mins button

$topwindow  -->  Button (-text => "DIABETES -- 20 mins",
   -pady => 10, -pady => 10,
   -relief => 'raised',
   -background => 'LightBlue3',
   -activebackground => 'LightBlue2',
   -foreground => 'Blue',
   -activeforeground => 'Red',
)
   -> pack(-side =>'right', -expand => 1);
   -> place(-relx=>0.5, -rely=>0.3);

# DIABETES 30mins button

$topwindow  -->  Button (-text => "DIABETES -- 30 mins",
   -pady => 10, -pady => 10,
   -relief => 'raised',
   -background => 'LightBlue3',
   -activebackground => 'LightBlue2',
   -foreground => 'Blue',
   -activeforeground => 'Red',
)
   -> pack(-side =>'bottom', -expand => 1);
   -> place(-relx=>0.5, -rely=>0.42);

# DIABETES 40mins button

$topwindow  -->  Button (-text => "DIABETES -- 40 mins",
   -pady => 10, -pady => 10,
   -relief => 'raised',
   -background => 'LightBlue3',
   -activebackground => 'LightBlue2',
   -foreground => 'Blue',
   -activeforeground => 'Red',
)
   -> pack(-side =>'bottom', -expand => 1);
   -> place(-relx=>0.5, -rely=>0.54);

# DIABETES 50mins button
CHAPTER 14. DIABETES DECISION-SUPPORT SYSTEM

\[ \text{Stopwindow} \rightarrow \text{Button (text} \rightarrow \text{"DIABETES -- 50 mins"}, \]
\[
- \text{padx} \rightarrow 10, - \text{pady} \rightarrow 10, \\
- \text{relief} \rightarrow \text{‘raised’}, \\
- \text{background} \rightarrow \text{‘LightBlue3’}, \\
- \text{activebackground} \rightarrow \text{‘LightBlue2’}, \\
- \text{foreground} \rightarrow \text{‘Blue’}, \\
- \text{activeforeground} \rightarrow \text{‘Red’}, \\
- \text{command} \rightarrow \text{\&diabetes50} \\
# - > \text{pack}(-\text{side} => \text{‘bottom’}, - \text{expand} => 1); \\
- > \text{place}(-\text{relx}=0.5, - \text{rely}=0.66); \\
# - - - - - - - - - - - - - - - - - - - - - - - - - - -
# DIABETES 60mins button
\[ \text{Stopwindow} \rightarrow \text{Button (text} \rightarrow \text{"DIABETES -- 60 mins"}, \]
\[
- \text{padx} \rightarrow 10, - \text{pady} \rightarrow 10, \\
- \text{relief} \rightarrow \text{‘raised’}, \\
- \text{background} \rightarrow \text{‘LightBlue3’}, \\
- \text{activebackground} \rightarrow \text{‘LightBlue2’}, \\
- \text{foreground} \rightarrow \text{‘Blue’}, \\
- \text{activeforeground} \rightarrow \text{‘Red’}, \\
- \text{command} \rightarrow \text{\&diabetes60} \\
# - > \text{pack}(-\text{side} => \text{‘bottom’}, - \text{expand} => 1); \\
- > \text{place}(-\text{relx}=0.5, - \text{rely}=0.78); \\
# - - - - - - - - - - - - - - - - - - - - - - - - - - -
# TEST--COFFEE demo button
\[ \text{Stopwindow} \rightarrow \text{Button (text} \rightarrow \text{"TEST--c"}, \]
\[
- \text{padx} \rightarrow 10, - \text{pady} \rightarrow 5, \\
- \text{relief} \rightarrow \text{‘raised’}, \\
- \text{background} \rightarrow \text{‘Green’}, \\
- \text{activebackground} \rightarrow \text{‘Yellow’}, \\
- \text{foreground} \rightarrow \text{‘Black’}, \\
- \text{activeforeground} \rightarrow \text{‘Red’}, \\
- \text{command} \rightarrow \text{\&testcoffee5} \\
# - > \text{pack}(-\text{side} => \text{‘bottom’}, - \text{expand} => 1); \\
- > \text{place}(-\text{relx}=0.05, - \text{rely}=0.36); \\
# - - - - - - - - - - - - - - - - - - - - - - - - - - -
# TEST--diabetes demo button
\[ \text{Stopwindow} \rightarrow \text{Button (text} \rightarrow \text{"TEST--d"}, \]
\[
- \text{padx} \rightarrow 10, - \text{pady} \rightarrow 5, \\
- \text{relief} \rightarrow \text{‘raised’}, \\
- \text{background} \rightarrow \text{‘Green’}, \\
- \text{activebackground} \rightarrow \text{‘Red’}, \\
- \text{foreground} \rightarrow \text{‘Black’}, \\
- \text{activeforeground} \rightarrow \text{‘Blue’}, \\
- \text{command} \rightarrow \text{\&testdiabetes} \\
# - > \text{pack}(-\text{side} => \text{‘bottom’}, - \text{expand} => 1); \\
- > \text{place}(-\text{relx}=0.05, - \text{rely}=0.45);
```python
# HOWTO use button
$topwindow -> Button (-text => "HOWTO use",
    -padx => 9, -pady => 10,
    -relief => 'raised',
    -background => 'LightGrey',
    -activebackground => 'Grey',
    -foreground => 'Blue',
    -activeforeground => 'Red',
    # -command => \&errorbox )
#--command => \&showhelp )
-pack(side => 'bottom', expand => 1);
-place(relx=>0.05, rely=>0.54);

# VIEW logfile button
$topwindow -> Button (-text => "VIEW logfile",
    -padx => 10, -pady => 10,
    -relief => 'raised',
    -background => 'LightGrey',
    -activebackground => 'Grey',
    -foreground => 'Blue',
    -activeforeground => 'Red',
    -command => \&viewlog )
-pack(side => 'bottom', expand => 1);
-place(relx=>0.05, rely=>0.66);

# VIEW calendar file button
$topwindow -> Button (-text => "VIEW calfile",
    -padx => 10, -pady => 10,
    -relief => 'raised',
    -background => 'LightGrey',
    -activebackground => 'Grey',
    -foreground => 'Blue',
    -activeforeground => 'Red',
    -command => \&viewcal )
-pack(side => 'bottom', expand => 1);
-place(relx=>0.05, rely=>0.78);

## HELP button
$topwindow -> Button(-text => "HELP on diabetes",
    -padx =>115, -pady =>10, -relief => 'flat',
    -background => 'LightGrey',
    -activebackground => 'Grey',
    -foreground => 'Blue',
    -activeforeground => 'Red',
    -command => \&help )
-place(relx=>0, rely=>0.9);
```
my $diabetes_error_message = "...ERROR running dn–alarm–diabetes2 ".thisprog;

MainLoop;

###=# SUBS==#=

sub quit { # clear the command–line terminal window and then exit
    system ("clear");
    exit();
}

sub diabetes20 {
    # $topwindow --> bell;
    # $result = $dialog1 --> Show;
    # if ($result eq "OK") {};
    # $topwindow --> destroy if Tk::Exists($topwindow);
    system ("perl ./dn–alarm–diabetes3.pl –t 20")
    and carp ($diabetes_error_message);
    # system ("perl ./dn–tkalarm.pl –geometry 320x380–50–300");
}

sub diabetes30 {
    # $topwindow --> bell;
    # $result = $dialog1 --> Show;
    # if ($result eq "OK") {};
    # $topwindow --> destroy if Tk::Exists($topwindow);
    system ("perl ./dn–alarm–diabetes3.pl –t 30")
    and carp ($diabetes_error_message);
    # system ("perl ./dn–tkalarm.pl –geometry 320x380–50–300");
}

sub diabetes40 {
    # $topwindow --> bell;
    # $result = $dialog1 --> Show;
    # if ($result eq "OK") {};
    # $topwindow --> destroy if Tk::Exists($topwindow);
    system ("perl ./dn–alarm–diabetes3.pl –t 40")
    and carp ($diabetes_error_message);
    # system ("perl ./dn–tkalarm.pl –geometry 320x380–50–300");
}

sub diabetes50 {
    # $topwindow --> bell;
    # $result = $dialog1 --> Show;
    # if ($result eq "OK") {};
    # $topwindow --> destroy if Tk::Exists($topwindow);
    system ("perl ./dn–alarm–diabetes3.pl –t 50")
    and carp ($diabetes_error_message);
    # system ("perl ./dn–tkalarm.pl –geometry 320x380–50–300");
}

#---------------------------------------------------------------
sub diabetes60 {
    ## $topwindow -> bell;
    ## $result = $dialog1 -> Show;
    ## if ($result eq "OK") {};
    ## $topwindow -> destroy if Tk::Exists($topwindow);
    system ("perl ./dn-alarm-diabetes3.pl -t 60")
    and carp ($diabetes_error_message);
    # system ("perl ./dn-talarm.pl -geometry 320x380-50-300");
}  # end of sub

sub testcoffee5 {
    ## test use only 1 min test ( -u 1 -i 1)
    ## as this will totally clear after 1 min
    system ("perl ./dn-alarm-coffee3.pl -u 1");
    # system ("perl ./dn-talarm.pl -geometry 320x380-50-300");
}  # end of sub

sub testdiabetes {
    ## if use parameters ( -u 1) only, then instant and no repeat!
    ## test use only 1 min test ( -u 1 -i 1)
    ## as this will totally clear after 1 min
    # system ("perl ./dn-alarm-coffeeRED.pl -u 1");
    system ("perl ./dn-alarm-demoRED.pl");
    # system ("kwrite ./anes-files/induction.txt -geometry 350x380-600-300");
    # system ("perl ./dn-talarm.pl -geometry 320x380-50-300");
}  # end of sub

sub errorbox {
    ## testing area
    ## $topwindow -> bell;
    ## $result = $dialog1 -> Show;
    ## if ($result eq "OK") {};
    ## $topwindow -> destroy if Tk::Exists($topwindow);
    $beep;
    system (qq{perl ./dn-errorbox.pl -i "testing the message box"});
    ## now reinstate the Tk diabetes alarm widget
    system ("perl ./dn-talarm.pl -geometry 320x380-50-300");
}  # end of sub

sub viewlog {
    $topwindow -> destroy if Tk::Exists($topwindow);
    if (-e "./dnalarm.log")
        (## use my dn-tviewer.pl utility to view the file
        system ("perl ./dn-tviewer.pl -i ./dnalarm.log");
        system ("perl ./dn-talarm.pl -geometry 320x380-50-300");
    )
    else { carp "....ERROR ....can't find file dnalarm.log [dn-talarm.pl]";
}
system ("perl ./dn−tkalarm.pl –geometry 320x380−50–300");
};

} ## end of the sub

sub viewcal {
    $topwindow −> destroy if Tk::Exists($topwindow);
    ## copy latest instance of the file
    ## this is a significant error if the copy fails
    my $thisdir = cwd;
    my $copy_string = "cp "$kalarm_calendar_path." "$thisdir.”/dn–calendar.ics";
    system $copy_string
    and carp "could not run $copy_string ($OS_ERROR)";
    #Perl−best–practice p 280
    ####
    ## now view the copied file
    if (−e "./dn–calendar.ics")
    {## use my dn–tkviewer.pl utility to view the file
        system ("perl dn–tkviewer.pl –−in ./dn–calendar.ics")
        and carp ("could not run Perl dn–tkviewer.pl "$thisprog,”
        ($OS_ERROR)";
        system ("perl ./dn–tkalarm.pl –geometry 320x380–50–300")
    } else {
        print "....ERROR:
    print "....can’t find file dn–calendar.ics>
    system ("perl ./dn–tkalarm.pl –geometry 320x380–50–300");
    }
}## end of the sub

sub help {
    ###### this displays the main diabetes help file
    #$stopwindow −> bell;
    ## $result = $dialog2 −> Show;
    $stopwindow −> destroy if Tk::Exists($stopwindow);
    # if (−e "camteama5dvi.dvi")
    if (−e "/diabetes/diabetes_intro.html")
    {## first remove the Tk screen
        #$stopwindow −> destroy if Tk::Exists($stopwindow);
        #$stopwindow−> bell; # beeps if click window (p 296)
        # system("xdevi camteama5dvi.dvi –paper a5 –geometry +20+20");
        # system("konqueror diabetes_intro.html");
    } else {
        print "....ERROR:
    print "....can’t find program <camteama5dvi.dvi>
    system ("perl ./dn–tkalarm.pl –geometry 320x380–50–300");
}
my @helparray = (  
    [  
        {  
            title => "\nHOWTO use \n",  
            header => "",  
            text => "This is a description of my application for the help."  
        }  
    ]  
);  
# end of sub

sub showhelp {  
## opens the small help window

# create the array of help contents to pass to the help module

my @helparray = (  
    [  
        {  
            title => "Overview",  
            header => "\nThis widget is an aid for use when anaesthetising a diabetic patient.  
\nIt uses the well established Linux KDE Kalarm Open Source alarm utility (www.astrojar.org.uk/linux/\nkalarm.html/).  
\nOnce a diabetes alert is set, a red alert window (reminding you to take a blood sugar) will open after the set elapsed time.  
\nTest the diabetes alert by first clicking on the green TEST button, which will generate a demo red alert (simulating the red DIABETES alert). To trigger the TRUE diabetes alert system just click on one of the blue DIABETES buttons.  
\nIf you are too busy to do a blood–sugar when the red alert window appears, just close the window, and the alert will continue to recur at 5–min intervals until you set a new alert.",  
            text => ""  
        }  
    ],  
    [  
        {  
            title => "Setting an alert",  
            header => "\nSimply click on one of the blue 'DIABETES' buttons. This will automatically set a new alert and delete any previous alert.\nThe new alert will appear after the specified time, and then recur every 5–mins until a new blue DIABETES alert is set—or until the existing alert is cancelled",  
            text => ""  
        }  
    ],  
    [  
        {  
            title => "Cancelling an alert",  
            header => "\nClick on the clock icon on the icon bar at the bottom of the screen (typically on the RHS). This will display all the current alarms (alerts).\nNow select the alarm to be cancelled (by right clicking on it), and then click on the 'delete' button, and close the window.",  
            text => ""  
        }  
    ],  
    [  
        {  
            title => "Testing",  
            header => "\nClick on the green TEST button: \n\nThis generates a demo COFFEE–break reminder (yellow).  
\nThis generates a RED coffee–break alert (+ beep) to simulate the red DIABETES alert.",  
            text => ""  
        }  
    ]  
);
### 14.4 Test demo programs (dn-alarm-demoRED.pl)

There are two test buttons which trigger demo programs; these show a yellow (dn-alarm-demoYELLOW.pl) and a red (dn-alarm-demoRED.pl) demo alert. The following is the ‘red’ demo program.

```perl
#!/usr/bin/perl
# dn-alarm--demoRED.pl
```
CHAPTER 14. DIABETES DECISION-SUPPORT SYSTEM

# RWDN Thurs 24April2006
## to look like a diabetes alarm
## main difference is that the trigger option is NOT used here

use warnings;
use strict;
use Carp; # allows croak ""
use Fatal qw(open close);

my $kalarm_calendar_path="/home/dick/.kde/share/apps/kalarm/calendar.ics"
my $OS_ERROR="";

my $until_unix= $time_now_unix+ 120; ## = 2mins in secs
my $until_string=localtime($until_unix);

my $until_ymdhm=ymdhm($until_string);

my $message=" time for a COFFEE—break. Ahh..."
my $out= "kalarm —b —c red —i 0005 $until $message"

my $out=" setting new RED COFFEE alarm\n"

$system($out) and croak "could not run $out ($OS_ERROR)" ;

#Perl—best—practice p 280
###==SIBS==

```perl
sub ymdhm {
    # format = yyyy-mm-dd–hh:hh
    # passing only one time_string into array
    my ($time_string) = @_;  # format = yyyy-mm-dd
    # need to determine the until time in the correct format for kalarm
    # routine modified from fields2PDATA.pl

    my @stgmt = split(/,/, $time_string);
    my $day = $stgmt[0];  # not used here
    my $month = $stgmt[1];
    my $date = $stgmt[2];
    my $st = $stgmt[3];
    my $year = $stgmt[4];

    # now split the time hh:mm:ss
    # but Kalarm requires that both month and date are in numerals
    if ($month eq "Jan") { $month = "01" }  # "Jan"
    if ($month eq "Feb") { $month = "02" }
    if ($month eq "Mar") { $month = "03" }
    if ($month eq "Apr") { $month = "04" }
    if ($month eq "May") { $month = "05" }
    if ($month eq "Jun") { $month = "06" }
    if ($month eq "Jul") { $month = "07" }
    if ($month eq "Aug") { $month = "08" }
    if ($month eq "Sep") { $month = "09" }
    if ($month eq "Oct") { $month = "10" }
    if ($month eq "Nov") { $month = "11" }
    if ($month eq "Dec") { $month = "12" }
```

---

The above code snippet is a subroutine in Perl that converts a string representing a date and time into a format suitable for the Kalarm system. It first splits the time_string into day, month, date, and time components and then formats the month and date to be in numerals. The code includes error checking for specific months and ensures the time is in the correct format for Kalarm.
14.5 Diabetes alarm program

(dn-alarm-diabetes3.pl)
else { print “ERROR: cannot copy the cal file\n”;
##==========================================
## set the eventFLAG
my $eventnumber=0; # counter to count the number of DIABETES events
my $eventFLAG="OFF";
open my $calfile, "<", "dn−calendar.ics"|| die “ERROR: can’t open file
dn−calendar.ics \n”;

## now read each line in the file, and place parameters into an array
print “...reading the CAL file line−by−line\n”;
print {$logg} “...reading the CAL file line−by−line\n”;

# reset these variables to zero BEFORE starting the WHILE loop
my $uid1 = 0;
my $uid2 = 0;
my $uid = "";
my $text1 = 0;
my $text2 = 0;
my $text = "";
my $dataline="";
my $event="";

# reset variables to zero
$uid1 = 0;
$uid2 = 0;
$uid = "";
$text1 = 0;
$text2 = 0;
$text = "";

# reset variables to zero

while {$calfile}{
  next LINE if /#/; #skip # comments
  next LINE if /%/; #skip % comments
  next LINE if /$/; #skip blank lines
  # grab the whole line as a string
  $dataline = $;
  chomp($dataline); # removes the line−ending
  # reset variables to zero
  $uid1 = 0;
  $uid2 = 0;
  $uid = "";
  $text1 = 0;
  $text2 = 0;
  $text = "";

  # @value=split (/[,]/, $dataline);
  # print $dataline;
  # replace CR/LF/space/ with visible chars = newbuffer
  # $dataline = s/\r/\n/; # $dataline = s/\n/\n/;
  # $dataline = s/\n/>/;
  # $dataline = s/\n/\n/;
  # print $dataline. "\n";
  if ($dataline=m/BEGIN:VEVENT/) { $eventFLAG="ON”, print “FLAG=ON\n”;
    $event="”;
    $event=$event.$dataline;
    # next LINE;
  };
  if ($eventFLAG eq “ON”) { $event=$event.$dataline;
## print "event=",$event,"\n";
} if ($dataline=~/END:VEVENT/) {
    $eventFLAG="OFF", print "FLAG=OFF \n";

    ## now analyse the event string to find UID and TEXT
    print "NEW event found -- checking for word DIABETES\n";
    if ($event=~/DIABETES/i) {
        ## increment event counter
        $eventnumber=$eventnumber + 1;

        ## get UID
        print "DIABETES event found \n";
        # get UID
        $uid1 = index $event, 'UID :KAlarm--';
        $uid2 = index $event, 'SEQUENCE';
        print "uid1 = ",$uid1, "\n";
        print "uid2 = ",$uid2, "\n";
        $uid = substr($event, ($uid1+5), ($uid2 - ($uid1+5)));
        print "UID = ", $uid, "\n";
        
        ## get the index positions for TEXT and ACTION
        $text1 = index $event, 'TEXT';
        $text2 = index $event, 'ACTION';
        print "text1 = ",$text1, "\n";
        print "text2 = ",$text2, "\n";
        $text = substr($event, ($text1+5), ($text2 - ($text1+5)));
        print "TEXT = ",$text, "\n";

        ## cancel the event
        my $cancel= "kalarm --cancelEvent ".$uid;
        print "cancelling existing DIABETES alarm\n";
        print "$logg "cancelling existing DIABETES alarm\n";
        print "$logg "sending command: "$cancel,"\n";
        print "$logg "sending command: "$cancel,"\n";
        print "$logg "$cancel,"\n";
        ## if more than one DIABETES event to cancel, then need to
        ## pause slightly as it takes time for each cancel to take effect
        if ($eventnumber > 1) {sleep 2};
        system(qq($cancel))
        and croak "could not run $cancel ($OS_ERROR)" ;
        #Perl—best—practice p 280

    } ## cancel the event

    ## get the event
    print "now look at next event
";
} # end of if contains word DIABETES conditional
else{## print "NEW event found -- checking for word DIABETES\n";
print "NO DIABETES word in this event, so looking for next
   event\n";
   # print "event = ", $event, "\n";
   next LINE;
   ####---------------- 
   ## finally dump the event string and start again
};

# print "***", $dataline, "\n";
$dataline="";
} ## end of the input loop reading the \$calfile

print "no more events found -- terminating now\n";
print "------------------\n";
# print "event string = ", $event,"\n";

##===========================
## get the commandline options ( using Getopt::Long)
## Perl--best--practice p 309
my $trigger_time_mins = 30; # mins
my $repeat_interval_mins = 5; # mins
my $until_time_mins = 1440; # mins = 24hrs
#my $message = qq("DIABETES: ");

my $options_okay = GetOptions( 
   'trigger=i' => \$trigger_time_mins,  #--trigger expects an integer mins
   'interval=i' => \$repeat_interval_mins, #--interval mins
   'until=i' => \$until_time_mins,  #--until mins = 1440 =24hrs
   #'message=s' => \$message, #--message
);

