DISSERTATION

Functional Electrical Stimulation of Human Denervated Muscle: Technical Equipment and Patient Study

ausgeführt zum Zwecke der Erlangung des akademischen Grades
eines Doktors der technischen Wissenschaften unter der Leitung von

Ao. Univ. Prof. Dipl.-Ing. DDDr. Frank Rattay

eingereicht an der Technischen Universität Wien
bei der Fakultät für Elektrotechnik

von

Dipl.-Ing. Christian Hofer
Matrikelnummer: 8825083
Schanzstraße 18/1501
1150 Wien

Wien, im Mai 2008

Für Uschi, Fabian, Amelie und meine Eltern!

Ich bedanke mich bei Prof. Dipl.-Ing. DDDr. Frank Rattay und Prof. Dipl.-Ing. Dr. Winfried Mayr für die konstruktiven Anregungen und die Unterstützung bei der Erstellung sowie für die Begutachtung meiner Arbeit.

Ebenfalls bin ich Doz. Dr. Dr. Helmut Kern und seinem Ärzteteam für die Durchführung der klinischen Studie und die Betreuung der Patienten dankbar. Speziell erwähnen möchte ich in diesem Zusammenhang Dr. Claudia Forstner, die bei der Entwicklung und klinischen Anwendung der elektrophysiologischen Messmethode sehr hilfreich war.

Vielen Dank auch an Ing. Stefan Löfler und Mag. Philipp Drewniak für die Unterstützung bei der Erfassung und Aufbereitung der Messdaten.

Abbreviations

ACh	acetylcholine
AP	action potential
CAT	computed axial tomography
CT	computed tomography
DC	direct current
DDM	denervated degenerated muscle
DICOM	digital imaging and communications in medicine
EMG	electromyography
ES	electrical stimulation
FES	functional electrical stimulation
HU	Hounsfield unit
ISI	inter-stimulus interval
MFCV	muscle fibre conduction velocity
MFRP	muscle fibre refractory period
MU	motor unit
MUAP	motor unit action potential
NiMH	nickel-metal hydride
PDA	personal digital assistant
PC	personal computer
pixel	picture element
ROI	region of interest
SCI	spinal cord injury
SEM	standard error of the mean
SNEMG	stimulation needle electromyography
SR	sarcoplasmic reticulum
T system	transverse tubule system
T tubule system	transverse tubule system
TTX	tetrodotoxin
voxel	volume element

Kurzfassung

In Europa erleiden jährlich ungefähr 20 Personen pro Million Einwohner eine Querschnittslähmung. Ein Viertel der Verletzungen liegt im Bereich Conus Cauda oder Cauda Equina und führt zu Denervation und schlaffer Lähmung der unteren Extremitäten. Als Folge treten extreme Atrophie und später Degeneration bei gleichzeitiger Vermehrung von Fett- und Bindegewebe im und um den Muskel auf. Die Funktionelle Elektrostimulation (FES) ist derzeit die einzige Möglichkeit, die degenerativen Veränderungen in der denervierten Muskulatur zu verzögern oder sogar rückgängig zu machen.

Da die technische Ausrüstung für die FES denervierter Muskulatur bisher nicht als Medizinprodukt für die Therapie verfügbar ist – nicht zuletzt aufgrund einschränkender Vorgaben der EU-Normen für Reizstromgeräte - war ein wichtiges Ziel der gegenständlichen Arbeit die Entwicklung eines sicheren für eigenverantwortliche Heimanwendung geeigneten Stimulationsgerätes für Patienten mit schlaffer Querschnittlähmung.

Neue nicht- bzw. minimal invasive Methoden, welche den Muskel einerseits hinsichtlich seiner morphologischen Zusammensetzung und andererseits seiner elektrophysiologischen Eigenschaften beschreiben, waren ebenfalls Entwicklungsziele im Rahmen dieser Arbeit und erlauben eine quantitative Erfassung des aktuellen Abbau- bzw. Trainingszustandes der denervierten Muskulatur. Eine dieser Methoden basiert auf der erweiterten Auswertung von CT-Querschnittbildern des Oberschenkels, eine zweite auf einer auf Nadelelektroden basierenden Technik, die Reizweiterleitung (M-Welle) einzelner Muskelfasern bzw. -fasergruppen zu vermessen.

Im Rahmen der klinischen Studie des europäischen Forschungsprojektes „RISE" wurden die neu entwickelten Geräte und Methoden an Patienten erprobt. Nach zwei Jahren FES-Training zeigten die Patienten abhängig von der individuellen Denervationszeit vor Beginn der Studie verschiedengradige Verbesserungen hinsichtlich Muskelquerschnittsfläche und –gewebe-verteilung sowie Leitgeschwindigkeit und Refraktärverhalten der Muskel-fasern. Die Ergebnisse korrelieren mit anderen klinischen Untersuchungen und Auswertungen von Muskelbiopsien.

Die vorliegende Arbeit liefert den Nachweis, dass denervierte Muskulatur mittels Anwendung entsprechender Stimulationsparameter und Protokolle trainierbar ist und erhalten werden kann. Damit können in der betroffenen Patientengruppe häufig auftretende Sekundärerkrankungen mit schwer-wiegenden Folgen wie Druckgeschwüre weitgehend verhindert werden.

Abstract

In Europe the rate of occurrence of spinal cord injury is approximately 20 per million citizens. About 25% of the patients have an injury in conus cauda or cauda equine which leads to denervation and flaccid paralysis of their lower extremities. This causes a severe loss of muscle mass (atrophy) and later degeneration accompanied by an increase in fat and connective tissue within and around the muscle. Functional electrical stimulation (FES) is currently the only way to slow down or even reverse the degenerative changes of denervated muscle.

Since technical equipment for FES of denervated muscle as a medical product is not available for therapy – not at least due to limitations by actual EU standards for stimulation equipment – one of the goals of this work was development of a suitable and safe stimulator for home based training.

Novel non- or minimal-invasive methods for assessment of muscle morphology and fibre specific electrophysiology were developed allowing a quantitative description of the muscle's tissue composition and electrophysiological properties. One method is based on CT scans of the patients thigh, a second is a special electrode needle based M-wave recording technique allowing to measure conduction velocity and refractory behaviour of single or small groups of muscle fibres.

Within the scope of a clinical study carried out in the context of EU research project "RISE" the developed equipment and methods were applied to patients. ES therapy with the developed stimulator was conducted over a period of two years. Depending on their denervation period before start of therapy the patients showed individual amounts of improvements of their muscle cross-sectional area and tissue composition and of their fibre conduction properties. The findings are in agreement with results from other clinical tests and biopsy analyses performed in the study.

The study showed the feasibility of training and maintenance of denervated muscles by application of long-duration impulses and appropriate protocols and a clear benefit in avoidance of secondary diseases like pressure sores - a frequent problem with severe consequences - in the target patient group.

1 Introduction

Electrical stimulation (ES) is a method where a certain amount of current is applied to electrically excitable tissue like nerves and muscles in order to activate these structures in most cases. The first known, documented application of electrical stimulation for therapy dates back to approximately 400 years before Christ. Hippocrates treated asthma and various other diseases by means of electrical shocks produced by torpedo fish. In the 18th and 19th century a multitude of electrical stimulation devices for therapeutic application were developed. Most of them were based on electrostatic generators (Mc Neal 1977).

Functional electrical stimulation (FES) is the application of electrical stimulation to restore or improve lost or impaired function of the body due to neurological disorders caused by spinal cord injury, stroke, traumatic brain injury or neurological disease. The key experiment for functional electrical stimulation was the frog's leg experiment described by Luigi Galvani 1791 (Mc Neal 1977). He connected nerve and muscle of a frog's leg with an arc consisting of two different metals. The contact voltage elicited by this setup depolarised the contacted nerve and consequentially caused a contraction of the muscles. With this experiment Galvani could show the connection between electrical stimulation of the nerve and the contraction of muscles innervated by it.

Clinical applications of this principle were always closely connected to the technological progress. The first functional application of electrical stimulation is the pacemaker for the heart. Paul M. Zoll was able to maintain heart action by a pacemaker with surface electrodes for the first time in 1952 (Zoll 1952). Furman and Schwedel treated a patient 96 days with a pacemaker in 1958 (1959). Because of the size of this pacemaker, it had to be carried on a trolley in front of the patient. The stimuli were applied trough percutaneous electrodes. A few months later the first fully implantable pacemaker was introduced. Nowadays far more than 500.000 patients per year are treated with cardiac pacemakers (Møller et al. 2002).

Other applications of functional electrical stimulation are restoring the function of the respiratory system (Judson and Glenn 1968, Talonen et al. 1990, Mayr et al. 1993), bladder and bowel (Ingersoll 1957, Brindley 1977, 1986), lower and upper extremity (Liberson 1962, Peckham et al. 1981, Marsolais et al. 1983, Holle et al. 1984, Kern et al. 1985, Keith et al. 1988),

the cochlear (Brackmann 1976, Banfai 1978, Hochmair et al. 1979), and the retina (Greenberg 1998, Uhlig et al. 2001).

All above mentioned applications, with the exception of the cardiac pacemaker, are based on the excitation of neural structures. The denervated muscle, however, has to be activated directly by electrical stimulation. That means that due to the absence of the nerve and thus the missing neuromuscular junction the cellular membrane of each muscle fibre has to be depolarised to elicit a contraction. This direct stimulation requires more energy and it is more difficult to achieve a contraction of the whole muscle since there is no branching nerve distributing the action potential to multiple parts of the muscle.

The use of electrical stimulation for denervated muscles is still considered to be controversial by many rehabilitation facilities and medical professionals. This is largely because current teaching and training for therapists is still based on the scientific and technological knowledge of the fifties and sixties. The literature contains a limited number of animal studies investigating the effects of long-term stimulation on denervated muscles, especially in the rat and rabbit. These have been designed mainly to address basic scientific issues (Mokrusch et al. 1990, Gunderson 1998). In virtually every case the muscles were denervated only for a short time before stimulation is commenced, and the results therefore have limited relevance to the human condition, where long-standing denervation has already resulted in severe atrophy and degeneration of muscle fibres. Other experimental work was mainly focused on strategies for reinnervation.

A clinical study on denervated muscles has, however, shown that degeneration after denervation could be at least slowed down, even with less than optimal equipment (Eichhorn et al. 1984).

An animal study demonstrated that FES delivered via implanted electrodes could maintain the action of the denervated sheep cricoarytenoid muscle for up to 18 months (Carraro et al. 1988, Zrunek et al. 1991). So far only two functional clinical studies have been published. One group from Slovenia has demonstrated correction of dropped foot by FES of the denervated tibialis anterior muscle (Valencic et al. 1986). The other study showed that the problems of muscle training and restoration of basic lower extremity movements in cases of flaccid paraplegia could be solved in principle, but also highlighted some of the problems that must be overcome before the technique could be introduced in clinical practice (Kern et al. 1999, 2002).

The target group for this study are spinal cord injured patients with denervated muscles. In Europe the incidence of spinal cord injury is approximately 20 per million inhabitants per year (Wyndaele and Wyndaele 2006) and about 25 % of the SCI patients have an injury also affecting the conus cauda or cauda equine leading to partial or full denervation of their lower extremities.

1.1 Aim of the study

- Design of appropriate stimulation equipment (stimulators).
- Development of novel assessment methods and the associated devices for evaluating the stimulation induced changes in the denervated muscles.
- Testing of the new developed equipment and methods within a clinical study.

2 The denervated muscle

2.1 Anatomy, Physiology (neuromuscular system)

In the human body two types of electrically excitable cells can be found: the nerve cell and the muscle cell. The excitability of these cells is the basis for the application of functional electrical stimulation. This chapter will give an overview about the anatomical and physiological fundamentals.

Structure and function of the nerve cell

The basic unit of nervous tissue is the nerve cell. The human nervous system consists of more than 10^{10} nerve cells (Silbernagl and Despopolous 1991). Typically a neuron consists of the soma (cell body), the axon and the dendrites which are emerging from the soma. In general the axon originating from the axon hillock of the soma transmits the action potential to other nerve or muscle cells. Axons are terminated by synapses which are connected to the soma or dendrite of a following neuron or to muscle cells (Figure 2.1).

In many neurons the axon is isolated by concentric layers of the Schwann's cells building up the myelin sheath. This myelin sheath is interrupted at certain distances by Ranvier's nodes. The so called myelinated nerve fibres with a rather high conduction velocity of 3 – 120 m/s are opposed by non-myelinated fibres with a conduction velocity of 0.5 – 2 m/s.

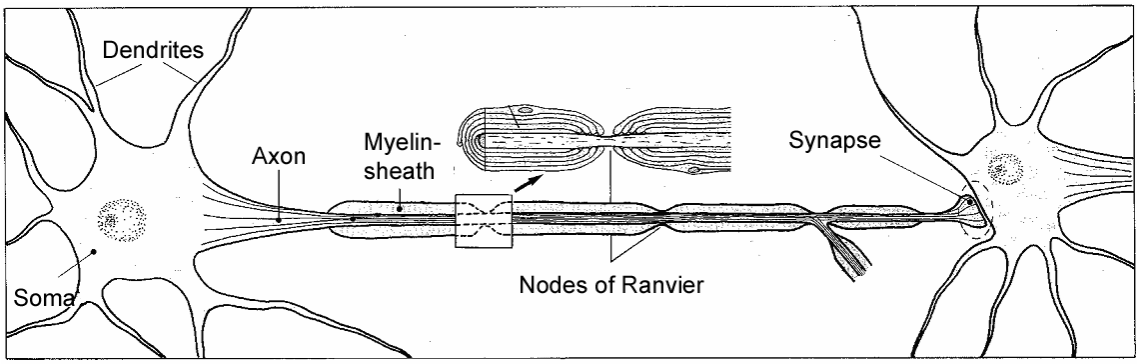


Figure 2.1: Composition of a nerve cell (adapted from Silbernagl and Despopolous 1991)

The nerve action potential is the fundamental unit used by the nervous system for transmitting information, it has a similar shape in all nerve fibres, with duration of about 1 ms and amplitude of approximately 100 mV.

Electrical excitability of the nerve membrane is closely related to the voltage dependent permeability of the membrane. Active ion transport through the cell membrane maintains a precise internal concentration of several important ions, including Na^+, K^+, Cl^- and Ca^{2+} (Figure 2.2).

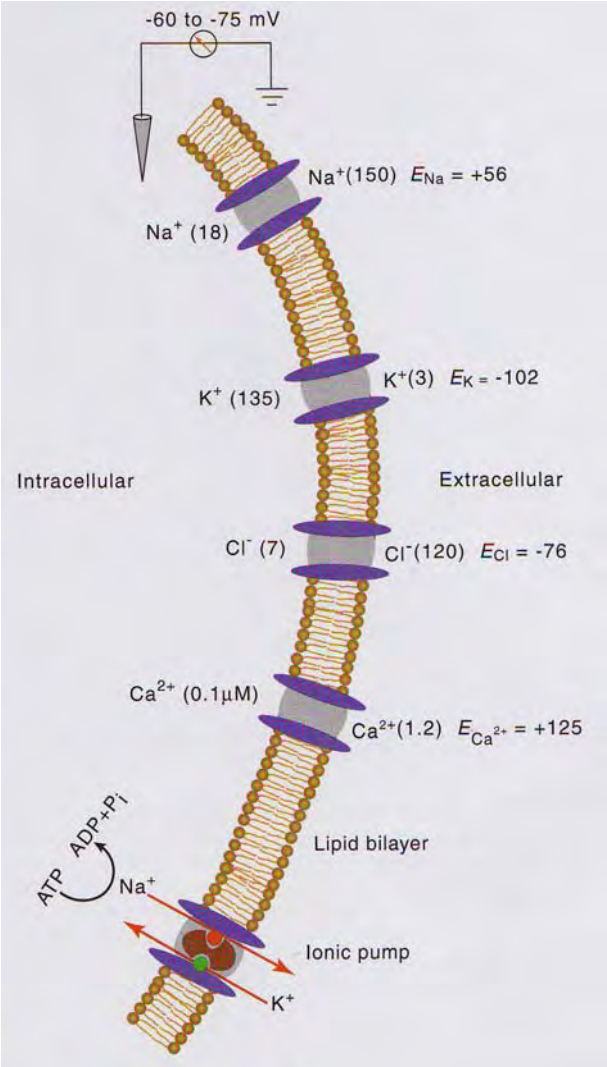


Figure 2.2: Resting membrane potential caused by differential distribution of ions inside and outside the cell membrane. Concentrations (in millimol/L except that for intracellular Ca^{2+}) of the ions are given in parentheses; their equilibrium potentials (E) for a typical mammalian neuron are indicated. (from McCormick 1999).

There is a big difference of the intracellular and extracellular concentrations of Na^+, K^+, Cl^- and Ca^{2+}. The ions have a tendency to diffuse according to their difference in concentration through the cell membrane but at rest the permeability of the membrane is very limited for these ions. This limited permeability and the operation of ionic pumps (e.g. Na^+ - K^+ ionic pump) are maintaining the difference in concentration of ions inside and outside the cell. Due to this unequal intra- and extracellular distribution of the ions a resting membrane potential of 70 – 90 mV is built up. The intracellular space carries a negative charge with respect to the extracellular space.

By application of an electrical stimulus to a nerve cell the voltage across the membrane changes and affects the ion conductance of voltage dependent ion channels within its membrane. If this stimulus exceeds a certain threshold, the ion current across the membrane rises suddenly and an action potential is generated. In a motor neuron this action potential propagates along the axon and finally causes a muscle twitch.

At the site of excitation following happens: The stimulus increases the membrane potential form about -70 mV towards 0 mV. When the threshold voltage is reached (typically –55 mV), the Na^+-channels of the membrane are activated, causing a transient increase of the membrane sodium conductance. Due to the influx of sodium into the cell the membrane potential is breaking down – depolarisation phase. This depolarisation activates the K^+-channels increasing the potassium conductance of the membrane whereas the sodium conductance already decreases and the membrane potential repolarises – repolarisation phase. The persisting activation of the K^+-channels (and other membrane properties) causes the membrane potential to fall below the resting membrane potential – hyperpolarisation phase (Figure 2.3).

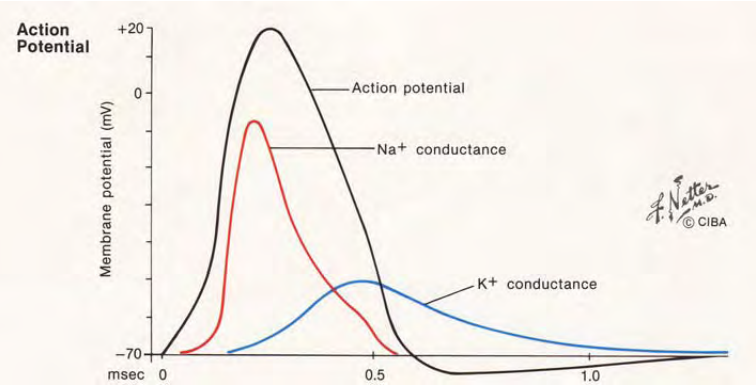


Figure 2.3: The generation of an action potential (from Netter 1983)

A stimulus eliciting an action potential does not have to be of electrical nature, it could be also a chemical, thermal or mechanical one. To cause a depolarisation certain conditions have to be met. The intensity has to be high enough and last long enough to reach the threshold level. This level depends on the tissue stimulated. Nerve cells can be activated with stimuli of much lower intensity than muscle fibres. Geometric factors are also influencing the threshold level. Nerve fibres with different diameter are showing different thresholds. The relation between intensity and duration of a sufficient stimulus is displayed by the strength-duration curve (Figure 2.4). This curve shows how long a current with given amplitude has to flow to elicit an action potential. The maximum current intensity not able to trigger an action potential even at infinite duration is called rheobase. The minimum pulse duration sufficient to depolarise the nerve or muscle fibre when the stimulus amplitude is twice the rheobase is called chronaxie.

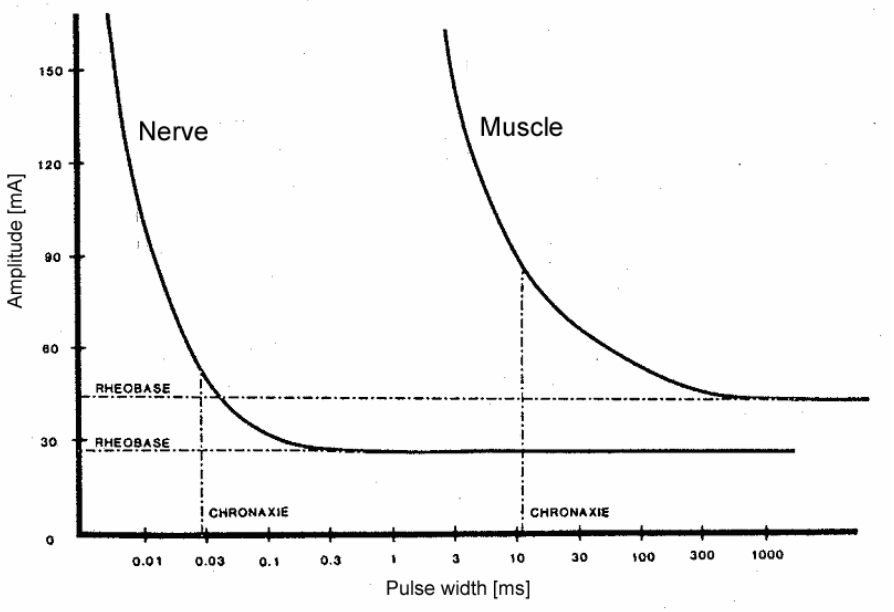


Figure 2.4: strength-duration curve (adapted from Benton 1981)

The shape of the action potential itself is independent from duration and intensity of the stimulus triggering it.

The structure of the skeletal muscle

The skeletal muscle cell has a diameter of 10 – 100 µm and a length of up to 20 cm. These muscle cells or fibres are bundled together to fibre bundles called the fascicles which are covered by the perimysium. The muscle on the other hand consists of a bundle of the fascicles covered by the epimysium. Contrary to most tissues in the human body the muscle fibre does not consist of a large number of cells. Instead the cell membranes have fused together to produce one large cell or syncytium containing multiple nuclei. The cell membrane of the muscle fibre is called the sarcolemma and the contents are called sarcoplasm. Within the fibre individual contractile units are called myofibrils which are divided into 2 µm long sarcomeres (Figure 2.5).

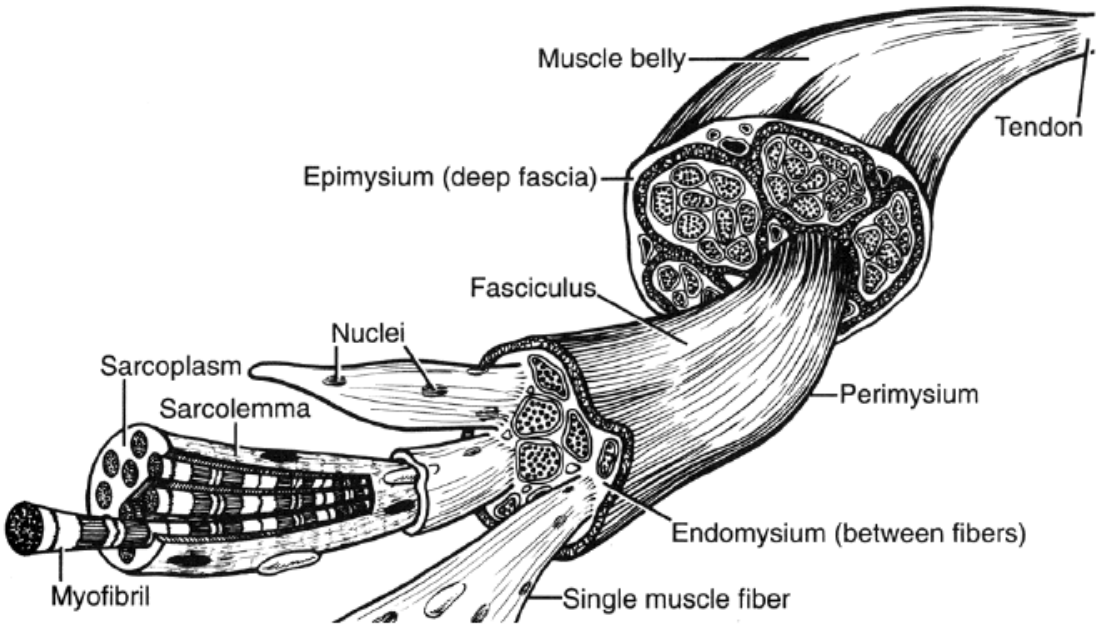


Figure 2.5: Composition of skeletal muscle. (McGinnis 1999)

The smallest functional unit of the muscle is the motor unit (MU), a combination of a single motor neuron and the muscle fibres innervated by it. According to the size and function of the muscle the number of muscle fibres building a motor unit is varying. Large muscles with high force output like the gastrocnemius muscle have MU's with a few hundred or thousand

muscle fibres. Whereas muscles for precise movements have MU's composed of very few fibres e.g. inferior and superior oblique muscles for control of eye movement have 2 – 3 fibres per motor unit.

The connection between the nerve and the muscle is called the motor end plate. It is effectively a synapse with acetylcholine acting as neurotransmitter (Figure 2.6). An action potential coming from the axon causes a release of acetylcholine from the nerve endings into the synaptic gap. The acetylcholine diffuses through the gap and activates specific ion channels on the sarcolemma. This causes a small change in the resting potential just sufficient to trigger the voltage gated sodium channels and to start the required action potential.

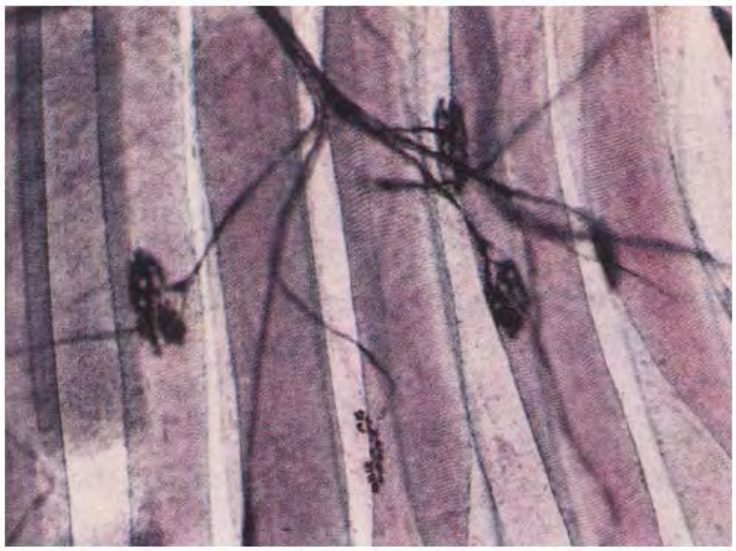


Figure 2.6: Motor end plates (from Wheater et al. 1979)

The action potential propagates along the sarcolemma and is distributed by the transverse tubule system (T tubule system) into the muscle fibre. T tubules come into close contact with the sarcoplasmic reticulum (SR), a form of the endoplasmic reticulum specialised for the release and reuptake of calcium in muscle fibres. The junction between the T tubules and the SR, called triads, are essential in the coupling between muscle excitation and contraction (Figure 2.7).

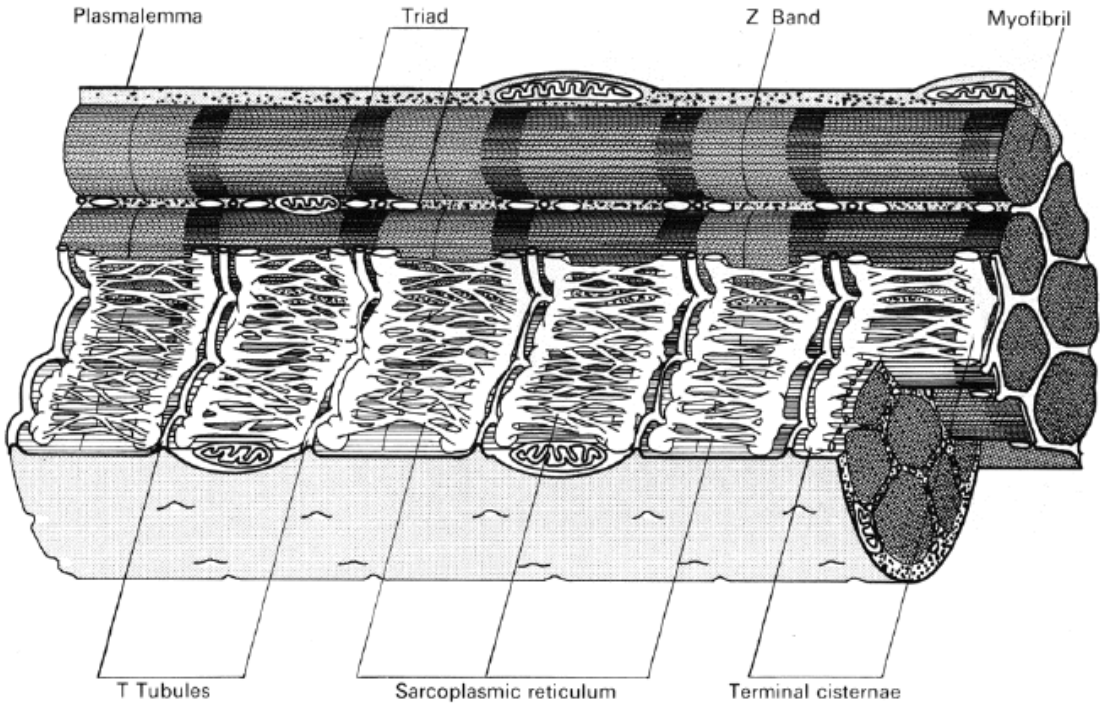


Figure 2.7: Structure of a skeletal muscle fibre (from Wheater et al. 1979)

The action potential transmitted into the fibre by the transverse tubule system causes a release of Ca^{2+} ions from the SR into the sarcoplasm leading to a contraction of the muscle fibre.

To a stimulus, with a strength exceeding the threshold for eliciting an AP, the muscle fibre always reacts with a contraction. Force generation of the muscle is controlled by two other parameters, the frequency of the stimuli and the number of motor units activated.

A single impulse elicits a muscle twitch, with increasing frequency the single muscle twitches are overlapping increasing the muscle tension. Above the twitch fusion frequency the muscle generates a steady level of force, or tetanus (Figure 2.8).

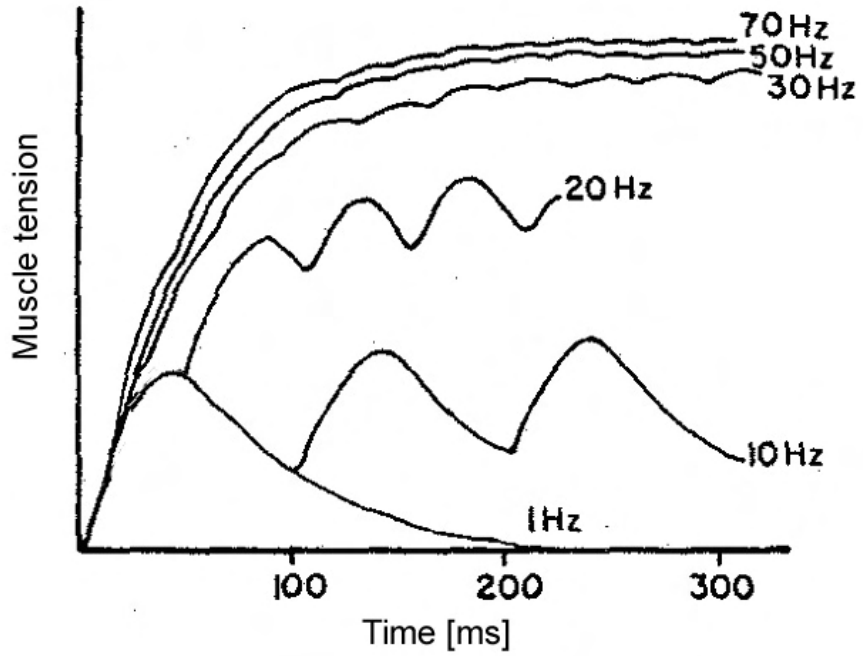


Figure 2.8: Superposition of muscle twitches with increasing stimulation frequency (from Benton et al. 1981)

The other way to control muscle force is to vary the number of simultaneously activated motor units. During a voluntary physiological contraction the motor neurons activated change continuously and also the frequency of the stimuli is varying. The advantage of this mechanism is that the muscle fatigues slower because the load is distributed over the whole muscle.

In contrast to this mechanism all motor neurons are firing simultaneously when excited by electrical stimuli. Additionally always the same area of the muscle is activated. This leads to faster fatigue of the muscle compared to normal voluntary activation. Further differences are caused by the recruitment order between physiologically and artificially excited muscles. Under physiological condition for generating low levels of force first thinner motor neurons are elicited, which are connected to slow fatigue resistant muscle fibres – type I fibres. If higher forces are demanded also the larger, faster motor units – type II fibres – are activated. Electrical stimulation activates the muscle in reverse order due to geometrical reasons. Thick fibres of motor neurons activating fast muscle fibres are more sensitive to an electrical stimulus than thinner fibres. That is why electrical stimulation causes faster muscle fatigue. In long-term stimulated muscles this reversed

recruitment induces a transformation of muscle fibre phenotype from the fast and fatigable glycolytic type II fibres to the slow and fatigue resistant aerobic type I fibres.

2.2 Changes induced by denervation

Denervation denotes the state that a muscle is deprived of the nerve normally activating it. The nerve influences the muscle by two different mechanisms: (1) by activating the muscle via electrical impulses (action potentials) causing the muscle to contract – neuromotor control and (2) potentially by delivering chemical substances via the axonal flow to the muscle – neurotrophic factors. The absence of these two different mechanisms is responsible for the changes in the muscle after denervation.

In literature the importance of the neuromotor control and neurotrophic factors is heavily discussed. At the beginning of studying the denervated muscle researchers were convinced that neurotrophic factors have a big influence on muscle properties but until today there are only a few studies providing evidence for changes in the denervated muscle caused by the missing neurotrophic factors of the nerve (Ramirez 1984, Behrens and Vergara 1992, Vergara et al. 1993, Ramirez et al. 1996 and 2003, Huang et al. 2002, Roy et al. 2002, Talon et al. 2005, Hyatt et al. 2006).

On the other hand almost all of the alterations of the muscle caused by short-term denervation can be reversed by electrical stimulation of the muscle as shown in animal experiments (Hník et al. 1962, Salmons and Vrbová 1969, Lømo and Westgaard 1975, Pette and Vrbová 1992, Awad et al. 2001, Salmons et al. 2005, Ashley et al. 2007). This indicates that activity is the major factor influencing muscle properties.

Causes for denervation can be of different nature. Denervation may be due to a disease where the death of motor neurons causes the denervation of muscle fibres (e.g. poliomyelitis), due to a chemical substance blocking the synaptic transmission of the action potential (such as botulinum toxin) or physical injury damaging peripheral nerves (as by accident).

This work is focused on spinal cord injured subjects with lesion in the conus cauda or the cauda equine. Depending on the completeness of the lesion this leads to full or partial denervation of both lower limbs. Contrary to paraplegic patients with spastic paralysis where the peripheral nerve (motor neuron) is preserved these patients do not show any muscle activity –

flaccid paralysis. The anatomical level of the spinal cord injury determines if the paralysed muscles are denervated or innervated (Figure 2.9).

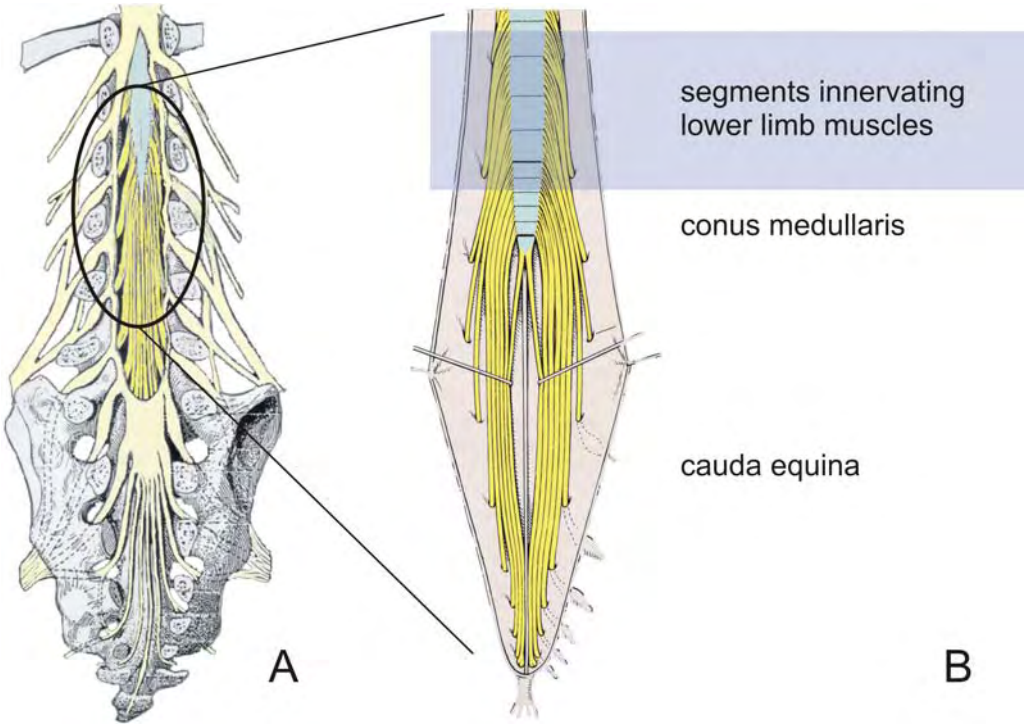


Figure 2.9: The spinal cord within the lumbar part of the spinal column. (A) The spinal cord (blue) extends only to L1 – L2 vertebral level, but the nerve rootlets (yellow) originating from the corresponding segments continue down to the appropriate vertebral column exit point. (B) Grey shaded are the spinal cord segments innervating the lower limb muscles. The tapering lower part at the terminal end of the spinal cord at the vertebral level of the first or second lumbar segment is the conus medullaris. After the spinal cord terminates, the spinal nerve rootlets continue forming the cauda equina. Lesions at the level of the conus medullaris or the cauda equina are leading to full or partial denervation of the lower limb muscles. (adapted from Duus 2001 p. 82 (B))

2.2.1 Morphological, structural changes

A severe loss of muscle mass (atrophy) represents the first obvious change after denervation. One of the first clinical observations reporting this were made by Oppenheim (1894). He described that the muscle fibres appeared smaller in diameter and their content disintegrated into granulated and lipid particles, which were resorbed, so that only sarcolemmal tubes containing the nuclei were left. The first experimental studies investigating denervation induced changes after section of the nerve were made at the end of the nineteenth and the beginning of the twentieth century as described by Gutmann and Zelená (1962). These studies were mostly restricted to short

term denervation periods. Sarah Tower (1935) studied in the cat the course of atrophy in denervated muscles for up to one year. She distinguished three stages of denervation induced muscle damage: atrophy, degeneration and fibrotic dedifferentiation.

Initial research on denervation atrophy in human muscle was carried out mostly in poliomyelitis or other forms of locomotor diseases like amyotrophies, progressive muscle atrophy, amyotrophic lateral sclerosis, etc. (Darkewitsch 1905, Kopits 1929, Adams et al. 1954). The drawback of these studies was that the onset and the extent of the denervation were seldom known. More accurate information concerning the changes in denervated muscle could be obtained in peripheral nerve lesions. In these cases the time of lesion and the nerves and muscles involved were usually known. Following nerve trauma the muscle fibre diameter is considerably reduced within the first year, some cases showing a great variation in diameter. Degenerative changes were not observed until three years after nerve interruption (Bowden and Gutmann 1944). At final stages (up to 25 years) of denervation the muscle is substituted by fibrous and adipose tissue. The findings, obtained from biopsies of denervated muscle, were also confirmed by recent studies (Kern et al. 2004, Boncompagni et al. 2007) see Figure 2.10.

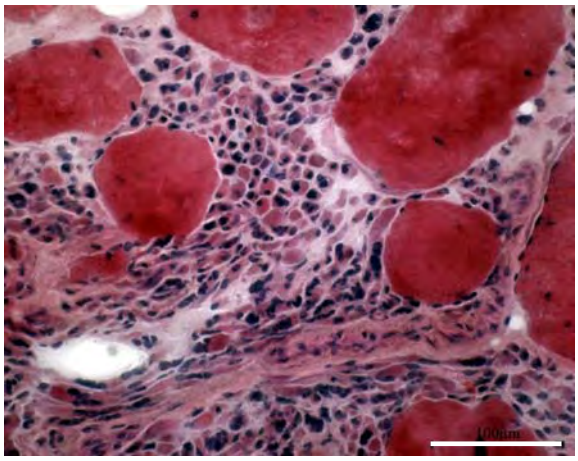


Figure 2.10: Cryosection stained with haematoxylin and eosin of a biopsy from vastus lateralis muscle of a subject 7.5 years denervated. Showing preserved large fibres surrounded by severely atrophic fibres. Bar: 100 µm (Biopsy harvested in the course of the RISE study, prepared by U. Carraro)

Alternative non-invasive methods to demonstrate changes in the size or structure of denervated muscle are measuring the thickness of the muscle

belly by ultrasound (Taylor 1993) or to determine muscle cross-sectional area by computed tomography (Termote 1980, Kern 1995). In the example of a computed tomography scan from the thigh cross-section of a denervated thigh shown below (Figure 2.11) the decrease of muscle is evident. The remaining muscle is surrounded by a thick layer of subcutaneous fat and also within the muscle the amount of non contractile tissue (intramuscular fat, connective tissue and collagen) is increased.

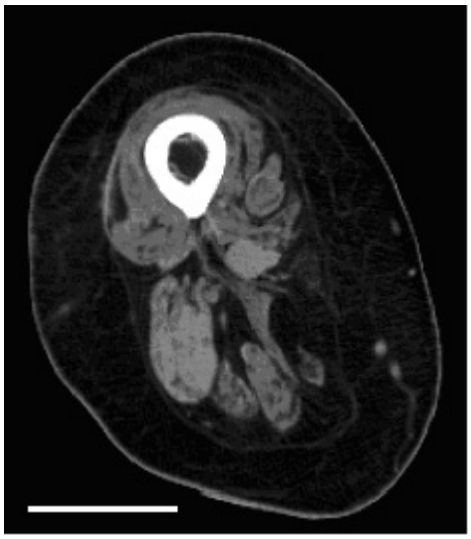


Figure 2.11: CT scan showing a cross-section of the right thigh 20cm below the top of the greater trochanter of a patient denervated for 8.7 years. Bar: 5 cm.

Changes in fibre diameter are more reliable than muscle weight, area or other macroscopic parameters since fibre size is not distorted by connective tissue, fat and collagen.

2.3 Electrophysiological changes

The reduction of resting membrane potential of the muscle fibre is the earliest sign of muscle denervation. Albuquerque and Thesleff (1968) found in the rat that 7-10 days after denervation the resting membrane potential was reduced by 10-15 mV. They also noticed an increase of threshold for generation of an action potential. This is in accordance with results of other research groups showing an increase of chronaxie in denervated muscles (Robert and Oester 1970, Midrio et al. 1997, Ashley et al. 2005), confirming a lower excitability.

Spontaneous activity is another characteristic electrophysiological feature of the denervated muscle. There are two different types of activity fibrillation potentials and positive sharp waves. A fibrillation potential which is the spontaneous depolarisation of a single muscle fibre has a duration less than 5 ms and a bi- or triphasic shape depending on the recording site (Figure 2.12 A) . If the needle is placed in the endplate region of the muscle the potential will be biphasic. When recorded outside the endplate region the potential will have a triphasic shape (Buchthal and Rosenfalck 1966). The positive sharp wave has a characteristic shape with an initial positive[1] deflection of short duration followed by a rather slow return, often with a negative phase of low amplitude, to baseline (Figure 2.12 B). The duration of the positive sharp wave ranges from several ms up to 100ms. Typical the spontaneous activity arises within one to three weeks after denervation of the muscle, concurrent with the reduction in membrane resting potential (Buchthal 1982).

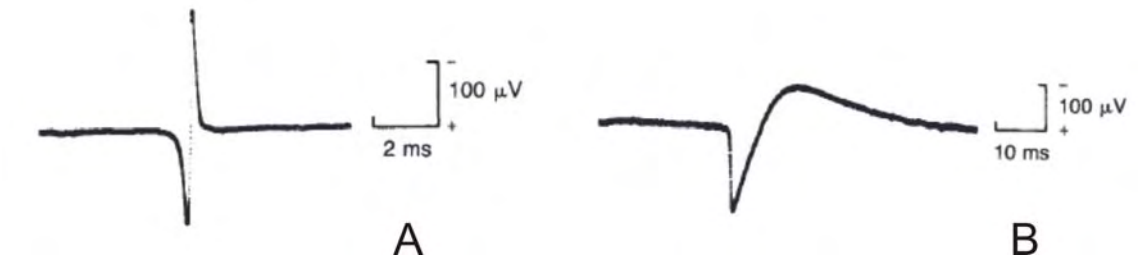


Figure 2.12: Fibrillation potential and positive sharp wave recorded with an intramuscular needle electrode. (A) Waveform of a single fibrillation potential which is the electric activity associated with a spontaneously contracting (fibrillating) muscle fibre. The potentials are bi- or triphasic spikes of short duration (less than 5 ms) with an initial positive phase and a peak-to-peak amplitude of less than 1 mV. (B) Positive sharp wave characterised by an initial positive deflection (<1 ms), its duration is usually less than 5 ms, and the amplitude is up to 1 mV. The negative phase is of low amplitude, with a duration of 10 to 100 ms. (adapted from AAEM Nomenclature Committee 2001)

[1] Note: The membrane potential is defined as the voltage across the membrane measured as the potential of the cell interior with respect to the extracellular space. Due to the fact, that the needle EMG is the extracellular recording of volume conducted APs of single or multiple muscle fibres, the measured voltage is going to be more negative if the membrane potential becomes more positive.

Other findings are an increase in the sensitivity of the membrane to acetylcholine (ACh) outside of the endplate region (Lømo and Westgaard 1975) and the appearance of a new type of sodium (Na^+) channels which are tetrodotoxin(TTX)-resistant in the membrane (Pappone 1980).

2.3.1 Biomechanical changes

The above described morphological and electrophysiological changes induced by denervation are accompanied by changes in functional properties of the muscle.

In general the denervated muscle exhibits a slower dynamic behaviour than a normal, innervated muscle (Carraro et al. 1982, Dulhunty 1985, Kotsias and Muchnik 1987, Al-Amood and Lewis 1989). The muscle twitch elicited by a single stimulus presents an increase of time to peak and half-relaxation time. Thus the total twitch contraction time is lengthened. This on the other hand reduces the minimum frequency necessary for producing tetanic contractions (fusion frequency). It appears that the reduction in shortening velocity is more pronounced in fast than in slow muscle (Gundersen 1985, Al-Amood and Lewis 1989).

The developed tension of the denervated muscle activated by electrical stimulation is markedly reduced. This reduction can be observed for twitch and for tetanic contractions where this reduction is even more pronounced. Following denervation the reduction in twitch force is progressive with a close relationship between the loss of force and the loss of contractile tissue due to muscle atrophy.

If maximal specific tension developed by the electrically activated muscle is measured the denervated muscle shows no difference or even a slight increase of tension compared to normal innervated muscle (Finol et al. 1981, Midrio et al. 1997, Germinario et al. 2002, Ashley et al. 2007)

3 Stimulation equipment

3.1 Biophysical principles of the extracellular activation of nerve and muscle fibres

The shape of a nerve axon or a muscle fibre is described by a cylindrical tube with the surface representing the fibre membrane separating the intracellular space from the outside. In order to examine the conditions for extracellular activation of such a fibre by electrical stimulation, an equivalent circuit of the fibre membrane is considered (Figure 3.1).

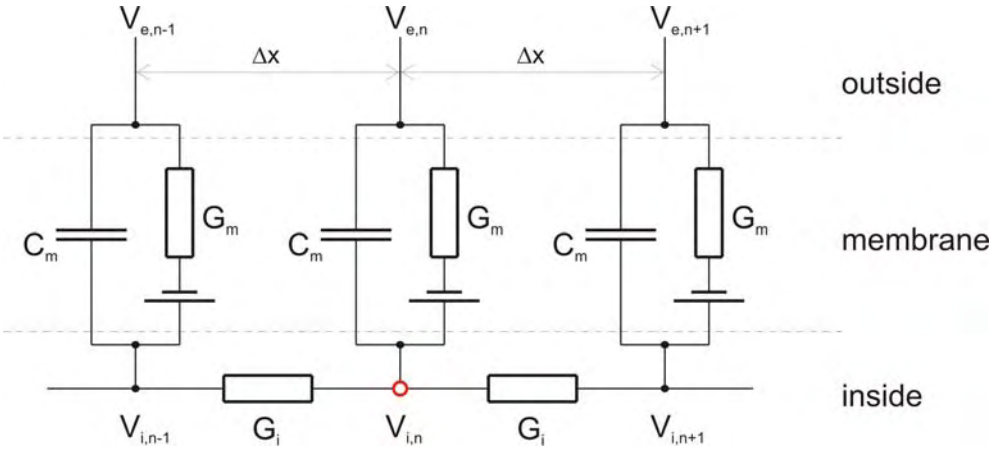


Figure 3.1: Network to simulate the currents in a nerve or muscle fibre. The fibre is segmented into cylinders with a length of Δx. The membrane of each segment is simulated by the highly nonlinear membrane conductance G_m, which depends on the activation state of the ion channels in the membrane, an ionic voltage source and the membrane capacity C_m. G_i is the intracellular conductance of the fibre and $V_{e,n}$ and $V_{i,n}$ are the external and internal potential of the n^{th} segment (after Rattay 1988).

Applying stimulation pulses to a nerve or muscle fibre with electrodes, either surface or intramuscular, the potential in the medium where the fibre is embedded changes in time as well as in spatial distribution. The current flow across the n^{th} segment (red ring in Figure 3.1) of the membrane can be described by:

$$0 = C_m \frac{d(V_{i,n} - V_{e,n})}{dt} + I_{ionic,n} + G_i(V_{i,n} - V_{i,n-1}) + G_i(V_{i,n} - V_{i,n+1}). \quad (1)$$

With the reduced voltage

$$V_n = V_{i,n} - V_{e,n} - V_{rest} \quad (2)$$

and inserting for the intracellular conductance

$$G_i = \frac{\pi d^2}{4\rho_i \Delta x} \quad (3)$$

as well as the membrane capacity

$$C_m = \pi d L c_m \quad (4)$$

equation (1) is transformed to

$$\frac{dV_n}{dt} = \left[\frac{d\Delta x}{4\rho_i L} \left(\frac{V_{n-1} - 2V_n + V_{n+1}}{\Delta x^2} + \frac{V_{e,n-1} - 2V_{e,n} + V_{e,n+1}}{\Delta x^2} \right) - i_{ionic,n} \right] \frac{1}{c_m}. \quad (5)$$

In equation (5) d designates the fibre diameter, ρ_i the intra-cellular resistivity, c_m the capacity of the membrane per area and L the active length of the membrane within the compartment. For an unmyelinated nerve or a muscle fibre $L=\Delta x$. The ionic currents across the fibre membrane are described by further differential equations e.g. the Hodgkin-Huxley equations for the unmyelinated or the Frankenhaeuser-Huxley equations for myelinated fibre (Hodgkin and Huxley 1952, Frankenhaeuser and Huxley 1964). In the case of the muscle fibre the current flow into the transverse tubular system has to be considered too, therefore the Hodgkin-Huxley-type model was extended by a passive element (Adrian and Peachey 1973, Henneberg and Roberge 1997). A detailed analysis of the different models regarding the activation of the denervated muscle fibre and the influence of the t system was carried out by Reichel (1999). For analysing the subthreshold response of the fibre the above introduced model is adequate.

In equation (5) the term

$$f_n(t) = \frac{V_{e,n-1} - 2V_{e,n} + V_{e,n+1}}{\Delta x^2} \tag{6}$$

describes the influence of the stimulation current applied with extracellular electrodes. It is the second difference quotient of the extracellular potential along the fibre. With Δx -> 0 the difference quotient becomes the second derivative of the extracellular potential with respect to x

$$f(x,t) = \frac{\partial^2 V_e(x,t)}{\partial x^2}. \tag{7}$$

$f(x,t)$ is called the activating function which explains the influence of an external applied electrical field on a muscle or nerve fibre. This concept was introduced by Rattay (1986).

For stimulating a muscle fibre the reduced voltage (2) has to become positive that is equivalent to a depolarisation of the fibre. This means that an action potential can be elicited in fibre segments where the activating function f_n is positive, in regions with negative f_n hyperpolarisation is produced (Rattay 1989).

The considerations made above are applicable for fibres which are activated by external current source in central parts of the fibre. In this case the activating function at the beginning or end of the fibre is assumed to be zero $f_n = 0$. If the fibre is stimulated near an ending the electrical network simulating currents in a fibre has to be adapted (Figure 3.2).

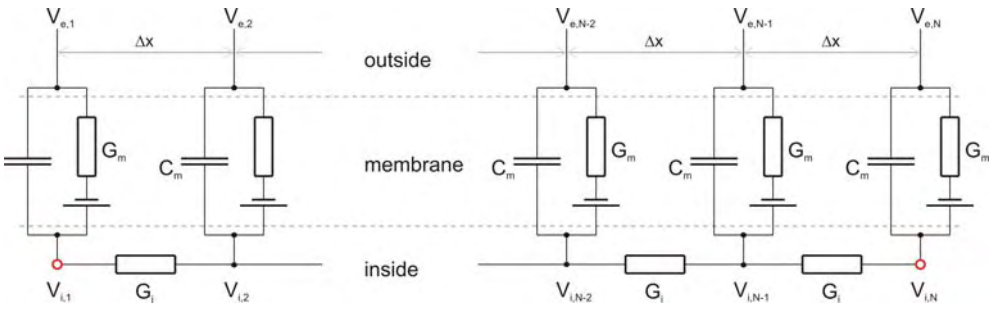


Figure 3.2: Network to simulate the currents at the fibre endings. The first and the last compartments of a fibre with a length of $N.\Delta x$ are shown. $V_{e,1}$, $V_{e,N}$, $V_{i,1}$ and $V_{i,N}$ are the internal and external potential of the first and last fibre segment. G_m is the nonlinear membrane conductance, C_m is the membrane capacity and G_i is the intracellular conductance of the fibre.

The current flow at the fibre endings, node *1* and *N* (red rings in Figure 3.2), is described by

$$\frac{dV_1}{dt} = \left[\frac{d}{4\rho_i \Delta x} \left(\frac{V_2 - V_1}{\Delta x} + \frac{V_{e,2} + V_{e,1}}{\Delta x} \right) - i_{ionic,n} \right] \frac{1}{c_m} \tag{8}$$

$$\frac{dV_N}{dt} = \left[\frac{d}{4\rho_i \Delta x} \left(\frac{V_{N-1} - V_N}{\Delta x} + \frac{V_{e,N-1} - V_{e,N}}{\Delta x} \right) - i_{ionic,n} \right] \frac{1}{c_m} \tag{9}$$

and the terminal activating function for the endings is

$$f_1(t) = \frac{d}{4c_m \Delta x \rho_i} \frac{V_{e,2} - V_{e,1}}{\Delta x} \tag{10}$$

$$f_N(t) = \frac{d}{4c_m \Delta x \rho_i} \frac{V_{e,N-1} - V_{e,N}}{\Delta x} . \tag{11}$$

In contrast to the activating function for central parts of the fibre this function depends from the first difference quotient of the extracellular potential and is weighted by the compartment length *Δx*.

In the case of a fibre stimulated by a point source embedded in an infinite homogeneous isotropic medium, the external potential is

$$V_e = \frac{\rho_e I_{el}}{4\pi r} = \frac{\rho_e I_{el}}{4\pi \sqrt{\left(\left(x - x_{el} \right)^2 + z_{el}^2 \right)}} \tag{12}$$

where the fibre axis is the *x* axis of the coordinate system, x_{el} and z_{el} the coordinates of the point source, ρ_e the resistivity of the external medium and I_{el} the electrode current. The central and terminal activating function for this fibre with different position of the point source regarding to the fibre end are shown by in Figure 3.3 (Rattay 2008).

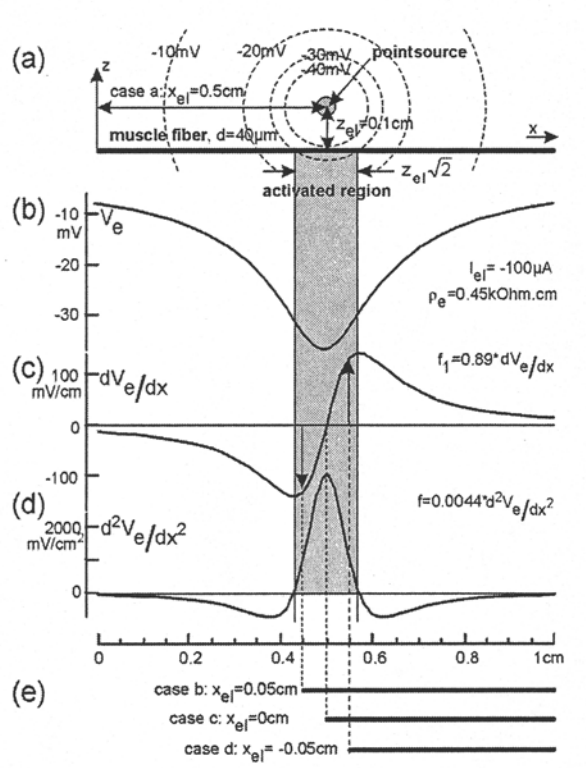


Figure 3.3: Stimulation of a fibre by a point source: geometry, electrical field, and activating functions. (a) Position of the electrode with regard to the fibre. (b) External potential along the fibre. (c) & (d) Terminal and central activating function. Grey marked is the area where the central activating function f_n is positive indicating the activated region. (e) Cutting the front part of the fibre results in a change of the terminal activating function in the first compartment of the fibre indicated by the arrows in (c). All other compartments get the same input as before cutting. Fiber parameters d = 0.004 cm, ρ_i = 0.173 kΩ.cm, c_m = 1.3 µF/cm², Δx = 0.005 cm (from Rattay 1989).

For a position as shown in Figure 3.3 (a) the central activating function f_n is the main input for fibre activation since the terminal activating function f_1, driving just one single compartment, only induces a slight hyperpolarisation at the fibre end. When moving the electrode from a central position to a position near the fibre end f_1 has an even increased hyperpolarisation effect, indicated by the left arrow in Figure 3.3 (c), and therefore the threshold for activation of the fibres rises (Figure 3.3 (e) case b). At an electrode position just over the fibre end the threshold level is the same as for activation in a central position since $f_1 = 0$ (Figure 3.3 (e) case c). Moving the electrode to a position beyond the fibre end, causes positive values for f_1, indicated by the right arrow in Figure 3.3 (c), thus activation is facilitated (Figure 3.3 (e) case d). The above described findings are leading to the conclusion that the fibre is most excitable for an electrode

position which is beyond the fibre end, because for this position both f_1 and f_n have large positive values (Rattay 2008).

In the same publication the author examined also the relation of the radial distance of the point source to the fibre in respect to the influence of the central and terminal activating function. He concluded that for a placement of the electrode close (near field) to the fibre the central activating function f_n is the main driving effect for excitation and for an rather distant placed electrode (far field) the terminal activation function becomes main driving input (Rattay 2008).

For the stimulation technology and method applied for training of denervated muscles in this work (see chapters 3.6 and 5.2) the above results should also apply. But since the quadriceps muscle consists of four muscles with different orientation of their fibres and spatial distribution of the fibre ends an optimal placement of the large surface electrodes can not be found. Nevertheless for fibres lying within more profound regions of the muscles the supporting effect of the terminal activating function has a large impact on fibre activation (Martinek et al. 2005). The influence of inhomogeneous tissue composition of the denervated muscle with intramuscular fat cells (Figure 2.10) on the fibre activation was studied by Stickler et al. (2008). It was demonstrated that by placing fat cells near the muscle fibre the electrical field was changed. This led to activation of the fibre not only at the end but also at the site of the inhomogeneity.

The above presented point source simulation model resembles the situation of the stimulation needle EMG (chapters 4.1 and 4.2). This technique is based on activating a few fibres with a monopolar stimulation needle and measuring the latency of the action potential recorded by a second needle electrode placed a few centimetres away along the muscle fibres. In this case the central activating function is the main driving input since only fibres in immediate vicinity of the needles tip are activated. Moreover, the region where the fibres are excited remains constant at $x_{el} \pm z_{el} / \sqrt{2}$ (Figure 3.3) regardless of the intensity of the stimulation pulses, as long as the position of the needle is kept constant with regard to the fibre (Rattay 1986). This is crucial for estimating the muscle fibre conduction velocity by measurement of the latency of the response and the distance between the needle electrodes for stimulation and recording.

3.2 Basic principles of electrical stimulation

An electrical field applied to nerve or muscle tissue evokes an ion transport through the membrane of the tissue and consequentially activates the cell. At the junction between the electrode and the tissue - in surface stimulation the skin - the electron current is converted into an ion current. The ion current is flowing from the positive electrode (anode) to the negative electrode (cathode). In bypassing this direction of the current leads to hyperpolarisation of the membrane under the anode and to depolarisation under the cathode. When the ion current density in the tissue is sufficient to shift the membrane potential above the threshold level an action potential is elicited.

