



**DEPARTMENT OF  
MEDICAL PHYSICS & BIOENGINEERING**

---

# **LPCI: AN INTERFACE FOR UPDATING PARAMETERS IN AN FES IMPLANT**

**SIMON P. HEARN**

*MSc. Biomedical Engineering & Medical Imaging*

Supervisors: Professor Nick Donaldson and Anne Vanhoest

AUGUST 2005



7029 Words

## **Abstract**

This report summarises the work done and results achieved in designing and building the new version of the SLARSI<sup>1</sup> PC interface (LPCI<sup>2</sup>); a software application intended for communication with a nerve root implant. This report provides background information on functional electrical stimulation (FES) and FES implants and presents a critical analysis of the old LPCI application. In response to this research, a design for the new version is outlined and its implementation process is described.

The major results of this work include the addition of urologic management functions, enhancement of the user interface, user documentation and a developer's guide.

---

<sup>1</sup> Sacro-Lumbar Anterior Root Stimulation Implant

<sup>2</sup> See section 5.8 for a discussion about the program name

## Contents

<b>1</b>	<b>INTRODUCTION</b>	<b>5</b>
1.1	AIM	5
1.2	OUTPUTS	5
<b>2</b>	<b>BACKGROUND</b>	<b>6</b>
2.1	SPINAL CORD INJURY AND ELECTRICAL STIMULATION	6
2.2	IMPLEMENTATION OF FES	7
2.2.1	Root Stimulator Implants	7
2.2.2	The SLARS Implant	8
2.2.3	The control box and stimulation programs	8
<b>3</b>	<b>CRITICAL ANALYSIS OF THE OLD LPCI</b>	<b>11</b>
<b>4</b>	<b>DESIGN OF THE NEW VERSION</b>	<b>13</b>
4.1	OBJECTIVES FOR THE NEW VERSION	13
4.1.1	General objectives	13
4.1.2	Specific objectives	13
4.2	DEVELOPMENT TOOLS	14
<b>5</b>	<b>IMPLEMENTATION</b>	<b>15</b>
5.1	GETTING ACQUAINTED	15
5.2	REMOVAL OF ALTERNATING GAIT SETS AND ADDITION OF UROLOGIC SETS	15
5.3	CHANGES TO CYCLING SETS AND TRAINING SETS	16
5.4	FREQUENCY SELECTION	17
5.5	STM FILES	17
5.6	CONTROL BOX RELATED CHANGES	18
5.7	PATIENT SPECIFIC CHANNELS	19
5.8	ENHANCEMENTS TO THE USER INTERFACE	19
5.9	PROGRAM NAME	21
<b>6</b>	<b>USER MANUAL &amp; DEVELOPERS GUIDE</b>	<b>22</b>
<b>7</b>	<b>DISCUSSION</b>	<b>23</b>
7.1	ACHIEVEMENTS	23
7.2	TESTING	23
7.3	CHALLENGES OVERCOME AND LESSONS LEARNED	24
7.3.1	Compiler incompatibility issues	24

7.3.2	Working on existing code.....	24
7.3.3	Common errors encountered.....	25
7.4	FUTURE DEVELOPMENTS .....	26
<b>8</b>	<b>REFERENCES.....</b>	<b>29</b>
<b>9</b>	<b>APPENDIX .....</b>	<b>30</b>
9.1	AN EXAMPLE OF AN STM FILE.....	30

# 1 Introduction

## 1.1 Aim

This project is a small but integral part of a larger research project, undertaken by the Implanted Devices Group at UCL, which aims to develop an implanted nerve root stimulator. The aim of this project is to design and implement the new version of the PC software used to interface with the implant's control box. The new version will reflect the recent changes to the hardware and make the process of updating parameters in the implant simpler more straightforward for patients and clinicians.

## 1.2 Outputs

Included with this report are the user manual, the developers guide and a CD-ROM. These should be regarded as outputs of the project and part of the final product being produced. The outputs complement the final report by describing how the software operates (user manual) and how it was programmed (developers guide). The CD-ROM contains the setup file which installs LPCI and the interactive help documentation, the user manual and developers guide, source code with Borland C++ 5.02 compiler, the help file project with Macromedia RoboHelp X5 and Inno Setup 5<sup>1</sup>.

---

<sup>1</sup> Software for creating setup files

## 2 Background

### 2.1 Spinal Cord Injury and Electrical Stimulation

Electrical stimulation is a term used to describe a wide variety of research and treatment approaches that use electrical signals to stimulate muscle contraction. Traditionally, electrical stimulation has been used in medicine as a short term treatment for pain relief but since the 1950's the use of electrical stimulation has diversified. In common use now are devices like pacemakers, dropped-foot stimulators and cochlear implants, all of which use electrical stimulation to produce a long term functional response. This type of stimulation is called FES (functional electrical stimulation). In this case FES is being used with paraplegics to restore function to their bladder, bowel and legs; this raises the quality of life of the patient, improving several aspects of their health and independence.

It has been shown that paralysis of the legs caused by spinal cord injury (SCI) can lead to secondary medical complications such as muscle atrophy, poor circulation, pressure sores, oedema, loss of bone mass, heart disease (due to lack of cardiovascular exercise) and poor self image [6]. Exercise is very important for paraplegics and while some use their arms in wheelchair sports this is not ideal as the shoulders can be easily damaged. Electrical stimulation of leg muscles provides a more suitable solution. Some research groups have explored stimulated walking



Fig 1. Paraplegic patient cycling a recumbent tricycle using an FES implant.

but have encountered additional problems with balance and complexity of muscle movements which make these systems costly, bulky and difficult to implement. Cycling on the other hand, using a recumbent tricycle or specially designed fixed exercise bike, eliminates problems of balance and simplifies muscle contractions to the legs only [6].