#--------------------------
## use 2 trailing spaces (to separate items)
my $kalarm="kalarm ";
my $bell="-b ";  ## --b
my $color="-c red ";
#$trigger_time_mins=; ## starttime
#-------------------------
#$repeat_interval_mins=5; # mins
my $interval="0000".$repeat_interval_mins;
my $intervalb=substr($intervala,4);
   print \$logg "interval= ", $intervalb,"\n";
my $repeat_interval="".i "$intervalb:" ";
   #------------------------
my $message=qq(" DIABETES -- repeat blood sugar ");

   print \$logg "bell = ", $bell,"\n";
   print \$logg "color = ", $color,"\n";
   print \$logg "trigger mins = ", $trigger_time_mins,"\n";
   print \$logg "interval mins = ", $repeat_interval_mins,"\n";
print {$logg} "until mins = ", Suntil_time_mins,"\n";

#-- -------------------------------------
## determine the new 'trigger' time
## determine final time (= trigger−time)
my $trigger_unix=$time_now_unix+($trigger_time_mins*60); # secs
## get local time string
my $trigger_string=localtime($trigger_unix);
## get ymdhm of trigger_time
my $trigger_ymdhm=ymdhm($trigger_string);
  # use the subroutine
  print "trigger time hh:mm = ", $trigger_hhmm,"\n";
  print {$logg} "trigger time = ", $trigger_ymdhm,"\n";
## write the correct trigger string for the Kalarm commandline
my $trigger="−t ".$trigger_ymdhm." "; # two trailing spaces

#-- -------------------------------------
## determine the correct until time (add 24hrs)
my $until_unix= $time_now_unix+($until_time_mins*60); # secs
my $until_string=localtime($until_unix);
my $until_ymdhm=ymdhm($until_string);
  # print "until time = ", $until_hhmm,"\n";
  print {$logg} "until time = ", $until_ymdhm,"\n";
my $until="−u ".$until_ymdhm." "; # the period during which it repeats
# format is $until="−u 2005−12−13−15:36 ";

#-- -------------------------------------
## testing with file – use the KDE geometry option to get width correct
## $file = " −f /home/dick/allfiles/akalarm/perl/help.txt ";
## $out= $kalarm.$bell.$color.$trigger.$repeat_interval.$until.$message;
my $out= $kalarm.$bell.$color.$trigger.$repeat_interval.$until.$message;
print "setting new DIABETES alarm \n"
print "sending Kalarm string = ", $out,"\n"
print {$logg} "setting new DIABETES alarm\n"
print {$logg} "sending Kalarm string = ", $out,"\n"
system(qq($out))
  and croak "could not run $out ($OS_ERROR)";
  #Perl−best−practice p 280

####===============SUBS==============================================
## ymdhm($time−string)
sub ymdhm {  
  # passing only one time_string into array
  my ($time_string) = @_;  
  #-- processing parameter [$time_string] \n"
  ## routine modified from fields2PDATA.pl

###**********************SUBS******************************************
CHAPTER 14. DIABETES DECISION-SUPPORT SYSTEM

14.6 File viewer program (dn-tkviewer.pl)

```perl
# note the main items are <space> separated except hh:mm:ss
## format is: Sun Jan 25 13:24:35 2004
## format is: Sun Jan 5 13:24:35 2004
## note **** get /two/ spaces after the Month if days <10
## see SUB tedname() in launchcam12.pl
##----------------------------------------------------------------------------------------
# if /two/ spaces in posn 8 and 9 then remove /one space/
if (substr($time_string,7,2) eq " ") {
  substr($time_string,7,2," ");
}
## replace spaces with commas
$time_string =~ tr / /,/
## make an array
my @stgmt=spli$t(/[,]/, $time_string);
## $day=$stgmt[0];    ## not used
my $month=$stgmt[1];
my $date=$stgmt[2];
my $st=$stgmt[3];
my $year=$stgmt[4];
## $noitems=$stgmt+1;  ## not used
## now split the time hh:mm:ss
## but Kalarm requires that the month and date is in numerals
if ($month eq "Jan") {$month="01"}
if ($month eq "Feb") {$month="02"}
if ($month eq "Mar") {$month="03"}
if ($month eq "Apr") {$month="04"}
if ($month eq "May") {$month="05"}
if ($month eq "Jun") {$month="06"}
if ($month eq "Jul") {$month="07"}
if ($month eq "Aug") {$month="08"}
if ($month eq "Sep") {$month="09"}
if ($month eq "Oct") {$month="10"}
if ($month eq "Nov") {$month="11"}
if ($month eq "Dec") {$month="12"}
my $ymdhm=$year."-$month."-$date."-$st."-$year;
return $ymdhm;
}#end of sub

#---------------------
```
#!/usr/bin/perl
## RN—tkviewer.pl (modified from RNtkview.pl)
my $thisprog="[dn—tkviewer.pl]" ; ## used in error messages
##
## RWD Nickalls
## Dec 16, 2005
## a simple TK fileviewer (takes filename as argument)
##
## BOOK = Mastering Perl Tk (by: Lidie S and Walsh N (O'Reilly, 2002)
## to get FullScreen mode at startup (p 307)
## —geometry widthXheight+Xoffset+Yoffset (NO spaces**page 409)
## $ perl tklaunch2.pl —geometry 1028x768 —0—0 ## page 409
## system ("perl ./tklaunch2.pl —geometry 300x400—50—300") }
## see p 233 PerlTK book
## see TEXT widget p 162
##
## use warnings;
use strict;
use Tk;
use Carp;
use Fatal; ## to give good failure error messages
use Getopt::Long; ## for command—line (see my prog ...diabetes2.pl)
#
## get the commandline options ( using Getopt::Long)
## Perl—best—practice p 309
## to allow an Input filename to view
my $input_filename = '<>';
my $options_okay = GetOptions(   
  'in=s' => \$input_filename, # --in option expects a string
);
## usage = $ perl dn—tkviewer.pl --in filename
##
## if ($input_filename eq '<>'){croak "...ERROR -- filename not specified "$thisprog;
##
## define an error message for use later
my $errormessage="...ERROR -- can’t find filename "<",\$input_filename.">
" "$thisprog;
## note that this error mesage must be outside the if(){} statement
##
## if (—e $input_filename){
##
## now make the widget

my $topwindow= MainWindow -> new();
$topwindow -> title("XENON file: \$input_filename");
my $text = $topwindow -> Scrolled("Text",
  # background => 'LightGrey',
  # default background colour is a very pale grey
  font => ['courier', '14'],
 )
  -> pack();
open my $VIEWFILE, "<", $input_filename||croak $errormessage, " [code A]" ;
14.7 Error message widget program
(dn-errorbox.pl)

#!/usr/bin/perl -w
## RN-errorbox.pl (modified from rntkalarm.pl)
my $thisprog = "[rn-errorbox.pl]"; #to define this program--name in error messages

## RWD Nickalls
## April 26, 2006.
## message boxes for Xenon
## Useful books: page 301 Perl core languages (Little Black Book)
## usage: $ perl dn-errorbox.pl -- in "error message is...."
## requires use of the explicit -- in tag
##
## BOOK = Mastering Perl Tk (by: Lidie S and Walsh N (O'Reilly, 2002)
## to get FullScreen mode at startup (p 307)
## geometry widthXheight+Xoffset+Yoffset (NO spaces**page 409)
## $ perl tklaunch2.pl --geometry 1028x768 -0-0 ## page 409
## system ("perl ./tklaunch2.pl --geometry 300x400 -50-300") }
##
use Tk;
use Carp;
use Fatal;
use Getopt::Long; ## gets options from command--line (see my prog ...diabetes2.pl)

## get the commandline options ( using Getopt::Long)
## Perl--best--practice p 309
## to allow an Input filename to view
my $message = "'--";
my $options_okay = GetOptions( 'in=s' => \$message, # -- in option expects a string );
## usage = $ perl rn--tkviewer.pl -- in filename
## if ($message eq "'--"){croak "...ERROR -- message not specified ".$thisprog," $!");
##
## write the word ERROR underlined
my $error="ERROR
MESSAGE\n---
\n\n";
my $boxmessage = $error.$message;
#
#−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
$topwindow = MainWindow -> new();
$topwindow -> title("XENON");
$topwindow -> Label(-text => $boxmessage,
    -wraplength => 200,
    -padx => 10,
    -background => 'Yellow',
    -foreground => 'Black',
    -height => 10,
    -width => 35
) -> place(-anchor => 'n')
    -> pack();
    # - > pack(-side => 'top'); #, - > expand => 1); MainLoop;
##−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
14.8 Screenshots

Figure 14.2:
Screen showing the diabetes alarm widget (right) and the Linux command-line window (left). The widget displays 5 blue time-option buttons (20–60 minutes) which initiate the red interval alarm as shown in the following figure.

Figure 14.3:
Screen showing the alarm help-window (bottom left) which opens by clicking on the ‘HOWto use’ button. The help-window doubles as a diabetes management information as well as a help feature for using the alarm widget itself.
CHAPTER 14. DIABETES DECISION-SUPPORT SYSTEM

Figure 14.4:
Screen showing in addition the Linux alarm window (bottom right) which opens by clicking on the ‘alarm’ icon on bottom bar.

Figure 14.5:
Screen showing the pop-up diabetes alarm. Clicking the ‘close’ button causes the alarm to close and re-appear in 5 minutes. Once a blood sugar has been done, then a new interval alarm is set by clicking on one of the time-option buttons (20–60 minutes) on the diabetes widget.
Chapter 15

Data storage, files and formats

15.1 Introduction

The Camomile data program generates two groups of stored numeric data, known as D-data (raw data from the Datex monitor), and binlog-data (data consisting of <UnixTime><parameter-value> pairs; one file for each parameter).

15.2 Filenames—time/date encoding

The data files associated with each operation are held in a time/date encoded directory. For example, the filename for a record started at 14.34 hrs on 26 November 2001 would be in the directory

/allfiles/camomiletop/theatredata/2001-Nov-26-1434

15.3 D-data.

This is the raw data from the Datex AS/3 monitor, and is saved to the file port-0.dnn in the main operation directory.

/allfiles/camomiletop/theatredata/2001-Nov-26-1434/port-0.dnn

Each data record consists of 321 bytes, and is saved as a comma separated string of 8-bit ASCII codes (000–255).

The format of the D-data is as follows (note that each 5 second data episode is formatted into blocks of 19 lines, each line starting with an identifying code sequence (AS3000 → AS3018). The first line of each block gives the time and date information.

....

AS300,14:40:19,23-09-2004 (d/m/y) Datex AS/3 monitor
AS301,126,062,001,003,005,000,000,000,000,025,000,205,022,082,065,000,000,000,000,000,000,000,000,000
AS302,000,001,000,189,255,097,220,044,000,000,000,189,189,000,189,034,029,031,029,035
The Camomile data program stores the comma-separated `<UnixTime>`<parameter-value> pairs (see example below) for each parameter in a separate file (a single file for the whole operation); for example the file for the systolic blood pressure is named `bp-s.binlog`. These files are stored in the `/fields/` subdirectory, as follows: 

```
/allfiles/camomiletop/theatredata/2001-Nov-26-1434/fields/bp-s.binlog
```

Later, each file is broken down into separate 1-hour files (called `.g0n` files; eg `bp-s.g01`, `bp-s.g02`, etc), preparatory to printing.

### 15.5 Drug-data

The Camomile data program keeps a log of the operation, start time, end time, keyboard entries, entries from the pull-down menus (drugs, anaesthetists, surgeons), and details of Alarms ON and OFF, and bad checksums, as shown in the example below. This entry
is written in a \TeX format, and is further processed to obtain the printed-out form of the drug log which is placed in the patients notes.

\%\%camomile
\%\%Camomile (v 0.1_040413b[c:Apr 15 2004@12:10:32])
\BeginLog\{2004-09-23,14:38:16\}%
\VersionStamp\{Camomile\}{0.1\_040413b}{Apr 15 2004@12:10:32}%
\%\ TruncateLog=0
\Note\{192\}\{opened logfile "/home/dick/allfiles/camomiletop/theatredata/2004-Sep-23-1438/base.log"\}%
\%\Mark\{2004-09-23,14:38:16\}%
\EntryDevice\{2004-09-23,14:38:16\}\{project\}\{start\}%
\%\Mark\{2004-09-23,14:39:34\}%
\EntryAnaesthetist\{E\}\{2004-09-23,14:39:26\}\{Dr R. W. D. Nickalls et al\}\}\%
\Mark\{2004-09-23,14:44:24\}%
\EntryDevice\{2004-09-23,14:44:24\}\{datex as3\}\{bad checksum 204,172\}%
\%\Mark\{2004-09-23,14:44:29\}%
\EntryDevice\{2004-09-23,14:44:29\}\{datex as3\}\{bad checksum 204,172\}%
\%\Mark\{2004-09-23,15:16:31\}%
\EntryDrug\{2004-09-23,15:16:25\}\{Morphine\}\{2\}\}%
\%\Mark\{2004-09-23,15:17:23\}%
\EntryDrug\{2004-09-23,15:16:33\}\{Epidural = (marcain 0.25) 5mls\}\}\%
\%\Mark\{2004-09-23,15:17:34\}%
\EntryAlarm\{E\}\{2004-09-23,15:17:34\}\{Alarm Off\}\{BP\}\{160 / 75\}%
\%\Mark\{2004-09-23,15:17:35\}%
\EntryDrug\{2004-09-23,15:17:24\}\{Epidural fentanyl 100 mcg\}\}\%
\%\Mark\{2004-09-23,15:34:49\}%
\EntryDrug\{2004-09-23,15:34:38\}\{Gelofusin\}\{500 IN\}\}%
\%\Mark\{2004-09-23,16:32:33\}%
\EntryDrug\{2004-09-23,16:32:28\}\{Neostigmine + Glycopyrrolate\}\}\%
\%\Mark\{2004-09-23,16:32:35\}%
\EntryDevice\{2004-09-23,16:32:35\}\{project\}\{stop\}%
\Note\{205\}\{closing logfile\}%
\%\EndLog\{2004-09-23,16:32:35\}%
\%\%sof
Part IV

Data processing—inline printing module
Chapter 16

Printing module—overview

16.1 Introduction

The anaesthesia data accumulated by the Camomile data-program is output and stored in the /fields/ directory of the current operation directory ($projdir), in the form of .binlog files, each one associated with a given parameter field, containing a series of (⟨time⟩, ⟨parameter value⟩) pairs.

When the Camomile data-program terminates control returns to the coordinating Perl program launchcam12.pl, which currently coordinates the data processing preliminary to the physical printing of the Anaesthesia Record itself. The aim of the printing process is to access the stored data in the /fields/ directory, and plot it in graphic form on A4 paper in such a way that each A4 sheet shows 1 hour of data.

All the data manipulation is done by the following small Perl programs which are stored in the /.../camomiletop/datexsim/printfiles/ directory.

- base2texd.pl ... does some ASCII to TeX conversion to log file
- cam2gnnh.pl ... generates the .data and .gnn files
- launchcam12.pl ... runs the Camomile program
- plotgnnk2.pl ... coordinates printing module
- printall.pl ... prints the paper sheets
- ptanes6.tex ... TeX file for the graphs
- prtdrug2.sty ... TeX style option for printing module
- prtdrug.tex ... TeX file for the log file

We now address the printing process in some detail, covering the various steps from the raw /field/ output data (acquired by the Camomile data-program) to the production of the paper endpoint—the Anaesthetic Record—which is placed in the patient notes. The full code of the eight or so Perl programs is listed in the subsequent chapters.
16.2 The start-time

A key piece of information required by the printing process is the start-time of the operation (or in practice, the start-time of data collection). The start-time is required for two main reasons as follows

- To define the directory name \textit{(projdir)} of the current operation so all related information can be stored there. The start-time is determined by the Perl program \texttt{launchcam12.pl} by grabbing the Unix-time and Local-time. This start-time is then used to construct a 'time encoded directory' (TED) by passing the time parameter to the subroutine \texttt{tedtime()} resulting in a suitable directory string. For example a typical TED directory string is as follows.


- To determine the number of 1-hour A4 printed records (i.e. we subtract the start-time from the time associated with the last recorded data item.

The following extracts from the Perl program \texttt{launchcam12.pl} illustrate the relevant steps in making the time encoded directory name.

```
# [launchcam12.pl]
...
# grab the starttime as GMT and Unixtime
$timenowgmt = localtime;
$timenowunix = time();
$projdir = tedname($timenowgmt);
# add the / at the end of the dir
# (so Camomile-program makes the /fields/ subdirectory
$projdir = $projdir."/";
...
...
sub tedname{
  ## returns a date/time encoded filename--> $projdir;
  my $startgmtstring =$_[0];
  ## format is: Sun Jan 25 13:24:35 2004
  ## format is: Sun Jan 5 13:24:35 2004
  ## note get two spaces after the Month if days <10
  # if two spaces in posn 8 and 9 then remove one
  if (substr($startgmtstring,7,2) eq " ") {substr($startgmtstring,7,2," ");}
  ## now replace spaces with commas
  $startgmtstring =~ tr/ /,/
  ## make an array
  @stgmt = split (/[ ,]/, $startgmtstring);
  $day = $stgmt[0], $month = $stgmt[1], $date = $stgmt[2], $hms = $stgmt[3];
  $year = $stgmt[4];
  $noitems = $#stgmt + 1;
  ## now extract the hh:mm:ss part to get the hh:mm
  @hhmmss = split (/[ ,]/, $hms);
  $hour = $hhmmss[0], $min = $hhmmss[1];
```

## force two-digit for date (= day-of-month)
## as unix gmt uses only 1 char if less than 10
if ($date<10){$date='0'.$date};
## format the datestring as 2004-01-22-1341
$datestring="$year-$month-$date-$hour$min";
return "/home/dick/allfiles/camomiletop/theatredata/"."$datestring";
};

### 16.3 Running the Camomile data program

In practice the operation time encoded directory (project directory) is actually created by the Camomile data program. To this end the Camomile data program is passed the required project directory name ($projdir) at start-up. This is done using Camomile’s -P command-line switch (together with the name of a required configuration file) as follows (note that this is a Perl program, and so the command has to be issued as part of the Perl system() command).

```perl
# [launchcam12.pl]
... $conf="../conf2/c_as3rn.conf"
$projdir="/home/user/camomiletop/theatredata/2004-Mar-18-11.23/"
system("../tarballs/camomile-0.1.040411/camomile/camomile -A 1 -P $projdir -c $conf");
```

### 16.4 After the Camomile data program exits

Once the Camomile data program has terminated, we then create a subdirectory in the project directory (called /pdata/—the ‘p’ indicating that this subdirectory relates to Printing data) to contain all the files required for printing as well as those generated during the printing process. While this directory can be placed anywhere, it is convenient during the current development period to keep all the files and directories relating to a given operation together, while at the same time keeping the camomile raw data separate from the derived processed data.

```perl
# [launchcam12.pl]
... $projpdatadir=$projdir."pdata/";
mkdir $projpdatadir;
```

### Start-time

Since various programs need to know the start-time (both in Unix-time and in GMT-time) we now make these times available by writing them to a special ASCII file (text file) called starttime.dat, which can then be read by any process needing this important information. The starttime.dat file is written by the program launchcam12.pl, as follows.

```perl
# [launchcam12.pl]
...
open (outfile1, ">$destinationfilename1"")
    ||die "ERROR: can't create file <starttime.dat>\n";
print (outfile1 "%% file name: startfile.dat: created $timenowgmt\n");
print (outfile1 "%% file generated by <launchcam.pl> RWD Nickalls\n");
print (outfile1 "%% file read by <plotgnnk2.pl> \n");
print (outfile1 "projectdir,$projdir\n");##use commas no spaces
print (outfile1 "starttime,$timenowunix,$timenowgmt\n");##no spaces
close (outfile1);

A typical starttime.dat produced by this code is as follows.

%% startfile.dat: created Mon Mar 29 10:26:28 2004
%% file generated by <launchcam.pl> RWD Nickalls
%% file read by <plotgnnk.pl>
projectdir,/home/dick/allfiles/camomiletop/theatredata/2004-Mar-29-1026/
starttime,1080552388,Mon Mar 29 10:26:28 2004

Note that we deliberately use commas to separate the key data-strings in the last two lines, as we can then easily manipulate the data-strings using the Perl split command for putting the relevant data-strings into arrays.

Copy required software tools

We now copy a suite of files (required for the printing process) from the /datexsim/printfiles/ directory to the /pdata/ directory.

#][launchcam12.pl]
...
## now copy all the <printfiles> tools to the /projdir/pdata/ dir
print "copying files from /datexsim/printfiles/ to ....../project/pdata/ \n";
system ("cp -v ./printfiles/.* $projpdatadir");
print "...... done\n";

Now everything is in place so we now move to the /pdata/ directory in preparation for the next phase—data processing—and call the Perl coordinating program plotgnnk2.pl as follows.

#][launchcam12.pl]
...
chdir $projpdatadir;

Data processing—launch program plotgnnk2.pl

The next phase is to process all the data and generate all the necessary .dvi, .pdf and .ps files so we can then print them out at a suitable time (usually at the end of the operation), and keep copies for archiving. All the data processing is coordinated by the Perl program plotgnnk2.pl, so the next thing it to launch this program as follows.

#][launchcam12.pl]
...
print "... now calling <perl ./plotgnnk2.pl> \n";
system ("perl ./plotgnnk2.pl");
During the data processing we write comments to the screen and also write detailed comments to the log file printlog.txt. In addition we keep a detailed log of the start times for a number of parameter files as these files are created in 1-hour chunks—this data is collected in the file timefile.txt.

### 16.5  Reading the `starttime.dat` file

We read the `starttime.dat` file right at the beginning of data processing, in order to access (a) the unix start-time, and (b) the name of the operation directory. This information is on the first and second data-lines in the file. Both these parameters are passed by the coordinating program `plotgnnk2.pl` to the program `cam2gnnh.pl`.

### 16.6  Accessing the Camomile-stored data

Both these parameters are passed by the coordinating program `plotgnnk2.pl` to the program `cam2gnnh.pl` which creates all the parameter `.data` files, and from these generates all the `.gnn` files.

#### Access the parameter fields (`camomilefields2tex.c`)

The output data is stored by the Camomile data program in the project sub-directory `/fields/` and so our first task is to access the data in a suitable format using the software access tool `camomilefield2tex` (a C program). This utility allows us to grasp the data and store it in a form suitable for post-processing. Although the original data is currently stored in ASCII files, this may well change during development. An example of the current `sat.binlog` structure is as follows (`sat.binlog`).