A set of factors determines the efficiency of electrical stimulation the most important of them will be discussed below.

3.2.1 Electrical parameters

Impulse duration and amplitude

For eliciting an action potential these two parameters have to exceed a certain threshold level specified by the strength-duration curve. By increasing the stimulation amplitude beyond the threshold more distant or fibres with a smaller diameter are depolarised. This enables to control the number of activated motor units and thus the strength of the muscle contraction (Figure 3.4). Above a maximum level of stimulation a further increase of amplitude does not lead to higher force output of the stimulated muscle since all fibres within the nerve innervating it are already depolarised (saturation amplitude).

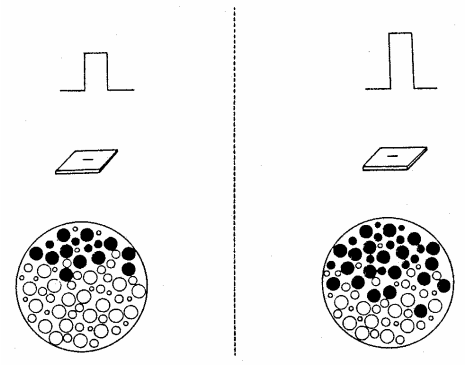


Figure 3.4: Recruitment of nerve fibres within a nerve depending on the stimulation amplitude (from Benton et al. 1981).

Analogue the contraction force of the stimulated muscle can be controlled by variation of the impulse duration. In this case the force output of the muscle is controlled only by changes in the stimulation pulse width between threshold and saturation level at constant stimulation amplitude (Figure 3.5).

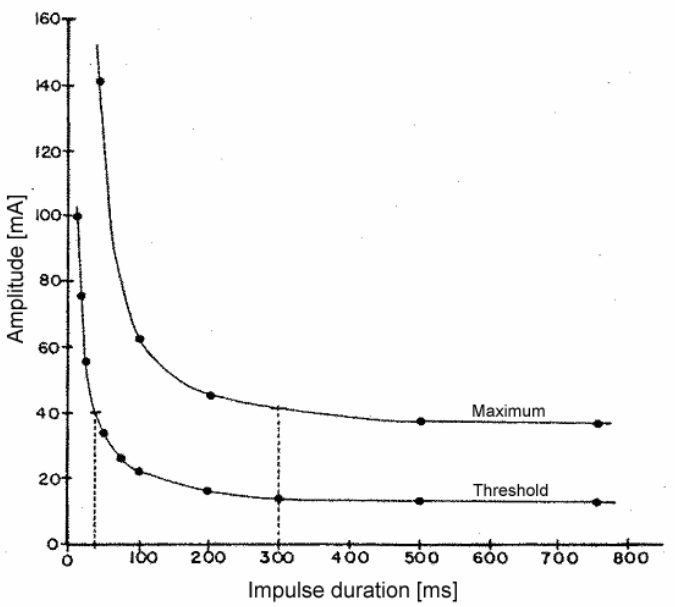


Figure 3.5: Control of muscle force by adjusting the stimulation pulse width while the amplitude is held constant at 40 mA (adapted from Benton et al. 1981).

Electrode position and size

The current density in the tissue is crucial for depolarising the membrane. Apart from the stimulus amplitude the current density is influenced by the size of the electrode, the distance between the electrodes, the distance between the electrode and the nerve or muscle fibre and the tissue composition. Small electrodes applied to the skin close to each other are activating primarily superficial tissue layers, whereas electrodes placed far apart are also activating deeper laying structures (Figure 3.6).

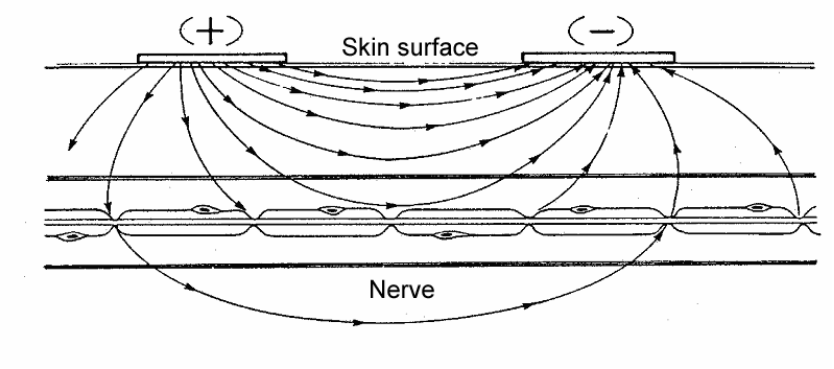


Figure 3.6: Illustration showing typical distribution of current evoked by electrical stimulation with surface electrodes (adapted from Benton et al. 1981).

Electrode size is another parameter which can be used to control the stimulation. For highly selective activation the size of the cathode could be reduced in order to increase current density under the electrode – different electrode. On the other hand the size of the anode or indifferent electrode can be increased to avoid activation.

The position of the active electrode with regard to the nerve is of importance since there are locations with low threshold for activation. It is of advantage to place the electrodes above those motor points (Figure 3.7).

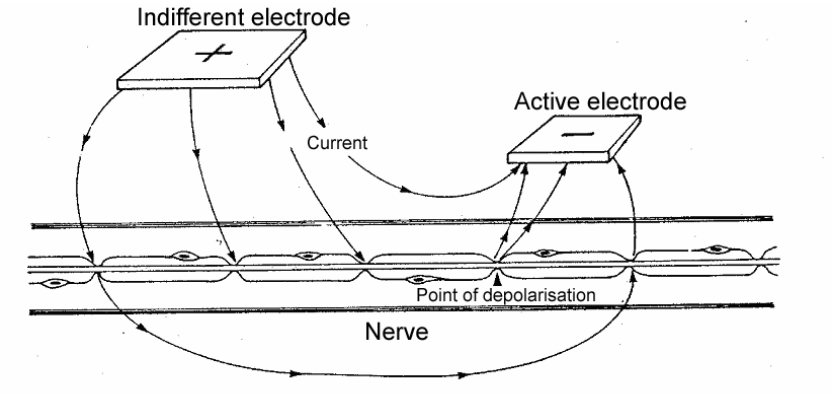


Figure 3.7: Illustration showing the influence of electrode size and location with regard to the nerve on the stimulation (adapted from Benton et al. 1981).

Impulse frequency

Muscle contraction elicited by electrical stimulation is different to normal physiologic activation. Fast fatigue is caused by the nature of electrical

stimulation to activate always the same fibres synchronously. Whereas under physiological conditions the nerve fibres activated within the nerve are changing during a contraction and they are firing asynchronously. This generates a smooth contraction of the muscle even at low stimulus frequency. A neuromuscular component of fatigue is supposed at stimulation frequencies above 40 Hz caused by depletion of the transmitter at the synapses. Additional the blood supply of the activated fibres is reduced during the contraction leading to an impairment of the metabolism.

Impulse shape

Stimulators generating rectangular impulses are the most common used today. The pulses could be monophasic, biphasic or monophasic with alternating polarity. Since monophasic impulses without any compensation phase have an average different from zero they cause a direct current (DC) in the tissue. This can induce skin irritation and corrosion of the electrodes, especially in long-term application. In order to prevent charge accumulation the output stages of most commercially available stimulators deliver only charge balanced pulses and are generally not DC coupled. Capacitive or inductive coupling is used to deliver the impulses instead (Figure 3.8).

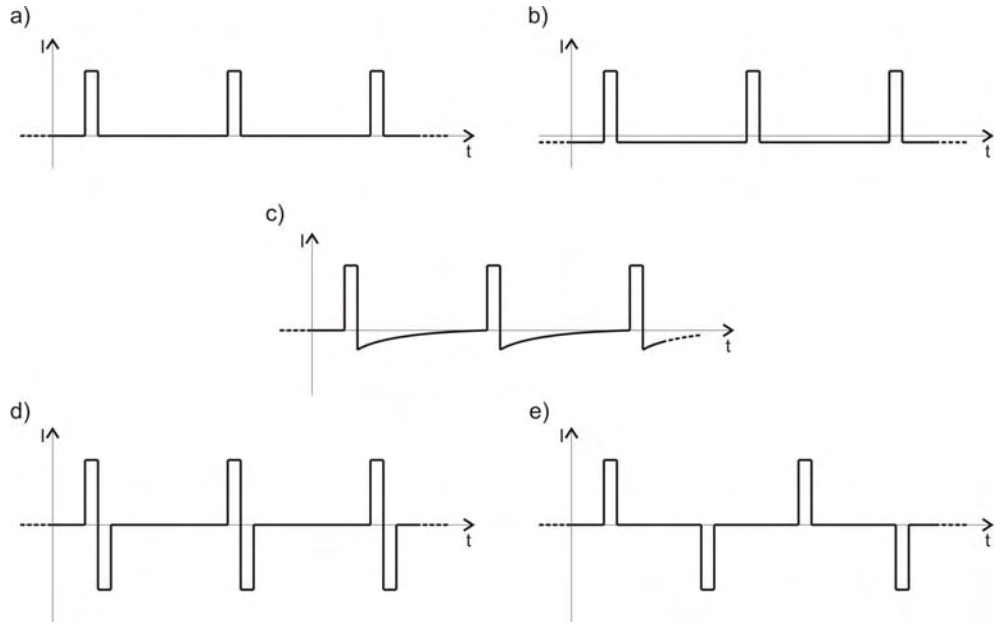


Figure 3.8: Different rectangular pulse shapes used for electrical stimulation: a) monophasic, b+c) monophasic with load compensation, d) biphasic symmetric and e) monophasic with alternating polarity.

3.2.2 Stimulator technology

Devices for electrical stimulation and the associated technology can by divided into different categories e.g. implantable stimulators and devices for surface stimulation or according to the application in devices for therapeutic and for functional electrical stimulation.

The most frequent implanted stimulation devices are the cardiac pacemaker with an incidence of above 500,000 implantations per year and the cochlear implant for stimulation of the auditory nerve in patients with hearing loss. In the last decades the field of application for implantable devices widened to deep brain stimulation for treatment of patients with Parkinson's disease and epidural spinal cord stimulation for treatment of pain or spasticity in neurological disorders. One of the most current developments is a micro miniature electrical stimulator that can be implanted by injection through a needle. It is intended to be used for drop foot stimulation, treatment of post-stroke shoulder subluxation, obstructive sleep apnoea and cough assist. The output stages of implantable devices are able to generate impulses with duration in the range of 10-450 µs, amplitudes of up to 10 V for constant voltage and 16 mA for constant current stimulators. Depending of the application the devices are able to generate pulse trains with frequencies of up to 200 Hz (deep brain stimulation, epidural stimulation) and several kilohertz (auditory nerve stimulation) respectively.

Commercially available surface stimulators are divided into small portable devices for home based use by the patients like devices for transcutaneous electrical nerve stimulation (TENS) or tabletop devices which are mainly used at rehabilitation centres. Usually the stimulators are providing two output channels for simultaneous stimulation at two different sites. The small portable devices are primarily applied in treatment of pain, reduction of spasticity after stroke, traumatic brain injury and spinal cord injury and reversing muscle weakness. Stimulation parameters are limited to pulse widths of 50-300 µs with amplitudes up to 70 mA and a stimulation frequency between 10 and 150 Hz.

The tabletop devices are capable of generating a broader range of stimulation parameters including so called exponentials currents with pulse duration of 100, 200 or even 500ms and intensities up to 130 mA. Most of these stimulators feature different programmes for eliciting tetanic contractions with different on and off time, a programme with frequency

modulated stimulation pulses, medium frequency stimulation and galvanic current. The last-named current is a direct current often used to transport drugs trough the skin into the body (iontophoresis). Moreover it is possible to change the pulse shape (rectangular, triangular) and polarity (positive, negative, biphasic) in some of the devices. In order to prevent skin irritation and electrolysis charge balanced stimulation pulses are mandatory except for galvanic currents. In case of biphasic stimulation pulses this is achieved automatically, for the monophasic pulses a compensation phase after each stimulation pulse for charge balance is required (Figure 3.8 b and 3.8 c).

All the above described stimulators for surface stimulation are used for therapeutic applications. For FES with surface electrodes only few devices are available on the market for applications like correction of dropped foot in patients with stroke, FES assisted cycling and walking in paraplegic patients (Parastep® System, MotionStim 8, ODFS Drop Foot Stimulator).

3.2.3 Electrode technology

There are three types of electrodes used for electrical stimulation depending on the application, each of them features advantages and disadvantages.

Surface electrodes:

This electrode type is the most often used since they are applied for functional as well as therapeutic electrical stimulation. They can be easily applied and removed. The main disadvantages are that their selectivity for activation of discrete muscles is poor and stimulation of deep muscles is not easy to achieve. For daily stimulation the time needed for donning and doffing and the difficulty to position the electrodes exactly the same way are additional drawbacks. To overcome this disadvantages attempts have been made to develop special garments with the electrodes incorporated to ensure proper electrode placement and reduce time for donning and doffing (Rafolt 1999). Two different types of surface electrodes are most widely used: electrodes made of silicone rubber applied to the skin with a wet sponge cloth or only with gel; and self adhesive electrodes composed of a conductive backing layer made of carbon film or knit metal fabric and a layer of self adhesive gel. Electrical stimulation via surface electrodes activates also sensory nerves thus, if the subject has intact or partial sensation, it is possible that the stimulation becomes uncomfortable. If surface electrodes are not proper applied to the skin (e.g. electrode peeling,

in homogeneous contact pressure, etc.), there is also the possibility of skin burns.

Implanted electrodes:

These electrodes are suited for long term application. They have a high selectivity and also stimulation of deep muscles is possible without activation of more superficial laying muscles. Further advantages are that no time is needed for donning and doffing, and their characteristics remain the same in long-term use. The main disadvantage of course is that they require surgery to be positioned. State of the art materials for implantable electrodes and leads are platinum, platinum-iridium and medical grade stainless steel.

Percutaneous electrodes:

These electrodes made of thin stainless steel require a minimal invasive procedure with a hypodermic needle to be placed intramuscular. The selectivity is improved with respect to surface electrodes and also stimulation of deep laying nerves or even direct stimulation of muscles is possible. In long-term application the main drawbacks are electrode breakage (they have to be replaced periodically) and problems with infections at the insertion site. One of their main applications is temporary use to evaluate feasibility of electrical stimulation before permanent implantation (e.g. spinal cord stimulation for controlling spasticity or pain).

3.3 Differences between “spastic, innervated” and “flaccid, denervated” muscles regarding electrical stimulation

The stimulation parameters applied for eliciting muscle contractions are depending on physiological conditions. Of particular importance for electrical stimulation is whether the connection between the muscle and the nerve innervating it is preserved or the muscle is denervated. In the latter case the muscle fibre membrane has to be depolarised by the applied electrical field directly, whereas in the first case the muscle is activated by the nerve innervating it. This phenomenological difference is essential for developing stimulation protocols since functional activation of denervated muscles requires electrical stimulation with long impulse duration in the range of 10 – 150 ms. For excitation of the nerve however impulse durations of 50 µs – 300us are sufficient. In special cases even longer pulse durations of up to 1

ms for functional activation of the thigh muscles are used. By lengthening the pulse width a more homogenous activation of the different thigh muscles is achieved (Bijak et al. 2005). Another effect of the longer pulses is the reduction of discomfort if the patient has preserved sensory function in the stimulated area.

In innervated muscle the nerve distributes the action potential over the whole muscle. This mechanism of distributing the stimulus is missing in the denervated muscle, thus it has to be replaced by an electrical field distribution capable of depolarising the fibres in every part of the muscle. Therefore a completely different approach from technical aspect is necessary to successfully activate the denervated muscle.

3.4 Special design criteria for stimulators for long-term denervated muscles

The studies related to denervated muscles carried out by several research groups (Eichhorn et al. 1984, Valencic et al. 1986, Kern et al. 1995, 1999 & 2002, Mayr et al. 2002) served as basis for establishing the properties for a stimulation device suited for therapy of the human denervated muscle.

A device suited for activation of the denervated muscle should be able to produce stimulation pulses with durations in the range between 1 and at least 100 ms, since the pulse width is the most important electrical parameter.

Another requirement for the output-stage of the stimulator is to generate high intensity impulses because of the low excitability of denervated and degenerated muscle fibres. The second reason for the necessity of high intensity stimulation is the fact that in long-term denervated patients the muscle is covered by a thick subcutaneous fat layer and also within the muscle itself fat and connective tissue is increased. This fat layer and the changed intramuscular tissue composition act as a parallel path for the stimulation current, thus only a part of the applied pulse energy is delivered to the muscle.

For treatment of incomplete denervated patients with preserved sensory nerve fibres, stimulation with triangular or trapezoidal impulses reduces the pain caused by electrical stimulation. Compared to rectangular pulses the above mentioned pulse shapes take advantage of the lost ability of the denervated muscle fibres to accommodate to a stimulus causing a relatively

slow increasing depolarising current (Figure 3.9). This provides selective activation of the denervated muscle fibres without stimulating intact sensory nerves or innervated muscle fibres (Hník et al. 1962, Edel 1991). A device especially designed for treatment of denervated muscle should therefore be able to generate different biphasic pulse shapes.

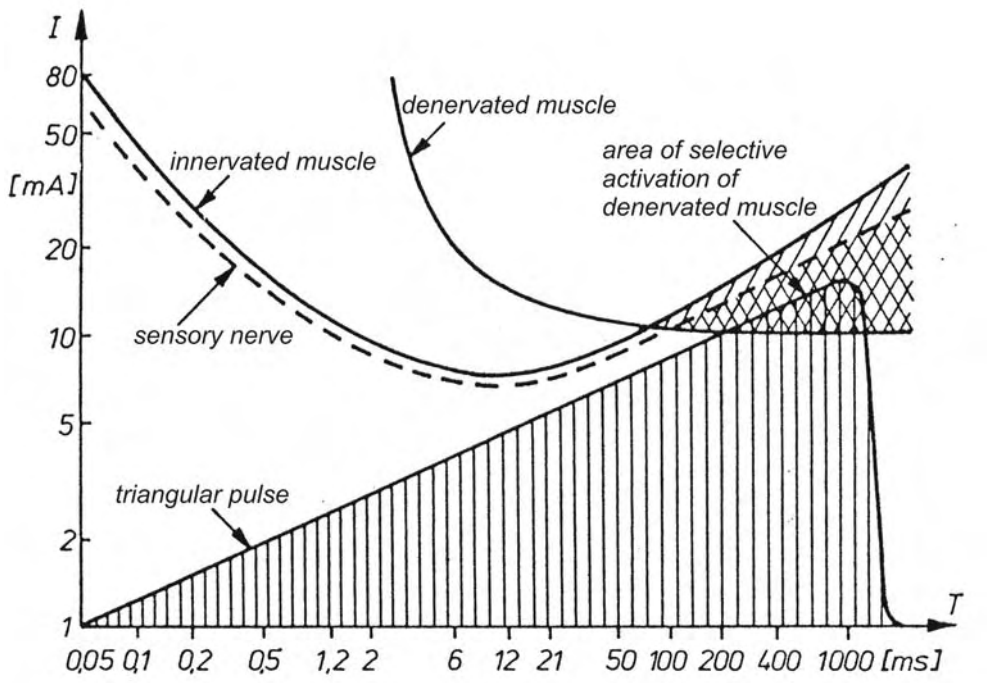


Figure 3.9: Strength duration curve of different excitable tissues for triangular pulses. (adapted from Edel 1991)

A two or more stimulation channel output stage is of advantage since this allows simultaneous therapy of different muscle groups and therefore reduces the patient's time expenditure for electrical stimulation training.

Additional desired features of the stimulation device are a user friendly interface for operating and programming, a built in memory for recording of the training activities and storing different stimulation programs especially adapted for home-based training.

For safety the stimulator should be battery powered to avoid direct line connection and during recharging the battery the unit has to be inactivated.

Since there are no nerves distributing the stimulation impulse, the use of large size electrodes generating an electrical field eliciting the whole muscle is essential. To distribute the applied stimulation current safely the

electrodes should be highly flexible in order to provide an even skin contact over the whole electrode area. This is of greatest importance because inhomogeneous current flow under the electrode easily leads to skin burns due to the high intensive stimulation protocols used.

3.5 Commercially available equipment, stimulators

Since electrical therapy for chronic denervated muscle is discussed controversial by physicians up to now there is no market for electrical stimulation devices especially designed for the needs of the denervated musculature.

Of limited use are the commercial available desktop stimulators able of generating so called exponential current with pulse duration of 100 or 200 ms. Stimulation with these parameters is suited only for slowing down atrophy in temporarily denervated muscles, because the amount of stimulation pulses (0.5 – 2 Hz) and the intensity applied with this setting is low. There is also a gap in the range of selectable pulse widths since these stimulators are only providing programmes with pulse duration up to 1 ms and the exponential current with 100 ms or longer pulse duration but nothing in between.

In Austria new stimulation devices emerged on the market in the last years with user programmable stimulation parameters allowing a continuous selection of stimulation parameters regarding pulse width, burst duration and pause. These devices are better suited for therapy of denervated muscles but nevertheless the devices have to meet the standards which are limiting the output parameters to maximal 300 mJ per pulse. This means that the maximum stimulation current is limited according the stimulation parameters chosen. Another drawback is that they have only one stimulation channel, calling for considerable commitment from the patients due to the fact that each muscle or muscle groups has to be treated individually. The resulting time span allocated for therapy of the patient's lower limbs is therefore two hours or more per day.

To increase the acceptance by physicians and to ease the application of the therapy for the patient new devices for surface stimulation of the denervated muscles are required. Special attention has to be paid to safe and easy handling of such devices since operating errors could cause severe skin burns or other problems.

3.6 A stimulator for functional activation of denervated muscles

This work was published in Artificial Organs 2002; 26(3):276-9.

3.6.1 Summary

In the last years various studies proved that electrical stimulation can improve contractile capability and restore muscle function in long-term denervated degenerated muscles. The low excitability of the muscle cells at the initial stage of training and surrounding connective tissue - acting as an electrical shunt - requires special stimulation parameters. Until now no appropriate devices (stimulators) are commercially available, therefore we were forced to design our own stimulators.

The control unit of the stimulators is based on a microprocessor for maximum flexibility regarding the generation of the parameters such as pulse amplitudes, pulse width, frequency, stimulation times, ramps etc. In addition the microprocessor design allows recording of training data such as stimulation date, time, duration and used programs.

The voltage constant output stage of the stimulator is able to generate biphasic charge balanced stimulation pulse with pulse width of 1-300 ms, stimulation voltages up to ±80 V (160 V_{ss}) and ±250 mA maximum stimulation current. To prevent direct current due to inexact charge compensation, the stimulation pulses are coupled capacitively. Simultaneous two-channel stimulation with independent intensity is possible.

The stimulators are programmed with notebook or personal digital assistants (PDA's) via infrared serial interface. This concept avoids stimulation with wrong parameters because the patient can only use the stimulation parameters pre-programmed for him in the outpatient clinic. For training at home only changes in stimulation intensity within given limits are possible. The portable units are powered by a internal rechargeable battery ensuring mains isolation. Highly efficient switched voltage regulators are used for power supply of all circuits to increase operating time of the stimulator.

3.6.2 Introduction

Direct electrical muscle stimulation with long pulse width – up to 300ms – has been the common treatment for denervated muscle. But this therapy is still being considered to be controversial by many rehabilitation centres and medical professionals. This is largely because current teaching and training for therapists is still based on scientific and technological knowledge of the fifties and sixties. Other sources of criticism are the contradictory statements regarding its effect on nerve growth and reinnervation (Boonstra et al. 1987, Eberstein A. and Eberstein B. 1996). Commercially available stimulation devices are only able to slow down atrophy or maintain the muscle during recovery after non permanent denervation. The stimulation parameters cannot be changed in the scope necessary for efficient therapy of permanent denervated muscles. Most of the stimulators are only able to generate pulse widths of 300, 200 or 100ms and much shorter pulses with about 1 ms pulse width but nothing between. Another limitation is induced by current EU standards for medical devices which are limiting the single pulse energy to 300 mJ per impulse. This is not sufficient for stimulating denervated degenerated muscles (DDM) because of the connective tissue surrounding the muscle acting as an electrical shunt and the long pulse duration needed for excitation of the denervated muscle fibres.

In more recent studies investigating the effects of electrical stimulation on denervated muscles especially build research prototypes or inappropriate commercially available devices adapted to the specific requirements were applied. (Eichhorn et al. 1984, Mokrusch and Neundorfer 1994, Kern 1995, Hofer et al. 1997, Kern et al. 1999, Woodcock et al. 1999).

3.6.3 Concept of a suitable stimulator

Based on experiences from previous experiments in the past years, the following concept for the design of the stimulator was made (Kern 1999).

Stimulation pulse width should be changeable continuously in a range between 1 and 300ms depending on the excitability of the muscle fibres. The output stage of the stimulator has to be capable of generating pulse energies high enough for effective treatment of long-time denervated and degenerated muscles at an initial stage of training where extremely long pulse durations and high pulse intensities are required for training.

The necessity of different biphasic pulse shapes for treatment of patients with incomplete denervation has two main reasons. First for specific

treatment of the denervated muscle fibres without eliciting contractions in the innervated parts of the muscle – altered accommodation in the denervated fibres. Second in case there are any sensory nerve fibres in the denervated area intact the use of triangular or trapezoidal shaped stimulation pulses is less distressing for the patient.

Multi channel simultaneous stimulation of different muscle groups should be possible in order to reduce daily training time.

The device should be controlled by a microprocessor for:

highest flexibility regarding the generation of stimulation parameters,

storing different stimulation programs that are pre-programmed at the outpatient clinic,

easy handling of the stimulator by the patient (only a few control elements: start/stop, therapy program, intensity) and

permanent recording of stimulation activities.

For safety the stimulator should be battery powered to avoid direct line connection and during recharging the battery the unit has to be inactivated.

Since there are no nerves distributing the stimulation impulse the use of large size electrodes is essential for ensuring a contraction of the whole muscle (Kern et al. 1999). Electrodes made of soft flexible conducting rubber that are applied with a wet sponge cloth or gel directly to the skin, already used in our previous trials, proved to be best suited for stimulation of denervated muscles.

3.6.4 Research prototype

Based on the above concept a stimulation system specially adapted to the needs of patients with degenerated denervated muscles was designed (Figure 3.10). Main parts of the stimulator are the control unit for generating pulse parameters, storing stimulation programs and training records, the power supply with the rechargeable battery supplying all components and generating the necessary high voltages for stimulation and the output stage with two independent channels for simultaneous stimulation of two different muscle groups.

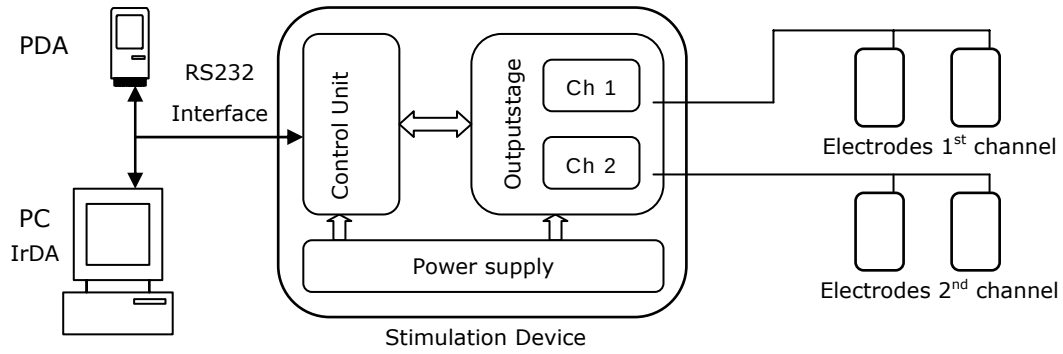


Figure 3.10: Stimulation system overview

Power supply:

For powering the stimulation device special NiMH battery packs made from single high capacity cells with additional thermistors monitoring the temperature were chosen. All circuits in the control unit and the output stage as well as the stimulation circuits are supplied by switching regulators in order to increase efficiency. This allows long training sessions without the necessity of recharging the stimulator. When running the stimulator with both stimulation channels driven at maximum power the overall power consumption is about 35W. To prevent failure of the stimulator due to battery under voltage an acoustic signal is warning the patient if battery low voltage occurs.

Output stage:

The output stage delivers current limited constant voltage impulses to avoid damage of tissue in case of electrode peeling or in case of electrode short circuits damage of the stimulator. Two separate stimulation channels with independent adjustable stimulation amplitude are realized. This allows simultaneous stimulation of two different muscle groups reducing the overall training time for the patient and thus increasing the compliance. To prevent direct current the stimulation pulses are capacitively coupled.

Control unit:

Based on a 80C517 microprocessor the control unit is generating all stimulation parameters such as pulse width, pulse shapes (4 different), stimulation frequency, stimulation on- and off-time, ramps, amplitude, program duration etc. Via infrared serial interface the stimulator is programmed at the outpatient clinic according to actual status of the patient. This interface is activated by a magnetic switch to prevent unintentionally programming of the stimulator. The control unit records training program, date and time of activation and stores this information in an onboard memory. This information is kept till the next visit to the outpatient clinic where the data is downloaded to the personal computer (PC) and added to the patient record for evaluation.

Fig. 3.11: A personal digital assistant showing the graphical user interface for programming the stimulator is shown.

A software for personal digital assistants (PDA) comprising all features of the PC based application was developed (Fig. 3.11). These small and cheap devices are a comfortable solution for on site programming instead of the need to carry around a notebook PC. The control panel of the stimulator consists of 2 panel coders and 4 push buttons. For safe handling of the device the user access is restricted to selecting different programs and varying the intensity within a preset range.

3.6.5 Discussion

With the developed stimulator training of DDM is possible increasing contractile capability and muscle bulk. Prolonged treatment of denervated degenerated muscle with electrical stimulation specially adapted to the current state of the muscle tissue improves the metabolism of the muscle cells and decreases the pulse width to get a muscle contraction. Shorter

stimulation pulse widths sufficient for training due to the increasing excitability of the muscle fibres are allowing higher stimulation frequencies eliciting tetanic contractions that are necessary to achieve the desired muscle fibre tension, constituting a hypertrophic stimulus.

For further reduction of daily training time and functional training like standing up with electrical stimulation the development of devices with 4 stimulation channels is necessary. This requires new high capacity accumulators and a very efficient power supply for the devices to ensure a sufficient long operating time.

Electrode garments with integrated electrodes and cables to simplify donning and doffing will also reduce training time and improve safety preventing inappropriate connections and electrode placement.

3.6.6 Additional details on the developed stimulator

The modular architecture of the stimulator allows easy repair and adaptation of the different parts of the device. Main parts are the power supply unit with the battery pack mounted at the back panel of the stimulator, the output stage generating the required stimulation pulses and the control unit for controlling the operation of the device (Figure 3.12).

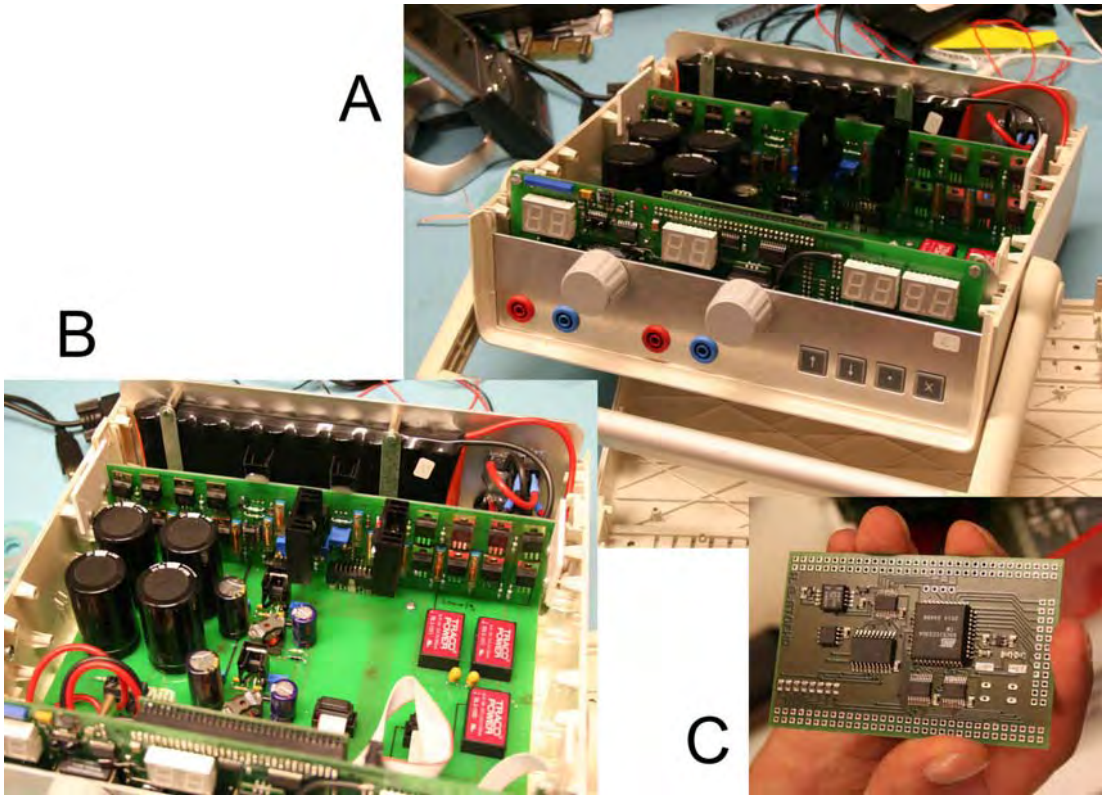


Figure 3.12: Pictures showing the inside of the developed two channel stimulation device. (A) View at the front of the stimulator - with the display filter removed – showing the control panel and the printed circuit print of the control unit with the 7-segment LED display. (B) Top view showing the power supply board of the stimulator and the output stage just in front of the battery pack mounted on the back panel. (C) Second generation main-processor board with extended memory (1GB EEPROM).

Power supply:

Special attention was paid to the efficiency of this stage in order to provide longer battery life and therefore extended training sessions without the need to recharge the stimulator. All different voltage levels needed are generated by switching regulators. The supply for the control circuits is built of commercially available regulator modules whereas the supply for the two stimulation circuits is provided by an especially designed regulator generating a constant output voltage of 80 V, providing up to 300 mA output current. For enhancing safety the unit is inactivated during charging of the battery. The printed circuit board also provides also the connections from the control unit to the output stage and the capacitors for coupling of the stimulation pulses (Figures 3.13).

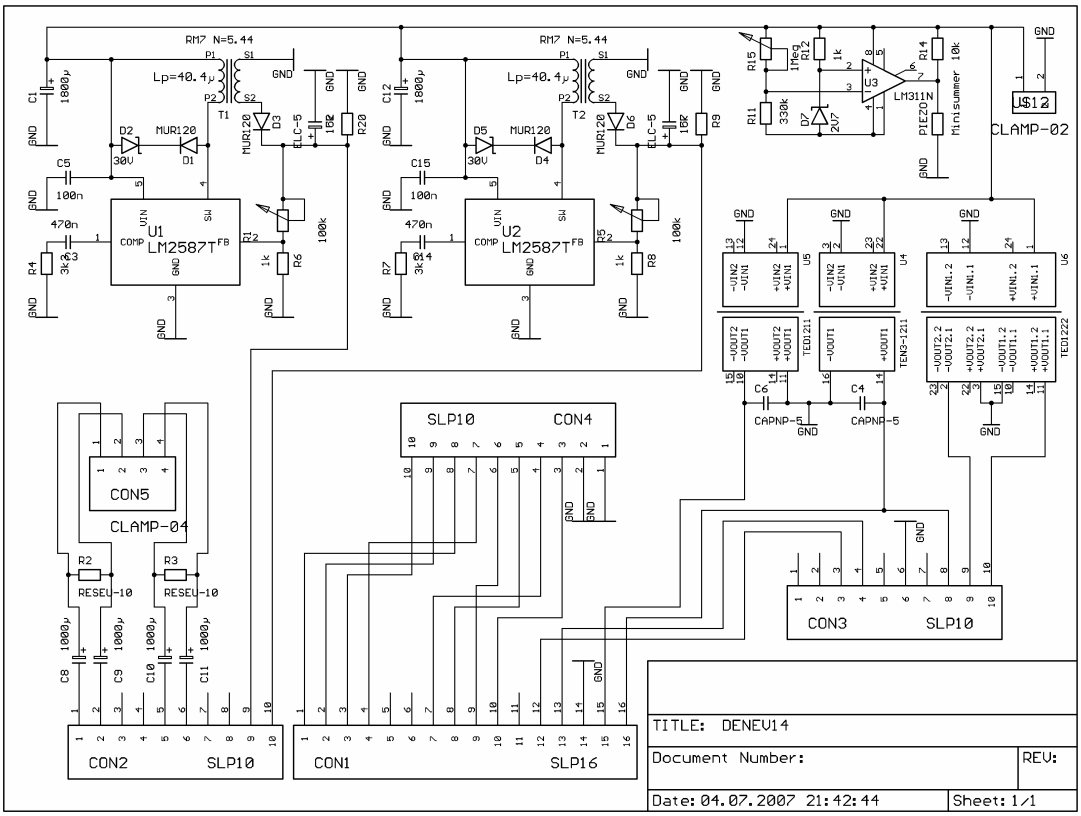


Figure 3.13: Circuit diagram of the power supply including the capacitors for decoupling the two output channels of the stimulator and the connections between control unit and output stage.

Control Unit:

All functions of the stimulation device are controlled by the micro processor based control unit. Easy and safe handling by the patient is one the key features of the stimulator. This was accomplished by reducing the operating elements of the user interface at the front panel of the stimulator. The patient can only choose one of the pre-programmed stimulation programmes, start or stop the programme, and adjust the stimulation intensity for each channel within given limits. An infrared interface enables connection to a notebook PC or palmtop for programming the stimulation device at the outpatient clinic. Wireless communication was selected to prevent any physical connection between the stimulator and external devices in order to maintain line isolation.

The control unit records all training activities by the patient and stores program, date, time of activation and amount of stimulation impulses applied in the memory of the main processor board (Figure 3.14). Every time when the patient has an examination at the outpatient clinic the data can be read out via the interface. Since the personal digital assistants (PDA) for programming and reading out the stimulators memory are more comfortable to carry around and are compared to notebook PCs much cheaper, a software application for PDA was developed. The application is written for Palm OS® v3.0 providing the same functions as the application for the notebook PC. In order to provide full operation with the different PDAs running Palm OS® continuous maintenance of the application was necessary, the most current version of the software is compatible with Palm OS® v5.4.9 (listing of the programme see appendix A).

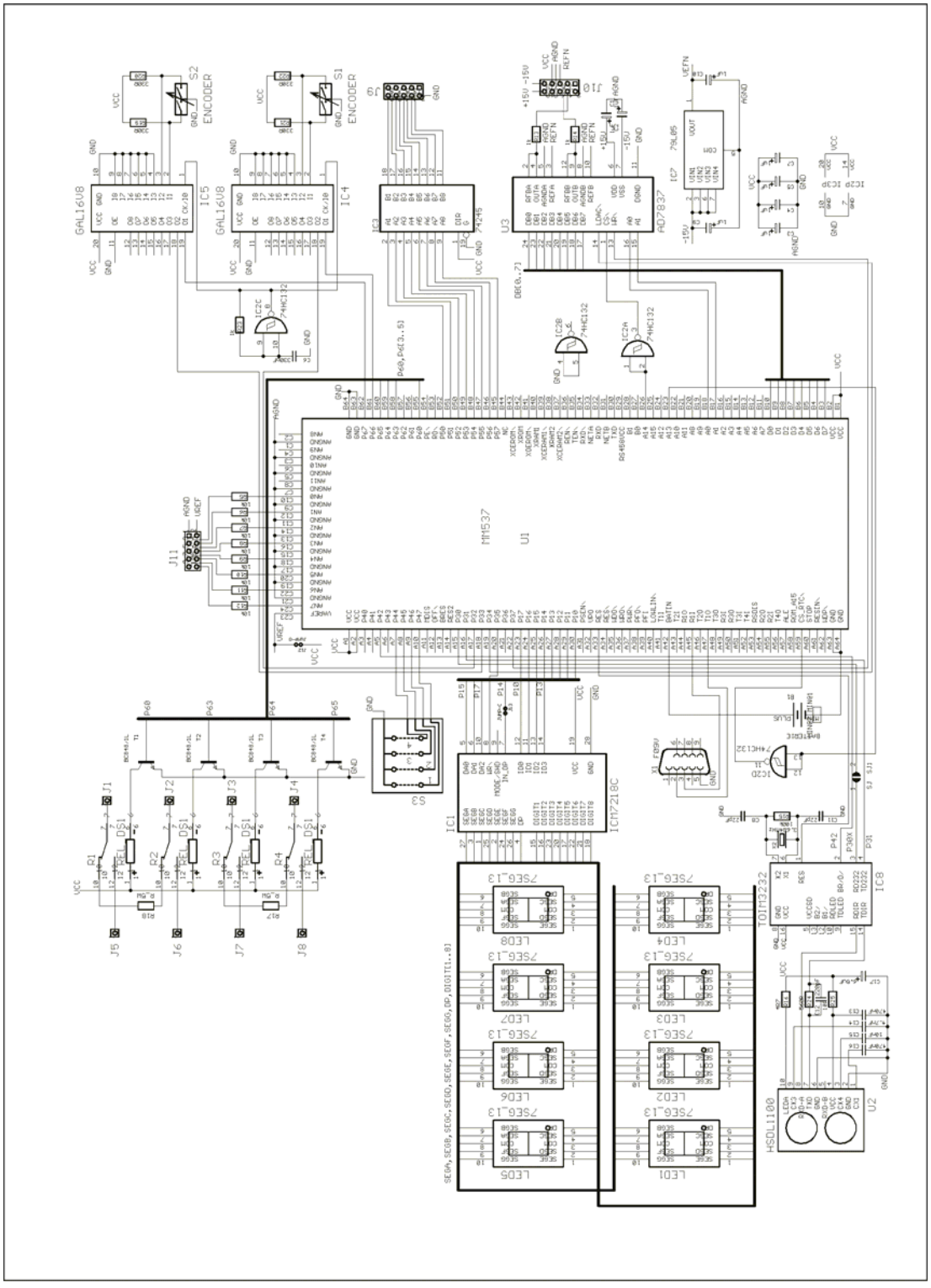


Figure 3.14: Circuit diagram of the control unit with the panel coders for adjusting the stimulation intensity, 7 segment LED displays and the four pushbuttons for selecting, starting and stopping the pre-programmed stimulation programs.

Output stage:

The two channel output stage is designed for constant voltage stimulation with adjustable limitation of maximum stimulation current. Stimulation intensity is controlled by high-voltage adjustable regulators which are also applied for limiting the maximal stimulation current. Since the channels are completely separated not only intensity but also all other stimulation parameters can be adjusted independently. In the current version of the stimulator this is not used so far but could be easily activated with an update of the stimulator's firmware. Implemented in the output stage as well, are circuits for measuring the actual applied stimulation voltage and current for monitoring of the impedance of the stimulation circuit. Evaluation of the impedance is used for stopping the stimulation in case of electrode short circuit or electrode peeling (Figure 3.15).

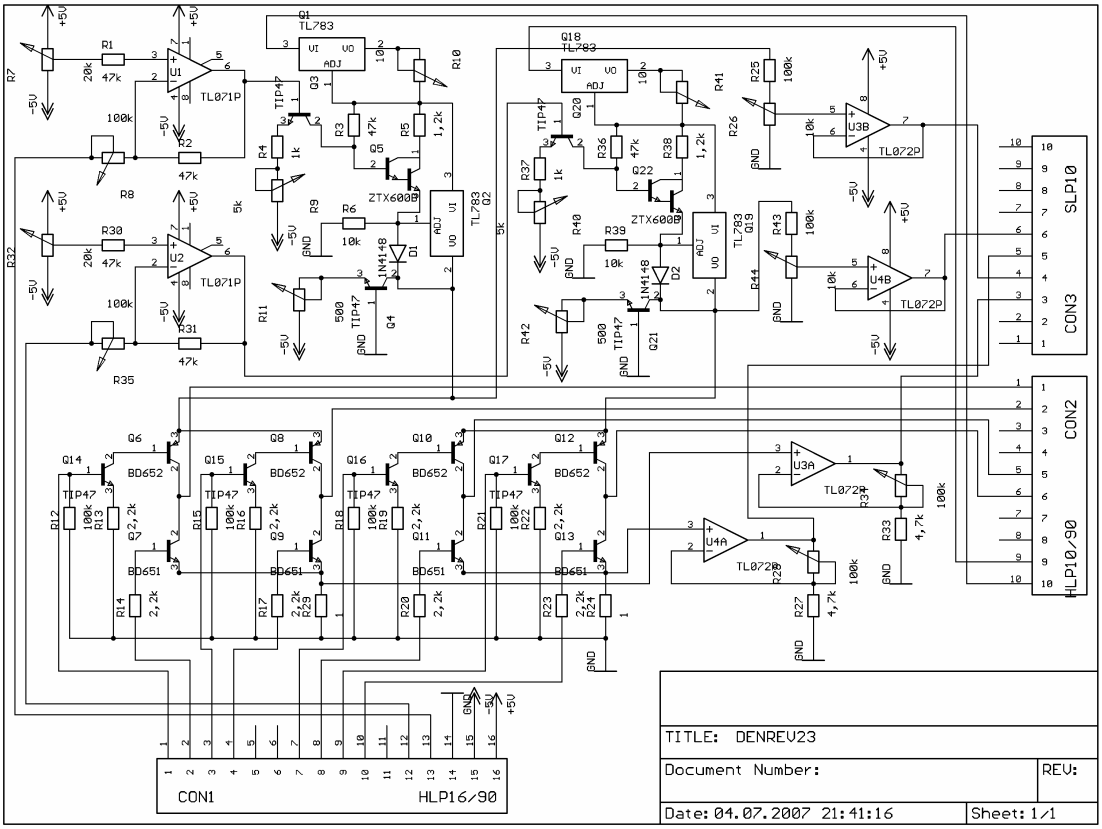


Figure 3.15: Circuit diagram of the stimulator's constant voltage output stage with two independent stimulation channels. High-voltage adjustable regulators (Q2, Q19) are used for controlling the stimulation intensity and for limiting the maximal output current (Q1, Q18).

3.7 Solution for experimental use (Experimental stimulator)

A small single channel stimulation device based on a new designed regulated switching power supply was built for experimental use. The device can be customized for different measurement protocols used in patients with denervated or innervated muscles. This is achieved by an interchangeable microcontroller which is pre-programmed for different tasks.

The main functional units of the device are the switching power supply, the microcontroller with the associated binary coded switches and the output stage. If necessary the stimulator can be fully controlled by an external computer in this case the microcontroller is bypassed.

This switching power supply is controlled by the potentiometer for adjusting the stimulation output intensity. It directly generates the desired output from the battery voltage. Normally needed additional regulators used for setting the output intensity, causing a drop in the efficiency of the stimulation device are not necessary.

For limiting the output current a voltage regulator controlling the voltage drop across an adjustable shunt is used. It allows limiting the stimulation current to a range which is safe for the application in patients.

The switching power supply is build up with a commercially available regulator (LM2587T by National Semiconductor) in a fly-back converter configuration and the standard design for the feedback circuit was altered allowing an adjustment of the output voltage from about 2 V up to 70 V. The desired output voltage is preset by a potentiometer and buffered by an electrometer amplifier and subtracted from the fed back output voltage (Figure 3.16). To keep the regulator stable over the whole output range the standard values for the compensation network were altered.

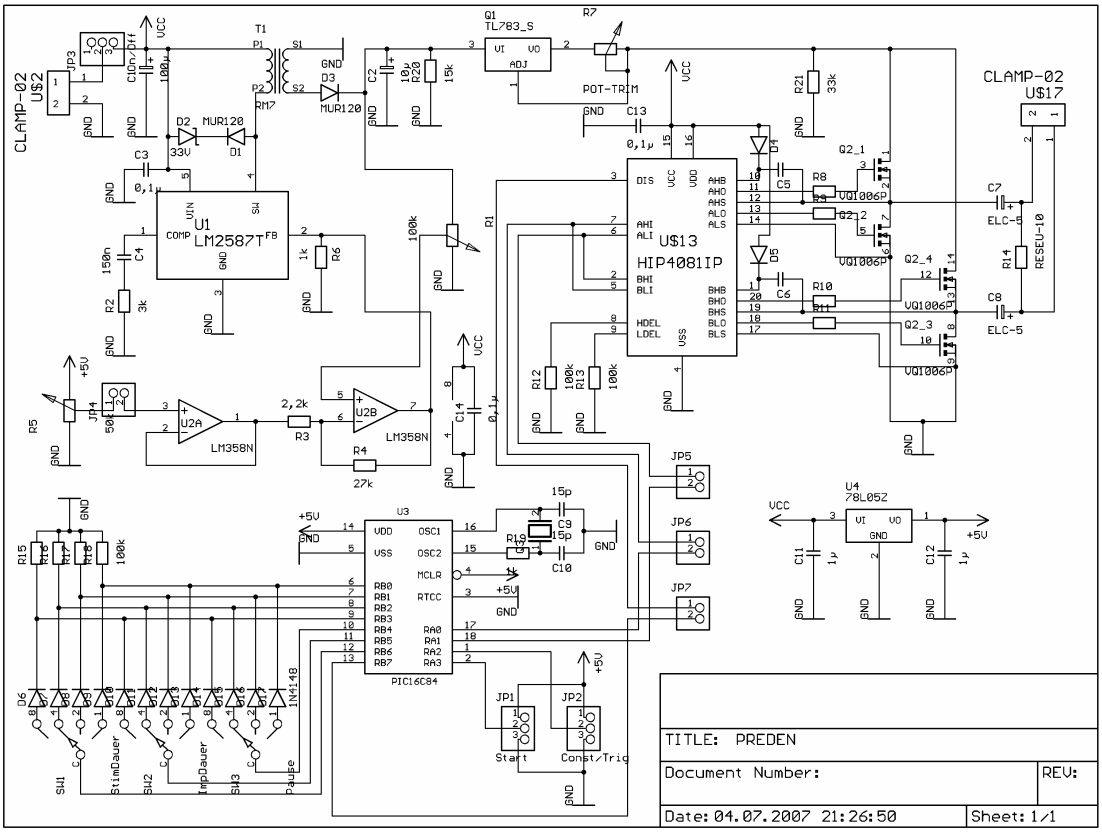


Figure 3.16: Circuit diagram of the stimulation device for experimental use. The main functional units are the switching power supply for generating the output voltage consisting of the LM2587T voltage regulator and the circuit for setting the desired output voltage level (Potentiometer R5 and the operational amplifier LM358N) together with regulator used for limitation of the stimulation current. A microcontroller (Microchip PIC16C84) and three binary coded switches controlling the stimulation parameters pulse with, frequency and number of stimuli applied. The output stage consisting of the bridge driver (Harris HIP4081IP) and the transistor switches. To prevent direct current stimulation even in case of device defect the stimulation pulses are decoupled by output capacitors. By opening the jumpers JP4, JP5, JP6 and JP7 the stimulator can be fully controlled by an external waveform generator or PC.

The stimulator is capable of generating single pulses, pulse trains with a given number of applied stimuli and continuous stimulation. A trigger output allows synchronization with recording devices or other stimulators.

The device is fast adaptable to different stimulation modes and parameters by changing the built in pre-programmed microcontroller. Easy and fast adjustment is provided by the simple user interface of the stimulator consisting of three binary coded switches, a push-button for triggering the stimulator, a switch for selecting continuous or triggered stimulation mode and the potentiometer for setting the output intensity (Figure 3.17).

Figure 3.17: Picture showing the front plate of the stimulator with the jacks for the electrode cables, output indicator LED, switch and push-button for triggering the stimulation and activating the continuous mode respectively, binary coded switches for setting the stimulation parameters, potentiometer for setting the output voltage and power on indicator LED (from left to right).

Depending on the programming of the microcontroller the pulse width can be modified from 0.1 ms up to 200 ms, the interval between consecutive pulses from 1 ms up to 2000 ms and the number of stimuli within a train from 1 up to 256 pulses.

4 Functional assessment of the denervated muscle

4.1 Stimulation Needle EMG / M-Wave

During an action potential small currents in the tissue are generated by the exchange of ions across the muscle fibre membrane. These currents are inducing changes of the potential in the tissue around the muscle fibre which can be measured. The recorded signal is called the electromyogram (EMG). The methods for measuring the EMG are distinguished in non-invasive techniques using surface electrodes and invasive methods using needle or fine-wire electrodes.

When recorded from the skin surface, the EMG signal has to pass various layers of connective tissue, subcutaneous fat and the skin before it is detected by the surface electrodes. This means that the signal is extremely attenuated and the higher frequency components are suppressed due to the low pass filter properties of the tissue. These effects are not essentially limiting the use of surface EMG for recording the activity of a muscle or muscle group where a large number of muscle fibres is active. However to record the action potentials of few or even a single muscle fibre one has to apply invasive techniques to pick up the signal in the immediate vicinity of the active fibre.

In subjects with denervation no voluntary activation of the affected muscle is possible since the nerve normally activating it is destroyed. Therefore electrical stimulation is used to elicit the muscle artificially. Stimulation of denervated muscle with surface electrodes works only if wide pulse widths and high intensities are applied, because the fibres within the denervated muscle have to be depolarised by the electrical current directly. The high stimulation intensity causes large artefacts driving the amplifiers for measurement of the EMG into overload whereas the long pulse duration causes a superposition of the stimulus artefact and the response of the activated fibres. In both cases it is impossible to measure and analyse the response of the muscle to the electrical stimulus. Therefore an invasive method for both activating and recording the response of the muscle fibres was chosen.

Activation of muscle fibres:

A monopolar needle electrode with insulated shaft and a small active area at the tip of 5 mm^2 (Medtronic, Skovlunde, Denmark, Ref. No. 9013R0232) was used for activation of a few muscle fibres in the direct vicinity of the needles tip (Figure 4.1 A). The needle was positioned about 10-15 cm proximal to the upper edge of the patella on the connection line between the spina iliaca anterior superior and the centre of the patella. A few centimetres away from the entry point of the needle a self adhesive electrode applied to the skin served as indifferent electrode. For activating the muscle fibres symmetric biphasic rectangular impulses with a duration of 100 µs (50 µs each phase) were applied with a constant voltage stimulator (see Chapter 3.7).

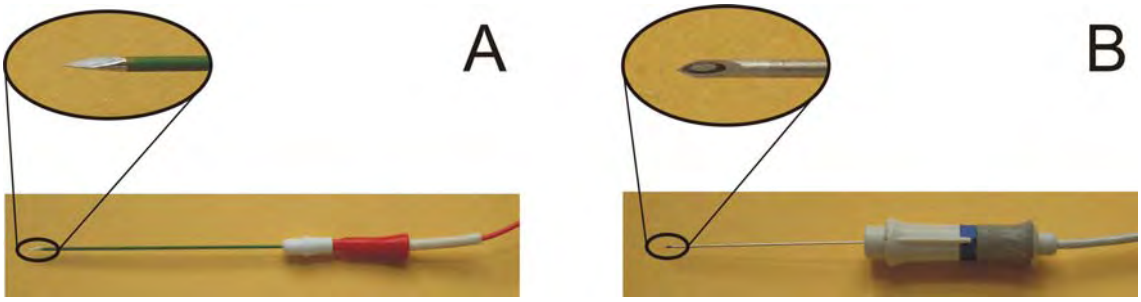


Figure 4.1: Different EMG needle types used for assessment of electrophysiological properties of denervated muscle. (A) Monopolar needle for activation of a few denervated muscle fibres in direct vicinity of the needles tip. Active area: 5 mm^2. (B) Concentric needle for recording action potentials from the activated muscle fibres. It is made up of a wire (core) inserted into a cannula. The core serves as recording electrode (active area: 0.07 mm^2) and the cannula serves as reference electrode.

Recording:

The second needle inserted into the muscle, a concentric needle electrode (Medtronic DCN37, Ref. No. 9013S0031), records the extracellular potentials evoked by the volume conducted action potentials of the activated muscle fibres (Figure 4.1 B). This needle was placed about 3 cm distally to the monopolar needle used for stimulation. After insertion of the recording electrode, single stimuli with a frequency of 1 Hz were delivered to the muscle by the monopolar needle electrode and the recording needle was repositioned until reproducible polyphasic potentials were recorded.

All recordings were conducted with the subjects lying in supine position with the legs extended. In each subject stimulation and recording needle electrodes were repositioned three times in order to examine different parts of the muscle. Accurate needle placement was crucial and sometimes not easy to achieve because of severe muscle atrophy.

4.1.1 Muscle fibre conduction velocity

The muscle fibre conduction velocity (MFCV) is the propagation speed of the action potential along the membrane of the muscle fibre. This parameter influences the transmission of the depolarisation throughout the whole fibre which is important in the process from the depolarisation of the sarcolemma (fibre membrane) to the contraction of the muscle fibre. The action potential is conducted into the interior of the muscle fibre by the T-tubules small invaginations of the sarcolemma where the signal triggers the calcium (Ca^{2+}) release from the sarcoplasmic reticulum. The Ca^{2+} diffuses among the thick and thin filaments where it binds to troponin on the thin filaments. This turns on the interaction between actin and myosin and the sarcomere contracts (Silbernagl and Despopoulos 1991, Dulhunty 2006). Since, normal conduction velocity provided, the action potential arrives almost simultaneously at the ends of all the tubules of the T system, the sarcomeres of the muscle fibre contract at the same time.

For measuring the MFCV single pulses were applied to the vastus intermedius muscle with the monopolar needle electrode. The interval between the stimuli was about 1 s or longer in order to provide enough time for the activated fibres to return to their initial state. This is important since the conduction velocity of the muscle fibres changes if consecutive stimuli are delivered to the muscle within too short time (Mihelin et al. 1991). The responses picked up by the concentric needle electrode inserted into the muscle a few centimetres away, were pre-amplified and band-pass filtered with a low frequency setting of 13 Hz and a high frequency setting of 11 kHz. After the pre-processing the signal was digitised by an A/D converter at a sampling frequency of 50 kHz and stored for off-line analyses on a personal computer. In order to calculate the MFCV the latencies of all positive peaks greater than 50 µV were determined (Figure 4.2).

In each thigh of the patient three different locations were measured by repositioning stimulation and recording needle in order to assess different parts of the muscle. For each needle position the maximum, minimum and mean MFCV were determined.

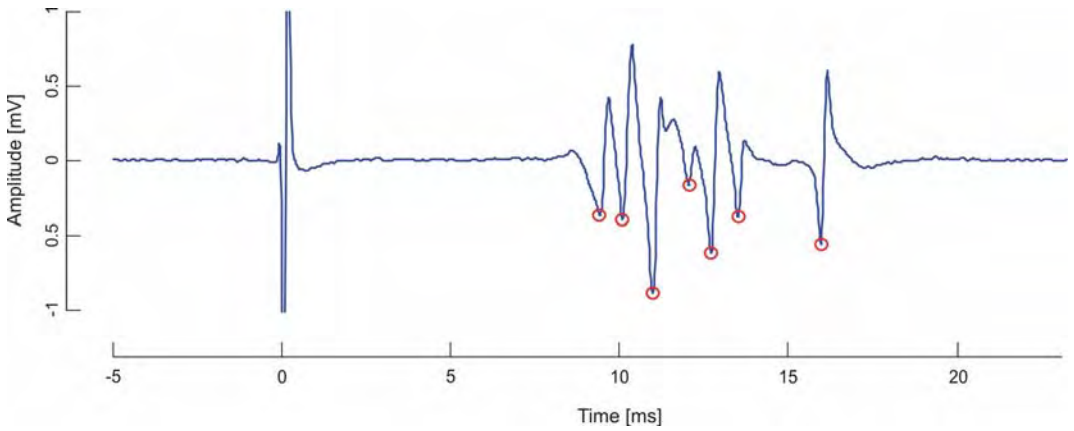


Figure 4.2: Stimulated EMG recording from the vastus intermedius muscle of a subject denervated for 50 months (35 months before starting FES therapy and 15 months of FES training). The first spike at 0 ms is the stimulus artefact. For calculating the MFCV the latencies of all positive spikes greater than 50 µV are determined (marked by red circles). The distance between the monopolar needle for stimulation and the concentric needle electrode for recording the responses was 29 mm. In this example the fastest fibre (first spike) showed a CV of 3.1 m/s, the slowest fibre of 1.8 m/s and the mean MFCV was 2.4 m/s.

4.1.2 Muscle fibre refractory period

The refractory period is defined as the time following an action potential when the sarcolemma of the muscle fibre or any other excitable membrane can not produce another action potential after a further stimulus. This period can be further divided into the absolute and the relative refractory period. During the absolute refractory no stimulus regardless of its intensity triggers an action potential whereas during the relative period abnormally large stimuli will evoke another action potential.

In this study the muscle fibre refractory period (MFRP) was investigated with the same setup used for measuring the MFCV. Because the muscle fibres had to be activated artificially, the moment of triggering an action potential could be controlled by electrical stimulation. The shortest inter-stimulus interval (ISI) showing still a response to the second stimulus was determined by measuring the responses of the fibres to double pulse stimulation where two stimuli with the same intensity and varying intervals are applied. This provides information about the electrophysiological recovery of the muscle fibre membrane after discharge by an action potential (Figure 4.3).

