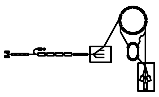

MS - Sweepermagnete

Gerätemodell und Softwareentwurf

L. Hechler

GSI, Gesellschaft für Schwerionenforschung mbH
Planckstraße 1, D-64291 Darmstadt
Tel.: 06159/71-0

Issued: 6. September 1999
Key: F-MS-01
Printed: 16. Oktober 2009

Beschreibung der Sweeper-Magnete des TK-Strippers.

?

Ein an der Seite weißt auf eine bisher noch ungeklärte Frage hin.

tbs

Ein an der Seite weist auf eine Stelle der Dokumentation hin, die noch ausgeführt oder vervollständigt werden muss.

Änderungsprotokoll			
Datum	GM-Version	Name	Kommentar
17. 12. 1998	MS_01	LH	Rahmen erstellt
12. 1. 1999	MS_01	LH	Beschreibung TK-Stripper; tnx Volker
2. 9. 1999	MS_01	LH	Gerätemodell-spezifische Properties
6. 9. 1999	MS_01	LH	Statusbits Rampengenerator
14. 10. 1999	MS_01	LH	Vorläufig fertig
7. 12. 1999	MS_01	LH	$f_{\mathrm{Q}} = 12\,\mathrm{Mhz}$ (bisher 8,192 MHz)
Mai 2000	–	MK	Überarbeitete und erweiterte TEX-Version, die sowohl in PostScript als auch in HTML konvertiert werden kann.
3. 12. 2002	MS_01	LH	BeamOn/Off-Istwert-Trigger im dyn. Status
23. 03. 2007	MS_01	LH	Gleichlaufgenauigkeit auf Basis von tmin = 120us
26. 03. 2007	MS_01	LH	$f_{\mathrm{Q}} = 12\,\mathrm{Mhz}$: trund = 10.667us (was 15.625); dImax/dt = 70kA/us (was 48)
23. 09. 2009	MS_01	LH	Beschreibung Hardware-Trigger

Inhaltsverzeichnis

Abbildungsverzeichnis

Teil I

Das Gerätemodell

1 Die Aufgabe des Gerätes

1.1 Der TK-Stripper

Die Sweepermagnete sind ein Teil des Strippersystems im Transferkanal TK.

Bei den bisherigen Strippersystemen konnten die Folien direkt in den Geradeausstrahl gefahren werden. Nach dem Umbau des UNILAC ist das im Mittel- und Hochstrombetrieb nicht mehr möglich. Der Strahl ist so energiereich, dass er die Folie thermisch zerstören würde. Aus diesem Grund müssen Ionenstrahlen mittlerer und hoher Energie über die Folie gewedelt werden (to sweep: fegen, überstreichen, ...), so dass während eines Pulses eine weitaus größere Folienfläche belastet wird.

Die Folienhalter mit den drei unterschiedlichen Folientypen, deren geplanten Abmessungen, sowie den Sweepbreiten und den Strahlflecklagen für verschiedene Betriebsmodi, sind in Abbildung 1 dargestellt.