```
## sat.binlog
1071580231,92
1071580236,92
1071580241,93
1071580246,93.5
1071580251,93
1071580256,93
1071580261,92.5
1071580266,92
...
...
```

Consequently, accessing the data via an access tool has the advantage that the post-processing can proceed independent of the particular data storage format.

The C program `camomilefield2tex` is a utility to access the stored data in a form suitable for post-processing (unfortunately this is awkward since it requires access to the `starttime.dat` file, and so this utility has since been simplified and rewritten in Perl so it gets the time by reading the data file itself, and is currently used in the stand-alone printing module—described in the next chapter). The current version of the program comes as `camomilefield2tex-0.1_040411.tgz` which expands to `/tarballs/camomilefield2tex-0.1_040411.tar.gz`. To install type: (do the `make install` as root).
To get the help info type:

```
$ camomilefield2tex --help
```

which gives:

```
-p <path> path of the /project/ directory
-f <parameter> parameter field name
-o <filename> output file name
-s <style> output style (tex, gnuplot)
-V version
--help this help information
```

Example of use.

```
camomilefield2tex -p $projdir -f sat -o sat.data -s gnuplot
```

We use the style `gnuplot` as this gives simple comma-separated fields which can be easily parsed by Perl.

### Calling the `camomilefield2tex.c` utility

The list of required parameter names is held in the array `@paramname` defined at the beginning of the program, as follows. In fact for thoracic anaesthesia we also need to display the ventilation plateau pressures (to be incorporated later).

```perl
@paramname = ("bp-s", "bp-d","ecg-hr","sat-hr","cvp","nibp-s","nibp-d",
"sat",
"o2-insp", "n2o-exp",
"co2-exp",
"tv-exp","co2-rr",
"vap-insp", "vap-exp", "mac-big" );
```

For each parameter-name we then generate a datafile by calling the utility program `camomilefield2tex` (the next line then generates all the `.gnn` files by calling the subroutine `makeGnnfiles`—see next section).

```perl
# [cam2gnnh.pl]
...  
@paramname = ("bp-s", "bp-d","ecg-hr","sat-hr","cvp","nibp-s","nibp-d",
"sat",
"o2-insp", "n2o-exp",
"co2-exp",
"tv-exp","co2-rr",
"vap-insp", "vap-exp", "mac-big" );

for ($j=0; $j<=$#paramname; $j=$j+1 )
{
    $ffile = $paramname[$j];
    $ofile = $projdir."pdata/"."$paramname[$j]".".data";
    ---> system ("camomilefield2tex -p $projdir -f $ffile -o $ofile -s gnuplot");
    ## now create all the .gnn files for the parameter
    makeGnnfiles($paramname[$j]);
}
```
The above `camomilefield2tex` command outputs all the stored parameter data for a given parameter into a file consisting of the following four comma separated fields on each line into the specified output file:

```
unix-time, gmt-time, elapsed-time, parameter-value
```

A typical example of the `sat.data` file is as follows. Note that the elapsed-time parameter on the first line is zero, and that both the unix-time and the elapsed-times increase in steps of 5 seconds (data is output from the Datex monitor every 5 seconds).

```
# [sat.data]
1071580231, 2003:12:16:13:10:31, 0, 92.000000
1071580261, 2003:12:16:13:11:01, 30, 92.500000
... ... ...
```

Armed with the above `.data` file for a given parameter, then we proceed to generate from this a series of 1-hour `.gnn` files, as described in the next section.

### Generate 1-hr `.gnn` files with subroutine `makegnnfiles()`

This role of this subroutine is to generate from the above parameter `.data` file (which may contain many hours of data) a series of 1-hour `.gnn` files suitable for use by the GNUplot graphing program. The `makegnnfiles()` subroutine is part of the Perl program `cam2gnnh.pl` (which is itself called by the co-ordinating Perl program `plotgnnk2.pl`). The subroutine is called with the field parameter name as follows.

```
makegnnfiles($paramname[$j]);
```

Calling the subroutine `makegnnfiles()` converts each of the raw output parameter data-files (`.data` files) into a series of 1-hour two-column space-separated data-files suitable for accessing by gnuplot. For example, a 4-hr `sat.data` file would be converted into four 1-hour files as follows: `sat.g01`, `sat.g02`, `sat.g03`, `sat.g04`.

The `makegnnfiles()` subroutine also makes the elapsed time for each file relative to the beginning of each hour by using the new computed “start-time” for each file as the zero-time, i.e. elapsed time within a `.gnn` file will run from 0—3599 secs (i.e. just 1 hour). We have three ⟨space⟩ delimited fields namely ⟨elapsed-time-(local)⟩, ⟨parameter⟩, ⟨unix-time⟩.

The subroutine figures out how to split up the `.data` file into 1-hour chunks by comparing the difference between the operation start-time and the unix-time on each line. Note that both the unix-time and gmt-time are passed to the `cam2gnnh.pl` program by the calling program (`plotgnnk2.pl`). If the elapsed time exceeds 1-hour, then the current `.gnn` file is closed, and the next one opened etc.

In practice, however, data is only retained at approximately 30–45 second intervals (this interval can be varied depending on the requirements). So although the data is originally stored every 5 seconds, the actual printed data is thinned out somewhat, purely
because there is a limit to what density of data can usefully be printed to the Anaesthesia Record. If better resolution is required, then higher resolution printing can be performed at a later date.

```perl
# [cam2gnnh.pl]
...
sub makegnnfiles {
    ## get the starttimeUNIX passed from commandline value --> @ARGV
    ## the starttimeUNIX is obtained originally from file <starttime.dat>
    $starttimeunix = @ARGV[0];
    # passing only one name into array
    my ($file) = @_;
    print "---processing parameter [$file] \n";
    # add the file-ending .dat
    $infilename=$file.".data"; ###*
    print "---the input filename is [$infilename] \n";
    open (infile, "<$infilename")||die "ERROR: can't find file $infilename \n";
    # make time-dependent out filename
    # start with hour set to zero
    $hour=0;
    #----------------------
    # start inputting lines of data
    # need to get the time associated with line 1
    #
    $interval=45; #secs
    $oldelapsedtime=0;
    LINE: while (<infile>){
        next LINE if /^#/; #skip comments
        next LINE if /^%/; #skip comments
        next LINE if /^$/; #skip blank lines
        # grab the whole line as a string
        $dataline = $_;
        # place the params into an array
        @value=split (/[\,]/, $dataline);
        # assign the elapsedtime and param values
        $unixtime=$value[0];
        $gmttime=$value[1]; #GMT yyyy:mm:dd:hh::mm:ss
        $elapsedtime = $value[2]; #elapsed-time (secs)
        $paramvalue=$value[3];
        chomp($paramvalue); # remove the line-ending to help maths
        #----------------------
        # multiply the rr values by 50 (to make them fit range 0--1000)
        if ($file eq "co2-rr"){$paramvalue=$paramvalue * 50};
        #----------------------
        ## save data only every $interval (secs)
        $elapsedtime=$unixtime-$starttimeunix; ## determine true elapsedtime
        if ($elapsedtime < $oldelapsedtime +$interval)
            {next LINE}
        else{
            $oldelapsedtime = $elapsedtime
        }
    }
```
#-----------------
#now print data into 1 hr files
# make NewElapsed time relative to beginning of new hour
# hour 1 = first real hour
# hour will be zero on first run thro algorithm so goes to else...
if ($elapsedtime <$hour *3600){
    $space=" ";
    # calculate new elapsed time from begining of new hour
    $newet=$elapsedtime-3600*($hour-1);
    print (outfile "$newet $space $paramvalue $space $unixtime\n");
} else{
    # close existing gnn file and open a new one (gnn+1)
    close (outfile);
    $hour=$hour + 1;
    #use two digits for the filename extension eg .g04
    if ($hour <10){$hour="0".$hour};
    $gnudatafilename=$file.".g".$hour;
    print "---the new output filename = $gnudatafilename \n";
    open (outfile,">$gnudatafilename")||die "can't open the outfile \n";
    # write some headers to the outfile
    $outfileheader1="## Camomile gnuplot datafilename = $gnudatafilename";
    $outfileheader2="## date?";
    print (outfile "$outfileheader1\n");
    print (outfile "$outfileheader2\n");
    # write info to the timefile
    print (timefile "$hour, $unixtime, $gmtime, $gnudatafilename\n");
    $space=" ";
    # calculate new elapsed time from begining of new hour
    $newet=$elapsedtime-3600*($hour-1);
    print (outfile "$newet $space $paramvalue $space $unixtime\n");
} #end of else{
} #end o while
close (infile);
close (outfile);
}

A typical example of a .gnn file (the file sat.g03) is as follows. There are three fields (elapsed-time, parameter-value, unix-time) which are space-separated. In this example the data was collected every 30-40 seconds or so and the elapsed-times are seen to be 31, 76, 121, ... etc. The unix-time field is retained as a check. The 03 in the filename extension .g03 indicates that it represents data collected during the third hour.

##[sat.g03]
31  87.500000  1080559619
76  88.000000  1080559664
121 89.500000  1080559709
166 93.000000  1080559754
211 94.500000  1080559799
256 95.000000  1080559844
The log-file (`timefile.txt`)

Concurrently with the previous process, the program `cam2gnnh.pl` creates the `timefile.dat` file which holds the start-times for each of the `.gnn` files (see below). This file is very useful as a check on the functioning of the `cam2gnnh.pl` program.

```
# [timefile.txt]
...
01, 1071580301, 2003:12:16:13:11:41, bp-s.g01
02, 1071583865, 2003:12:16:14:11:5, bp-s.g02
03, 1071587465, 2003:12:16:15:11:5, bp-s.g03
...
01, 1071580276, 2003:12:16:13:11:16, sat.g01
02, 1071583840, 2003:12:16:14:10:40, sat.g02
03, 1071587440, 2003:12:16:15:10:40, sat.g03
...
```

The base.log file (`baselog.data`)

After processing all the parameter fields → `.gnn` files we then access (extract) the anaesthetists log file (`base.log`) using the `camomilefield2tex` utility as before, only this time using the `.l` switch and the `-s tex` option since we are wanting to access a log file.

```
# [cam2gnnh.pl]
...
  system ("camomilefield2tex -p $projdir -l base -o baselog.data -s tex");
```

Note that since we are running this command from within the `/pdata/` subdirectory then the default location for the output files is the current directory.

16.7 Write the GNUplot scripts for each graph

Each 1-hour page of the Anaesthesia Record consists of six separate graphs, each showing a time plot of several parameters. Each separate graph requires its own so called `.gnu` file (script) which sets up the graph structure and plots each parameter inside it. All this is coordinated by the Perl program `plotgnnk2.pl`, and so we will look in more detail how this is done.

Each parameter to be plotted has its own `.gnn` parameter file (not absolutely necessary but very convenient in practice—see previous section). To facilitate this, we

---

\(^1\)Not to be confused with the `.gnn` data files.
arrange that each 1-hour .gnn file has its elapsed time starting from zero, which greatly simplifies the plotting process.

The most difficult part of generating the .gnu files (one file per graph) is to construct the time-base, such that all .g01 parameter files are plotted on graphs showing the start and end times of the first hour, and also of the 15-minute vertical lines which are also drawn.

The timebase parameter $\textit{timeline}$

The time markings along the x-axis are drawn using the GNUplot set xtics() command which, in this case, takes a complicated parameter which is the string $\textit{timeline}$. In practice, for each hour the particular time-base used will be the same for all graphs drawn using parameters values from files having the same gnn value; say, .g02 files for example.

The following code determines this string for each hour, tailoring it to accomodate the time interval associated with each .gnn value, so as we move from one hour to the next then the time associated with each hour increases accordingly.

```plaintext
# [ploggnnk2.pl]
...
# determine the earliest start time from G01 files in timefile.dat file
# put the start-time-GMT[year:month:day:hrs:mins:sec] into an array
# then determine how many hours worth of Gnn files there are
# $st is the start-time hh:mm:ss from the <starttime.dat> file (see above)
$JJ=gnnmax("01");  ## returns gnnMax
print (printlog "start-time = [$st] 
");
print (printlog "GnnMax = $gnnmax 
");
# extract the separate hh, mm, ss values
@start_time= split (/[:]/, $st);
$starthour = $start_time[0];
$startminute=$start_time[1];
$startsecond=$start_time[2];
#----------------
# now print all the graphs for all Gnn files from 01 to GnnMax
for ($gnn=1; $gnn<=$gnnmax; $gnn = $gnn+1)
{
  # first determine time in secs to the begining of next full hour
  $deltah = 3600 - ($startminute*60 + $startsecond);
  # generate correct start-hour depending on Gnn value
  $h = $starthour + $gnn;
  $hminus1=$h-1;  $hplus1=$h+1;
  if ($h==0) {$hminus1=23};
  if ($h==23) {$hplus1=0};
  $q=900;  $qq=1800;  $qqq=2700;  $qqqq=3600;
  # force 24hour clock
  if ($h <10){$h="0".$h};
  if ($hminus1 <10){$hminus1="0".$hminus1};
  if ($hplus1 <10){$hplus1="0".$hplus1};
  $deltahminusqqqq=$deltah-$qqqq;
  $deltahminusqqq=$deltah-$qqq;
```

$deltahminusqq=$deltah-$qq;
$deltahminusq=$deltah-$q;
$deltahplusqqqq=$deltah+$qqqq;
$deltahplusqqq=$deltah+$qqq;
$deltahplusqq=$deltah+$qq;
$deltahplusq=$deltah+$q;

#---------------
$t1 = "$hminus1.00"." $deltahminusqqqq;
$t2 = "$hminus1.15"." $deltahminusqqq;
$t3 = "$hminus1.30"." $deltahminusqq;
$t4 = "$hminus1.45"." $deltahminusq;
$t5 = "$h.00"." $deltah;
$t6 = "$h.15"." $deltahplusq;
$t7 = "$h.30"." $deltahplusqq;
$t8 = "$h.45"." $deltahplusqqq;
$t9 = "$hplus1.00"." $deltahplusqqqq;
$timeline="$t1,$t2,$t3,$t4,$t5,$t6,$t7,$t8,$t9";

Armed with the time-base we can start making (write to) the .gnu files. In the following we illustrate the code for writing the sat.gnu script file (which will be processed by the GNUplot program eventually). First we check that the 'hour' value incorporated into the .gnn string always has two digits (i.e. 4 → 04 and hence we obtain g04), and defining the graph height to be used, we then open the output file and proceed.

#[plotgnnk2.pl]
...
# first make sure the gnn string has three characters
if ($gnn <10){$gnn="0".$gnn};
# define the graph heights
$smallheight=0.43; ## for all other graphs
...
...
## now create the sat file -------------------------
open(satfile, ">plot-sat.gnu")
  ||die "ERROR: can't open plot-sat.gnu file
";
print (satfile "#!/usr/bin/gnuplot
");
print (satfile "# plot-sat.gnu script made by plotgnnk2.pl
");
print (satfile "set terminal latex
");
print (satfile "set output "plot-sat.pic" \n"");
print (satfile "set size 1.40,$smallheight\n"");
print (satfile "set xtics($timeline)\n"");
print (satfile "set ytics ("80","90","100")\n"");
print (satfile "set y2tics (80,90,100)\n"");
print (satfile "set nokey\n");
print (satfile "set grid\n");
print (satfile "xmin=0;xmax=3600\n");
print (satfile "ymin=80; ymax=100\n");
print (satfile "plot [xmin:xmax][ymin:ymax] \n"");
$satfilename="sat..".$gnn;
$fo2filename="o2-insp".".g"."gnn;

if (-e $satfilename)
    {print (satfile " \
"$satfilename" using 1:2 with linespoints 4 8,\\n")}
else {print (printlog " ---**** no sat.gnn files\\n")};

if (-e $fo2filename)
    {print (satfile " \
"$fo2filename" using 1:2 with linespoints 4 10,\\n")}
else {print (printlog " ---**** no fo2.gnn files\\n")};

$dummyline = " -20 with lines 1 # dummy line";
print (satfile "$dummyline \\
")
close (satfile);

It is significant here that in the last few lines of this code we have used the line

print (bpfile "$dummyline \\
")

This is to solve a problem which would arise should one or more of the parameter files not exist, as in this situation GNUplot graph plotting would fail since it requires that the final line must not have a comma at the end. By using a ‘dummy’ line (which has no comma and only plots a point below the graph (-20) and hence is never visibly plotted) as the final line, we are able to handle the failure of all or some of the parameter lines which therefore can all have a terminal comma.

### 16.8 Run GNUplot on all the .gnu files

Once all the .gnu files have been written, then we run GNUplot on each one to generate each figure in \LaTeX\;e picture format. Each printed sheet has five figures arranged horizontally from top to bottom. The legends are on the right hand side so they are not obscured by the binding when placed in the patient notes.

`#[plotgnnk2.pl]
...`

print (printlog "---running GNUPLOT on all the .gnu files\\n")
system ("gnuplot plot-bp.gnu");
system ("gnuplot plot-sat.gnu");
system ("gnuplot plot-fo2.gnu");
system ("gnuplot plot-co2.gnu");
system ("gnuplot plot-tv.gnu");
system ("gnuplot plot-vap.gnu");
print (printlog "............GNUPLOT ... done\\n");

### 16.9 Write the header line for the printouts

Each printed sheet has a header indicating the start-time (GMT and unix) and the .dvi filename (which indicates which hour the sheet refers to) as follows:

Record start-time: Thu Feb 12 12:11:19 2004 \ Unix 1076587879 anes-04.dvi
This is written to a file (header.dat) as follows, and then read back when needed for printing.

```perl
# [plotgnnk2.pl]
...
print "writing the <gnnheader.dat> file to contain header for Anes record \n";
open (outfile5, ">gnnheader.dat")||die "ERROR: can't create file <gnnheader.dat>\n";
$timenow = localtime;
print (outfile5 "%% gnnheader.dat: created $timenow\n");
print (outfile5 "%% file generated by <plotgnnk2.pl> RWD Nickalls\n");
print (outfile5 "\\header{$starttimeunix}{$originalgmt}{$fname}\n");
close (outfile5);
```

### 16.10 Typeset the graphic pages using \LaTeX\(2\epsilon\)

We now typeset the graphic pages and create the output formats .dvi, .ps, and .pdf on the fly. The \TeX\ file for the graphs is prtanes6.tex. The style option is prtdrug2.sty. We create the PostScript files using dvips. We create the .pdf files using pdflatex.

```bash
print (printlog "---running LATEX on prtanes6.tex\n");
system ("pslatex prtanes6.tex");
$dvifilename="anes-"."$gnn.".dvi";
# copy the .dvi file to have a gnn.dvi filename
system ("cp -v prtanes6.dvi $dvifilename");
# make the .ps files
$psfilename="anes-"."$gnn.".ps";
system ("dvips $dvifilename -o $psfilename");
print (printlog ".........LATEX ...done\n");
# now make the pdf files
system ("pdflatex prtanes6.tex");
$pdffilename="anes-"."$gnn.".pdf";
# copy the .pdf file to include a ..gnn.pdf filename
system ("cp -v prtanes6.pdf $pdffilename");
```

### 16.11 Typeset the drug file using \LaTeX\(2\epsilon\)

Processing the drug file (log file) is slightly more complicated owing to the fact that the typesetting is done using \LaTeX\(2\epsilon\). Consequently, since the anaesthetists can enter data using the keyboard we need to filter out all non-\TeX\ material (essentially to “escape” certain ASCII characters; for example, we would modify % rightarrow \% etc). This conversion is currently done by the Perl program base2texd.pl, which processes the original log-file (baselog.data) to the “filtered” file baselognew.data.

We now typeset the “filtered” drug-file and create the output formats .dvi, .ps, and .pdf on the fly as before. The \TeX\ file for the graphs is prtdrug.tex. The style option is prtdrug2.sty. We create the PostScript files using dvips. We create the .pdf files using pdflatex.
16.12 Printing the paper sheets

Finally, we print out all the sheets making up the Anaesthesia Record. This currently consists of one or more ‘drug’ sheets (the log file), together with a number of 1-hour graphic sheets presenting the measured parameters. These are usually printed out in the operating theatre and placed in the patient notes.

In practice a small Perl program (printall.pl) sends the final files to the printer in reverse order as follows.