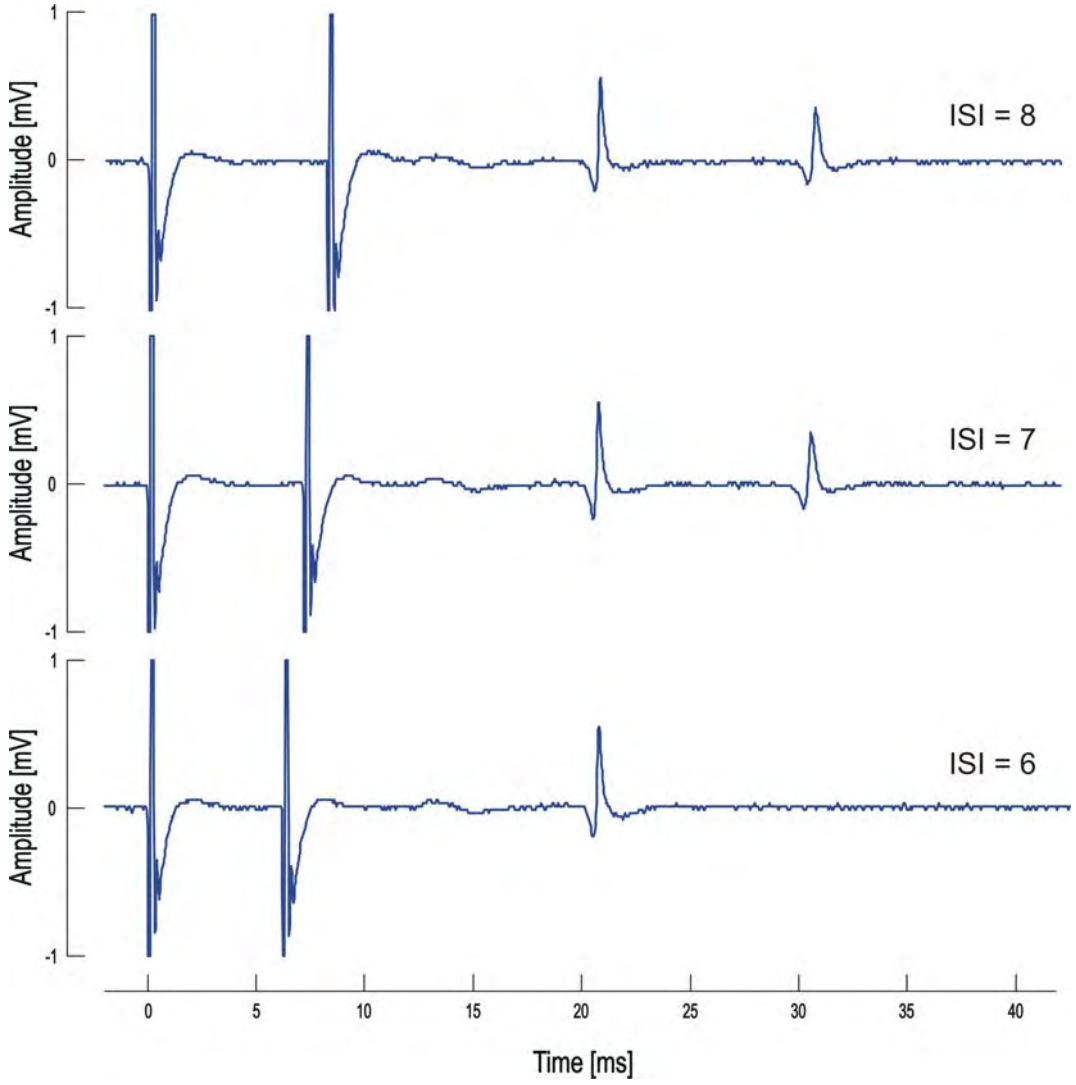


Figure 4.3: Double pulse recordings from a subject denervated for 20 months. The two first spikes in each trace at 0 ms and 6, 7, 8 ms respectively are artefacts caused by the stimulation pulses followed by the responses of a single muscle fibre. In contrast to most of the recordings in this study responses from only one muscle fibre are picked up in the example shown. At about 20 ms the response to the first stimulus and at 30 ms the response to the second stimulus is shown. In this case the fibre shows no refractory behaviour for pulses with an inter-stimulus interval of 7 and 8 ms. At an ISI of 6 ms the muscle fibre is refractory to the second pulse.

In contrast to nerve conduction studies and invasive electromyography (EMG), measurement of MFCV and MFRP is not routinely used as diagnostic tool in neuromuscular disorders. One reason might be that commonly used techniques are time consuming and difficult to handle. Nevertheless the influence of several neuromuscular diseases on MFCV and MFRP has been studied (Gruener et al. 1979, Hoeven van der et al. 1989, Mihelin et al. 1991, Cruz-Martinez and Arpa 1999).

4.2 In vivo assessment of conduction velocity and refractory period of denervated muscle fibres

This work was published in Artificial Organs 2005; 29(6):436-9.

4.2.1 Abstract

Stimulation needle electromyography was used to study the muscle fibre conduction velocity and refractory period in 4 patients with long-term denervation of the lower limb muscles due to lesion of the conus cauda or cauda equina (2 untrained and 2 trained by functional electrical stimulation). In untrained patients the results demonstrated that propagation velocity is reduced and refractory period of the muscle fibre is increased with time of denervation. The patients performing electrical stimulation training showed higher conduction velocities and reduced refractory periods despite of longer lasting denervation. This suggests that electrical stimulation training is effective to improve the electrical properties of the muscle fibre. Since the obtained data show a good correlation to other clinical tests and biopsy investigations, this method could serve as an additional measurement technique to specify the status of the denervated muscle. Further animal experiments and clinical studies are necessary to proof the results in comparison to more invasive established techniques.

4.2.2 Introduction

Stimulation needle electromyography (SNEMG) is a method which is usually not applied during routine electrodiagnostic patient examination. However several research groups have performed studies with different methods and patient groups, because this technique seems to provide valuable additional information about the status of the muscles examined.

The methods applied for measuring muscle fibre conduction velocity (MFCV) comprise surface EMG (Sollie et al. 1985, Hoeven van der et al. 1993) as well as different invasive techniques with concentric needles (Buchthal et al. 1955, Hoeven van der et al. 1993), monopolar needles (Chino et al. 1984) or single fibre needle electrodes (Trontelj and Stålberg 1983, Troni et al. 1983, Cruz Martinez 1989, Mihelin et al. 1991). A second valuable parameter, the shortest inter-stimulus interval (ISI) of a double pulse stimulation showing still a second response can be obtained with the same measurement setup. This parameter is providing information about the muscle fibre refractory period (MFRP). In this work both muscle fibre conduction velocity as well as muscle fibre refractory period was

investigated in patients with permanent (chronic) denervation of the quadriceps muscle due to a lesion of the conus cauda or cauda equina.

The recordings were performed in conjunction with the European project "RISE" that deals with functional electrical stimulation (FES) of denervated muscles (Mayr et al. 2004).

4.2.3 Materials and methods

The method applied in this study is similar to the one reported by v. d. Hoeven et al. (1993) in the biceps brachii muscle.

Stimulation:

For direct activation of the denervated muscle fibres or muscle fibre bundles a monopolar needle (Medtronic, Skovlunde, Denmark, Ref. No. 9013R0232) with an active area of the uninsulated tip of 5 mm^2 was inserted into the vastus intermedius muscle. The needle position was about 10-15 cm proximal to the upper edge of the patella on the connection line between the spina iliaca anterior superior and the centre of the patella with the patient lying in supine position, the leg fully extended. A silicone rubber electrode applied with gel to the skin surface was placed about 2 cm proximal to the needle acting as the indifferent stimulation electrode (Figures 4.4 and 4.5).

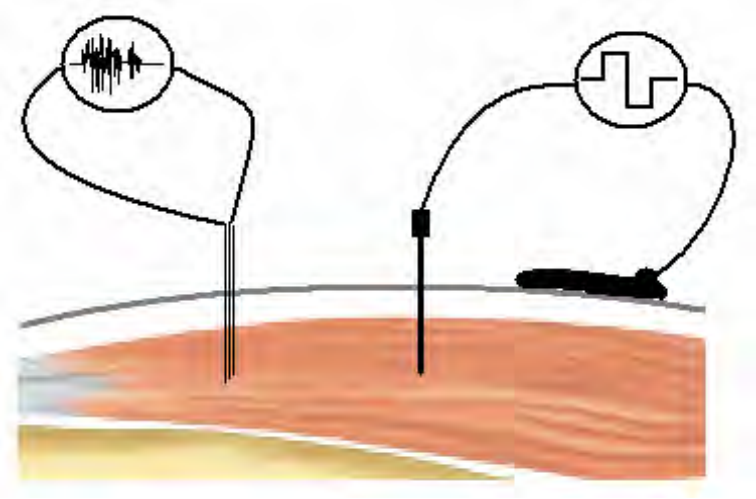


Figure 4.4: Schematic drawing of the SNEMG measurement setup. The monopolar needle with an uninsulated tip is stimulating against a surface electrode. The response of the activated muscle fibres is recorded by a concentric needle electrode placed in the muscle about 3 cm distally.

The muscle fibres were activated with a constant voltage stimulator delivering symmetric biphasic rectangular impulses with a duration of 0.1 ms (50 µs each phase). Single impulses or double pulses (doublets) with inter-stimulus intervals between 1 ms and 15 ms were applied.

With this stimulation setup only a small number of muscle fibres in immediate vicinity of the tip of the needle is activated since the current density is decreasing very quick with increasing distance from the needle tip.

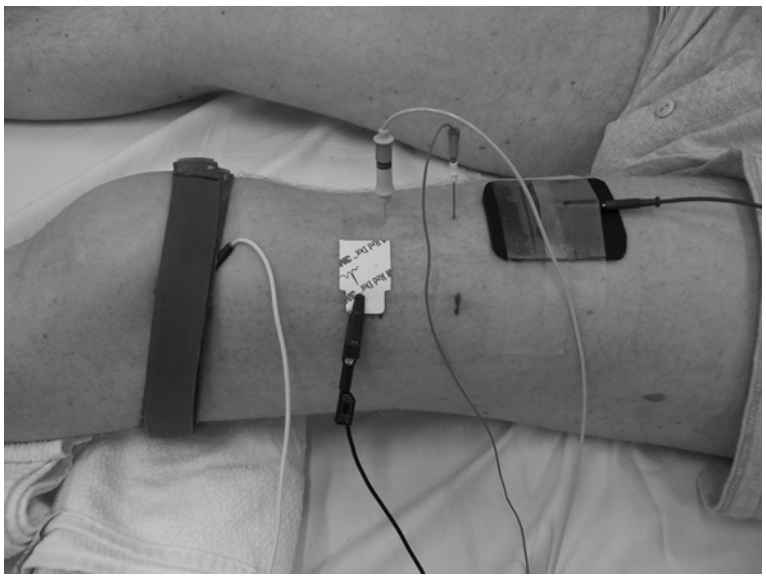


Figure 4.5: Picture of the clinical SNEMG measurement setup. The stimulation needle is placed approximately 15 cm proximal to the upper edge of the patella. A silicone rubber surface electrode placed proximal to the needle is the indifferent stimulation electrode. The concentric needle electrode inserted into the muscle distally records the muscle fibre response to the stimulus. In this setup two different types of ground electrodes were used.

Recording:

The concentric recording needle electrode (Medtronic DCN37 Ref. No. 9013S0031) was placed in the muscle about 3 cm distally from the stimulating needle electrode (Figures 4.4 and 4.5). Care was taken to position both needles perpendicular to the skin surface and in parallel to each other. The recording electrode was repositioned until reproducible polyphasic potentials were recorded (Figure 4.6). The signal was pre-amplified (v = 100) and bandpass filtered (13 Hz –11 kHz).

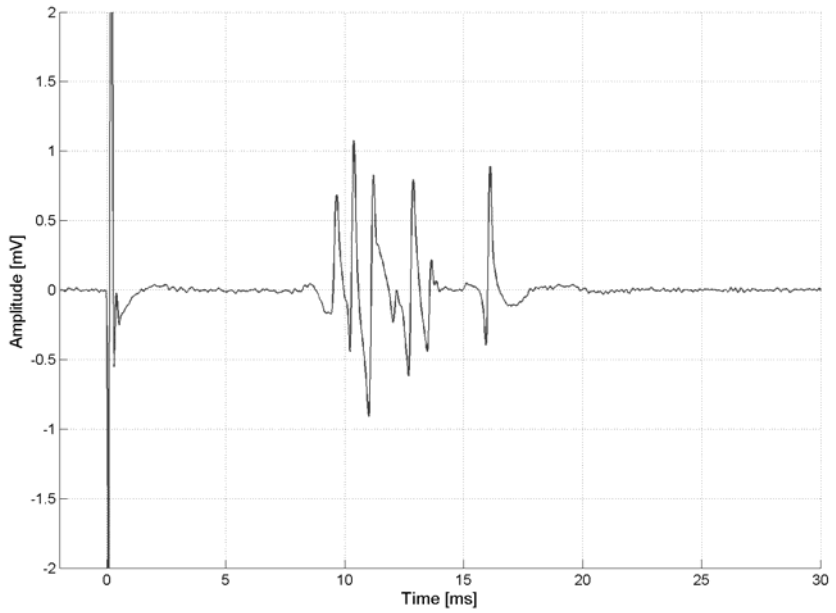


Figure 4.6: Example of a polyphasic response of the denervated vastus intermedius muscle to a single pulse stimulation recorded with SNEMG. The first spike (at 0 ms) in the trace is the stimulus artefact.

With a custom built amplifier designed to meet the special requirements for this measurement method: extremely high input impedance, very low input bias and offset currents, driven ground electrode and driven cable shield for minimizing the effect of the cable capacity.

After pre-amplification and filtering the EMG was digitized via a 12bit A/D-converter at a sampling rate f_s of 50 kHz and stored for off-line analysis of the data.

Patients:

For our measurements we selected two untrained spinal cord injured (SCI) patients (denervated for 10 and 30 months) and two SCI patients (denervated for 33 and 65 months) which had already performed an electrical stimulation training for 6 and 15 months in accordance to the protocol of the "RISE" patient study (Mayr et al. 2004). The patients have all a permanent denervation of the thigh muscles due to lower motor neuron lesion.

4.2.4 Results

MFCV: Till present we have assessed the maximum MFCV by measuring the latency between stimulus and response onset (Figure 4.7). In the patients without electrical stimulation training the maximum conduction velocity (unilateral samples) was 2.4 m/s (10 months denervated) and 1.2 m/s (30 months denervated). The FES trained patients showed a higher maximum MFCV. In the patient with 6 months of training the values for the maximum MFCV were 2.6 m/s in the left and 2.5 m/s in the right leg. The patient with 15 months of FES training showed a maximum MFCV of 3.6 m/s (left side) and 3.2 m/s (right side).

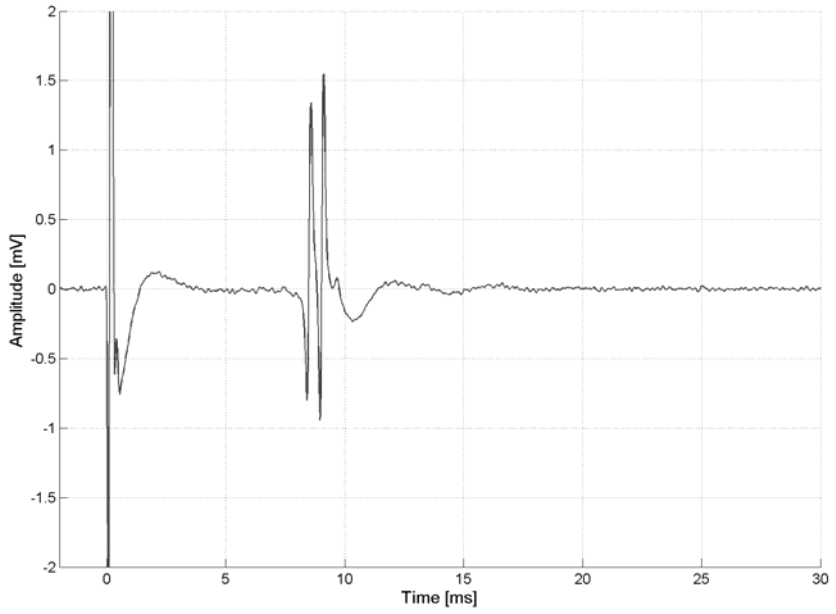


Figure 4.7: A SNEMG recording from the patient with 15 months of FES training (denervated for 65 months). Response of the vastus intermedius muscle to a single pulse stimulation. The first spike (at 0 ms) in the trace is the stimulus artefact. Stimulation amplitude was 25 V and impulse duration 0.1 ms.

MFRP: The MFRP is determined by the shortest ISI with response to the second stimulus (Figure 4.8). For a measured shortest ISI with the second pulse eliciting response the actual value of the MFRP is in the range between the measured ISI and 1 ms below. This uncertainty is caused by the applied measurement method using steps of 1 ms for gradually reducing the ISI.

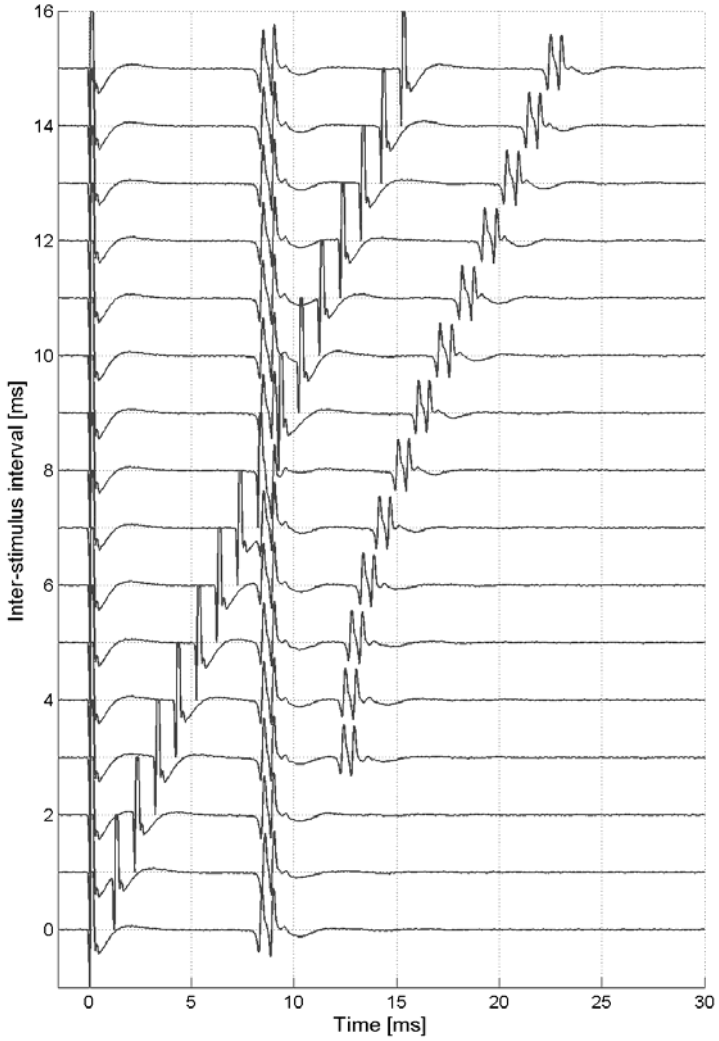


Figure 4.8: SNEMG recording sequence with double pulse stimulation. The ISI is step by step reduced from 15 ms to 1 ms while all other stimulation parameters are held constant (from top to bottom). In the last trace on the bottom the response to single pulse is displayed. In this example the examined muscle fibres are showing a shortest ISI with response of 3 ms (4^{th} trace counted from bottom). The propagation velocity of the second response decreases in the transient area between 3 and 7 ms ISI. Recording from the patient with 15 months of FES training (denervated for 65 months).

The patients without FES training showed a shortest ISI with response of 8 ms (30 months denervated) and 4 ms (10 months denervated). In the patient with 6 months of FES training (33 months denervated) the shortest

ISI was 5 ms in both legs and in the 15 months FES trained patient (65 months denervated) the shortest ISI was 2-3 ms in the left and 3 ms in the right leg (Table 4.1).

Patient	**Denervation time**	**FES training**	**MFCV** [m/s]		**MFRP** [ms]	
	months	months	left	right	left	right
A.C.	10	-	2.4	-	4	-
H.T.	30	-	-	1.2	-	8
G.H.	33	6	2.6	2.5	5	5
M.S.	65	15	3.6	3.2	3	2-3

Table 4.1: Summary of the results from the measurements performed in the two untrained and two FES trained patients. Only unilateral measurement was performed in the untrained patients. For comparison, Cruz-Martinez reported a maximum MFCV of 4.48 m/s in the vastus medialis muscle of healthy adult volunteers (Cruz-Martinez 1989).

4.2.5 Discussion

The difference of MFCV in the untrained patients can be explained due to the different period of denervation (10 and 30 months) and consequently a different degree of atrophy of the denervated thigh muscles. The muscle fibre diameter is known to be a main variable determining fibre conduction velocity (Håkansson 1956). Another factor reducing the propagation velocity is the decrease of the muscle fibre membrane resting potential as a consequence by long-term denervation. It is known that between 12 and 24 months of denervation muscle degeneration starts in addition to the ongoing atrophy process. This is accompanied by changes in the depolarization rate and capacitance of the muscle fibre membrane which will additionally decrease the MFCV (Gruener et al. 1979). It is likely that these changes in the muscle membrane properties are also responsible for the markedly difference in the MFRP in the untrained patients. The measurements in the FES trained patients are showing higher conduction velocities up to 3.6 m/s. Moreover the shortest ISI with response was in the range of or below the values of the untrained patients although they were denervated for a longer period. This suggests that electrical stimulation

training is able to improve the electro physiological properties of the long-term denervated muscle fibre.

4.2.6 Conclusions

The obtained data seem to show a good correlation to other clinical tests and biopsy investigations. The method could serve as a powerful new clinical measurement method for quantification of the actual excitability status of a denervated muscle. This could be especially helpful in the supervision of home based FES training for restoration of long-term denervated degenerated muscles, as the method is simple enough to apply in routine check-ups.

Of course it is necessary to confirm the results in both animal and clinical experiments to proof the reproducibility and ensure validity of results in comparison to more invasive but established techniques like biopsy analyses.

4.3 Computed tomography

Computed tomography (CT), also known as computed axial tomography (CAT or CT scan), was developed by Sir Godfrey N. Hounsfield in the early 1970s at the Thorn EMI Central Research Laboratories, England. The first clinical application of the CT a brain-scan was made in 1972.

A CT scanner is an X-ray machine which produces cross-sectional, two dimensional images of the body. It generates a fan-shaped beam of X-rays rotating around the body. The X-rays pass through the body and are detected by a ring of electronic sensors sensitive to radiation located around the patient's body (Figure 4.9). Depending on the radio-density of the tissue between the X-ray source and the detector the beam is attenuated.

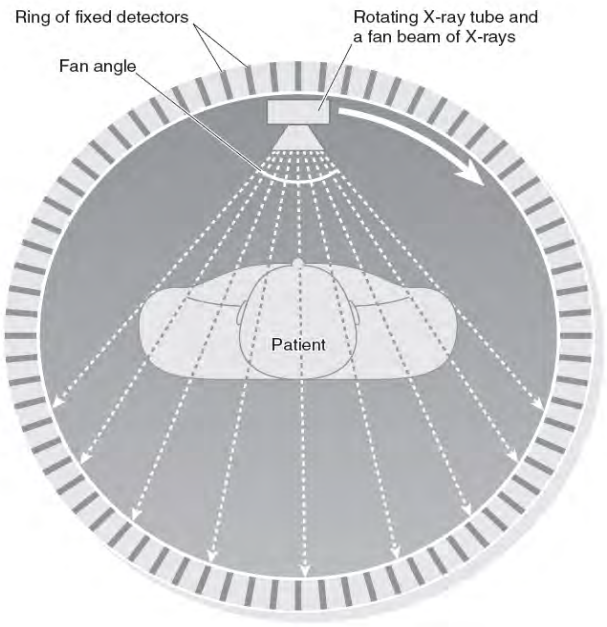


Figure 4.9: Schematic drawing showing the principle of acquiring a CT scan. An X-ray source is rotating around the patient and the radiation emitted is detected on the opposite side either by a rotating segment or a fixed ring of detectors. (from 'Introduction to CT physics', www.intl.elsevierhealth.com/e-books/pdf/940.pdf).

The acquired CT slices are divided into a matrix of 512x512 volume elements (voxels). These intensity readings are passed to a computer, which then calculates the attenuation value of each voxel within the cross-section (slice) of the body. The image is then calculated from this array of voxels converting it into a matrix of 512x512 picture elements (pixels) with each pixel assigned the attenuation value of the corresponding voxel.

The attenuation values are mapped on a scale ranging from –1024 up to +1023 (+3071) arbitrary units named Hounsfield units (HU). On this scale water is assigned an attenuation value of 0 and air –1024 (Figure 4.10).

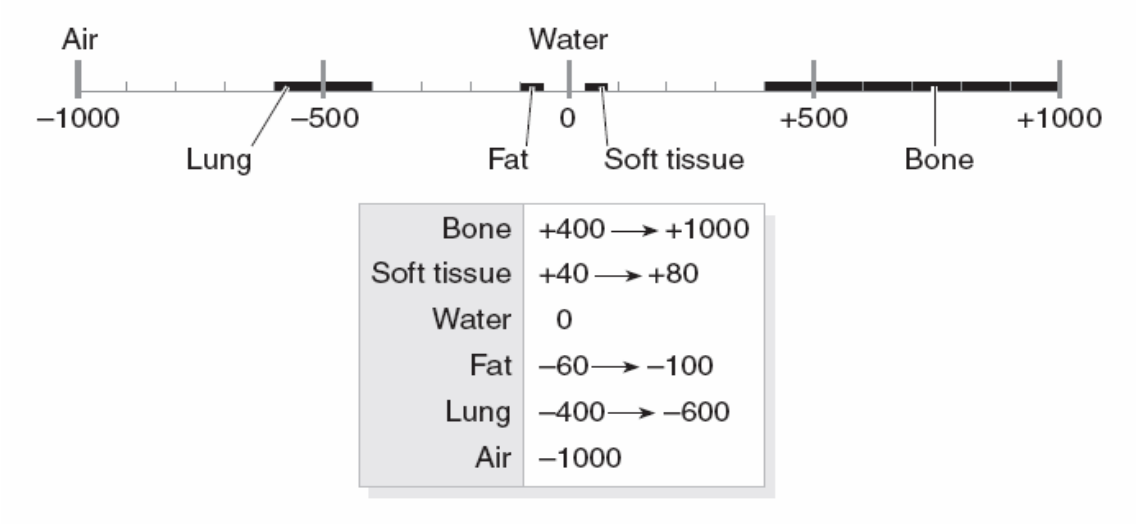


Figure 4.10: The Hounsfield scale of CT numbers. The scale assigns water an HU value of 0 and air –1000. Modern CT scanners have an extended scale ranging from –1024 up to 3071 (from 'Introduction to CT physics', www.intl.elsevierhealth.com/e-books/pdf/940.pdf).

In the image generated from this data pixels with a HU value of –1024 are displayed black and pixels with +1023 (+3071) are displayed in white colour. All CT numbers in between are represented by different shades of grey. Since the human eye is not able to distinguish between 2000 different shades of grey, only a limited range (window) of the scale is displayed, this is called 'windowing'.

The term 'window centre' or 'window level' designates the central CT number or HU of all the numbers within the window. The window width is set that it covers the HU of all the tissues of interest and these are displayed as 256 shades of grey. Tissues with CT numbers outside this range are displayed as either black or white. In all of the CT-scans shown in this work the window centre was set to 50 and the window width to 350. This setting is used for examination of muscle tissue since healthy muscle has an average HU value of 45.

4.3.1 Muscle cross-sectional area measured by computed tomography

Muscle complete cross-sectional area and density (measured in HU) of quadriceps m. and hamstrings were determined as described in Mödlin et al. (2005). Patients were positioned in a way that the top of the trochanter major of both femur bones was aligned in the same plane of the CT-scan. This provided a baseline for all further measurements which were done at distances of 10, 20, 30 and sometimes 40 cm below this reference plane (Figure 4.11). The measurement protocol is routinely carried out at the Department of Radiology at the Wilhelminenspital Wien for many years. The protocol has been retained unchanged in order to keep the acquired data comparable to previous made measurements from other studies. Since the CT-scan 20 cm below the baseline scan normally shows the largest cross-sectional area and therefore the biggest changes this scan was chosen for most of the analyses.

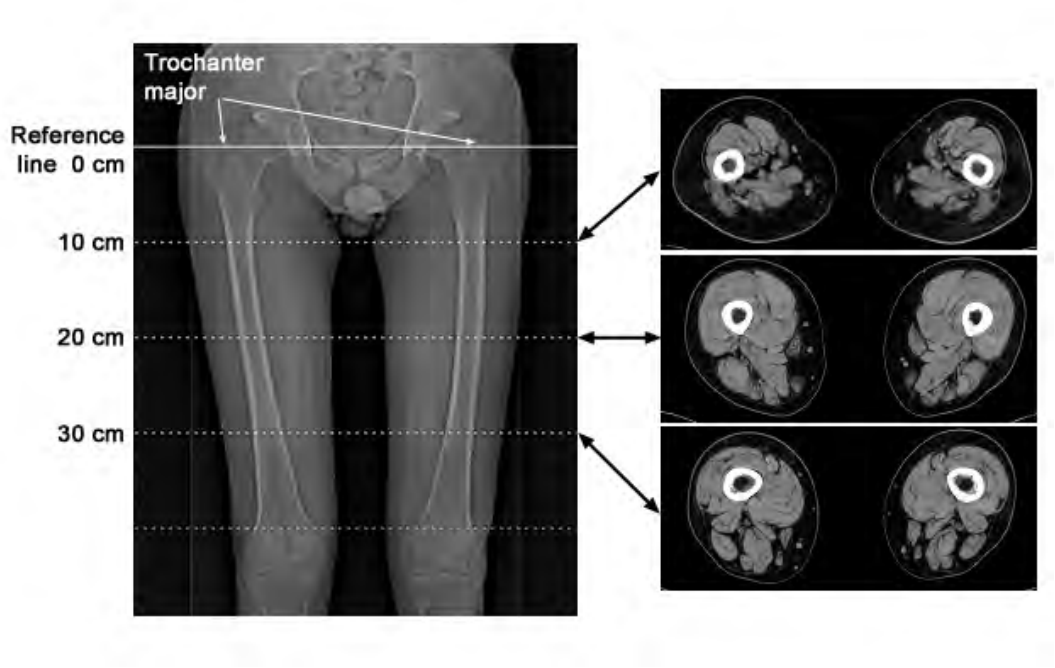


Figure 4.11: Left: CT scan showing a frontal view of a patient's thigh. The thin white line connecting the trochanter major of both thigh bones serves as baseline for the further scans made at different levels below this baseline (dotted lines). Right: CT scans showing the corresponding cross-sections of the thighs 10 cm, 20 cm and 30 cm below the reference line.

The cross-sectional area was determined by manually marking the muscles belonging to the quadriceps muscle group and the hamstrings respectively (Figure 4.12). In patients with severe atrophy of the thigh muscles it was sometimes very difficult to identify the muscles forming the quadriceps or hamstrings group. The data acquired by computer tomography with a Siemens Somatom Plus 4 was converted into DICOM format and stored. This allows exchange and analysis of the acquired images with software independent from manufacturer of the device used for data acquisition.

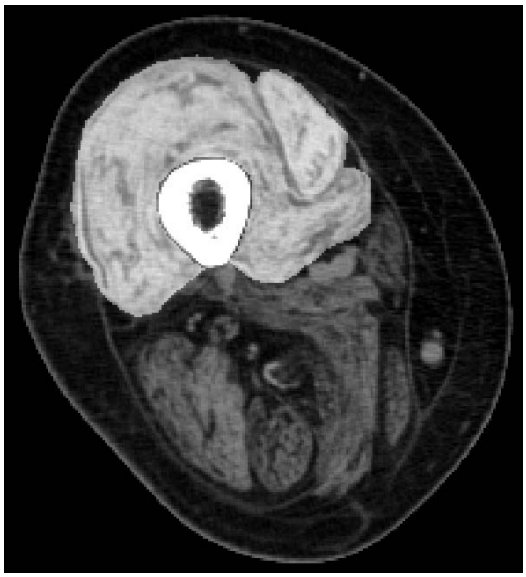


Figure 4.12: CT scan showing the cross-section of the right thigh 20 cm below the trochanter major from a patient denervated for 5.4 years. The highlighted area marks the quadriceps muscle.

To analyse the scans of the patient's thighs a software for image analysis with a graphical user interface was written in Matlab®. It provides functions for cutting out measurement artefacts, zooming for easier marking small regions of interest and automated calculation of the cross-sectional area and the corresponding mean tissue density (CT number or Hounsfield units) of the selected area (Matlab® script see appendix B). A screenshot of the developed software is shown in Figure 4.13. The processed images are saved for reviewing the analysed data or comparison to other scans later on.

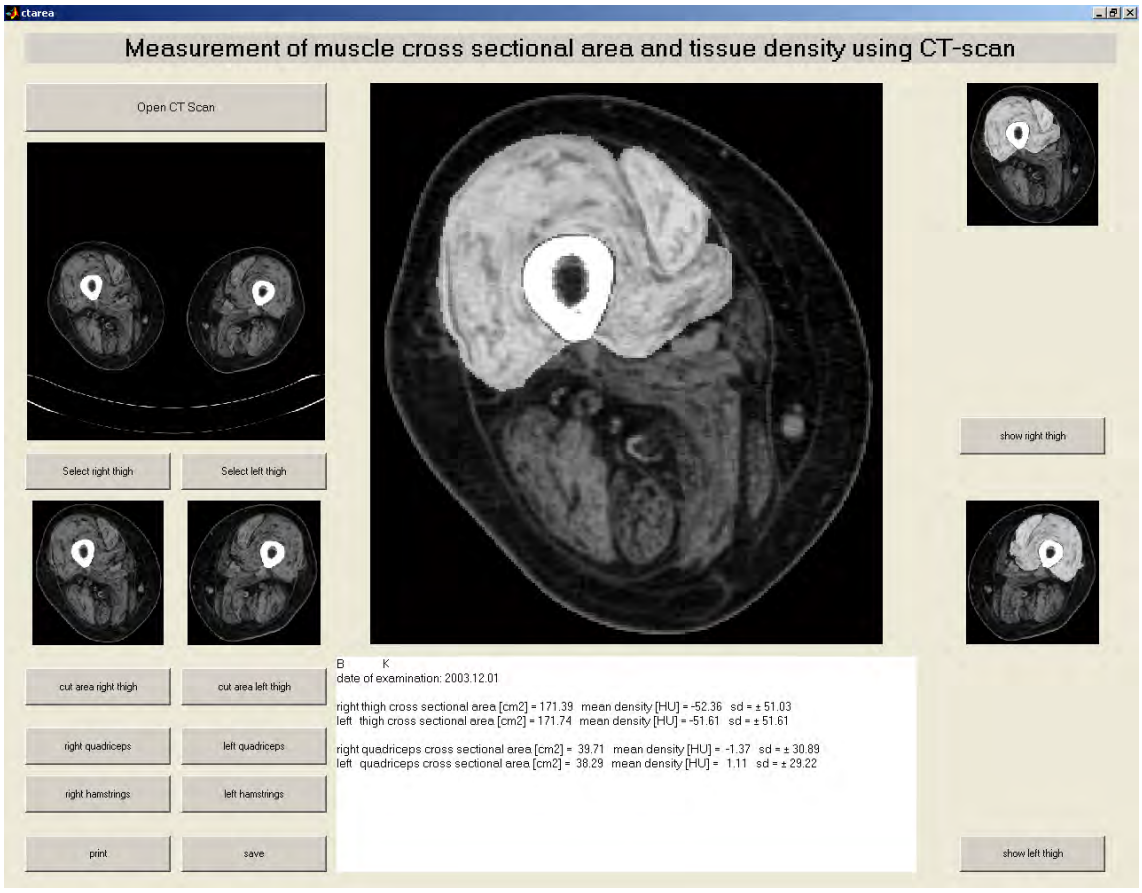


Figure 4.13: Screenshot showing the graphical user interface of the software developed. The centre window is used for zooming into image and marking the region of interest to be analysed. The left upper window displays the whole image acquired by the CT scanner below the cut out regions showing the right and left thigh. On the right side right and left thighs are pictured with the already analysed regions highlighted. By invoking the image of the right or left thigh the image is displayed in the centre window for checking and if necessary editing the marked regions. Example from a patient denervated for 5.4 years.

To illustrate the morphological changes induced by complete denervation of the thigh muscles in the figure below CT-scans of four patients denervated for different periods of time are shown (Figure 4.14).

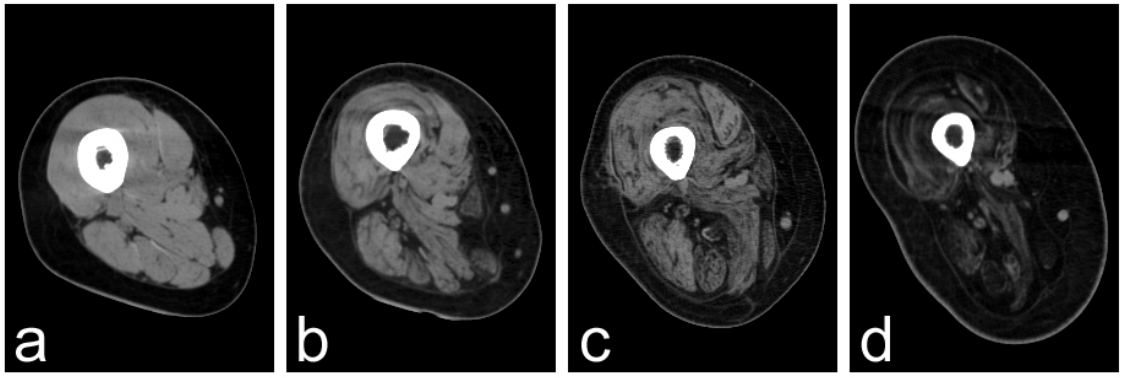


Figure 4.14: CT-scans showing a cross-section of the right thigh 20 cm below the trochanter major. Four patients denervated for different periods of time: a) 0.8, b) 1.7, c) 5.4 and d) 7.7 years. All patients suffered from denervation due to lesion of the conus cauda or cauda equina.

During the early stages of denervation the severe muscle atrophy is the most prominent alteration in the affected limb. In panel a) the right thigh of a male patient denervated for 0.8 years is shown. The measured cross-sectional area of the quadriceps muscle was 31.6 cm². Compared to the results of studies in healthy subjects (Bulcke et al. 1979, Termote et al. 1980 and Maughan et al. 1983) which showed an average cross-sectional area of about 80 cm² for male subjects, this means a reduction in muscle area of 60.5% within the first year. All other muscles, shown in the thigh cross-section above are affected very similar. Within the muscle no alterations compared to normal innervated muscles are visible in the CT-scan.

At later stages of denervation, with progression of atrophy, an enlarged space between the individual muscles of the thigh is observed in the CT-scans. These intermuscular areas are filled predominantly with fat and loose connective tissue (see Figure 4.14 panel b, c and d). Within the muscles an increase of adipose, collagenous and connective tissue is observed by analyses of muscle biopsy. Due to the lower radio density and therefore low attenuation values of these tissues the denervated muscles are displayed in darker shades of grey compared to normal muscle in the CT-scans (Figure 4.14 panel c, d).

The reduction of muscle cross-sectional area is not necessarily in conjunction with a reduced cross-sectional area or reduced circumference of

the thigh, due to the fact that in many patients muscle atrophy is accompanied with an increase of subcutaneous fatty tissue.

4.3.2 Quantification of tissue components, thigh and muscle composition (Distribution of density of tissue)

Measurement of cross-sectional area is one method to describe the state of the muscle, but looking only at the area could be misleading, since as already mentioned above the composition of the tissue within the cross-section is crucial too. A muscle highly infiltrated with non contractile tissue might have a large cross-sectional area however is not able to produce sufficient force. Therefore, to quantify the composition of the muscle or whole thigh the distribution of the tissue density within the muscle or thigh was analysed.

The density distribution was calculated with the help of an image analysing software especially designed for examination of the thighs. It allows zooming out the left and right thigh from the standard CT-scans made at the different levels below the greater trochanter of the thigh bone. Within each thigh it is possible to mark two different regions of interest (ROIs) to be analysed (Figure 4.15). In this work the knee extensor and flexor muscles respectively were chosen for analysis.

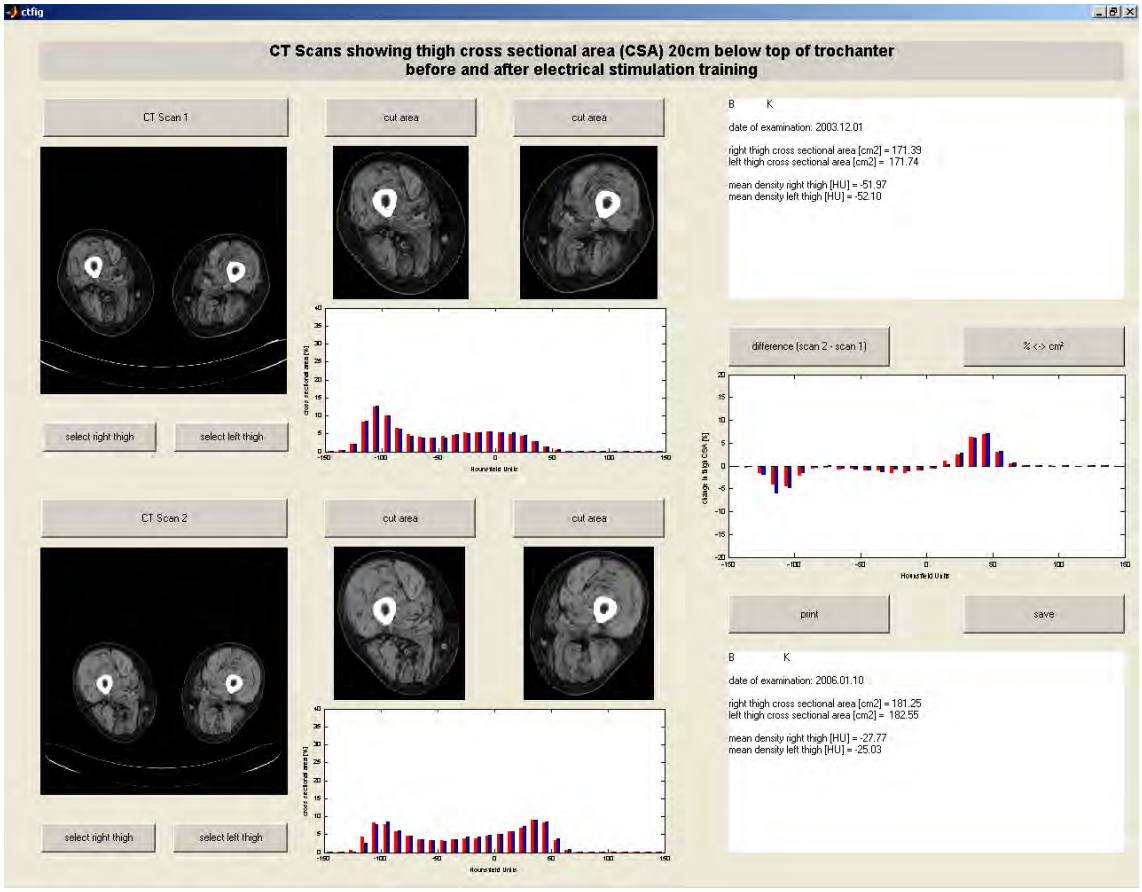


Figure 4.15: Screenshot of the software developed for analysing the tissue composition of the thigh cross-section. In the upper left and centre the CT scan of both, left and right thigh as well as the histogram showing the density distribution before commencing electrical stimulation training is displayed. Below the CT scan and histogram after 2 years electrical stimulation training is displayed. On the right side the difference of the two histograms is presented, indicating an increase of tissue with a density in the range of about +50 HU and decrease of less dense tissue. Data displayed is normalised to the thigh cross-sectional area. Example from a patient denervated for 5.4 years.

For each selected region the cross-sectional area, the mean tissue density and the standard deviation was calculated. The histogram of the density distribution was determined by sorting all pixels within the ROI according to their assigned density value into bins. Bin size was 10 HUs and the range was from –200 up to +1000 HU. For displaying the analysed data this range was limited to –150 up to 150 HU, since all tissues of interest have densities within this range, e.g. muscle tissue approximately 40 – 60 HU and fat about –100 HU (Figure 4.16). Bone tissue is suppressed with this setting. In contrast to other studies (Goodpaster et al. 2000) the range of analysed muscle attenuation was not limited to 0 – 100 HU because also changes in

the content of intermuscular adipose tissue within a selected ROI was considered. All histograms were normalized with the corresponding area of the analysed ROI to allow comparison of different muscles or muscle groups within a patient or between different patients.

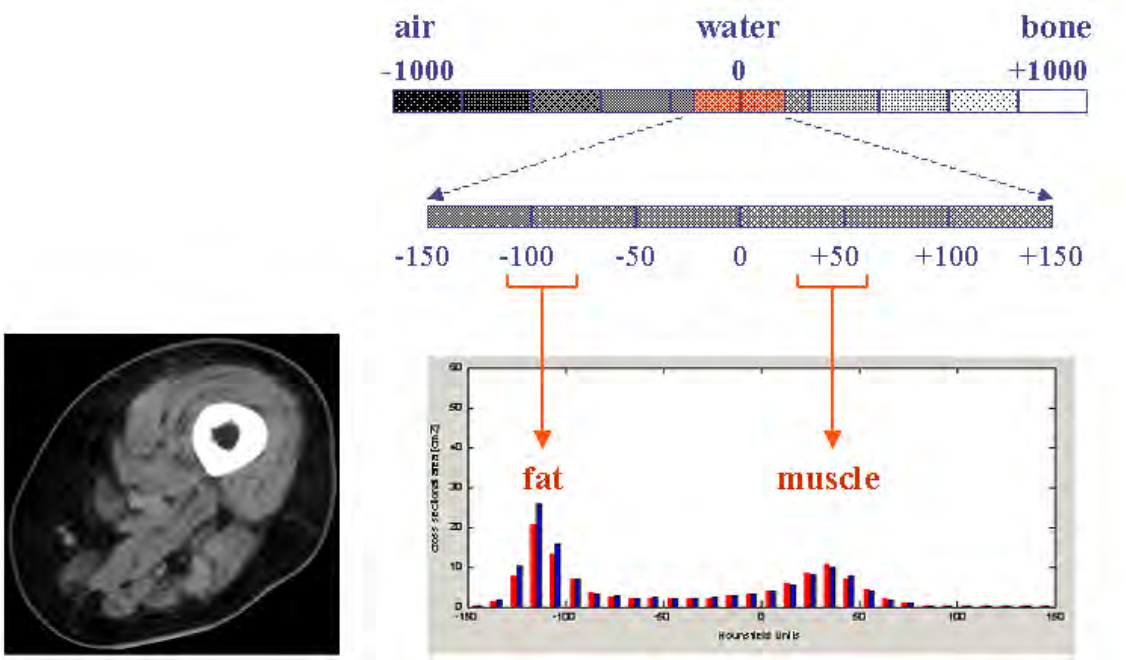


Figure 4.16: Illustration showing the Hounsfield scale and the histogram of the density distribution of the cross-section from the left thigh of a patient.

Two different methods were used to analyse the density distribution within selected areas. An automatic method for analysing the whole thigh and a method were ROIs within the thigh are selected manually. The automatic selection of the ROI is very fast and able to deliver information of the composition of the whole thigh. Changes in content of tissue with different density before and after stimulation training are easily detected by this method. Disadvantage is that the results are influenced by weight gain or loss of the patient and it is not possible to analyse certain tissues (muscles, fat) within the thigh. This is due to the fact that the ROI is determined by detecting changes in density of the analysed pixels, this works fine for interfaces between materials or tissues with a large difference in CT attenuation e.g. air and skin surface of the thigh. For analysing particular muscles or muscle groups within the thigh the ROI has to be selected manually especially in denervated, degenerated muscle highly infiltrated with fat and connective tissue.

To demonstrate how the density of the quadriceps muscle is influenced by longstanding denervation the density distribution in four patients with different periods of denervation is shown in figure 4.17.

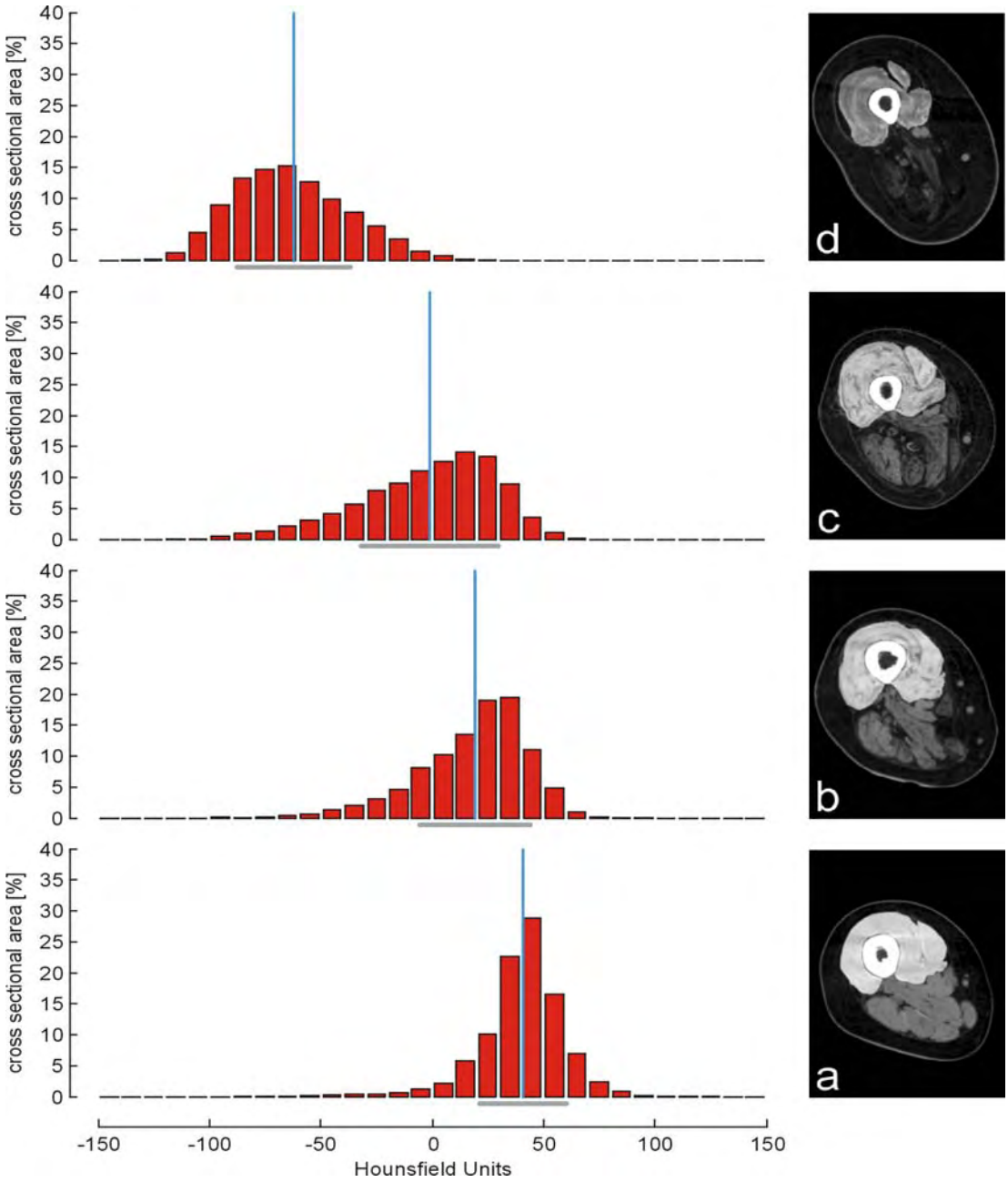


Figure 4.17: Illustration showing the tissue density distribution of the quadriceps muscle and the corresponding CT scan with the quadriceps muscle highlighted in patients with different time of denervation. Examples from the same four patients as in figure 4.14 denervated for a) 0.8, b) 1.7, c) 5.4 and d) 7.7 years. Blue lines are indicating the mean tissue density, the grey bars below the histograms the standard deviation (±1 SD) from the mean.

4.4 Isometric strength testing, knee extension torque

To assess the contractile properties of the denervated knee extensor muscles a custom-made knee dynamometer was used. The subjects sat on the chair with approximate 100° hip angle secured by a strap fastening the hip. Measurements were made at 90° knee angle with the lower leg fixed to the lever of the dynamometer just above the ankle. The axis of rotation of the knee joint was aligned with the axis of rotation of the lever. Strain gauges attached to the lever near the axis measured the knee extension torque exerted by the stimulated thigh muscles. Before start of each measurement the torque was set to zero in order to preclude any bias by gravity, temperature drift or altered position of the leg. Each lower limb was assessed separately to rule out any influences by the non activated limb (Figure 4.18).

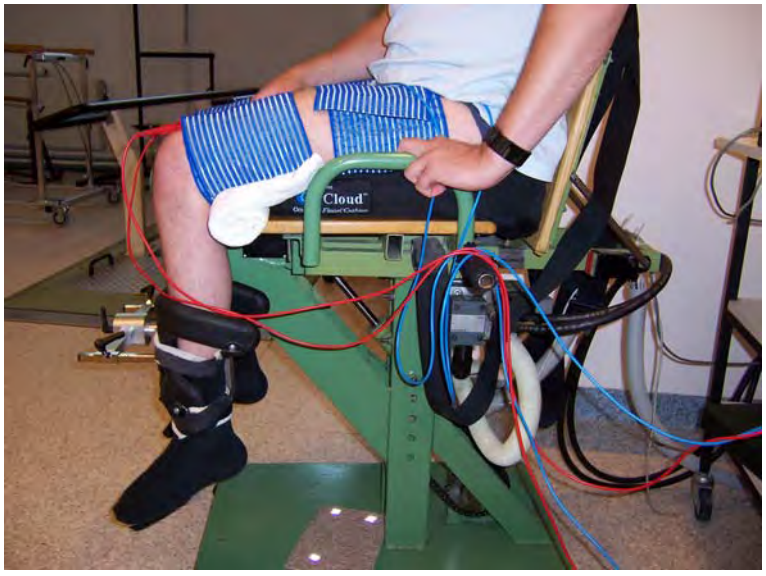


Figure 4.18: Subject sitting on the knee dynamometer. Silicon rubber electrodes for activation of the quadriceps muscle attached with bandages to the thigh.

To obtain information about the necessary stimulation pulse width for activation of the denervated quadriceps muscle, single pulse twitch torque measurements were carried out on the knee dynamometer. For this purpose the muscle was activated by single stimuli with varying pulse width (from

1.3 to 145.4 ms) and maximum stimulation current of ±250 mA. The knee extension torque produced by the muscle twitch was recorded.

Force generation capacity of the quadriceps muscle was tested by measuring the knee extension torque produced by tetanic contraction. Pulse trains with a frequency of 20 Hz, pulse width of 36.3 ms and duration of 2 s were applied to the muscle and the stimulation intensity was stepwise increased up to a maximal stimulation current of ±250 mA. The development of force with increasing stimulation intensity and the maximal contraction force were analysed.

The measurements were performed three times throughout the study: at inclusion before FES training, after one year and at the end of the study after two years of electrical stimulation training.

5 Human application (patient study)

The patient study was carried out in the context of the research project "Use of electrical stimulation to restore standing in paraplegics with long-term denervated degenerated muscles (DDM)" (acronym: "RISE") funded by the European Community within the 5th Framework Program. The consortium included thirteen European partner institutions from Austria, United Kingdom, Italy, Slovenia, Germany and Iceland and additional six subcontractors from Austria and Germany. Nine out of the nineteen were spinal cord injury centres. The project started with November 1, 2001 and ended with May 31, 2006.

Based on preliminary work (Kern 1995, Kern et al. 1999 and 2002, Mayr et al. 2001 and 2002, Zrunek et al. 1991) the aim of the project was to develop a novel clinical rehabilitation method and the associated technical equipment for patients suffering from long-term flaccid paraplegia (denervated degenerated muscles - DDM) with no chance of recovery of the nervous system.

5.1 Patients

Patients suffering from flaccid paraplegia (denervation of lower extremity muscles, conus cauda syndrome) were selected for the study because they are especially good candidates for this approach. After receiving the positive vote from our local ethical committee recruitment of the patients was started. A first screening of the patients was carried out at the participating rehabilitation centres. Patients who met the inclusion criteria and were willing to participate in the study were invited to Vienna.

In Vienna the patients were checked again if they were fulfilling the inclusion criteria and after obtaining their informed consent, a comprehensive assessment procedure was carried out in order to get detailed information about their neurophysiologic and muscular status at inclusion into the study. Demographic data of the patients included into the study is shown in table 5.1.

Subject	Sex	Birthday	Entry date	Paraplegic since	Skeletal level of lesion	Sensory level of lesion
B.A.	m	16.05.1983	12.08.2003	03.11.2002	L1	T11
B.B.T.	m	15.02.1979	15.08.2003	27.07.1999	T12,L1	T10
B.K.	m	01.10.1956	01.12.2003	07.07.1998	T11/12	T9
B.D.	m	06.05.1966	17.02.2003	03.10.2001	T12	T11
C.A.	m	18.01.1965	17.08.2004	28.10.2003	T11/12	T8
C.M.	m	12.02.1976	24.04.2003	12.07.2001		
D.D.	m	30.06.1979	08.03.2004	29.05.2003	T11/12	T12
E.J.	m	19.01.1949	18.08.2003	19.01.1996	T12,L1	T9
E.R.	m	02.05.1951	01.09.2003	04.03.2000	T12,L1	T6
F.R.	m	20.04.1955	16.09.2003	23.05.2000	T9/10	T9
H.B.	f	19.09.1957	17.03.2003	23.06.1994	T11/12	T12
H.G.	m	12.11.1966	20.01.2004	18.05.2002	T11/12	T10
J.W.	m	03.02.1973	26.04.2004	24.03.2003	T12,L4/5	T10
J.P.I.	m	20.04.1962	05.07.2004	25.04.2003	T12,L1	L2
L.M.	m	27.08.1966	17.11.2003	04.10.1997	T12	L1
L.U.	m	12.04.1960	19.04.2004	21.02.1998	T12,L1	T12
L.K.	f	11.02.1974	22.09.2003	18.01.1996	T5	T4
N.T.	m	13.03.1984	20.04.2004	26.06.2003	T11	T10
P.B.	m	08.05.1977	09.12.2003	09.03.2002	L1	L1
S.M.	f	18.07.1965	05.05.2003	27.08.2002	T12	T11
S.M.	m	03.08.1959	19.05.2003	09.06.2000	T12	T10
S.A.	m	27.05.1965	15.12.2003	30.10.1997	T8/9,T11	above T6
S.AS.	m	20.07.1973	24.11.2003	27.02.2003	T11,T12	T9
ST.A.	m	01.03.1950	04.08.2003	26.06.1966	T12	T12
T.M.	m	29.04.1969	30.12.2003	01.09.1985	T12	T10
T.H.	f	24.06.1955	06.07.2004	19.04.2001	T12,L1	T11
T.B.	m	04.09.1974	19.01.2004	15.03.2003	T11/12	T10
W.E.	f	25.07.1948	07.04.2003	05.08.1994	L1	T11

Table 5.1: Data showing the skeletal and neurological level of lesion of the 28 subjects included into the RISE study. Only 5 subjects were female, the mean age was 37.9 years (range from 20.1-54.7 yrs) and the mean denervation time was 5.12 years (range from 0.69 up to 37.13). Marked grey are the subjects which left the study prematurely.

Drop out: Six patients left the study premature during the first year and two in the second year of the study. The reasons for drop out were different. In five cases the motivation and compliance to perform the electrical stimulation training were bad, one patient reported on constipation due to electrical stimulation, one had osteomyelitis in the spine and another had to cancel the stimulation due to a broken femur and implantation of a femoral nail.

Twenty patients finished the study. For the final examinations the patients had to come to Vienna again and underwent the same assessment procedure as at the beginning to rule out any signs of reinnervation. This provides evidence that the improvements in the patients quadriceps muscles are due to the electrical stimulation training and not because of

changes in peripheral nerve system. Only one patient showed discrete signs of reinnervation (needle EMG discrete MUAP discharge in one limb).

5.1.1 Inclusion criteria

- complete or incomplete conus and cauda equina paraplegia (T12-L1) with denervation of muscles
- innervation lacking for up to 9 years
- complete or nearly complete absence of sensation, particularly in the thigh
- neurologically demonstrated flaccid paralysis with no spasticity
- intact skin (no decubital ulcers)
- EMG: fibrillations, spontaneous activity, absence of voluntary activity
- m. quadriceps femoris not excitable with current pulses of 100mA and duration 5 ms
- no hazardous infections or diseases

5.1.2 Neurophysiologic assessment

To assess the neurophysiologic status of the patient at the time of inclusion into the study a comprehensive protocol was setup. This has been carried out for two reasons: (i) to ensure that the patient meets the inclusion criteria for the study and (ii) to describe the patients neurophysiologic condition accurately. The same procedure was repeated at the end of the study after two years of electrical stimulation training. By comparing the results from the beginning and end of the study any changes in the neurophysiologic status of the patients were detected. This made it possible to rule out that changes in the properties of the denervated muscle were due to reinnervation or other regeneration processes.