As well as the previously mentioned secondary medical complications, SCI also leads to problems with the bladder and bowel; patients have difficulty with evacuation, incontinence, pain and distension [4]. FES can help to restore continence and ensure complete evacuation by controlling the muscles in the bladder and the bowel [7]. The

use of FES implants has been shown to decrease the frequency of urinary tract infections and increase continence [8] as well as reducing the cost of bowel and bladder management in patients with SCI by up to 80% [2].

In some patients, however, the sensory nerves are still intact causing possible complications as reflex sensory impulses can cause spontaneous voiding leading to incontinence. This can be avoided by cutting the posterior roots responsible for the sensory impulses [7].

## **2.2 Implementation of FES**

### **2.2.1 Root Stimulator Implants**

There are three main methods of implementing FES; the first uses external electrodes glued to the surface of the skin; the second uses wires passed through the skin directly to the desired muscle; the third uses electrodes and stimulators surgically implanted into the patient [3]. The first method is the most straight forward as surgery is not necessary, but requires more time and effort to place all the electrodes; this makes it an unattractive option for paraplegics. The drawback from a researcher's point of view is that the stimulation is non-specific, i.e. contraction of one particular muscle is very difficult. The second method overcomes this by providing direct links to muscles but is unsuitable for long term use because of the wires that protrude from the skin.

The third option is the best implementation for long term use; the only major consideration being the placement of the electrodes. One option is to place the electrodes directly onto the peripheral nerves near to the target muscles but this involves extensive surgery in order to activate a large number of muscles. A far less invasive approach is to place the implant at the anterior roots of the lower spinal nerves [3]. This type of implant is often called a Root Stimulator Implant (RSI) and comes in many varieties depending on which nerves are being targeted, for example, lumbar nerves (to control legs) or sacral nerves (to control bowel, bladder and sexual function) and whether the implant deals with posterior nerves (from sensory neurons) or anterior nerves (from motor neurons). The main varieties of RSI are SARI (Sacral Anterior RSI) which controls just the motor function of the bowel and bladder, SPARI (Sacral Posterior & Anterior RSI) which control motor function and detects sensory information from the bowel and bladder (which is useful for blocking sensory

impulses in patients with intact posterior nerves where reflex voiding can occur thus improving continence), LARSI (Lumbar Anterior RSI) to exercise the leg muscles and finally SLARSI (Sacro-Lumbar Anterior RSI) which controls the motor function of the legs, bowels and bladder.

### 2.2.2 The SLARS Implant

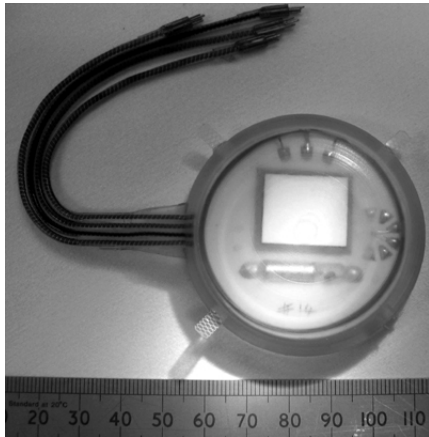


Fig 2. The SLARS implant

The implant used in this project has three main functions. Firstly to enable the patient to ride a recumbent tricycle, secondly to allow the patient to train their leg muscles in their homes and thirdly to provide bowel and bladder management for the patient. In order to implement each of these functions, a 10-channel SLARSI is connected to both sides of the anterior spinal roots L3 to S4 which control motor function of the lower body. See Table 1 for more details.

Channel	Nerve root	Function
1	L3 Left	Legs
2	L3 Right	Legs
3	L4 Left	Legs
4	L4 Right	Legs
5	L5 Left	Legs
6	L5 Right	Legs
7	S1 Left	Toes and Foot
8	S1 Right	Toes and Foot
9	S2 Left and Right	Bowel and Bladder
10	S3 & S4 Left and Right	Bowel and Bladder

Table 1: SLARSI connections and corresponding functions.

### 2.2.3 The control box and stimulation programs

The implant is triggered by signals from a control box via an RF link across the skin. These signals are interpreted as stimulation strengths for each of the 10 channels. The control box can store up to 8 programs in its memory. The programs come in



three main types; cycling, training and urologic. Each sends different signals to the implant depending on their function. For example, a cycling program would send signals to contract the leg muscles needed for propulsion.

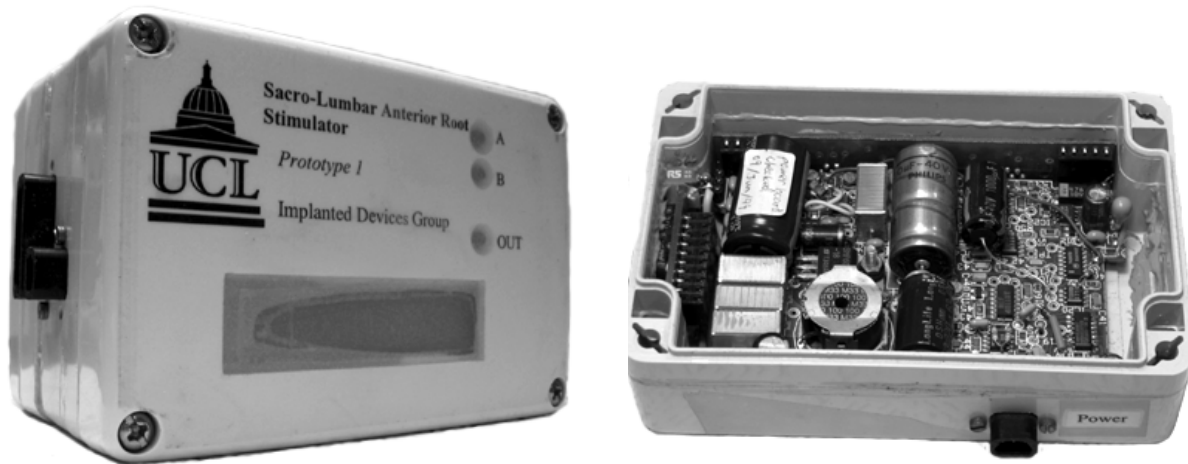


Fig 3. The control box prototype, with an open view on the right.