```perl
#!/usr/bin/perl
# printALL.pl
# do graphs in reverse order
if (-e "anes-10.dvi") {system("dvips anes-10.dvi")} else{};
if (-e "anes-09.dvi") {system("dvips anes-09.dvi")} else{};
if (-e "anes-08.dvi") {system("dvips anes-08.dvi")} else{};
if (-e "anes-07.dvi") {system("dvips anes-07.dvi")} else{};
if (-e "anes-06.dvi") {system("dvips anes-06.dvi")} else{};
if (-e "anes-05.dvi") {system("dvips anes-05.dvi")} else{};
if (-e "anes-04.dvi") {system("dvips anes-04.dvi")} else{};
if (-e "anes-03.dvi") {system("dvips anes-03.dvi")} else{};
if (-e "anes-02.dvi") {system("dvips anes-02.dvi")} else{};
if (-e "anes-01.dvi") {system("dvips anes-01.dvi")} else{};
# print the drug sheet last (on top)
if (-e "anes-drug.dvi") {system("dvips anes-drug.dvi")} else {};
```
Chapter 17

Typesetting programs

17.1 prtanes6.tex

\documentclass[a4paper]{article}
\usepackage[dvips]{color,graphicx}
%\usepackage[pdftex]{color,graphicx}
\usepackage{times}
\usepackage{latexsym} %% for \Box symbol
%\%\%\usepackage{graphicx} %% for rotate[]{} in dvips/pdf only
\usepackage{prtdrug2}
\usepackage{miscrwdn} %% needed for cupBOX and cupframebox

%%% redefine the \tenrm command output by GNUplot
\newcommand{\tenrm}{\rmfamily\normalsize}

%%%------symbols modified from my medicine.sty--------
\newcommand{\jotwo}{\ensuremath{\mbox{\scriptsize O}_2}}
\newcommand{\jcotwo}{\ensuremath{\mbox{\scriptsize CO}_2}}
\newcommand{\etcotwo}{ET\ensuremath{_{\jcotwo}}}
\newcommand{\fiotwo}{F\ensuremath{\textsc{i}_{\jotwo}}}
\newcommand{\ntwoo}{\ensuremath{\mbox{N}_2}\mbox{O}}

%%%---------------

\voffset -1.75cm
\oddsidemargin -11mm
\textwidth 20cm
\textheight 25cm  \% was 25.5

\begin{document}
\% note that all the empty lines are essential for the layout
\% as \vspace{} requires a preceeding emptyline

238
\thispagestyle{empty}
\vspace*{-1.8cm}
\newcommand{\patientlabel}{\framebox{\rule[-10mm]{0cm}{3.3cm}\hspace{2.2cm}Patient label\hspace{2.2cm}}}
\noindent\hspace{10.1cm}\patientlabel
\vspace{-3.5cm}
\noindent\hspace{2.3cm}{\color{blue}\LARGE AN\AE STHESIA RECORD}
\vspace{3mm}
\noindent\hspace{5.2cm}\textsf{Nottingham City Hospital} \hspace{2.27cm}{\color{blue}\textsf{NHS Trust}}
\vspace{-2mm}
\input{gnnheader.dat} \hspace{1.7cm}
\vspace{-4mm}
\input{plot-bp.pic}\hspace{-20}
\vspace{-4mm}
\input{plot-sat.pic}\hspace{-20}
\vspace{-4mm}
\input{plot-fo2.pic}\hspace{-20}
\vspace{-4mm}
\input{plot-co2.pic}\hspace{-20}
\vspace{-4mm}
\vspace{-2mm}
\vspace{-2mm}
\noindent\input{plot-tv.pic}\hfill

%%---------------------------------------------------------
%% now put on the right axis for Resp rate (0, 5,10,15,20).
\vspace{-32.5mm}\noindent\hspace{158.5mm} 20 $\bullet$
\vspace{1.4mm}\noindent\hspace{158.5mm} {15}
\vspace{1.4mm}\noindent\hspace{159mm} {10} \% was .8mm
\vspace{1.4mm}\noindent\hspace{159mm} {5}
\vspace{1.4mm}\noindent\hspace{159mm} {0} \% was 189
%
\vspace{-4mm}
%*************************
\vspace{7mm}
\noindent\input{plot-vap.pic}\hfill
%******************
%%==========labels=====================
\hspace{16.75cm}% was 16.5
\begin{minipage}{2cm}
%%---BP---
inv BP $\circ$
NIBP $\Box$
\vspace{6mm}
HR$_{oxim}^\bullet\mbox{--}\bullet$
HR$_{ecg}^\bullet$
\vspace{5.5mm}
CVP ---
%------SAT----------
\vspace{18.5mm}
SAT $\circ$
\vspace{6.5mm}\fiotwo \ $\bullet$
\vspace{12.1mm}\ntwoo \ $\Box$
$P_{plateau}^{\circ}$

$\text{etcotwo} \ \diamond$

$TV_{exp}^{\Box}$

$VAP_{insp}^{\ldots}$

$VAP_{exp}^{---}$

$MAC_{age}^{\Diamond}$
17.2 \texttt{prtdrug2.sty}

\begin{verbatim}
\% prtdrug2sty
\% rwd nickalls April 15, 2004
\% LaTeX version + modification of Simon's Camomile-record.sty
\%------------------
\typeout{***************************************************}
\typeout{* This is prtdrug2.sty \<04 Feb 2004\>}
\typeout{* Copyright (c) Camomile Group 2003-4}
\typeout{* Written by RWD Nickalls & Simon Dales}
\typeout{***************************************************}
\%---------------------
\newcommand{\n}{{\par\vspace{0.15\baselineskip}}}\%
\%---------------------
\newcommand{\BeginLog}[1]{{\noindent{\bfseries Begin Log at #1\vspace{0.5\baselineskip}\hrule\vspace{0.5\baselineskip}\n}}}\%
\%---------------------
\newcommand{\EndLog}[1]{{\strut\vspace{-0.7\baselineskip}\hrule\vspace{0.5\baselineskip}\noindent{\bfseries End Log at #1\n}}}\%
\%---------------------
\newcommand{\VersionStamp}[3]{}% do nothing
\% #1#2#3 = \{Camomile\}{0.0\_040120}{Feb 3 2004@15:53:15}
\newcommand{\VersionStamp}[3]{{\noindent{\bfseries Computer Program:} #1; Version \url{#2}, #3\n}}\%
\%---------------------
\newcommand{\Note}[2]{{\noindent{\bfseries Note} (#1): #2\n}}\%
\%---------------------
\newcommand{\Mark}[1]{} %% do nothing\%
\%---------------------
\%---------------------
\newcommand{\EntryDevice}[3]{} %% do nothing\%
\%-------------------------
\newcommand{\EntryAlarm}[5]{{\noindent#2\myspace{\bfseries Alarm:} \(#4: \#5\)\n}}\%
\%#1#2#3#4#5 { E,time,alarmon/off, alarm, value}
\%---------------------
\newcommand{\myspace}{\hspace{6mm}} %% two spaces
\%---------------------
\%\def\EntryDrug#1#2#3#4{% time,drug,qty,comment
\newcommand{\EntryDrug}[4]{\noindent{\bfseries Drug:} #1, (#2, #3)\n}%
\%\def\EntryDrug\#1\#2\#3\#4\{\% time,drug,qty,comment
\newcommand{\EntryDrug}[4]{\noindent#1\myspace{\bfseries Drug:} \(#2 \#3\)\n}%
\%---------------------
\newcommand{\EntryTimer}[4]{{\% count30=#3 \% seconds (see Knuth p 118)\%
\divide\count30 by 60 \% gives the minutes\%
\noindent\myspace{\bfseries Timer:} \interval set to \the\count30 \% mins \(#4\)\n}%
\%\def\EntryTimer\#1\#2\#3\#4\{\% time0,timeout,del,comment
\newcommand{\EntryTimerDiabetes}[4]{{\%
\count30=#3 \% seconds (see Knuth p 118)\%
\end{verbatim}
\divide\count30 by 60 \%
% gives the minutes\%
\noindent#1\myspace{\bfseries Timer (diabetes):}\ \ review in \{\the\count30\ mins\} (#4)\n%\%
% note Simon actually has 5 fields for diabetes timer
%\def\EntryTimerDiabetes#1#2#3#4{% time0,time1,delay,comment
%----------------------------------------------------------
\newcommand{\EntryAnaesthetist}[4]{\noindent#2\myspace{\bfseries An{\ae}sthetist:}\ \ #3}%%
% #1#2#3#4{type,time,name,comment}
%----------------------------------
\newcommand{\EntrySurgeon}[4]{\noindent#2\myspace{\bfseries Surgeon:}\ \ #3}%%
% #1#2#3#4{type,time,name,comment}
%------------------------------------------------------------------
\newcommand{\EntryPatientEvent}[6]{\noindent#1\myspace{\bfseries Patient:}\ \ #4 yrs, #2 kg, #3 cm, #5}%%#1#2#3#4#5#6{time,mass,height,age,isMale,comment
%------------------------------------------------------------------
\newcommand{\EntryPatientEventJ}[7]{\noindent#1\myspace{\bfseries Patient:}\ \ #5 yrs, #3 kg, #4 cm, #6, (#7)}%%#1#2#3#4#5#6#7{comment,time,mass,height,age,(M/F), Jobno
%------------------------------------------------------------------
\def\Conc#1#2{% legend,value
 #1=#2%
 }%
%-------------------
\def\Dosage#1#2{% legend,value
 #1=#2%
 }%
%-----------------------------------------------
\newcommand{\EntryBloodLoss}[3]{\noindent#1\myspace{\bfseries Blood Loss:}\ \ #2 #3}%%
%\def\EntryBloodLoss#1#2#3{% time,amount,comment
%--------------------------------------------
\newcommand{\EntryUrine}[3]{\noindent#1\myspace{\bfseries Urine output:}\ \ #2 #3}%%
%\def\EntryUrine#1#2#3{% time,amount,comment
%--------------------------------------------
\newcommand{\EntryComment}[2]{\noindent#1\myspace{\bfseries Comment:}\ \ {$\left\{\parbox{10cm}{#2}\right.$} \n}%%\def\EntryComment#1#2{% time,comment
%%==============================================prtanes stuff here==================================%
% header for the prtanes graph file
\newcommand{\header}[3]{\vspace{3mm}
\hfill Record start-time: #2 \hspace{5mm}unix #1\hspace{5mm}#3\hspace{3.3cm}
\vspace{3mm}}
% uses the three parameters #1#2#3 ={ unixtime, gmttime, gunfilename}
%\%eof
17.3 prtdrug.tex

%%% prtdrug.tex
%%% testing inputting base file
%%%----------------
\documentclass[a4paper]{article}
%\usepackage{camomile-record}
\usepackage[dvips]{color,graphicx}
\usepackage{times}
\usepackage{geometry}\geometry{hscale=0.8,vscale=0.7}
\usepackage[url]
\usepackage{decimal}
\usepackage{prtdrug2}
\usepackage{fancyhdr}
\begin{document}

%%%===========header=============================
\newcommand{\patientlabel}{%
    \framebox[\rule[-10mm]{0cm}{3.3cm}]{
        \hspace{2.2cm}Patient label\hspace{2.2cm}}}

\noindent\hspace{10.1cm}\patientlabel
\vspace{-3.5cm}
\noindent\hspace{2.3cm}{\color{blue}\LARGE AN\AE{}STHESIA RECORD}
\vspace{3mm}
\noindent\hspace{5.2cm}\textsf{Nottingham City Hospital} %\hspace{2.3cm}
\noindent\hspace{5.0cm}\hspace{2.27cm}{\color{blue}\textsf{NHS Trust}}

%------
\vspace{2.2cm}\vspace{4mm} 1.7
%\noindent\hspace{2mm}\vbox{%
%\begin{tabular}{|ll|}
%\hline
%\sc Date: \rule{0pt}{12pt} & \today \ \ \\
%\sc Operation: & \hspace{5.5cm} \ \ \\
%\sc Anaesthetists: & RWD Nickalls \textit{et al.} \ \ \\
%\sc Surgeons: & \ \ \\
%\hline
%\end{tabular}
%}
%\hline
%%==================================
\pagestyle{fancy}
\fancyhead{}
\fancyfoot{}
\rhead{Anæsthesia Record---Log File\hspace{1cm}\thepage}
\lhead{\thepage}
\lfoot{\hrule\vspace{0.5\baselineskip}
copyright RWD Nickalls, S Dales \& A Nice 1994--2004: {\sc anæsthesia record system---camomile---}{\newline}{\textit{Linux}}
{\newline}{\textsc{email} \textit{dicknickalls@compuserve.com}}
%
\%\%---------------------
\%\% check location of the base.log file
\%\% getting the base.log file from parent dir
\typeout{** getting the base.log file from parent dir}
\input{baselognew.data}
\%\%---------------------
\end{document}
\%\%==========footnote=============================17.4 printall.tex
#!/usr/bin/perl
### printALL.pl
## prints all the anes-nn.dvi and anes-drug.dvi files
#---------------------------------------
#-w ## turned off for the moment
##------------------
## do in reverse order with drug on top
if (-e "anes-10.dvi") {system("dvips anes-10.dvi");} else{};
if (-e "anes-09.dvi") {system("dvips anes-09.dvi");} else{};
if (-e "anes-08.dvi") {system("dvips anes-08.dvi");} else{};
if (-e "anes-07.dvi") {system("dvips anes-07.dvi");} else{};
if (-e "anes-06.dvi") {system("dvips anes-06.dvi");} else{};
if (-e "anes-05.dvi") {system("dvips anes-05.dvi");} else{};
if (-e "anes-04.dvi") {system("dvips anes-04.dvi");} else{};
if (-e "anes-03.dvi") {system("dvips anes-03.dvi");} else{};
if (-e "anes-02.dvi") {system("dvips anes-02.dvi");} else{};
if (-e "anes-01.dvi") {system("dvips anes-01.dvi");} else{print "no anes-nn.dvi files available\n"};
# print the drug sheet last (on top)
if (-e "anes-drug.dvi") {print "...printing file anes-drug.dvi\n"; system("dvips anes-drug.dvi");}
else {print "no anes-drug.dvi file available\n"};
##---------------------------------------
__END__
Part V

Data processing—stand-alone printing module
Chapter 18

Printing—the stand-alone (SA) module

18.1 Introduction

Although the automated ‘in-line’ printing module (described in chapter X) worked well in processing the data immediately at the end of an operation (by clicking on the ‘print last case’ button on the launcher widget), it was difficult to implement retrospectively—for example, when wanting to re-processing a different database of .binlog files (typically placed in the /fields/ subdirectory).

The /pdata/ sub-directory contains the original output of processed data. A typical directory structure of an operation database which, for example, started at 13:42 hrs on September 23, 2005, is as follows.

```plaintext
.../camomiletop/theatredata/2005-Sep-23-1342/
.../camomiletop/theatredata/2005-Sep-23-1342/fields/
.../camomiletop/theatredata/2005-Sep-23-1342/pdata/
```

A new ‘stand-alone’ printing module was therefore developed, which (a) was simpler (i.e. did not use Simon Dales’ camomilefields2tex C-program, or need to read the starttime.dat file), and (b) could be pointed at a particular /fields/ subdirectory to generate the full printable anaesthesia record in the usual way. The output of all data processed by this SA module is stored in a separate /PDATA/ sub-directory (i.e. we preserve the original /pdata/ sub-directory) as follows.

```plaintext
.../camomiletop/theatredata/2005-Sep-23-1342/
.../camomiletop/theatredata/2005-Sep-23-1342/fields/
.../camomiletop/theatredata/2005-Sep-23-1342/pdata/
.../camomiletop/theatredata/2005-Sep-23-1342/PDATA/
```

The suite of Perl programs making up this ‘stand-alone’ module is coordinated by the Perl program processdata.pl. All the programs and scripts required for processing and printing are stored in the
CHAPTER 18. PRINTING—THE STAND-ALONE (SA) MODULE

//.../camomiletop/datexsim/printfiles/ directory. The various programs are as follows.

- `processdata.pl` ... coordinates the module (in the 'operation' directory)
- `fields2PDATA.pl` ... main program in the `dir(PDATA)` dir
- `binlog2gnn.pl` ... converts .binlog files to .gnn files
- `binlog2data.pl` ... converts .binlog files to .data files
- `prtanes6.tex` ... TeX file for typesetting the graphs
- `prtdrug2.sty` ... TeX style option required by printdrug.tex
- `prtdrug.tex` ... TeX file for typesetting the drug page
- `base2texd.pl` ... ASCII to TeX conversion from keyboard entry log file

18.2 Running the `processdata.pl` script

To start the process we first need to move the Perl script `processdata.pl` into the appropriate operation directory (e.g., `/2005-Sep-23-1423/`); we then need to move to that directory and type the following at the commandline.

```bash
perl processdata.pl
```

In due course the script will be made to take the PATH of the operation directory as a parameter, in which case the user will type something like the following, from any location (or even within a script).

```bash
perl processdata.pl .../camomiletop/theatredata/2005-Sep-23-1342
```

The key steps performed by this module are as follows (the relevant program/script is shown in a box):

- Create a sub-directory called `/PDATA/` `processdata.pl`
- Move key files into the `/PDATA/` sub-directory `processdata.pl`
- Determine the start-time of data collection `fields2PDATA.pl`
- Convert the Unix-time in `.binlog` files → local-time in `.data` files `binlog2data.pl`
- Split up the `.data` files into 1-hr `.gnn` files `binlog2gnn.pl`
- Convert the `.gnn` files into GNUplot scripts for plotting `binlog2gnn.pl`
- Run `gnuplot` to generate the separate graphs in LATEX format
- Run LATEX to typeset the graphs and keyboard entry log `*.tex` as the anaesthetic record

We now address the printing process in some detail, covering the various steps from the raw `.binlog` files output by the Camomile data module to the production of the paper endpoint—the Anaesthetic Record—which is placed in the patient notes. The full code of the eight or so Perl programs is listed in the subsequent chapters.
[a1] Create the log file and make new directory

```perl
#!/usr/bin/perl
use Carp; # better error messages
use File::Copy; # for copying files
use Cwd; # for grabbing PATH of current working directory
use FindBin; # gets name of perl script and base dir

open (logfile, ">./processdata.log") || die "ERROR: can't open file <processdata.log>

# get progName and its base dir
$name1=$FindBin::Bin;
$programname=$FindBin::Script;
print (logfile "this LOG generated by program < ",$programname," > 

$timenow=localtime();
print (logfile $timenow,"\n");
print (logfile "Running program: ",$name1,"/", $programname,"\n");
$thisdir=cwd; # grab the PATH of current working dir
print (logfile $thisdir,"\n");
# create the /PDATA/ dir
mkdir '/PDATA',0744; # format = mkdir dir, mode (black book p 283)
```

[a2] Copy the required software tools to the /PDATA/ directory

We now copy a suite of files (required for data processing and printing) from the /datexsim/printfiles/ directory to the /PDATA/ directory. We use the secure copy command from the File::Copy module. Note that with this command we can only copy one file at a time. In the extract below, we copy the file fields2PDATA.pl.

```perl
#!/usr/bin/perl
...
# copy the required printTOOLS files from /camomiletop/datexsim/printfiles/ to .../PDATA/
$fromdir="../../datexsim/printfiles/";
$file1="fields2PDATA.pl";
if ($! eq ") {print (logfile "...["$file1,"]... file copied OK \n")
else {print (logfile "...["$file1,"] *** COPY ERROR: ", $!,"\n")
...
```

After copying all the files (currently six files) we then have everything in place for processing the data, so we now move to the /pdata/ directory in preparation for the next phase—data processing—and call the Perl coordinating program fields2PDATA.pl as follows.

```perl
#!/usr/bin/perl
...
$PDATAdir="PDATA";
chdir $PDATAdir;
```

[b] Data processing—launch program fields2PDATA.pl

The data processing is coordinated by the Perl script fields2PDATA.pl, so the next thing is (a) first check we are in the correct directory (/PDATA/), and if so, then to launch
the program (using the system() command), writing appropriate comments to the logfile as we go.

```perl
# [processdata.pl]
...  
## check we are in the correct directory
print (logfile "the current dir is: \n");
$thisdir=cwd; ## grab the current working dir
print (logfile $thisdir,"\n");
## now call fields2PDATA.pl
$perlprog="fields2PDATA.pl";
print (logfile "CALLing program <","$perlprog,">");
if (-e $perlprog) {print "\n CALLing program ", $perlprog,"\n";
    print (logfile "... OK...done\n");
    system("perl ./"."$perlprog")
} else{print "....ERROR: can't find file <$perlprog>\n";
    print (logfile " ** ERROR: can't find file <$perlprog>\n")};
```

### Determine the start-time

The first thing the fields2PDATA.pl script does is to determine the start-time by reading the time associated with the first data point in each of the .binlog files in the /fields/ directory, and selecting the earliest as defining the working start-time. Armed with a working start-time, we can then determine an ‘elapsed-time’ for each data-event. In practice these times are expressed as so-called Unix-time (seconds since 1st Jan 1970).

Each line of a typical .binlog file is a comma-separated data-pair, where the first item is the Unix time, and the second item is the parameter value. An example of a typical sat.binlog structure is as follows (sat.binlog).

```bash
## sat.binlog
1071580231,92
1071580236,92
1071580241,93
1071580246,93.5
1071580251,93
1071580256,93
1071580261,92.5
1071580266,92
...
...
```

The fields2PDATA.pl script starts by determining the earliest data entry time for each of the .binlog files, and then setting this earliest time as the $starttimeunix variable.

It does this by reading only the first Unix-time entry in each of the .binlog files (reading each filename from an array of all such filenames), and determining the earliest time. It also writes comments to the logfile so we can check its progress if we need to investigate any errors.

```bash
[[fields2PDATA.pl]]
```
## make an array of all required input filenames
## we are running this from the /PDATA/ dir
@fieldfilename = (
    "../fields/bp-d.binlog",
    "../fields/bp-s.binlog",
    "../fields/ecg-rr.binlog",
    "../fields/co2-exp.binlog",
    "../fields/co2-insp.binlog",
    "../fields/co2-rr.binlog",
    "../fields/cvp.binlog",
    "../fields/ecg-hr.binlog",
    "../fields/co2-exp.binlog",
    "../fields/co2-insp.binlog",
    "../fields/co2-rr.binlog",
    "../fields/mac-big.binlog",
    "../fields/mac-n2o.binlog",
    "../fields/mac-vap.binlog",
    "../fields/mv-exp.binlog",
    "../fields/n2o-exp.binlog",
    "../fields/n2o-insp.binlog",
    "../fields/n2o-exp.binlog",
    "../fields/nibp-d.binlog",
    "../fields/nibp-s.binlog",
    "../fields/o2-insp.binlog",
    "../fields/pplat.binlog",
    "../fields/sat.binlog",
    "../fields/sat-hr.binlog",
    "../fields/temp[0].binlog",
    "../fields/temp[1].binlog",
    "../fields/tv-exp.binlog",
    "../fields/tv-insp.binlog",
    "../fields/vap-code.binlog",
    "../fields/vap-exp.binlog",
    "../fields/vap-insp.binlog"
);

# get each .binlog file in turn, and read the first line for UNIX time
for ($j=0; $j<=$#fieldfilename; $j=$j+1 )
{
    $ifile = $fieldfilename[$j];
    if (-e $ifile) {
        open (fieldsfile, "<$ifile") || die "ERROR: can't open file $ifile\n"
    }
    else {print (printlog $ifile, " does NOT exist\n");
        next}
    print "...reading the fields file <bp-d.binlog> to access UNIX time\n";
    $n=0; ## line counter
    LINE: while (<fieldsfile>){
        next LINE if /^#/; # skip # comments
        next LINE if /^%/; # skip % comments
        next LINE if /^$/; # skip blank lines
        # grab the whole line as a string
        $dataline = $_;
        $n=$n+1; ## increment line counter
    }
chomp($dataline);  # removes the line-ending
## print the line to the log file
print (printlog $dataline,", filename = ", $ifile, "\n");
#---------------------
# print "the line is: $dataline\n";
# place the two params into an array
@value = split (/\[,\]/, $dataline);
## get no of items (should be only two items)
$nitems = $#value + 1;
print "no of items in the line = $nitems\n";
---------------------
$time = $value[0];
$parametervalue = $value[1];
## determine the least time (J = file counter)
if ($j==1){$starttimeunix=$time}
else {
    if ($time < $starttimeunix) {$starttimeunix = $time};
}
## only require the first UNIXtime from this file
if ($n==1){last}  #n is line counter
};  # end of line loop
};  # end of file loop
close (fieldsfile);
print (printlog "...finished reading all the .binlog files \n");

d | Decode the Unix start-time → local-time

The start-time (in Unix-time) is required later by the subroutine makegnnfiles() in the script binlog2gnn.pl in order to be able to split up the .data files created by the script binlog2data.pl into one-page data files (files containing data which will be typeset on a single page of the Anaesthetic Record)\(^1\)

We now decode the Unix start-time.