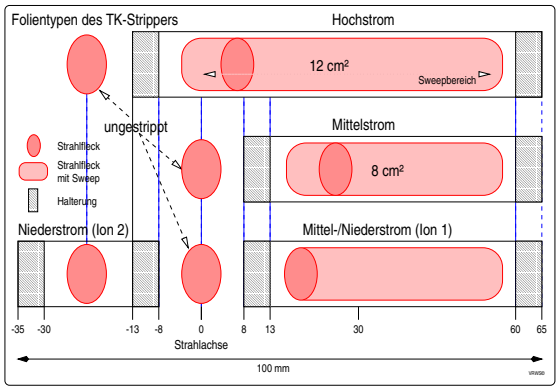

Abbildung 1: Folientypen

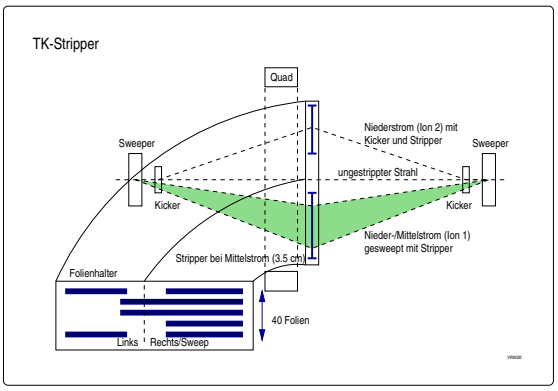

Abbildung 2: Anordnung der Folien

Die Anordnung der Folien ist in Abbildung 2 dargestellt. Es handelt sich hierbei um ein Folienband mit 40 Folien. Der Verlauf der Bahnen von zwei Ionen (einmal Mittel-/Hochstrom gesweept, einmal Niederstrom mit Stripper) und eines dritten ungestrippten Strahls ist angedeutet.

Das TK-Strippersystem besteht insgesamt aus dem Stripper mit 40 Folien, zwei Sweepern, zwei Kickern, einem und einem fokussierenden Quadrupol. Mit dieser Kombination sind verschiedene Betriebsmodi des Strippers möglich.

- Abbildung 3 zeigt den Hochstrommodus, bei dem eine breite Folie gewählt wird. Die Sweeper bewegen (sweepen) den Strahlfleck innerhalb eines Pulses über die gesamte, 68 mm breite Stripperfolie (vergleiche Abbildung 1). Das Sweepen des Strahlflecks sorgt für die gleichmäßige thermische Belastung der Folie. Für diesen Betriebsmodus müssen die Kicker auf Null stehen.

  Befindet sich eine breite Folie im Strahlweg, kann in anderen virtuellen Beschleunigern mit Hilfe der Kicker nur ungestrippter Strahl transportiert werden. Die Kicker lenken den Strahl auf einem für den gesamten Strahlpuls konstanten Weg an der breiten Hochstromfolie vorbei.

In diesem Fall müssen die Sweeper auf Null stehen.

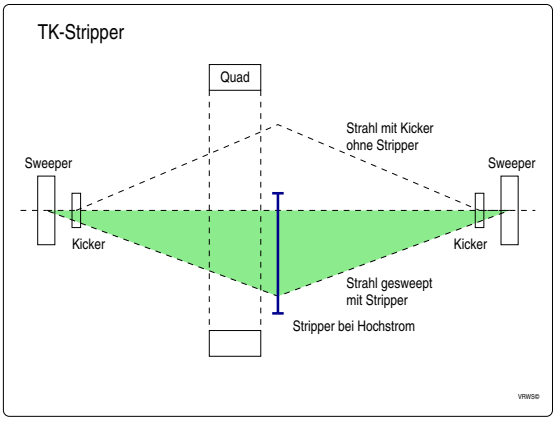

Abbildung 3: Hochstrommode

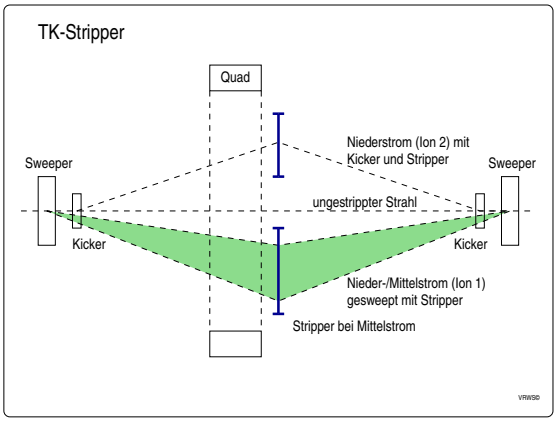

Abbildung 4: Mittel-/Niederstrommode

- Abbildung 4 zeigt die Anordnung für den Mittel-/Niederstrom-Mode. Hier wird eine schmalere Folie als im Hochstrommode für den Mittelstrom-Strahlweg genutzt. Die Sweeper sorgen auch hier für eine gleichmäßige Belastung der Folie. Der gesweepte Bereich reicht allerdings nicht bis zum Weg eines Geradeausstrahles.