The control box has an LCD display and buttons to allow selection of the required program. Training and urologic programs are static and are run linearly from the control box memory. Cycling programs on the other hand take inputs from the crank and the throttle on the tricycle. A 7-bit encoder measures the crank angle and updates the signal strengths for each of the 8 leg muscle electrodes in the implant depending on the phase angle of the crank. The patient can control the level of stimulation between preset maximum and minimum values by using a throttle on the tricycle. This enables the correct muscles to be contracted by the correct amount at the correct time in the cycle [6]. The only problem is that there is a delay between stimulation of nerves and action against the pedals; this causes a phase lag between the desired action and the actual action. For example, the patient's right leg is at angle A. The crank encoder measures this angle and triggers the control box to send the appropriate signals to the implant. The implant stimulates the nerves and the signals get passed down to the leg. The leg muscles contract and the pedals are pushed. The problem is that by the time the pedals are pushed, the leg has moved to angle B which needs a slightly different stimulation pattern to produce optimum thrust on the pedals. Thus the patient is constantly working at less than optimum. The solution to this problem is the use of a phase advance, so that when the leg is at angle A, the stimulation pattern for angle B is sent, this ensures the optimum stimulation occurs at the time when the action is produced.

One of the aims of FES cycling is to exercise and strengthen the leg muscles. As this occurs, the parameters for the cycling programs (electrode signal strength and pulse duration for each channel) need to be updated. Likewise, the bowel and bladder become stronger with use and thus their program parameters also need updating. This is achieved by using a software application called LPCI (SLARSI PC Interface<sup>1</sup>). The user, typically a clinician, sits at a computer connected to the control box and enters the new data using a graphical interface, then transfers it to the control box. It is this piece of software that is the project's main focus.

---

<sup>1</sup> See section 5.8 for a discussion about the program name.

### 3 Critical analysis of the old LPCI

At the start of this project, a version of LPCI was already being used for research purposes (see figure 4 below). An analysis was made of the old LPCI, the results of which have been incorporated into the user manual and developers guide for the new version. See the attached documents for more information on what LPCI does and how it was implemented.

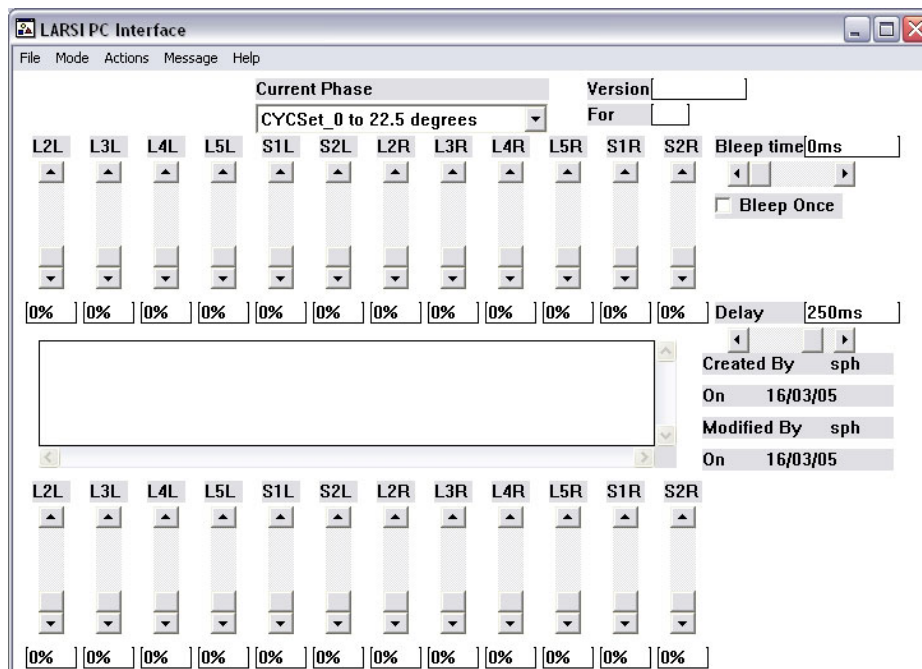


Fig 4. The old LPCI – creating a new cycling set [see user manual for more information]

The old LPCI application worked with a different implant to the one used here; a LARSI was used which had 12 channels connected to the roots L2 to S2. The implant was designed for cycling, training and alternating gait exercises hence the old LPCI was designed to work with three types of set: cycling and training, which are very similar to the new LPCI, and alternating gait which was divided into 14 phases based on a somewhat simple decomposition of the human gait as 14 symmetrical sequences.

Having only been used for research purposes; the old LPCI had not been used by any clinicians, which were the intended users of the final system, only by a few members of the research group. Those that had used it reported that it was more straight-forward than the alternative option which involves writing assembly code but they also mention that it was confusing and still took a long time to perform the required tasks.

## Limitations and Shortcomings

The major complaints from those who have used the old LPCI pertained to how difficult it was to use.