```
# [fields2PDATA.pl]
...
# $starttimeunix has been determined above
$starttimegmt = localtime($starttimeunix);
$originalgmt = $starttimegmt;  ## needed for printing header on anaes sheet (below)
print (printlog "starttimeunix =$starttimeunix\n");
print (printlog "starttimegmt = $starttimegmt\n");
print (printlog "----------------------------- \n");
```

\(^1\)Typically a page contains 1 hour of data (sampled at 45 second intervals), but it is useful to be able to devote single pages to a shorter period of time, in order to view the data in greater resolution—say, every 5 seconds, having only 6 minutes of data per page.
## note **** get /two/ spaces after the Month if days <10
## modified from SUB tedname() in launchcam12.pl
##-------------------------------------------
# if two spaces in posn 8 and 9 then remove one
if (substr($starttimegmt,7,2) eq " ") {substr($starttimegmt,7,2," ");
## replace spaces with commas
$starttimegmt =~ tr/ /,/
## make an array
@stgmt=split (/[,]/, $starttimegmt);
$day=$stgmt[0];
$month=$stgmt[1];
$date=$stgmt[2];
$st=$stgmt[3];
$year=$stgmt[4];
$noitems=$#stgmt+1;
print (printlog "....extracted starttimeUNIX [starttimeunix]\n");
print (printlog "....extracted starttimeGMT [starttimegmt]\n");
print (printlog "....extracted no. of gmt items = $ngmtitems ($corr)\n");
print (printlog "....extracted starttime hh:mm:ss [st]\n")
print "starttime=$starttimegmt\n";
print " no of gmt items = $noitems\n";
print "the gmt part is: $day,$month,$date,$st,$year\n";
#--------------------
#####? need to include some error checking ie abort if probem with the times
######## goto LASTLINE; ## abort program

### Running the script binlog2gnn.pl

We now (a) convert each .binlog file into a .data file (see below), and then (b) each of these is split into a series of 1-page .gnn files, e.g. g01, g02, ... etc., (each representing 1-hour periods), such that the data of each .gnn file is destined to be typeset on a single page of the Anaesthetic Record.

# [fields2PDATA.pl]
...
system ("perl binlog2gnn.pl $starttimeunix");

### Convert .binlog files to .data files

The program binlog2gnn.pl first rewrites each .binlog file into a more useful and informative .data files, each line of which will then also include two extra data items, namely (a) a local-time translation of the Unix-time, and (b) the elapsed-time since the start of data collection (the start-time).

The script binlog2gnn.pl CALLs the binlog2data.pl script to perform this particular task.

# [binlog2gnn.pl]
...
#!/usr/bin/perl
$starttimeunix = $ARGV[0]; ## used by the SUB Makegnnfiles()
open (timefile, ">timefile.dat")||die "ERROR: can't open file timefile.dat
";
##----------------
# make an array of all required paremater names used for printing anaes Record
@paramname = ("bp-s", "bp-d", "ecg-hr", "sat-hr", "cvp", "nibp-s", "nibp-d",
"sat", "o2-insp", "n2o-exp", "co2-exp",
"tv-exp", "co2-rr", "plat", "vap-insp", "vap-exp", "mac-big");
# get each parameter .binlog file in turn
for ($j=0; $j<=$#paramname; $j=$j+1 )
{
  $ifile = $paramname[$j]; ## NO .binlog file-extension here
  system ("perl binlog2data.pl $ifile") ;
...
}

A typical example of the sat.data file is as follows. Note that the elapsed-time parameter on the first line is zero, and that both the unix-time and the elapsed-times increase in steps of 5 seconds (data is output from the Datex monitor every 5 seconds).

# [sat.data]
1071580231, 2003:12:16:13:10:31, 0, 92.000000
1071580261, 2003:12:16:13:11:1, 30, 92.500000
...

Armed with the above .data file for a given parameter, then we proceed to generate from this a series of 1-page .gnn files (each typically of 1-hour duration), as described in the next section.

G Generate 1-page .gnn files with subroutine makegnnfiles()

This role of this subroutine is to generate from the new parameter .data file (which may contain many hours of data, since it contains all the data held in the original .binlog file) a series of 1-page .gnn files suitable for use by the GNUplot graphing program—each .gnn file generating a single page of the typeset Anaesthetic Record.

The makegnnfiles() subroutine is part of the Perl program binlog2gnn.pl (which is itself called by the co-ordinating Perl program fields2PDATA.pl). The subroutine is called with the field parameter name (for example, bp-d, or sat-hr) as follows.

makegnnfiles($paramname[$j]);

Calling the subroutine makegnnfiles() converts each of the parameter .data files into a series of 1-page duration two-column space-separated data-files suitable
for accessing by gnuplot. For example, a 4-hr sat.data file would typically be converted into four page-files (1-hour per page) as follows: sat.g01, sat.g02, sat.g03, sat.g04 (generally known as the .gnn files).

The `makegnnfiles()` subroutine also generated an elapsed time for each data-point within each page-file relative to the beginning of each page (typically, each hour) by using the new computed “start-time” for each page-file as the zero-time, i.e. the elapsed time within a 1-hour .gnn file will run from 0—3599 secs (i.e. just 1 hour per page in this case). We have three ⟨space⟩ delimited fields namely ⟨elapsed-time-(local)⟩, ⟨parameter⟩, ⟨unix-time⟩.

The subroutine works out how to split up the .data file into 1-page chunks (of 1-page time periods) by using the difference between the operation start-time and the unix-time on each line of data. Note that the Unix start-time was passed to the binlog2gnn.pl program by the calling program (fields2PDATA.pl). If the elapsed time exceeds the page-duration (the default is 1-hour), then the current .gnn file is closed, and the next one opened etc.

In practice, however, the default sampling-interval is 45 second intervals (this interval can be easily varied depending on the graph-plotting/typesetting requirements). So although the original .binlog data accumulates every 5 seconds (from the Datex AS/3 monitor), the actual printed data is typically thinned out somewhat, purely because there is a limit to the density of data which can usefully be printed on the Anaesthesia Record. If better resolution is required, then higher resolution printing can be performed at a later date, by making both the sampling-interval and the page-duration shorter, for example, we could plot all the data by making the sampling-interval (from the .data-file) → 0 seconds, and having a page-duration of 6 minutes—that is by plotting 72 data-points (at 5-second intervals) per 6-minute page.

```perl
# [binlog2gnn.pl]
...
sub makegnnfiles {  
  ## get the starttimeUNIX passed from commandline value --> @ARGV  
  ## the starttimeUNIX is obtained originally from file <starttime.dat>  
  $starttimeunix = $ARGV[0];  
  # passing only one name into array  
  my ($file) = @_;  
  print "---processing parameter [$file] \n";  
  # add the file-ending .dat  
  $infilename=$file.".data"; #***  
  print "---the input filename is [$infilename] \n";  
  open (infile, "<$infilename") || die "ERROR: can't find file $infilename \n";  
  # now make time-dependent out filename  
  # start with hour set to zero  
  $hour=0;  
  #----------  
  # start inputting lines of data  
  # need to get the time associated with line 1  
  #  
  $interval=45; #secs  
  $oldelapsedtime=0;  
  LINE: while (<$infilename>){
    next LINE if /"#/; #skip comments
```
next LINE if /^%/; #skip comments
next LINE if /^$/; #skip blank lines
# grab the whole line as a string
$dataline = $_;
# place the params into an array
@value=split (/[,]/, $dataline);
# print "$value[0] $value[1] $value[2]\n";
# assign the elapsedtime and param values
$unixtime=$value[0];
$gmtime=$value[1]; #GMT yyyy:mm:dd:hh::mm:ss
$elapsedtime = $value[2]; #elapsed-time (secs)
$paramvalue=$value[3];
chomp($paramvalue); # remove the line-ending to help maths
---------------------
# multiply the rr values by 50 (to make them fit range 0--1000)
if ($file eq "co2-rr"){$paramvalue=$paramvalue * 50};
----------------
## save data only every $interval (secs)
$elapsedtime=$unixtime-$starttimeunix; ## determine true elapsedtime
if ($elapsedtime < $oldelapsedtime +$interval)
{next LINE}
else{$oldelapsedtime = $elapsedtime}
#---------------
#now print data into 1 hr files
# make NewElapsed time relative to begining of new hour
# hour 1 = first real hour
# hour will be zero on first run thro algorithm so goes to else...
if ($elapsedtime <$hour *3600){
    $space=" ";
    # calculate new elapsed time from begining of new hour
    $newet=$elapsedtime-3600*($hour -1);
    print (outfile "$newet $space $paramvalue $space $unixtime\n");  
}
else{
    # close existing gnn file and open a new one (gnn+1)
    close (outfile);
    $hour=$hour + 1;
    #use two digits for the filename extension eg .g04
    if ($hour <10){$hour="0".$hour};
    $gnudatafilename=$file.".g".$hour;
    print "---the new output filename = $gnudatafilename \n";
    open (outfile,">$gnudatafilename")||die "can't open the outfile \n";
    # write some headers to the outfile
    $outfileheader1="## Camomile gnuplot datafilename = $gnudatafilename"
    $outfileheader2="# date?";
    print (outfile "$outfileheader1\n");
    print (outfile "$outfileheader2\n");
    # write info to the timefile
    print (timefile "$hour, $unixtime, $gmtime, $gnudatafilename\n");
$space=" ";
# calculate new elapsed time from beginning of new hour
$newet=$elapsedtime-3600*($hour-1);
print (outfile "$newet $space $paramvalue $space $unixtime\n");
}#end of else{
}#end of while
close (infile);
close (outfile);
}#$

A typical example of a .gnn file (the file sat.g03) is as follows. There are three fields (elapsed-time, parameter-value, unix-time) which are space-separated. In this example the data was collected every 30-40 seconds or so and the elapsed-times are seen to be 31, 76, 121, ... etc. The unix-time field is retained as a check. The .03 in the filename extension .g03 indicates that it represents data collected during the third hour.

##[sat.g03]

31 87.500000 1080559619
76 88.000000 1080559664
121 89.500000 1080559709
166 93.000000 1080559754
211 94.500000 1080559799
256 95.000000 1080559844
301 95.000000 1080559889
346 95.000000 1080559934
391 95.000000 1080559979
436 94.500000 1080560024

The log-file (timefile.txt)

Concurrently with the previous process, the program cam2gnnh.pl creates the timefile.dat file which holds the start-times for each of the .gnn files (see below). This file is very useful as a check on the functioning of the cam2gnnh.pl program.

##[timefile.txt]

... 01, 1071580301, 2003:12:16:13:11:41, bp-s.g01
02, 1071583865, 2003:12:16:14:11:5, bp-s.g02
03, 1071587465, 2003:12:16:15:11:5, bp-s.g03
...
... 01, 1071580276, 2003:12:16:13:11:16, sat.g01
02, 1071583840, 2003:12:16:14:10:40, sat.g02
03, 1071587440, 2003:12:16:15:10:40, sat.g03
...  ...
The base.log file (baselog.data)

After processing all the parameter fields → .gnn files we then access (extract) the anaesthetists log file (base.log) using the camomilefield2tex utility as before, only this time using the .l switch and the -s tex option since we are wanting to access a log file.

```bash
[[cam2gnnh.pl]]...
system("camomilefield2tex -p $projdir -l base -o baselog.data -s tex") ;
```

Note that since we are running this command from within the /pdata/ subdirectory then the default location for the output files is the current directory.

18.3 Write the GNUplot scripts for each graph

Each 1-hour page of the Anaesthesia Record consists of six separate graphs, each showing a time plot of several parameters. Each separate graph requires its own so-called .gnu file (script) which sets up the graph structure and plots each parameter inside it. All this is coordinated by the Perl program plotgnnk2.pl, and so we will look in more detail how this is done.

Each parameter to be plotted has its own .gnn2 parameter file (not absolutely necessary but very convenient in practice—see previous section). To facilitate this, we arrange that each 1-hour .gnn file has its elapsed time starting from zero, which greatly simplifies the plotting process.

The most difficult part of generating the .gnu files (one file per graph) is to construct the time-base, such that all .g01 parameter files are plotted on graphs showing the start and end times of the first hour, and also of the 15-minute vertical lines which are also drawn.

The timebase parameter $timeline

The time markings along the x-axis are drawn using the GNUplot set xtics() command which, in this case, takes a complicated parameter which is the string $timeline. In practice, for each hour the particular time-base used will be the same for all graphs drawn using parameters values from files having the same gnn value; say, .g02 files for example.

The following code determines this string for each hour, tailoring it to accommodate the time interval associated with each .gnn value, so as we move from one hour to the next then the time associated with each hour increases accordingly.

```bash
[[fields2PDATA.pl]]...
# determine the earliest start time from G01 files in timefile.dat file
# put the start-time-GMT[year:month:day:hrs:mins:sec] into an array
# then determine how many hours worth of Gnn files there are
# $st is the start-time hh:mm:ss from the <starttime.dat> file (see above)
$JJ=gnnmax("01"); ## returns gnnMax
print (printlog "start-time = [$st] 
");
```

\[^2\text{Not to be confused with the .gnn data files.}\]
print (printlog "GnnMax = $gnnmax \n");
# extract the separate hh, mm, ss values
@start_time= split (/[:]/, $st);
$starthour = $start_time[0];
$startminute=$start_time[1];
$startsecond=$start_time[2];
#----------
# now print all the graphs for all Gnn files from 01 to GnnMax
for ($gnn=1; $gnn<=$gnnmax; $gnn = $gnn+1)
{
  # first determine time in secs to the begining of next full hour
  $deltah = 3600 - ($startminute*60 + $startsecond);
  # generate correct start-hour depending on Gnn value
  $h = $starthour + $gnn;
  $hminus1=$h-1; $hplus1=$h+1;
  if ($h==0) {$hminus1=23};
  if ($h==23) {$hplus1=0};
  $q=900; $qq=1800; $qqq=2700; $qqqq=3600;
  # force 24hour clock
  if ($h <10){$h="0".$h};
  if ($hminus1 <10){$hminus1="0".$hminus1};
  if ($hplus1 <10){$hplus1="0".$hplus1};
  $deltahminusqqqq=$deltah-$qqqq;
  $deltahminusqqq=$deltah-$qqq;
  $deltahminusqq=$deltah-$qq;
  $deltahminusq=$deltah-$q;
  $deltahplusqqqq=$deltah+$qqqq;
  $deltahplusqqq=$deltah+$qqq;
  $deltahplusqq=$deltah+$qq;
  $deltahplusq=$deltah+$q;
  #----------
  $t1 = "$hminus1.00".$deltahminusqqqq;
  $t2 = "$hminus1.15".$deltahminusqqq;
  $t3 = "$hminus1.30".$deltahminusqq;
  $t4 = "$hminus1.45".$deltahminusq;
  $t5 = "$h.00".$deltah;
  $t6 = "$h.15".$deltahplusq;
  $t7 = "$h.30".$deltahplusqq;
  $t8 = "$h.45".$deltahplusqqq;
  $t9 = "$hplus1.00".$deltahplusqqqq;
  $timeline="$t1,$t2,$t3,$t4,$t5,$t6,$t7,$t8,$t9";

  Armed with the time-base we can start making (write to) the .gnu files. In the following we illustrate the code for writing the sat.gnu script file (which will be processed by the GNUplot program eventually). First we check that the 'hour' value incorporated into the .gnn string always has two digits (i.e. 4 → 04 and hence we obtain g04), and defining the graph height to be used, we then open the output file and proceed.

#[fields2PDATA.pl]
...
# first make sure the gnn string has three characters
if ($gnn <10){$gnn="0".$gnn};
# define the graph heights
$smallheight=0.43; ## for all other graphs
...
## now create the sat file -------------------------
open(satfile, ">plot-sat.gnu")
||die "ERROR: can't open plot-sat.gnu file\n";
print (satfile "#!/usr/bin/gnuplot\n");
print (satfile "# plot-sat.gnu script made by plotgnnk2.pl\n");
print (satfile "set terminal latex\n");
print (satfile "set output "plot-sat.pic" \n");
print (satfile "set size 1.40,$smallheight\n");
print (satfile "set xtics($timeline)\n");
print (satfile "set ytics ("" 80,","" 90,"" 100)\n");
print (satfile "set y2tics (80, 90, 100)\n");
print (satfile "set nokey\n");
print (satfile "set grid\n");
print (satfile "xmin=0;xmax=3600\n");
print (satfile "ymin=80; ymax=100\n");
print (satfile "plot [xmin:xmax][ymin:ymax] \n");
$satfilename="sat".".g".$gnn;
$fo2filename="o2-insp".".g".$gnn;

if (-e $satfilename)
 {print (satfile " "$satfilename" using 1:2 with linespoints 4 8,\n")
 else {print (printlog " ---**** no sat.gnn files\n")};

if (-e $fo2filename)
 {print (satfile " "$fo2filename" using 1:2 with linespoints 4 10,\n")
 else {print (printlog " ---**** no fo2.gnn files\n")};

$dummyline = " -20 with lines 1 # dummy line";
print (satfile "$dummyline \n");
close(satfile);

It is significant here that in the last few lines of this code we have used the line
print (bpfile "$dummyline \n");

This is to solve a problem which would arise should one or more of the parameter files
not exist, as in this situation GNUplot graph plotting would fail since it requires that the
final line must not have a comma at the end. By using a ‘dummy’ line (which has no
comma and only plots a point below the graph (-20) and hence is never visibly plotted)
as the final line, we are able to handle the failure of all or some of the parameter lines
which therefore can all have a terminal comma.
18.4 Run GNUplot on all the .gnu files

Once all the .gnu files have been written, then we run GNUplot on each one to generate each figure in \LaTeX\,\e\ picture format. Each printed sheet has five figures arranged horizontally from top to bottom. The legends are on the right hand side so they are not obscured by the binding when placed in the patient notes.

```bash
# [fields2PDATA.pl]
... print (printlog "---running GNUPLLOT on all the .gnu files
");
system ("gnuplot plot-bp.gnu");
system ("gnuplot plot-sat.gnu");
system ("gnuplot plot-fo2.gnu");
system ("gnuplot plot-co2.gnu");
system ("gnuplot plot-tv.gnu");
system ("gnuplot plot-vap.gnu");
print (printlog "...........GNUPLLOT ... done
");
```

18.5 Write the header line for the printouts

Each printed sheet has a header indicating the start-time (GMT and unix) and the .dvi filename (which indicates which hour the sheet refers to) as follows:

Record start-time: Thu Feb 12 12:11:19 2004 unix 1076587879 anes-04.dvi

This is written to a file (header.dat) as follows, and then read back when needed for printing.

```bash
# [fields2PDATA.pl]
... print "writing the <gnnheader.dat> file to contain header for Anes record

"; open (outfile5, ">gnnheader.dat")||die "ERROR: can’t create file <gnnheader.dat>

"; $timenow = localtime;
print (outfile5 "%% gnnheader.dat: created $timenow

");
print (outfile5 "%% file generated by <plotgnnk2.pl> RWD Nickalls

");
$fname="anes-".$gnn ".dvi";
print (outfile5 "\\header{$starttimeunix}{$originalgmt}{$fname}

");
close (outfile5);
print ".....<gnnheader.dat>.... done

");
```

18.6 Typeset the graphic pages using \LaTeX\,\e\textnormal{2}\textsuperscript{\textgreek{e}}

We now typeset the graph pages and create the output formats .dvi, .ps, and .pdf on the fly. The \TeX\,\e\ file for the graphs is prtanes6.tex. The style option is prtdrug2.sty. We create the PostScript files using dvips. We create the .pdf files using pdflatex.