The protocol consisted of two parts a clinical protocol performed by a experienced physician manually testing and scoring the patient and a laboratory protocol where the patients were examined with different electrophysiological methods providing a very detailed insight on the patients condition.

The clinical protocol consisted of the neurological examination of motor and sensory function below and above the level of the lesion. The motor function protocol comprised a description of muscle group size, the range of passive and active movement of all limb joints, stretch reflexes, and scores from manual testing of muscles including evaluation of preserved and impaired functional movements. Sensory functions were evaluated qualitatively for all four modalities: touch, vibration, pinprick and temperature. After recording the threshold for perception of all four modalities within all corresponding lumbar and sacral dermatomes also the presence and distribution of altered sensation such as allodynia and dysesthesia and sensation was examined. Temporal and spatial summation of short lasting afferent inputs after repetitive stimulation within the same or different locations were evaluated as well.

Laboratory methods were used to evaluate motor unit activity elicited by stretch reflex, H-reflex, cutaneous muscle reflexes, and reinforcement manoeuvres. Sixteen channel poly-electromyography recording, with surface electrodes attached to muscle groups of lower limbs and trunk was used to record muscle activity during several different active and passive manoeuvres. Standard needle EMG and measurements of conduction time were performed, together with transcranial and lumbosacral magnetic stimulation. The sympathetic skin response, the change in skin potential following arousal stimulation, was recorded.

5.2 Stimulation training protocols, execution of training and time expenditure

The patients were carefully instructed how to perform their electrical stimulation training properly. Handling of the stimulator and donning and doffing the electrodes were practiced at the Wilhelminenspital during their four day visit for assessment at the time of inclusion into the study. An operating manual for the patients describing the handling of the stimulation device and the associated equipment, as well as a programming manual for the physician was written in order to provide additional information (see appendix C). Based on the experience from our pilot work (Kern et al. 1999 and 2002) an electrical stimulation protocol especially adapted to the needs of denervated muscles was setup.

In general all patients used the following electrical stimulation parameters:

1. frequency 2Hz, impulse duration 120ms, impulse pause 500ms, On 5s/ Off2s;
2. frequency 20Hz impulse duration 40ms, impulse pause 10ms, On 2s/ Off2s.
3. Once a day 30 minutes for each muscle group.

The electrical stimulation training was carried out at home by the patients with the pre-programmed stimulation device (Hofer et al. 2002). Although the quadriceps muscle was the target muscle in the study most of the patients stimulated also their gluteus and calf muscles to maintain these muscles too. Regular check up visits at their responsible rehabilitation centre provided support in case of problems related to the stimulation training or other medical problems.

In the first year of stimulation the patients used rubber electrodes, placed in a wet sponge pouch. This type of electrodes allows a safe application of the stimulation therapy but due to the relative high impedance of this electrode attachment the training intensity is reduced. No side effects- in particular burnings were seen during this stage of training.

After one year of stimulation the rubber electrodes were applied with gel directly on the skin to increase stimulation intensity (Figure 5.1). In this case some of the patients had rare skin burnings at the edges of the electrodes.

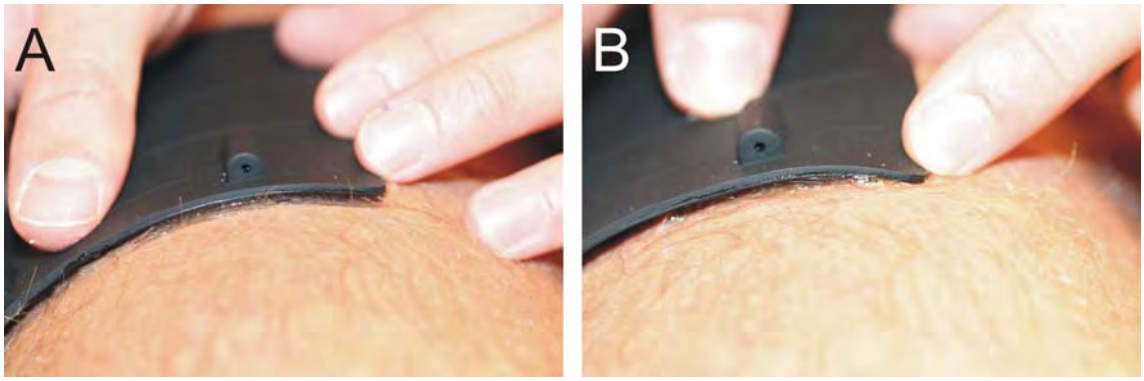


Figure 5.1: Commercially available standard rubber electrodes attached to the skin surface with conductive gel. (A) Illustration of an electrode attached with constant pressure over the whole area. (B) Demonstration of inhomogeneous contact pressure with higher pressure at the electrodes edges. The conductive gel is spilled out from the gap between the electrode and the skin surface. Burnings due to the direct contact between the electrode and the skin could be caused.

Therefore a new safety electrode was developed (Mayr 2007) with a special geometry preventing increased contact pressure at the edges and direct contact of the rubber electrode to the skin. In addition the prominent isolated border on the skin faced electrode side prevents the conductive gel from being spilled out ensuring a homogeneous distribution of the stimulation current across the whole electrode area (Figure 5.2).

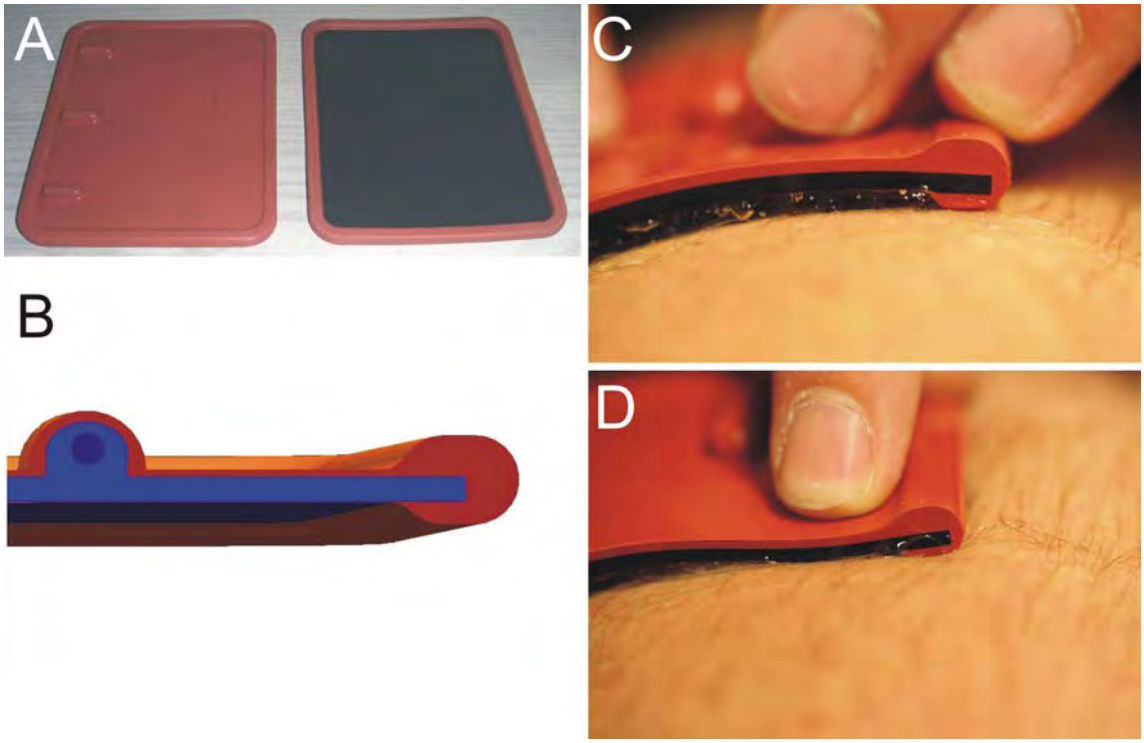


Figure 5.2: Safety electrode. (A) Picture showing the isolated backside of the electrode and the active side applied to the skin with a small isolated border. (B) Cross-section of the electrode: blue conductive rubber material, red isolating silicon rubber. (C, D) Pictures demonstrating the application of the electrode to the skin surface. (C) Normal evenly distributed contact pressure. (D) Inhomogeneous contact pressure with higher pressure at the electrode edges.

5.3 Results

5.3.1 Required pulse duration for activation of denervated muscles

The average twitch force of all twenty subjects measured after one and two years increased with stimulation training for all measured pulse widths.

Maximal activation of the quadriceps muscle was achieved with 72.7 ms long stimulation pulses. This was the same for both measurements. By analysing the single twitch torque normalised with the maximum twitch torque it is easy to recognise that the characteristics did not change. That means that there was no shift to shorter pulse widths between the first and second year of electrical stimulation training (Figure 5.3). For stimulation training a pulse width in the range of about 40 ms was chosen because at this duration the muscle already produces approximately 90% of its maximal twitch force and the stimulation pulses are short enough to allow stimulation with a frequency of 20 Hz. This is crucial for generating tetanic muscle contractions.

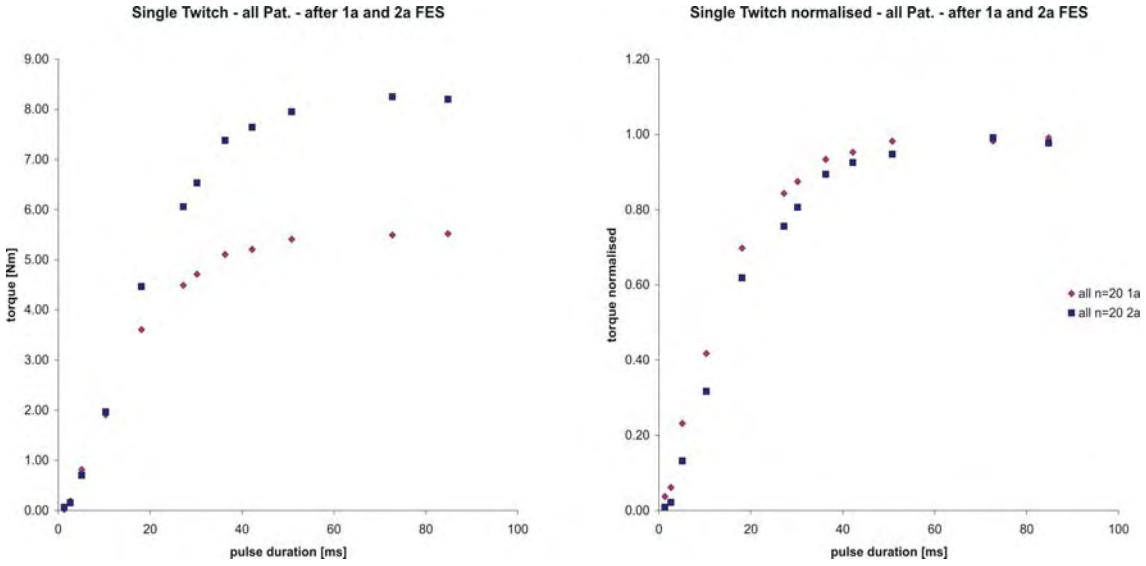


Figure 5.3: Twitch torque elicited by single pulse stimulation with varying pulse width. Left diagram is showing mean developed single twitch torque of all subjects (n=20) after one and two years of stimulation therapy. The diagram on the right side shows the twitch torque referenced to the maximum twitch torque. Measurements were performed with maximum amplitude of ±250 mA and symmetric rectangular biphasic impulses.

When viewing the data divided into the three subject groups according to their maximum tetanic muscle force at the end of the study (see chapter 5.3.2) it is interesting that all groups show the same behaviour regarding the pulse width needed for maximal twitch force. Off course the level of developed force is different. Also the increase of single twitch torque between the first and second year of electrical stimulation therapy is varying with the largest increase in the "strong" group and no change in the "weak" group (Figure 5.4).

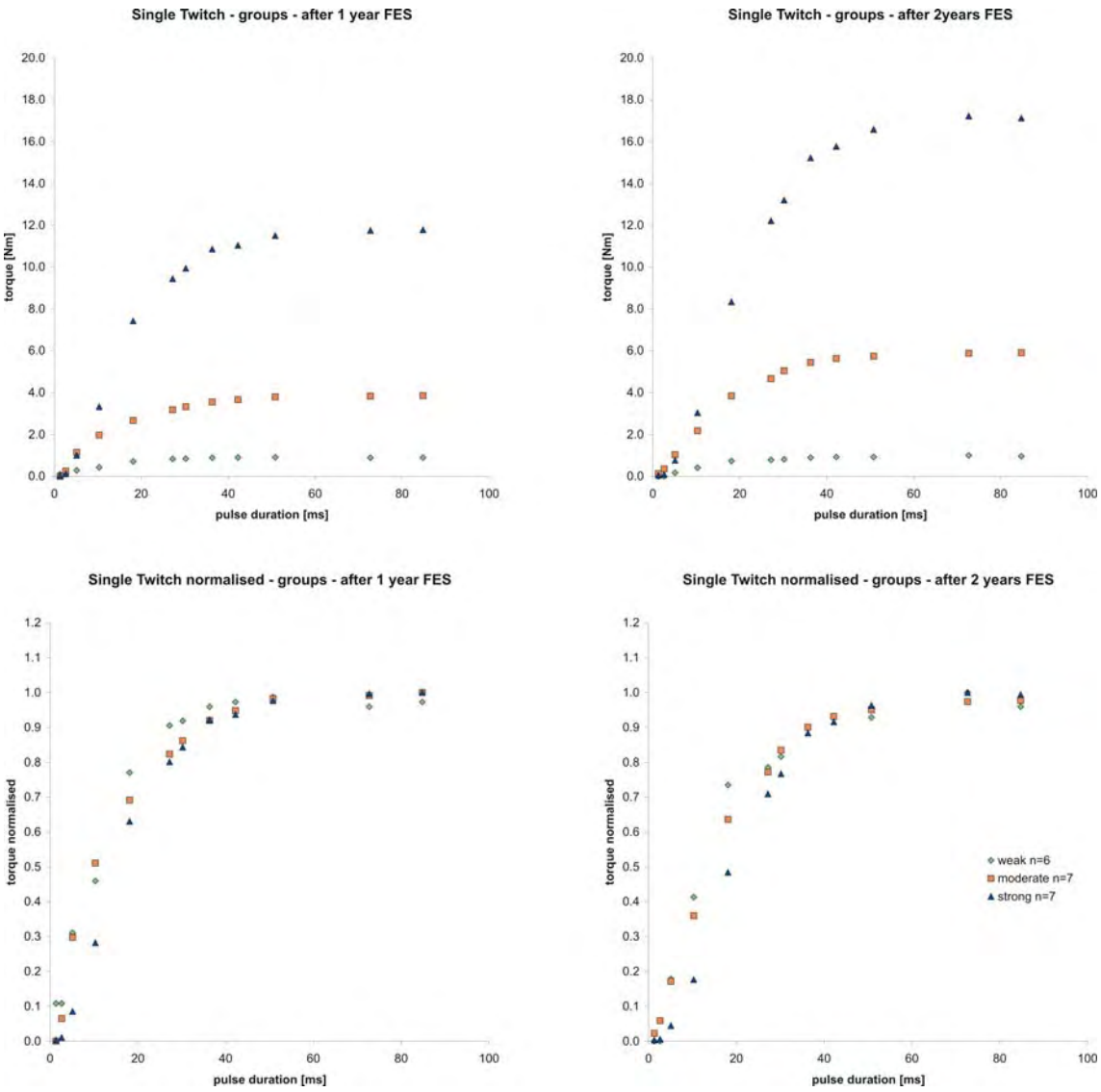


Figure 5.4: Single twitch torque measurement after one (left side) and two years (right side) of electrical stimulation training. The subjects were divided into three groups according to their maximum knee extension torque after two years of electrical stimulation training (see chapter 5.3.2). Diagrams in the top row showing the absolute value of the mean twitch torque for the three groups (weak, moderate and strong). Below the data is depicted normalised with the maximum elicited twitch torque for each group. Measurements were performed with maximum amplitude of ±250 mA and symmetric rectangular biphasic impulses.

Four subjects had additional twitch torque measurements after six month of electrical stimulation therapy. It is apparent by looking at the normalised twitch torque, that at this earlier stage of training wider pulse widths are necessary to achieve maximal contraction force from the denervated

muscle. The stimulation pulse width for generating 90% of the maximal torque was doubled compared to the measurements after one and two years of training respectively (Figure 5.5).

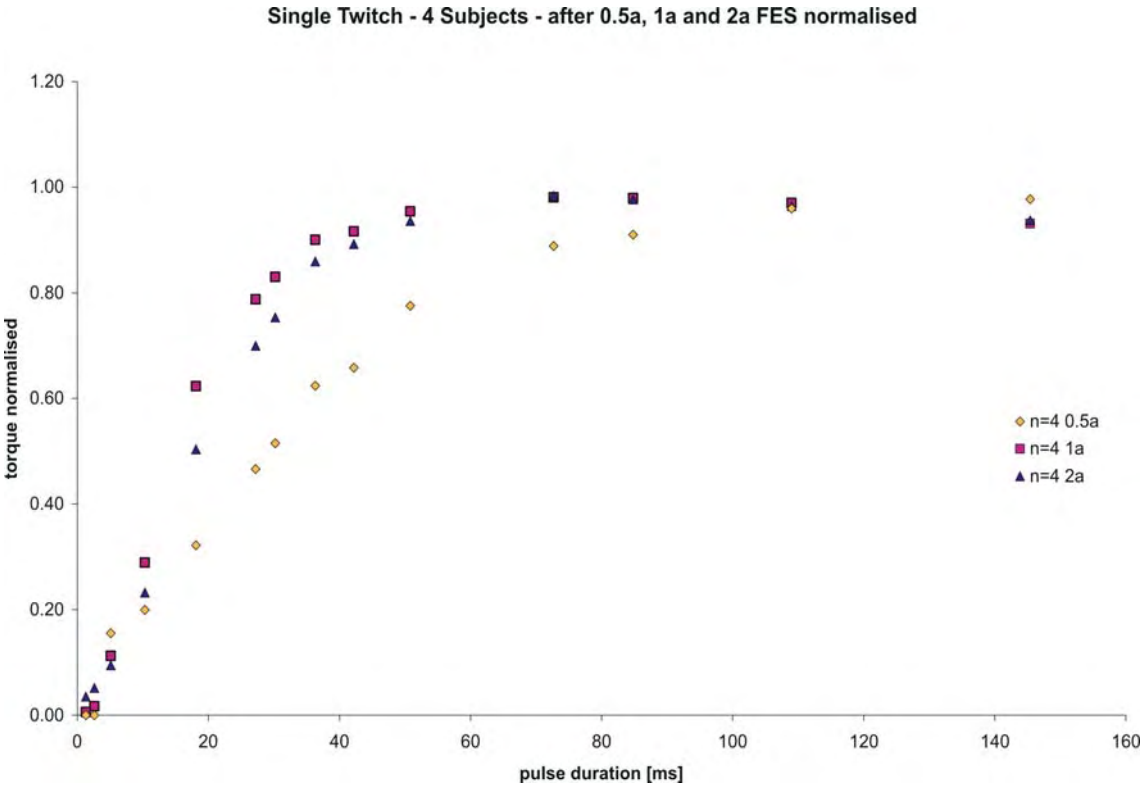


Figure 5.5: Single twitch torque measured at different pulse widths normalised by maximal elicited twitch torque, measured with a stimulation pulse amplitude of ±250 mA and symmetric rectangular biphasic impulses.

5.3.2 Increase of knee extension torque

Isometric measurement of the knee extension torque was carried out three times within the clinical study: at the inclusion into the study prior to any electrical stimulation, after one year of therapy and at the end of the study. During the first assessment the subjects showed zero or very low knee extension torque (below 1 Nm) only in three subjects a somewhat higher torque between 1 and 5 Nm was measured. This means that their quadriceps muscle could not be activated by electrical stimulation at the initial stage of training or if the muscle was activated it failed to generate mechanical output. Only some slight contraction was observed in some of the subjects when looking at the surface of the thigh. After the first year of electrical stimulation training the knee extension torque increased in three

subjects to values above 15 Nm, in seven subjects the torque was between 5 – 15 Nm and in the remaining ten almost no improvement was seen (below 5 Nm). At the end of the study in seven subjects a knee extension torque above 15 Nm (max. 30.6 Nm) was elicited by electrical activation of the quadriceps muscle. Seven subjects had an extension torque of 5 – 10 Nm and six were still under 5 Nm. The mean isometric torque for all patients is displayed in Figure 5.6. The differences in torque are highly significant ($p < 0.001$) between the measurements at the beginning and after the first year of therapy and significant ($p < 0.01$) for mean torque after the first and the second year of electrical stimulation training.

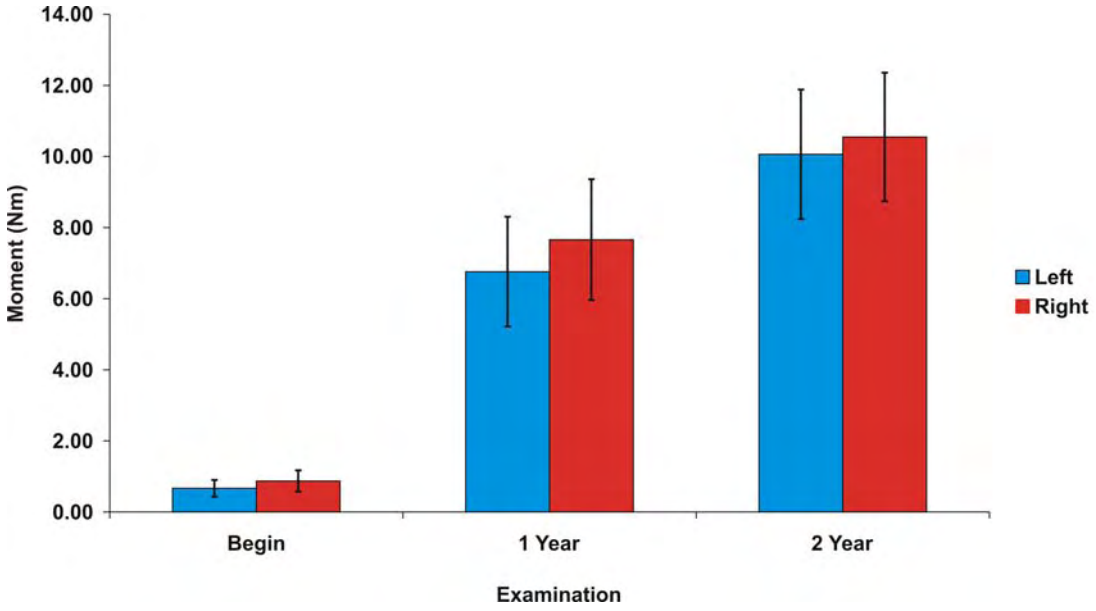


Figure 5.6: Isometric knee extension torque measured while activating the quadriceps muscle by electrical stimulation. Knee angle during measurement was 90°. Error bars indicating the standard error of the mean (SEM).

Depending on time elapsed between the injury and the start of the electrical stimulation therapy the patients showed varying degrees of improvement. Patients with a denervation time under 3 years benefited from the electrical stimulation therapy much more than longer denervated ones. But even those with long standing denervation showed slight improvements in knee extension torque (Figure 5.7).

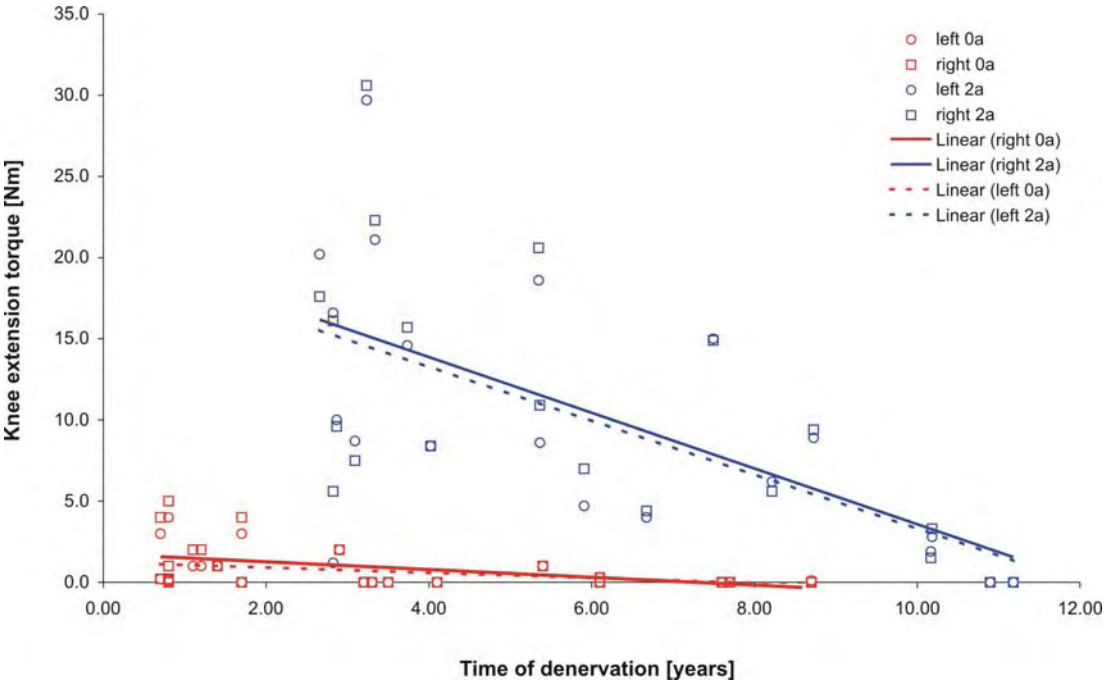


Figure 5.7: Diagram showing the maximal knee extension torque elicited by electrical activation of the quadriceps muscle. Red circles and squares at the inclusion into the study without any prior training, blue after two years of electrical stimulation therapy. Despite of the longer time since onset of the paralysis almost all subjects showed an increase in knee extension torque. It is obvious that in subjects denervated for a longer period the increase in knee extension torque was smaller than short time denervated subjects.

According to the outcome of the force measurement at the end of the study the subjects were divided into three groups. The first group of subjects with a big improvement of knee extension torque were called the "strong" responders. Patients in this group showed a knee extension torque above 15 Nm at the end of the study. The ones with a torque between 5 and 15 Nm were called "moderate" responders. All remaining subjects with a torque below 5 Nm were called the "weak" responders. In this group of patients the electrical stimulation training did not improve the force generation capacity of the quadriceps muscle. The three groups "weak", "moderate" and "strong" were used also for the following analyses of data from CT-scans, electrophysiological measurements.

5.3.3 Change in size of muscle (muscle cross-sectional area)

Muscle cross-sectional area was determined by CT scan three times throughout the study. For most of the analyses the data of the CT scan 20

cm below the trochanter major was chosen since this scan is near the maximal muscle bulk in all studied subjects. Comparison of the mean cross-sectional quadriceps muscle area of all 20 subjects at the beginning and the end of the study showed a significant ($p < 0.001$) increase from 26.1 cm² up to 37.7 cm² (Figure 5.8).

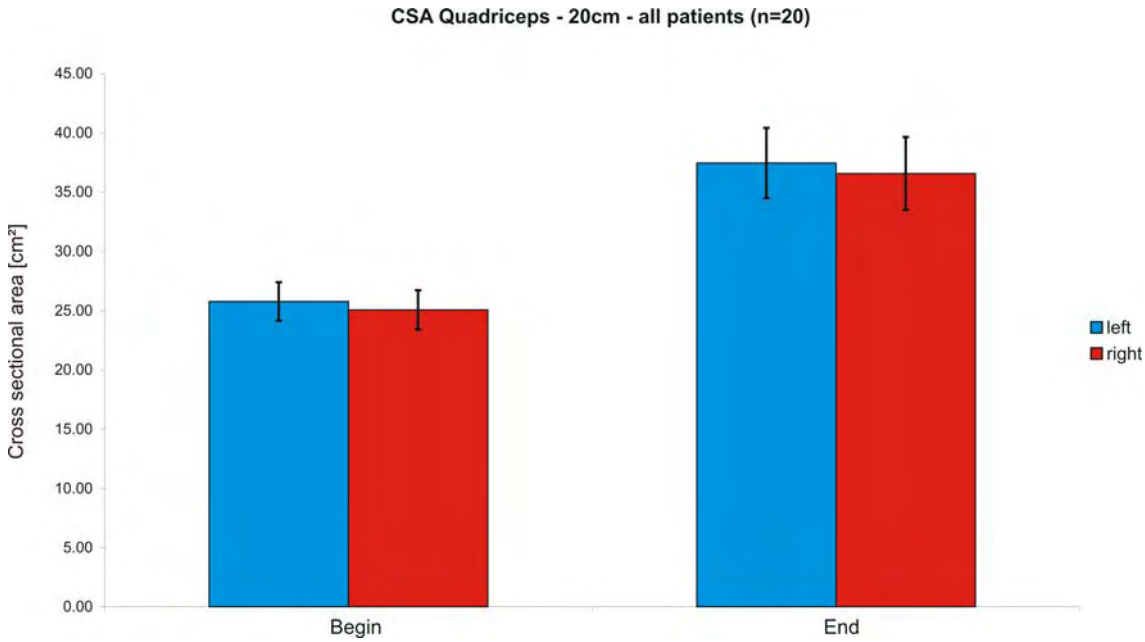


Figure 5.8: Cross-sectional area of the quadriceps muscle of all patients measured 20 cm distal to the greater trochanter of femoral bone. Measurements were taken after inclusion into the study but before commencing electrical stimulation training and after two years of training showing a mean increase in muscle cross-sectional area of 45.6%. Error bars are indicating ± SEM.

When splitting the patients into the three groups according to the outcome of the torque measurements (weak, moderate and strong group), the results of the regarding the muscle cross-sectional area show similar characteristics. The group with no or almost no improvement in developed muscle force also exhibits only a slight, not significant ($p > 0.05$) increase of muscle cross-sectional area from 22.3 cm² up to 26.1 cm². Whereas in the other two groups the size of the muscle bulk increases significantly, with the largest change in the "strong" group from 30.6 cm² up to 51.3 cm² (Figure 5.9).

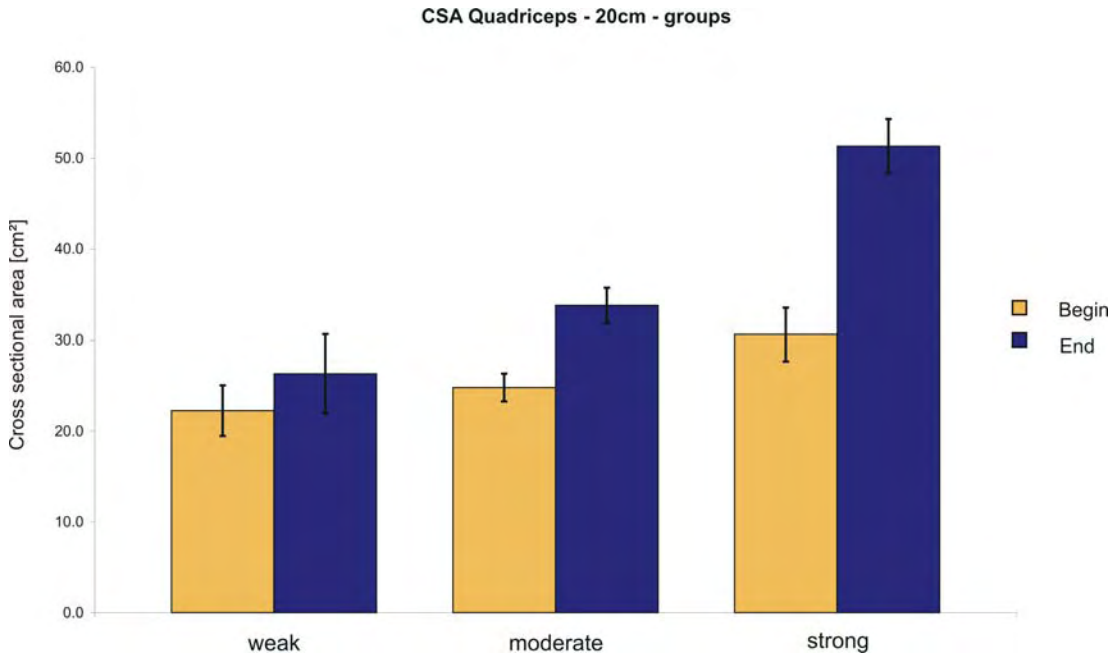


Figure 5.9: Increase of quadriceps muscle cross-sectional area after two years of electrical stimulation training. Grouping of the patients ("weak", "moderate" and "strong") according to the knee extension moment measured at the end of the study. Changes are significant for the groups "moderate" ($p < 0.01$) and "strong" ($p < 0.001$). Values are means ± SEM.

Although the electrical stimulation therapy was focused on the quadriceps muscle with the electrodes attached to the front side of the thigh the hamstrings muscles were activated by the applied electrical stimuli also. This effect is proved not only by the palpable contraction in these muscles but also in the increase of muscle cross-sectional area measured by CT scans. Comparing the hamstrings muscles area before commencing the stimulation training and at the end of the study a significant ($p < 0.001$) increase from 24.1 cm² to 29.1 cm² was detected (Figure 5.10).

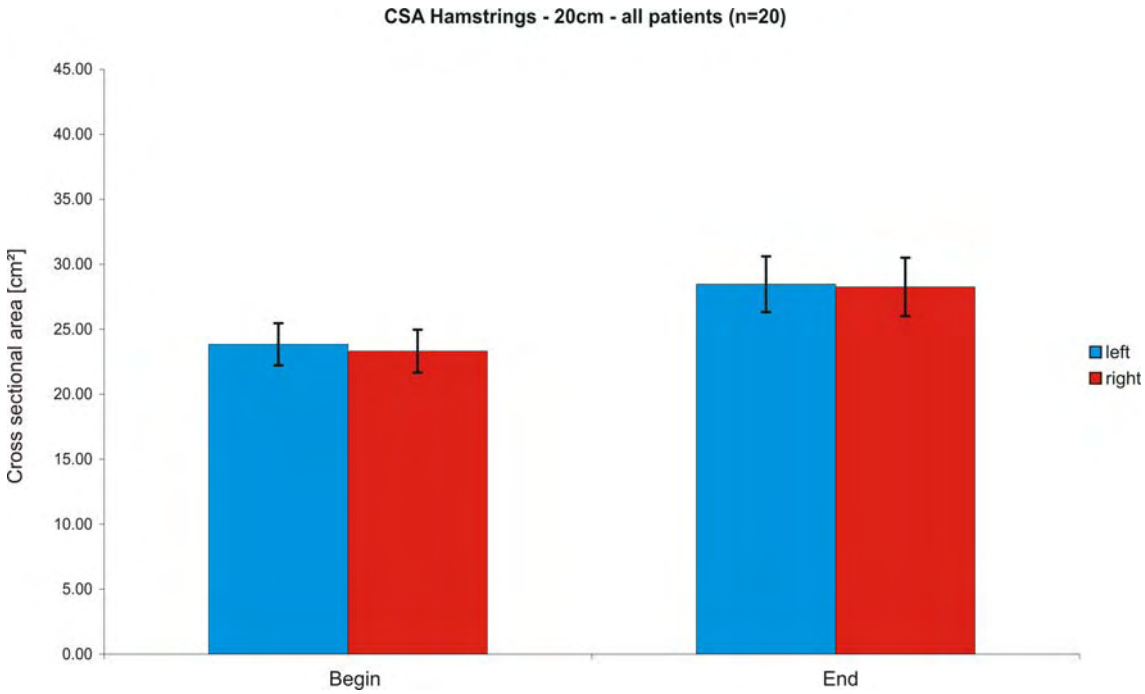


Figure 5.10: Cross-sectional area of the hamstrings muscles of all patients measured 20 cm distal to the greater trochanter of femoral bone. Measurements were taken after inclusion into the study but before commencing electrical stimulation training and after two years of training showing an increase in mean muscle cross-sectional area of 20.3%. Error bars are indicating ± SEM.

The muscle cross-sectional area of the quadriceps in the CT scans at the level 10 cm below the greater trochanter changed significantly from 18.0 cm² to 21.8 cm² whereas the hamstrings muscles area remained unchanged 31.2 cm² before and 31.0 after stimulation training

At the level 30 cm below the greater trochanter significant increases in cross-sectional area were found for both quadriceps and hamstrings muscles. The area of the quadriceps was 23.2 cm² before and 32.3 cm² after training and the hamstrings muscle area increased from 17.7 cm² to 22.4 cm².

All data regarding the muscle cross-sectional area at the different CT scan levels before and after the stimulation training is summarized in table 5.2.

Patient group		all (n=20)	weak (n=6)	medium (n=7)	strong (n=7)
Quad Area [cm²]					
Scan level	FES				
10cm	before	18.03 ± 1.36	15.11 ± 2.24	16.39 ± 1.76	22.17 ± 2.34
	after	21.81 ± 1.91 †	15.26 ± 2.11	18.44 ± 0.88	30.80 ± 2.77 ‡
20cm	before	26.07 ± 1.57	22.26 ± 2.79	24.80 ± 1.53	30.61 ± 2.95
	after	37.69 ± 2.94 ‡	26.13 ± 4.34	33.79 ± 1.94 †	51.34 ± 3.01 ‡
30cm	before	23.23 ± 1.70	19.99 ± 3.13	21.75 ± 2.80	27.49 ± 2.53
	after	32.34 ± 2.48 ‡	25.52 ± 2.75 *	29.22 ± 2.45 *	41,32 ± 4.76 †
Ham Area [cm²]					
Scan level	FES				
10cm	before	31.18 ± 2.25	23.66 ± 2.28	29.94 ± 2.73	38.86 ± 4.06
	after	30.97 ± 1.94	25.28 ± 1.71	27.58 ± 2.41	39.22 ± 2.89
20cm	before	24.14 ± 1.60	19.77 ± 1.47	22.80 ± 1.65	29,24 ± 3.42
	after	29.10 ± 2.15 ‡	22.37 ± 2.11	25.62 ± 2.14	38,36 ± 3.38 ‡
30cm	before	17.70 ± 1.37	14.52 ± 1.79	17.38 ± 1.46	20.74 ± 3.08
	after	22.41 ± 2.01 †	16.45 ± 2.20	18.93 ± 2.00	30.99 ± 3.09 *

Table 5.2: Muscle cross sectional area at different CT scan levels before and after training for all patients and divided into the groups according to maximum developed knee extension torque after 2 years of FES. Significant differences between pre and post training area are denoted *, † and ‡ for $p < 0.05$, $p < 0.01$ and $p < 0.001$, respectively.

5.3.4 Change in tissue composition

Additional information about the condition of the denervated muscle apart from the muscle cross-sectional area was obtained by analysing the attenuation of the X-ray radiation in the CT scans. The quadriceps and hamstrings muscles in the thigh cross-section were marked manually. Afterwards the CT number or Hounsfield Unit (HU) of each pixel within the selected region of interest (ROI) was determined and the mean density within the ROI as well as a histogram showing the density distribution was calculated. In order to prepare a histogram with the mean density distribution of all patients, the data for each patient was normalized with the area of the corresponding ROI.

Analysing the quadriceps muscle of all patients the mean density was significantly ($p < 0.01$) increased by the stimulation training. Splitting into the different groups according to their force output reveals a slight not significant ($p > 0.05$) decrease of density in the patients with no measurable force output after 2 years of FES in contrast to a significant ($p < 0.05$) and highly significant increase ($p < 0.001$) for the moderate and strong group, respectively (Figure 5.11).

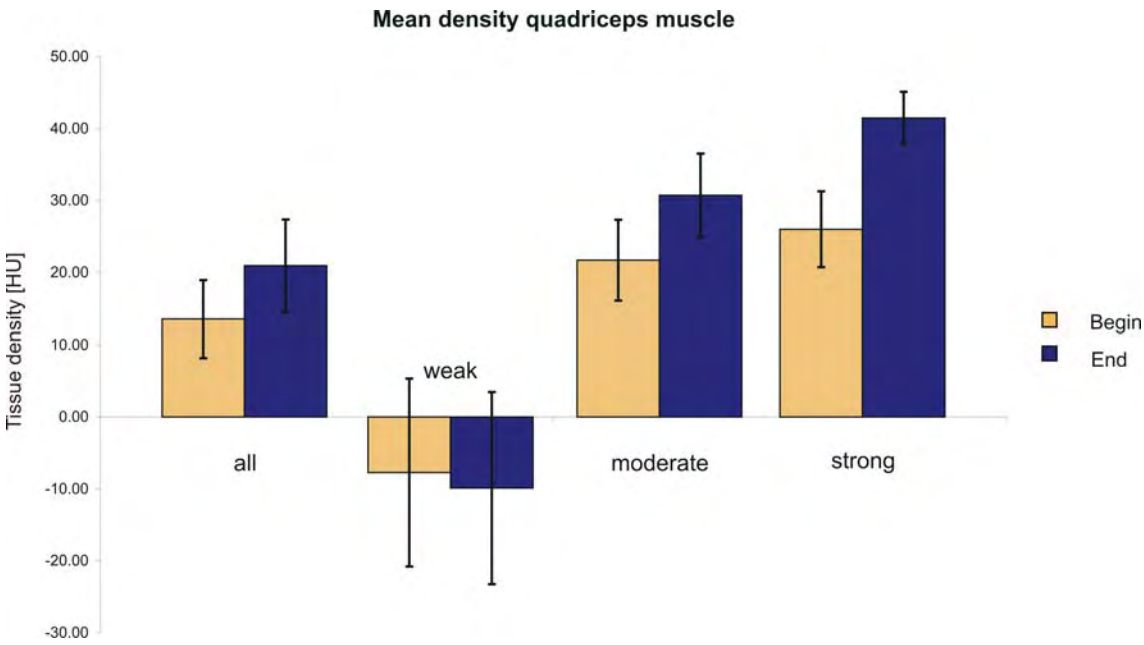


Figure 5.11: Mean density of the quadriceps muscle before and after 2 years of FES training of all patients (n = 20) and grouped according to the outcome of the force measurements (weak n = 6, moderate n = 7, strong n = 7). Bars are indicating ± SEM.

Figure 5.12 (A & B) shows the histogram of the quadriceps muscle density 20 cm below the greater trochanter before (Figure 5.12 A) and after two years (Figure 5.12 B) of electrical stimulation training. Comparing the two histograms it is evident that the distribution is shifted to the right side denoting an increase of more dense tissue above 40 HU and decrease of tissue with lower densities below 40 HU (Figure 5.12 C). This indicates a reduction of tissues (e.g. fat, collagen) with lower CT density in the examined muscles.

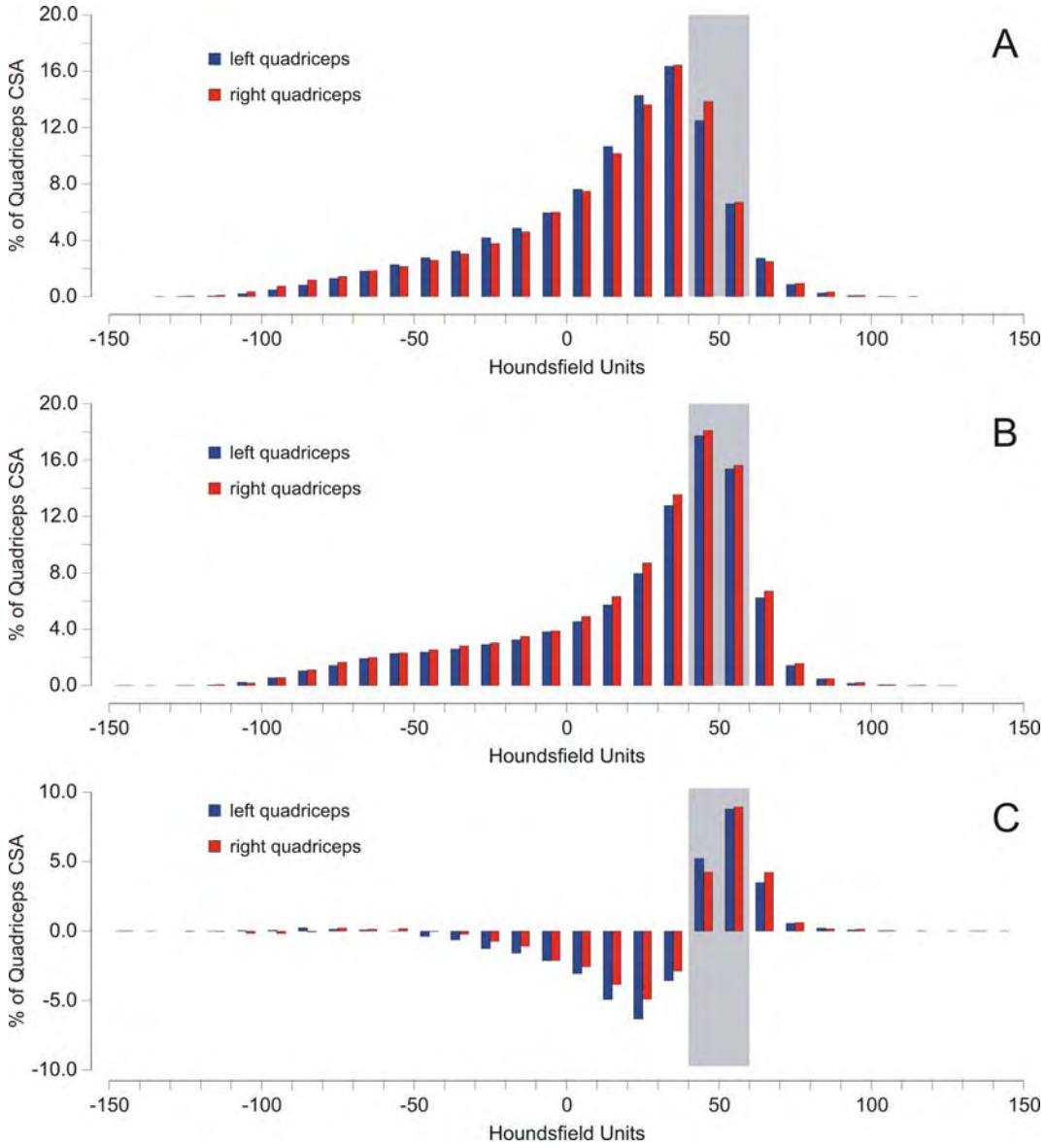


Figure 5.12: Histogram showing the CT density distribution of the quadriceps muscle 20 cm below the greater trochanter (mean of all patients n = 20). (A) Before starting the electrical stimulation training. (B) After two years of electrical stimulation training. (C) Histogram showing the difference in the density distribution before and after training. Grey shaded areas marking the HU range of healthy muscle.

The muscle density distribution in the three patient groups weak, moderate and strong showed differences in the shape. At the beginning of the study the group (weak) of subjects developing almost no knee extension torque exhibited a flat distribution without a distinct peak. Whereas in the other two groups (medium, strong) a distribution with a peak in the range of 30 – 40 HU was found (Figure 5.13).

After two years of FES training the density distribution in the weak group was similar to the beginning. In the strong and moderate group the distribution was shifted to the right side equivalent with an increase of tissue with high and decrease of tissue with low density. This indicates a rise in content of muscle fibres within the analysed muscle area and a reduction of fat and collagen. Since the strong and moderate group exhibited very similar behaviour only the distribution of the strong group is displayed in Figure 5.13.

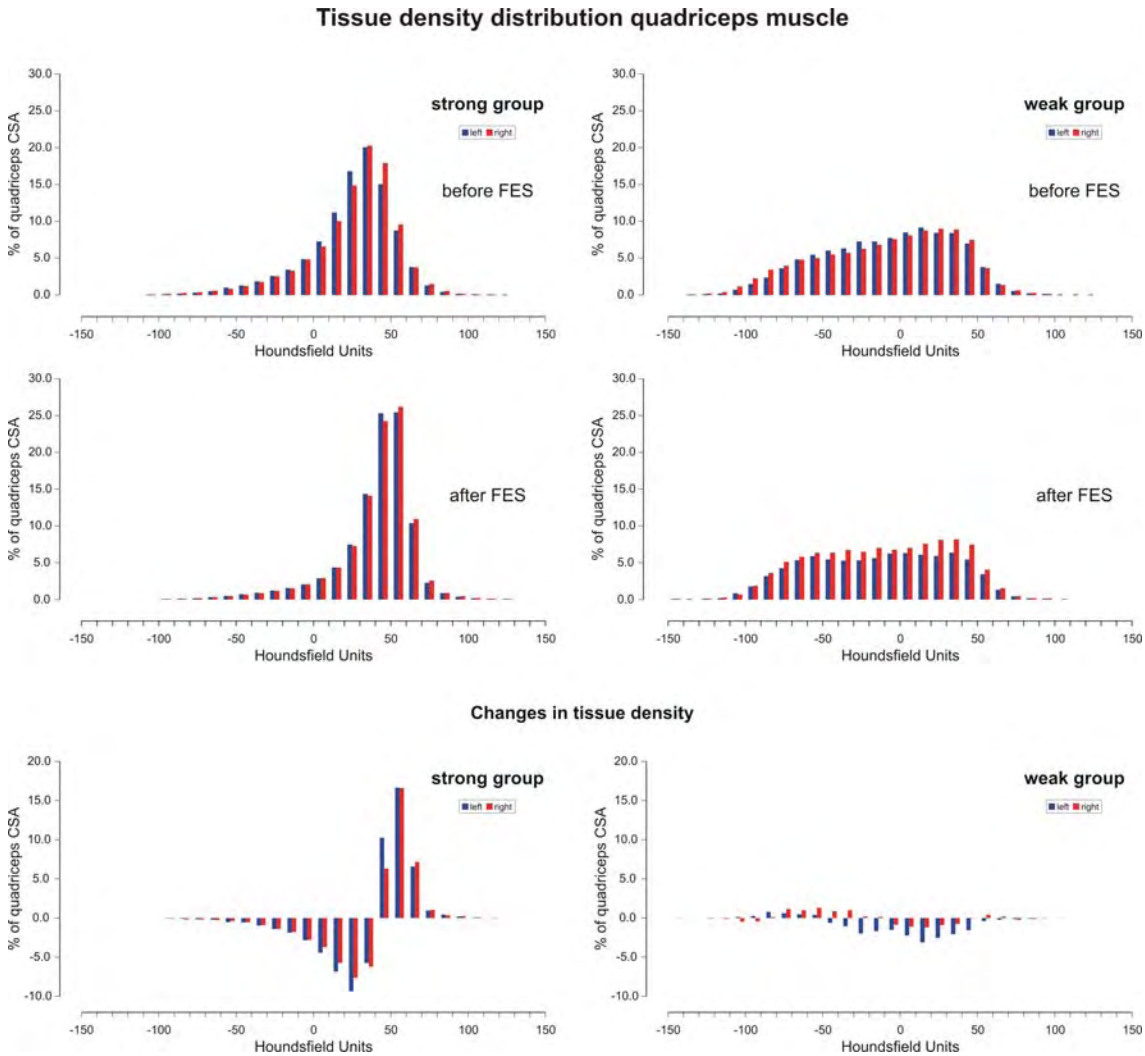


Figure 5.13: Histograms showing the CT density distribution of the quadriceps muscle 20 cm below the greater trochanter in patients with big improvement (left column: strong group n = 7) in muscle function and with in patients with no response (right column: weak group n = 6). Top and central histograms are showing the distribution before starting and after two years of electrical stimulation training, respectively. The bottom histograms are showing the difference in the density distribution before and after training.

5.3.5 Electrophysiological changes (MFCV, ISI)

In three patients MFCV and ISI was measured at the time of inclusion into the RISE study without prior FES training and again after two years of FES training according to the therapy protocol. Two of them with a relatively short period of denervation of approximately one year had mean MFCV of about 2 m/s. The patient which was denervated for 3.2 years had a considerably slower mean MFCV of 1.0 m/s. The shortest ISI with response was 4 ms for the one year denervated and 8 ms for the 3.2 years denervated patient. After two years of FES therapy the differences in the electrophysiological parameters MFCV and ISI between the three patients disappeared (Table 5.3).

Subject			S1	S2	S3
Denervation Time		yrs	0.8	1.2	3.2
before FES	MFCV				
	mean	m/s	2.14	1.97	1.00
	max.	m/s	2.34	3.65	1.18
	min.	m/s	1.53	1.12	0.77
	ISI	ms	4.0	4.2	8.0
after FES	MFCV				
	mean	m/s	2.44	2.26	2.22
	max.	m/s	3.19	3,20	3.05
	min.	m/s	1.77	1,02	1.59
	ISI	ms	3.3	3.2	3.0

Table 5.3: MFCV and shortest ISI of the three subjects assessed at inclusion into the RISE study and after two years of FES training. Denervation time is defined as the period between onset of denervation and start of FES training.

Analysing the MFCV of the thirteen patients which were examined after one year and again after two years of electrical stimulation training, they showed a significant ($p < 0.05$) increase in conduction velocity for mean and maximum MFCV and a slight but not significant increase for the minimum MFCV (Figure 5.14).

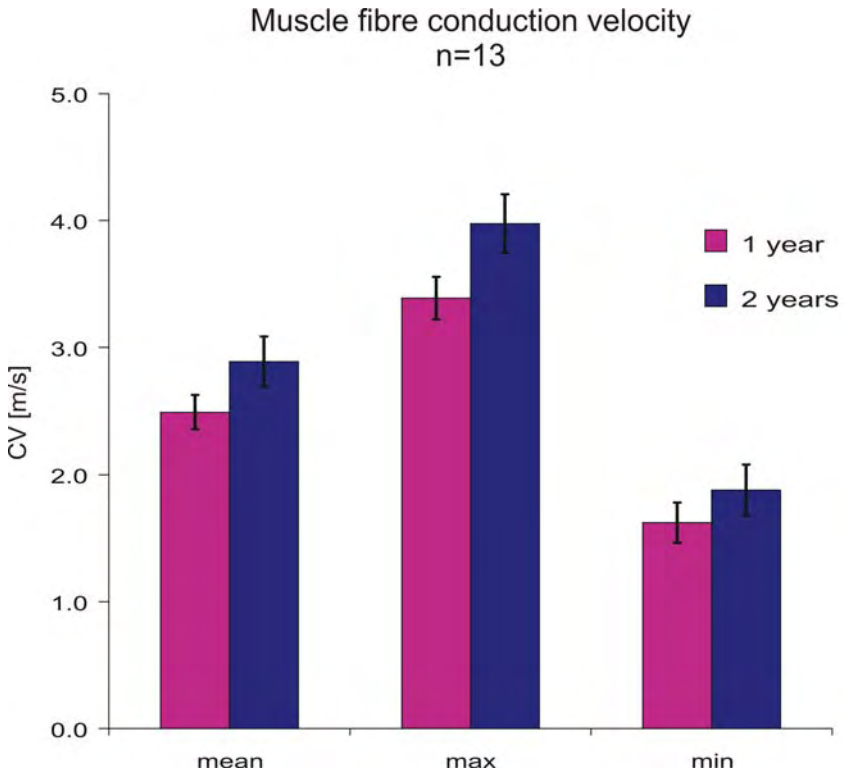


Figure 5.14: MFCV measured in the quadriceps muscle of thirteen patients after one and again after two years of electrical stimulation training. The differences are significant for mean and maximum MFCV ($p < 0.05$). Error bars are indicating ± SEM.

Pooling all measured muscle fibres after one and two years of electrical therapy and calculating the distribution of the CV confirms this finding (Figure 5.15). Even after two years of training there are still some very slow conducting muscle fibres (CV < 1.0 m/s) detected which explain that the minimum MFCV is not altered severely. Otherwise there is an apparent increase of fast conducting fibres responsible for the significant increase in the mean (from 2.5 to 2.9 m/s) and maximum MFCV (from 3.4 to 4.0 m/s).

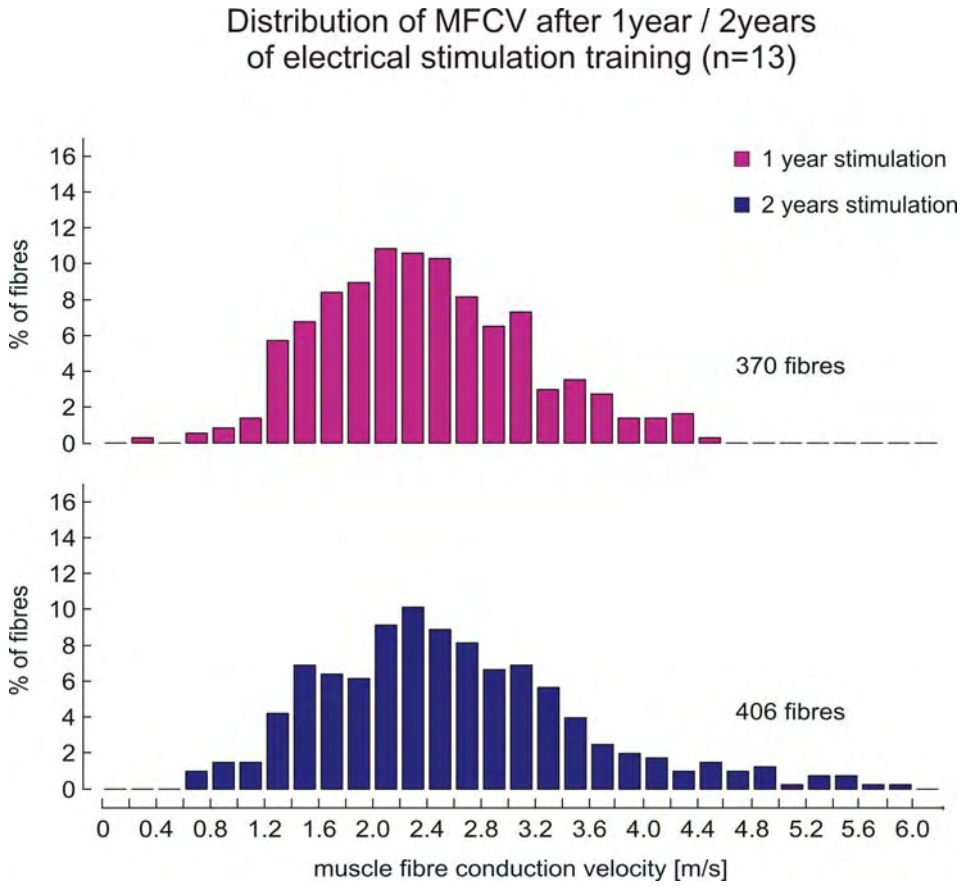


Figure 5.15: Distribution of MFCV of all measured muscle fibres in the patients which were measured after one and two years of stimulation training. Data is normalised with the number of measured fibres.

When examining the data acquired at the end of study divided into the three groups according to the maximal tetanic force developed, the most obvious difference between the three groups is, that the distribution of MFCV lost the Gaussian shape in the "weak" and "moderate" group (Figure 5.16). Compared to healthy muscle the distribution of the "strong" group is of similar shape but shifted to the left which means that these muscles consist of slower conducting fibres.

The mean conduction velocity is 2.2 m/s for the weak group and 2.5 m/s for the other two groups with a smaller standard deviation of ±0.8 m/s compared to ±1.3 m/s for the "strong" group. It is evident that by analysing only the mean MFCV the differences between the three groups are not characterized well.

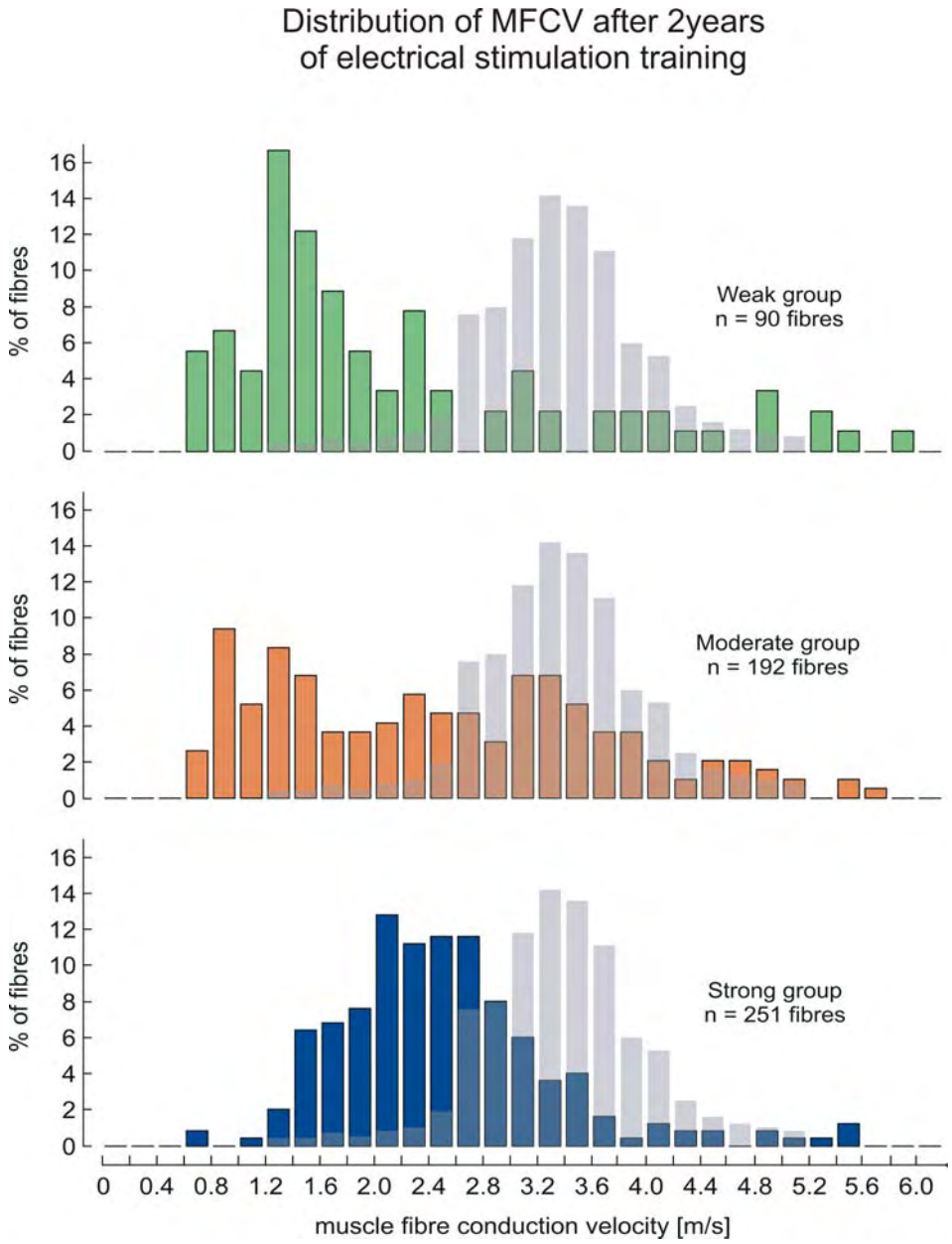


Figure 5.16: Distribution of the MFCV of all measured muscle fibres at the end of the study. Shaded grey distribution in the background shows the data of healthy subjects measured by Cruz-Martinez 1989. The shape of the distribution of the strong group is almost Gaussian again whereas the other two groups are showing a widespread distribution of MFCV without any distinct shape.