The following points were made<sup>1</sup>:

1. The cancel button in the initial dialog box asking for user initials did not work.
2. It was unclear what each of the menu items did.
3. It was unclear when a set could be edited and when it could not.
4. The distinction between edit mode and online mode was not stated.
5. It was disconcerting when the program was busy; it looked as if it had crashed.
6. The program did nothing if the user attempted to add/delete a set when in edit mode. Although this was the correct behaviour, it confused the user.
7. The common procedures required several steps. For example, adding a set to the control box required LPCI to be in online mode and for the desired memory location to be free. If it was not free, the user would have had to save the new set to the pc, go back online, select the existing set in the control box, delete the existing set, re-load the new set, go back online again and finally add the new set.
8. New sets could easily be lost by accident, for example if the user created a new set without first saving the one they were working on.
9. There was insufficient labelling for the controls on the screen.
10. Phase and set names were abbreviated and difficult to understand.
11. Some of the message dialogs were unhelpful and confusing.
12. Minor bugs occurred, for example errors occurred when data was read from the control box immediately after a set had been added to it. Also the bleep once tick box was not cleared when a new set was created.
13. There was no on-screen help file.
14. The user documentation was incomplete and difficult to follow.

---

<sup>1</sup> See the user manual for more information on the specific terms used here

## 4 Design of the new version

### 4.1 Objectives for the new version

#### 4.1.1 General objectives

The new version of the software has to both reflect the recent developments of the implant and control box and address the limitations and shortcomings of the old LPCI.

#### 4.1.2 Specific objectives

##### 1) Changes needed to reflect recent developments of the system

- a) The most important development of the system is the addition of urologic functions and removal of alternating gait programs. This requires an overhaul of the LPCI code.
- b) The old version of LPCI was designed for a 12 channel LARSI since it was used only for stimulating leg muscles. The new implant is a 10 channel SLARSI as described above. This change requires a restructure of the channel definitions.
- c) The control box has also changed since the old LPCI was created. To make it simpler to use it now only requires 3 buttons in normal use instead of 8. This means the button actions have to be altered since it is LPCI that defines what the buttons do in each state<sup>1</sup>.
- d) The ability to select the stimulation frequency for a particular program is needed since altering the frequency could provide more optimal stimulation.

##### 2) Changes to enhance the user friendliness

- a) The program should be less confusing and simpler to use.
- b) The problems outlined in the previous section should be solved.
- c) An interactive help file should be added to the program, giving the user direct access to tutorials and help topics.
- d) A user manual should be produced to explain in detail how to use the application.

##### 3) A developer's guide should be produced to enable much easier and quicker development of LPCI in the future.

---

<sup>1</sup> See developers guide for more information on states

## 4.2 Development Tools

The options for programming the application were either to start from scratch using a modern application builder such as Borland C++ Builder or Macromedia Flash MX or to alter the original code. The modern compilers would have been easier to use since they employ a visual approach which would require much less programming experience. They can also produce a very professional looking application. But since significant work had already been done on the LPCI program it was decided that it should be adapted rather than starting from scratch. The possibility of upgrading the code to be compatible with Borland C++ Builder was explored but deemed too complicated and time consuming to be viable. It would require the windows and all controls and dialogs to be re-built in C++ Builder, then the supporting code would have to be updated to be compatible with the newer version of OWL (Object Windows Library – a collection of ready made functions to do with windows interface objects). It was decided that the original code would be updated using the same compiler as it was created with, Borland C++ 5.02. There was considerable difficulty obtaining the correct compiler, this was mainly due to lack of information from the original developer<sup>1</sup>.

---

<sup>1</sup> See the discussion section below for more information

## **5 Implementation**

### **5.1 Getting acquainted**

The initial task of the project was to scrutinise the code to understand the relationship between the various functions. For example, to figure out what happens when the user double clicks the LPCI.exe icon. In order to plan the required updates, specifically the introduction of urologic sets, the communications between LPCI and the control box had to be examined. In order to achieve this, the function to add a set to the control box was altered to output a text file rather than sending the information directly. This option proved doubly useful since during development of the application there was no access to the control box.

### **5.2 Removal of alternating gait sets and addition of urologic sets**

The alternating gait function was removed from the system because it proved to be ineffective due to the complex muscle stimulations patterns needed. Very few patients could stand alone and the need for additional support made this function impractical. This meant all references to alternating gait sets had to be removed from the program, including all constants, variables and functions. These were then replaced with references to urologic sets, adapting where necessary.

Urologic sets come in three types; bowel, bladder and erection. This prompted the introduction of sub-sets which would be identical in every way except their name and the memory address where they reside in the control box. Instead of programming a set of functions for each sub-type, they were programmed to use the same set of functions with a variable to distinguish between them.

Urologic sets have 3 stimulation phases called pre-fatigue, on and off. The reason for these phases lies in the physiology of the bladder. The sphincters and detrusor (bladder wall) both contract during the on-phase cancelling the effect of each other. However, being different type of muscles, the external sphincter relaxes faster during the off-phase, allowing a burst of urine to be expelled since the bladder is still contracted. Weakening the sphincter during the pre-fatigue phase using a pulse not strong enough to stimulate the detrusor enhances this effect [1]. The control box has one memory location for each urologic sub-set so LPCI was programmed to select the memory location automatically, based on the sub-type. To ensure an existing set

couldn't be overwritten by accident, a check was programmed in, warning the user when they were about to overwrite an existing set.

Since urologic sets deal with the lower spinal roots (S2, 3&4), only two stimulation channels are used, S2 and S34. The other 8 channels are set to zero and not displayed on-screen (see figure 5). Urologic phases include an additional variable; the current level. This can be set to low, medium or high for each phase. Generally the sacral roots require a higher current than the lumbar roots; this variable allows the user to select the most optimum current for the patient.