```bash
print (printlog "---running LATEX on prtanes6.tex\n
");
system ("pslatex prtanes6.tex\n
");
$dvifilename="anes-".$gnn.".dvi";
```

18.7 Typeset the drug file using \LaTeX 2\epsilon

Processing the drug file (log file) is slightly more complicated owing to the fact that the typesetting is done using \LaTeX 2\epsilon. Consequently, since the anaesthetists can enter data using the keyboard we need to filter out all non-\TeX material (essentially to ‘escape’ certain ASCII characters; for example, we would modify % \rightarrow \% etc). This conversion is currently done by the Perl program base2texd.pl, which processes the original log-file (baselog.data) to the ‘filtered’ file baselognew.data.

We now typeset the ‘filtered’ drug-file and create the output formats .dvi, .ps, and .pdf on the fly as before. The \TeX file for the graphs is prtdrug.tex. The style option is prtdrug2.sty. We create the PostScript files using dvips. We create the .pdf files using pdflatex.

18.8 Printing the paper sheets

Finally, we print out all the sheets making up the Anaesthesia Record. This currently consists of one or more ‘drug’ sheets (the log file), together with a number of 1-hour graphic sheets presenting the measured parameters. These are usually printed out in the operating theatre and placed in the patient notes.

In practice a small Perl program (printall.pl) sends the final files to the printer in reverse order as follows.

#!/usr/bin/perl

# process the baselog.data file
system("perl ./base2texd.pl");
# now latex the prtdrug file
system("latex ./prtdrug.tex");
# copy the .dvi file to have a anes-drug.dvi filename
system("cp -v prtdrug.dvi anes-drug.dvi");
# make the PS version of the .dvi file
system("dvips anes-drug.dvi -o anes-drug.ps");
# make the pdf file
system("pdflatex prtdrug.tex");
# copy the .pdf file to have a anes-drug.pdf filename
system("cp -v prtdrug.pdf anes-drug.pdf");
# printALL.pl

# do graphs in reverse order
if (-e "anes-10.dvi") {system("dvips anes-10.dvi")} else{);
if (-e "anes-09.dvi") {system("dvips anes-09.dvi")} else{);
if (-e "anes-08.dvi") {system("dvips anes-08.dvi")} else{);
if (-e "anes-07.dvi") {system("dvips anes-07.dvi")} else{);
if (-e "anes-06.dvi") {system("dvips anes-06.dvi")} else{);
if (-e "anes-05.dvi") {system("dvips anes-05.dvi")} else{);
if (-e "anes-04.dvi") {system("dvips anes-04.dvi")} else{);
if (-e "anes-03.dvi") {system("dvips anes-03.dvi")} else{);
if (-e "anes-02.dvi") {system("dvips anes-02.dvi")} else{);
if (-e "anes-01.dvi") {system("dvips anes-01.dvi")} else{);
# print the drug sheet last (on top)
if (-e "anes-drug.dvi") {system("dvips anes-drug.dvi")} else {);
Chapter 19

Printing—the stand-alone (SA-06) module

19.1 Introduction

Although the automated ‘in-line’ printing module (described in chapter X) worked well in processing the data immediately at the end of an operation (by clicking on the ‘print last case’ button on the launcher widget), it was difficult to implement retrospectively—for example, when wanting to re-processing a different database of .binlog files (typically placed in the /fields/ subdirectory).

The /pdata/ sub-directory contains the original output of processed data. A typical directory structure of an operation database which, for example, started at 13:42 hrs on September 23, 2005, is as follows.

    .../camomiletop/theatredata/2005-Sep-23-1342/
    .../camomiletop/theatredata/2005-Sep-23-1342/fields/
    .../camomiletop/theatredata/2005-Sep-23-1342/pdata/

A new ‘stand-alone’ printing module was therefore developed, which (a) was simpler (i.e. did not use Simon Dales’ camomilefields2tex C-program, or need to read the starttime.dat file), and (b) could be pointed at a particular /fields/ subdirectory to generate the full printable anaesthesia record in the usual way. The output of all data processed by this SA module is stored in a separate /PDATA/ sub-directory (i.e. we preserve the original /pdata/ sub-directory) as follows.

    .../camomiletop/theatredata/2005-Sep-23-1342/
    .../camomiletop/theatredata/2005-Sep-23-1342/fields/
    .../camomiletop/theatredata/2005-Sep-23-1342/pdata/
    .../camomiletop/theatredata/2005-Sep-23-1342/PDATA/

The suite of Perl programs making up this ‘stand-alone’ module is coordinated by the Perl program processdata.pl. All the programs and scripts required for processing and printing are stored in the
processdata.pl ... coordinates the module (in the ‘operation’ directory)
fields2PDATA.pl ... main program in the \dir{PDATA} dir
binlog2gnn.pl ... converts .binlog files to .gnn files
binlog2data.pl ... converts .binlog files to .data files
prtanes6.tex ... TeX file for typesetting the graphs
prtdrug2.sty ... TeX style option required by prtdrug.tex
prtdrug.tex ... TeX file for typesetting the drug page
base2texd.pl ... ASCII to TeX conversion from keyboard entry log file

19.2 Running the processdata.pl script

To start the process we first need to move the Perl script processdata.pl into the appropriate operation directory (e.g., /2005-Sep-23-1423/); we then need to move to that directory and type the following at the commandline.

perl processdata.pl

In due course the script will be made to take the PATH of the operation directory as a parameter, in which case the user will type something like the following, from any location (or even within a script).

perl processdata.pl .../camomiletop/theatredata/2005-Sep-23-1342

The key steps performed by this module are as follows (the relevant program/script is shown in a box):

- Create a sub-directory called /PDATA/ processdata.pl
- Move key files into the /PDATA/ sub-directory processdata.pl
- Determine the start-time of data collection fields2PDATA.pl
- Convert the Unix-time in .binlog files → local-time in .data files binlog2data.pl
- Split up the .data files into 1-hr .gnn files binlog2gnn.pl
- Convert the .gnn files into GNUplot scripts for plotting binlog2gnn.pl
- Run gnuplot to generate the separate graphs in \LaTeX{} format
- Run \LaTeX{} to typeset the graphs and keyboard entry log *.tex as the anaesthetic record

We now address the printing process in some detail, covering the various steps from the raw .binlog files output by the Camomile data module to the production of the paper endpoint—the Anaesthetic Record—which is placed in the patient notes. The full code of the eight or so Perl programs is listed in the subsequent chapters.
Create the log file and make new directory

```perl
# [processdata.pl]
use Carp;       ## better error messages
use File::Copy; ## for copying files
use Cwd;        ## for grabbing PATH of current working directory
use FindBin;    ## gets name of perl script and base dir

open (logfile, "./processdata.log") || die "ERROR: can't open file <processdata.log>\n";

get progName and its base dir
$name1=$FindBin::Bin;
$programname=$FindBin::Script;
print (logfile "this LOG generated by program < ", $programname," > \n");
$time=localtime();
print (logfile $time,"\n");
print (logfile "Running program: ", $name1,"/", $programname,"\n");
$thisdir=cwd;       ## grab the PATH of current working dir
print (logfile "$thisdir","\n");

# create the /PDATA/ dir
mkdir 'PDATA', 0744; ## format = mkdir dir, mode (black book p 283)
```

Copy the required software tools to the /PDATA/ directory

We now copy a suite of files (required for data processing and printing) from the /datexsim/printfiles/ directory to the /PDATA/ directory. We use the secure copy command from the File::Copy module. Note that with this command we can only copy one file at a time. In the extract below, we copy the file fields2PDATA.pl.

```perl
# [processdata.pl]
...

# copy the required printTOOLS files from /camomiletop/datexsim/printfiles/ to .../PDATA/
$fromdir="/..../datexsim/printfiles/";

$file1="fields2PDATA.pl";

copy ($fromdir.$file1 , "/PDATA/"");

if ($! eq "") {print (logfile "...[","$file1","]... file copied OK \n");}
else {print (logfile "...[","$file1","] *** COPY ERROR: ", $!,"\n");}
...
```

After copying all the files (currently six files) we then have everything in place for processing the data, so we now move to the /pdata/ directory in preparation for the next phase—data processing—and call the Perl coordinating program fields2PDATA.pl as follows.

```perl
# [processdata.pl]
...

$PDATAdir="PDATA/";
chdir $PDATAdir;
```

Data processing—launch program fields2PDATA.pl

The data processing is coordinated by the Perl script fields2PDATA.pl, so the next thing is (a) first check we are in the correct directory (/PDATA/), and if so, then to launch
the program (using the \texttt{system()} command), writing appropriate comments to the logfile as we go.

\texttt{# [processdata.pl]}
\texttt{...}
\texttt{## check we are in the correct directory}
\texttt{print (logfile "the current dir is: \
");}
\texttt{$thisdir=cwd; \# \# grab the current working dir}
\texttt{print (logfile $thisdir,"\n");}
\texttt{## now call fields2PDATA.pl}
\texttt{$perlprog="fields2PDATA.pl";}
\texttt{print (logfile "CALLing program <","$perlprog,">";)}
\texttt{if (-e $perlprog) \{print \"\n CALLing program ", $perlprog,"\n";}
\texttt{ print (logfile \"... OK...done\n");}
\texttt{system("perl ./"."$perlprog")\}}
\texttt{else{print \"...ERROR: can't find file <$perlprog>\n";}
\texttt{ print (logfile \" ** ERROR: can't find file <$perlprog>\n\")\}}

\section{Determine the start-time}

The first thing the \texttt{fields2PDATA.pl} script does is to determine the start-time by reading the time associated with the first data point in each of the \texttt{.binlog} files in the /fields/ directory, and selecting the earliest as defining the working start-time. Armed with a working start-time, we can then determine an 'elapsed-time' for each data-event. In practice these times are expressed as so-called Unix-time (seconds since 1st Jan 1970).

Each line of a typical \texttt{.binlog} file is a comma-separated data-pair, where the first item is the Unix time, and the second item is the parameter value. An example of a typical \texttt{sat.binlog} structure is as follows (\texttt{sat.binlog}).

\texttt{## sat.binlog}
\texttt{1071580231,92}
\texttt{1071580236,92}
\texttt{1071580241,93}
\texttt{1071580246,93.5}
\texttt{1071580251,93}
\texttt{1071580256,93}
\texttt{1071580261,92.5}
\texttt{1071580266,92}
\texttt{...}
\texttt{...}

The \texttt{fields2PDATA.pl} script starts by determining the earliest data entry time for each of the \texttt{.binlog} files, and then setting this earliest time as the $\texttt{starttimeunix}$ variable.

It does this by reading only the first Unix-time entry in each of the \texttt{.binlog} files (reading each filename from an array of all such filenames), and determining the earliest time. It also writes comments to the logfile so we can check its progress if we need to investigate any errors.

\texttt{# [fields2PDATA.pl]}
## make an array of all required input filenames
## we are running this from the /PDATA/ dir
@fieldfilename = (  
    "../fields/bp-d.binlog",
    "../fields/bp-s.binlog",
    "../fields/ecg-rr.binlog",
    "../fields/co2-exp.binlog",
    "../fields/co2-imp.binlog",
    "../fields/co2-rr.binlog",
    "../fields/cvp.binlog",
    "../fields/ecg-hr.binlog",
    "../fields/ecg-rr.binlog",
    "../fields/mac-big.binlog",
    "../fields/mac-n2o.binlog",
    "../fields/mac-vap.binlog",
    "../fields/mv-exp.binlog",
    "../fields/n2o-exp.binlog",
    "../fields/nibp-d.binlog",
    "../fields/nibp-s.binlog",
    "../fields/o2-imp.binlog",
    "../fields/pplat.binlog",
    "../fields/sat.binlog",
    "../fields/sat-hr.binlog",
    "../fields/temp[0].binlog",
    "../fields/temp[1].binlog",
    "../fields/tv-exp.binlog",
    "../fields/tv-imp.binlog",
    "../fields/vap-code.binlog",
    "../fields/vap-exp.binlog",
    "../fields/vap-imp.binlog"
);

# get each .binlog file in turn, and read the first line for UNIX time
for ($j=0; $j<=$#fieldfilename; $j++ )
{
    $ifile = $fieldfilename[$j];
    if (-e $ifile) {
        open (fieldsfile, "<$ifile") || die "ERROR: can't open file $ifile\n"
    }
    else {print (printlog $ifile, " does NOT exist\n");
        next}
    print "...reading the fields file <bp-d.binlog> to access UNIX time\n";
    $n=0; # line counter
    while (<fieldsfile>){
        $n++;
        # grab the whole line as a string
        $dataline = ".;"
        $n=$n+1; # increment line counter
chomp($dataline); # removes the line-ending
## print the line to the log file
print (printlog $dataline,"", filename = ", $ifile, "\n");
#---------------------
# print "the line is: $dataline\n";
# place the two params into an array
@value=split (/[\,]/, $dataline);
## get no of items (should be only two items)
$nitems= $#value +1;
print "no of items in the line = $nitems\n";
#---------------------
$time=$value[0];
$parametervalue=$value[1];
## determine the least time (J = file counter)
if ($j==1){$starttimeunix=$time}
else {
    if ($time < $starttimeunix) {$starttimeunix = $time};
}
## only require the first UNIXtime from this file
if ($n==1)(last) #n is line counter
}; # end of line loop
}; # end of file loop
close (fieldsfile);
print (printlog "...finished reading all the .binlog files \n");

\[Decode the Unix start-time → local-time\]

The start-time (in Unix-time) is required later by the subroutine makegnnfiles() in the script binlog2gnn.pl in order to be able to split up the .data files created by the script binlog2data.pl into one-page data files (files containing data which will be typeset on a single page of the Anaesthetic Record).\[1\]

We now decode the Unix start-time.

# [fields2PDATA.pl]
...
# $starttimeunix has been determined above
$starttimegmt= localtime($starttimeunix);
$originalgmt=$starttimegmt; ## needed for printing header on anaes sheet (below)
print (printlog "starttimeunix =$starttimeunix\n");
print (printlog "starttimegmt = $starttimegmt\n");
print (printlog "----------------------------- \n");

## now put the starttimeGMT into an array
#---------------------------------------------
# note the main items are <space> separated except hh:mm:ss
# format is: Sun Jan 25 13:24:35 2004
# format is: Sun Jan  5 13:24:35 2004

\[Typically a page contains 1 hour of data (sampled at 45 second intervals), but it is useful to be able to devote single pages to a shorter period of time, in order to view the data in greater resolution—say, every 5 seconds, having only 6 minutes of data per page.\]
## note **** get /two/ spaces after the Month if days <10
## modified from SUB tedname() in launchcam12.pl
##-------------------------------------------
# if two spaces in posn 8 and 9 then remove one
if (substr($starttimegmt,7,2) eq " ") {substr($starttimegmt,7,2," ");}
##print " tr string = $startgmtstring\n";
## replace spaces with commas
$starttimegmt =~ tr/ /,;/
## make an array
@stgmt=split (/[,]/, $starttimegmt);
$day=$stgmt[0];
$month=$stgmt[1];
$date=$stgmt[2];
$st=$stgmt[3];
$year=$stgmt[4];
$noitems=$#stgmt+1;
print (printlog "....extracted starttimeUNIX [$starttimeunix]\n");
print (printlog "....extracted starttimeGMT [$starttimegmt]\n");
print (printlog "....extracted no. of gmt items = $noitems ($corr)\n");
print (printlog "....extracted gmt part is: $day,$month,$date,$st,$year\n");
print "starttime=$starttimegmt\n";
print " no of gmt items = $noitems\n";
print "the gmt part is: $day,$month,$date,$st,$year\n";
#-------------------
###? need to include some error checking ie abort if probem with the times
####
getic LASTLINE; ## abort program

### Running the script binlog2gnn.pl

We now (a) convert each .binlog file into a .data file (see below), and then (b) each of these is split into a series of 1-page .gnn files, e.g., g01, g02, ... etc., (each typically representing 1-hour periods), such that the data of each .gnn file is destined to be typeset on a single page of the Anaesthetic Record.

# [fields2PDATA.pl]
...  
system ("perl binlog2gnn.pl $starttimeunix");

### Convert .binlog files to .data files

The program binlog2gnn.pl first rewrites each .binlog file into a more useful and informative .data files, each line of which will then also include two extra data items, namely (a) a local-time translation of the Unix-time, and (b) the elapsed-time since the start of data collection (the start-time).

The script binlog2gnn.pl CALLs the binlog2data.pl script to perform this particular task.

# [binlog2gnn.pl]
...
#!/usr/bin/perl
$starttimeunix = $ARGV[0]; ## used by the SUB Makegnnfiles()
open (timefile, ">timefile.dat")||die "ERROR: can’t open file timefile.dat\n";

# make an array of all required paremater names used for printing anaes Record
@paramname = ("bp-s", "bp-d","ecg-hr","sat-hr","cvp","nibp-s","nibp-d",
"sat", "o2-insp", "n2o-exp", "co2-exp",
"tv-exp","co2-rr","pplat", "vap-insp", "vap-exp", "mac-big" );

#get each parameter .binlog file in turn
for ($j=0; $j<=$#paramname; $j=$j+1 )
{
    $ifile = $paramname[$j]; ## NO .binlog file-extension here
    system ("perl binlog2data.pl $ifile") ;
}

A typical example of the sat.data file is as follows. Note that the elapsed-time parameter on the first line is zero, and that both the unix-time and the elapsed-times increase in steps of 5 seconds (data is output from the Datex monitor every 5 seconds).

# [sat.data]
1071580231, 2003:12:16:13:10:31, 0, 92.000000
1071580261, 2003:12:16:13:11:01, 30, 92.500000

Armed with the above .data file for a given parameter, then we proceed to generate from this a series of 1-page .gnn files (each typically of 1-hour duration), as described in the next section.

Generate 1-page .gnn files with subroutine makegnnfiles()

This role of this subroutine is to generate from the new parameter .data file (which may contain many hours of data, since it contains all the data held in the original .binlog file) a series of 1-page .gnn files suitable for use by the GNUplot graphing program—each .gnn file generating a single page of the typeset Anaesthetic Record.

The makegnnfiles() subroutine is part of the Perl program binlog2gnn.pl (which is itself called by the co-ordinating Perl program fields2PDATA.pl). The subroutine is called with the field parameter name (for example, bp-d, or sat-hr) as follows.

makegnnfiles($paramname[$j]);

Calling the subroutine makegnnfiles() converts each of the parameter .data files into a series of 1-page duration two-column space-separated data-files suitable
for accessing by gnuplot. For example, a 4-hr sat.data file would typically be converted into four page-files (1-hour per page) as follows: sat.g01, sat.g02, sat.g03, sat.g04 (generally known as the .gnn files).

The `makegnnfiles()` subroutine also generated an elapsed time for each data-point within each page-file relative to the beginning of each page (typically, each hour) by using the new computed “start-time” for each page-file as the zero-time, i.e. the elapsed time within a 1-hour .gnn file will run from 0—3599 secs (i.e. just 1 hour per page in this case). We have three (space) delimited fields namely ⟨elapsed-time-(local)⟩, ⟨parameter⟩, ⟨unix-time⟩.

The subroutine works out how to split up the .data file into 1-page chunks (of 1-page time periods) by using the difference between the operation start-time and the unix-time on each line of data. Note that the Unix start-time was passed to the binlog2gnn.pl program by the calling program (fields2PDATA.pl). If the elapsed time exceeds the page-duration (the default is 1-hour), then the current .gnn file is closed, and the next one opened etc.

In practice, however, the default sampling-interval is 45 second intervals (this interval can be easily varied depending on the graph-plotting/typesetting requirements). So although the original .binlog data accumulates every 5 seconds (from the Datex AS/3 monitor), the actual printed data is typically thinned out somewhat, purely because there is a limit to the density of data which can usefully be printed on the Anaesthesia Record. If better resolution is required, then higher resolution printing can be performed at a later date, by making both the sampling-interval and the page-duration shorter, for example, we could plot all the data by making the sampling-interval (from the .data-file) → 0 seconds, and having a page-duration of 6 minutes—that is by plotting 72 data-points (at 5-second intervals) per 6-minute page.

```bash
# [binlog2gnn.pl]
...
sub makegnnfiles {
    ## get the starttimeUNIX passed from commandline value --> @ARGV
    ## the starttimeUNIX is obtained originally from file <starttime.dat>
    $starttimeunix = $ARGV[0];
    # passing only one name into array
    my ($file) = @{$ARGV};
    print "---processing parameter [$file] \n";
    # add the file-ending .dat
    $infilename=$file.".data"; ###*
    print "---the input filename is [$infilename] \n";
    open (infile, "<$infilename")||die "ERROR: can’t find file $infilename \n";
    # now make time-dependent out filename
    # start with hour set to zero
    $hour=0;
    #-------------------
    # start inputting lines of data
    # need to get the time associated with line 1
    #
    $interval=45; #secs
    $oldelapsedtime=0;
    LINE: while (<$infilename>){
        next LINE if /^#/; # skip comments
        ...
next LINE if /^%/; #skip comments
next LINE if /^$/; #skip blank lines
# grab the whole line as a string
$dataline = $_;
# place the params into an array
@value = split (/[,]/, $dataline);
# print " $value[0] $value[1] $value[2]\n";
# assign the elapsedtime and param values
$unixtime=$value[0];
$gmtime=$value[1]; #GMT yyyy:mm:dd:hh::mm:ss
$elapsedtime = $value[2]; #elapsed-time (secs)
$paramvalue=$value[3];
chomp($paramvalue); # remove the line-ending to help maths

#-------------------
# multiply the rr values by 50 (to make them fit range 0--1000)
if ($file eq "co2-rr"){$paramvalue=$paramvalue * 50};
#----------------
## save data only every $interval (secs)
$elapsedtime=$unixtime-$starttimeunix; ## determine true elapsedtime
if ($elapsedtime < $oldelapsedtime +$interval)
 {next LINE}
else{$oldelapsedtime = $elapsedtime}
#---------------
#now print data into 1 hr files
# make NewElapsed time relative to beginning of new hour
# hour 1 = first real hour
# hour will be zero on first run thro algorithm so goes to else...
if ($elapsedtime <$hour *3600)

{space=" ";
 # calculate new elapsed time from beginning of new hour
$newet = $elapsedtime-3600*($hour -1);
 print (outfile "$newet $space $paramvalue $space $unixtime\n");
}
else{
 # close existing gnn file and open a new one (gnn+1)
close (outfile);
$hour=$hour + 1;
#use two digits for the filename extension eg .g04
if ($hour <10){$hour="0".$hour};
$gnudatafilename=$file.".g".$hour;
print "---the new output filename = $gnudatafilename \n";
open (outfile,">$gnudatafilename")||die "can't open the outfile \n";
# write some headers to the outfile
$outfileheader1="# Camomile gnuplot datafilename = $gnudatafilename"
$outfileheader2="# date?\n";
print (outfile "$outfileheader1\n");
print (outfile "$outfileheader2\n");
# write info to the timefile
print (timefile "$hour, $unixtime, $gmtime, $gnudatafilename\n");
$space= " ";
# calculate new elapsed time from beginning of new hour
$newet =$elapsedtime - 3600 * ($hour - 1);
print (outfile "$newet $space $paramvalue $space $unixtime
");
}$end of else{
$}end o while
close (infile);
close (outfile);
}#$

A typical example of a .gnn file (the file sat.g03) is as follows. There are three fields (elapsed-time, parameter-value, unix-time) which are space-separated. In this example the data was collected every 30–40 seconds or so and the elapsed-times are seen to be 31, 76, 121, ... etc. The unix-time field is retained as a check. The 03 in the filename extension .g03 indicates that it represents data collected during the third hour.

## [sat.g03]
31 87.500000 1080559619
76 88.000000 1080559664
121 89.500000 1080559709
166 93.000000 1080559754
211 94.500000 1080559799
256 95.000000 1080559844
301 95.000000 1080559889
346 95.000000 1080559934
391 95.000000 1080559979
436 94.500000 1080560024
...
...

g The log-file (timefile.txt)

Concurrently with the previous process, the program cam2gnnh.pl creates the timefile.dat file which holds the start-times for each of the .gnn files (see below). This file is very useful as a check on the functioning of the cam2gnnh.pl program.