  Damit kann in anderen virtuellen Beschleunigern ein ungestrippter Geradeausstrahl (Kicker und Sweeper stehen auf Null) sowie ein Niederstromstrahl mit Kickern und Stripper (Sweeper stehen auf Null) transportiert werden. Die zweite, schmale Folie ist nur für den Niederstrom vorgesehen.

  Varianten dieses Modus mit Null bis zwei Folien sind hier natürlich möglich.

- Zu Testzwecken („fädeln“) ist es auch möglich, die Sweeper statisch zu betreiben. In diesem Modus wird der *gesamte* (Niederstrom-) Strahlpuls über *einen* definierten Strahlweg geleitet.

Der Quadrupol dient zur Fokusierung des vom ersten Sweeper bzw. Kicker ausgelenkten Strahls. Der zweite Sweeper bzw. Kicker lenkt den Strahl wieder auf die Strahlachse ein.

1.2 Die Sweeper

Sweeper sind hochgenaue Magnete. Mit ihrer abfallende Rampe, deren Dauer in der Regel der des Strahlpulses entspricht, wird der Strahlpuls von außen (großer Ablenkwinkel) nach innen (kleiner Ablenkwinkel bzw. Ablenkwinkel Null) über die gesamte Breite der Stripperfolie gewedelt. Dies ist in Abbildung 5 dargestellt.

Vom Operating eingestellt wird das Feld im Flattop und die Dauer der abfallenden Rampe bis zum Nulldurchgang.

Die beiden Sweeper haben in der Regel unterschiedliche Einstellungen wegen der Umladung der Teilchen an der Stripperfolie. Der in Strahlrichtung erste Sweeper - also der *vor* dem Stripper - fährt in der Regel mehr Strom als der hintere.

Wichtig ist der *Gleichlauf* der beiden Sweeper-Rampen. Nur so kann der Strahl nach Aus- und Einlenkung wieder auf die Achse kommen.

Der Übergang vom konstanten Flattop in die Rampe muss verrundet werden (siehe Kapitel 1.2). Das Verrunden hat zur Folge, dass der Strahl zunächst nur langsam über die Folie gewedelt wird.

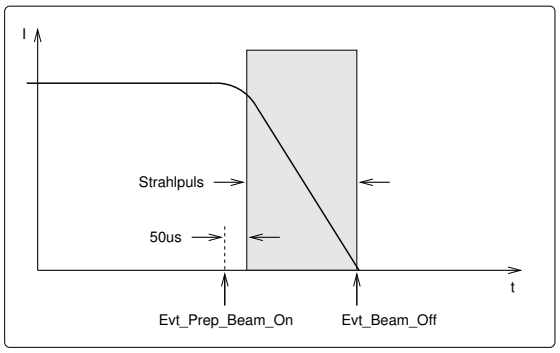

Abbildung 5: Sweeper-Rampe

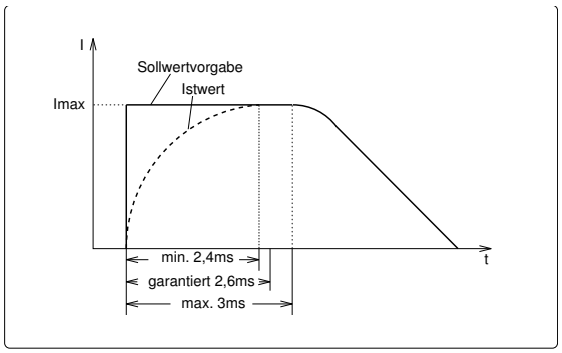

Abbildung 6: Vorgabe-Grenzwerte

Dies ist zulässig, da der Strahl am Anfang noch nicht so energiereich ist, um die Folie thermisch zu beschädigen.

Die Rampe wird mit einem Hardware-Trigger gestartet, da ein Software-Trigger zu ungenau ist. Der Hardware-Trigger wird vom Event `Prep_Beam_On` abgeleitet.