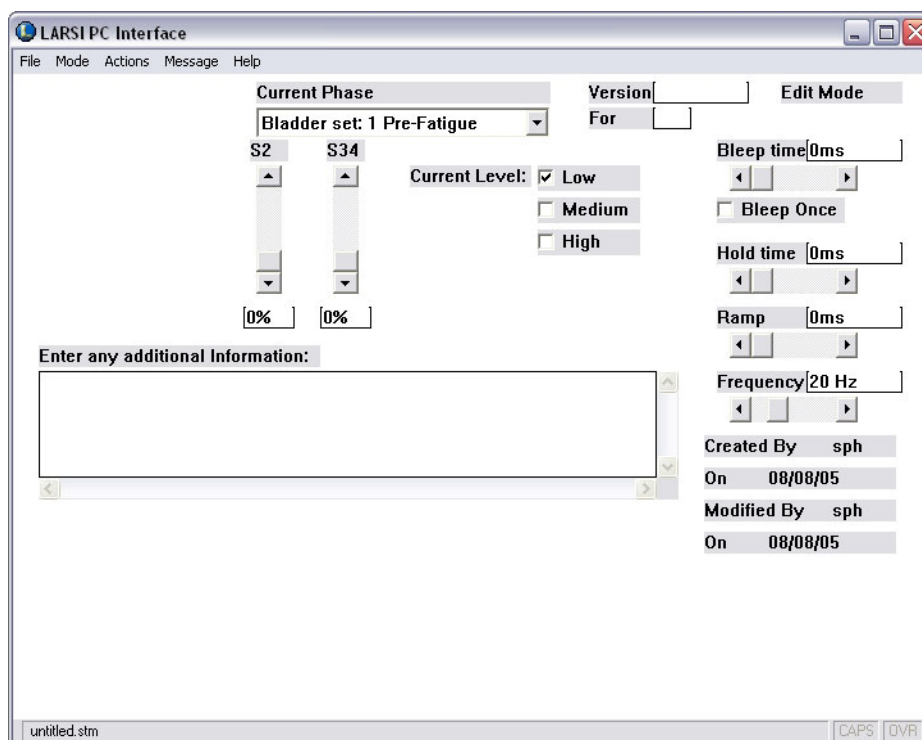


Fig 5. New LPCI – creating a new urologic set [see user manual for more information]

### 5.3 Changes to cycling sets and training sets

The number of channels for cycling and training sets was reduced from 12 to 8; L3, L4, L5 & S1 (left and right). This only affected the on-screen controls, as the remaining 4 channels were kept, two of them for the urologic sets and two spare channels for future development. The number of channels sent to the control box was kept at 12, with the unused channels set to zero.



The memory allocation for cycling sets was reduced from 6 to 2; this caused only minor changes to the program. A feature was added that allows overwriting (with the user's permission) of cycling sets. This means users no longer have the trouble of having to delete sets before adding new ones to the control box.

It was decided to split training sets into three sub-types, wheelchair training, couch training, and weight training. This was implemented in the same way as the sub-types for the urologic sets, with common functions and a variable to distinguish them. Like the urologic sets, one memory location was allocated to each sub-type of training set.

#### **5.4 Frequency selection**

In order to allow the clinician to select the most effective frequency for the stimulation, a frequency level variable was added to each phase. Allowing the clinician to select different frequencies for different phases within a set was not deemed necessary at the time of development, so LPCI forces each phase to have the same frequency. The only exception to this is the pre-fatigue phase of the urologic sets. This phase was allowed to have a different frequency as fatigue of muscles is more effective at higher frequencies. Since the control box regulates the frequency by sending pulses at fixed intervals, it made sense to send the period of the stimulations rather than the frequency. The period was calculated using the following formula:

$$\text{Period} = \text{clock speed} / (5 \times \text{frequency})$$

Where the clock speed is 500 kHz and the factor of 5 derives from the 5 interrupts it takes to send one pulse train.

#### **5.5 STM files**

STM (stimulation) files are the text files produced when the user saves a set to the PC. They were re-designed to reflect the changes made to LPCI. This included a new format for the urologic sets, the addition of frequency levels and the alteration of stimulation channel settings. See the appendix for an example of a bladder stimulation file.

### 5.6 Control box related changes

As mentioned above, the control box was re-designed both externally with a reduction of buttons to three, and internally with an overhaul of the menu system as shown below.