## [timefile.txt]
...
...
01, 1071580301, 2003:12:16:13:11:41, bp-s.g01
02, 1071583865, 2003:12:16:14:11:5, bp-s.g02
03, 1071587465, 2003:12:16:15:11:5, bp-s.g03
...
...
01, 1071580276, 2003:12:16:13:11:16, sat.g01
02, 1071583840, 2003:12:16:14:10:40, sat.g02
03, 1071587440, 2003:12:16:15:10:40, sat.g03
...
...
The base.log file (baselog.data)

After processing all the parameter fields → .gnn files we then access (extract) the anaesthetists log file (base.log) using the camomilefield2tex utility as before, only this time using the .1 switch and the -s tex option since we are wanting to access a log file.

```
# [cam2gnnh.pl]
...
system("camomilefield2tex -p $projdir -l base -o baselog.data -s tex") ;
```

Note that since we are running this command from within the /pdata/ subdirectory then the default location for the output files is the current directory.

### 19.3 Write the GNUplot scripts for each graph

Each 1-hour page of the Anaesthesia Record consists of six separate graphs, each showing a time plot of several parameters. Each separate graph requires its own so called .gnu file (script) which sets up the graph structure and plots each parameter inside it. All this is coordinated by the Perl program plotgnnk2.pl, and so we will look in more detail how this is done.

Each parameter to be plotted has its own .gnn2 parameter file (not absolutely necessary but very convenient in practice—see previous section). To facilitate this, we arrange that each 1-hour .gnn file has its elapsed time starting from zero, which greatly simplifies the plotting process.

The most difficult part of generating the .gnu files (one file per graph) is to construct the time-base, such that all .g01 parameter files are plotted on graphs showing the start and end times of the first hour, and also of the 15-minute vertical lines which are also drawn.

#### The timebase parameter $timeline

The time markings along the x-axis are drawn using the GNUplot set xtics() command which, in this case, takes a complicated parameter which is the string $timeline. In practice, for each hour the particular time-base used will be the same for all graphs drawn using parameters values from files having the same gnn value; say, .g02 files for example.

The following code determines this string for each hour, tailoring it to accommodate the time interval associated with each .gnn value, so as we move from one hour to the next then the time associated with each hour increases accordingly.

```
# [fields2PDATA.pl]
...
# determine the earliest start time from G01 files in timefile.dat file
# put the start-time-GMT[year:month:day:hrs:mins:sec] into an array
# then determine how many hours worth of Gnn files there are
# $st is the start-time hh:mm:ss from the <starttime.dat> file (see above)
$JJ=gnnmax("01"); ## returns gnnMax
print (printlog "start-time = 
")
```

2 Not to be confused with the .gnn data files.
print (printlog "GnnMax = $gnnmax \n");
# extract the separate hh, mm, ss values
@start_time= split (/[[:]]/, $st);
$starthour = $start_time[0];
$startminute=$start_time[1];
$starttime=$start_time[2];

###

## ? make an array to hold the starttimes of each gnn file
## these parameters are also used in binlog2GNN.pl to define the page size
## and sampling interval (from the .data files)
$pageseconds=440; ## = 88 x 5secs = no of seconds per typeset page
$interval=2; ## the sampling interval

###=====================================================================
# now print all the graphs for all Gnn files from 01 to GnnMax
for ($gnn=1; $gnn<=$gnnmax; $gnn = $gnn+1)
{
  print (printlog "=======================================================
  ");
  print (printlog "-----starting FOR/NEXT loop with Gnn = $gnn (gnnMax = $gnnmax)\n");
  ## the xtics() line is different for each Gnn

###=====================================================================
# now write the timeline (xtics) string for GNUplot
# work with unix time (seconds)
$gnnstartunix= $starttimeunix + ($gnn -1)*$pageseconds;
$gnnstarttime=colonformattime($gnnstartunix);
## make an array
@mytime($tyear, $tmonth, $tday, $thour, $tmin, $tsec)=split (/[[:]]/, $gnnstarttime);
@mytime=split (/[[:]]/, $gnnstarttime);
$thour=$mytime[3];
$tmin =$mytime[4];
$tsec =$mytime[5];

###=====================================================================
# note that the output from colonFormatedTime is hrs and mins are two digits
# so do not need to add extra zero if <10 etc initially, but only if later
# determine the timeSecs ($ts) of the minute lines
$hn=$tmin + 1; # add 1 as the first minute mark is the /next/ full minute
if ($m > 59) { $m = m%60; $h=$h + 1; if ($h>23){$h = $h%24};
  ## force leading zero of <10
  $m= substr("00".$m, -2); $h= substr("00".$h, -2); $ts=60-$tsec;
  $t1=qq("$h:$m"). $ts; 
  ## GNUplot xtics format = ,timestring<space>x-value(secs),
  $m=$m+1;
  if ($m > 59) { $m = m%60; $h=$h + 1; if ($h>23){$h = $h%24}; }
  $m= substr("00".$m, -2); $h= substr("00".$h, -2);
Armed with the time-base we can start making (write to) the .gnu files. In the following we illustrate the code for writing the sat.gnu script file (which will be processed by the GNUplot program eventually). First we check that the 'hour' value incorporated into the .ggn string always has two digits (i.e. 4 → 04 and hence we obtain g04), and defining the graph height to be used, we then open the output file and proceed.

# [fields2PDATA.pl] ...
# first make sure the gnn string has three characters
if ($gnn <10){$gnn="0".$gnn};
# define the graph heights
$smallheight=0.43; ## for all other graphs
...
## now create the sat file -------------------------
open(satfile, ">plot-sat.gnu")
||die "ERROR: can't open plot-sat.gnu file
"
print (satfile "#!/usr/bin/gnuplot"
);
print (satfile "# plot-sat.gnu script made by plotgnnk2.pl\n"
);
print (satfile "set terminal latex\n"
);
print (satfile "set output "plot-sat.pic" \n"
);
print (satfile "set size 1.40,$smallheight\n"
);
print (satfile "set xtics($timeline)\n"
);
print (satfile "set ytics ("80," 90," 100)\n"
);
print (satfile "set y2tics (80, 90, 100)\n"
);
print (satfile "set nokey\n"
);
print (satfile "set grid\n"
);
print (satfile "xmin=0;xmax=3600\n"
);
print (satfile "ymin=80; ymax=100\n"
);
print (satfile "plot \[xmin:xmax\][ymin:ymax] \\
");
$satfilename="sat".".g".$gnn;
$fo2filename="o2-insp".".g".$gnn;
if (-e $satfilename)
{print (satfile " "$satfilename\" using 1:2 with linespoints 4 8,\\"
")
else {print (printlog " -----**** no sat.gnn files\n")};
if (-e $fo2filename)
{print (satfile " "$fo2filename\" using 1:2 with linespoints 4 10,\\"
")
else {print (printlog " -----**** no fo2.gnn files\n")};
$dummyline = " -20 with lines 1 # dummy line"
print (satfile "$dummyline \n"
);
close (satfile);

It is significant here that in the last few lines of this code we have used the line
print (bpfile "$dummyline \n"
);

This is to solve a problem which would arise should one or more of the parameter files
not exist, as in this situation GNUplot graph plotting would fail since it requires that the
final line must not have a comma at the end. By using a ‘dummy’ line (which has no
comma and only plots a point below the graph (-20) and hence is never visibly plotted)
as the final line, we are able to handle the failure of all or some of the parameter lines
which therefore can all have a terminal comma.
19.4 Run GNUplot on all the .gnu files

Once all the .gnu files have been written, then we run GNUplot on each one to generate each figure in \TeX picture format. Each printed sheet has five figures arranged horizontally from top to bottom. The legends are on the right hand side so they are not obscured by the binding when placed in the patient notes.

```plaintext
# [fields2PDATA.pl]
...
print (printlog "---running GNUPLPLOT on all the .gnu files\n");
system ("gnuplot plot-bp.gnu");
system ("gnuplot plot-sat.gnu");
system ("gnuplot plot-fo2.gnu");
system ("gnuplot plot-co2.gnu");
system ("gnuplot plot-tv.gnu");
system ("gnuplot plot-vap.gnu");
print (printlog "...........GNUPLPLOT ... done\n");
```

19.5 Write the header line for the printouts

Each printed sheet has a header indicating the start-time (GMT and unix) and the .dvi filename (which indicates which hour the sheet refers to) as follows:

Record start-time: Thu Feb 12 12:11:19 2004 unix 1076587879 anes-04.dvi

This is written to a file (header.dat) as follows, and then read back when needed for printing.

```plaintext
# [fields2PDATA.pl]
...
print "writing the <gnnheader.dat> file to contain header for Anes record \n";
open (outfile5, ">gnnheader.dat")||die "ERROR: can’t create file <gnnheader.dat>\n";
$timenow = localtime;
print (outfile5 "%% gnnheader.dat: created $timenow\n");
print (outfile5 "%% file generated by <plotgnnk2.pl> RWD Nickalls\n");
$fname="anes-"."$gnn.".dvi";
print (outfile5 "\\header{$starttimeunix}{$originalgmt}{$fname}\n");
close (outfile5);
print "......<gnnheader.dat>.... done\n");
```

19.6 Typeset the graphic pages using \LaTeX\text{2e}

We now typeset the graph pages and create the output formats .dvi, .ps, and .pdf on the fly. The \TeX file for the graphs is prtanes6.tex. The style option is prtdrug2.sty. We create the PostScript files using \texttt{dvips}. We create the .pdf files using \texttt{pdflatex}.

```plaintext
print (printlog "---running LATEX on prtanes6.tex\n");
system ("pslatex prtanes6.tex\n");
$dvifilename="anes-"."$gnn.".dvi";
```
19.7 Typeset the drug file using \LaTeX\ 2ε

Processing the drug file (log file) is slightly more complicated owing to the fact that the typesetting is done using \LaTeX\ 2ε. Consequently, since the anaesthetists can enter data using the keyboard we need to filter out all non-\LaTeX\ material (essentially to ‘escape’ certain ASCII characters; for example, we would modify \texttt{\%rightarrow} \texttt{\\%} etc). This conversion is currently done by the Perl program \texttt{base2texd.pl}, which processes the original log-file (\texttt{baselog.data}) to the ‘filtered’ file \texttt{baselognew.data}.

We now typeset the ‘filtered’ drug-file and create the output formats .dvi, .ps, and .pdf on the fly as before. The \LaTeX\ file for the graphs is \texttt{prtdrug.tex}. The style option is \texttt{prtdrug2.sty}. We create the PostScript files using \texttt{dvips}. We create the .pdf files using \texttt{pdflatex}.

```
#!/usr/bin/perl

# process the baselog.data file
system("perl ./base2texd.pl");
# now latex the prtdrug file
system("latex ./prtdrug.tex");
# copy the .dvi file to have a anes-drug.dvi filename
system("cp -v prtdrug.dvi anes-drug.dvi");
# make the PS version of the .dvi file
system("dvips anes-drug.dvi -o anes-drug.ps");
# make the pdf file
system("pdflatex prtdrug.tex");
# copy the .pdf file to have a gnn.pdf filename
system("cp -v prtdrug.pdf anes-drug.pdf");
```

```
# copy the .dvi file to have a gnn.dvi filename
system("cp -v prtnanes6.dvi $dvifilename");
# make the .ps files
$psfilename="anes-"."$gnn.".ps";
system("dvips $dvifilename -o $psfilename");
print (printlog "...........LATEX ...done\n");
# now make the pdf files
system("pdflatex prtanes6.tex");
$pdffilename="anes-"."$gnn.".pdf";
# copy the .pdf file to include a ..gnn.pdf filename
system("cp -v prtanes6.pdf $pdffilename");
```

19.8 Printing the paper sheets

Finally, we print out all the sheets making up the Anaesthesia Record. This currently consists of one or more ‘drug’ sheets (the log file), together with a number of 1-hour graphic sheets presenting the measured parameters. These are usually printed out in the operating theatre and placed in the patient notes.

In practice a small Perl program (printall.pl) sends the final files to the printer in reverse order as follows.

```bash
#!/usr/bin/perl
```
# printALL.pl
# do graphs in reverse order
if (-e "anes-10.dvi") {system("dvips anes-10.dvi")} else{);
if (-e "anes-09.dvi") {system("dvips anes-09.dvi")} else{);
if (-e "anes-08.dvi") {system("dvips anes-08.dvi")} else{);
if (-e "anes-07.dvi") {system("dvips anes-07.dvi")} else{);
if (-e "anes-06.dvi") {system("dvips anes-06.dvi")} else{);
if (-e "anes-05.dvi") {system("dvips anes-05.dvi")} else{);
if (-e "anes-04.dvi") {system("dvips anes-04.dvi")} else{);
if (-e "anes-03.dvi") {system("dvips anes-03.dvi")} else{);
if (-e "anes-02.dvi") {system("dvips anes-02.dvi")} else{);
if (-e "anes-01.dvi") {system("dvips anes-01.dvi")} else{);
# print the drug sheet last (on top)
if (-e "anes-drug.dvi") {system("dvips anes-drug.dvi")} else {};
Chapter 20

processdata.pl


#!/usr/bin/perl -w
## processdata.pl
## RWD Nickalls Oct 30, 2005
#-----------------------------------
use Carp; ## better error messages
use File::Copy; ## for copying files
use Cwd; ## for grabbing current directory name
use FindBin; ## gets name of perl program

## processdata.pl
## RWD Nickalls
##
## a module for coordinating the processing of all fields data to /PDATA/
## and which DOES /NOT/ USE Simon Dales' camomilefiles2tex program.
## this module runs from the time-encoded dir itself.
## and processes all the Field files to final anes charts without needing to
## use the <starttime.dat. file (since the prog <fields2PDATA.pl> reads all
## the binlog files to determine the earliest start time).
## This program creates the /PCOPY/ subdir, copies across the necessary printfiles,
## and then CALLS the program (fields2PDATA.pl)
#-----------WARNING---------------------
## (1) remember to change the path of the /printfiles/ when using in theatre
## (2) need to delete part which copies this prog back to /printfiles/ etc
#---------------------------------------
## processdata.pl (from printlast.pl)
## October 16, 2005
## to process all the data - as a standalone file
##-------------------
## 1) read the starttime.dat if it exists, else read all the fields files to
## get earliest UNIX time

    open (logfile, "./processdata.log") || die "ERROR: can't open file <processdata.log>\n";

    $line="------------------------------------------"

## get progName and its base dir

    $name1=$FindBin::Bin;
    $programname=$FindBin::Script;
    print (logfile "this LOG generated by program < ",$programname," > \n");
    $timenow=localtime();
    print (logfile $timenow,"\n");
    print (logfile "Running program: ",$name1,"/", $programname,"\n");
    print (logfile $line,"\n");

##--------get this starting directory----------

    print (logfile "the current (starting) dir is: \n");
    system("pwd");
    $thisdir=cwd;
    print (logfile $thisdir,"\n");

## create the /PDATA/ dir

    print (logfile "creating ./PDATA directory\n");
    mkdir 'PDATA',0744; ## format = mkdir dir, mode (black book p 283)
    ## now check the dir

    print (logfile $line,"\n");

## copy printTOOLS files to /PDATA/

    print (logfile "copying all required printfiles from /datexsim/printfiles/ to /PDATA/ \n");
    $fromdir="../..//datexsim/printfiles/"
    $file1="fields2PDATA.pl";
copy ($fromdir.$file1, "./PDATA");
if ($! eq "") {print (logfile "...[",$file1,"]... file copied OK \n")
else {print (logfile "...[",$file1,"] *** COPY ERROR: ", $!,"\n")}

$file2="binlog2gnn.pl"; #(uses Dick's binlog2data.pl)
copy ($fromdir.$file2, "./PDATA");
if ($! eq "") {print (logfile "...[",$file2,"]... file copied OK \n")
else {print (logfile "...[",$file2,"] *** COPY ERROR: ", $!,"\n")}

$file21="binlog2data.pl"; ## CALLED by cam2gnnH2
copy ($fromdir.$file21, "./PDATA");
if ($! eq "") {print (logfile "...[",$file21,"]... file copied OK \n")
else {print (logfile "...[",$file21,"] *** COPY ERROR: ", $!,"\n")}

$file3="prtanes6.tex";
copy ($fromdir.$file3, "./PDATA");
if ($! eq "") {print (logfile "...[",$file3,"]... file copied OK \n")
else {print (logfile "...[",$file3,"] *** COPY ERROR: ", $!,"\n")}

$file4="prtdrug.tex";
copy ($fromdir.$file4, "./PDATA");
if ($! eq "") {print (logfile "...[",$file4,"]... file copied OK \n")
else {print (logfile "...[",$file4,"] *** COPY ERROR: ", $!,"\n")}

$file5="prtdrug2.sty";
copy ($fromdir.$file5, "./PDATA");
if ($! eq "") {print (logfile "...[",$file5,"]... file copied OK \n")
else {print (logfile "...[",$file5,"] *** COPY ERROR: ", $!,"\n")}

$file6="base2texd.pl";
## converts base.log/baselog.data --> something which TeX can print
copy($fromdir.$file6, "./PDATA");
if ($! eq "") {print (logfile "...[",$file6,"]... file copied OK \n")
else {print (logfile "...[",$file6,"] *** COPY ERROR: ", $!,"\n")}

print (logfile $line,\n"\n");
##============================================================================
## move to the required dir
print (logfile "changing DIR to /PDATA/ dir\n");
$PDATAdir="PDATA";
chdir $PDATAdir;
##note that  chdir is a PERL command (but cd is a Linux BASH command)
## now check we are in the correct directory
print (logfile "the current dir is: \n");
system("pwd"); ## writes to screen
$thisdir=cwd;
print (logfile $thisdir,\n"\n");
#----------------------------------
print (logfile $line,"\n"); ##==================================
# now we can start crunching the Field files
# now call fields2PDATA.pl
$perlprog="fields2PDATA.pl";
print (logfile "CALLing program <",$perlprog,">");
if (-e $perlprog) {print "\n CALLing program ", $perlprog,"\n";
    print (logfile "... OK...done\n");
    system("perl ./"."$perlprog")}
elself{print "...ERROR: can't find file <$perlprog>\n";
    print (logfile " ** ERROR: can't find file <$perlprog>\n")};
print (logfile $line,"\n"); ##==================================
# return to orig directory
print "...returning to original directory\n";
print (logfile "returning to original DIR\n");
chdir ".";
# check the dir
print (logfile "the current dir is: \n");
system("pwd"); ## writes to screen
$thisdir=cwd;
print (logfile $thisdir,"\n");
print (logfile $line,"\n"); ##==================================
close (logfile);
__END__
Chapter 21

fields2PDATA.pl

December 13, 2014 /allfiles/book-xenon/ch-fields2PDATA.tex

#!/usr/bin/perl
## fields2PDATA.pl
## -w ## turned off for the moment
#------------------
# /camomiletop/datexsim/printfiles/fields2PDATA.pl (orig from plotgnnK2.pl)
# for gnuplot graphs with right-side y2labels
# prog for plotting Gnn files from .binlog files/cam2
# Dick Nickalls
# October 16,2005

#============================= rember
## remember to use latest version of files:
## cam2gnnH.pl
## plotgnnK2.pl
## prtanes6.tex
## base2tex.pl
## prt.drug2.sty
## prtdrug.tex
##======new changes=======================
## Feb 25 2004
## plot pplateau pressure = pplat.binlog
## also plot rr on fo2 graph as well to catch rr >20
#========================================
# this prog is run from within the /projdir/PDATA/ dir
#========================================
# create a printer-log file
  open(printlog, ">printlog.txt")||die "ERROR: can’t open printlog.txt file\n";
##

286
$gmt = localtime();
print (printlog "printlog.txt, ", $gmt, "\n");
print (printlog "log of the printing module [fields2PDATA.pl]\n");
print (printlog "...this program is CALLED by < processdata.pl >\n");
print (printlog "-------------start of [perl fields2PDATA.pl]-------------\n");

##===============determine the UNIXstarttime from binlog files=================

## make an array of all required input filenames
## we are running this from the /PDATA/ dir

@fieldfilename = (
    "../fields/bp-d.binlog",
    "../fields/bp-s.binlog",
    "../fields/ecg-rr.binlog",
    "../fields/co2-exp.binlog",
    "../fields/co2-insp.binlog",
    "../fields/co2-rr.binlog",
    "../fields/cvp.binlog",
    "../fields/ecg-hr.binlog",
    "../fields/ecg-rr.binlog",
    "../fields/mac-big.binlog",
    "../fields/mac-n2o.binlog",
    "../fields/mac-vap.binlog",
    "../fields/mv-exp.binlog",
    "../fields/n2o-exp.binlog",
    "../fields/n2o-exp.binlog",
    "../fields/n2o-exp.binlog",
    "../fields/n2o-exp.binlog",
    "../fields/nibp-d.binlog",
    "../fields/nibp-s.binlog",
    "../fields/o2-insp.binlog",
    "../fields/pplat.binlog",
    "../fields/sat.binlog",
    "../fields/sat-hr.binlog",
    "../fields/temp[0].binlog",
    "../fields/temp[1].binlog",
    "../fields/tv-exp.binlog",
    "../fields/tv-insp.binlog",
    "../fields/vap-code.binlog",
    "../fields/vap-exp.binlog",
    "../fields/vap-exp.binlog"
);

# get each .binlog file in turn, and read the first line for UNIXtime
for ($j=0; $j<=$#fieldfilename; $j=$j+1 )
{
    $ifile = $fieldfilename[$j];
    if (-e $ifile) {
        open (fieldsfile, "<$ifile")||die "ERROR: can't open file $ifile\n";
    }
    else {print (printlog $ifile, " does NOT exist\n";
        next}
print "...reading the fields file <bp-d.binlog> to access UNIX time\n";
$n=0;  # counter

LINE: while (<fieldsfile>){
    next LINE if /^#/;  # skip # comments
    next LINE if /^%/;  # skip % comments
    next LINE if /^$/;  # skip blank lines
    # grab the whole line as a string
    $dataline = $_;
    $n=$n+1;  # increment counter
    chomp($dataline);  # removes the line-ending
    ## print the line to the log file
    print (printlog $dataline,", filename = ", $ifile, "\n");
    #print "the line is: $dataline\n";
    # place the two params into an array
    @value = split (/\[/,, $dataline);
    ## get no of items (should be only two items)
    $nitems= $#value +1;
    print "no of items in the line = $nitems\n";
    #---------------------
    $time=$value[0];
    $parametervalue=$value[1];
    ## determine the least time (J = file counter)
    if ($j==1){$starttimeunix=$time}
    else {
        if ($time < $starttimeunix) {$starttimeunix = $time};
    }
    ## only require the first UNIXtime from this file
    if ($n==1){last}  # n is line counter
};  # end of line loop
};  #end of file loop
close (fieldsfile);

print (printlog "...finished reading all the .binlog files \n");