Regarding the muscle fibre refractory period the quadriceps muscle of the patients exhibits a slight decrease of shortest ISI (Figure 5.17). This reduction in shortest ISI from 4.7 ms after the first year of electrical

stimulation training down to 3.9 ms after the second year was not significant.

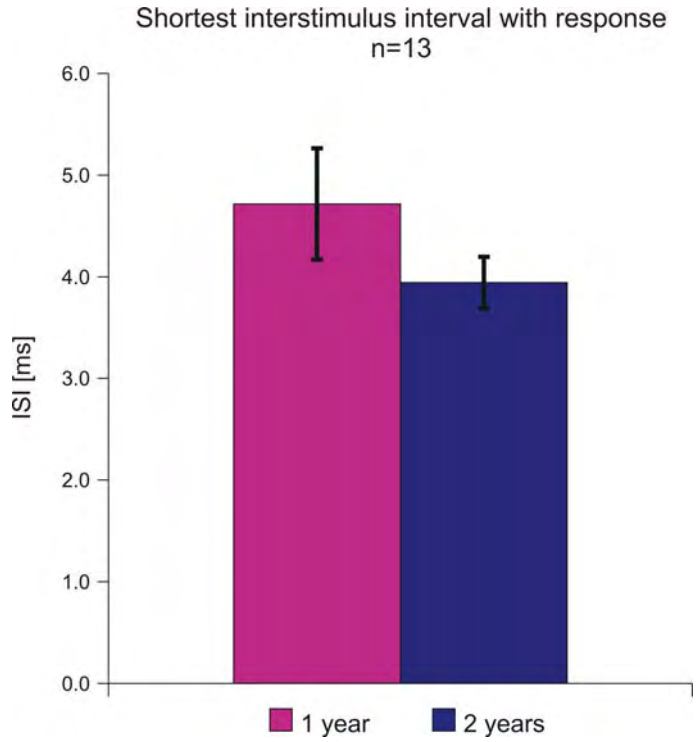


Figure 5.17: Changes in the muscle fibre refractory period between the first and second year of electrical stimulation training. Error bars are indicating ± SEM.

5.3.6 Functional outcome

At the end of the study 9 patients were able to perform full knee extension while sitting on a bed or chair with the quadriceps muscle activated by electrical stimulation. This test was conducted without additional weight on the ankle, lifting the lower leg against gravity to horizontal position. None of the patients was able to perform full knee extension at the time of inclusion without any prior stimulation training.

Five patients were able to perform standing-up and sitting down exercises with their electrically activated thigh muscles supported by their arms using specially developed training aids. Moreover three of them were already able to step on place simulating the swing phase of gait by alternately switching on and off stimulation of the right and left leg respectively. During this the

patients stood in a standing frame using their arms for keeping balance and partially supporting their body weight. After some exercises in the standing frame two subjects were also able to walk with special mobile parallel bars. The distance covered was ranging from 6 to 37 meters. Stimulation of the quadriceps muscle is controlled by a switch mounted on the supporting device operated by the patient with the fingers of the left or right hand. For bringing the leg forward during the swing phase the stimulation of the quadriceps muscle on the corresponding side is turned off and the leg is moved forward by elevating and rotating the pelvis with the intact trunk muscles. Since the patients are paralysed completely there is no sensory feedback from the lower limbs for this reason the foot is positioned under visual control and the balance is kept by the subject's arms (Figure 5.17 A - D).

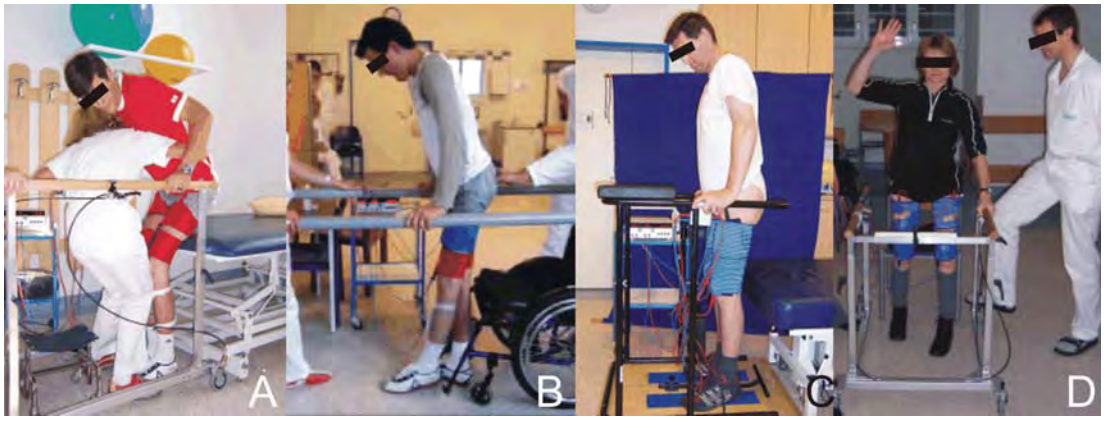


Figure 5.17: (A) First stand-up trial with electrical stimulation training of m. quadriceps. (B) Standing in parallel-bar with supported electrical stimulation of m. quadriceps. (C) Stand-up training performed in a standing-frame, switching of loading between right and left leg. (D) Patient walking with electrically activated quadriceps muscles supported by special mobile parallel bars.

6 Discussion

Stimulation equipment. With the currently applied training protocol it is necessary for the patient to perform stimulation therapy 5 times a week (e.g. 2 days stimulation, 1 day pause, 3 stimulation, 1 day pause). Therefore a device suited for home-based therapy was built.

The concept of a microprocessor controlled stimulation device with preset programmes individually adapted to the patient's condition proved to work well. None of the patients had troubles with operating the device and all were satisfied with the reliability. The only point for critics was the battery life time since after about one year of use, the capacity dropped and it was not possible to perform the whole daily training programme with a fully charged device. This rather fast drop in capacity was ascribed to the high impulse currents drawn by the stimulator. Hence, the battery pack had to be replaced by a more stable pack made of high performance cells after the first year of the study during the follow up examination at the Wilhelminenspital.

Overall the developed stimulators are adequate for therapy of denervated, degenerated muscles. The range of stimulation impulse width is wide enough to ensure sufficient stimulation for all stages of denervation induced atrophy and degeneration. Nevertheless the applied therapy protocol is very time consuming and calls for considerable commitment from the patient. Even with the two channel device the daily time expenditure for FES therapy of three different muscle groups per limb (gluteus, quadriceps and calf muscles) exceeds two hours (3 times 30 minutes and time for donning and doffing).

In three patients with long-standing denervation, thus a high amount of intramuscular fat and connective tissue as well as a thick subcutaneous fat layer (Figure 6.1), it appears that the stimulation intensity was not sufficient to increase their muscles cross-sectional area and improve the ability to produce force, although they were compliant and regularly carried out the training program.

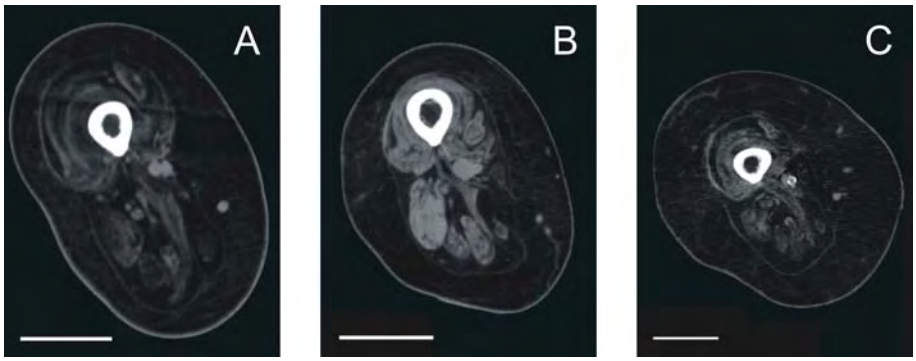


Figure 6.1: CT scans of three patients not responding the FES therapy. Displayed is the cross-section of the right thigh 20 cm below the grater trochanter at the time of inclusion into the clinical study. Patients were denervated for (A) 7.7, (B) 8.7 and (C) 8.7 years respectively. Bars 5 cm.

A further increase of stimulation intensity above the present limit of ±250 mA is problematic, since it is very difficult to safely apply such high currents. Even small changes in the contact resistance between electrode and skin could lead to high local current densities causing skin burns.

Main requirements for the next generation of stimulation equipment for patients with denervated lower limb muscles are therefore, stimulators with more than two (four or six) independent output channels and e.g. segmented safety electrodes with built in circuitries for controlling the current density. Benefits of such a multi-channel system would be a reduction in time expenditure for training since all relevant muscle groups could be treated simultaneously and better coordinated more natural movement during functional applications such as standing up and walking. Electrode garments with integrated electrodes and cables would simplify donning and doffing, resulting in further reduction of training time and simultaneous improvement of safety by preventing inappropriate connections and electrode placement, as well as providing more evenly distributed contact pressure.

The suggested safety electrodes in combination with circuits for control of the applied current density should provide safe application of the even higher maximum current intensities than our present limit, which could provide new prospective for patients not benefiting from the currently applied protocol.

Stimulation protocol, excitability. Defining an optimal therapy protocol and stimulation parameters is limited by several constraints.

The current daily training time per muscle or muscle group of 30 minutes has to be seen as upper limit. This seems to be not very long, but for treating two or more muscle groups per limb the total therapy time per day is 2 hours or more, as already mentioned above. Longer stimulation times would therefore interfere too much with the patients' life, since many of them have a profession.

The stimulation intensity is limited by the skin's capacity to conduct the stimulation currents without getting damaged. With the currently applied equipment a maximum impulse current of ±250 mA is the limit for safe application of FES training. In fact this limit is sometimes critical, if the electrodes are not properly applied or peel off during the stimulation training. The new safety electrode developed by Mayr (2007) with a special geometry preventing high current densities at the edges partially solves the problem of applying high stimulation currents safely (Figure 5.2)

Excitation of the denervated muscles needs long duration impulses, this limits the maximal stimulation frequency. However, since the denervated muscles exhibit slower contraction characteristics, as described in animal studies on rats (Carraro et al. 1982, Al-Amood and Lewis 1989) and rabbits (Ashley et al. 2007), even with the reduced stimulation frequency a fused tetanus is achieved after a conditioning training phase.

Experiences made in preliminary studies showed, that even in well trained patients, treated with FES for many years, biphasic rectangular pulses with a minimum pulse width of 40 ms are necessary to activate the denervated muscle (Kern 1995). The results of the single twitch measurements carried out during the RISE clinical study demonstrate that after two years of FES therapy a pulse width of 40 ms is necessary to elicit twitches developing 90% of the maximum twitch torque in the examined muscle. The maximum twitch torque of the quadriceps muscle was determined by applying impulses with maximum intensity of ±250 mA and increasing pulse width from 1.3 ms up to 145 ms. When looking at the data normalised to the maximum twitch torque no difference regarding the necessary pulse width for activation of the muscle can be found between the three patient groups "strong", "moderate" and "weak", although there is a huge difference in the level of developed twitch force (Figure 5.4). With the applied training protocols and stimulation equipment a further reduction in the required pulse width for 90% twitch torque is not expected, as the muscles already

showed the same behaviour after the first year of training (Figures 5.3). This leads to the assumption that an increase in excitability of the denervated muscle takes place during the first year of FES therapy. Measurements made in four patients after the first 6 months of training confirm this assumption, because at this earlier stage training pulses with a duration of about 80 ms were necessary to elicit 90% of the muscle maximum twitch torque (Figure 5.5). Despite of the reduction of the required pulse duration for activation in any case the denervated muscles remained still much less excitable than innervated muscles.

With the above described limitation regarding the required pulse width, the inter-pulse interval was reduced to 10 ms in order to attain a minimum stimulation frequency of 20 Hz. This is necessary for eliciting fused tetanic contractions to induce a strong muscle fibre tension, constituting a hypertrophy stimulus in the denervated muscle fibres.

Taking into account all constraints the resulting stimulation parameters for ES training were: pulse current up to ±250 mA; biphasic rectangular pulses with 40 ms pulse duration and 10 ms inter-pulse interval; stimulation burst duration of 2 s and 2 s pause between the bursts; total training time per muscle or muscle group 30 minutes.

Assessing the denervated muscle:

Systematic stimulation of the denervated muscle artificially generating contractile activity enhanced the capacity to generate force, increased the cross-sectional area as well as the density in CT-scans and improved the electrophysiological properties (excitability, MFCV, MFRP) of the muscle.

Force generation. At the beginning all of the studied patients were not able to produce a maximum isometric knee extension torque above 5 Nm at 90° knee angle. Stimulation brought about a significant increase of torque to 10.3 Nm (average over all studied patients) with a measured maximum torque of 30.6 Nm in one male and 19.6 Nm in one female patient. This is still only approximately 10 % of the maximum voluntary knee extension torques developed by healthy, young individuals (Maughan et al. 1983, Narici et al. 1996). One reason for low the value can be found in the torque measurement procedure for assessing the quadriceps muscle. The applied stimulation parameters induce an electrical field which is not only confined to the knee extensor muscles but also activates part of the hamstrings muscles simultaneously. Therefore, the resulting knee extension torque measured with the knee dynamometer is reduced by antagonistic co-

contraction (Figure 6.2). To quantify the influence of co-activating the hamstrings muscles a new measurement technique based on a pendulum test was designed (Gallasch et al. 2005). The future results gained by this method will help to evaluate the impact of the hamstrings muscles on the torque measurement.

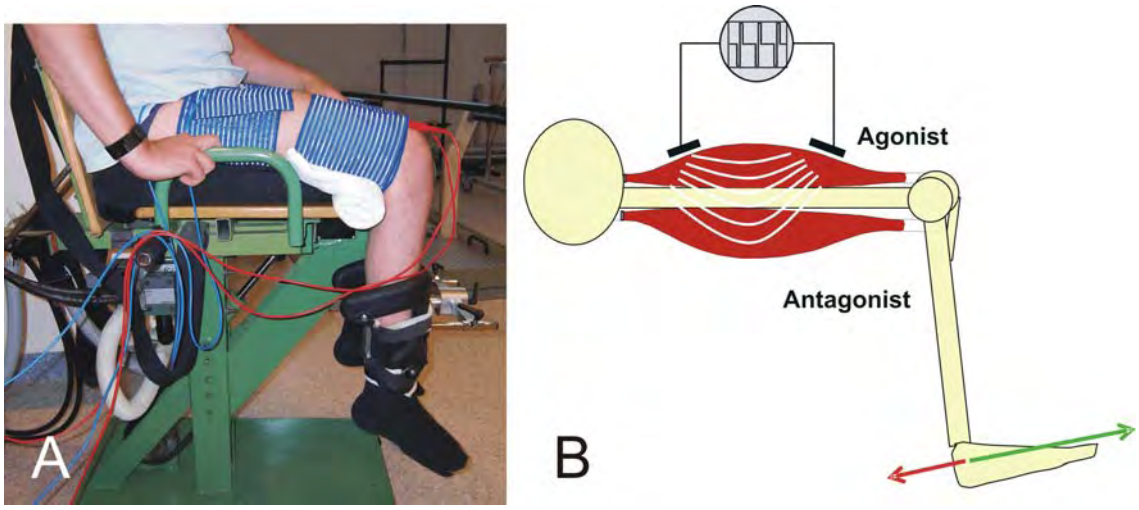


Figure 6.2: Measurement of knee extension torque. (A) Picture showing the subject placed on the knee dynamometer. Electrodes attached to the front of the thigh. (B) Drawing illustrating electrical field distribution in the thigh. Since the electrical field is spread also to the hamstring muscles (antagonists) the measured torque is reduced. The green arrow indicates the extension torque generated by the quadriceps muscle and the red arrow the torque produced by the hamstrings. With the dynamometer only the resulting knee torque is detected.

However, the co-activation of the antagonists can not be hold responsible alone for the very low measured torque. Results of an animal study conducted in the scope of the RISE project demonstrated that the maximum tetanic force of denervated-stimulated muscle was only half that of innervated control muscles. The reduced capacity to generate force was ascribed to incomplete restoration of muscle cross-sectional area and the proportion of that area occupied by muscle fibres, since the specific tension in maximal tetanic contractions measured in those muscles was not different from their innervated control (Ashley et al. 2008).

Cross-sectional area. Analysing the data acquired from CT scans showed a significant improvement in the cross-sectional area of the denervated-stimulated quadriceps muscle with the largest increase (+ 68 %) in the "strong" group (Figures 5.8 & 5.9). The area measured at the end of the study corresponds to 45 % and 64 % of healthy subjects for mean of all

studied patients and mean of patients belonging to the "strong" group, respectively (healthy reference: Bulcke et al. 1979, Termote et al. 1980, Maughan et al.1983). It seems that the FES therapy was more effective in restoring the muscle size than the capacity to generate force, which is in line with the outcome of the rabbit study (Ashley et al. 2007).

Nevertheless a correlation between the cross-sectional area of the quadriceps muscle and the developed knee extension torque is observed (Figure 6.3).

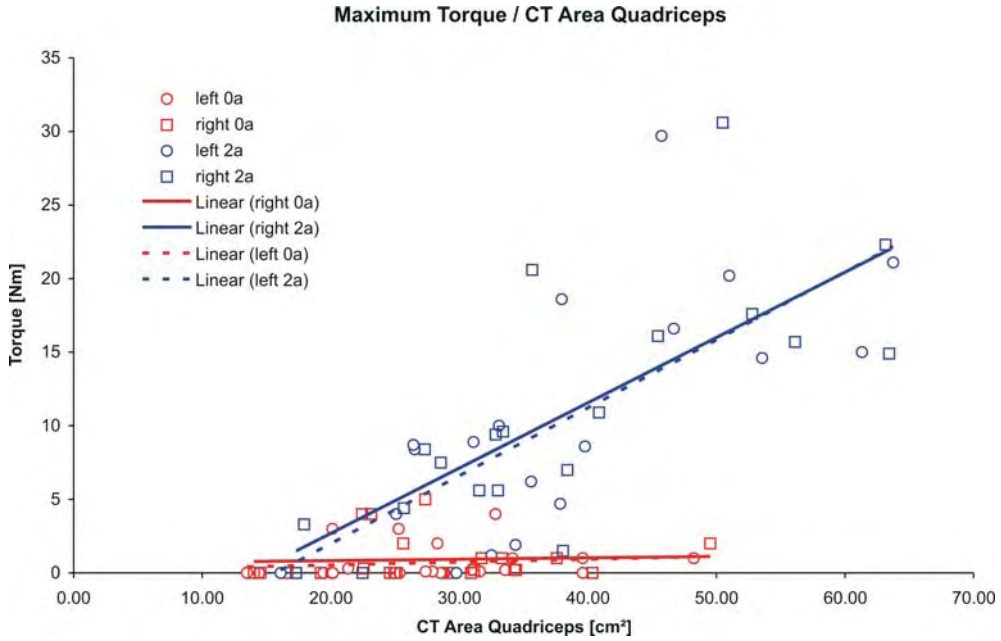


Figure 6.3: Diagram showing the relation between developed knee extension torque against the muscle cross-sectional area for left and right limb at the beginning and after 2 years of FES therapy (n = 20). Linear regression for the 2 year measurement: left $r = 0.71$, $p < 0.001$; right $r = 0.75$, $p < 0.001$.

CT density, density distribution. However, the mean density of 21.1 HU in the quadriceps muscles of all patients after two years of FES training was considerably lower compared to healthy innervated muscle with a density of 50 - 60 HU, leading to the assumption that within the analysed area the content of muscle fibres is reduced. This result derived from the analysis of the CT scan data is confirmed by the biopsies harvested from the quadriceps muscle at the beginning and the end of the study. The tissue composition of the samples, indicates an increase of non-contractile intra muscular tissue such as loose connective tissue, fat and collagen compared to normal (Figure 6.3). Electrical stimulation slightly improves the muscle

fibre content but it is still far below normal muscle with approximately 90% muscle fibre content.

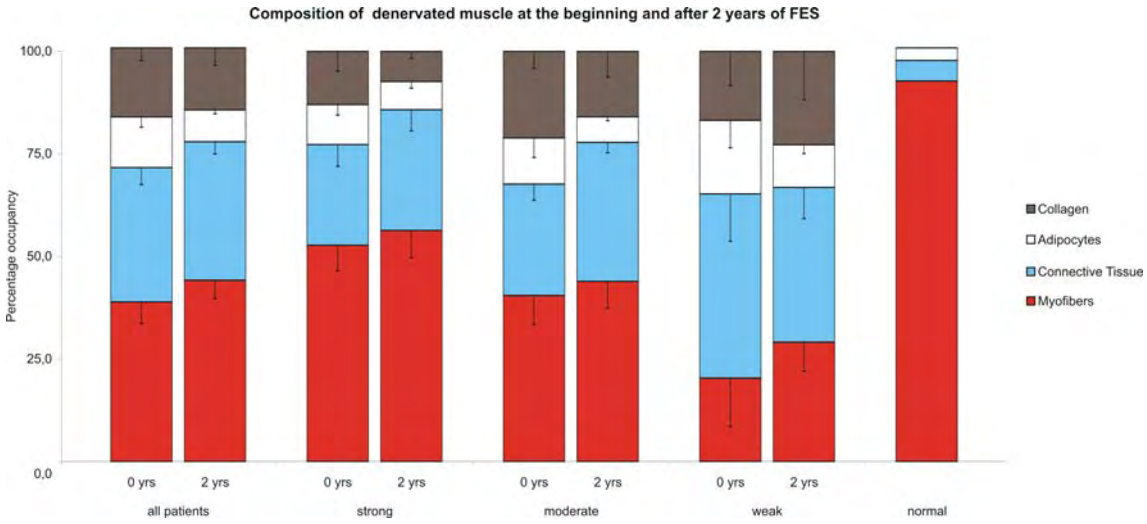


Figure 6.3: Column chart displaying the tissue composition in the muscle biopsies of the RISE patients before and after FES therapy. The rightmost column shows the tissue composition in a muscle biopsy from a normal healthy subject.

The reduced muscle fibre content together with the co-activation of the hamstrings muscles may explain the inconsistency between the restoration of the muscle cross-sectional area and the maximal knee extension torque measured.

Considering the results of the CT density measurement, made at the beginning of the study prior to any FES, it is evident that the muscle density is much lower and the de4nsity distribution appears different in the weak group compared to the other two groups (Figure 5.11 and 5.13). The patients of the moderate and strong groups are not distinguishable by the density measurement before FES therapy. Two years of treatment did not cause any significant changes regarding the density in the patients of the weak group, whereas in the moderate and strong groups a significant increase of muscle density was measured.

The finding suggests that determining the distribution of CT density within the muscle before treatment could be a useful tool to predict if a patient would benefit from FES therapy with the currently available technology and application protocol.

MFCV, MFRP. Initial aim was to setup a non-invasive method for assessing the electrophysiological parameters MFCV and MFRP but due to the limitations described in chapter 4.1 a minimal invasive method by inserting two needles into the investigated muscle, the SNEMG was the final outcome. The recorded parameters allow monitoring of changes in the electrophysiological properties of individual muscle fibres within the denervated muscle. In contrast to other studies (Cruz-Martinez 1989, Hoeven van der et al. 1993) placement of the recording needle was often not guided by a visible or palpable muscle twitches because in many cases the few activated fibres were lying under a thick layer of subcutaneous fat and the contractions elicited by needle stimulation were too weak to provide reliable contraction feedback. Movement of the stimulation needle was often useful to indicate the fibre orientation and helped to position the needle. The quadriceps muscle is a pennate muscle meaning that the exact orientation of the fibres is difficult to guess making the positioning of the needles even more difficult. This seems to be one the reasons why many studies investigating the MFCV were carried out in non-pennate muscles e.g. the biceps muscle were the fibres are oriented along the long axis of the muscle (Troni et al. 1983, Chino et al. 1984, Hoeven van der et al. 1993). Nevertheless with the developed technique the examination time for both thighs was reduced to about 45 minutes a time span which was well tolerated by the patients.

MFCV in the recorded in thirteen patients after the first and second year of FES therapy showed a significant increase for the fastest and mean CV of all measured fibres. No significant change was found in the minimum MFCV. By analysing the recorded data not for each patient individually, but pooling all fibres measured after one and two years of FES, the results are confirmed, because even after two years of training very slow conducting fibres (< 1.2 m/s) were detected (Figure 5.15).

Apart from electrophysiological parameters like the resting membrane potential (Gruener et al. 1979), the main factor influencing the CV is the fibre diameter (Håkansson 1956), suggesting that the results of the MFCV measurement and the muscle fibre diameter analysed by the biopsies should be related. Comparing the distribution of the fibre diameter for the patients grouped according to their functional outcome and the distribution of the MFCV after two years of training, strengthens this assumption (Figure 5.16 and 6.4).

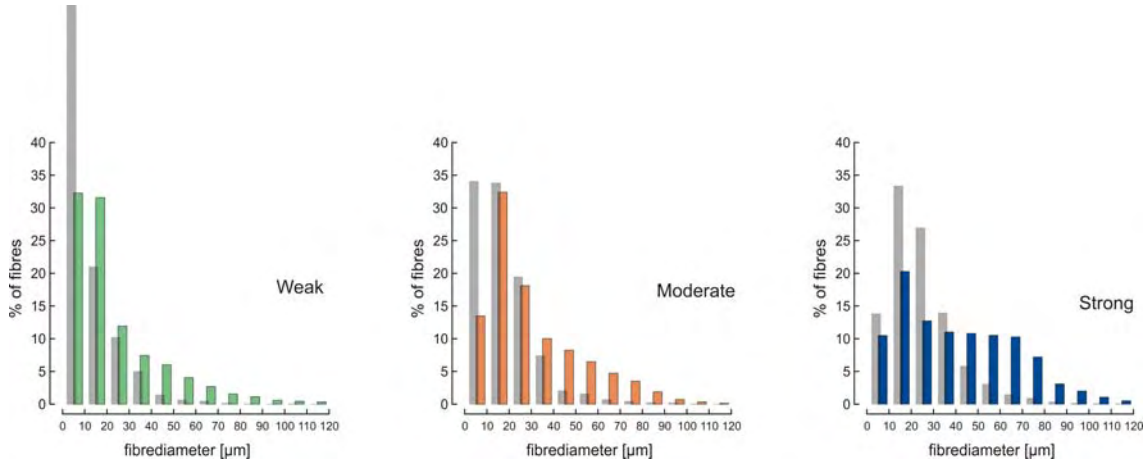


Figure 6.4: Histograms displaying the distribution of fibre diameter in the quadriceps muscle after 2 years of electrical stimulation training. Grouping according to the knee extension torque developed at the end of the study. Left shifted grey columns in the background are the initial values at inclusion into the study (Biopsies harvested in the course of the RISE study, prepared by U. Carraro).

The distribution of the fibre diameter in the weak and strong group indicates that most of the fibres are still very small in diameter and the shape of the distribution is not Gaussian like. In the moderate group slightly more medium sized fibres are found and both groups show some very big fibres which is also reflected in the MFCV distribution, where some very fast conducting fibres in these groups were detected. A reduced amount of small fibres with an increased number of medium sized fibres in the strong group indicates a distribution that is more similar to normal innervated muscle, but still the shape is not Gaussian. Contrary to the findings derived from the biopsy the MFCV in this group shows only very few slow conducting fibres and an almost Gaussian distribution. An explanation could be found in the fact that the fibres with a larger diameter exhibit a lower threshold for activation by ES. In the case of a muscle with a mixed sized fibre population this means that fibres with a larger diameter are activated more likely and therefore the results of the MFCV measurement will be biased to fibres with a larger diameter and consequently higher conduction velocity. Nevertheless even in this group of patients with the largest improvements and rather good muscle function MFCV was not fully restored (Figure 5.16).

MFRP measured by the shortest ISI with response showed no significant changes between first and second year of FES training, this corresponds to

the single twitch measurements where also no changes were detected after the first year of training (see above or chapter 5.3.1). In three patients, denervated for 0.8, 1.2 and 3.2 years, measurements at the time of inclusion into the study and after two years of FES showed a reduction in the shortest ISI in all patients with the most pronounced change in longest denervated one (Table 5.2). The results suggest that most of the improvements regarding the muscle fibre membrane properties take place during the first year of FES therapy, which is in good correlation to the single twitch measurements giving information on the excitability of the muscle fibres (see above).

Conclusions. The developed stimulation device and associated equipment is appropriate for treating denervated, degenerated muscles. All stimulation parameters required by the currently used training protocols can be realised with the equipment. A further increase of the output intensity is problematic, because it is very difficult to apply the stimulation current safely without damaging the skin. Next generation of stimulation equipment should provide more than two stimulation channels to reduce overall daily therapy time by simultaneous training of different muscle groups e.g. quadriceps, gluteus and calf muscles. This would also allow a functional training with a better more physiologic stand up and gait pattern.

The data acquired by the new assessment methods MFCV, MFRP and morphometric distribution of tissue density within the muscle cross-section show good correlation to functional tests like force or torque measurement and biopsy investigations. This indicates that the methods could serve as powerful new clinical tools to quantify the actual status of a denervated muscle and in some respects replace more invasive examination procedures like muscle biopsies. That would be of great advantage for supervising FES training for restoration of long-term denervated degenerated muscles, as the methods are simple enough to be applied in routine check-ups. Measurement of the tissue density distribution within the muscle even allows predicting to a certain extend if the patient would benefit from FES therapy or not.

The work shows that artificially induced contraction activity is not capable of fully restoring the denervated muscle. There are, however, marked improvements regarding the muscle cross-sectional area and density, the

electrophysiological properties MFCV and MFRP, as well as a recovery in the ability of the muscle to generate force. Secondary benefits induced by the above described changes are better blood perfusion of muscle and skin, better muscle cushioning with improved pressure distribution along the skin surface and therefore a decreased propensity to develop pressure sores. From psychological point of view the improved cosmetic appearance of the lower limbs results in enhanced self-esteem of the patients. These results justify the design of the specialised equipment for training and assessment and the commitment by the patients.

Literature

AAEM Nomenclature Committee. AAEM glossary of terms in electrodiagnostic medicine. Muscle Nerve 2001; (suppl 10):S1–50.

Adams RD, Denny-Brown D, Pearson CM. Diseases of muscle; a study in pathology. P. B. Hoeber, New York 1954.

Adrian RH, Peachey LD. Reconstruction of the action potential of frog sartorius muscle. J Physiol 1973; 235(1):103-31.

Al-Amood WS, Lewis DM. A comparison of the effects of denervation on the mechanical properties of rat and guinea-pig skeletal muscle. J Physiol 1989; 414:1-16.

Albuquerque EX, Thesleff S. A comparative study of membrane properties of innervated and chronically denervated fast and slow skeletal muscles of the rat. Acta Physiol Scand 1968; 73(4):471-80.

Ashley Z, Sutherland H, Lanmuller H, Unger E, Li F, Mayr W, Kern H, Jarvis JC, Salmons S. Determination of the chronaxie and rheobase of denervated limb muscles in conscious rabbits. Artif Organs 2005; 29(3):212-5.

Ashley Z, Salmons S, Boncompagni S, Protasi F, Russold M, Lanmuller H, Mayr W, Sutherland H, Jarvis JC. Effects of chronic electrical stimulation on long-term denervated muscles of the rabbit hind limb. J Muscle Res Cell Motil 2007; 28(4-5):203-17.

Ashley Z, Sutherland H, Russold M, Lanmuller H, Mayr W, Jarvis JC, Salmons S. Therapeutic stimulation of denervated muscles: The influence of pattern. Muscle Nerve 2008; in press.

Awad SS, Lightowlers RN, Young C, Chrzanowska-Lightowlers ZM, Lomo T, Slater CR. Sodium channel mRNAs at the neuromuscular junction: distinct patterns of accumulation and effects of muscle activity. J Neurosci 2001; 21(21):8456-63.

Banfai P. A surgical approach for the cochlear implant (author's transl). HNO 1978; 26(3):85-9.

Benton LA, Baker LL, Bowman BR, Waters RL. Functional electrical stimulation — A Practical Clinical Guide. Rancho Rehabilitation Engineering. Program, Rancho Los Amigos Medical Center, 7601 East Imperial Highway, Downey, California, 2nd edition, 1981.

Behrens MI, Vergara C. Increase of apamin receptors in skeletal muscle induced by colchicine: possible role in myotonia. Am J Physiol 1992; 263(4 Pt 1):C794-802.

Bijak M, Rakos M, Hofer C, Mayr W, Strohhofer M, Raschka D, Kern H. Stimulation parameter optimization for FES supported standing up and walking in SCI patients. Artif Organs 2005; 29(3):220-3.

Boncompagni S, Kern H, Rossini K, Hofer C, Mayr W, Carraro U, Protasi F. Structural differentiation of skeletal muscle fibers in the absence of innervation in humans. Proc Natl Acad Sci U S A 2007; 104(49):19339-44.

Boonstra AM, Va Weerden TW, Eisma WH, Pahlplatz VBM, Oosterhuis HJGH. The effect of low-frequency electrical stimulation on denervation atrophy in man. Scand J Rehab Med 1987; 19(3):127-34.

Bowden REM, Gutmann E. Denervation and re-innervation of human voluntary muscle. Brain 1944; 67:273-313.

Brackmann DE. The cochlear implant; basic principles. Laryngoscope 1976; 86(3):373-88.

Brindley GS. An implant to empty the bladder or close the urethra. J Neurol Neurosurg Psychiatry 1977; 40(4):358-69.

Brindley GS, Polkey CE, Rushton DN, Cardozo L. Sacral anterior root stimulators for bladder control in paraplegia: the first 50 cases. J Neurol Neurosurg Psychiatry 1986; 49(10):1104-14.

Buchthal F. Fibrillations: clinical electrophysiology. In Culp WJ, Ochoa J (eds): Abnormal nerves and muscle generators. Oxford University Press, New York 1982;632-62.

Buchthal F, Guld C, Rosenfalck P. Propagation velocity in electrically activated muscle fibers in man. Acta Physiol Scand 1955; 34:75-89.

Buchthal F, Rosenfalck P. Spontaneous electrical activity of human muscle. Electroencephalogr Clin Neurophysiol 1966; 20(4):321-36.

Bulcke JA, Termote JL, Palmers Y, Crolla D. Computed tomography of the human skeletal muscular system. Neuroradiology 1979; 17:127-36.

Carraro U, Dalla Libera L, Catani C, Danieli-Betto D. Chronic denervation of rat diaphragm: selective maintenance of adult fast myosin heavy chains. Muscle Nerve 1982; 5(7):515-24.

Carraro U, Catani C, Saggin L, Zrunek M, Szabolcs M, Gruber H, Streinzer W, Mayr W, Thoma H. Isomyosin changes after functional electrostimulation of denervated sheep muscle. Muscle Nerve 1988; 11(10):1016-28.

Chino N, Noda Y, Oda N. Conduction study in human muscle fibers in situ- A useful technique for diagnosing myopathies. Electromyogr clin Neurophysiol 1984; 58:513-6.

Cruz-Martinez A. Conduction velocity in human muscle fibers in situ. Study in upper and lower limbs in healthy adults. Electromyogr clin Neurophysiol 1989; 29:363-8.

Cruz-Martinez A, Arpa J. Muscle fiber conduction velocity in situ (MFCV) in denervation, reinnervation and disuse atrophy. Acta Neurol Scand 1999; 100:337-40.

Darkewitsch L. Die pathologische Anatomie der Muskeln. Handb. der path. Anat. des Nervensystems, Vol. 2, Berlin 1905.

Dulhunty AF. Excitation-contraction coupling and contractile properties in denervated rat EDL and soleus muscles. J Muscle Res Cell Motil 1985; 6(2):207-25.

Dulhunty AF. Excitation-contraction coupling from the 1950s into the new millennium. Clin Exp Pharmacol Physiol 2006; 33(9):763-72.

Duus P. Neurologisch-topische Diagnostik: Anatomie, Physiologie, Klinik. Georg Thieme Verlag Suttgart, New York; 7.Auflage, 2001.

Eberstein A., Eberstein B. Electrical stimulation of denervated muscle: is it worthwhile? Med Sci Sports Exerc 1996; 28(12):1463-9.

Eichhorn K, Schubert W, David E. Maintenance, training and functional use of denervated muscles. J Biomed Eng 1984; 6:205-11.

Edel H. Fibel der Elektrodiagnostik und Elektrotherapie. 6., bearb. Aufl. Verlag Gesundheit GmbH, Berlin 1991.

Frankenhaeuser B, Huxley AF. The action potential in the myelinated nerve fiber of xenopus laevis as computed on the basis of voltage clamp data. J Physiol 1964; 171:302-15.

Furman S, Schwedel JB. Cardiac pacing with endocavitary electrodes. N Engl J Med 1959; 261:943-8.

Gallasch E, Rafolt D, Kinz G, Fend M, Kern H, Mayr W. Evaluation of FES-induced knee joint moments in paraplegics with denervated muscles. Artif Organs 2005; 29(3):207-11.

Goodpaster BH, Kelley DE, Thaete FL, He J, Ross R. Skeletal muscle attenuation determined by computed tomography is associated with skeletal muscle lipid content. J Appl Physiol. 2000; 89(1):104-10.

Greenberg RJ. Analysis of Electrical Stimulation of the Vertebrate Retina—Work Towards a Retinal Prosthesis. The Johns Hopkins University Baltimore, MD. PhD thesis 1998.

Gruener R, Stern LZ, Weisz RR. Conduction velocities in single fibers of diseased human muscle. Neurology 1979; 29:1293-7.

Gundersen K. Early effects of denervation on isometric and isotonic contractile properties of rat skeletal muscles. Acta Physiol Scand 1985; 124(4):549-55.

Gundersen K. Determination of muscle contractile properties: the importance of the nerve. Acta Physiol Scand 1998; 162:333-41.

Gutmann E, Zelená J. Morphological changes in the denervated muscles. In Gutmann E (ed): The denervated muscle. Publishing House of the Czechoslovak Academy of Sciences, Prague 1962;57-102.

Håkansson CH. Conduction velocity and amplitude of the action potential as related to circumference in the isolated fibre of frog muscle. Acta Physiol Scand 1956; 37:14-34.

Henneberg KA, Roberge FA. Simulation of propagation along an isolated skeletal muscle fiber in an isotropic volume conductor. Ann Biomed Eng 1997; 25(1):5-28.

Hník P, Škorpil V, Vyklický L. Diagnosis and therapy of denervation atrophy. In Gutmann E (ed): The denervated muscle. Publishing House of the Czechoslovak Academy of Sciences, Prague 1962; 433-66.

Hochmair ES, Hochmair-Desoyer IJ, Burian K. Experience with implanted auditory nerve stimulator. Trans Am Soc Artif Intern Organs 1979; 25:357-61.

Hodgkin AL, Huxley AF. A quantitative description of membrane current and its application to conduction and excitation in nerve. J Physiol 1952; 117(4):500-44.

Hoeven van der JH, Zwarts MJ, van Weerden TW. Muscle fiber conduction velocity in amyothrophic lateral sclerosis and traumatic lesions of the plexus brachialis. Electroenceph clin Neurophysiol 1993; 89:304-10.

Hofer C, Kern H, Mayr W, Stöhr H, Abou-Zahra S. Funktionelle Elektrostimulation denervierter Muskulatur. Österr Z Phys Med 1997; (7) Supplement 2.

Hofer C, Mayr W, Stöhr H, Unger E, Kern H. A stimulator for functional activation of denervated muscles. Artif Organs 2002; 26(3):276-9.

Hofer C, Forstner C, Mödlin M, Jäger H, Mayr W, Kern H. In vivo assessment of conduction velocity and refractory period of denervated muscle fibers. Artif Organs 2005; 29(6):436-9.

Holle J, Frey M, Gruber H, Kern H, Stöhr H, Thoma H. Functional electrostimulation of paraplegics: experimental investigations and first clinical experience with an implantable stimulation device. Orthopedics 1984; 17:1145-56.

Huang S, Wang F, Hong G, Wan S, Kang H. Protective effects of ciliary neurotrophic factor on denervated skeletal muscle. J Huazhong Univ Sci Technolog Med Sci 2002; 22(2):148-51.

Hyatt JP, Roy RR, Baldwin KM, Wernig A, Edgerton VR. Activity-unrelated neural control of myogenic factors in a slow muscle. Muscle Nerve 2006; 33(1):49-60.

Ingersoll EH, Jones LL, Hegre ES. Effect on urinary bladder of unilateral stimulation of pelvic nerves in the dog. Am J Physiol. 1957; 189(1):167-72.

Judson JP, Glenn WW. Radio-frequency electrophrenic respiration. Long-term application to a patient with primary hypoventilation. JAMA 1968; 203(12):1033-7.

Kern H. Funktionelle Elektrostimulation paraplegischer Patienten. Österr Z Phys Med 1995; (5) Heft 1 Supplementum:1–79.

Kern H, Frey M, Holle J, Mayr W, Schwanda G, Stöhr H, Thoma H. Functional electrostimulation of paraplegic patients--1 year's practical application. Results in patients and experiences. Z Orthop Ihre Grenzgeb 1985; 123(1):1-12.

Kern H, Hofer C, Strohhofer M, Mayr W, Richter W, Stöhr H. Standing up with denervated muscles in humans using functional electrical stimulation. Artif Organs 1999; 23(5):447-52.

Kern H, Hofer C, Mödlin M, Forstner C, Raschka-Högler D, Mayr W, Stöhr H. Denervated muscles in humans: limitations and problems of currently used functional electrical stimulation training protocols. Artif Organs 2002; 26(3):216-8.

Keith MW, Peckham PH, Thrope GB, Buckett JR, Stroh KC, Menger V. Functional neuromuscular stimulation neuroprostheses for the tetraplegic hand. Clin Orthop Rel Res 1988; 233:25-33.

Kopits I. Beiträge zur Muskelpathologie. Histologische Befunde an Muskeln, Nerven und Blutgefäßen in Spät- und Endstadien peripherer Lähmungen, mit besonderer Berücksichtigung der Poliomyelitis anterior acuta. Arch Orthop Trauma Surg 1929; 27:277-403.

Kotsias BA, Muchnik S. Mechanical and electrical properties of denervated rat skeletal muscles. Exp Neurol 1987; 97(3):516-28.

Liberson WT. Functional electrotherapy. Trans Am Soc Artif Intern Organs 1962; 8:373-7.

Lømo T, Westgaard RH. Further studies on the control of ACh sensitivity by muscle activity in the rat. J Physiol 1975; 252(3):603-26.

Marsolais EB, Kobetic R. Functional walking in paralyzed patients by means of electrical stimulation. Clin Orthop 1983; 175:30-6.

Martinek J, Reichel M, Rattay F, Mayr W. Analysis of calculated electrical activation of denervated muscle fibers in the human thigh. Artif Organs 2005; 29(6):444-7.

Maughan RJ, Watson JS, Weir J. Strength and cross-sectional area of human skeletal muscle. J Physiol 1983; 338:37-49.

Mayr W. Oberflächenelektrode. Austrian Patent: AT 503 420 B1; 2007: A842.

Mayr W, Bijak M, Girsch W, Holle J, Lanmüller H, Thoma H, Zrunek M. Multichannel stimulation of phrenic nerves by epineural electrodes. Clinical experience and future developments. ASAIO J 1993; 39(3):729-35.

Mayr W, Bijak M, Rafolt D, Sauermann S, Unger E, Lanmüller H. Basic design and construction of the Vienna FES implants: existing solutions and prospects for new generations of implants. Med Eng Phys 2001; 23(1):53-60.

Mayr W, Hofer C, Bijak M, Rafolt D, Unger E, Sauermann S, Lanmueller H, Kern H. Functional electrical stimulation (FES) of long-term denervated muscles: existing and prospective technological solutions. BAM 2002; 12(6):287-90.

Mayr W, Hofer C, Rafolt D, Bijak M, Lanmüller H, Reichel M, Sauermann S, Unger E. The EU-Project RISE: General overview and engineering aspects. Proceedings of the 8th Vienna Inernational Workshop on Functional Electrical Stimulation 2004; ISBN 3-900928-07-9:24-6.

McCormick DA. Membrane Potential and Action Potential. In Zigmond MJ et al. (eds): Fundamental Neuroscience. Academic Press, 1999;129-54.

McGinnis PM. Biomechanics of Sport and Exercise. 1999 Champaign, IL: Human Kinetics.

McNeal DR. 2000 Years of electrical stimulation. In Hambrecht and Reswick (eds): Functional Electrical Stimulation. Marcel Dekker Inc., New York, 1977;3-35.

Midrio M, Danieli-Betto D, Megighian A, Betto R. Early effects of denervation on sarcoplasmic reticulum properties of slow-twitch rat muscle fibres. Pflugers Arch 1997; 434(4):398-405.

Mihelin M, Trontelj JV, Stålberg E. Muscle fiber recovery functions studied with double pulse stimulation. Muscle Nerve 1991; 14:739-47.

Mödlin M, Forstner C, Hofer C, Mayr W, Richter W, Carraro U, Protasi F, Kern H. Electrical stimulation of denervated muscles: first results of a clinical study. Artif Organs 2005;29(3):203-6.

Møller M, Arnsbo P, Asklund M, Christensen PD, Gadsbøll N, Svendsen JH, Klarholt E, Kleist KE, Mortensen PT, Pietersen A, Simonsen EH, Thomsen PE, Vesterlund T, Wiggers R. Quality assessment of pacemaker implantations in Denmark. Europace 2002; 4(2):107-12.

Mokrusch T, Engelhardt A, Eichhorn KF, Prischenk G, Prischenk H, Sack G, Neundörfer B. Effects of long-impulse electrical stimulation on atrophy and fibre type composition of chronically denervated fast rabbit muscle. J Neurol 1990; 237:29-34.

Mokrusch T, Neundorfer B. Electrotherapy of permanently denervated muscle - long term experiment. Eur J Phys Med & Reha 1994; 4(5):166-73.

Narici MV, Hoppeler H, Kayser B, Landoni L, Claassen H, Gavardi C, Conti M, Cerretelli P. Human quadriceps cross-sectional area, torque and neural activation during 6 months strength training. Acta Physiol Scand 1996; 157(2):175-86.

Netter FH. The Ciba Collection of Medical Illustrations. Nervous System, Part I: Anatomy and Physiology. Ciba Pharmaceutical Company, Ciba-Geigy Corporation, USA, 1983.

Oppenheim H. Handbuch; Heilkunde der Nerven. Berlin 1894.

Pappone PA. Voltage-clamp experiments in normal and denervated mammalian skeletal muscle fibres. J Physiol 1980; 306:377-410.

Peckham PH, Poon CW, Ko WH, Marsolais EB, Rosen JI. Multi-channel implantable stimulator for control of paralyzed muscle. IEEE Trans Biomed Eng 1981; 28:530-6.

Pette D, Vrbová G. Adaptation of mammalian skeletal muscle fibers to chronic electrical stimulation. Rev Physiol Biochem Pharmacol 1992; 120:115-202.

Rafolt D. Vorrichtung zur Anwendung und Aufbewahrung von Hautelektroden bei Elektrostimulation und Messung bioelektrischer Signale am menschlichen Körper. Austrian Patent: AT 404797 B; 1999.

Ramirez BU. Axonal transport blockade and denervation have qualitatively different effects upon skeletal muscle metabolism. J Neurobiol 1984; 15(2):119-26.

Ramirez BU, Behrens MI, Vergara C. Neural control of the expression of a Ca(2+)-activated K+ channel involved in the induction of myotonic-like characteristics. Cell Mol Neurobiol 1996; 16(1):39-49.

Ramirez BU, Retamal L, Vergara C. Ciliary neurotrophic factor (CNTF) affects the excitable and contractile properties of innervated skeletal muscles. Biol Res 2003; 36(3-4):303-12.

Rattay F. Analysis of models for external stimulation of axons. IEEE Trans Biomed Eng 1986; 33(10):974-7.

Rattay F. Analysis of models for extracellular fiber stimulation. IEEE Trans Biomed Eng 1989; 36(7):676-82.

Rattay F. Current distance relations for fiber stimulation with pointsources. IEEE Trans Biomed Eng 2008; 55(3):1122-7.

Reichel M. Funktionelle Elektrostimulation denervierter Skelettmuskulatur – Modellbildung und Simulation (in German). PhD thesis, Vienna Univ Technol, Vienna, Austria, 1999.

Robert ED, Oester YT. Electrodiagnosis of nerve-impulse deprived skeletal muscle. J Appl Physiol 1970; 28(4):439-43.

Roy RR, Zhong H, Monti RJ, Vallance KA, Edgerton VR. Mechanical properties of the electrically silent adult rat soleus muscle. Muscle Nerve 2002; 26(3):404-12.

Salmons S, Vrbová G. The influence of activity on some contractile characteristics of mammalian fast and slow muscles. J Physiol 1969; 201(3):535-49.

Salmons S, Ashley Z, Sutherland H, Russold MF, Li F, Jarvis JC. Functional electrical stimulation of denervated muscles: basic issues. Artif Organs 2005; 29(3):199-202.

Silbernagl S, Despopoulos A. Taschenatlas der Physiologie. 4.,überarb. Aufl. Thieme, Stuttgart; New York, 1991.

Sollie G, Hermens HJ, Boon KL, Wallinga-de Jonge W, Zilvold G. The measurement of the conduction velocity of muscle fibers with surface EMG according to the cross-correlation method. Electromyogr Clin Neurophysiol 1985; 25:193-204.

Stickler Y, Martinek J, Hofer C, Rattay F. A finite element model of the electrically stimulated human thigh: Changes due to denervation and training. Artif Organs 2008; in press.

Talon S, Giroux-Metges MA, Pennec JP, Guillet C, Gascan H, Gioux M. Rapid protein kinase C-dependent reduction of rat skeletal muscle voltage-gated sodium channels by ciliary neurotrophic factor. J Physiol 2005 Jun 15; 565(Pt 3):827-41.

Talonen PP, Baer GA, Häkkinen V, Ojala JK. Neurophysiological and technical considerations for the design of an implantable phrenic nerve stimulator. Med Biol Eng Comput 1990; 28(1):31-7.

Taylor PN, Ewins DJ, Fox B, Grundy D, Swain ID. Limb blood flow, cardiac output and quadriceps muscle bulk following spinal cord injury and the effect of training for the Odstock functional electrical stimulation standing system. Paraplegia. 1993; 31(5):303-10.

Termote JL, Baert A, Crolla D, Palmers Y, Bulcke JA. Computed tomography of the normal and pathologic muscular system. Radiology. 1980; 137(2):439-44.

Tower S. Atrophy and degeneration in skeletal muscle. Am J Anat 1935; 56:1-43.

Troni W, Cantello R, Rainero I. Conduction velocity along human muscle fibers in situ. Neurology 1983; 33:1453-9.

Trontelj JV, Stålberg E. Responses to electrical stimulation of denervated human muscle fibres recorded with single fibre EMG. J Neurol Neurosurg Psychiatry 1983; 46:305-9.

Uhlig CE, Taneri S, Benner FP, Gerding H. Electrical stimulation of the visual System. From empirical approaches to the development of visual implants. Ophthalmologe 2001; 98:1089-96.

Valencic V, Vodovnik L, Stefancic M, Jelnikar T. Improved motor response due to chronic electrical stimulation of denervated tibialis anterior muscle in humans. Muscle Nerve 1986; 9:612-7.

Vergara C, Ramirez B, Behrens MI. Colchicine alters apamin receptors, electrical activity, and skeletal muscle relaxation. Muscle Nerve 1993 Sep; 16(9):935-40.

Wheater PR, Burkitt HG, Daniels VG. Functional Histology. Churchill Livingstone, Edinburgh, 1979.

Woodcock AH, Taylor PN, Ewins DJ. Long pulse biphasic electrical stimulation of denervated muscle. Artif Organs 1999; 23(5):457-9.

Wyndaele M, Wyndaele JJ. Incidence, prevalence and epidemiology of spinal cord injury: what learns a worldwide literature survey? Spinal Cord 2006; 44(9):523-9.

Zoll PM. Resuscitation of the heart in ventricular standstill by edeanal electric stimulation. N Engl J Med 1952; 247(20):768-71.

Zrunek M, Bigenzahn W, Mayr W, Unger E, Feldner-Busztin H. A laryngeal pacemaker for inspiration-controlled, direct electrical stimulation of the denervated posterior cricoarytenoid muscle in sheep. Eur Arch Otorhinolaryngol 1991; 248(8):445-8.