Für die Rampe ist eine Steigung von Null möglich.

Wie werden die Sweeper eingestellt, wenn eine kürzere Folie für Nieder- oder Mittelstrom (siehe Abbildung 4) genutzt werden soll, die nicht bis zum Geradeaus-Strahl reicht? ?

Die beiden Istwerte werden mit Hardware-Triggern, die von den Events `Prep_Beam_On` (4) und `Beam_Off` (8) abgeleitet werden, in die Interface-Hardware eingelatched.

2 Die Hardware des Gerätes

2.1 Rampengenerator

Bedingt durch die geforderte Genauigkeit sind die Netzgeräte sehr empfindlich bezüglich der Veränderung der Sollwertvorgaben. Der Übergang vom konstanten Flattop in die Rampe wird daher verrundet, um ein Überschwingen der Regelung zu vermeiden.

Die Verrundung und die Rampe werden digital mit einem Funktionsgenerator, dem sog. Rampengenerator, erzeugt.

Bedingt durch die interne Regelung des Netzgerätes und die zulässige Verlustleistung, müssen gewisse Grenzwerte bei der Sollwertvorgabe eingehalten werden. Der vorgegebene Flattop-Strom muss einlaufen bzw. einschwingen. Beim Nennstrom darf eine bestimmte Flattop-Dauer nicht überschritten werden, damit das Gerät nicht ausfällt. Dies ist in Abbildung 6 dargestellt.

Folgende Kennwerte der Sweeper sind wichtig für die Steuerung:

- Minimale Einlaufzeit von Null auf Nennwert inklusive Einschwingzeit

  $t_{\mathsf{Einlauf}} = 2.4\,\mathsf{ms}$

- Garantierte Einlaufzeit von Null auf Nennwert inklusive Einschwingzeit

  $t_{\mathsf{Einlauf}} = 2.6\,\mathsf{ms}$

- Maximale Flattopzeit inklusive Einschwingzeit bis zum Start der Rampe, genauer gesagt, bis zum Start der Verrundung

  $$t_{\mathsf{Flattop}} \leq 3\,\mathsf{ms}$$

- Minimale und maximale Zeit der abfallenden Rampe

  $$120\,\mu\mathsf{s} \leq t_{\mathsf{Rampe}} \leq 1\,\mathsf{ms}$$

2.2 A/D-Wandler

Die beiden DAC/ADC-Karten müssen am Schalter `CONF ADR` auf die Adressen 1 bzw. 2 eingestellt werden. Die linke Karte, die neben der Interfacekarte, bekommt die Adresse 2. Die rechte bekommt die Adresse 1.

Die rechte DAC/ADC-Karte ist für den Sollwert und den Istwert bei `Prep_Beam_On` zuständig, die linke für den Istwert bei `Beam_Off`.

2.3 Hardware-Trigger

2.3.1 Sollwert-Trigger

Der Hardware-Trigger zum Start der Rampe des Rampengenerators wird vom Event `Prep_Beam_On` abgeleitet. Er wird an die Buchse `EXT CLK` der Interface-Karte angeschlossen. Die Interface-Karte muss mindestens vom Typ `FG 380.221` sein.

Das Triggersignal muss nicht-invertiert sein. Der Generator wird mit der positiven Flanke des Triggers gestartet.

Der D/A-Wandler selbst ist freilaufend, da er ständig die vom Rampenenerator geschickten Werte konvertieren muss.

Aus Gründen des Gleichlaufs muss der Trigger zum Start des Rampengenerators für *beide* Sweeper aus *einem* Event-Dekoder abgeleitet werden!

2.3.2 Istwert-Trigger

Die beiden Hardware-Trigger zum Triggern der zwei A/D-Wandler werden von den Events `Prep_Beam_On` und `Beam_Off` abgeleitet. Sie werden an die Buchsen `EXT TRIG` der DAC/ADC-Karten angeschlossen. Die DAC/ADC-Karten müssen mindestens vom Typ `FG 429.064` sein.