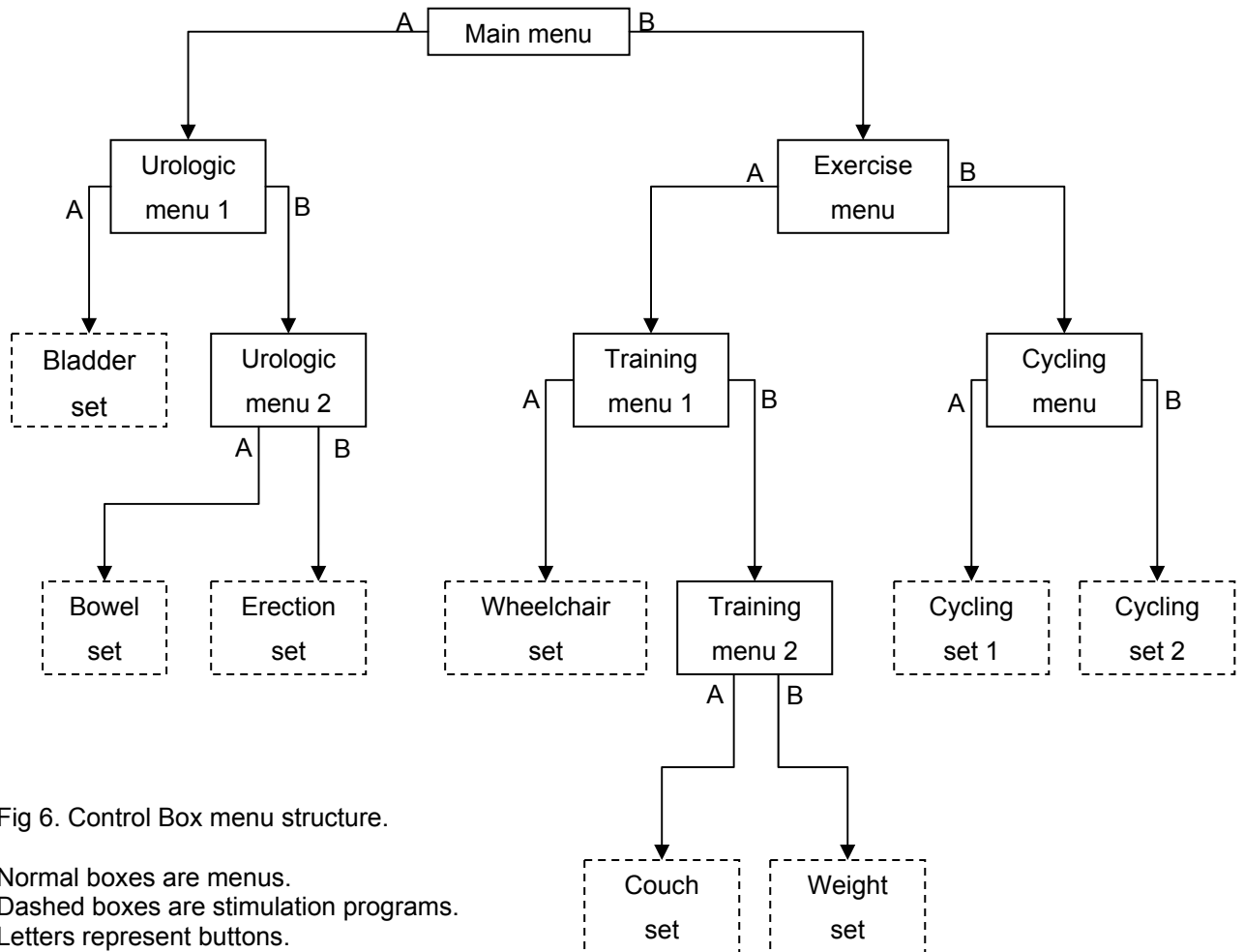


Fig 6. Control Box menu structure.

Normal boxes are menus.  
 Dashed boxes are stimulation programs.  
 Letters represent buttons.

In terms of changes to the software, the initialisation sequence that sets up the quiet states in the control box was altered to reflect the new structure. One key addition was the activation of control box buttons for sets that have not been loaded.

In the old version of LPCI, the control box buttons which execute sets was only activated when the set it corresponded to had been added to the control box, ensuring that users couldn't select a set that didn't exist, thus entering an empty area of memory and crashing the control box. The result of this was that nothing happened when the button was pressed and therefore the user was unaware of what was happening.

With the new system, the button was programmed to link back to the menu with a given transition code that could be read by the control box. It would then be aware that the user was trying to activate a set that wasn't there and could react to this by displaying a message for the user.

### **5.7 Patient specific channels**

The old version of LPCI was designed for two patients only: defined as JH and CB. Only sets with these initials could be added to control boxes. The reason for this was that each patient had different channel settings, for example JH had S2R in channel 5 and CB had S1R in the same channel, so these settings were hard coded into the program to ensure patients didn't receive stimulation in the wrong channel.

Because the new version of LPCI will be used with other patients, this restriction was removed and one universal channel setting is used by all patients. This may not be the ideal solution as it demands that surgeons connect the nerve roots in a very specific way rather than the simplest way for them. Therefore, a standard block of code used to link roots and channels for a specific patient was left as comment (hidden code). This way, in the event of a surgeon not connecting the roots as expected, it would still be possible to modify LPCI to include this specific patient. An even better solution would be to include a feature in the software that enables clinicians to add patient specific channels from the user interface rather than editing the source code.

### **5.8 Enhancements to the user interface**

The following list is a record of the main enhancements designed to make LPCI more user friendly<sup>1</sup>. They were mostly implemented in response to the limitations and shortcomings mentioned earlier.

- To reassure the user that the program is working a "Busy, please wait" message was programmed to be displayed whenever the application is in the middle of a task.
- LPCI was programmed to detect whether the set on-screen has been modified and to warn the user to save it to the computer when they try to create a new

---

<sup>1</sup> See the user manual for more information on the specific terms used here

set, load a set or exit the program. This ensures that no data is accidentally lost.

- The user is given the option to overwrite existing sets in the control box. This simplifies the adding procedure, avoiding having to delete the old set first.
- Users are also permitted to add sets to the control box from within edit mode as opposed to having to go online before adding the set. This is much faster and less confusing.
- To delete a set the user needs to be in online mode in order to select a set within the control box. In the old version of LPCI, if the user tried to delete a set when in edit mode nothing happened; this was confusing. Now if this happens, a message box informs the user that they have to be online.
- A status bar was added to the application. This displays the name of the displayed set; if the set is from the PC the path and filename is displayed, if the set is from the control box the title of the set is displayed. The status bar also displays comments as the user selects a menu item.
- The very first dialog asking for user's initials was fixed so that the cancel button exits the application.
- On-screen indicators were added to inform the user which mode LPCI is in, i.e. edit or online. An indicator showing whether verbose messaging is on was also added.
- Phase and set names in the drop down boxes were changed to their full names rather than an abbreviation; this makes it clearer as to what the user is selecting.
- The public messages (as opposed to the optional verbose messages) were changed to give more information in user friendly language.
- The on-screen controls were given more labels, for example the additional information box was labelled and the difference between the maximum and minimum stimulation levels in a cycling set was made clearer.
- The information boxes at the bottom of the screen in training sets was removed, the original purpose of these is not entirely clear.
- The minor bugs observed were fixed.
- The usage log menu items were removed. This is because the log wasn't working properly and it was decided to remove the ability to use it until it was fixed in the control box program.