Appendix

A Stimulation device programme listing

P:\DenStim\Src\DenStim.c
Printed at 16:24 on 23 Apr 2008 **Page 1 of 30**

```
/**********************************************************************
 *
 *
 * PROJECT:  PalmOS
 * FILE:     DenStim.c
 * AUTHOR:   DI Christian Hofer: January 12, 2006       13:37
 *
 * DECLARER: DenStim
 *
 * DESCRIPTION: Version for both stimulator types with RS232 connection,
 *                              IrComm and improved transmission protocol
 *                              Future work to do: e.g. database,
 *                              Impulse Pause of 400, 1000, 2000ms added
 *                              Amplitude settings 90 and 99 (max) added
 *                              Verifying of Stim-Programs for old devices added
 *                              Serial port activation modified for Palm m500 EZ processor
 *                              Serial port timing modified for Palm m500
 *
 *                              Programm modified to work with Palm OS 5.0 PACE not ready
 *                                      because initialization of serial port modified in SDK
 *                              Initialization routines hidden bugs removed
 *                              Programm de-init changed, no remaining memory chunks left
 *                                      anymore
 *
 *                              MiF Stimulation added
 *                              Raw IrDA Protocol works now with Palm OS 5.0 on TT3
 *                              RS232 8E1 protocol for evaluation purposes added
 *
 *                              Serial port activation with different baud rate 1200 or 9600
 *
 *                              After firmware revision by H.Stöhr timing of programm optimized
 *
 **********************************************************************/
#include <PalmOS.h>
#include "DenStimRsc.h"

/**********************************************************************
 *
 *   Entry Points
 *
 **********************************************************************/


/**********************************************************************
 *
 *   Internal Structures
 *
 **********************************************************************/
typedef struct
        {
        Char replaceme;
        } DenStimPreferenceType;

typedef struct
        {
        Char replaceme;
        } DenStimAppInfoType;

typedef DenStimAppInfoType* DenStimAppInfoPtr;


/**********************************************************************
 *
 *   Global variables
 *
 **********************************************************************/
//static Boolean HideSecretRecords;

Boolean g142Version = false, gIrComm = true, gMiFVersion = false, gNewVersion = false, gPalmOS5 = false, gRS2328E1 = false;
UInt16  hor_ver_step_dauer100[30][3] = {{1,19,13},{1,17,15},{1,14,18},{1,12,21},{1,10,26},{1,8,32},
                                                                            {1,7,37},{1,6,43},{1,5,51},{1,4,64},{1,3,85},{2,5,102},
                                                                            {1,2,128},{3,5,153},{2,3,171},{3,4,192},{1,1,256},{7,6,
299},
                                                                            {4,3,341},{3,2,384},{2,1,512},{5,2,640},{3,1,768},{7,2,
896},
                                                                            {4,1,1024},{9,2,1152},{5,1,1280},{6,1,1536},{8,1,2048},
                                                                            {12,1,3072}},
                hor_ver_step_dauer142[30][3] = {{1,27,13},{1,24,15},{1,20,18},{1,17,21},{1,13,26},{1,11,33},
                                                                                {1,9,40},{1,8,45},{1,7,51},{1,5,72},{1,4,90},{2,7,1
03},
                                                                                {3,8,136},{3,7,155},{1,2,181},{3,5,218},{3,4,272},{
5,6,302},
                                                                                {1,1,363},{7,6,424},{7,5,508},{7,4,636},{2,1,727},{
7,3,848},
                                                                                {3,1,1090},{10,3,1211},{4,1,1454},{9,2,1635},{6,1,2
181},
                                                                                {9,1,3271}};

UInt16  gSerialRefNum = NULL, gBaudRate = 9600;

MemHandle       stringArrayPtrH = NULL;
MemHandle       listContentsH = NULL;


/**********************************************************************
 *
 *   Internal Constants
 *
 **********************************************************************/
#define appFileCreator                     'CDSt'
```

P:\DenStim\Src\DenStim.c
Printed at 16:24 on 23 Apr 2008 **Page 2 of 30**

```
#define appVersionNum              0x01
#define appPrefID                  0x00
#define appPrefVersionNum          0x01


// Define the minimum OS version we support
#define ourMinVersion   sysMakeROMVersion(3,3,0,sysROMStageRelease,0)

// Define serial communication settings used
#define Setting8N2      srmSettingsFlagBitsPerChar8 | srmSettingsFlagStopBits2
#define Setting8E1      srmSettingsFlagBitsPerChar8 | srmSettingsFlagStopBits1 | srmSettingsFlagParityOnM | srmSettingsFlagParityEv
enM

// Define maximum string length for reading protocol
#define maxlength       1024
//#define maxlength     2048

/**********************************************************************
 *
 *   Internal Functions
 *
 **********************************************************************/
static Err RomVersionCompatible(UInt32 requiredVersion, UInt16 launchFlags);
static FieldPtr SetFieldTextFromHandle(UInt16 fieldID, MemHandle newTextHandle);
static FieldPtr SetFieldTextFromString(UInt16 fieldID, Char* stringP);
static void CopyStringFromFieldText(UInt16 fieldID, Char* stringP);
static void ClearFieldText(UInt16 fieldID);
static void UpdateScrollBar(UInt16 fieldID, UInt16 scrollBarID);
static void LineScroll(Int16 numLines, UInt16 fieldID, UInt16 scrollBarID, Boolean redraw);
static void PageScroll(WinDirectionType direction, UInt16 fieldID, UInt16 scrollBarID);
static void SetMainStimType(void);
static Boolean ActivateSerialPort(UInt32 serialSettings);
static Boolean DeactivateSerialPort(void);
static UInt32 SendSerialData(Char* serStringPtr);
static UInt32 ReadSerialData(Char* serStringPtr, UInt32 serStringSize);
static Boolean CheckROMVersion(void);
static void TestSerialCommunication(void);
static void PulseShapeDraw(FormPtr frmP);
static void SetProgramm(FormPtr frmP);
static void VerifyProgramm(FormPtr frmP);
static void ReadProgramm(FormPtr frmP);
static void * GetObjectPtr(UInt16 objectID);


/**********************************************************************
 *
 * FUNCTION:    SetFieldTextFromHandle
 *
 * DESCRIPTION: Set the handle of the text field specified by the fieldID
 *                              to the new text to display and releases the old handle
 *
 * PARAMETERS:  ID of text field, Handle of new text
 *
 * RETURNED:    Pointer to text field
 *
 * REVISION HISTORY: V 1.0      04/05/2000
 *
 *
 **********************************************************************/
static FieldPtr SetFieldTextFromHandle(UInt16 fieldID, MemHandle newTextHandle)
{
        MemHandle       oldTextHandle;
        FormPtr         frm = FrmGetActiveForm();
        FieldPtr        fieldP;

        fieldP = FrmGetObjectPtr(frm, FrmGetObjectIndex(frm, fieldID));
        ErrNonFatalDisplayIf(!fieldP, "Missing Field!!!");

        oldTextHandle = FldGetTextHandle(fieldP);

        FldSetTextHandle(fieldP, newTextHandle);
        FldDrawField(fieldP);

        if (oldTextHandle)
                MemHandleFree(oldTextHandle);

        return fieldP;
}

/**********************************************************************
 *
 * FUNCTION:    SetFieldTextFromString
 *
 * DESCRIPTION: Set the field´s text to the string specified by the pointer
 *                              to it, allocates a handle and copies the string to it
 *
 * PARAMETERS:  ID of text field, Pointer to the string
 *
 * RETURNED:    Pointer to text field
 *
 * REVISION HISTORY: V 1.0      04/05/2000
 *
 *
 **********************************************************************/
static FieldPtr SetFieldTextFromString(UInt16 fieldID, Char* stringP)
{
        MemHandle               textHandle;

        textHandle = MemHandleNew(StrLen(stringP) + 1);
        if (!textHandle)
```

```
194                 return NULL;
195
196         StrCopy(MemHandleLock(textHandle), stringP);
197         MemHandleUnlock(textHandle);
198
199         return SetFieldTextFromHandle(fieldID, textHandle);
200 }
201
202 /***********************************************************************
203  *
204  * FUNCTION:    CopyStringFromFieldText
205  *
206  * DESCRIPTION: Copies the field´s text to the string specified by the pointer
207  *                              to it
208  *
209  * PARAMETERS:  ID of text field, Pointer to the string
210  *
211  * RETURNED:    Nothing
212  *
213  * REVISION HISTORY: V 1.0      11/05/2000
214  *
215  *
216  ***********************************************************************/
217 static void CopyStringFromFieldText(UInt16 fieldID, Char* stringP)
218 {
219         Char*           textP;
220         FormPtr         frmP = FrmGetActiveForm();
221         FieldPtr        fieldP;
222         MemHandle       textHandle;
223
224         fieldP = FrmGetObjectPtr(frmP, FrmGetObjectIndex(frmP, fieldID));
225         textHandle = FldGetTextHandle(fieldP);
226         if (textHandle)
227                 {
228                 FldSetTextHandle(fieldP, NULL);
229                 textP = MemHandleLock(textHandle);
230                 StrCopy(stringP, textP);
231                 MemHandleUnlock(textHandle);
232                 FldSetTextHandle(fieldP, textHandle);
233                 FldDrawField(fieldP);
234                 }
235         else
236                 StrCopy(stringP, "");
237         return;
238 }
239 /***********************************************************************
240  *
241  * FUNCTION:    ClearFieldText
242  *
243  * DESCRIPTION: Clears the text from field specified by the ID
244  *
245  * PARAMETERS:  ID of field
246  *
247  * RETURNED:    nothing
248  *
249  * REVISION HISTORY: V 1.0      04/05/2000
250  *
251  *
252  ***********************************************************************/
253 static void ClearFieldText(UInt16 fieldID)
254 {
255         SetFieldTextFromHandle(fieldID, NULL);
256 }
257
258 /***********************************************************************
259  *
260  * FUNCTION:    UpdateScrollBar
261  *
262  * DESCRIPTION: Updates scrollbar of text field
263  *
264  * PARAMETERS:  ID of text field, ID of scroll bar
265  *
266  * RETURNED:    nothing
267  *
268  * REVISION HISTORY: V 1.0      15/05/2000
269  *
270  *
271  ***********************************************************************/
272 static void UpdateScrollBar(UInt16 fieldID, UInt16 scrollBarID)
273 {
274         FormPtr         frmP = FrmGetActiveForm();
275         FieldPtr        fieldP;
276         ScrollBarPtr scrollP;
277         UInt16          currentPosition, textHeight, fieldHeight, maxValue;
278
279         fieldP = FrmGetObjectPtr(frmP, FrmGetObjectIndex(frmP, fieldID));
280         FldGetScrollValues(fieldP, &currentPosition, &textHeight, &fieldHeight);
281
282         if (textHeight > fieldHeight)
283                 maxValue = textHeight - fieldHeight;
284         else
285                 if (currentPosition)
286                         maxValue = currentPosition;
287                 else
288                         maxValue = 0;
289         scrollP = FrmGetObjectPtr(frmP, FrmGetObjectIndex(frmP, scrollBarID));
290         SclSetScrollBar(scrollP, currentPosition, 0, maxValue, fieldHeight - 1);
291 }
292
293 /***********************************************************************
```



```
294  *
295  * FUNCTION:    LineScroll
296  *
297  * DESCRIPTION: Scrolls text field n lines
298  *
299  * PARAMETERS:  Number of lines to scroll, ID of text field, ID of scroll bar,
300  *                              Boolean for redrawing scroll bar
301  *
302  * RETURNED:    nothing
303  *
304  * REVISION HISTORY: V 1.0      15/05/2000
305  *
306  *
307  ***********************************************************************/
308 static void LineScroll(Int16 numLines, UInt16 fieldID, UInt16 scrollBarID, Boolean redraw)
309 {
310         FormPtr         frmP = FrmGetActiveForm();
311         FieldPtr        fieldP;
312
313         fieldP = FrmGetObjectPtr(frmP, FrmGetObjectIndex(frmP, fieldID));
314         if (numLines < 0)
315                 FldScrollField(fieldP, -numLines, winUp);
316         else
317                 FldScrollField(fieldP, numLines, winDown);
318         if ((FldGetNumberOfBlankLines(fieldP) && numLines < 0) || redraw)
319                 UpdateScrollBar(fieldID, scrollBarID);
320 }
321
322 /***********************************************************************
323  *
324  * FUNCTION:    PageScroll
325  *
326  * DESCRIPTION: Scrolls text field one page up or down
327  *
328  * PARAMETERS:  Direction to scroll, ID of text field, ID of scroll bar
329  *
330  * RETURNED:    nothing
331  *
332  * REVISION HISTORY: V 1.0      15/05/2000
333  *
334  *
335  ***********************************************************************/
336 static void PageScroll(WinDirectionType direction, UInt16 fieldID, UInt16 scrollBarID)
337 {
338         FormPtr         frmP = FrmGetActiveForm();
339         FieldPtr        fieldP;
340
341         fieldP = FrmGetObjectPtr(frmP, FrmGetObjectIndex(frmP, fieldID));
342         if (FldScrollable(fieldP, direction))
343         {
344                 Int16 linesToScroll = FldGetVisibleLines(fieldP) - 1;
345
346                 if (direction == winUp)
347                         linesToScroll = -linesToScroll;
348                 LineScroll(linesToScroll, fieldID, scrollBarID, true);
349         }
350 }
351
352 /***********************************************************************
353  *
354  * FUNCTION:    SetMainStimType
355  *
356  * DESCRIPTION: Set the contents of the impulse width list
357  *                              according to used the stimulation device
358  *
359  * PARAMETERS:
360  *
361  * RETURNED:
362  *
363  * REVISION HISTORY: V 1.2      16/09/2003 16:15
364  *
365  *
366  ***********************************************************************/
367 static void SetMainStimType(void)
368 {
369         FormPtr         frmP = FrmGetActiveForm();
370
371 // Set impulse duration list according to stimlator type
372         if (listContentsH)
373         {
374                 Char*   stringPtr;
375                 UInt16  itemNumber, selectedItem, selectedItemMiF, hilfsvar;
376                 Char    impD[10];
377
378                 selectedItem = LstGetSelection(GetObjectPtr(MainImpDList));
379                 selectedItemMiF = LstGetSelection(GetObjectPtr(MainMiFImpDList));
380
381                 // When the MainForm is initialized seltectedItem can be invalid
382                 if ((selectedItem < 0) || (selectedItem > 30))
383                         selectedItem = 0;
384                 if ((selectedItemMiF < 0) || (selectedItemMiF > 30))
385                         selectedItemMiF = 0;
386
387                 if(stringArrayPtrH)
388                 {
389                         MemHandleUnlock(stringArrayPtrH);
390                         MemHandleFree(stringArrayPtrH);
391                 }
392                 MemHandleUnlock(listContentsH);
393                 stringPtr = MemHandleLock(listContentsH);
```

```
394                 for (itemNumber = 0; itemNumber < 30; ++itemNumber)
395                 {
396                         if (g142Version && !gNewVersion)
397                                 StrIToA(impD, hor_ver_step_dauer142[itemNumber][2]);
398                         else
399                                 StrIToA(impD, hor_ver_step_dauer100[itemNumber][2]);
400                         hilfsvar = StrLen(impD);
401                         impD[hilfsvar] = impD[hilfsvar - 1];
402                         impD[hilfsvar - 1] = ',';
403                         impD[hilfsvar + 1] = '\0';
404                         StrCopy(stringPtr, impD);
405                         stringPtr += StrLen(stringPtr) + 1;
406                 }
407                 MemHandleUnlock(listContentsH);
408                 stringArrayPtrH = SysFormPointerArrayToStrings(MemHandleLock(listContentsH), 30);
409                 LstSetListChoices(GetObjectPtr(MainImpDList), MemHandleLock(stringArrayPtrH), 30);
410
411                 LstSetSelection(GetObjectPtr(MainImpDList), selectedItem);
412                 LstMakeItemVisible(GetObjectPtr(MainImpDList), selectedItem);
413                 stringPtr = LstGetSelectionText(GetObjectPtr(MainImpDList),
414                                                                                         LstGetSelection(GetObjectPtr(MainImpDList))
    );
415                 CtlSetLabel(GetObjectPtr(MainImpDPopTrigger), stringPtr);
416
417                 MemHandleUnlock(stringArrayPtrH);
418                 MemHandleUnlock(listContentsH);
419                 stringArrayPtrH = SysFormPointerArrayToStrings(MemHandleLock(listContentsH), 30);
420                 LstSetListChoices(GetObjectPtr(MainMiFImpDList), MemHandleLock(stringArrayPtrH), 30);
421
422                 LstSetSelection(GetObjectPtr(MainMiFImpDList), selectedItemMiF);
423                 LstMakeItemVisible(GetObjectPtr(MainMiFImpDList), selectedItemMiF);
424                 stringPtr = LstGetSelectionText(GetObjectPtr(MainMiFImpDList),
425                                                                                         LstGetSelection(GetObjectPtr(MainMiFImpDLis
    t)));
426                 CtlSetLabel(GetObjectPtr(MainMiFImpDPopTrigger), stringPtr);
427         }
428
429 // Set programm push buttons according to stimulator type
430         ClearFieldText(MainTextField);
431         if (!g142Version && !gNewVersion)
432         {
433                 if (!((CtlGetValue(GetObjectPtr(MainProg1PushButton))) ||
434                                         (!CtlGetValue(GetObjectPtr(MainProg2PushButton))) ||
435                                                 (!(CtlGetValue(GetObjectPtr(MainProg3PushButton)))))
436                         CtlSetValue(GetObjectPtr(MainProg3PushButton),1);
437                 CtlSetValue(GetObjectPtr(MainProg4PushButton),0);
438                 CtlSetValue(GetObjectPtr(MainProg5PushButton),0);
439                 CtlSetValue(GetObjectPtr(MainProg6PushButton),0);
440                 CtlSetValue(GetObjectPtr(MainProg7PushButton),0);
441                 CtlSetValue(GetObjectPtr(MainProg8PushButton),0);
442                 CtlSetValue(GetObjectPtr(MainProg9PushButton),0);
443                 FrmHideObject(frmP, FrmGetObjectIndex(frmP, MainProg4PushButton));
444                 FrmHideObject(frmP, FrmGetObjectIndex(frmP, MainProg5PushButton));
445                 FrmHideObject(frmP, FrmGetObjectIndex(frmP, MainProg6PushButton));
446                 FrmHideObject(frmP, FrmGetObjectIndex(frmP, MainProg7PushButton));
447                 FrmHideObject(frmP, FrmGetObjectIndex(frmP, MainProg8PushButton));
448                 FrmHideObject(frmP, FrmGetObjectIndex(frmP, MainProg9PushButton));
449                 SetFieldTextFromString(MainTextField, "    Netz");
450         }
451         else
452         {
453                 FrmShowObject(frmP, FrmGetObjectIndex(frmP, MainProg4PushButton));
454                 FrmShowObject(frmP, FrmGetObjectIndex(frmP, MainProg5PushButton));
455                 FrmShowObject(frmP, FrmGetObjectIndex(frmP, MainProg6PushButton));
456                 FrmShowObject(frmP, FrmGetObjectIndex(frmP, MainProg7PushButton));
457                 FrmShowObject(frmP, FrmGetObjectIndex(frmP, MainProg8PushButton));
458                 FrmShowObject(frmP, FrmGetObjectIndex(frmP, MainProg9PushButton));
459                 SetFieldTextFromString(MainTextField, "    Akku");
460         }
461
462 // Set MiF scrolldown list according to stimulator type
463         if (!gMiFVersion)
464         {
465                 CtlHideControl(GetObjectPtr(MainMiFAnzPopTrigger));
466                 CtlHideControl(GetObjectPtr(MainMiFImpDPopTrigger));
467                 FrmHideObject(frmP, FrmGetObjectIndex(frmP, MainMiFAnzLabel));
468                 FrmHideObject(frmP, FrmGetObjectIndex(frmP, MainMiFImpDLabel));
469                 FrmHideObject(frmP, FrmGetObjectIndex(frmP, MainUnit7Label));
470         }
471         else
472         {
473                 CtlShowControl(GetObjectPtr(MainMiFAnzPopTrigger));
474                 CtlShowControl(GetObjectPtr(MainMiFImpDPopTrigger));
475                 FrmShowObject(frmP, FrmGetObjectIndex(frmP, MainMiFAnzLabel));
476                 FrmShowObject(frmP, FrmGetObjectIndex(frmP, MainMiFImpDLabel));
477                 FrmShowObject(frmP, FrmGetObjectIndex(frmP, MainUnit7Label));
478         }
479 }
480 /***********************************************************************
481  *
482  * FUNCTION:    ActivateSerialPort
483  *
484  * DESCRIPTION: Opens the serial Port for Communication with Stimulator
485  *                              Settings: 1200 or 9600 baud,    8,N,2 for RS232 and IrDA
486  *
487  * PARAMETERS:  Pointer to serial port number
488  *
489  * RETURNED:    false if the port was opened successfully
490  *              true if an error occured
491  *
```



```
492  * REVISION HISTORY: V 1.4      05/12/2005
493  *
494  *                              Serial port activation modified for Palm m500
495  *                              Serial port activation modified for Palm TT3
496  *                              Serial port activation with different baud rates
497  *
498  ***********************************************************************/
499 static Boolean ActivateSerialPort(UInt32 serialSettings)
500 {
501
502         Err                     err;
503         UInt16          length;
504         UInt32          value, srmSettingsFlags;
505
506         err = FtrGet(sysFileCSerialMgr, sysFtrNewSerialPresent, &value);
507         ErrNonFatalDisplayIf(err != 0, "Can't find New Serial Manager");
508         // Get Verion of Palm OS
509         FtrGet(sysFtrCreator, sysFtrNumROMVersion, &value);
510         if (!(value < sysMakeROMVersion(5,0,0,sysROMStageRelease,0)))
511                 gPalmOS5 = true;
512         // If Palm OS 5.0 or higher set gPalmOS5 true
513         if (gIrComm)
514                 err = SrmOpen(serPortIrPort, gBaudRate, &gSerialRefNum);
515         else
516                 err = SrmOpen(serPortCradleRS232Port, gBaudRate, &gSerialRefNum);
517         if (err != 0)
518         {
519                 if ( err == serErrAlreadyOpen)
520                 {
521                         FrmAlert(SerialInUseAlert);
522                         SrmClose(gSerialRefNum);
523                 }
524                 else
525                         FrmAlert(CantOpenSerialAlert);
526                 return true;
527         }
528         srmSettingsFlags = serialSettings;
529         length = sizeof(srmSettingsFlags);
530         err = SrmControl(gSerialRefNum, srmCtlSetFlags, &srmSettingsFlags, &length);
531         if (gIrComm)
532         {
533                 srmSettingsFlags = 0;
534                 err = SrmControl(gSerialRefNum, srmCtlIrDAEnable, &srmSettingsFlags, &length);
535                 if (err != 0)
536                 {
537                         FrmAlert(DebuggingInfoAlert);
538                 }
539         }
540         SysTaskDelay(SysTicksPerSecond()/30);
541         if (gPalmOS5)
542                 SysTaskDelay(SysTicksPerSecond() / 10);
543         SrmReceiveFlush(gSerialRefNum, 1);
544         return false;
545 }
546
547 /***********************************************************************
548  *
549  * FUNCTION:    DeactivateSerialPort
550  *
551  * DESCRIPTION: Closes the serial Port for Communication with Stimulator
552  *
553  * PARAMETERS:  Pointer to serial port number
554  *
555  * RETURNED:    false if the port was opened successfully
556  *              true if an error occured
557  *
558  * REVISION HISTORY: V 1.0      04/05/2000
559  *
560  *
561  ***********************************************************************/
562 static Boolean DeactivateSerialPort(void)
563 {
564         Err     err;
565
566         if (gSerialRefNum != 0)
567         {
568                 SrmReceiveFlush(gSerialRefNum, 1);
569                 err = SrmClose(gSerialRefNum);
570                 ErrNonFatalDisplayIf(err != 0, "Can't close Serial Port");
571                 return false;
572         }
573         return true;
574 }
575
576 /***********************************************************************
577  *
578  * FUNCTION:    SendSerialData
579  *
580  * DESCRIPTION: Sends data via serial port to stimulator
581  *
582  * PARAMETERS:  Pointer to string which will be transmitted
583  *                              Pointer to serial port number
584  *
585  * RETURNED:    Number of bytes sent
586  *
587  * REVISION HISTORY: V 1.2      06/12/2005
588  *
589  *
590  ***********************************************************************/
591 static UInt32 SendSerialData(Char* serStringPtr)
```

```
592 {
593         Err                     err = 0;
594         UInt32          count;
595         char            serialSendData[100];
596 //      char            serialSendData[] = "test\n"; \n newline(linefeed), \r carriage return, \t tab,...
597
598         StrCopy (serialSendData, serStringPtr);
599         if (gIrComm)
600         {
601                 if (gPalmOS5)
602                 {
603                         err = SrmControl(gSerialRefNum, srmCtlRxDisable, NULL, NULL);
604                         if (err != 0)
605                                 FrmCustomAlert(DebuggingInfoAlert, "Can't disable IR receive!", "", "");
606                         SysTaskDelay(SysTicksPerSecond() / 50);
607                 }
608                 count = SrmSend(gSerialRefNum, serialSendData, StrLen(serialSendData), &err);
609                 err = SrmSendWait(gSerialRefNum);
610                 if (err)
611                         SrmSendFlush(gSerialRefNum);
612                 ErrNonFatalDisplayIf(err != 0, "Data not transmitted");
613
614                 if (gPalmOS5)
615                 {
616                         err = SrmControl(gSerialRefNum, srmCtlRxEnable, NULL, NULL);
617                         if (err != 0)
618                                 FrmCustomAlert(DebuggingInfoAlert, "Can't enable IR receive!", "", "");
619                 }
620                 SrmReceiveFlush(gSerialRefNum,0);
621                 SysTaskDelay(SysTicksPerSecond() / 40);
622         }
623         else
624         {
625                 count = SrmSend(gSerialRefNum, serialSendData, StrLen(serialSendData), &err);
626                 err = SrmSendWait(gSerialRefNum);
627                 if (err)
628                         SrmSendFlush(gSerialRefNum);
629                 ErrNonFatalDisplayIf(err != 0, "Data not transmitted");
630                 SysTaskDelay(SysTicksPerSecond() / 50);
631         }
632         return count;
633 }
634
635 /***********************************************************************
636  *
637  * FUNCTION:    ReadSerialData
638  *
639  * DESCRIPTION: Reads data from serial port sent by the stimulator
640  *
641  * PARAMETERS:  Pointer to string which will be received
642  *                              Number of bytes to read
643  *                              Pointer to serial port number
644  *
645  * RETURNED:    Number of bytes read
646  *
647  * REVISION HISTORY: V 1.1      12/01/2006
648  *
649  *
650  ***********************************************************************/
651 static UInt32 ReadSerialData(Char* serStringPtr, UInt32 serStringSize)
652 {
653         Err                     err;
654         UInt32          count = 0;
655
656         err = SrmReceiveWait(gSerialRefNum, serStringSize - 1, 30);
657         switch (err)
658                 {
659                 case serErrLineErr:
660                         SrmReceiveFlush(gSerialRefNum, 1);
661                         break;
662
663                 case serErrTimeOut:
664                         SrmReceiveFlush(gSerialRefNum, 1);
665                         break;
666
667                 default:
668                         count = SrmReceive(gSerialRefNum, serStringPtr, serStringSize - 1, 0, &err);
669                         break;
670                 }
671         *(serStringPtr + count) = '\0'; //null terminate serString
672         return count;
673 }
674
675 /***********************************************************************
676  *
677  * FUNCTION:    CheckROMVersion
678  *
679  * DESCRIPTION: Reads ROM version of the stimulation device and
680  *                              sets global variables gNewVersion, g142Version and
681  *                              gMiFVersion according to the device
682  *
683  * PARAMETERS:
684  *
685  * RETURNED:
686  *
687  * REVISION HISTORY: V 1.4      12/01/2006
688  *
689  *
690  ***********************************************************************/
691 static Boolean CheckROMVersion(void)
```



```
692 {
693         UInt32  count;
694         Char    ROMString[12], ROMReadString[] = "X";
695         Boolean prevNew, prev142, prevMiF;
696
697         prevNew = gNewVersion;
698         prev142 = g142Version;
699         prevMiF = gMiFVersion;
700         SrmReceiveFlush(gSerialRefNum, 5);
701         count = SendSerialData(ROMReadString);
702         count = ReadSerialData(ROMString, sizeof(ROMString));
703         if (count == sizeof(ROMString) - 1)
704         {
705                 switch (ROMString[5])
706                 {
707                         case 'M':
708                                 gNewVersion = false;
709                                 g142Version = false;
710                                 gMiFVersion = false;
711                                 break;
712
713                         case 'e':
714                                 gNewVersion = false;
715                                 g142Version = true;
716                                 gMiFVersion = false;
717                                 break;
718
719                         case 'S':
720                                 gNewVersion = false;
721                                 g142Version = true;
722                                 gMiFVersion = true;
723                                 break;
724
725                         case 'J':
726                                 gNewVersion = true;
727                                 g142Version = false;
728                                 gMiFVersion = true;
729                                 break;
730
731                         default:
732                                 gNewVersion = false;
733                                 g142Version = false;
734                                 gMiFVersion = false;
735                                 break;
736                 }
737         }
738         SysTaskDelay(SysTicksPerSecond() / 20);
739         SrmReceiveFlush(gSerialRefNum, 5);
740         if ((prevNew == gNewVersion) && (prevMiF == gMiFVersion) && (prev142 == g142Version))
741                 return false;
742         else
743                 return true;
744 }
745
746 /***********************************************************************
747  *
748  * FUNCTION:    TestSerialCommunication
749  *
750  * DESCRIPTION: Reads input data and displays it in Receive text field
751  *
752  *
753  * PARAMETERS:
754  *
755  * RETURNED:    Nothing
756  *
757  * REVISION HISTORY: V 1.2      20/12/2005 09:13
758  *
759  *
760  ***********************************************************************/
761 static void TestSerialCommunication(void)
762 {
763         Err             err;
764         UInt32  count;
765         Char    testString[16], inData[maxlength] = ""; //, debMsg[4];
766
767         CopyStringFromFieldText(TestCommandField, testString);
768         count = SendSerialData(testString);
769         do
770                 {
771                 FormPtr         frmP = FrmGetActiveForm();
772                 FieldPtr        fieldP;
773                 Char            receiveString[17], tempData[maxlength] = "";
774                 UInt16          currentPosition, textHeight, fieldHeight, numLines;
775
776                 err = SrmReceiveWait(gSerialRefNum, sizeof(receiveString) - 1, 30);
777                 count = SrmReceive(gSerialRefNum, receiveString, sizeof(receiveString) - 1, 0, &err);
778 //      Code for debugging: Check error number to determine if time out or line error occurs
779 //              if (err != 0)
780 //              {
781 //                      StrIToA(debMsg, err);
782 //                      FrmCustomAlert(DebuggingInfoAlert, "IR receive error:", debMsg, "");
783 //              }
784                 *(receiveString + count) = '\0';        //null terminate serString
785                 if ((StrLen(inData) + StrLen(receiveString)) > maxlength)
786                         {
787                         StrCopy(tempData, inData + StrLen(receiveString));
788                         StrCopy(inData, tempData);
789                         }
790                 StrCat(inData, receiveString);
791                 SetFieldTextFromString(TestTextField, inData);
```

```
792                 fieldP = FrmGetObjectPtr(frmP, FrmGetObjectIndex(frmP, TestTextField));
793                 FldGetScrollValues(fieldP, &currentPosition, &textHeight, &fieldHeight);
794                 numLines = textHeight - fieldHeight;
795                 LineScroll(numLines, TestTextField, TestTextScrollBar, true);
796                 }
797         while (!err);
798         SrmClearErr(gSerialRefNum);
799         ClearFieldText(TestCommandField);
800         return;
801 }
802
803 /***********************************************************************
804  *
805  * FUNCTION:    PulseShapeDraw
806  *
807  * DESCRIPTION: Draws the different pulse shape depending on
808  *                              the data of gadget
809  *
810  * PARAMETERS:
811  *
812  * RETURNED:    Nothing
813  *
814  * REVISION HISTORY: V 1.0      05/06/2000 23:43
815  *
816  *
817  ***********************************************************************/
818 static void PulseShapeDraw(FormPtr frmP)
819 {
820         MemHandle       data;
821         UInt16*         dataPtr;
822         UInt16          fromX, fromY, toX, toY;
823         RectangleType bounds;
824
825         data = FrmGetGadgetData(frmP, FrmGetObjectIndex(frmP, MainImpFGadget));
826         if (data)
827         {
828                 dataPtr = MemHandleLock(data);
829                 FrmGetObjectBounds(frmP, FrmGetObjectIndex(frmP, MainImpFGadget), &bounds);
830                 WinEraseRectangle(&bounds, 0);
831                 fromX = bounds.topLeft.x - 4;
832                 fromY = bounds.topLeft.y + bounds.extent.y / 2;
833                 toX = fromX + 4;
834                 toY = fromY;
835                 WinDrawLine(fromX, fromY, toX, toY);
836                 fromX = bounds.topLeft.x + bounds.extent.x - 1;
837                 fromY = bounds.topLeft.y + bounds.extent.y / 2;
838                 toX = fromX + 4;
839                 toY = fromY;
840                 WinDrawLine(fromX, fromY, toX, toY);
841                 switch(*dataPtr)
842                 {
843                         case 0:
844                                 fromX = bounds.topLeft.x;
845                                 fromY = bounds.topLeft.y + bounds.extent.y / 2;
846                                 toX = fromX;
847                                 toY = bounds.topLeft.y;
848                                 WinDrawLine(fromX, fromY, toX, toY);
849                                 fromX = toX;
850                                 fromY = toY;
851                                 toX = fromX + bounds.extent.x / 2;
852                                 toY = fromY;
853                                 WinDrawLine(fromX, fromY, toX, toY);
854                                 fromX = toX;
855                                 fromY = toY;
856                                 toX = fromX;
857                                 toY = fromY + bounds.extent.y - 1;
858                                 WinDrawLine(fromX, fromY, toX, toY);
859                                 fromX = toX;
860                                 fromY = toY;
861                                 toX = bounds.topLeft.x + bounds.extent.x - 1;
862                                 toY = fromY;
863                                 WinDrawLine(fromX, fromY, toX, toY);
864                                 fromX = toX;
865                                 fromY = toY;
866                                 toX = fromX;
867                                 toY = bounds.topLeft.y + bounds.extent.y / 2;
868                                 WinDrawLine(fromX, fromY, toX, toY);
869
870                                 break;
871                         case 1:
872                                 fromX = bounds.topLeft.x;
873                                 fromY = bounds.topLeft.y + bounds.extent.y / 2;
874                                 toX = fromX + bounds.extent.x / 2;
875                                 toY = bounds.topLeft.y;
876                                 WinDrawLine(fromX, fromY, toX, toY);
877                                 fromX = toX;
878                                 fromY = toY;
879                                 toX = fromX;
880                                 toY = fromY + bounds.extent.y - 1;
881                                 WinDrawLine(fromX, fromY, toX, toY);
882                                 fromX = toX;
883                                 fromY = toY;
884                                 toX = bounds.topLeft.x + bounds.extent.x - 1;
885                                 toY = bounds.topLeft.y + bounds.extent.y / 2;
886                                 WinDrawLine(fromX, fromY, toX, toY);
887                                 break;
888
889                         case 2:
890                                 fromX = bounds.topLeft.x;
```



```
891                                 fromY = bounds.topLeft.y + bounds.extent.y / 2;
892                                 toX = fromX + bounds.extent.x / 2;
893                                 toY = bounds.topLeft.y;
894                                 WinDrawLine(fromX, fromY, toX, toY);
895                                 fromX = toX;
896                                 fromY = toY;
897                                 toX = fromX;
898                                 toY = fromY + bounds.extent.y / 2;
899                                 WinDrawLine(fromX, fromY, toX, toY);
900                                 fromX = toX;
901                                 fromY = toY;
902                                 toX = bounds.topLeft.x + bounds.extent.x - 1;
903                                 toY = bounds.topLeft.y + bounds.extent.y - 1;
904                                 WinDrawLine(fromX, fromY, toX, toY);
905                                 fromX = toX;
906                                 fromY = toY;
907                                 toX = fromX;
908                                 toY = bounds.topLeft.y + bounds.extent.y / 2;
909                                 WinDrawLine(fromX, fromY, toX, toY);
910                                 break;
911
912                         case 3:
913                                 fromX = bounds.topLeft.x;
914                                 fromY = bounds.topLeft.y + bounds.extent.y / 2;
915                                 toX = fromX + bounds.extent.x / 4;
916                                 toY = bounds.topLeft.y;
917                                 WinDrawLine(fromX, fromY, toX, toY);
918                                 fromX = toX;
919                                 fromY = toY;
920                                 toX = fromX + bounds.extent.x / 2;
921                                 toY = fromY + bounds.extent.y - 1;
922                                 WinDrawLine(fromX, fromY, toX, toY);
923                                 fromX = toX;
924                                 fromY = toY;
925                                 toX = bounds.topLeft.x + bounds.extent.x - 1;
926                                 toY = bounds.topLeft.y + bounds.extent.y / 2;
927                                 WinDrawLine(fromX, fromY, toX, toY);
928                                 break;
929
930                         default:
931                                 fromX = toX = bounds.topLeft.x;
932                                 fromY = toY = bounds.topLeft.y;
933                                 break;
934                 }
935                 MemHandleUnlock(data);
936         }
937         return;
938 }
939
940 /***********************************************************************
941  *
942  * FUNCTION:    SetProgramm
943  *
944  * DESCRIPTION: Retrieves data from MainForm and sends it to the
945  *                              stimulation device
946  *
947  * PARAMETERS:
948  *
949  * RETURNED:    Nothing
950  *
951  * REVISION HISTORY: V 1.6      12/01/2006 10:15
952  *
953  *
954  ***********************************************************************/
955
956 static void SetProgramm (FormPtr frmP)
957 {
958         UInt16          akt_kanal1 = 1, akt_kanal2 = 1, nur_kanal1 = 0, hilfsvar;
959         UInt32          count;
960         Char            sendString[7], hilfsStr[9];
961         Char*           dataStringPtr;
962         Char*           trigLabelPtr;
963         MemHandle       data;
964         UInt16*         dataPtr;
965         FieldPtr        fieldP;
966
967         if (CheckROMVersion())
968                 SetMainStimType();
969
970 // Anzeige für Programmübertragung
971         fieldP = GetObjectPtr(MainTextField);
972         ClearFieldText(MainTextField);
973         StrCopy(hilfsStr, "Sendet ");
974         SetFieldTextFromString(MainTextField, hilfsStr);
975
976 // Rampe Ein Zeitdauer einstellen
977         dataStringPtr = LstGetSelectionText(GetObjectPtr(MainRamEList),
978                                                                                 LstGetSelection(GetObjectPtr(MainRamEList))
    );
979         if (g142Version && !gNewVersion)
980                 hilfsvar = (StrAToI(dataStringPtr) / 35);
981         else
982                 hilfsvar = (StrAToI(dataStringPtr) / 25);
983         StrCopy(sendString, "C");
984         StrCat(sendString, (StrIToH(hilfsStr, hilfsvar)) + 6);
985         StrCat(sendString, " ");
986         count = SendSerialData(sendString);
987
988 // Rampe Ein Anfangswert fix auf 0 stellen
989         StrCopy(sendString, "D00 ");
```

```
990         count = SendSerialData(sendString);
991
992 // Rampe Aus Zeitdauer einstellen
993         dataStringPtr = LstGetSelectionText(GetObjectPtr(MainRamALi
994                                                                     LstGetSelection(GetObjectPtr(MainRamAList))
    );
995         if (gl42Version && !gNewVersion)
996                 hilfsvar = (StrAToI(dataStringPtr) / 35);
997         else
998                 hilfsvar = (StrAToI(dataStringPtr) / 25);
999         StrCopy(sendString, "E");
1000        StrCat(sendString, (StrIToH(hilfsStr, hilfsvar)) + 6);
1001        StrCat(sendString, " ");
1002        count = SendSerialData(sendString);
1003
1004 // Rampe Aus Endwert fix auf 0 stellen
1005        StrCopy(sendString, "F00 ");
1006        count = SendSerialData(sendString);
1007
1008 // Schwelldauer einstellen
1009        dataStringPtr = LstGetSelectionText(GetObjectPtr(MainSchDLi
1010                                                                    LstGetSelection(GetObjectPtr(MainSchDList))
    );
1011        if (gl42Version && !gNewVersion)
1012                hilfsvar = 700 * StrAToI(dataStringPtr);
1013        else
1014                hilfsvar = 1000 * StrAToI(dataStringPtr);
1015        StrPrintF(sendString, "G%x ", hilfsvar);
1016        count = SendSerialData(sendString);
1017
1018 // Schwellpause einstellen
1019        dataStringPtr = LstGetSelectionText(GetObjectPtr(MainSchPLi
1020                                                                    LstGetSelection(GetObjectPtr(MainSchPList))
    );
1021        if (gl42Version && !gNewVersion)
1022                hilfsvar = 700 * StrAToI(dataStringPtr);
1023        else
1024                hilfsvar = 1000 * StrAToI(dataStringPtr);
1025        if (hilfsvar == 0)
1026                hilfsvar = 1;
1027        StrPrintF(sendString, "H%x ", hilfsvar);
1028        count = SendSerialData(sendString);
1029
1030 // Impulsdauer einstellen
1031        hilfsvar = LstGetSelection(GetObjectPtr(MainImpDList));
1032        StrCopy(sendString, "A");
1033        if (gl42Version && !gNewVersion)
1034                StrCat(sendString, (StrIToH(hilfsStr, hor_ver_step_dauer142[hilfsvar][0])) + 6);
1035        else
1036                StrCat(sendString, (StrIToH(hilfsStr, hor_ver_step_dauer100[hilfsvar][0])) + 6);
1037        StrCat(sendString, " ");
1038        count = SendSerialData(sendString); // Horizontal Step
1039        StrCopy(sendString, "B");
1040        if (gl42Version && !gNewVersion)
1041                StrCat(sendString, (StrIToH(hilfsStr, hor_ver_step_dauer142[hilfsvar][1])) + 6);
1042        else
1043                StrCat(sendString, (StrIToH(hilfsStr, hor_ver_step_dauer100[hilfsvar][1])) + 6);
1044        StrCat(sendString, " ");
1045        count = SendSerialData(sendString); // Vertikal Step
1046
1047 // Impulspause einstellen
1048        dataStringPtr = LstGetSelectionText(GetObjectPtr(MainImpPLi
1049                                                                    LstGetSelection(GetObjectPtr(MainImpPList))
    );
1050        if (gNewVersion)
1051        {
1052                hilfsvar = StrAToI(dataStringPtr) * 10;
1053                StrPrintF(sendString, "I%x ", hilfsvar);
1054        }
1055        if (gl42Version)
1056        {
1057                hilfsvar = StrAToI(dataStringPtr) * 7;
1058                StrPrintF(sendString, "I%x ", hilfsvar);
1059        }
1060        if (!gl42Version && !gNewVersion)
1061        {
1062                if (!((StrAToI(dataStringPtr) * 10) > hor_ver_step_dauer100[hilfsvar][2]))
1063                {
1064                        nur_kanal1 = 1;
1065                        LstSetSelection(GetObjectPtr(MainAmpl2List), 0);
1066                        trigLabelPtr = LstGetSelectionText(GetObjectPtr(MainAmpl2List),
1067                                                                                LstGetSelection(GetObjectPtr(MainAm
    pl2List)));
1068                        CtlSetLabel(GetObjectPtr(MainAmpl2PopTrigger), trigLabelPtr);
1069                        hilfsvar = StrAToI(dataStringPtr) * 10;
1070                        StrPrintF(sendString, "I%x ", hilfsvar);
1071                }
1072                else
1073                {
1074                        hilfsvar = ((StrAToI(dataStringPtr) * 10) - hor_ver_step_dauer100[hilfsvar][2]) / 2;
1075                        StrPrintF(sendString, "I%x ", hilfsvar);
1076                }
1077        }
1078        count = SendSerialData(sendString);
1079
1080 // MiF Impulsanzahl einstellen (nach welcher Anzahl von Impulsen folgt MiF-Impuls)
1081        if (gMiFVersion)
1082        {
1083                dataStringPtr = LstGetSelectionText(GetObjectPtr(MainMiFAnzList),
1084                                                                        LstGetSelection(GetObjectPtr(MainMi
```



```
    FAnzList)));
1085                if (!(*dataStringPtr == 'D'))
1086                        hilfsvar = StrAToI(dataStringPtr);
1087                else
1088                        hilfsvar = 0;
1089                StrCopy(sendString, "(");
1090                StrCat(sendString, (StrIToH(hilfsStr, hilfsvar)) + 6);
1091                StrCat(sendString, " ");
1092                count = SendSerialData(sendString);
1093        }
1094
1095 // MiF Impulsdauer einstellen
1096        if (gMiFVersion)
1097        {
1098                hilfsvar = LstGetSelection(GetObjectPtr(MainMiFImpDList));
1099                StrCopy(sendString, "+");
1100                if (gNewVersion)
1101                {
1102                        StrCat(sendString, (StrIToH(hilfsStr, hor_ver_step_dauer100[hilfsvar][1])) + 6);
1103                        StrCat(sendString, (StrIToH(hilfsStr, hor_ver_step_dauer100[hilfsvar][0])) + 6);
1104                }
1105                if (gl42Version)
1106                {
1107                        StrCat(sendString, (StrIToH(hilfsStr, hor_ver_step_dauer142[hilfsvar][1])) + 6);
1108                        StrCat(sendString, (StrIToH(hilfsStr, hor_ver_step_dauer142[hilfsvar][0])) + 6);
1109                }
1110                StrCat(sendString, " ");
1111                count = SendSerialData(sendString); // Vertikal Step, Horizontal Step
1112        }
1113
1114 // Grundwert Kanal 1 einstellen
1115        dataStringPtr = LstGetSelectionText(GetObjectPtr(MainAmpl1List),
1116                                                                        LstGetSelection(GetObjectPtr(MainAmpl1List)
    ));
1117        if (!(*dataStringPtr == 'D'))
1118        {
1119                hilfsvar = ((float)StrAToI(dataStringPtr) * 255 / 100);
1120                StrCopy(sendString, "J");
1121                StrCat(sendString, (StrIToH(hilfsStr, hilfsvar)) + 6);
1122                StrCat(sendString, " ");
1123                count = SendSerialData(sendString);
1124        }
1125        else
1126                akt_kanal1 = 0;
1127
1128 // Grundwert Kanal 2 einstellen
1129        dataStringPtr = LstGetSelectionText(GetObjectPtr(MainAmpl2List),
1130                                                                        LstGetSelection(GetObjectPtr(MainAmpl2List)
    ));
1131        if (!(*dataStringPtr == 'D'))
1132        {
1133                hilfsvar = ((float)StrAToI(dataStringPtr) * 255 / 100);
1134                StrCopy(sendString, "K");
1135                StrCat(sendString, (StrIToH(hilfsStr, hilfsvar)) + 6);
1136                StrCat(sendString, " ");
1137                count = SendSerialData(sendString);
1138        }
1139        else
1140                akt_kanal2 = 0;
1141
1142 // Stimulationskanäle Ein/Aus, 1 Kanalbetrieb aktivieren
1143        hilfsvar = akt_kanal1 + akt_kanal2 * 2 + nur_kanal1 * 4 + gl42Version * 16;
1144        StrCopy(sendString, "Q");
1145        StrCat(sendString, (StrIToH(hilfsStr, hilfsvar)) + 6);
1146        StrCat(sendString, " ");
1147        count = SendSerialData(sendString);
1148
1149 // Encoder1 fix auf 1 einstellen
1150        hilfsvar = ((float)1 * 255 / 100 + 0.5);
1151        StrCopy(sendString, "L");
1152        StrCat(sendString, (StrIToH(hilfsStr, hilfsvar)) + 6);
1153        StrCat(sendString, " ");
1154        count = SendSerialData(sendString);
1155
1156 // Encoder2 fix auf 1 einstellen
1157        hilfsvar = ((float)1 * 255 / 100 + 0.5);
1158        StrCopy(sendString, "M");
1159        StrCat(sendString, (StrIToH(hilfsStr, hilfsvar)) + 6);
1160        StrCat(sendString, " ");
1161        count = SendSerialData(sendString);
1162
1163 // Stimulationsimpulsform einstellen
1164        data = FrmGetGadgetData(frmP, FrmGetObjectIndex(frmP, MainImpFGadget));
1165        if (data)
1166        {
1167                dataPtr = MemHandleLock(data);
1168                hilfsvar = *dataPtr;
1169                MemHandleUnlock(data);
1170                StrCopy(sendString, "P");
1171                StrCat(sendString, (StrIToH(hilfsStr, hilfsvar)) + 6);
1172                StrCat(sendString, " ");
1173                count = SendSerialData(sendString);
1174        }
1175
1176 // Stimulationszeitdauer einstellen
1177        dataStringPtr = LstGetSelectionText(GetObjectPtr(MainProgDList),
1178                                                                        LstGetSelection(GetObjectPtr(MainProgDList)
    ));
1179        StrCopy(sendString, "T");
1180        StrCat(sendString, dataStringPtr);
```

```
1181        StrCat(sendString, " ");
1182        count = SendSerialData(sendString);
1183
1184 // Encoder Max fix auf 99 einstellen
1185        hilfsvar = 99 * 255 / 100;
1186        StrCopy(sendString, "V");
1187        StrCat(sendString, (StrIToH(hilfsStr, hilfsvar)) + 6);
1188        StrCat(sendString, " ");
1189        count = SendSerialData(sendString);
1190
1191        if (g142Version || gNewVersion)
1192        {
1193 // Impedanzmaximum auf ? einstellen
1194                StrCopy(sendString, "YFFFF ");
1195                count = SendSerialData(sendString);
1196
1197 // Impedanzminimum auf ? einstellen
1198                StrCopy(sendString, "Z0000 ");
1199                count = SendSerialData(sendString);
1200        }
1201
1202 // Programm in Programmspeicher Nr. 1 - 9 übernehmen
1203        if (CtlGetValue(GetObjectPtr(MainProg1PushButton)))
1204                StrCopy(sendString, "N1 ");
1205        if (CtlGetValue(GetObjectPtr(MainProg2PushButton)))
1206                StrCopy(sendString, "N2 ");
1207        if (CtlGetValue(GetObjectPtr(MainProg3PushButton)))
1208                StrCopy(sendString, "N3 ");
1209        if (CtlGetValue(GetObjectPtr(MainProg4PushButton)))
1210                StrCopy(sendString, "N4 ");
1211        if (CtlGetValue(GetObjectPtr(MainProg5PushButton)))
1212                StrCopy(sendString, "N5 ");
1213        if (CtlGetValue(GetObjectPtr(MainProg6PushButton)))
1214                StrCopy(sendString, "N6 ");
1215        if (CtlGetValue(GetObjectPtr(MainProg7PushButton)))
1216                StrCopy(sendString, "N7 ");
1217        if (CtlGetValue(GetObjectPtr(MainProg8PushButton)))
1218                StrCopy(sendString, "N8 ");
1219        if (CtlGetValue(GetObjectPtr(MainProg9PushButton)))
1220                StrCopy(sendString, "N9 ");
1221        count = SendSerialData(sendString);
1222
1223        if (!(gIrComm || gRS232ßE1))
1224        {
1225                // Wait until all characters are echoed
1226                SrmReceiveWait(gSerialRefNum, StrLen(sendString), 15);
1227                SrmReceiveFlush(gSerialRefNum, 1);
1228        }
1229
1230 // Warten bis Programm in den Speicher übernommen
1231        SysTaskDelay(SysTicksPerSecond() / 5);
1232
1233 //     FrmAlert(DebuggingInfoAlert);
1234        return;
1235 }
1236
1237 /*****************************************************************
1238  *
1239  * FUNCTION:    VerifyProgramm
1240  *
1241  * DESCRIPTION: Compares data from MainForm and stimulation device
1242  *                          and sends incorrect settings again
1243  *
1244  * PARAMETERS:
1245  *
1246  * RETURNED:    Nothing
1247  *
1248  * REVISION HISTORY: V 1.6      12/01/2006 10:23
1249  *
1250  *
1251  *****************************************************************/
1252 static void VerifyProgramm(FormPtr frmP)
1253 {
1254        Err                 err;
1255        UInt16          hilfsvar, impulsDauer;
1256        UInt16          repetition = 0,akt_kanal1 = 1, akt_kanal2 = 1, nuc_kanal1 = 0, echo = 0;
1257        UInt32          count;
1258        Char            progString[130], sendString[7], hilfsStr[9], infoStr[9], debMsg[4];
1259        Char*           trigLabelPtr;
1260        MemHandle       data;
1261        Boolean         repeat = false;
1262        FieldPtr        fieldP;
1263
1264
1265 // Code for Debugging Start
1266 //                      fieldP = GetObjectPtr(MainTextField);
1267 //                      ClearFieldText(MainTextField);
1268 //                      StrCopy(sendString, (StrIToH(hilfsStr, (UInt8)progString[0x3F + echo]) + 6));
1269 //                      SetFieldTextFromString(MainTextField, sendString);
1270 //                      FrmAlert(DebuggingInfoAlert);
1271 // Code for Debugging End
1272
1273        if (CheckROMVersion())
1274                SetMainStimType();
1275        SysTaskDelay(SysTicksPerSecond() / 10);
1276        if (gPalmOS5)
1277                SysTaskDelay(SysTicksPerSecond() / 10);
1278
1279        do
1280        {
```



```
1281                repeat = false;
1282                ++repetition;
1283
1284 // Anzeige für Anzahl der Programmübertragungversuche aktuallisieren
1285                fieldP = GetObjectPtr(MainTextField);
1286                ClearFieldText(MainTextField);
1287                StrCopy(infoStr, "Sendet ");
1288                StrCat(infoStr, StrIToA(hilfsStr, (UInt32)repetition));
1289                SetFieldTextFromString(MainTextField, infoStr);
1290
1291 // Laden vom Programmspeicher Programm Nr. 1 - 9
1292                if (CtlGetValue(GetObjectPtr(MainProg1PushButton)))
1293                        StrCopy(sendString, "O1 ");
1294                if (CtlGetValue(GetObjectPtr(MainProg2PushButton)))
1295                        StrCopy(sendString, "O2 ");
1296                if (CtlGetValue(GetObjectPtr(MainProg3PushButton)))
1297                        StrCopy(sendString, "O3 ");
1298                if (g142Version || gNewVersion)
1299                {
1300                        if (CtlGetValue(GetObjectPtr(MainProg4PushButton)))
1301                                StrCopy(sendString, "O4 ");
1302                        if (CtlGetValue(GetObjectPtr(MainProg5PushButton)))
1303                                StrCopy(sendString, "O5 ");
1304                        if (CtlGetValue(GetObjectPtr(MainProg6PushButton)))
1305                                StrCopy(sendString, "O6 ");
1306                        if (CtlGetValue(GetObjectPtr(MainProg7PushButton)))
1307                                StrCopy(sendString, "O7 ");
1308                        if (CtlGetValue(GetObjectPtr(MainProg8PushButton)))
1309                                StrCopy(sendString, "O8 ");
1310                        if (CtlGetValue(GetObjectPtr(MainProg9PushButton)))
1311                                StrCopy(sendString, "O9 ");
1312                }
1313
1314                count = SendSerialData(sendString);
1315
1316                if (!(gIrComm || gRS232ßE1))
1317                {
1318                        // Wait until all characters are echoed
1319                        SrmReceiveWait(gSerialRefNum, StrLen(sendString), 15);
1320                        SrmReceiveFlush(gSerialRefNum, 1);
1321                }
1322                SysTaskDelay(SysTicksPerSecond() / 5);
1323
1324        // Auslesen der Programmparameter (128 Bytes + "?" + '\0')
1325                StrCopy(sendString, "?");
1326                count = SendSerialData(sendString);
1327                if (gIrComm || gRS232ßE1)
1328                {
1329                        err = SrmReceiveWait(gSerialRefNum, sizeof(progString) - 2, 120);
1330                        count = SrmReceive(gSerialRefNum, progString, sizeof(progString) - 2, 0, &err);
1331                }
1332                else
1333                {
1334                        err = SrmReceiveWait(gSerialRefNum, sizeof(progString) - 1, 120);
1335                        count = SrmReceive(gSerialRefNum, progString, sizeof(progString) - 1, 0, &err);
1336                }
1337                if (err)
1338                {
1339                        StrIToA(debMsg, err);
1340                        FrmCustomAlert(DebuggingInfoAlert, "Error", debMsg, "while receiving device settings!");
1341                }
1342                else
1343                {
1344                        *(progString + count) = '\0'; // null terminate progString
1345                        if (!(gIrComm || gRS2328E1))
1346                                echo = 1;
1347
1348                        if (!(progString[0x21 + echo] & 0x01))
1349                        {
1350                                // Kanal 1 deaktiviert
1351                                trigLabelPtr = LstGetSelectionText(GetObjectPtr(MainAmpl1List),
1352                                                                                                LstGetSelection(GetObjectPtr(MainAm
    pl1List)));
1353                                if (!(*trigLabelPtr == 'D'))
1354                                {
1355                                        akt_kanal1 = 0;
1356                                        repeat = true;
1357                                }
1358                        }
1359                        else
1360                        {
1361                                // Kanal 1 Grundamplitude vergleichen
1362                                trigLabelPtr = LstGetSelectionText(GetObjectPtr(MainAmpl1List),
1363                                                                                                LstGetSelection(GetObjectPtr(MainAm
    pl1List)));
1364                                hilfsvar = ((float)StrAToI(trigLabelPtr) * 255 / 100);
1365                                if (!((UInt8)progString[0x3F + echo] == hilfsvar))
1366                                {
1367                                        StrCopy(sendString, "J");
1368                                        StrCat(sendString, (StrIToH(hilfsStr, hilfsvar)) + 6);
1369                                        StrCat(sendString, " ");
1370                                        count = SendSerialData(sendString);
1371                                        repeat = true;
1372                                }
1373                        }
1374
1375                        if (!(progString[0x21 + echo] & 0x02))
1376                        {
1377                                // Kanal 2 deaktiviert
1378                                trigLabelPtr = LstGetSelectionText(GetObjectPtr(MainAmpl2List),
```

```
1379                                                                                    LstGetSelection(GetObjectPtr(MainAm
    pl2List)));
1380                            if (!(*trigLabelPtr == 'D'))
1381                            {
1382                                    akt_kanal2 = 0;
1383                                    repeat = true;
1384                            }
1385                    }
1386                    else
1387                    {
1388                            // Kanal 2 Grundamplitude vergleichen
1389                            trigLabelPtr = LstGetSelectionText(GetObjectPtr(MainAmpl2List),
1390                                                                                    LstGetSelection(GetObjectPtr(MainAm
    pl2List)));
1391                            hilfsvar = ((float)StrAToI(trigLabelPtr) * 255 / 100);
1392                            if (!((UInt8)progString[0x40 + echo] == hilfsvar))
1393                            {
1394                                    StrCopy(sendString, "K");
1395                                    StrCat(sendString, (StrIToH(hilfsStr, hilfsvar)) + 6);
1396                                    StrCat(sendString, " ");
1397                                    count = SendSerialData(sendString);
1398                                    repeat = true;
1399                            }
1400                    }
1401
1402                    // Impulsdauer vergleichen
1403                    hilfsvar = LstGetSelection(GetObjectPtr(MainImpDList));
1404                    if (g142Version && !gNewVersion)
1405                    {
1406                            impulsDauer = hor_ver_step_dauer142[hilfsvar][2];
1407                            if (!((((UInt8)progString[0x32 + echo]) == hor_ver_step_dauer142[hilfsvar][0]) &&
1408                                                    (((UInt8)progString[0x33 + echo]) == hor_ver_step_dauer142[hilfsvar][1])))
1409                            {
1410                                    StrCopy(sendString, "A");
1411                                    StrCat(sendString, (StrIToH(hilfsStr, hor_ver_step_dauer142[hilfsvar][0])) + 6);
1412                                    StrCat(sendString, " ");
1413                                    count = SendSerialData(sendString); // Horizontal Step
1414                                    StrCopy(sendString, "B");
1415                                    StrCat(sendString, (StrIToH(hilfsStr, hor_ver_step_dauer142[hilfsvar][1])) + 6);
1416                                    StrCat(sendString, " ");
1417                                    count = SendSerialData(sendString); // Vertikal Step
1418                                    repeat = true;
1419                            }
1420                    }
1421                    else
1422                    {
1423                            impulsDauer = hor_ver_step_dauer100[hilfsvar][2];
1424                            if (!((((UInt8)progString[0x32 + echo]) == hor_ver_step_dauer100[hilfsvar][0]) &&
1425                                                    (((UInt8)progString[0x33 + echo]) == hor_ver_step_dauer100[hilfsvar][1])))
1426                            {
1427                                    StrCopy(sendString, "A");
1428                                    StrCat(sendString, (StrIToH(hilfsStr, hor_ver_step_dauer100[hilfsvar][0])) + 6);
1429                                    StrCat(sendString, " ");
1430                                    count = SendSerialData(sendString); // Horizontal Step
1431                                    StrCopy(sendString, "B");
1432                                    StrCat(sendString, (StrIToH(hilfsStr, hor_ver_step_dauer100[hilfsvar][1])) + 6);
1433                                    StrCat(sendString, " ");
1434                                    count = SendSerialData(sendString); // Vertikal Step
1435                                    repeat = true;
1436                            }
1437                    }
1438
1439                    // Impulspause vergleichen
1440                    if (g142Version && !gNewVersion)
1441                            hilfsvar = StrAToI(LstGetSelectionText(GetObjectPtr(MainImpPList),
1442                                                                                    LstGetSelection(GetObjectPtr(MainImpPList))
    )) * 7;
1443                    else
1444                            hilfsvar = StrAToI(LstGetSelectionText(GetObjectPtr(MainImpPList),
1445                                                                                    LstGetSelection(GetObjectPtr(MainImpPList))
    )) * 10;
1446                    if (!((UInt8)progString[0x21 + echo] & 0x04) && !g142Version && !gNewVersion)
1447                            if (hilfsvar < impulsDauer)
1448                            {
1449                                    nur_kanal1 = 1;
1450                                    repeat = true;
1451                            }
1452                            else
1453                                    hilfsvar = (hilfsvar - impulsDauer) / 2;
1454                    if (!((UInt8)progString[0x3D + echo] * 256 + (UInt8)progString[0x3C + echo] == hilfsvar))
1455                    {
1456                            StrPrintF(sendString, "I%x ", hilfsvar);
1457                            count = SendSerialData(sendString);
1458                            repeat = true;
1459                    }
1460
1461                    // Übertragen der Statusbits für Stimulationskanäle Ein/Aus, 1 Kanalbetrieb
1462                    if (repeat)
1463                    {
1464                            hilfsvar = akt_kanal1 + akt_kanal2 * 2 + nur_kanal1 * 4 + (g142Version || gNewVersion) * 16;
1465                            StrCopy(sendString, "Q");
1466                            StrCat(sendString, (StrIToH(hilfsStr, hilfsvar)) + 6);
1467                            StrCat(sendString, " ");
1468                            count = SendSerialData(sendString);
1469                    }
1470
1471                    // MiF Impulsanzahl vergleichen (nach welcher Anzahl von Impulsen folgt MiF-Impuls)
1472                    if (gMiFVersion)
1473                    {
1474                            trigLabelPtr = LstGetSelectionText(GetObjectPtr(MainMiFAnzList),
```



```
1475                                                                                    LstGetSelection(GetObjectPtr(MainMi
    FAnzList)));
1476                            if (!(*trigLabelPtr == 'D'))
1477                                    hilfsvar = StrAToI(trigLabelPtr);
1478                            else
1479                                    hilfsvar = 0;
1480                            if (!((((UInt8)progString[0x48 + echo]) == hilfsvar))
1481                            {
1482                                    trigLabelPtr = LstGetSelectionText(GetObjectPtr(MainMiFAnzList),
1483                                                                                                                    LstGetSelec
    tion(GetObjectPtr(MainMiFAnzList)));
1484                                    if (!(*trigLabelPtr == 'D'))
1485                                            hilfsvar = StrAToI(trigLabelPtr);
1486                                    else
1487                                            hilfsvar = 0;
1488                                    StrCopy(sendString, "(");
1489                                    StrCat(sendString, (StrIToH(hilfsStr, hilfsvar)) + 6);
1490                                    StrCat(sendString, " ");
1491                                    count = SendSerialData(sendString);
1492                                    repeat = true;
1493                            }
1494                    }
1495
1496                    // MiF Impulsdauer vergleichen
1497                    if (gMiFVersion)
1498                    {
1499                            hilfsvar = LstGetSelection(GetObjectPtr(MainMiFImpDList));
1500                            if (g142Version && !gNewVersion)
1501                            {
1502                                    impulsDauer = hor_ver_step_dauer142[hilfsvar][2];
1503                                    if (!((((UInt8)progString[0x49 + echo]) == hor_ver_step_dauer142[hilfsvar][0]) &&
1504                                                                    (((UInt8)progString[0x4A + echo]) == hor_ver_step_dauer142[hilfsvar
    ][1])))
1505                                    {
1506                                            StrCopy(sendString, "+");
1507                                            StrCat(sendString, (StrIToH(hilfsStr, hor_ver_step_dauer142[hilfsvar][1])) + 6);
1508                                            StrCat(sendString, (StrIToH(hilfsStr, hor_ver_step_dauer142[hilfsvar][0])) + 6);
1509                                            StrCat(sendString, " ");
1510                                            count = SendSerialData(sendString); // Vertikal Step, Horizontal Step
1511                                            repeat = true;
1512                                    }
1513                            }
1514                            else
1515                            {
1516                                    impulsDauer = hor_ver_step_dauer100[hilfsvar][2];
1517                                    if (!((((UInt8)progString[0x49 + echo]) == hor_ver_step_dauer100[hilfsvar][0]) &&
1518                                                                    (((UInt8)progString[0x4A + echo]) == hor_ver_step_dauer100[hilfsvar
    ][1])))
1519                                    {
1520                                            StrCopy(sendString, "+");
1521                                            StrCat(sendString, (StrIToH(hilfsStr, hor_ver_step_dauer100[hilfsvar][1])) + 6);
1522                                            StrCat(sendString, (StrIToH(hilfsStr, hor_ver_step_dauer100[hilfsvar][0])) + 6);
1523                                            StrCat(sendString, " ");
1524                                            count = SendSerialData(sendString); // Vertikal Step, Horizontal Step
1525                                            repeat = true;
1526                                    }
1527                            }
1528                    }
1529
1530                    // Schwelldauer vergleichen
1531                    if (g142Version && !gNewVersion)
1532                            hilfsvar = StrAToI(LstGetSelectionText(GetObjectPtr(MainSchDList),
1533                                                                                    LstGetSelection(GetObjectPtr(MainSchDList))
    )) * 700;
1534                    else
1535                            hilfsvar = StrAToI(LstGetSelectionText(GetObjectPtr(MainSchDList),
1536                                                                                    LstGetSelection(GetObjectPtr(MainSchDList))
    )) * 1000;
1537                    if (!((((UInt8)progString[0x39 + echo] * 256 + (UInt8)progString[0x38 + echo]) == hilfsvar))
1538                    {
1539                            StrPrintF(sendString, "G%x ", hilfsvar);
1540                            count = SendSerialData(sendString);
1541                            repeat = true;
1542                    }
1543
1544                    // Schwellpause vergleichen
1545                    if (g142Version && !gNewVersion)
1546                            hilfsvar = StrAToI(LstGetSelectionText(GetObjectPtr(MainSchPList),
1547                                                                                    LstGetSelection(GetObjectPtr(MainSchPList))
    )) * 700;
1548                    else
1549                            hilfsvar = StrAToI(LstGetSelectionText(GetObjectPtr(MainSchPList),
1550                                                                                    LstGetSelection(GetObjectPtr(MainSchPList))
    )) * 1000;
1551                    if (hilfsvar == 0)
1552                            hilfsvar = 1;
1553                    if (!((((UInt8)progString[0x3B + echo] * 256 + (UInt8)progString[0x3A + echo]) == hilfsvar))
1554                    {
1555                            StrPrintF(sendString, "H%x ", hilfsvar);
1556                            count = SendSerialData(sendString);
1557                            repeat = true;
1558                    }
1559
1560                    // Rampe Ein Zeitdauer vergleichen
1561                    trigLabelPtr = LstGetSelectionText(GetObjectPtr(MainRamEList),
1562                                                                                    LstGetSelection(GetObjectPtr(MainRamEList))
    );
1563                    if (g142Version && !gNewVersion)
1564                            hilfsvar = (StrAToI(trigLabelPtr) / 35);
1565                    else
```

```
1566                            hilfsvar = (StrAToI(trigLabelPtr) / 25);
1567                    if (!((UInt8)progString[0x34 + echo] == hilfsvar))
1568                    {
1569                            StrCopy(sendString, "C");
1570                            StrCat(sendString, (StrIToH(hilfsStr, hilfsvar)) + 6);
1571                            StrCat(sendString, " ");
1572                            count = SendSerialData(sendString);
1573                            repeat = true;
1574                    }
1575
1576                    // Rampe Aus Zeitdauer vergleichen
1577                    trigLabelPtr = LstGetSelectionText(GetObjectPtr(MainRamAList),
1578                                                                    LstGetSelection(GetObjectPtr(MainRamAList))
        );
1579                    if (g142Version && !gNewVersion)
1580                            hilfsvar = (StrAToI(trigLabelPtr) / 35);
1581                    else
1582                            hilfsvar = (StrAToI(trigLabelPtr) / 25);
1583                    if (!((UInt8)progString[0x36 + echo] == hilfsvar))
1584                    {
1585                            StrCopy(sendString, "E");
1586                            StrCat(sendString, (StrIToH(hilfsStr, hilfsvar)) + 6);
1587                            StrCat(sendString, " ");
1588                            count = SendSerialData(sendString);
1589                            repeat = true;
1590                    }
1591
1592                    //      Programmdauer vergleichen
1593                    trigLabelPtr = LstGetSelectionText(GetObjectPtr(MainProgDList),
1594                                                                    LstGetSelection(GetObjectPtr(MainProgDList)
    ));
1595                    hilfsvar = StrAToI(trigLabelPtr);
1596                    if (!(((UInt8)((UInt8)progString[0x41 + echo] / 16) * 10 + (UInt8)progString[0x41 + echo] % 16) == hilfsvar
    ))
1597                    {
1598                            StrCopy(sendString, "T");
1599                            StrCat(sendString, trigLabelPtr);
1600                            StrCat(sendString, " ");
1601                            count = SendSerialData(sendString);
1602                            repeat = true;
1603                    }
1604
1605                    // Impulsform vergleichen
1606                    data = FrmGetGadgetData(frmP, FrmGetObjectIndex(frmP, MainImpFGadget));
1607                    if (data)
1608                    {
1609                            UInt16* dataPtr;
1610
1611                            dataPtr = MemHandleLock(data);
1612                            hilfsvar = *dataPtr;
1613                            MemHandleUnlock(data);
1614                            if (!(hilfsvar == (UInt16)progString[0x3E + echo]))
1615                            {
1616                                    StrCopy(sendString, "P");
1617                                    StrCat(sendString, (StrIToH(hilfsStr, hilfsvar)) + 6);
1618                                    StrCat(sendString, " ");
1619                                    count = SendSerialData(sendString);
1620                                    repeat = true;
1621                            }
1622                    }
1623
1624                    // Encoder Maximum vergleichen (fix auf 99 eingestellt)
1625                    hilfsvar = 99 * 255 / 100;
1626                    if (!((UInt8)progString[0x43 + echo] == hilfsvar))
1627                    {
1628                            StrCopy(sendString, "V");
1629                            StrCat(sendString, (StrIToH(hilfsStr, hilfsvar)) + 6);
1630                            StrCat(sendString, " ");
1631                            count = SendSerialData(sendString);
1632                            repeat = true;
1633                    }
1634
1635                    // Encoder1 vergleichen (fix auf 1 eingestellt)
1636                    hilfsvar = ((float)1 * 255 / 100 + 0.5);
1637                    if (!((UInt8)progString[0x30 + echo] == hilfsvar))
1638                    {
1639                            StrCopy(sendString, "L");
1640                            StrCat(sendString, (StrIToH(hilfsStr, hilfsvar)) + 6);
1641                            StrCat(sendString, " ");
1642                            count = SendSerialData(sendString);
1643                            repeat = true;
1644                    }
1645
1646                    // Encoder2 vergleichen (fix auf 1 eingestellt)
1647                    hilfsvar = ((float)1 * 255 / 100 + 0.5);
1648                    if (!((UInt8)progString[0x31 + echo] == hilfsvar))
1649                    {
1650                            StrCopy(sendString, "M");
1651                            StrCat(sendString, (StrIToH(hilfsStr, hilfsvar)) + 6);
1652                            StrCat(sendString, " ");
1653                            count = SendSerialData(sendString);
1654                            repeat = true;
1655                    }
1656
1657                    if (g142Version || gNewVersion)
1658                    {
1659                    // Impedanzmaximum auf ? einstellen
1660                            hilfsvar = 0xFFFF;
1661                            if (!((UInt8)progString[0x45 + echo] * 256 + (UInt8)progString[0x44 + echo] == hilfsvar))
1662                            {
```



```
1663                                    StrPrintF(sendString, "Y%x ", hilfsvar);
1664                                    count = SendSerialData(sendString);
1665                                    repeat = true;
1666                            }
1667
1668                    // Impedanzminimum auf ? einstellen
1669                            hilfsvar = 0x0000;
1670                            if (!((UInt8)progString[0x47 + echo] * 256 + (UInt8)progString[0x46 + echo] == hilfsvar))
1671                            {
1672                                    StrPrintF(sendString, "Z%x ", hilfsvar);
1673                                    count = SendSerialData(sendString);
1674                                    repeat = true;
1675                            }
1676                    }
1677
1678
1679                    // Wenn repeat dann Programm neu in den Programmspeicher übernehmen
1680                    if (repeat)
1681                    {
1682                            if (CtlGetValue(GetObjectPtr(MainProg1PushButton)))
1683                                    StrCopy(sendString, "N1 ");
1684                            if (CtlGetValue(GetObjectPtr(MainProg2PushButton)))
1685                                    StrCopy(sendString, "N2 ");
1686                            if (CtlGetValue(GetObjectPtr(MainProg3PushButton)))
1687                                    StrCopy(sendString, "N3 ");
1688                            if (CtlGetValue(GetObjectPtr(MainProg4PushButton)))
1689                                    StrCopy(sendString, "N4 ");
1690                            if (CtlGetValue(GetObjectPtr(MainProg5PushButton)))
1691                                    StrCopy(sendString, "N5 ");
1692                            if (CtlGetValue(GetObjectPtr(MainProg6PushButton)))
1693                                    StrCopy(sendString, "N6 ");
1694                            if (CtlGetValue(GetObjectPtr(MainProg7PushButton)))
1695                                    StrCopy(sendString, "N7 ");
1696                            if (CtlGetValue(GetObjectPtr(MainProg8PushButton)))
1697                                    StrCopy(sendString, "N8 ");
1698                            if (CtlGetValue(GetObjectPtr(MainProg9PushButton)))
1699                                    StrCopy(sendString, "N9 ");
1700
1701                            count = SendSerialData(sendString);
1702
1703                            if (!(gIrComm || gRS2328E1))
1704                            {
1705                                    // Wait until all characters are echoed
1706                                    SrmReceiveWait(gSerialRefNum, StrLen(sendString), 15);
1707                                    SrmReceiveFlush(gSerialRefNum, 1);
1708                            }
1709                    // Warten bis Programm in den Speicher übernommen
1710                            SysTaskDelay(SysTicksPerSecond() / 5);
1711                    }
1712            }
1713
1714            }       while ((repeat) && (repetition < 3));
1715
1716            if (repetition == 3)
1717                    FrmAlert(TransmittingErrorAlert);
1718
1719            SetMainStimType();
1720            return;
1721 }
1722
1723 /*****************************************************************************
1724  *
1725  * FUNCTION:    ReadProgramm
1726  *
1727  * DESCRIPTION: Retrieves data from MainForm and sends it to the
1728  *                              stimulation device
1729  *
1730  * PARAMETERS:
1731  *
1732  * RETURNED:    Nothing
1733  *
1734  * REVISION HISTORY: V 1.5      19/09/2003 16:58
1735  *
1736  *
1737  *****************************************************************************/
1738 static void ReadProgramm(FormPtr frmP)
1739 {
1740          Err                 err;
1741          UInt16          hilfsvar, impulsDauer, echo = 0;
1742          UInt32          count;
1743          Char            progString[130], sendString[7], infoStr[12], debMsg[4];
1744          Char*           trigLabelPtr;
1745          MemHandle       data;
1746          FieldPtr        fieldP;
1747
1748
1749
1750          if (CheckROMVersion())
1751                  SetMainStimType();
1752          SysTaskDelay(SysTicksPerSecond()/10);
1753          if (gPalmOS5)
1754                  SysTaskDelay(SysTicksPerSecond() / 10);
1755
1756 // Laden vom Programmspeicher Programm Nr. 1 - 9
1757          if (CtlGetValue(GetObjectPtr(MainProg1PushButton)))
1758                  StrCopy(sendString, "O1 ");
1759          if (CtlGetValue(GetObjectPtr(MainProg2PushButton)))
1760                  StrCopy(sendString, "O2 ");
1761          if (CtlGetValue(GetObjectPtr(MainProg3PushButton)))
1762                  StrCopy(sendString, "O3 ");
```

```
1763        if (gl42Version || gNewVersion)
1764        {
1765                if (CtlGetValue(GetObjectPtr(MainProg4PushButton)))
1766                        StrCopy(sendString, "04 ");
1767                if (CtlGetValue(GetObjectPtr(MainProg5PushButton)))
1768                        StrCopy(sendString, "05 ");
1769                if (CtlGetValue(GetObjectPtr(MainProg6PushButton)))
1770                        StrCopy(sendString, "06 ");
1771                if (CtlGetValue(GetObjectPtr(MainProg7PushButton)))
1772                        StrCopy(sendString, "07 ");
1773                if (CtlGetValue(GetObjectPtr(MainProg8PushButton)))
1774                        StrCopy(sendString, "08 ");
1775                if (CtlGetValue(GetObjectPtr(MainProg9PushButton)))
1776                        StrCopy(sendString, "09 ");
1777        }
1778
1779        count = SendSerialData(sendString);
1780        if (!(gIrComm || gRS232E1))
1781        {
1782                // Wait until all characters are echoed
1783                SrmReceiveWait(gSerialRefNum, StrLen(sendString), 15);
1784                SrmReceiveFlush(gSerialRefNum, 1);
1785        }
1786        SysTaskDelay(SysTicksPerSecond()/5);
1787
1788 // Anzeige für Auslesen des Programms Nr.X aktualisieren
1789        fieldP = GetObjectPtr(MainTextField);
1790        ClearFieldText(MainTextField);
1791        StrCopy(infoStr, "Prog. lesen ");
1792        SetFieldTextFromString(MainTextField, infoStr);
1793
1794 // Auslesen der Programmparameter (128 Bytes + "?" + '\0')
1795        StrCopy(sendString, "?");
1796        count = SendSerialData(sendString);
1797        if (gIrComm || gRS232E1)
1798        {
1799                err = SrmReceiveWait(gSerialRefNum, sizeof(progString) - 2, 120);
1800                count = SrmReceive(gSerialRefNum, progString, sizeof(progString) - 2, 0, &err);
1801        }
1802        else
1803        {
1804                err = SrmReceiveWait(gSerialRefNum, sizeof(progString) - 1, 120);
1805                count = SrmReceive(gSerialRefNum, progString, sizeof(progString) - 1, 0, &err);
1806        }
1807        if (err)
1808        {
1809                StrIToA(debMsg, err);
1810                FrmCustomAlert(DebuggingInfoAlert, "Error", debMsg, "while receiving device settings!");
1811        }
1812        else
1813        {
1814                *(progString + count) = '\0'; // null terminate progString
1815                if (!(gIrComm || gRS232E1))
1816                        echo = 1;
1817
1818                if (!(progString[0x21 + echo] & 0x01))
1819                {
1820                        LstSetSelection(GetObjectPtr(MainAmpl1List), 0);
1821                        trigLabelPtr = LstGetSelectionText(GetObjectPtr(MainAmpl1List),
1822                                                                                                LstGetSelection(GetObjectPtr(MainAmpl1List)
)));
1823                        CtlSetLabel(GetObjectPtr(MainAmpl1PopTrigger), trigLabelPtr);
1824                }
1825                else
1826                {
1827                        for (count = 1; count < LstGetNumberOfItems(GetObjectPtr(MainAmpl1List)); ++count)
1828                        {
1829                                hilfsvar = StrAToI(LstGetSelectionText(GetObjectPtr(MainAmpl1List), count));
1830                                if (((UInt8)progString[0x3F + echo] * 100 / 255) <= hilfsvar)
1831                                {
1832                                        LstSetSelection(GetObjectPtr(MainAmpl1List), count);
1833                                        trigLabelPtr = LstGetSelectionText(GetObjectPtr(MainAmpl1List),
1834                                                                                                                LstGetSelection(GetObjectPtr(MainAm
pl1List)));
1835                                        CtlSetLabel(GetObjectPtr(MainAmpl1PopTrigger), trigLabelPtr);
1836                                        break;
1837                                }
1838                        }
1839                }
1840
1841                if (!(progString[0x21 + echo] & 0x02))
1842                {
1843                        LstSetSelection(GetObjectPtr(MainAmpl2List), 0);
1844                        trigLabelPtr = LstGetSelectionText(GetObjectPtr(MainAmpl2List),
1845                                                                                                LstGetSelection(GetObjectPtr(MainAmpl2List)
)));
1846                        CtlSetLabel(GetObjectPtr(MainAmpl2PopTrigger), trigLabelPtr);
1847                }
1848                else
1849                {
1850                        for (count = 1; count < LstGetNumberOfItems(GetObjectPtr(MainAmpl2List)); ++count)
1851                        {
1852                                hilfsvar = StrAToI(LstGetSelectionText(GetObjectPtr(MainAmpl2List), count));
1853                                if (((UInt8)progString[0x40 + echo] * 100 / 255) <= hilfsvar)
1854                                {
1855                                        LstSetSelection(GetObjectPtr(MainAmpl2List), count);
1856                                        trigLabelPtr = LstGetSelectionText(GetObjectPtr(MainAmpl2List),
1857                                                                                                                LstGetSelection(GetObjectPtr(MainAm
pl2List)));
1858                                        CtlSetLabel(GetObjectPtr(MainAmpl2PopTrigger), trigLabelPtr);
```



```
1859                                        break;
1860                                }
1861                        }
1862                }
1863
1864                for (count = 0; count < LstGetNumberOfItems(GetObjectPtr(MainRamEList)); ++count)
1865                {
1866                        if (gl42Version)
1867                                hilfsvar = StrAToI(LstGetSelectionText(GetObjectPtr(MainRamEList), count)) / 35;
1868                        else
1869                                hilfsvar = StrAToI(LstGetSelectionText(GetObjectPtr(MainRamEList), count)) / 25;
1870                        if ((UInt8)progString[0x34 + echo] == hilfsvar)
1871                        {
1872                                LstSetSelection(GetObjectPtr(MainRamEList), count);
1873                                trigLabelPtr = LstGetSelectionText(GetObjectPtr(MainRamEList),
1874                                                                                                                LstGetSelection(GetObjectPtr(MainRa
mEList)));
1875                                CtlSetLabel(GetObjectPtr(MainRamEPopTrigger), trigLabelPtr);
1876                                break;
1877                        }
1878                }
1879
1880                for (count = 0; count < LstGetNumberOfItems(GetObjectPtr(MainRamAList)); ++count)
1881                {
1882                        if (gl42Version)
1883                                hilfsvar = StrAToI(LstGetSelectionText(GetObjectPtr(MainRamAList), count)) / 35;
1884                        else
1885                                hilfsvar = StrAToI(LstGetSelectionText(GetObjectPtr(MainRamAList), count)) / 25;
1886                        if ((UInt8)progString[0x36 + echo] == hilfsvar)
1887                        {
1888                                LstSetSelection(GetObjectPtr(MainRamAList), count);
1889                                trigLabelPtr = LstGetSelectionText(GetObjectPtr(MainRamAList),
1890                                                                                                                LstGetSelection(GetObjectPtr(MainRa
mAList)));
1891                                CtlSetLabel(GetObjectPtr(MainRamAPopTrigger), trigLabelPtr);
1892                                break;
1893                        }
1894                }
1895
1896                for (count = 0; count < LstGetNumberOfItems(GetObjectPtr(MainImpDList)); ++count)
1897                {
1898                        if (gl42Version && !gNewVersion)
1899                        {
1900                                if ((((UInt8)progString[0x32 + echo]) == hor_ver_step_dauer142[count][0]) &&
1901                                                                        (((UInt8)progString[0x33 + echo]) == hor_ver_step_dauer142[count][1]))
1902                                {
1903                                        LstSetSelection(GetObjectPtr(MainImpDList), count);
1904                                        LstMakeItemVisible(GetObjectPtr(MainImpDList), count);
1905                                        trigLabelPtr = LstGetSelectionText(GetObjectPtr(MainImpDList),
1906                                                                                                                LstGetSelection(GetObjectPtr(MainIm
pDList)));
1907                                        CtlSetLabel(GetObjectPtr(MainImpDPopTrigger), trigLabelPtr);
1908                                        impulsDauer = hor_ver_step_dauer142[count][2];
1909                                        break;
1910                                }
1911                        }
1912                        else
1913                        {
1914                                if ((((UInt8)progString[0x32 + echo]) == hor_ver_step_dauer100[count][0]) &&
1915                                                                        (((UInt8)progString[0x33 + echo]) == hor_ver_step_dauer100[count][1]))
1916                                {
1917                                        LstSetSelection(GetObjectPtr(MainImpDList), count);
1918                                        LstMakeItemVisible(GetObjectPtr(MainImpDList), count);
1919                                        trigLabelPtr = LstGetSelectionText(GetObjectPtr(MainImpDList),
1920                                                                                                                LstGetSelection(GetObjectPtr(MainIm
pDList)));
1921                                        CtlSetLabel(GetObjectPtr(MainImpDPopTrigger), trigLabelPtr);
1922                                        impulsDauer = hor_ver_step_dauer100[count][2];
1923                                        break;
1924                                }
1925                        }
1926                }
1927
1928                for (count = 0; count < LstGetNumberOfItems(GetObjectPtr(MainImpPList)); ++count)
1929                {
1930                        if (gl42Version && !gNewVersion)
1931                                hilfsvar = StrAToI(LstGetSelectionText(GetObjectPtr(MainImpPList), count)) * 7;
1932                        else
1933                                hilfsvar = StrAToI(LstGetSelectionText(GetObjectPtr(MainImpPList), count)) * 10;
1934                        if (!(((UInt8)progString[0x21 + echo] & 0x04) && !gl42Version && !gNewVersion)
1935                                if (hilfsvar < impulsDauer)
1936                                        hilfsvar = 0;
1937                                else
1938                                        hilfsvar = (hilfsvar - impulsDauer) / 2;
1939                        if (((UInt8)progString[0x3D + echo] * 256 + (UInt8)progString[0x3C + echo]) <= hilfsvar)
1940                        {
1941                                LstSetSelection(GetObjectPtr(MainImpPList), count);
1942                                trigLabelPtr = LstGetSelectionText(GetObjectPtr(MainImpPList),
1943                                                                                                                LstGetSelection(GetObjectPtr(MainIm
pPList)));
1944                                CtlSetLabel(GetObjectPtr(MainImpPPopTrigger), trigLabelPtr);
1945                                break;
1946                        }
1947                }
1948
1949
1950                if (gMiFVersion)
1951                {
1952                        for (count = 0; count < LstGetNumberOfItems(GetObjectPtr(MainMiFAnzList)); ++count)
1953                        {
```

```
1954                                    if (count == 0)
1955                                            hilfsvar = 0;
1956                                    else
1957                                            hilfsvar = StrAToI(LstGetSelectionText(GetObjectPtr(MainMiFAnzList), count));
1958                                    if ((UInt8)progString[0x48 + echo] <= hilfsvar)
1959                                    {
1960                                            LstSetSelection(GetObjectPtr(MainMiFAnzList), count);
1961                                            trigLabelPtr = LstGetSelectionText(GetObjectPtr(MainMiFAnzList),
1962                                                                                                    LstGetSelection(GetObjectPtr(MainMi
FAnzList)));
1963                                            CtlSetLabel(GetObjectPtr(MainMiFAnzPopTrigger), trigLabelPtr);
1964                                            break;
1965                                    }
1966                            }
1967                    }
1968
1969                    if (gMiFVersion)
1970                    {
1971                            for (count = 0; count < LstGetNumberOfItems(GetObjectPtr(MainMiFImpDList)); ++count)
1972                            {
1973                                    if (g142Version && !gNewVersion)
1974                                            if ((((UInt8)progString[0x49 + echo]) == hor_ver_step_dauer142[count][0]) &&
1975                                                                            (((UInt8)progString[0x4A + echo]) == hor_ver_step_dauer142[count][1
]))
1976                                            {
1977                                                    LstSetSelection(GetObjectPtr(MainMiFImpDList), count);
1978                                                    LstMakeItemVisible(GetObjectPtr(MainMiFImpDList), count);
1979                                                    trigLabelPtr = LstGetSelectionText(GetObjectPtr(MainMiFImpDList),
1980                                                                                                    LstGetSelection(GetObjectPt
r(MainMiFImpDList)));
1981                                                    CtlSetLabel(GetObjectPtr(MainMiFImpDPopTrigger), trigLabelPtr);
1982                                                    break;
1983                                            }
1984                                    else
1985                                            if ((((UInt8)progString[0x49 + echo]) == hor_ver_step_dauer100[count][0]) &&
1986                                                                            (((UInt8)progString[0x4A + echo]) == hor_ver_step_dauer100[count][1
]))
1987                                            {
1988                                                    LstSetSelection(GetObjectPtr(MainMiFImpDList), count);
1989                                                    LstMakeItemVisible(GetObjectPtr(MainMiFImpDList), count);
1990                                                    trigLabelPtr = LstGetSelectionText(GetObjectPtr(MainMiFImpDList),
1991                                                                                                    LstGetSelection(GetObjectPt
r(MainMiFImpDList)));
1992                                                    CtlSetLabel(GetObjectPtr(MainMiFImpDPopTrigger), trigLabelPtr);
1993                                                    break;
1994                                            }
1995                            }
1996                    }
1997
1998                    for (count = 0; count < LstGetNumberOfItems(GetObjectPtr(MainSchDList)); ++count)
1999                    {
2000                            if (g142Version && !gNewVersion)
2001                                    hilfsvar = StrAToI(LstGetSelectionText(GetObjectPtr(MainSchDList), count)) * 700;
2002                            else
2003                                    hilfsvar = StrAToI(LstGetSelectionText(GetObjectPtr(MainSchDList), count)) * 1000;
2004                            if (((UInt8)progString[0x39 + echo] * 256 + (UInt8)progString[0x38 + echo]) == hilfsvar)
2005                            {
2006                                    LstSetSelection(GetObjectPtr(MainSchDList), count);
2007                                    trigLabelPtr = LstGetSelectionText(GetObjectPtr(MainSchDList),
2008                                                                                            LstGetSelection(GetObjectPtr(MainSc
hDList)));
2009                                    CtlSetLabel(GetObjectPtr(MainSchDPopTrigger), trigLabelPtr);
2010                                    break;
2011                            }
2012                    }
2013
2014                    for (count = 0; count < LstGetNumberOfItems(GetObjectPtr(MainSchPList)); ++count)
2015                    {
2016                            if (g142Version && !gNewVersion)
2017                                    hilfsvar = StrAToI(LstGetSelectionText(GetObjectPtr(MainSchPList), count)) * 700;
2018                            else
2019                                    hilfsvar = StrAToI(LstGetSelectionText(GetObjectPtr(MainSchPList), count)) * 1000;
2020                            if (hilfsvar == 0)
2021                                    hilfsvar = 1;
2022                            if (((UInt8)progString[0x3B + echo] * 256 + (UInt8)progString[0x3A + echo]) == hilfsvar)
2023                            {
2024                                    LstSetSelection(GetObjectPtr(MainSchPList), count);
2025                                    trigLabelPtr = LstGetSelectionText(GetObjectPtr(MainSchPList),
2026                                                                                            LstGetSelection(GetObjectPtr(MainSc
hPList)));
2027                                    CtlSetLabel(GetObjectPtr(MainSchPPopTrigger), trigLabelPtr);
2028                                    break;
2029                            }
2030                    }
2031
2032                    for (count = 0; count < LstGetNumberOfItems(GetObjectPtr(MainProgDList)); ++count)
2033                    {
2034                            hilfsvar = StrAToI(LstGetSelectionText(GetObjectPtr(MainProgDList), count));
2035                            if (((UInt8)((UInt8)progString[0x41 + echo] / 16) * 10 + (UInt8)progString[0x41 + echo] % 16) == hilfsvar)
2036                            {
2037                                    LstSetSelection(GetObjectPtr(MainProgDList), count);
2038                                    trigLabelPtr = LstGetSelectionText(GetObjectPtr(MainProgDList),
2039                                                                                            LstGetSelection(GetObjectPtr(MainPr
ogDList)));
2040                                    CtlSetLabel(GetObjectPtr(MainProgDPopTrigger), trigLabelPtr);
2041                                    break;
2042                            }
2043                    }
2044
2045                    data = FrmGetGadgetData(frmP, FrmGetObjectIndex(frmP, MainImpFGadget));
```



```
2046                    if (data)
2047                    {
2048                            UInt16* dataPtr;
2049
2050                            dataPtr = MemHandleLock(data);
2051                            *dataPtr = (UInt16)progString[0x3E + echo];
2052                            MemHandleUnlock(data);
2053                            PulseShapeDraw(frmP);
2054                    }
2055            }
2056
2057            SetMainStimType();
2058
2059            return;
2060 }
2061
2062 /***********************************************************************
2063  *
2064  * FUNCTION:    GetObjectPtr
2065  *
2066  * DESCRIPTION: This routine returns a pointer to an object in the current
2067  *              form.
2068  *
2069  * PARAMETERS:  formId - id of the form to display
2070  *
2071  * RETURNED:    VoidPtr
2072  *
2073  * REVISION HISTORY: V 1.0 27/04/2000
2074  *
2075  *
2076  ***********************************************************************/
2077 static void * GetObjectPtr(UInt16 objectID)
2078         {
2079         FormPtr frmP;
2080
2081
2082         frmP = FrmGetActiveForm();
2083         return (FrmGetObjectPtr(frmP, FrmGetObjectIndex(frmP, objectID)));
2084 }
2085
2086 /***********************************************************************
2087  *
2088  * FUNCTION:    TestFormDoCommand
2089  *
2090  * DESCRIPTION: This routine performs the menu command specified.
2091  *
2092  * PARAMETERS:  command  - menu item id
2093  *
2094  * RETURNED:    nothing
2095  *
2096  * REVISION HISTORY:    V 1.2 26/05/2000 15:36
2097  *
2098  *
2099  ***********************************************************************/
2100 static Boolean TestFormDoCommand(UInt16 command)
2101 {
2102         Boolean handled = false;
2103
2104
2105         switch (command)
2106                 {
2107                 case InfoAboutDenStim:
2108                         MenuEraseStatus (0);
2109                         AbtShowAbout (appFileCreator);
2110                         handled = true;
2111                         break;
2112
2113                 default:
2114                         break;
2115
2116                 }
2117         return handled;
2118 }
2119
2120 /***********************************************************************
2121  *
2122  * FUNCTION:    TestFormHandleEvent
2123  *
2124  * DESCRIPTION: This routine is the event handler for the
2125  *              "TestForm" of this application.
2126  *
2127  * PARAMETERS:  eventP  - a pointer to an EventType structure
2128  *
2129  * RETURNED:    true if the event has handle and should not be passed
2130  *              to a higher level handler.
2131  *
2132  * REVISION HISTORY:    V 1.2 26/05/2000 17:55
2133  *
2134  *
2135  ***********************************************************************/
2136 static Boolean TestFormHandleEvent(EventPtr eventP)
2137 {
2138     Boolean handled = false;
2139     FormPtr frmP = FrmGetActiveForm();
2140
2141         switch (eventP->eType)
2142                 {
2143                 case menuEvent:
2144                         handled = TestFormDoCommand(eventP->data.menu.itemID);
2145                         break;
```

```
2146
2147                case frmOpenEvent:
2148                        FrmDrawForm(frmP);
2149                        FrmSetFocus(frmP, FrmGetObjectIndex(frmP, TestCommandField));
2150                        handled = true;
2151                        break;
2152
2153                case frmCloseEvent:
2154                        FrmEraseForm(frmP);
2155                        FrmDeleteForm(frmP);
2156                        handled = true;
2157                        break;
2158
2159                case fldChangedEvent:
2160                        UpdateScrollBar(TestTextField, TestTextScrollBar);
2161                        handled = true;
2162                        break;
2163
2164                case ctlSelectEvent:
2165                        if (eventP->data.ctlSelect.controlID == TestSendButton)
2166                                {
2167                                TestSerialCommunication();
2168                                handled = true;
2169                                break;
2170                                }
2171                        if (eventP->data.ctlSelect.controlID == TestBackButton)
2172                                {
2173                                FrmGotoForm(MainForm);
2174                                handled = true;
2175                                break;
2176                                }
2177
2178                case sclRepeatEvent:
2179                        LineScroll(eventP->data.sclRepeat.newValue - eventP->data.sclRepeat.value,
2180                                                TestTextField, TestTextScrollBar,  false);
2181                        break;
2182
2183                case keyDownEvent:
2184                        if (eventP->data.keyDown.chr == pageUpChr)
2185                        {
2186                                PageScroll(winUp, TestTextField, TestTextScrollBar);
2187                                handled = true;
2188                        }
2189                        else
2190                        {
2191                                if (eventP->data.keyDown.chr == pageDownChr)
2192                                {
2193                                        PageScroll(winDown, TestTextField, TestTextScrollBar);
2194                                        handled = true;
2195                                }
2196                        }
2197                        break;
2198
2199                default:
2200                        break;
2201
2202                }
2203        return handled;
2204 }
2205
2206 /***********************************************************************
2207  *
2208  * FUNCTION:    MainFormInit
2209  *
2210  * DESCRIPTION: This routine initializes the MainForm form.
2211  *
2212  * PARAMETERS:  frmP - pointer to the MainForm form.
2213  *
2214  * RETURNED:    nothing
2215  *
2216  * REVISION HISTORY: V 1.2 16/09/2003 10:25
2217  *
2218  *
2219  ***********************************************************************/
2220 static void MainFormInit(FormPtr frmP)
2221 {
2222         Char*           trigLabelPtr;
2223         MemHandle       data = MemHandleNew(sizeof(UInt16));
2224
2225 // Check which type of stimulator is connected to serial port
2226
2227         CheckROMVersion();
2228
2229 // Initialize impulse width list contents and programm push buttons
2230 // according to simulator type
2231         listContentsH = MemHandleNew(256);
2232         MemHandleLock(listContentsH);
2233         SetMainStimType();
2234
2235
2236 // Initialize data for Gadget
2237         if (data)
2238         {
2239                 UInt16* dataPtr;
2240
2241                 dataPtr = MemHandleLock(data);
2242                 *dataPtr = 0;
2243                 MemHandleUnlock(data);
2244                 FrmSetGadgetData(frmP, FrmGetObjectIndex(frmP, MainImpFGadget), data);
2245         }
```



```
2246
2247 // Set default values of stimulation parameters
2248         CtlSetValue(GetObjectPtr(MainProg1PushButton), 1);
2249         CtlSetValue(GetObjectPtr(MainProg3PushButton), 0);
2250
2251         LstSetSelection(GetObjectPtr(MainAmpl1List), 1);
2252         trigLabelPtr = LstGetSelectionText(GetObjectPtr(MainAmpl1List),
2253                                                                 LstGetSelection(GetObjectPtr(MainAmpl1List)
        ));
2254         CtlSetLabel(GetObjectPtr(MainAmpl1PopTrigger), trigLabelPtr);
2255
2256         LstSetSelection(GetObjectPtr(MainAmpl2List), 1);
2257         trigLabelPtr = LstGetSelectionText(GetObjectPtr(MainAmpl2List),
2258                                                                 LstGetSelection(GetObjectPtr(MainAmpl2List)
        ));
2259         CtlSetLabel(GetObjectPtr(MainAmpl2PopTrigger), trigLabelPtr);
2260
2261         LstSetSelection(GetObjectPtr(MainImpDList), 15);
2262         LstMakeItemVisible(GetObjectPtr(MainImpDList), 15);
2263         trigLabelPtr = LstGetSelectionText(GetObjectPtr(MainImpDList),
2264                                                                 LstGetSelection(GetObjectPtr(MainImpDList))
        );
2265         CtlSetLabel(GetObjectPtr(MainImpDPopTrigger), trigLabelPtr);
2266
2267         LstSetSelection(GetObjectPtr(MainImpPList), 10);
2268         trigLabelPtr = LstGetSelectionText(GetObjectPtr(MainImpPList),
2269                                                                 LstGetSelection(GetObjectPtr(MainImpPList))
        );
2270         CtlSetLabel(GetObjectPtr(MainImpPPopTrigger), trigLabelPtr);
2271
2272         LstSetSelection(GetObjectPtr(MainMiFAnzList), 0);
2273         trigLabelPtr = LstGetSelectionText(GetObjectPtr(MainMiFAnzList),
2274                                                                 LstGetSelection(GetObjectPtr(MainMiFAnzList
        ));
2275         CtlSetLabel(GetObjectPtr(MainMiFAnzPopTrigger), trigLabelPtr);
2276
2277         LstSetSelection(GetObjectPtr(MainMiFImpDList), 25);
2278         LstMakeItemVisible(GetObjectPtr(MainMiFImpDList), 15);
2279         trigLabelPtr = LstGetSelectionText(GetObjectPtr(MainMiFImpDList),
2280                                                                 LstGetSelection(GetObjectPtr(MainMiFImpDLis
    t)));
2281         CtlSetLabel(GetObjectPtr(MainMiFImpDPopTrigger), trigLabelPtr);
2282
2283         LstSetSelection(GetObjectPtr(MainSchDList), 2);
2284         trigLabelPtr = LstGetSelectionText(GetObjectPtr(MainSchDList),
2285                                                                 LstGetSelection(GetObjectPtr(MainSchDList))
        );
2286         CtlSetLabel(GetObjectPtr(MainSchDPopTrigger), trigLabelPtr);
2287
2288         LstSetSelection(GetObjectPtr(MainSchPList), 5);
2289         trigLabelPtr = LstGetSelectionText(GetObjectPtr(MainSchPList),
2290                                                                 LstGetSelection(GetObjectPtr(MainSchPList))
        );
2291         CtlSetLabel(GetObjectPtr(MainSchPPopTrigger), trigLabelPtr);
2292
2293         LstSetSelection(GetObjectPtr(MainRamEList), 3);
2294         trigLabelPtr = LstGetSelectionText(GetObjectPtr(MainRamEList),
2295                                                                 LstGetSelection(GetObjectPtr(MainRamEList))
        );
2296         CtlSetLabel(GetObjectPtr(MainRamEPopTrigger), trigLabelPtr);
2297
2298         LstSetSelection(GetObjectPtr(MainRamAList), 0);
2299         trigLabelPtr = LstGetSelectionText(GetObjectPtr(MainRamAList),
2300                                                                 LstGetSelection(GetObjectPtr(MainRamAList))
        );
2301         CtlSetLabel(GetObjectPtr(MainRamAPopTrigger), trigLabelPtr);
2302
2303         LstSetSelection(GetObjectPtr(MainProgDList), 2);
2304         trigLabelPtr = LstGetSelectionText(GetObjectPtr(MainProgDList),
2305                                                                 LstGetSelection(GetObjectPtr(MainProgDList)
        ));
2306         CtlSetLabel(GetObjectPtr(MainProgDPopTrigger), trigLabelPtr);
2307
2308         return;
2309 }
2310
2311
2312 /***********************************************************************
2313  *
2314  * FUNCTION:    MainFormDoCommand
2315  *
2316  * DESCRIPTION: This routine performs the menu command specified.
2317  *
2318  * PARAMETERS:  command  - menu item id
2319  *
2320  * RETURNED:    nothing
2321  *
2322  * REVISION HISTORY: V 1.3 03/11/2000
2323  *
2324  *
2325  ***********************************************************************/
2326 static Boolean MainFormDoCommand(UInt16 command)
2327 {
2328         Err             error;
2329         Boolean handled = false;
2330         Char    sendString[7];
2331 // Code for Debugging Start
2332 //      UInt32          value;
2333 //      FieldPtr        fieldP;
2334 //      Char            debugStr[7];
```

```
2335 // Code for Debugging End
2336
2337
2338         switch (command)
2339                 {
2340                 case InfoAboutDenStim:
2341                         MenuEraseStatus (0);
2342                         AbtShowAbout (appFileCreator);
2343                         handled = true;
2344                         break;
2345
2346                 case TestSerialCom:
2347                         MenuEraseStatus (0);
2348                         FrmGotoForm(TestForm);
2349                         handled = true;
2350                         break;
2351
2352                 case TestStartStop:
2353                         MenuEraseStatus (0);
2354                         if (gMiFVersion)
2355                         {
2356                                 StrCopy(sendString, "/4 ");
2357                                 SendSerialData(sendString);
2358                         }
2359                         handled = true;
2360                         break;
2361
2362                 case OptionsIrComm8E1:
2363                         MenuEraseStatus (0);
2364                         gIrComm = true;
2365                         gRS232E1 = false;
2366                         error = DeactivateSerialPort();
2367                         if (error)
2368                                 FrmCustomAlert(DebuggingInfoAlert, "Serial port deactivation failed!", "", "");
2369                         error = ActivateSerialPort(Setting8E1);
2370                         if (error)
2371                                 FrmCustomAlert(DebuggingInfoAlert, "Serial port activation failed!", "", "");
2372                         handled = true;
2373                         break;
2374
2375                 case OptionsIrComm8N2:
2376                         MenuEraseStatus (0);
2377                         gIrComm = true;
2378                         gRS232E1 = false;
2379                         error = DeactivateSerialPort();
2380                         if (error)
2381                                 FrmCustomAlert(DebuggingInfoAlert, "Serial port deactivation failed!", "", "");
2382                         error = ActivateSerialPort(Setting8N2);
2383                         if (error)
2384                                 FrmCustomAlert(DebuggingInfoAlert, "Serial port activation failed!", "", "");
2385                         handled = true;
2386                         break;
2387
2388                 case OptionsRS2328N2:
2389                         MenuEraseStatus (0);
2390                         gIrComm = false;
2391                         gRS2328E1 = false;
2392                         error = DeactivateSerialPort();
2393                         if (error)
2394                                 FrmCustomAlert(DebuggingInfoAlert, "Serial port deactivation failed!", "", "");
2395                         error = ActivateSerialPort(Setting8N2);
2396                         if (error)
2397                                 FrmCustomAlert(DebuggingInfoAlert, "Serial port activation failed!", "", "");
2398                         handled = true;
2399                         break;
2400
2401                 case OptionsRS2328E1:
2402                         MenuEraseStatus (0);
2403                         gIrComm = false;
2404                         gRS2328E1 = true;
2405                         error = DeactivateSerialPort();
2406                         if (error)
2407                                 FrmCustomAlert(DebuggingInfoAlert, "Serial port deactivation failed!", "", "");
2408                         error = ActivateSerialPort(Setting8E1);
2409                         if (error)
2410                                 FrmCustomAlert(DebuggingInfoAlert, "Serial port activation failed!", "", "");
2411                         handled = true;
2412                         break;
2413
2414                 case OptionsBaudRate9600:
2415                         MenuEraseStatus (0);
2416                         gBaudRate = 9600;
2417                         error = DeactivateSerialPort();
2418                         if (error)
2419                                 FrmCustomAlert(DebuggingInfoAlert, "Serial port deactivation failed!", "", "");
2420                         error = ActivateSerialPort(Setting8E1);
2421                         if (error)
2422                                 FrmCustomAlert(DebuggingInfoAlert, "Serial port activation failed!", "", "");
2423                         handled = true;
2424                         break;
2425
2426                 case OptionsBaudRate1200:
2427                         MenuEraseStatus (0);
2428                         gBaudRate = 1200;
2429                         error = DeactivateSerialPort();
2430                         if (error)
2431                                 FrmCustomAlert(DebuggingInfoAlert, "Serial port deactivation failed!", "", "");
2432                         error = ActivateSerialPort(Setting8E1);
2433                         if (error)
2434                                 FrmCustomAlert(DebuggingInfoAlert, "Serial port activation failed!", "", "");
```



```
2435                         handled = true;
2436                         break;
2437
2438                 default:
2439                         handled = false;
2440                         break;
2441
2442                 }
2443         return handled;
2444 }
2445
2446 /***********************************************************************
2447  *
2448  * FUNCTION:    MainFormDeInit
2449  *
2450  * DESCRIPTION: This routine releases memory used by the MainForm form.
2451  *
2452  * PARAMETERS:  frmP - pointer to the MainForm form.
2453  *
2454  * RETURNED:    nothing
2455  *
2456  * REVISION HISTORY:
2457  *
2458  *
2459  ***********************************************************************/
2460 static void MainFormDeInit(FormPtr frmP)
2461 {
2462         MemHandle data;
2463
2464         data = FrmGetGadgetData(frmP, FrmGetObjectIndex(frmP, MainImpFGadget));
2465         if (data)
2466                 MemHandleFree(data);
2467
2468         if (stringArrayPtrH)
2469         {
2470                 MemHandleFree(stringArrayPtrH);
2471                 stringArrayPtrH = NULL;
2472         }
2473
2474         if (listContentsH)
2475         {
2476                 MemHandleFree(listContentsH);
2477                 listContentsH = NULL;
2478         }
2479         FrmEraseForm(frmP);
2480         FrmDeleteForm(frmP);
2481
2482 }
2483
2484 /***********************************************************************
2485  *
2486  * FUNCTION:    MainFormHandleEvent
2487  *
2488  * DESCRIPTION: This routine is the event handler for the
2489  *              "MainForm" of this application.
2490  *
2491  * PARAMETERS:  eventP  - a pointer to an EventType structure
2492  *
2493  * RETURNED:    true if the event has handle and should not be passed
2494  *              to a higher level handler.
2495  *
2496  * REVISION HISTORY: V 1.1 26/05/2000
2497  *
2498  *
2499  ***********************************************************************/
2500 static Boolean MainFormHandleEvent(EventPtr eventP)
2501 {
2502     Boolean handled = false;
2503     FormPtr frmP = FrmGetActiveForm();
2504
2505
2506         switch (eventP->eType)
2507                 {
2508                 case menuEvent:
2509                         handled = MainFormDoCommand(eventP->data.menu.itemID);
2510                         PulseShapeDraw(frmP);
2511                         break;
2512
2513                 case frmOpenEvent:
2514                         FrmDrawForm (frmP);
2515                         MainFormInit(frmP);
2516                         PulseShapeDraw(frmP);
2517                         handled = true;
2518                         break;
2519
2520                 case frmCloseEvent:
2521                         MainFormDeInit (frmP);
2522                         handled = true;
2523                         break;
2524
2525                 case penDownEvent:
2526                         {
2527                                 RectangleType bounds;
2528
2529                                 FrmGetObjectBounds(frmP, FrmGetObjectIndex(frmP, MainImpFGadget), &bounds);
2530                                 if (RctPtInRectangle(eventP->screenX, eventP->screenY, &bounds))
2531                                 {
2532                                         MemHandle       data;
2533                                         UInt16*         dataPtr;
2534
```

```
2535                                    data = FrmGetGadgetData(frmP, FrmGetObjectIndex(frmP, MainImpFGadget));
2536                                    if (data)
2537                                    {
2538                                            dataPtr = MemHandleLock(data);
2539                                            if (*dataPtr < 3)
2540                                                    ++ *dataPtr;
2541                                            else
2542                                                    *dataPtr = 0;
2543                                            MemHandleUnlock(data);
2544                                    }
2545                                    PulseShapeDraw(frmP);
2546                                    handled = true;
2547                            }
2548
2549                            FrmGetObjectBounds(frmP, FrmGetObjectIndex(frmP, MainTextField), &bounds);
2550                            if (RctPtInRectangle(eventP->screenX, eventP->screenY, &bounds))
2551                            {
2552                                    if (gNewVersion)
2553                                    {
2554                                            gNewVersion = false;
2555                                            gl42Version = true;
2556                                    }
2557                                    else
2558                                            if (gl42Version)
2559                                                    gl42Version = false;
2560                                            else
2561                                                    gNewVersion = true;
2562                                    SetMainStimType();
2563                                    handled = true;
2564                            }
2565
2566                            FrmGetObjectBounds(frmP, FrmGetObjectIndex(frmP, MainMiFAnzLabel), &bounds);
2567                            if (RctPtInRectangle(eventP->screenX, eventP->screenY, &bounds))
2568                            {
2569                                    gMiFVersion = !gMiFVersion;
2570                                    SetMainStimType();
2571                                    handled = true;
2572                            }
2573                    }
2574                    break;
2575
2576            case ctlSelectEvent:
2577                    if (eventP->data.ctlSelect.controlID == MainReadProgButton)
2578                    {
2579                            // action
2580                            ReadProgramm(frmP);
2581                            handled = true;
2582                            break;
2583                    }
2584
2585                    if      (eventP->data.ctlSelect.controlID == MainSetProgButton)
2586                    {
2587                            // action
2588                            SetProgramm(frmP);
2589                            VerifyProgramm(frmP);
2590                            handled = true;
2591                            break;
2592                    }
2593
2594            default:
2595                    break;
2596
2597            }
2598
2599        return handled;
2600 }
2601
2602
2603 /***********************************************************************
2604  *
2605  * FUNCTION:    AppHandleEvent
2606  *
2607  * DESCRIPTION: This routine loads form resources and set the event
2608  *              handler for the form loaded.
2609  *
2610  * PARAMETERS:  event  - a pointer to an EventType structure
2611  *
2612  * RETURNED:    true if the event has handle and should not be passed
2613  *              to a higher level handler.
2614  *
2615  * REVISION HISTORY:
2616  *
2617  *
2618  ***********************************************************************/
2619 static Boolean AppHandleEvent( EventPtr eventP)
2620 {
2621        UInt16 formId;
2622        FormPtr frmP;
2623
2624
2625        if (eventP->eType == frmLoadEvent)
2626                {
2627                // Load the form resource.
2628                formId = eventP->data.frmLoad.formID;
2629                frmP = FrmInitForm(formId);
2630                FrmSetActiveForm(frmP);
2631
2632                // Set the event handler for the form.  The handler of the currently
2633                // active form is called by FrmHandleEvent each time is receives an
2634                // event.
```



```
2635                switch (formId)
2636                        {
2637                        case MainForm:
2638                                FrmSetEventHandler(frmP, MainFormHandleEvent);
2639                                break;
2640
2641                        case TestForm:
2642                                FrmSetEventHandler(frmP, TestFormHandleEvent);
2643                                break;
2644
2645                        default:
2646                                ErrFatalDisplay("Invalid Form Load Event");
2647                                break;
2648
2649                        }
2650                return true;
2651                }
2652
2653        return false;
2654 }
2655
2656
2657 /***********************************************************************
2658  *
2659  * FUNCTION:    AppEventLoop
2660  *
2661  * DESCRIPTION: This routine is the event loop for the application.
2662  *
2663  * PARAMETERS:  nothing
2664  *
2665  * RETURNED:    nothing
2666  *
2667  * REVISION HISTORY:
2668  *
2669  *
2670  ***********************************************************************/
2671 static void AppEventLoop(void)
2672 {
2673        UInt16 error;
2674        EventType event;
2675
2676
2677        do {
2678                EvtGetEvent(&event, evtWaitForever);
2679
2680                if (! SysHandleEvent(&event))
2681                        if (! MenuHandleEvent(0, &event, &error))
2682                                if (! AppHandleEvent(&event))
2683                                        FrmDispatchEvent(&event);
2684
2685
2686        } while (event.eType != appStopEvent);
2687 }
2688
2689
2690 /***********************************************************************
2691  *
2692  * FUNCTION:     AppStart
2693  *
2694  * DESCRIPTION:  Get the current application's preferences.
2695  *
2696  * PARAMETERS:   nothing
2697  *
2698  * RETURNED:     Err value 0 if nothing went wrong
2699  *
2700  * REVISION HISTORY: V 1.0 02/05/2000
2701  *
2702  *
2703  ***********************************************************************/
2704 static Err AppStart(void)
2705 {
2706     DenStimPreferenceType prefs;
2707     UInt16 prefsSize;
2708
2709
2710        // Read the saved preferences / saved-state information.
2711        prefsSize = sizeof(DenStimPreferenceType);
2712        if (PrefGetAppPreferences(appFileCreator, appPrefID, &prefs, &prefsSize, true) !=
2713                noPreferenceFound)
2714                {
2715                }
2716
2717    return 0;
2718 }
2719
2720
2721 /***********************************************************************
2722  *
2723  * FUNCTION:    AppStop
2724  *
2725  * DESCRIPTION: Save the current state of the application.
2726  *
2727  * PARAMETERS:  nothing
2728  *
2729  * RETURNED:    nothing
2730  *
2731  * REVISION HISTORY: V 1.0 02/05/2000
2732  *
2733  *
2734  ***********************************************************************/
```

```
2735 static void AppStop(void)
2736 {
2737         DenStimPreferenceType prefs;
2738         FormPtr frmP = FrmGetActiveForm();
2739         UInt16  frmId = FrmGetActiveFormID();
2740
2741         switch (frmId)
2742         {
2743                 case MainForm:
2744                         MainFormDeInit (frmP);
2745                         break;
2746
2747                 case TestForm:
2748                         FrmEraseForm(frmP);
2749                         FrmDeleteForm(frmP);
2750                         break;
2751
2752                 default:
2753                         ErrFatalDisplay("Invalid Form ID");
2754                         break;
2755         }
2756
2757
2758         // Write the saved preferences / saved-state information.  This data
2759         // will be backed up during a HotSync.
2760         PrefSetAppPreferences (appFileCreator, appPrefID, appPrefVersionNum,
2761                 &prefs, sizeof (prefs), true);
2762 }
2763
2764
2765 /*!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
2766  * All Code from here to end of file should not use global variables !!!
2767  *!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!*/
2768
2769
2770 /***********************************************************************
2771  *
2772  * FUNCTION:    RomVersionCompatible
2773  *
2774  * DESCRIPTION: This routine checks that a ROM version is meet your
2775  *              minimum requirement.
2776  *
2777  * PARAMETERS:  requiredVersion - minimum rom version required
2778  *                                (see sysFtrNumROMVersion in SystemMgr.h
2779  *                                for format)
2780  *              launchFlags     - flags that indicate if the application
2781  *                                UI is initialized.
2782  *
2783  * RETURNED:    error code or zero if rom is compatible
2784  *
2785  * REVISION HISTORY:
2786  *
2787  ***********************************************************************/
2788 static Err RomVersionCompatible(UInt32 requiredVersion, UInt16 launchFlags)
2789 {
2790         UInt32 romVersion;
2791
2792         // See if we're on in minimum required version of the ROM or later.
2793         FtrGet(sysFtrCreator, sysFtrNumROMVersion, &romVersion);
2794         if (romVersion < requiredVersion)
2795         {
2796                 if ((launchFlags & (sysAppLaunchFlagNewGlobals | sysAppLaunchFlagUIApp)) ==
2797                         (sysAppLaunchFlagNewGlobals | sysAppLaunchFlagUIApp))
2798                         {
2799                         FrmAlert (RomIncompatibleAlert);
2800
2801                         // Pilot 1.0 will continuously relaunch this app unless we switch to
2802                         // another safe one.
2803                         if (romVersion < ourMinVersion)
2804                                 AppLaunchWithCommand(sysFileCDefaultApp, sysAppLaunchCmdNormalLaunch, NULL);
2805                         }
2806
2807                 return (sysErrRomIncompatible);
2808         }
2809         return (0);
2810 }
2811
2812 /***********************************************************************
2813  *
2814  * FUNCTION:    DenStimMain
2815  *
2816  * DESCRIPTION: This is the main entry point for the application.
2817  * PARAMETERS:  cmd - word value specifying the launch code.
2818  *              cmdPB - pointer to a structure that is associated with the launch code.
2819  *              launchFlags -  word value providing extra information about the launch.
2820  *
2821  * RETURNED:    Result of launch
2822  *
2823  * REVISION HISTORY: V 1.1 03/11/2000
2824  *
2825  *
2826  ***********************************************************************/
2827 static UInt32 DenStimMain(UInt16 cmd, MemPtr cmdPBP, UInt16 launchFlags)
2828 {
2829         Err error;
2830
2831         error = RomVersionCompatible (ourMinVersion, launchFlags);
2832         if (error) return (error);
2833
2834
```



```
2835         switch (cmd)
2836                 {
2837                 case sysAppLaunchCmdNormalLaunch:
2838                         error = AppStart();
2839                         if (error)
2840                                 return error;
2841
2842                         error = ActivateSerialPort(Setting8E1);
2843
2844                         FrmGotoForm(MainForm);
2845                         AppEventLoop();
2846
2847                         error = DeactivateSerialPort();
2848
2849                         AppStop();
2850                         break;
2851
2852                 default:
2853                         break;
2854
2855                 }
2856
2857         return 0;
2858 }
2859
2860
2861 /***********************************************************************
2862  *
2863  * FUNCTION:    PilotMain
2864  *
2865  * DESCRIPTION: This is the main entry point for the application.
2866  *
2867  * PARAMETERS:  cmd - word value specifying the launch code.
2868  *              cmdPB - pointer to a structure that is associated with the launch code.
2869  *              launchFlags -  word value providing extra information about the launch.
2870  * RETURNED:    Result of launch
2871  *
2872  * REVISION HISTORY: V 1.0 02/05/2000
2873  *
2874  *
2875  ***********************************************************************/
2876 UInt32 PilotMain(UInt16 cmd, MemPtr cmdPBP, UInt16 launchFlags)
2877 {
2878     return DenStimMain(cmd, cmdPBP, launchFlags);
2879 }
2880
```