#---------------------
#$starttimeunix  has been determined above
$starttimegmt= localtime($starttimeunix);
$originalgmt=$starttimegmt;  # needed for printing header on anaes sheet (below)
print (printlog "starttimeunix =$starttimeunix\n");
print (printlog "starttimegmt = $starttimegmt\n");
print (printlog "--------------------------- \n");
#---------------------

## now put the starttimeGMT into an array

#** note the main items are <space> separated except hh:mm:ss
#** format is:  Sun Jan 25 13:24:35 2004
#** format is:  Sun Jan 5 13:24:35 2004
## note **** get /two/ spaces after the Month if days <10
## see SUB tedname() in launchcam12.pl
# if two spaces in posn 8 and 9 then remove one
if (substr($starttimegmt,7,2) eq " ") {substr($starttimegmt,7,2," ");

## replace spaces with commas
$starttimegmt = " tr/ /,;"

## make an array
@stgmt=split (/\[/, $starttimegmt);
$day=$stgmt[0];
$month=$stgmt[1];
$date=$stgmt[2];
$st=$stgmt[3];
$year=$stgmt[4];
$noitems=$#stgmt+1;

print (printlog........extracted starttimeUNIX [starttimeunix]\n"");
print (printlog........extracted starttimeGMT [starttimegmt]\n"");
print (printlog........extracted no. of gmt items = $ngmtitems ($corr)\n"");
print (printlog........extracted gmt part is: $day,$month,$date,$st,$year\n"");
print (printlog........extracted starttime hh:mm:ss [$st]\n"");
print "starttime=$starttimegmt"
print "no of gmt items = $ngmtitems"
print "the gmt part is: $day,$month,$date,$st,$year"

#------------------
#####? need to include some error checking ie abort if problem with the times

### went to LASTLINE; # abort program

#------------------
# now run cam2gmnH.pl to process all the X.binlog files --> X.data files
print (printlog "running command [perl cam2gmnH.pl $starttimeunix $projdir]\n"");

### system ("perl cam2gmnH.pl $starttimeunix $projdir");
system ("perl binlog2gmn.pl $starttimeunix");
print (printlog .......OK\n"");
print (printlog "=======================================================\n"");

#------------------
# PLOTTING/PRINTING

## determine the earliest start time from GO1 files in timefile.dat file
# put the start-time-GMT[year:month:day:hrs:mins:sec] into an array
$JJ=gnnmax("01"); # returns gnnMax
print (printlog "=======================================================\n"");
print (printlog "start-time = [$st] \n"");
print (printlog "GnnMax = $gnnmax \n"");

# $st is the start-time hh:mm:ss from the <starttime.dat> file (see above)
```perl
## extract the separate hh, mm, ss values
@start_time = split (/[:]/, $st);
$starthour = $start_time[0];
$startminute = $start_time[1];
$startsecond = $start_time[2];
print (printlog "graphs: extracted start hour/min/sec are [$starthour, $startminute, $startsecond]
");

# now print all the graphs for all Gnn files from 01 to GnnMax
for ($gnn=1; $gnn<=$gnnmax; $gnn = $gnn+1)
{
    print (printlog "=======================================================
");
    print (printlog "-----starting FOR/NEXT loop with Gnn = $gnn (gnnMax = $gnnmax)\n");
    # the xtics() line is different for each Gnn
    # print "$starthour,$startminute, $startsecond \n";
    # determine time in secs to the begining of next full hour
    $deltah = 3600 - ($startminute*60 + $startsecond);
    print (printlog "deltah = $deltah\n");
    # generate correct start-hour depending on Gnn value
    $h = $starthour + $gnn;
    $hminus1 = $h-1; $hplus1 = $h+1;
    if ($h==0) {$hminus1=23};
    if ($h==23) {$hplus1=0};
    $q=900; $qq=1800; $qqq=2700; $qqqq=3600;
    # force 24hour clock
    if ($h <10){$h="0".$h};
    if ($hminus1 <10){$hminus1="0".$hminus1};
    if ($hplus1 <10){$hplus1="0".$hplus1};
    $deltahminusqqqq=$deltah-$qqqq;
    $deltahminusqqq=$deltah-$qqq;
    $deltahminusqq=$deltah-$qq;
    $deltahminusq=$deltah-$q;
    $deltahplusqqqq=$deltah+$qqqq;
    $deltahplusqqq=$deltah+$qqq;
    $deltahplusqq=$deltah+$qq;
    $deltahplusq=$deltah+$q;
    $timeline="$t1,$t2,$t3,$t4,$t5,$t6,$t7,$t8,$t9";
    print (printlog "set xtics($timeline)\n");
    print (printlog "---starting to write all the .gnn files\n");
```

## first make sure the gnn string has three characters
if ($gnn < 10) { $gnn = "0" . $gnn; }

## define the graph heights
$bigheight = 0.9; ## for bp graph
$smallheight = 0.43; ## for all other graphs

## now create the BP file
open(bpfile, ">plot-bp.gnu") || die "ERROR: can't open plot-bp.gnu file\n";
print (bpfile "#!/usr/bin/gnuplot\n");
print (bpfile "## plot-bp.gnu script made by plotgnnk2.pl\n");
print (bpfile "set terminal latex\n");
print (bpfile "set output "plot-bp.pic\n"); set noxtics
 OF LA TEX

set size 1.40, $bigheight

FIG\&$2\$ \$\bullet\$ 1\n

print (bpfile "set y2tics (0, 20, 50, 100, 150, 200)\n");
print (bpfile "set y2label.......\n");
print (satfile "set y2label 'Sat $\circ$ FIO$_2$, $\bullet$ ' 1\n");
print (bpfile "set nokey\n");
print (bpfile "set grid\n");
print (bpfile "xmin=0; xmax=3600\n");
print (bpfile "ymin=0; ymax=200\n");
print (bpfile "plot [xmin:xmax][ymin:ymax] \n");
print (bpfile " 20 with lines 1,\n");
print (bpfile " 50 with lines 1,\n");
print (bpfile " 100 with lines 1,\n");
print (bpfile " 150 with lines 1,\n");

$bpsfilename="bp-s".".g".$gnn;
$bdfilename="bp-d".".g".$gnn;

$nibpsfilename="nibp-s".".g".$gnn;
$nibdfilename="nibp-d".".g".$gnn;

$hrrecfilename="ecg-hr".".g".$gnn;
$hrroxfilename="sat-hr".".g".$gnn;
$cdfilename="cvp".".g".$gnn;

if (-e $bpsfilename)
    {print (bpfile " "$bpsfilename" using 1:2 with linespoints 1 9,\n"});
else {print (printlog " ---- no bp-s.gnn files\n")};

if (-e $bpdfilename)
    {print (bpfile " "$bpdfilename" using 1:2 with linespoints 1 8,\n"});
else {print (printlog " ---- no bp-d.gnn files\n")};

if (-e $nibpsfilename)
    {print (bpfile " "$nibpsfilename" using 1:2 with linespoints 1 3,\n"});
else {print (printlog "----- no nibp-s.gnn files\n");}

if (-e $nibpdfilename)
{print (bpfile " \$nibpdfilename\" using 1:2 with linespoints 1 3,\n")
else {print (printlog "----- no nibp-d.gnn files\n");}

#-----------------------------
if (-e $hrecgfilename)
{print (bpfile " \$hrecgfilename\" using 1:2 with points 1 10,\n")}
else {print (printlog "----- no hr-ecg.gnn files\n");}

if (-e $hroximfilename)
{print (bpfile " \$hroximfilename\" using 1:2 with linespoints 1 10,\n")}
else {print (printlog "----- no hr-oxim.gnn files\n");}

if (-e $cvpfilename)
{print (bpfile " \$cvpfilename\" using 1:2 with lines 1,\n")}
else{print (printlog "----- no cvp.gnn files\n");

## need to use a dummyline to allow the graph frame to appear even if no data points,
## and so allow the last line to have a comma if the following line gets ommitted
## so we make the dummyline have no final comma
## we do this by drawing a line below the graph-- ie it does not appear
$dummyline = " -20 with lines 1 \# dummy line";
print (bpfile "$dummyline \n");

close (bpfile);
print (printlog "--BP.gnu ....done\n");

#====================================================
## now create the sat file -------------------------
open(satfile, ">plot-sat.gnu")||die "ERROR: can't open plot-sat.gnu file\n");
print (satfile "#!/usr/bin/gnuplot\n");
print (satfile "# plot-sat.gnu script made by plotgnnk2.pl\n");
print (satfile "set terminal latex\n");
print (satfile "set output "plot-sat.pic" \n");
print (satfile "set size 1.40,$smallheight\n");
print (satfile "set xtics($timeline)\n");
print (satfile "set ytics ("80," 90," 100)\n");
print (satfile "set y2tics (80, 90, 100)\n");
#$y2label = qq(\"\\Sat \$\circ\\FIO\$_2\$ \bullet\")
print (satfile "set y2label \"Sat \$\circ\ FIO\$_2\$ \bullet\" 1\n");
print (satfile "set nokey\n");
print (satfile "set grid\n");
print (satfile "xmin=0;xmax=3600\n");
print (satfile "ymin=80; ymax=100\n");
print (satfile "plot [xmin:xmax] [ymin:ymax] \n");
$satfilename="sat".".g".$gnn;
$fo2filename="o2-insp".".g".$gnn;

if (-e $satfilename)

{print (satfile " \"$satfilename\" using 1:2 with linespoints 4 8,\\n")
else {print (printlog " ---**** no sat.gnn files\n")};

if (-e $fo2filename)
{print (satfile " \"$fo2filename\" using 1:2 with linespoints 4 10,\\n")
else {print (printlog " ---**** no fo2.gnn files\n")};

print (satfile "$dummyline \n");
close (satfile);
print (printlog "---SAT.gnu ....done\n");
#===================================================
## now create the FO2 file (FIO2 + N2O) -------------------------
open(fo2file, ">plot-fo2.gnu")||die "ERROR: can't open plot-fo2.gnu file\n";
print (fo2file "#!/usr/bin/gnuplot\n");
print (fo2file "# plot-fo2.gnu script made by plotg01a.pl\n");
print (fo2file "set terminal latex\n");
print (fo2file "set output "plot-fo2.pic" \n");
print (fo2file "set size 1.388,$smallheight\n"); #was 1.4
print (fo2file "set xtics($timeline)\n");
print (fo2file "set noytics\n");
print (fo2file "set y2tics (10, 30, 50, 70)\n");
#printf (satfile "set ytics (\"\" 10,\"\" 30,\"\" 50,\"\" 70)\n\n");
# $ylabel = qq("\%\\Sat $\circ$\\FIO\_2$ $\bullet$"");
print (fo2file "set y2label \"hello\\ hello "\n");
print (fo2file "set nokey\n");
print (fo2file "set grid\n");
print (fo2file "xmin=0;xmax=3600\n");
print (fo2file "ymin=10; ymax=70\n");
print (fo2file "plot [xmin:xmax][ymin:ymax] \n\n";
print (fo2file " 30 with lines 1,\\n");
print (fo2file " 50 with lines 1,\\n");
$fo2filename="o2-insp".".g".$gnn;
$n2ofilename="n2o-exp".".g".$gnn;
$pltfilename="pplat".".g".$gnn;

if ( -e $fo2filename)
{print (fo2file " \"$fo2filename\" using 1:2 with linespoints 4 10,\\n")
else {print (printlog " ---**** no fo2.gnn files\n")};

if (-e $n2ofilename)
{print (fo2file " \"$n2ofilename\" using 1:2 with linespoints 4 3,\\n")
else {print (printlog " ---**** no n2o.gnn files\n")};

## using diamonds (as for MAC)
if (-e $pltfilename)
{print (fo2file " \"$platfilename\" using 1:2 with linespoints 4 8,\\n")
else {print (printlog " ---**** no plat.gnn files\n")};

print (fo2file "\$dummyline \n")
close (fo2file);
print (printlog "---FO2.gnu ....done\n");

#===================================================
## now create the CO2 file -------------------------
open(co2file, ">plot-co2.gnu")||die "ERROR: can't open plot-co2.gnu file\n";
print (co2file "#!/usr/bin/gnuplot\n");
print (co2file "# plot-co2.gnu script made by plotg01a.pl\n");
print (co2file "set terminal latex\n");
print (co2file "set output "plot-co2.pic" \n\f");
print (co2file "set size 1.387,$smallheight\n"); #was 1.4
print (co2file "set xtics($timeline)\n");
print (co2file "set noytics\n");
print (co2file "set y2tics (2, 4, 6, 8, 10)\n");
# $ylabel = qq("\\%\\Sat \$\circ\$\FIO\$_2\$ \bullet");
print (co2file "set y2label "hello\ hello \"\n");
print (co2file "set nokey\n");
print (co2file "set grid\n");
print (co2file "xmin=0;xmax=3600\n");
print (co2file "ymin=2; ymax=10\n");
print (co2file "plot [xmin:xmax][ymin:ymax] \n\n\n");
print (co2file " 4 with lines 1,\\n");
print (co2file " 6 with lines 1,\\n");
print (co2file " 8 with lines 1,\\n");
$co2expfilename="co2-exp".".g".$gnn;
$rrfilename="co2-rr".".g".$gnn; ##plot rr here also

if (-e $co2expfilename)
{print (co2file " \"$co2expfilename\" using 1:2 with linespoints 4 1,\\n")
else {print (printlog " ---**** no co2-exp.gnn files\n")};

## we also plot the rr here to catch values >20
if (-e $rrfilename)
{print (co2file " \"$rrfilename\" using 1:2 with linespoints 4 10,\\n")
else {print (printlog " ---**** no rr.gnn files\n")};

print (co2file "\$dummyline \n");
close (co2file);
print (printlog "---CO2.gnu ....done\n");

#===================================================
## now create the TV file (tv + rr) -------------------------
open(tvfile, ">plot-tv.gnu")||die "ERROR: can't open plot-tv.gnu file\n";
print (tvfile "#!/usr/bin/gnuplot\n");
print (tvfile "# plot-tv.gnu script made by plotg01a.pl\n");
print (tvfile "set terminal latex\n");
print (tvfile "set output \"plot-tv.pic\" \n");
print (tvfile "set size 1.415,\$smallheight\n");
print (tvfile "set xtics($timeline)\n");
print (tvfile "set noytics\n");
print (tvfile "set y2tics (0, 250, 500, 750, 1000)\n");
# $y2label = qq("\\%\\Sat \$\circ\$\\FIO\$_2\$ \$\bullet\$\n");
print (tvfile "set y2label "hello\\ hello "\n");
print (tvfile "set nokey\n");
print (tvfile "set grid\n");
print (tvfile "xmin=0;xmax=3600\n");
print (tvfile "ymin=0; ymax=1000\n");
print (tvfile "plot [xmin:xmax][ymin:ymax] \\n");
print (tvfile " 250 with lines 1,\n");
print (tvfile " 500 with lines 1,\n");
print (tvfile " 750 with lines 1,\n");
$tvexpfilename="tv-exp".g$gnn;
$rrfilename="co2-rr".g$gnn;
if (-e $tvexpfilename)
  {print (tvfile " "$tvexpfilename" using 1:2 with linespoints 4 3,\n")
else {print (printlog " ---**** no tv-exp.gnn files\n")};
if (-e $rrfilename)
  {print (tvfile " "$rrfilename" using 1:2 with linespoints 4 10,\n")
else {print (printlog " ---**** no rr.gnn files\n")};
print (tvfile "$dummyline \n");
close (tvfile);
print (printlog "---TV.gnu ....done\n");
#===================================================
## now create the Vap file (vapIN, vapOUT, MAC) -------------------------
open(vapfile, ">plot-vap.gnu")||die "ERROR: can't open plot-vap.gnu file\n");
print (vapfile "#!/usr/bin/gnuplot\n");
print (vapfile "# plot-vap.gnu script made by plotg01a.pl\n");
print (vapfile "set terminal latex\n");
print (vapfile "set output \"plot-vap.pic\" \n");
print (vapfile "set size 1.376,\$smallheight\n");
print (vapfile "set xtics($timeline)\n");
print (vapfile "set noytics\n");
print (vapfile "set y2tics (0, 1, 2, 3, 4)\n");
# $y2label = qq("\\%\\Sat \$\circ\$\\FIO\$_2\$ \$\bullet\$\n");
print (vapfile "set y2label "hello\\ hello "\n");
print (vapfile "set nokey\n");
print (vapfile "set grid\n");
print (vapfile "xmin=0;xmax=3600\n");

print (vapfile "ymin=0; ymax=4\n");
print (vapfile "plot [xmin:xmax][ymin:ymax] \n");
print (vapfile " 1 with lines 1,\n");
print (vapfile " 2 with lines 1,\n");
print (vapfile " 3 with lines 1,\n");
$vapexpfilename="vap-exp".".g".$gnn;
$vapinspfilename="vap-insp".".g".$gnn;
$macbigfilename="mac-big".".g".$gnn;

if (-e $vapexpfilename)
    {print (vapfile " "$vapexpfilename" using 1:2 with lines 1,\n")}
else {print (printlog " --**** no vap-exp.gnn files\n")};

if (-e $vapinspfilename)
    {print (vapfile " "$vapinspfilename" using 1:2 with lines 2,\n")}
else {print (printlog " --**** no vap-insp.gnn files\n")};

if (-e $macbigfilename)
    {print (vapfile " "$macbigfilename" using 1:2 with points 4 1,\n")}
else {print (printlog " --**** no mac-big.gnn files\n")};

print (vapfile "$dummyline \n");
close (vapfile);
print (printlog "---VAP.gnu ....done\n");
#===================================================
# now run GNUplot on the .GNU files
print (printlog "---running GNUPLOT on all the .gnu files\n");
system ("gnuplot plot-bp.gnu");
system ("gnuplot plot-sat.gnu");
system ("gnuplot plot-fo2.gnu");
system ("gnuplot plot-co2.gnu");
system ("gnuplot plot-tv.gnu");
system ("gnuplot plot-vap.gnu");
print (printlog "...........GNUPLOT ... done\n");

##===============gnnheader.dat file=============================
print "writing the <gnnheader.dat> file to contain header for Anes record \n";

open (outfile5, ">gnnheader.dat")||die "ERROR: can’t create file <gnnheader.dat>\n";
#
$timenow = localtime;
print (outfile5 "%% gnnheader.dat: created $timenow\n");
print (outfile5 "%% file generated by <plotgnnk2.pl> RWD Nickalls\n");
$fname="anes-".$gnn.".dvi";
print (outfile5 "\\header{$starttimeunix}{$originalgmt}{$fname}\n");
## note that here originalgmt = starttimegmt
close (outfile5);
print "......<gnnheader.dat>.... done\n";

###==============================================

print (printlog "---running LATEX on prtanes6.tex\n");
system ("pslatex prtanes6.tex"); ### use pslatex
$dvifilename="anes-".$gnn.".dvi";
## copy the .dvi file to have a gnn.dvi filename
system ("cp -v prtanes6.dvi $dvifilename");
## make the .ps files
$psfilename="anes-".$gnn.".ps";
system ("dvips $dvifilename -o $psfilename");
print (printlog ". . . . . . . . . . . . LATEX ...done\n");
## make the pdf files---
system ("pdflatex prtanes6.tex"); ### use pslatex
$pdffilename="anes-".$gnn.".pdf";
## copy the .pdf file to have a gnn.pdf filename
system ("cp -v prtanes6.pdf $pdffilename");
##------------------
## view the output graphs
## system ("gv $psfilename");
##------------------
## show the .dvi file on the screen
## system ("xdvi $dvifilename");
## now send file to the printer
## system ("dvips $dvifilename");
###### goto OUTLINE; #***************
##------------------
## print the .dvi file to printer
## system ("dvips prtanes6.dvi");
##--------------------------------------------------
}

###=====process the prtdrug stuff=========================

### process the baselog.data file
system ("perl ./base2texd.pl");
## now latex the prtdrug file
system ("latex ./prtdrug.tex");
## copy the .dvi file to have a anes-drug.dvi filename
system ("cp -v prtdrug.dvi anes-drug.dvi");

###--------------------------
#####

#### make the PS version of the .dvi file to printer

system ("dvips anes-drug.dvi -o anes-drug.ps");  

####-----make the pdf file---

system ("pdflatex prtdrug.tex");

## copy the .pdf file to have a gnn.pdf filename

system ("cp -v prtdrug.pdf anes-drug.pdf");

------------------

## view the output .ps graphs

## system ("gv anes-drug.ps");

## print out

# system ("dvips anes-drug.dvi");


LASTLINE:;
OUTLINE:;

close;

#=======================SUBS================================

sub gnnmax{

## returns total number of hours (gnnMax)

## by scanning the file <timefile.dat>

## the <timefile.dat> file is made by

## the SUB makegnnfiles() in prog cam2gnnH.pl

$gnnmax=0;

## open the file for input

open (timefile, "<timefile.dat") || die "ERROR: can't open file timefile.dat"

#------------------

LINE: while (<timefile>){

next LINE if /^#/; #skip # comments

next LINE if /^%$/; #skip % comments

next LINE if /^$/; #skip blank lines

# grab the whole line as a string

# hour, unixtime, gmtime, gnnfilename

$dataline = $_;

chomp($dataline); # removes the line-ending

print (printlog "[SUB start_time] dataline string (timefile.dat) = $dataline\n");

# place the params into an array

@value = split (/[\,]/, $dataline);

$hour = $value[0];

$time_unix = $value[1];

$time_gmt = $value[2]; #GMT yyyy:mm:dd:hh::mm:ss

gfile = $value[3];
# get the largest Gnn value (gnnmax)
    if ($hour >= $gnnmax) {$gnnmax=$hour};
} # end of while{
    close (timefile);
    print (printlog "[SUB start_time] GnnMax = $gnnmax\n");
    return $gnnmax;
} #end of sub

#
##=================================
__END__