Der von `Prep_Beam_On` abgeleitete Trigger wird an die DAC/ADC-Karte mit der `CONF ADR` 1 angeschlossen. Der von `Beam_Off` abgeleitete Trigger wird an die DAC/ADC-Karte mit der `CONF ADR` 2 angeschlossen.

Die Triggersignale müssen nicht-invertiert sein. Die Istwerte werden mit der positiven Flanke der Trigger konvertiert und gelatscht.

3 Die Schnittstelle zum Gerät

3.1 Der Rampengenerator

Die gesamte Sollwertvorgabe geschieht mittels des Rampengenerators. Die Rampe inklusive Verrundung wird digital generiert, da es analogen Methoden u. a. an Reproduzierbarkeit mangelt. Der

Rampengenerator ist eine Erweiterung des EPLDs auf der Interfacekarte.

Die Eigenschaften des Rampengenerators, Berechnung und Normierung der Sollwertvorgaben, die Rückrechnung der Vorgaben und eine Fehlerbetrachtung sind in Kapitel 4.1 zusammengefasst.

3.2 A/D-Wandler

Zum Lesen der Istwerte arbeitet das Gerät mit 16 Bits breiten A/D-Wandlern. Die Normierung ist wie folgt:

$$7FFF_{hex} \widehat{=} I_{Nenn}$$

3.3 Definition der Statusbits des Netzgerätes

Die Bits 0 ... 7 sind die systemweit einheitlichen Software-Statusbits. Die Statusbits im Einzelnen sind in der folgenden Tabelle zusammengefasst.

Bit	Name	Bedeutung	
		High (1)	Low (0)
0	Netz	ein	aus
1	Hoheit	Rechner	Hand
2	(reserved)	–	–
3	(reserved)	–	–
4	Emergency	no	yes
5	Interlock	no	yes
6	HW Warning	no	yes
7	SW Warning	no	yes
8	Netz	ein	aus
9	Netzspannung U_{Netz}	ok	zu niedrig
10	Temperatur Netzgerät	ok	zu hoch
11	Kühlwasser Netzgerät	ok	fehlt
12	Laststrom I_{ist}	ok	zu hoch
13	not used	–	–
14	Temperatur Magnet	ok	zu hoch
15	Kühlwasser Magnet	ok	fehlt
16	Stromsymmetrie ΔI_T	ok	zu hoch
17	Netzstrom I_{p1}	ok	zu hoch
18	not used	–	–
19	not used	–	–
20	Lastspannung $U_{v\,15}$	ok	zu hoch
21	not used	–	–
22	Stromwandler	ok	saturiert
23	Erdschluss	nein	ja
24	Hoheit	Rechner	Hand
25	not used	–	–
26	not used	–	–
27	not used	–	–
28	not used	–	–
29	not used	–	–
30	not used	–	–
31	not used	–	–

Eine Hardware-Warnung liegt vor, wenn eines der folgenden Bits den angegebenen Status anzeigt:

Bit	Name	Status
9	Netzspannung U_{Netz}	zu niedrig
10	Temperatur Netzgerät	zu hoch
11	Kühlwasser Netzgerät	fehlt
12	Laststrom I_{ist}	zu hoch
14	Temperatur Magnet	zu hoch
15	Kühlwasser Magnet	fehlt

3.4 Definition der Statusbits des Rampengenerators

Bit	Name	Bedeutung	
		High (1)	Low (0)
0...3	EPLD-Version	–	–
4	SW-Trigger	enabled	disabled
5	not used	–	–
6	BeamOff-Istwert-Trigger	angekommen	fehlt
7	BeamOn-Istwert-Trigger	angekommen	fehlt
8	Rampenstart-Trigger	angekommen	fehlt
9	Programmiersequenz	falsche Reihenfolge	ok
10	Timeout	aufgetreten	nicht aufgetreten
11	Rampe	anulliert	nicht anulliert
12	Zustand Rampengenerator	idle	(not idle)
13	Zustand Rampengenerator	working	(not working)
14	Zustand Rampengenerator	waiting for trigger	(not waiting)
15	not used	–	–

Die Bits 0...3 und 4 sind statisch. Die Bits 6 bis 15 sind dynamisch. Sie ändern sich von Zyklus zu Zyklus.