B Matlab scripts for CT analysis and MFCV & MFRP


```
function varargout = ctarea(varargin)

% CTAREA M-file for ctarea.fig by Christian Hofer
% Analysis of the CT scans showing the cross section of thigh calculating
% the area and the density (mean and histogram of distribution)of the
% selected ROIs
%
%      CTAREA, by itself, creates a new CTAREA or raises the existing
%      singleton*.
%
%      H = CTAREA returns the handle to a new CTAREA or the handle to
%      the existing singleton*.
%
%      CTAREA('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in CTAREA.M with the given input arguments.
%
%      CTAREA('Property','Value',...) creates a new CTAREA or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before ctarea_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to ctarea_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help ctarea

% Last Modified by GUIDE v2.5 24-Feb-2006 10:40:50

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @ctarea_OpeningFcn, ...
                   'gui_OutputFcn',  @ctarea_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin & isstr(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before ctarea is made visible.
function ctarea_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to ctarea (see VARARGIN)

% Choose default command line output for ctarea
handles.output = hObject;
```



```

% Update handles structure
guidata(hObject, handles);

set(hObject, 'Units', 'pixel');
set(hObject, 'Position', [ 60 60 1024-60 768-60]);
handles.hinit(1:120)=0;
handles.HU=-195:10:995;
handles.hleftquad=handles.hinit;
handles.hrightquad=handles.hinit;
handles.hlefttham=handles.hinit;
handles.hrightham=handles.hinit;
handles.fig=figure;
close(handles.fig);

guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = ctarea_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
axes(handles.axes1);
cla;
axes(handles.axes2);
cla;
axes(handles.axes3);
cla;
axes(handles.axes4);
cla;
axes(handles.axes5);
cla;
axes(handles.axes6);
cla;
axes(handles.axes7);
cla;
axes(handles.axes8);
cla;
[s1, s2] = uigetfile ('*.*', 'Open File');
cd (s2);
handles.info=dicominfo ([s2 s1]);
CTScan=dicomread(handles.info);
handles.CTScan = double(CTScan)*handles.info.RescaleSlope + handles.info.Rescale
Intercept;
handles.hinit(1:120)=0;
axes(handles.axes1);
```

```
imshow(handles.CTScan,[-125 225]);
try
    handles.infotext(1) = {[handles.info.PatientsName.FamilyName, ' ']};
end
try
    handles.infotext(1) = {[handles.info.PatientsName.FamilyName, ' ',handles.info.PatientsName.GivenName(1),'.']};
end
try
    handles.infotext(1) = {[handles.info.PatientName.FamilyName, ' ']};
end
try
    handles.infotext(1) = {[handles.info.PatientName.FamilyName, ' ',handles.info.PatientName.GivenName(1),'.']};
end
handles.infotext(2) = {['date of examination: ', handles.info.AcquisitionDate(1:4),'.',handles.info.AcquisitionDate(5:6),'.',handles.info.AcquisitionDate(7:8)]};
handles.infotext(4)={[]};
handles.infotext(5)={[]};
handles.infotext(7)={[]};
handles.infotext(8)={[]};
handles.infotext(10)={[]};
handles.infotext(11)={[]};
set(findobj(handles.text2),'String',handles.infotext);

guidata(hObject, handles);


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
axes(handles.axes2);
cla;
axes(handles.axes4);
cla;
axes(handles.axes5);
cla;
axes(handles.axes6);
cla;
axes(handles.axes4);
imshow(handles.CTScan,[-350 100]);
[x,y]=ginput(2);
handles.SelectedAreaRight=handles.CTScan(y(1):y(2),x(1):x(2));
axes(handles.axes4);
%imshow(handles.SelectedAreaRight,[-350 100]);
imshow(handles.SelectedAreaRight,[-125 225]);
axes(handles.axes2);
imshow(handles.SelectedAreaRight,[-125 225]);
handles.SelectedAreaRight(find(handles.SelectedAreaRight < -200)) = NaN;
handles.SelectedAreaRight(find(handles.SelectedAreaRight > 1000)) = 1000;
[pixelnumber,n]=size(find(~isnan(handles.SelectedAreaRight)));
thigharea = pixelnumber*handles.info.PixelSpacing(1)*handles.info.PixelSpacing(2)/100;
[pixelnumber,n]=size(find(handles.SelectedAreaRight < 200));
hrightmean = mean(handles.SelectedAreaRight(find(handles.SelectedAreaRight < 200)));
hrightstd = std(handles.SelectedAreaRight(find(handles.SelectedAreaRight < 200)));
```



```
handles.infotext(4)={[sprintf('right thigh cross sectional area [cm2] = %6.2f',thigharea) '   '...
                                    sprintf('mean density [HU] = %6.2f',hrightmean) '   '...
                                    sprintf('sd = ±%6.2f',hrightstd)]};
set(findobj(handles.text2),'String',handles.infotext);

guidata(hObject, handles);

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
axes(handles.axes3);
cla;
axes(handles.axes4);
cla;
axes(handles.axes7);
cla;
axes(handles.axes8);
cla;
axes(handles.axes4);
imshow(handles.CTScan,[-350 100]);
[x,y]=ginput(2);
handles.SelectedAreaLeft=handles.CTScan(y(1):y(2),x(1):x(2));
axes(handles.axes4);
%imshow(handles.SelectedAreaLeft,[-350 100]);
imshow(handles.SelectedAreaLeft,[-125 225]);
axes(handles.axes3);
imshow(handles.SelectedAreaLeft,[-125 225]);
handles.SelectedAreaLeft(find(handles.SelectedAreaLeft < -200)) = NaN;
handles.SelectedAreaLeft(find(handles.SelectedAreaLeft > 1000)) = 1000;
[pixelnumber,n]=size(find(~isnan(handles.SelectedAreaLeft)));
thigharea = pixelnumber*handles.info.PixelSpacing(1)*handles.info.PixelSpacing(2)/100;
[pixelnumber,n]=size(find(handles.SelectedAreaLeft < 200));
hleftmean = mean(handles.SelectedAreaLeft(find(handles.SelectedAreaLeft < 200)));
hleftstd = std(handles.SelectedAreaLeft(find(handles.SelectedAreaLeft < 200)));
handles.infotext(5)={[sprintf('left   thigh cross sectional area [cm2] = %6.2f',thigharea) '   '...
                                    sprintf('mean density [HU] = %6.2f',hleftmean) '   '...
                                    sprintf('sd = ±%6.2f',hleftstd)]};
set(findobj(handles.text2),'String',handles.infotext);

guidata(hObject, handles);


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
axes(handles.axes4);
imshow(handles.SelectedAreaRight,[-350 100]);
%imshow(handles.SelectedAreaRight,[-125 225]);
[yf,xf]=size(handles.SelectedAreaRight);
[x,y]=ginput(2);
if x(1)<1
```

```
231     x(1)=1;
232 end
233 if y(1)<1
234     y(1)=1;
235 end
236 if x(2)>xf
237     x(2)=xf;
238 end
239 if y(2)>yf
240     y(2)=yf;
241 end
242 handles.SelectedAreaRight(y(1):y(2),x(1):x(2))=-200;
243 handles.SelectedAreaRight(find(handles.SelectedAreaRight < -199)) = NaN;
244
245 axes(handles.axes4);
246 imshow(handles.SelectedAreaRight,[-350 100]);
247 %imshow(handles.SelectedAreaRight,[-125 225]);
248 axes(handles.axes2);
249 imshow(handles.SelectedAreaRight,[-125 225]);
250
251 [pixelnumber,n]=size(find(~isnan(handles.SelectedAreaRight)));
252 thigharea = pixelnumber*handles.info.PixelSpacing(1)*handles.info.PixelSpacing(2
    )/100;
253 [pixelnumber,n]=size(find(handles.SelectedAreaRight < 200));
254 hrightmean = mean(handles.SelectedAreaRight(find(handles.SelectedAreaRight < 200
    )));
255 hrightstd = std(handles.SelectedAreaRight(find(handles.SelectedAreaRight < 200))
    );
256 handles.infotext(4)={[sprintf('right thigh cross sectional area [cm2] = %6.2f',t
    higharea) '   '...
257                                     sprintf('mean density [HU] = %6.2f',hr
    ightmean) '   '...
258                                     sprintf('sd = ±%6.2f',hrightstd)]};
259 set(findobj(handles.text2),'String',handles.infotext);
260
261 guidata(hObject, handles);
262
263 % --- Executes on button press in pushbutton5.
264 function pushbutton5_Callback(hObject, eventdata, handles)
265 % hObject    handle to pushbutton5 (see GCBO)
266 % eventdata  reserved - to be defined in a future version of MATLAB
267 % handles    structure with handles and user data (see GUIDATA)
268 axes(handles.axes4);
269 imshow(handles.SelectedAreaLeft,[-350 100]);
270 %imshow(handles.SelectedAreaLeft,[-125 225]);
271 [yf,xf]=size(handles.SelectedAreaLeft);
272 [x,y]=ginput(2);
273 if x(1)<1
274     x(1)=1;
275 end
276 if y(1)<1
277     y(1)=1;
278 end
279 if x(2)>xf
280     x(2)=xf;
281 end
282 if y(2)>yf
283     y(2)=yf;
284 end
285 handles.SelectedAreaLeft(y(1):y(2),x(1):x(2))=-200;
286 handles.SelectedAreaLeft(find(handles.SelectedAreaLeft < -199)) = NaN;
```



```
287
288 axes(handles.axes4);
289 imshow(handles.SelectedAreaLeft,[-350 100]);
290 %imshow(handles.SelectedAreaLeft,[-125 225]);
291 axes(handles.axes3);
292 imshow(handles.SelectedAreaLeft,[-125 225]);
293
294 [pixelnumber,n]=size(find(~isnan(handles.SelectedAreaLeft)));
295 thigharea = pixelnumber*handles.info.PixelSpacing(1)*handles.info.PixelSpacing(2
    )/100;
296 [pixelnumber,n]=size(find(handles.SelectedAreaLeft < 200));
297 hleftmean = mean(handles.SelectedAreaLeft(find(handles.SelectedAreaLeft < 200)))
    ;
298 hleftstd = std(handles.SelectedAreaLeft(find(handles.SelectedAreaLeft < 200)));
299 handles.infotext(5)={[sprintf('left   thigh cross sectional area [cm2] = %6.2f',
    thigharea) '   '...
300                                     sprintf('mean density [HU] = %6.2f',
    hleftmean) '   '...
301                                     sprintf('sd = ±%6.2f',hleftstd)]};
302 set(findobj(handles.text2),'String',handles.infotext);
303
304 guidata(hObject, handles);
305
306 % --- Executes on button press in pushbutton6.
307 function pushbutton6_Callback(hObject, eventdata, handles)
308 % hObject    handle to pushbutton6 (see GCBO)
309 % eventdata  reserved - to be defined in a future version of MATLAB
310 % handles    structure with handles and user data (see GUIDATA)
311 axes(handles.axes4);
312 cla;
313 axes(handles.axes5);
314 cla;
315 handles.QuadricepsMaskRight=zeros(size(handles.SelectedAreaRight));
316
317 axes(handles.axes4);
318 imshow(handles.SelectedAreaRight,[-125 225]);
319 [x,y]=ginput(2);
320 imshow(handles.SelectedAreaRight(y(1):y(2),x(1):x(2)),[-125 225]);
321 handles.QuadricepsMaskRight(y(1):y(2),x(1):x(2))=roipoly;
322 axes(handles.axes4);
323 imshow((handles.SelectedAreaRight + handles.QuadricepsMaskRight*120),[-125 225])
    ;
324 axes(handles.axes5);
325 imshow((handles.SelectedAreaRight + handles.QuadricepsMaskRight*120),[-125 225])
    ;
326
327 [pixelnumber, n]=size(find(handles.QuadricepsMaskRight == 1));
328 quadricepsarea = pixelnumber*handles.info.PixelSpacing(1)*handles.info.PixelSpac
    ing(2)/100;
329 meanhu = mean(handles.SelectedAreaRight(find(handles.QuadricepsMaskRight == 1)))
    ;
330 stdhu = std(handles.SelectedAreaRight(find(handles.QuadricepsMaskRight == 1)));
331 handles.infotext(7)={[sprintf('right quadriceps cross sectional area [cm2] = %6.
    2f',quadricepsarea) '   '...
332                                     sprintf('mean density [HU] = %6.2f',
    meanhu) '   '...
333                                     sprintf('sd = ±%6.2f',stdhu)]};
334 set(findobj(handles.text2),'String',handles.infotext);
335 hright = hist(handles.SelectedAreaRight(find(handles.QuadricepsMaskRight == 1)),
    handles.HU);
336 handles.hrightquad = hright/pixelnumber*100;
```

```
337
338 figure(handles.fig);
339 subplot(2,1,1);
340 bar(handles.HU,[handles.hrightquad' handles.hinit'],0.8,'r');
341 hold on;
342 bar(handles.HU,[handles.hinit' handles.hleftquad'],0.8,'b');
343 hold off;
344 axis([-150 150 0 40]);
345 xlabel('Hounsfield Units');
346 ylabel('cross sectional area [%]');
347
348 guidata(hObject, handles);
349
350 % --- Executes on button press in pushbutton7.
351 function pushbutton7_Callback(hObject, eventdata, handles)
352 % hObject    handle to pushbutton7 (see GCBO)
353 % eventdata  reserved - to be defined in a future version of MATLAB
354 % handles    structure with handles and user data (see GUIDATA)
355 axes(handles.axes4);
356 cla;
357 axes(handles.axes7);
358 cla;
359 handles.QuadricepsMaskLeft=zeros(size(handles.SelectedAreaLeft));
360
361 axes(handles.axes4);
362 imshow(handles.SelectedAreaLeft,[-125 225]);
363 [x,y]=ginput(2);
364 imshow(handles.SelectedAreaLeft(y(1):y(2),x(1):x(2)),[-125 225]);
365 handles.QuadricepsMaskLeft(y(1):y(2),x(1):x(2))=roipoly;
366 axes(handles.axes4);
367 imshow((handles.SelectedAreaLeft + handles.QuadricepsMaskLeft*120),[-125 225]);
368 axes(handles.axes7);
369 imshow((handles.SelectedAreaLeft + handles.QuadricepsMaskLeft*120),[-125 225]);
370
371 [pixelnumber, n]=size(find(handles.QuadricepsMaskLeft == 1));
372 quadricepsarea = pixelnumber*handles.info.PixelSpacing(1)*handles.info.PixelSpac
    ing(2)/100;
373 meanhu = mean(handles.SelectedAreaLeft(find(handles.QuadricepsMaskLeft == 1)));
374 stdhu = std(handles.SelectedAreaLeft(find(handles.QuadricepsMaskLeft == 1)));
375 handles.infotext(8)={[sprintf('left   quadriceps cross sectional area [cm2] = %6
    .2f',quadricepsarea) '   '...
376                                         sprintf('mean density [HU] = %6.2f',
    meanhu) '   '...
377                                         sprintf('sd = ±%6.2f',stdhu)]};
378 set(findobj(handles.text2),'String',handles.infotext);
379
380 hleft = hist(handles.SelectedAreaLeft(find(handles.QuadricepsMaskLeft == 1)),han
    dles.HU);
381 handles.hleftquad = hleft/pixelnumber*100;
382
383 figure(handles.fig);
384 subplot(2,1,1);
385 bar(handles.HU,[handles.hrightquad' handles.hinit'],0.8,'r');
386 hold on;
387 bar(handles.HU,[handles.hinit' handles.hleftquad'],0.8,'b');
388 hold off;
389 axis([-150 150 0 40]);
390 xlabel('Hounsfield Units');
391 ylabel('cross sectional area [%]');
392
393 guidata(hObject, handles);
```



```
394
395 % --- Executes on button press in pushbutton8.
396 function pushbutton8_Callback(hObject, eventdata, handles)
397 % hObject    handle to pushbutton8 (see GCBO)
398 % eventdata  reserved - to be defined in a future version of MATLAB
399 % handles    structure with handles and user data (see GUIDATA)
400 axes(handles.axes4);
401 cla;
402 axes(handles.axes6);
403 cla;
404 handles.HamstringsMaskRight=zeros(size(handles.SelectedAreaRight));
405
406 axes(handles.axes4);
407 imshow(handles.SelectedAreaRight,[-125 225]);
408 [x,y]=ginput(2);
409 imshow(handles.SelectedAreaRight(y(1):y(2),x(1):x(2)),[-125 225]);
410 handles.HamstringsMaskRight(y(1):y(2),x(1):x(2))=roipoly;
411 axes(handles.axes4);
412 imshow((handles.SelectedAreaRight + handles.HamstringsMaskRight*120),[-125 225])
    ;
413 axes(handles.axes6);
414 imshow((handles.SelectedAreaRight + handles.HamstringsMaskRight*120),[-125 225])
    ;
415
416 [pixelnumber, n]=size(find(handles.HamstringsMaskRight == 1));
417 hamstringsarea = pixelnumber*handles.info.PixelSpacing(1)*handles.info.PixelSpac
    ing(2)/100;
418 meanhu = mean(handles.SelectedAreaRight(find(handles.HamstringsMaskRight == 1)))
    ;
419 stdhu = std(handles.SelectedAreaRight(find(handles.HamstringsMaskRight == 1)));
420 handles.infotext(10)={[sprintf('right hamstrings cross sectional area [cm2] = %6
    .2f',hamstringsarea) '   '...
421                                         sprintf('mean density [HU] = %6.2f',
    meanhu) '   '...
422                                         sprintf('sd = ±%6.2f',stdhu)]};
423 set(findobj(handles.text2),'String',handles.infotext);
424 hright = hist(handles.SelectedAreaRight(find(handles.HamstringsMaskRight == 1)),
    handles.HU);
425 handles.hrightham = hright/pixelnumber*100;
426
427 figure(handles.fig);
428 subplot(2,1,2);
429 bar(handles.HU,[handles.hrightham' handles.hinit'],0.8,'r');
430 hold on;
431 bar(handles.HU,[handles.hinit' handles.hleftham'],0.8,'b');
432 hold off;
433 axis([-150 150 0 40]);
434 xlabel('Hounsfield Units');
435 ylabel('cross sectional area [%]');
436
437 guidata(hObject, handles);
438
439
440 % --- Executes on button press in pushbutton9.
441 function pushbutton9_Callback(hObject, eventdata, handles)
442 % hObject    handle to pushbutton9 (see GCBO)
443 % eventdata  reserved - to be defined in a future version of MATLAB
444 % handles    structure with handles and user data (see GUIDATA)
445 axes(handles.axes4);
446 cla;
447 axes(handles.axes8);
```

```
448 cla;
449 handles.HamstringsMaskLeft=zeros(size(handles.SelectedAreaLeft));
450
451 axes(handles.axes4);
452 imshow(handles.SelectedAreaLeft,[-125 225]);
453 [x,y]=ginput(2);
454 imshow(handles.SelectedAreaLeft(y(1):y(2),x(1):x(2)),[-125 225]);
455 handles.HamstringsMaskLeft(y(1):y(2),x(1):x(2))=roipoly;
456 axes(handles.axes4);
457 imshow((handles.SelectedAreaLeft + handles.HamstringsMaskLeft*120),[-125 225]);
458 axes(handles.axes8);
459 imshow((handles.SelectedAreaLeft + handles.HamstringsMaskLeft*120),[-125 225]);
460
461 [pixelnumber, n]=size(find(handles.HamstringsMaskLeft == 1));
462 hamstringsarea = pixelnumber*handles.info.PixelSpacing(1)*handles.info.PixelSpac
    ing(2)/100;
463 meanhu = mean(handles.SelectedAreaLeft(find(handles.HamstringsMaskLeft == 1)));
464 stdhu = std(handles.SelectedAreaLeft(find(handles.HamstringsMaskLeft == 1)));
465 handles.infotext(11)={[sprintf('left   hamstrings cross sectional area [cm2] = %
    6.2f',hamstringsarea) '   '...
466                                         sprintf('mean density [HU] = %6.2f',
    meanhu) '   '...
467                                         sprintf('sd = ±%6.2f',stdhu)]};
468 set(findobj(handles.text2),'String',handles.infotext);
469 hleft = hist(handles.SelectedAreaLeft(find(handles.HamstringsMaskLeft == 1)),han
    dles.HU);
470 handles.hleftham = hleft/pixelnumber*100;
471
472 figure(handles.fig);
473 subplot(2,1,2);
474 bar(handles.HU,[handles.hrightham' handles.hinit'],0.8,'r');
475 hold on;
476 bar(handles.HU,[handles.hinit' handles.hleftham'],0.8,'b');
477 hold off;
478 axis([-150 150 0 40]);
479 xlabel('Hounsfield Units');
480 ylabel('cross sectional area [%]');
481
482 guidata(hObject, handles);
483
484
485 % --- Executes on button press in pushbutton10.
486 function pushbutton10_Callback(hObject, eventdata, handles)
487 % hObject    handle to pushbutton10 (see GCBO)
488 % eventdata  reserved - to be defined in a future version of MATLAB
489 % handles    structure with handles and user data (see GUIDATA)
490 print;
491
492
493 % --- Executes on button press in pushbutton11.
494 function pushbutton11_Callback(hObject, eventdata, handles)
495 % hObject    handle to pushbutton11 (see GCBO)
496 % eventdata  reserved - to be defined in a future version of MATLAB
497 % handles    structure with handles and user data (see GUIDATA)
498
499 [pixelnumber, n]=size(find(handles.HamstringsMaskRight == 1));
500 hright = hist(handles.SelectedAreaRight(find(handles.HamstringsMaskRight == 1)),
    handles.HU);
501 handles.hrightham = hright/pixelnumber*100;
502
503 [pixelnumber, n]=size(find(handles.HamstringsMaskLeft == 1));
```



```
504 hleft = hist(handles.SelectedAreaLeft(find(handles.HamstringsMaskLeft == 1)),han
    dles.HU);
505 handles.hleftham = hleft/pixelnumber*100;
506
507 Data = [handles.HU' handles.hleftquad' handles.hrightquad'...
508         handles.hleftham' handles.hrightham'];
509
510 guidata(hObject, handles);
511
512 [s1,s2] = uiputfile('*.*', 'Save File');
513 dlmwrite ([s2 s1 '.dat'], Data, '\t');
514 %saveas(gcf,[s2 s1],'jpg');
515 saveas(gcf,[s2 s1],'m');
516
517
518 % --- Executes on button press in pushbutton12.
519 function pushbutton12_Callback(hObject, eventdata, handles)
520 % hObject    handle to pushbutton12 (see GCBO)
521 % eventdata  reserved - to be defined in a future version of MATLAB
522 % handles    structure with handles and user data (see GUIDATA)
523
524 figure(handles.fig);
525 %subplot(2,1,1);
526 %bar(handles.HU,[handles.hrightquad' handles.hinit'],0.8,'r');
527 bar(handles.HU,handles.hrightquad',0.8,'r');
528 %hold on;
529 %bar(handles.HU,[handles.hinit' handles.hleftquad'],0.8,'b');
530 %hold off;
531 axis([-150 150 0 40]);
532 xlabel('Hounsfield Units');
533 ylabel('cross sectional area [%]');
534 %subplot(2,1,2);
535 %bar(handles.HU,[handles.hrightham' handles.hinit'],0.8,'r');
536 %bar(handles.HU,handles.hrightham',0.8,'r');
537 %hold on;
538 %bar(handles.HU,[handles.hinit' handles.hleftham'],0.8,'b');
539 %hold off;
540 %axis([-150 150 0 40]);
541 %xlabel('Hounsfield Units');
542 %ylabel('cross sectional area [%]');
543
544 axes(handles.axes4);
545 %imshow((handles.SelectedAreaRight + handles.HamstringsMaskRight*120 + handles.Q
    uadricepsMaskRight*120),[-125 225]);
546 imshow((handles.SelectedAreaRight + handles.QuadricepsMaskRight*120),[-125 225])
    ;
547
548 % --- Executes on button press in pushbutton13.
549 function pushbutton13_Callback(hObject, eventdata, handles)
550 % hObject    handle to pushbutton13 (see GCBO)
551 % eventdata  reserved - to be defined in a future version of MATLAB
552 % handles    structure with handles and user data (see GUIDATA)
553
554 figure(handles.fig);
555 subplot(2,1,1);
556 bar(handles.HU,[handles.hrightquad' handles.hinit'],0.8,'r');
557 hold on;
558 bar(handles.HU,[handles.hinit' handles.hleftquad'],0.8,'b');
559 hold off;
560 axis([-150 150 0 40]);
561 xlabel('Hounsfield Units');
```

```
562 ylabel('cross sectional area [%]');
563 subplot(2,1,2);
564 bar(handles.HU,[handles.hrightham' handles.hinit'],0.8,'r');
565 hold on;
566 bar(handles.HU,[handles.hinit' handles.hlefthamˈ],0.8,'b');
567 hold off;
568 axis([-150 150 0 40]);
569 xlabel('Hounsfield Units');
570 ylabel('cross sectional area [%]');
571
572 axes(handles.axes4);
573 %imshow((handles.SelectedAreaLeft + handles.HamstringsMaskLeft*120 + handles.Qua
    dricepsMaskLeft*120),[-125 225]);
574 imshow((handles.SelectedAreaLeft + handles.QuadricepsMaskLeft*120),[-125 225]);
```

```
% DPulse mfile for analysing stimulation needle EMG measurments
% by Christian Hofer

clear all;
% importing the data measured with DasyLab into Matlab
[s1, s2] = uigetfile ('*.ASC', 'Open File');
cd (s2);
A=dlmread (s1,'\t',10,0);

A=A(:,[1 2])*1000;
O=A;
% filtering data

% 4th order butterworth low pass filter at 10kHz
[Num,Den] = butter(4,10000/50000);
A=filter(Num,Den,A);
% 4th order butterworth high pass filter at 20Hz
[Num,Den] = butter(4,20/50000,'high');
%2nd order butterworth high pass filter at 500Hz
%[Num,Den] = butter(2,500/50000,'high');
A=filter(Num,Den,A);

% searching for the stimulation artefacts and extracting the data to a separate
matrix
i = 1;
n = 1;
height = 5;     % window height in mV
%width = 120;     % window width in ms
width = 150;     % window width in ms
pre = 4;        % pretrigger in ms
p = pre/1000*50000;
w = width*(1/1000*50000)-p;     % Schreibweise wegen Darstellung von w als Integ
er
x=length(A(:,1));
while n < (x - w)
    if (A(n,1)>0.5  | A(n,1)<-0.5)
        if 0 < (n - p)
            C(:,i)=A(n-p:n+w,1);
            if i==1
                O1=O(n-p:n+w,1);
            end
            i=i+1;
            n=n+w;
        else
            n=n+w;
        end
    else
        n=n+1;
    end
end

if (i > 18)
    i =18;
end;

B=1:length(C(:,1));
B=B/50000*1000-pre;

figure;
orient landscape;
set (gcf,'Name',s1);
```



```
title([s2,s1],'FontSize',12);
hold on;
plot(B(1,:),C(:,1));
set (findobj ('Type','line'),'LineWidth',0.8);
plot(B(1,:),O1,':r');
set (findobj ('Type','line'),'LineWidth',0.5);
hold off;
axis([-pre 40 -height height]);
grid on;
xlabel('Time [ms]');
ylabel('Amplitude [mV]');
pause (0.1);
height=input('MWave max. Amplitude [mV]=');
axis([-pre 40 -height height]);

impd=input('Impulsdauer [ms]=');
da=75;
dauer=da * (50000 / 1000);
de=15;
delay = (de + impd) * (50000 / 1000);
height=input('MWave max. Amplitude [mV]=');

close;  %close figure after setting height and width according to actual signal

k = find (C > height);
C(k) = height;
k = find (C < -height);
C(k) = -height;

S=zeros(size(C));
D=zeros(size(C));
M=zeros(size(C));

% Delaywerte die zur Impulsdauer addiert werden

for s=3:i-1
    S(1+delay+4:dauer+delay+4,s)=C(1:dauer,1);  %Wert 4 zu delay addiert wg. Sti
mulatorverzögerung
    S(:,s)=S(:,s)+C(:,1);
    D(:,s)=C(:,s)-C(:,1);
%   D(:,s)=C(:,s)-C(:,2);   %Versuchweise 2. Antwort subtrahiert von Doppelpulsa
ntwort
    M(1:dauer,s)=D(1+delay+4:dauer+delay+4,s);  %Wert 4 zu delay addiert wg. Sti
mulatorverzögerung
    delay=delay-50;
end

figure;
orient tall;
grid on;
set (gcf,'Name',s1);
title('2nd Impulse EMG','FontSize',14);
xlabel('Time [ms]');
%ylabel('Amplitude [mV]');
ylabel([s2 s1]);
axis([-1.3 da -height height*16]);
hold on;
for n=2:16;
    plot(B(1,:),(D(:,19-n) + height*(n-1)));
    set (findobj ('Type','line'),'LineWidth',0.8);
end;
```

```
118 plot(B(1,:),C(:,1),'r');
119 set (findobj ('Type','line'),'LineWidth',0.8);
120 plot(B(1,:),C(:,2));
121 set (findobj ('Type','line'),'LineWidth',0.8);
122 %axis([-pre da -height 4]);
123 hold off;
124
125 figure;
126 orient tall;
127 grid on;
128 set (gcf,'Name',s1);
129 title('Double-Pulse EMG','FontSize',14);
130 xlabel('Time [ms]');
131 %ylabel('Amplitude [mV]');
132 ylabel([s2 s1]);
133 axis([-1.3 da -height height*16]);
134 hold on;
135 for n=2:16;
136     plot(B(1,:),(C(:,n+1) + height*(17-n)));
137     set (findobj ('Type','line'),'LineWidth',.5);
138 %    pause;
139 end;
140 plot(B(1,:),C(:,2));
141 set (findobj ('Type','line'),'LineWidth',.5);
142 %pause;
143 plot(B(1,:),C(:,1),'r');
144 set (findobj ('Type','line'),'LineWidth',.5);
145 %pause;
146 %axis([-pre da -height 4]);
147 hold off;
148
149 %print
150 %cd c:\programme\matlab\work
```