- Icons were created for LPCI; this gives a more professional look.
- An interactive help file was created and made accessible from within the program. A free trial of Macromedia RoboHelp X5 was used to construct the help file<sup>1</sup>.

## **5.9 Program Name**

Since the new version of LPCI would be working with a SLARSI rather than a LARSI, LPCI would actually be changed to SPCI, but it was decided to keep the original name until a more appropriate, catchier name can be thought up. In terms of the programming, the application would always be called LPCI as it would take a lot of work to change it. But on the outside it is fairly easy to change.

---

<sup>1</sup> See attached CD-Rom

## **6 User manual & developers guide**

Updating the LPCI software was only one of the objectives of this project. The other objectives were to produce a user manual and a developers guide. These documents are included with this report as outputs of this project.

The user manual was designed to be distributed with the software so that the users can quickly and simply learn how to use it. It was seen as a vital part of the final product so much care was taken to ensure all the relevant information was included. It describes the different parts of the user interface; the controls and menus and gives tutorials on how to perform common tasks such as creating a new data set and adding it to the control box.

The developers guide was designed for anyone who may wish to develop the software in the future. It acts a reference tool supporting the source code by providing information on all the functions and how they relate to each other. It also describes how LPCI communicates with the control box giving information such as memory addresses and explaining the codes used in writing a state.

## 7 Discussion

### 7.1 Achievements

The outputs from this project should ensure that clinicians with no specific IT skills can access the control box of a SLARSI system and update the parameters within it. This will have a direct effect on the patients using the SLARSI system enabling them to receive a swift and accurate response when they need alterations to their stimulation programs.

In line with the aims of this project, the following achievements have been made:

- A new version of LPCI compatible with the new implant and control box has been developed.
- The user interface has been enhanced by incorporating additional features making the program simpler and more straightforward.
- An interactive help file has been created.
- A user's guide has been produced.
- A developer's guide has been produced.

Although the enhancement of the user interface has been great, there are still many improvements that could be made<sup>1</sup>. Further development was not undertaken either because of time constraints, lack of experience or lack of development tools.

### 7.2 Testing

The necessity to work alongside and communicate with the potential users is vital in developing software since there is no point in creating the software if the users can't use it or don't like it. For this reason, time was set aside for user testing and feedback. Unfortunately, due to time constraints caused by problems detailed below the program was not tested with the clinicians who will eventually be using it, but was, however, tested with members of the research group who were unfamiliar with LPCI. They reported that the software was easy to understand and operate particularly with the help file on hand. It is hoped that user testing will occur in the future to help identify and solve any outstanding problems and will also enable the addition of a troubleshooting section in the users guide.

---

<sup>1</sup> See section 7.4 for more details

## 7.3 Challenges overcome and lessons learned

### 7.3.1 Compiler incompatibility issues

The old LPCI program was developed in 2000. Since then, the original programmer of LPCI has moved on from the department. Unfortunately, sufficient documentation was not provided and the version of the Borland compiler used could not be ascertained. Borland C++ Builder v5 was obtained first since it was a fairly modern version but when the source code files were imported and compiled, errors occurred because the Object Windows Library (OWL) files referenced in the source code could not be found. The OWL files were obtained and the code was compiled again with no luck as the compiler couldn't understand the OWL files. It became obvious that an incompatible compiler was being used. After much research and troubleshooting, the correct compiler was found – Borland C++ 5.02. This compiler is much older than C++ Builder and has a more primitive development environment. Unfortunately, even with the correct compiler, the code didn't compile correctly because of errors due to missing sections of code in two of the source files.

After replacing the missing sections the code compiled successfully. These difficulties could have been avoided if sufficient documentation had been available and so to avoid similar problems in the future this project is supported by suitable documentation.

This project has also highlighted the need for code to be written in a more portable format that is future proof, so that a new compiler “off the shelf” would be compatible. It is now clear that future trends play a very important part in designing software; developers need to be sure that their software can be upgraded ten years down the line.

### 7.3.2 Working on existing code

Before any of the code could be upgraded, it needed to be understood. This involved going through all the functions working out what they did, how they did it and how they fitted together. This was a laborious task but was a vital part of the development process since the previous developer hadn't left this information. The results of this analysis are presented in the developers guide. Also, the code has been amply documented with comments explaining the changes made and any more complex parts.



### 7.3.3 Common errors encountered

#### 1. Writing to control box

The most common and frustrating errors encountered were to do with reading from and writing to the control box. When writing to the control box there are three elements that have to tally up with the number of bytes being sent:

- a) *The format specifiers within the “sprintf” command.* This command writes the variables which are to be sent to the control box into a buffer (a temporary character string variable). The programmer can specify the format in which the variables are to be written using format specifiers, e.g. a decimal integer can be converted to a two-digit hex code using the specifier %02X. The number of format specifiers must equal the number of variables, else the compiler won't detect the error and the program will just crash when that code is executed.
- b) *The Motorola S-record byte count.* All data written to the control box is done so using S-records, the designated method of communicating with a Motorola micro-processor. Each S-record has to define how many bytes it is sending; if this is incorrect then the communication will fail with no indication that it has done so. Obviously this is very important to get right.
- c) *The “number of bytes to write” declaration in the WriteFile command.* The WriteFile command is used to send the contents of the buffer mentioned above to the communications port on the PC and on to the control box. The number of bytes to be written has to be declared here. This can get very confusing since the number quoted here is not the same as in the s-records. For a start the number of bytes in the s-records is a hex number, here it is a decimal. But the main difference is that here, each character is counted as a byte whereas in the s-records a byte is counted as a pair of characters. This is due to the different ways the characters are represented in the compiler and in the Motorola chip.