Bit 6: Der Trigger für den ADC zum Konvertieren des Istwertes beim Event Beam_Off ist ausgeblieben.

Bit 7: Der Trigger für den ADC zum Konvertieren des Istwertes beim Event Prep_Beam_On ist ausgeblieben.

Bit 8: Der Trigger zum Start der Rampe ist ausgeblieben.

Bit 9: Die vorgeschriebene Programmiersequenz *Delta*, *Delay*, *Flattop* wurde nicht eingehalten.

Bit 10: Zur Timeout-Zeit war die Rampe noch nicht gestartet. Das Gerät wurde mit der schnellsten Abwärtsrampe (größtmögliche Steilheit) auf Null gefahren.

Bit 11: Rampe wurde anulliert. Der Rampengenerator wurde vor dem Start der programmierten Rampe per Funktionskode gestoppt und das Gerät mit der schnellsten Abwärtsrampe (größtmögliche Steilheit) auf Null gefahren.

Bit 12: Rampengenerator wartet auf die Programmierung der Sollwerte.

Bit 13: Rampengenerator arbeitet die Verrundung oder die Rampe ab.

Bit 14: Rampengenerator wartet auf den Trigger (HW oder SW).

3.5 Interlock Interrupt

Interlocks werden gepollt.

3.6 Data Request (DRQ) Interrupts

DRQ-Interrupts werden nicht erwartet bzw. verarbeitet.

3.7 Data Ready (DRD) Interrupts

DRD-Interrupts werden nicht erwartet bzw. verarbeitet.

3.8 Funktionscodes der Interfacekarte

Funktionscode		Modus	Bedeutung
Name	Hex		
ifb_reset	01	Funktion	Reset
ifb_power_on	02	Funktion	Netz einschalten
ifb_power_off	03	Funktion	Netz ausschalten
ifb_soll_1	06	Schreiben	Decrement
ifb_soll_2	07	Schreiben	Delay
ifb_soll_3	08	Schreiben	Flattop-Strom
ifb_intr_mask	12	Schreiben	Interruptmaske
ifb_dev_fct_13	20	Funktion	Rampenstart via Software triggern
ifb_dev_fct_14	21	Funktion	Software-Trigger enable
ifb_dev_fct_15	22	Funktion	Software-Trigger disable
ifb_wr_ifa_mode	60	Funktion	Rampengenerator-Modus einschalten
ifb_ist_1	81	Lesen	Strom bei Prep_Beam_On
ifb_ist_2	82	Lesen	Strom bei Beam_Off
ifb_dev_data_1	91	Lesen	Status des Rampengenerators
ifb_rd_ifa_mode	97	Lesen	Rampengenerator-Modus
ifb_rdstat	C0	Lesen	Gerätestatus, Bits 8 bis 15
ifb_rdstat_1	C1	Lesen	Gerätestatus, Bits 16 bis 23
ifb_rdstat_2	C2	Lesen	Gerätestatus, Bits 24 bis 31
ifb_rdstat_int	C9	Lesen	Status der Interfacekarte
ifb_rdstat_int_3	CC	Lesen	Version der Interfacekarte

4 Die Ansteuerung des Gerätes

Hier wird beschrieben, wie das Gerät (die Hardware) angesteuert werden muss. Das beinhaltet die Anforderungen *vom* Gerät als auch die Anforderungen *an* das Gerät.

4.1 Der Rampengenerator

Im Folgenden wird zwischen *Sollwert* und *Vorgabewert* unterschieden. Die Begriffe sind wie folgt definiert:

Sollwert: Die Sollwerte sind die Werte, die vom Operating via Property an das Gerät, also

an die VME-Gerätesoftware (USRs) geschickt werden.