C Operating and programming manual for the developed stimulation equipment (in German)

Inhaltsverzeichnis:

Bedienungsanleitung

Beschreibung des Gerätes

Allgemeine Beschreibung

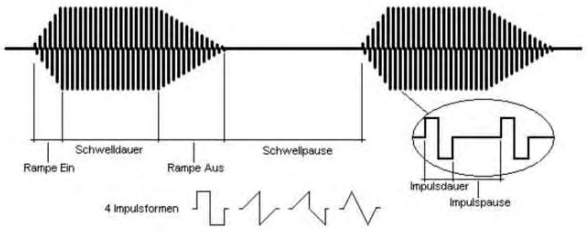
Das speziell für die Therapie denervierter Muskulatur entwickelte Elektrostimulationsgerät wird durch einen Akkumulator mit Energie versorgt. Zwei getrennte Stimulationskanäle erlauben gleichzeitiges Training von 2 verschiedenen Muskeln bzw. Muskelgruppen und damit eine Verkürzung der Gesamttherapiezeit.

Bis zu 9 verschiedene, freiprogrammierbare Stimulationsprogramme können mittels eines PDA (Personal Digital Assistant) in das Stimulationsgerät geladen werden. Ein Mikrocontroller übernimmt die Speicherung der Programme, Steuerung des Gerätes sowie die Kommunikation mit dem PDA über Infrarot-Schnittstelle (IrDA) und sorgt somit für größtmöglichen Bedienkomfort und einfache Bedienbarkeit.

Stimulationsprogramme

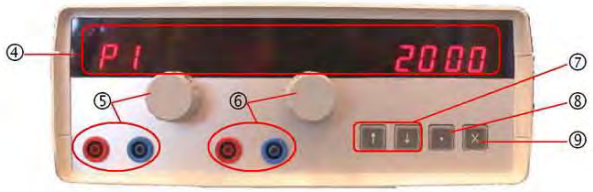
Nach Bedarf können bis zu 9 verschiedene Stimulationsprogramme auf den Speicherplätzen P1 – P9 abgelegt werden. Folgende Parameter können dabei entsprechend dem gewünschten Stimulationsprogramm innerhalb der angegebenen Grenzen eingestellt werden:

- Anfangsintensität Kanal 1 bei Start des Programms: 0 – 99%
- Anfangsintensität Kanal 2 bei Start des Programms: 0 – 99%
- Impulsdauer des biphasischen Stimulationsimpulses: 1,3 – 327,1ms
- Impulspause zwischen den Einzelimpulsen: 1 – 2000ms
- Impulsform: 4 verschiedene biphasische Stromformen sind möglich
- Schwelldauer: 1 – 10s
- Schwellpause: 0 – 10s
- Rampe Ein: 0 – 500ms
- Rampe Aus: 0 – 2000ms
- Programm- bzw. Stimulationsdauer: 3 – 40min



Beschreibung der Bedienelemente

Vorderseite des Gerätes:



Rückseite des Gerätes:

① EIN/AUS - Schalter

② Sicherung

③ Ladebuchse

④ LED - Anzeige für Programm, Intensität Kanal 1 und Kanal 2, verbleibende Programmlaufzeit

⑤ Intensitätsregler Kanal 1 und Anschlussbuchsen Kanal 1

⑥ Intensitätsregler Kanal 2 und Anschlussbuchsen Kanal 2

⑦ Programmwahltasten

⑧ Taste für Programm - Neustart bzw. - Wiederaufruf

⑨ START/STOPP - Taste zum Aktivieren, vorübergehenden Anhalten bzw. vorzeitigen Beenden des gewählten Therapieprogramms

① EIN/AUS - Schalter

In der Stellung „0“ wird das Stimulationsgerät allpolig über ein Relais vom eingebauten Akkumulator getrennt. Durch Kippen des Schalters in die Position „I“ wird das Gerät eingeschaltet. Auf der LED - Anzeige an der Vorderseite des Gerätes erscheint dann automatisch das Programm „P1“ und die programmierte Stimulationsdauer.

② Sicherung

Sollte sich das Stimulationsgerät nicht einschalten lassen und das Ladegerät nicht an der Ladebuchse des Stimulationsgerätes angeschlossen sein, kann die Ursache eine defekte Sicherung sein. In diesem Fall kann die Sicherung durch eine in Nennwert und Auslösecharakteristik gleichwertige getauscht werden.

③ Ladebuchse

Während des Ladevorgangs kann das Elektrostimulationsgerät nicht verwendet werden, da aus Sicherheitsgründen der Akkumulator mit dem angeschlossenen Ladegerät vom Stimulationsgerät über ein Relais allpolig getrennt wird. Erst nach dem Entfernen des Ladegerät-Steckers aus der Ladebuchse des Stimulationsgerätes kann die Therapie durchgeführt werden.

④ LED - Anzeige

Die LED - Anzeige zeigt je nach Betriebszustand folgende Informationen an:

- Programmnummer
- Programmdauer bzw. verbleibende Restdauer
- Intensität Kanal 1 und Kanal 2

⑤ Intensitätsregler und Anschlussbuchsen Kanal 1

Nachdem das gewünschte Stimulationsprogramm gewählt und gestartet wurde, kann mit dem Intensitätsregler die gewünschte Stimulationsamplitude für den Kanal 1 eingestellt werden. Ist das Programm angehalten oder noch nicht gestartet, hat der Regler keine Funktion.
Die Buchsen dienen zum Anschluss der Elektrodenkabel an den Kanal 1 des Stimulationsgerätes.

Achtung: Vor dem Start des Stimulationsprogramms immer die Zuordnung der Kabel zu den Stimulationskanälen überprüfen, um ein irrtümliches „kreuzweises“ Stimulieren zu verhindern.

⑥ Intensitätsregler und Anschlussbuchsen Kanal 2

Nachdem das gewünschte Stimulationsprogramm gewählt und gestartet wurde, kann mit dem Intensitätsregler die gewünschte Stimulationsamplitude für den Kanal 2 eingestellt werden. Ist das Programm angehalten oder noch nicht gestartet, hat der Regler keine Funktion.

Die Buchsen dienen zum Anschluss der Elektrodenkabel an den Kanal 1 des Stimulationsgerätes.

Achtung: Vor dem Start des Stimulationsprogramms immer die Zuordnung der Kabel zu den Stimulationskanälen überprüfen, um ein irrtümliches „kreuzweises“ Stimulieren zu verhindern.

⑦ Programmwahltasten

Mit den Programmwahltasten kann entsprechend der Pfeilrichtung das gewünschte Stimulationsprogramm gewählt werden. Deaktivierte Stimulationsprogramme bzw. deren Speicherplätze werden übersprungen, die Anzeige springt z.B. bei Wahl der nächsthöheren Programmnummer von „P2“ auf „P6“.

⑧ Programm – Neustart bzw. - Wiederaufruf

Mit dieser Taste kann das Stimulationsprogramm nach Ablauf der Therapiezeit neugestartet werden. Intensität und Programmdauer springen auf die programmierten Ausgangswerte zurück.

⑨ START/STOPP - Taste

Nach der Wahl des Stimulationsprogramms wird mit dieser Taste das Programm gestartet. Nochmaliges Drücken der Taste stoppt die Stimulation. Die eingestellte Stimulationsamplitude und verbleibende Restdauer bleiben gespeichert, sodass die Stimulation nach der Unterbrechung mit den vorher eingestellten Werten fortgesetzt wird.

Behandlung (Durchführung der Stimulation)

Elektrodenanlage

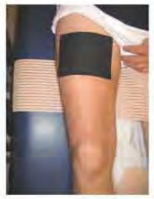
Gummielektroden mit Elektrodentaschen

Die Elektrodentaschen aus Schwammtuch sollen beinahe tropfnass mit der dicken Schicht (mehrlagige Seite) auf die Haut gelegt werden. Dabei ist zu beachten, dass die Öffnungen für die Elektrodenplatten voneinander abgewandt sind. Dies gewährleistet eine möglichst großflächige Verteilung des elektrischen Feldes.
Die Gummielektroden vollständig in die Elektrodentaschen einschieben. Die Elektrode an den Ecken nicht knicken, da an den umgebogenen Ecken durch die erhöhte Stromdichte Verbrennungen auftreten können.


- beinahe tropfnass
- mit dicker Seite auf die Haut
- Öffnung der Elektrodentaschen möglichst weit auseinander
- vollständig in die Elektrodentasche einschieben
- keine umgebogenen Ecken oder Kanten

Weichgummielektroden mit Elektrodengel

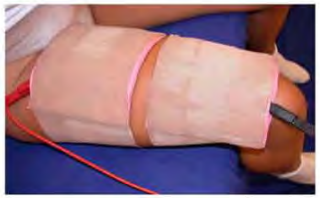
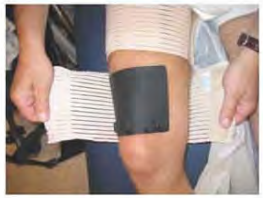
Elektrodengel gleichmäßig über die gesamte Fläche der Elektrode auftragen und diese auf die Haut auflegen. Danach möglichst nicht mehr auf der Haut verschieben und mittels Bandagen oder elastischer Fixiergurte befestigen.


- ausreichend Elektrodengel auftragen
- gesamte Fläche der Elektrode bestreichen

Fixierung der Elektroden

Jede Elektrode einzeln mit einer Bandage oder einem elastischen Fixiergurt mit Klettverschluss, welcher die gesamte Elektrodenfläche bedeckt, befestigen. Dabei auf einen möglichst gleichmäßigen Anlagedruck achten. Einzeln deshalb, damit ein elektrischer Nebenschluss durch feuchte Bandagen ausgeschlossen wird (Verminderung der Wirksamkeit der Therapie!).

Achtung: In der Bandage dürfen keine metallischen Fäden mitverwoben sein – Verbrennungsgefahr!

- Elektroden einzeln befestigen
- gleichmäßiger Anlagedruck

Wahl des Stimulationsprogramms

Nachdem die Elektroden angelegt und an das Stimulationsgerät angeschlossen wurden, kann das entsprechende Stimulationsprogramm mit den Programmwahltasten gewählt werden. Mit der START/STOPP - Taste wird danach das Programm aktiviert und die gewünschte Behandlungsintensität mit den Reglern für Kanal 1 und Kanal 2 eingestellt. Die Stimulation kann jederzeit durch nochmaliges Drücken der START/STOPP - Taste unterbrochen werden. Die vorher eingestellte Stimulationsamplitude und die verbleibende restliche Behandlungsdauer bleiben dabei gespeichert. Soll die Stimulation komplett neu gestartet werden, ist dies mit der Taste für Programm - Neustart bzw. - Wiederaufruf möglich.

Behandlungsende

Nach Ablauf der voreingestellten Programmdauer wird das Stimulationsprogramm gestoppt. Ist ein vorzeitiges Ende der Therapie gewünscht, so kann das laufende Programm jederzeit mit der START/STOPP - Taste angehalten werden.

Laden des Elektrostimulationsgerätes

Je nach verordnetem Trainingsprogramm können mit dem Stimulationsgerät 1 – 2 (max. 3) Trainingseinheiten durchgeführt werden. Ein Summton signalisiert die zu Ende gehende Kapazität des Akkumulators. Zum Laden wird das mitgelieferte Ladegerät an die Ladebuchse auf der Rückseite des Stimulationsgerätes angesteckt. Eine zuerst blinkende, dann dauernd rot leuchtende LED (Leuchtdiode) auf dem Ladegerät zeigt den Ladevorgang an. Nach Abschluss des Ladevorgangs schaltet das Ladegerät automatisch auf Erhaltungsladung um (grüne Leuchtdiode). Ein Überladen des Elektrotherapiegeräts ist mit dem mitgelieferten Ladegerät nicht möglich.
Ungefähr einmal pro Monat sollte der Akkumulator vor dem Wiederaufladen vollständig entleert werden. Dazu muss die rote Taste am Ladegerät gedrückt und solange gehalten werden bis die rote LED zu blinken beginnt. Nach der vollständigen Entladung des Akkumulators wird der Ladevorgang automatisch gestartet.

Achtung: Während des Ladevorgangs kann das Elektrostimulationsgerät nicht verwendet werden!

Pflege, Wartung von Elektrostimulationsgerät und Zubehör

Wartung des Elektrostimulationsgerätes

Das Gerät hat keine durch den Anwender zu wartende Teile. Im Falle von Fehlfunktionen ist die Wartung oder Reparatur nur durch das zuständige Fachpersonal zulässig.

Reinigung des Elektrostimulationsgerätes

Das Gerät wird mit einem trockenen oder leicht feuchten Tuch außen abgewischt. Keinesfalls nass abwaschen oder Reinigungsmittel verwenden.

Achtung: Eingedrungenes Wasser kann einen Kurzschluss verursachen.

Pflege der Elektroden

Die Gummielektroden werden nach Gebrauch mit lauwarmem Wasser abgespült, ohne daran zu reiben (Graphitverlust bedeutet erhöhten Widerstand).
Danach diese zum Trocknen auflegen oder mit einem Tuch bzw. einer Küchenrolle abtupfen.

Pflege der Elektrodentaschen (Schwammtuch)

Die Elektrodentaschen werden nach jedem Gebrauch ebenfalls in warmem Wasser ausgespült. Einmal pro Woche sollten diese bei 60° C in der Waschmaschine mit etwas Waschpulver gereinigt werden.

Pflege der Bandagen und der Fixiergurte

Die Bandagen oder Fixiergurte ebenfalls zumindest einmal pro Woche ausspülen oder waschen und regelmäßig auswechseln.

Elektrodenkabel

Elektrodenkabel nie knicken, nicht dehnen und nach Gebrauch nicht zu straff bzw. über scharfe Kanten aufwickeln.
Regelmäßige Kontrolle, ob Schäden an der Isolation der Kabel vorhanden sind. Ist dies der Fall, Kabel durch neue ersetzen.

Elektrodenklemmen

Die Klemmen sollten regelmäßig gereinigt werden, da verschmutzte Elektrodenklemmen den Übergangswiderstand dramatisch erhöhen. Dies führt zu einer Reduktion der Wirksamkeit der Therapie (sehr schwache Muskelkontraktion).
Bei starker, hartnäckiger Verschmutzung (dunkler Belag durch Gelreste etc.) auf den Metallteilen der Elektrodenklemmen, diese mit Stahlwolle oder ähnlichem entfernen.

Probleme, Lösungsvorschläge

Rötungen:

Eine gleichmäßige Rötung der Haut unter den Elektroden ist normal (gesteigerte Durchblutung) und verschwindet nach einigen Stunden wieder.
Tritt die Rötung unregelmäßig (z.B. punktförmig) auf, muss sie im weiteren Verlauf genau beobachtet werden. Bleiben die Hautveränderungen bestehen, sollte Kontakt mit dem betreuenden Arzt aufgenommen werden.

Verbrennungen:

Auf Grund der fehlenden Sensibilität in den Beinen kann eine unsachgemäße Elektrodenanlage zu hohen lokalen Stromdichten führen und eine Verbrennung der Haut hervorrufen.

Punktförmige Verbrennungen werden trocken behandelt. Damit eine weitere Stimulation möglich ist, wird die betroffene Stelle während jeder Strombehandlung mit Zinkpaste und kleinen Silikon-Gummiplättchen isoliert.
Erst nach völliger Abheilung (nach ca. 2 - 4 Wochen) kann wieder ohne Isolierung stimuliert werden.

Größere Verbrennungen müssen vom Arzt behandelt werden. Die Stimulation muss in diesem Fall ausgesetzt werden.

Mit zunehmender Stimulationsdauer (2 - 3 Monate) adaptiert sich die Haut an die Elektrostimulation und das Risiko einer Verbrennung nimmt in der Regel deutlich ab.
Nach ca. 4 - 5 Monaten treten solche Verbrennungen nur mehr bei unsachgemäßer Handhabung der Stimulationselektroden auf.

Schwache Muskelkontraktion:

Bei zu starker Muskelermüdung bzw. Überbeanspruchung kann es zu einem Abnehmen der Muskelkontraktionsstärke kommen. Grund dafür können zu hohe Wiederholungszahlen oder Serienzahlen, zu hohe Zusatzgewichte, zu kurze Serienpausen oder zu intensives Wochentrainingsprogramm sein.
Eine ungünstige Körperhaltung während der Stimulation kann Ursache für schwache Kontraktion der stimulierten Muskulatur sein.
In all diesen Fällen muss Kontakt mit dem betreuenden Arzt aufgenommen werden, der eine Adaptierung des Stimulationstrainings vornimmt.

Bei Auftreten von Krankheiten, die das allgemeine Wohlbefinden deutlich beeinträchtigen (z. B. grippaler Infekt), soll die Elektrostimulation ausgesetzt werden und Kontakt mit dem betreuenden Arzt aufgenommen werden.

Keine bzw. unregelmäßige Muskelkontraktion bei voller Stimulation:

Bei voller Stimulation erfolgt keine oder nur schwache Muskelreaktion, wenn in der Kabelverbindung ein Fehler aufgetreten ist. D. h., es kann entweder das Kabel am Stimulationsgerät nicht richtig angesteckt sein oder an der Elektrode nicht richtig angeklemmt sein bzw. kann ein Kabelbruch (Wackelkontakt) vorliegen.

Wenn sich die Elektroden oder die Elektrodentaschen berühren, kann es zu einem Kurzschluss kommen. Vor Stimulationsbeginn muss daher auf die korrekte Positionierung der Elektroden oder der Elektrodentaschen geachtet werden.

Eine weitere Ursache für eine schwache Muskelkontraktion kann ein zu hoher Übergangswiderstand sein. Dieser kann durch eine Verschmutzung der Elektrodenklemmen (Elektrodenabrieb und Gelreste) bedingt sein, aber auch durch eine Verunreinigung des Schwammtuches durch Hautschuppen, Fett und die mikrobielle Flora der Haut sowie durch einen Verschleiss der Elektrodentaschen.

Der Übergangswiderstand kann auch durch mangelnde Feuchtigkeit der Elektrodentaschen oder Verwendung von zu wenig Elektrodengel erhöht werden.
Auch die Anwendung einer fetthaltigen Creme vor der Elektrostimulation steigert den Hautwiderstand.

Diese Probleme können vermieden werden, wenn folgendes beachtet wird:

Die Klemmen sollen regelmäßig kontrolliert werden und bei Bedarf mit Stahlwolle oder ähnlichem gereinigt werden.

Eine Verschmutzung des Schwammtuches kann durch Ausspülen nach der Stimulation sowie regelmäßiges Waschen in der Waschmaschine vermieden werden.
Weiters sollen die Elektrodentaschen in regelmäßigen Abständen erneuert werden.
Die Elektrodentaschen müssen ausreichend befeuchtet werden (tropfnass), bei Verwendung von Gelelektroden muss genügend Gel aufgetragen werden.

Die Haut über der zu stimulierenden Muskulatur soll vor dem Training gereinigt und entfettet werden.

Stromgefühl im Bauch-, Beckenbereich:

Kommt es zu einem Auftreten von Stromgefühl im Bauch-, Beckenbereich während der Stimulation ist es zu einem „gekreuzten" Anstecken der Elektrodenkabel am Stimulationsgerät gekommen.
Vor Beginn der Stimulation soll daher immer die korrekte Zuordnung der Kabel zu den Stimulationskanälen überprüft werden.

Stimulationsgerät lässt sich nicht einschalten:

Dieses Problem kann auftreten, wenn das Ladegerät noch an der Ladebuchse angesteckt ist, die Sicherung defekt ist oder der Akkumulator vollständig entladen ist.
Je nach Ursache muss das Ladegerät vom Stimulationsgerät abgesteckt werden, die Sicherung getauscht werden oder das Gerät aufgeladen werden.

Summton während der Stimulation:

In diesem Fall ist der Akkumulator fast vollständig entladen und muss mittels Ladegerät aufgeladen werden.

Gerät funktioniert nicht:

Sollte das Gerät aus einem für den Anwender nicht ersichtlichen Grund nicht ordnungsgemäß funktionieren (alle oben angeführten möglichen Fehlerursachen ausgeschlossen), so muss Kontakt mit dem technischen Fachpersonal aufgenommen werden.

Übersicht Probleme, Lösungsvorschläge

PROBLEM	MÖGLICHE URSACHEN		LÖSUNG
Verbrennungen	unsachgemäße Elektrodenanlage	Elektrodenplatte berührt die Haut Elektrodentaschen zu wenig feucht zu wenig Elektrodengel Anlagedruck inhomogen (punktförmig, abgehobene Elektroden, umgebogene Elektroden in den Elektrodentaschen, unregelmäßiges Bandagieren) metallische Fäden in Fixierbandagen	Elektrodenplatten ganz in die Elektrodetaschen einschieben Elektrodentaschen ausreichend befeuchten (tropfnass) ausreichend Elektrodengel auftragen gleichmäßiger Anlagedruck der gesamten Elektrode Kontrolle der Elektroden in den Elektrodentaschen Bandagen ohne metallische Fäden verwenden bei größeren Verbrennungen Kontrolle durch den Betreuenden Arzt
	zu dünne, verschlissene Elektrodentaschen		regelmäßiges Austausche der Elektrodentaschen
keine, schwache bzw. unregelmäßige Muskelkontraktion bei voller Stimulation	zu starke Muskelermüdung bzw. Überbeanspruchung	Serienpause zu kurz Serien zu zahlreich Wiederholungszahl zu hoch zu intensives Wochentrainingsprogramm zusätzliche Trainingsgewichte zu hoch	Rücksprache mit dem betreuenden Arzt
	ungünstige Körperhaltung während der Stimulation		Rücksprache mit dem betreuenden Arzt
	Krankheiten (Grippe, sonstige Infektionen..)		Aussetzen der Elektrostimulation, Kontaktaufnahme mit dem betreuenden Arzt
	Fehler bei Kabelverbindungen	nicht sachgemäß angesteckt bzw. angeklemmt Kabelbruch	Kontrolle der Kabelverbindung mit dem Stimulationsgerät bzw. den Elektroden Kontrolle der Kabel
	Elektrodentaschen berühren sich	Kurzschlussgefahr	Kontrolle der Elektrodenanlage vor Start des Stimulationsprogramms
	stark verschmutzte Elektrodenklemme	zu hoher Widerstand	regelmäßige Kontrolle der Klemmen bei Bedarf Reinigung mit Stahlwolle oder ähnlichem
	Elektrodentaschen sind stark verschmutzt		Ausspülen der Taschen nach der Stimulation, 1x wöchentlich waschen
	Elektrodentaschen sind verschlissen		regelmäßiges Austauschen der Elektrodentaschen
	Elektrodentaschen zu wenig feucht		Elektrodentaschen beinahe tropfnass verwenden
	zu wenig Elektrodengel		ausreichend Elektrodengel auftragen
	verbrauchte Gummielektroden		Elektroden gegen neue austauschen
	Haut vor der Therapie zu fett eingecremt	zu hoher Widerstand	Elektroden auf gereinigte, entfettete Haut legen
Stromgefühl im Bauch-, Beckenbereich	Fehler beim Anklemmen der Elektroden	falsche "gekreuzte" Zuordnung der Stimulationskanäle	Kontrolle der Zuordnung der Kabel zu den Stimulationskanälen
Stimulationsgerät lässt sich nicht einschalten	Ladegerät an der Ladebuchse angesteckt		Ladegerät vom Stimulationsgerät abstecken
	Sicherung defekt		Sicherung tauschen
	Akkumulator vollständig entladen		Akkumulator aufladen
Summton während der Stimulation	Akkumulator fast leer		Stimulationsgerät vor dem Training aufladen
Gerät funktioniert nicht	technisches Problem im Gerät		Rücksprache mit techn. Fachpersonal

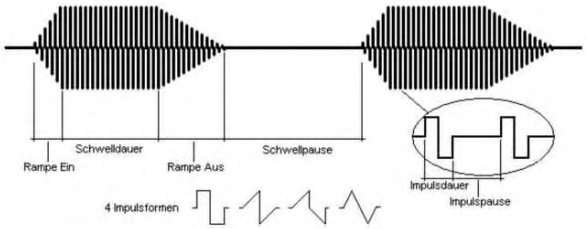
Programmieranleitung

Inhaltsverzeichnis:

Beschreibung des Konzepts

Die Programmierung der Elektrostimulationsgeräte erfolgt mit Hilfe einer auf dem Palm-OS Betriebsystem basierenden Software. Der Vorteil ist die einfache, intuitiv zu bedienende Benutzeroberfläche und die hohe Mobilität, da kein Personal-Computer oder Notebook für die Programmierung der Geräte notwendig ist. Durch die Kommunikation mittels Infrarot-Schnittstelle ist zwischen dem PDA (Personal Digital Assistant) und dem Elektrotherapiegerät keine elektrisch leitende Verbindung notwendig.

Die Stimulationsparameter im Überblick



Programmierung des Elektrostimulationsgerätes

1. Elektrostimulationsgerät einschalten (Kippschalter auf der Rückseite). Sollte sich das Gerät nicht einschalten lassen, kontrollieren, ob das Ladegerät angesteckt oder die Sicherung defekt ist.

2. Zur Aktivierung der Infrarot-Schnittstelle im Elektrostimulationsgerät den Stabmagneten über der Anzeige für das Stimulationsprogramm auf das Gerät legen.

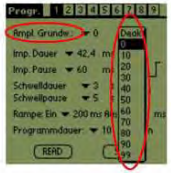
3. PDA Einschalten

4. Programm „DenStim“ auf dem PDA starten

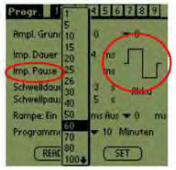
5. In der Benutzeroberfläche des Programms den gewünschten Programmspeicherplatz für das zu speichernde Programm wählen (1-9)

6. Die Parameter des Programms einstellen:

Grundwert der Amplitude für Kanal 1 und Kanal 2 einstellen bzw. den Stimulationskanal deaktivieren, wenn nur 1 Kanal verwendet werden soll oder beide Kanäle deaktivieren, um das gewählte Programm am Stimulationsgerät zu deaktivieren.



Impulsdauer und Impulspause wählen und durch mehrmaliges Tippen auf die grafische Darstellung des Stimulationsimpulses die Impulsform festlegen.



Die Schwelldauer („Burst-Dauer“) und die Schwellpause (Zeitdauer zwischen den Bursts) einstellen.

Die Rampe am Anfang („Rampe Ein“) und am Ende der Schwellung („Rampe Aus“) wählen.

Die Stimulationszeit einstellen („Programmdauer“).

7. Programm auf das Elektrostimulationsgerät übertragen
Die Schaltfläche „SET“ auf der Benutzeroberfläche des Programms antippen und den PDA in Richtung der Anzeige des Programmspeicherplatzes halten.

Achtung: Es muss eine Sichtverbindung zwischen den Infrarot-Schnittstellen auf dem PDA und dem Stimulationsgerät bestehen.

Auf dem Bildschirm des PDA erscheint statt dem Stimulationsgerätetyp („Akku“ oder „Netz“) die Meldung „Sendet“ und die Anzahl der Übertragungsversuche (max. 3 Versuche). Nachdem das Programm gespeichert wurde, springt die Anzeige (am PDA) wieder zurück und zeigt den Typ des Stimulationsgerät an. Ist die Übertragung nicht erfolgreich, so erscheint eine Fehlermeldung.

Nach erfolgreicher Übertragung des Programms erscheint am Stimulationsgerät der entsprechende Programmspeicherplatz und die gewählte Stimulationsdauer. Wird ein Programm übertragen während der Patient stimuliert, wird das laufende Programm angehalten und das soeben gespeicherte Programm auf der LED-Anzeige angezeigt.

Lesen der im Elektrostimulationsgerät gespeicherten Programme

1. Elektrostimulationsgerät einschalten (Kippschalter auf der Rückseite). Sollte sich das Gerät nicht einschalten lassen, kontrollieren, ob das Ladegerät angesteckt oder die Sicherung defekt ist.

2. Zur Aktivierung der Infrarot-Schnittstelle im Elektrostimulationsgerät den Stabmagneten über der Anzeige für das Stimulationsprogramm auf das Gerät legen.

3. PDA Einschalten

4. Programm „DenStim" auf dem PDA starten

5. In der Benutzeroberfläche der Anwendung den gewünschten Programmspeicherplatz für das zu lesende Programm wählen (1-9)

6. Programm lesen
   Die Schaltfläche „READ" auf der Benutzeroberfläche des Programms antippen und den PDA in Richtung der Anzeige des Programmspeicherplatzes halten.

   Achtung: Es muss eine Sichtverbindung zwischen den Infrarot-Schnittstellen auf dem PDA und dem Stimulationsgerät bestehen.

   Nachdem das Programm gelesen wurde, springt die Anzeige (am PDA) um und zeigt die im Elektrostimulationsgerät gespeicherten Parameter des ausgelesenen Programmspeicherplatzes an. Ist die Übertragung nicht erfolgreich, so erscheint eine Fehlermeldung.

Nachdem das Programm ausgelesen wurde, erscheint am Stimulationsgerät der entsprechende Programmspeicherplatz und die Stimulationsdauer für dieses Programm. Wird ein Programm gelesen während der Patient stimuliert, wird das laufende Programm angehalten und das soeben ausgelesene Programm auf der LED-Anzeige angezeigt.

Zusätzliche Funktionen: Datum, Uhrzeit, Protokoll

Benutzeroberfläche für die zusätzlichen Funktionen im Stimulationsgerät über das Menü „Test" aufrufen.
Auswählen der Schaltleiste „Serial Com.", im Display erscheint dann eine Zeile zur Eingabe der Befehle und ein mehrzeiliges Textfeld für die Anzeige der vom Stimulationsgerät gesendeten Daten.

Lesen und Setzen des Datums und der Uhrzeit im Stimulationsgerät

In der Befehlszeile den Kleinbuchstaben „u" für Uhrzeit eingeben und auf die Schaltfläche „SEND" tippen. Im Testfeld erscheint nach erfolgreicher Übertragung des Befehles das im Stimulationsgerät eingestellte Datum und die Uhrzeit.

Anzeigeformat: MM/TT hh:mm:ss

Zum Setzen des Datums und der Uhrzeit in der Befehlszeile den Großbuchstaben „U" gefolgt vom zu setzenden Datum und der Uhrzeit (z.B.: U0721135700 für den 21. Juli Uhrzeit: 13:57:00).
Hier erfolgt keine Rückmeldung vom Stimulationsgerät nach dem Übertragen der Daten mit der Schaltfläche „SEND", um das gesetzte Datum und die Uhrzeit zu kontrollieren, müssen die Daten wie oben beschrieben gelesen werden.

Abrufen und Löschen des Stimulationsprotokolls

Zum Abrufen des im Stimulationsgerät gespeicherten Stimulationsprotokolls den Großbuchstaben „R“ in der Befehlszeile eingeben und auf die Schaltflächen „SEND“ tippen. Im Textfeld erscheint dann das im Stimulationsgerät gespeicherte Protokoll.
Achtung: Die Übertragung kann je nach Umfang des gespeicherten Protokolls lange dauern. Während dieser Zeit können keine weiteren Befehle zum Stimulationsgerät gesendet werden.

Soll das Protokoll im Stimulationsgerät gelöscht werden den Großbuchstaben „S“ in der Befehlszeile eingeben und auf die Schaltflächen „SEND“ tippen. Im Textfeld erscheint hier keine Rückmeldung vom Stimulationsgerät. Zur Bestätigung der Löschung des Protokolls, Daten wie oben beschrieben auslesen.

Anzeigeformat der Protokollmeldungen

Einschalten des Elektrostimulationsgerätes:

MM/TT hh:mm:ss RESET

Start des Stimulationsprogramms mit der Nr. „X“ (1-9):

MM/TT hh:mm:ss PROGRAMM X

Anhalten bzw. Beenden des Stimulationsprogramms:

MM/TT hh:mm:ss STOP

Neu Programmierung eines Stimulationsprogrammes:

MM/TT hh:mm:ss NEU PROGRAMMIERUNG

Fehlermeldungen

Ist keine Übertragung über die Infrarot-Schnittstelle möglich bzw. sind die übertragenen Daten nicht vollständig, so erscheint am Display des PDA die Fehlermeldung „Data not transmitted!“.

Sichtverbindung zwischen den Infrarot-Schnittstellen muss bestehen. Optimale Distanz (ca. 30 – 50cm) für die Datenübertragung zwischen PDA und Stimulationsgerät einhalten.

Wenn nach 3 Versuchen das Stimulationsprogramm nicht fehlerfrei zum Elektrostimulationsgerät übertragen wurde, erscheint am Display des PDA die Fehlermeldung „Programm was not correct transmitted! Please try again“. In diesem Fall ist die Kommunikation zwischen PDA und Elektrostimulationsgerät möglich, doch wird die Datenübertragung durch eine andere Infrarotstrahlung emittierende Quelle gestört.

Versuch, die Übertragungsstrecke zwischen PDA und Stimulationsgerät gegen die Störstrahlung abzuschirmen.

Achtung: Bei den zusätzlichen Funktionen, welche über die Benutzeroberfläche „SerialCom Test“ aufgerufen werden, erfolgt keine Kontrolle der Datenübertragung.

Curriculum vitae

• Personal data

Christian Hofer

Born: July 18th, 1968, Vienna, Austria

Citizenship: Austria

• Education, academic and professional background

Polytechnical College (HTBLVA Wien 20, TGM), 1982-1987.

Civil Service, 1987 –1988.

Studies of Electrical Engineering and Electronics, University of Technology Vienna, 1988 – 1995.

Diploma Thesis at the Institute of Control and Industrial Electronics in cooperation with the Institute of Biomedical Engineering and Physics of the Medical University of Vienna, 1995.

Title: „Entwicklung eines Stimulators zur funktionellen Elektrostimulation der gelähmten Hand"

Graduation (passed with distinction): 1995.

Research collaborator at the Ludwig Boltzmann Institute for Electrical Stimulation and Physical Rehabilitation, Institute of Physical Medicine and Rehabilitation, Wilhelminenspital Vienna, since 1995.

Participated in the design and execution of several national and international research projects among 3 projects funded by the European Commission.

Reviewer for Artificial Organs, since 2005.

- **Scientific publications**

Meyerspeer M, Mandl T, Reichel M, Mayr W, Hofer C, Kern H, Moser E. Effects of functional electrical stimulation in denervated thigh muscles of paraplegic patients mapped with T (2) imaging. MAGMA 2008; Apr 19 (epub ahead of print).

Kern H, Hofer C, Mödlin M, Mayr W, Vindigni V, Zampieri S, Boncompagni S, Protasi F, Carraro U. Stable muscle atrophy in long-term paraplegics with complete upper motor neuron lesion from 3- to 20-year SCI. Spinal Cord 2008; 46(4):293-304.

Hofer C, Huber K, Dietl H, Mödlin M, Vogelauer M, Carraro U, Protasi F, Kern H. Funktionelle Elektrostimulation bei querschnittgelähmten Patienten. Orthopädie-Technik 2007; 7: 528-35.

Boncompagni S, Kern H, Rossini K, Hofer C, Mayr W, Carraro U, Protasi F. Structural differentiation of skeletal muscle fibers in the absence of innervation in humans. Proc Natl Acad Sci U S A 2007; 104(49):19339-44.

Minassian K, Persy I, Rattay F, Dimitrijevic MR, Hofer C, Kern H. Posterior root-muscle reflexes elicited by transcutaneous stimulation of the human lumbosacral cord. Muscle Nerve 2007; 35(3):327-36.

Kern H, Rossini K, Carraro U, Mayr W, Vogelauer M, Hoellwarth U, Hofer C. Muscle biopsies show that FES of denervated muscles reverses human muscle degeneration from permanent spinal motoneuron lesion. J Rehabil Res Dev 2005; 42(3 Suppl 1):43-53.

Hofer C, Forstner C, Mödlin M, Jäger H, Mayr W and Kern H. In Vivo Assessment of Conduction Velocity and Refractory Period of Denervated Muscle Fibers. Artif Organs 2005; 29(3):436-9.

Bijak M, Rakos M, Hofer C, Mayr W, Strohhofer M, Raschka D, Kern H. Stimulation parameter optimization for FES supported standing up and walking in SCI patients. Artif Organs 2005; 29(3):220-3.

Mödlin M, Forstner C, Hofer C, Mayr W, Richter W, Carraro U, Protasi F, Kern H. Electrical stimulation of denervated muscles: first results of a clinical study. Artif Organs 2005; 29(3):203-6.

Mayr W, Hofer C, Bijak M, Lanmüller H, Rafolt D, Reichel M, Sauermann S, Unger E, Kern H. (abstract) EU-Projekt RISE: FES denervierter Muskulatur. Biomedizinische Technik / Biomedical Engineering Band 48, Erg Band 1, 2003, 52.

Rafolt D, Gallasch E, Fend M, Hofer C, Bijak M, Lanmüller H, Sauermann S, Unger E, Mayr W. (abstract) Mechanomyographische Verfahren zur nichtinvasiven Bewertung der Muskeldynamik. Biomedizinische Technik / Biomedical Engineering Band 48, Erg Band 1, 2003, 58.

Hofer C, Mayr W, Stöhr H, Unger E, Kern H. A stimulator for functional activation of denervated muscles. Artif Organs 2002; 26:276-9.

Bijak M, Mayr W, Rakos M, Hofer C, Lanmuller H, Rafolt D, Reichel M, Sauermann S, Schmutterer C, Unger E, Russold M, Kern H. The Vienna functional electrical stimulation system for restoration of walking functions in spastic paraplegia. Artif Organs 2002; 26:224-7.

Kern H, Hofer C, Modlin M, Forstner C, Raschka-Högler D, Mayr W, Stöhr H. Denervated muscles in humans: limitations and problems of currently used functional electrical stimulation training protocols. Artif Organs 2002; 26:216-8.

Mayr W, Hofer C, Bijak M, Rafolt D, Unger E, Sauermann S, Lanmueller H, Kern H. Functional Electrical Stimulation (FES) of denervated muscles: existing and prospective technological solutions Basic Appl Myology 2002; 12(6):287-90.

Kern H, Hofer C, Mödlin M, Forstner C, Mayr W, Richter W. Functional Electrical Stimulation (FES) of Long-Term Denervated Muscles in Humans: Clinical Observations and Laboratory Findings. Basic Appl Myology 2002; 12(6):291-9.

Gföhler M, Angeli T, Eberharter T, Lugner P, Mayr W, Hofer C. Test bed with force-measuring crank for static and dynamic investigations on cycling by means of functional electrical stimulation. IEEE Trans Neural Syst Rehabil Eng 2001; 9(2):169-80.

Kern H, Hofer C, Strohhofer M, Mayr W, Mödlin M, Forstner C, Richter W. (abstract) Denervated Muscles in Human First Results of Training with Electrical Stimulation. Basic Appl Myology 2000; 10(1&2): 73.

Mayr W, Bijak M, Girsch W, Hofer C, Lanmüller H, Rafolt D, Rakos M, Sauermann S, Schmutterer C, Schnetz G, Unger E, Freilinger G. MYOSTIM-FES to prevent muscle atrophy in microgravity and bed rest: preliminary report. Artif Organs 1999; 23(5):428-31.

Kern H, Hofer C, Strohhofer M, Mayr W, Richter W, Stöhr H. Standing up with denervated muscles in humans using functional electrical stimulation. Artif Organs 1999; 23(5):447-52.

Rakos M, Freudenschuss B, Girsch W, Hofer C, Kaus J, Meiners T, Paternostro T, Mayr W. Electromyogram-controlled functional electrical stimulation for treatment of the paralyzed upper extremity. Artif Organs 1999; 23(5):466-9.

Bijak M, Hofer C, Lanmüller H, Mayr W, Sauermann S, Unger E, Kern H. Personal computer supported eight channel surface stimulator for paraplegic walking: first results. Artif Organs 1999; 23(5):424-7.

Hofer C, Kern H, Mayr W, Stöhr H, Abou-Zahra S. Funktionelle Elektrostimulation denervierter Muskulatur. Österr Z Phys Med 1997; (7) Supplement 2.