## 2) Reading from the control box

Reading from the control box requires as much care as writing to it. The control box has been designed to respond to inputs by sending back an echo of what was written. While this is very helpful in debugging commutation errors, it has been the cause of many errors itself. When the control box sends data to the PC, the PC stores it in a buffer ready to be read by LPCI. A very common error was not reading enough data, thus leaving a few bytes in the buffer. This meant that the next time data was read from the control box it was preceded by the remnant of the last batch of data. This meant that LPCI was not receiving what it was expecting. The solution to this was to make sure the number of bytes being read was equal to what the control box was sending. Another way would be to flush the buffer after each communication.

## 3) Memory allocations

A relatively minor error which caused a lot of problems was the allocation of memory within the control box. After increasing the size of a state by adding the current level and the frequency; the training sets were overlapping and overwriting each other. This was fixed by allowing enough space for each set plus some extra for future development.

## 7.4 Future developments

1. The program should be tested with potential users to identify any areas which still need development.
2. Instead of having to first load a set from the computer then add it to the control box; it would be quicker and easier if the user could add sets to the control box directly from an stm file on the computer. Even better would be the ability to add multiple stm files at once, this would be ideal in the case where the control box has crashed and needs restoring.
3. There could be an auto save feature that would save new sets created by the user when they go online or add the set to the control box. This would prevent the accidental loss of the set.

4. Currently the procedure that checks whether a new set has been added to the control box correctly only looks at the first two bytes of the set, the state number, to see if it is what is expected. While this was sufficient in the old version, it is not a good enough test in the new version. This was mainly because of the ability to overwrite sets as the new set would have the same state number as the old one. To rectify this, a complete check of the entire set would have to be made. This would include saving the set being added in a temporary stm file so that it could be compared with the control box.
5. The control box could be made to bleep during procedures such as adding sets, deleting sets and initialising. This would reassure the user that communication is occurring.
6. Currently, if the user wants help with a particular procedure or control they have to load the help file and search manually. It would be easier if they could access the help topics directly from the control or menu item they want help on. This could be done by having right click menus on all the controls that link to the corresponding help topic.
7. There could be a function where the user can add channel settings for a new patient. This would be far better than the current situation which requires the new settings to be hard coded by a programmer.
8. The “busy, please wait” messages could be made more informative with information on what LPCI is actually doing. There could also be a progress bar showing how much time is left. Although care will have to be taken to ensure this is a true estimate of time left rather than the unreliable estimates often seen in modern applications.
9. Instead of defining cycling phases as fixed angles each phase could have a start and stop angle. This would make the cycling sets much more customisable and would optimise the cycling action.
10. The code could be updated to be compatible with a newer, easier to use compiler such as Borland C++ Builder.

11. Instead of connecting the box to a PC to update the parameters the system could be made all-in-one. So the control box would have a small screen displaying the parameters of the selected set, buttons on the control box would then allow the user to alter the parameters without having to connect it to a PC. This type of device has been manufactured for a different stimulator system made by a German company, Medel [5].

## 8 References

- [1] Brindley G S, Polkey C E and Rushton D N 1982 Sacral Anterior Root Stimulators for Bladder Control in Paraplegia *Paraplegia* **20** 365-381
- [2] Creasey G H and Dahlberg J E 2001 Economic Consequences of an Implanted Neuroprosthesis for Bladder and Bowel Management *Arch. Phys. Med. Rehabil.* **82** 1520-1525
- [3] Donaldson N de N, Rushton D and Tromans T 1997 Neuroprosthesis for leg function after spinal-cord injury *Lancet* **350** 711
- [4] Menter R, Weitzenkamp D, Cooper D, Bingley J, Charlifue S and Whiteneck G 1997 Bowel management outcomes in individuals with long-term spinal cord injuries *Spinal Cord* **35** 608-612
- [5] Negård N -O, Schauer T, de Gersigny J, Hesse S and Raisch J 2005 Application Programming Interface and PC control for the 8 channel stimulator MOTIONSTIM8. *Proc. of the 10th Annual Conference of the International Functional Electrical Stimulation Society (IFESS), Montreal, Canada*
- [6] Perkins T A, Donaldson N de N, Hatcher N A C, Swain I D and Wood D E 2002 Control of leg powered paraplegic cycling using stimulation of the lumbo-sacral anterior spinal nerve roots. *IEEE Trans. Neural Sys. Rehab. Eng.* **10**(3) 158-164
- [7] Rushton D N 1997 Functional Electrical Stimulation *Physiol. Meas.* **18**(4) 241-275
- [8] Vastenholt J M, Snoek G J, Buschman H P J, van der Aa H E, Alleman E R J and Ijzerman M J 2003 A 7-year follow-up of sacral anterior root stimulation for bladder control in patients with a spinal cord injury: quality of life and users' experiences *Spinal Cord* **41** 397-402

## 9 Appendix

### 9.1 An example of an stm file

```
#####
#
#      Bladder Pattern          Version 1.00      #
#      For AA                   #
#      Created on 08/07/05 by sph #
#      Last Updated on 01/08/05 by sph #
#
#####
```

#### Pre-Fatigue Phase

```
Frequency 40Hz
Ramp      2000ms
Hold      6000ms
Bleep     300ms x3
S2        20%
S34       10%
Level     Low
```

#### ON Phase

```
Frequency 20Hz
Ramp      2000ms
Hold      6000ms
Bleep     1000ms x1
S2        80%
S34       75%
Level     Medium
```

#### OFF Phase

```
Frequency 20Hz
Ramp      2000ms
Hold      6000ms
Bleep     0ms x3
S2        0%
S34       0%
Level     Low
```

Description of Set:  
This is a bladder set.