Vorgabewert: Die Vorgabewerte werden von der Gerätesoftware (USRs) aus den Sollwerten berechnet, im Dualport-RAM abgelegt und vom EQM an das Geräte-Interface, den Rampengenerator, geschickt.

4.1.1 Eigenschaften des Rampengenerators

Der Rampengenerator wird mit drei Vorgabewerten versorgt. Sind diese programmiert, stellt er den Flattopwert am Gerät ein und wartet auf einen Trigger. Ab dem Trigger läuft die eingestellte Startverzögerung. Nach Ablauf dieser Verzögerungszeit wird die Verrundung und anschließend die Rampe generiert. Die Generierung ist beendet, wenn der Wert für die Rampe gleich Null ist.

Verrundung und Rampe werden erzeugt, indem der Rampengenerator periodisch neu berechnete Werte an den DAC schickt. Der Algorithmus zur Berechnung dieser sogenannten Stützpunkte ist in Kapitel 4.1.2 auf Seite 16 erklärt.

Der Rampengenerator ist Teil der Funktionalität der Interface-Karte, steht aber nicht automatisch zur Verfügung, sondern muss explizit eingeschaltet werden (mittels Funktionskode).

Die drei zur Generierung der Rampe notwendigen Vorgabewerte sind

1. die Höhe des Flattop-Stromes,
2. die Stromdifferenz, die pro Schritt (Stützpunkt) vom momentanen Strom abzuziehen ist und
3. die Startverzögerung.

Die Vorgabewerte müssen in der Reihenfolge

1. Stromdifferenz,
2. Startverzögerung,
3. Flattop-Strom,

programmiert werden.

Der Rampengenerator arbeitet intern mit einer Quarzfrequenz von 12 MHz.

Die Stützpunkte für Verrundung und Rampe werden mit der halben Quarzfrequenz generiert.

Die Anzahl der Verrundungsschritte - der Stützpunkte - zwischen Flattop und linearer Rampe beträgt 64. Dieser Wert ist konstant (also unabhängig von der Steilheit der Rampe).

Der Rampengenerator wird mit einem externen Trigger gestartet. Der Start kann verzögert werden. Die Startverzögerung wird mit der vollen Quarzfrequenz generiert.

Die Rampe muss 3,0 ms nach dem Programmieren des Flattop-Stromes gestartet sein. Ist dies nicht der Fall, weil kein Trigger gekommen ist, generiert der Rampengenerator einen Timeout-Fehler und fährt mit der grössten Rampensteilheit auf Null. Läuft die Timeout-Zeit während der Rampe ab, wird *kein* Timeout-Fehler generiert. Die Rampe wird zu Ende gefahren.

Eine Rampe mit der Steilheit Null wird gefahren, wenn der Vorgabewert für die Stromdifferenz gleich Null ist. Dann bleibt der Flattopwert bis zum Timeout erhalten. In diesem Fall wird *kein* Timeout-Fehler generiert.

4.1.2 Berechnung der Vorgabewerte

Vom Operating werden die Sollwerte Flattop-Feld, Rampendauer und Startverzögerung geschickt. Daraus und aus den Eigenschaften des Rampengenerators müssen die Vorgaben für das Interface berechnet werden.

Es bedeuten:

B_{ft}	Das Flattop-Feld vom Operating.
I_{ft}	Der Flattop-Strom, der via Polynom aus B_{ft} berechnet wird und normiert direkt an den Rampengenerator geschickt werden kann.
t_{ft}	Die Rampendauer vom Operating inklusive Verrundungszeit.
t_{D}	Die Startverzögerung zwischen Trigger und Start der Rampe, ebenfalls vom Operating.
ΔI_{step}	Die Stromdifferenz, die pro Schritt des Rampengenerators vom momentanen Strom abgezogen werden soll.
f_{Q}	Die interne Quarzfrequenz des Rampengenerators.
n_{rund}	Die Anzahl der Verrundungsschritte des Rampengenerators (und gleichzeitig der Kehrwert des Faktors, mit dem ΔI_{step} entwertet wird, um den ersten Verrundungsschritt zu generieren).

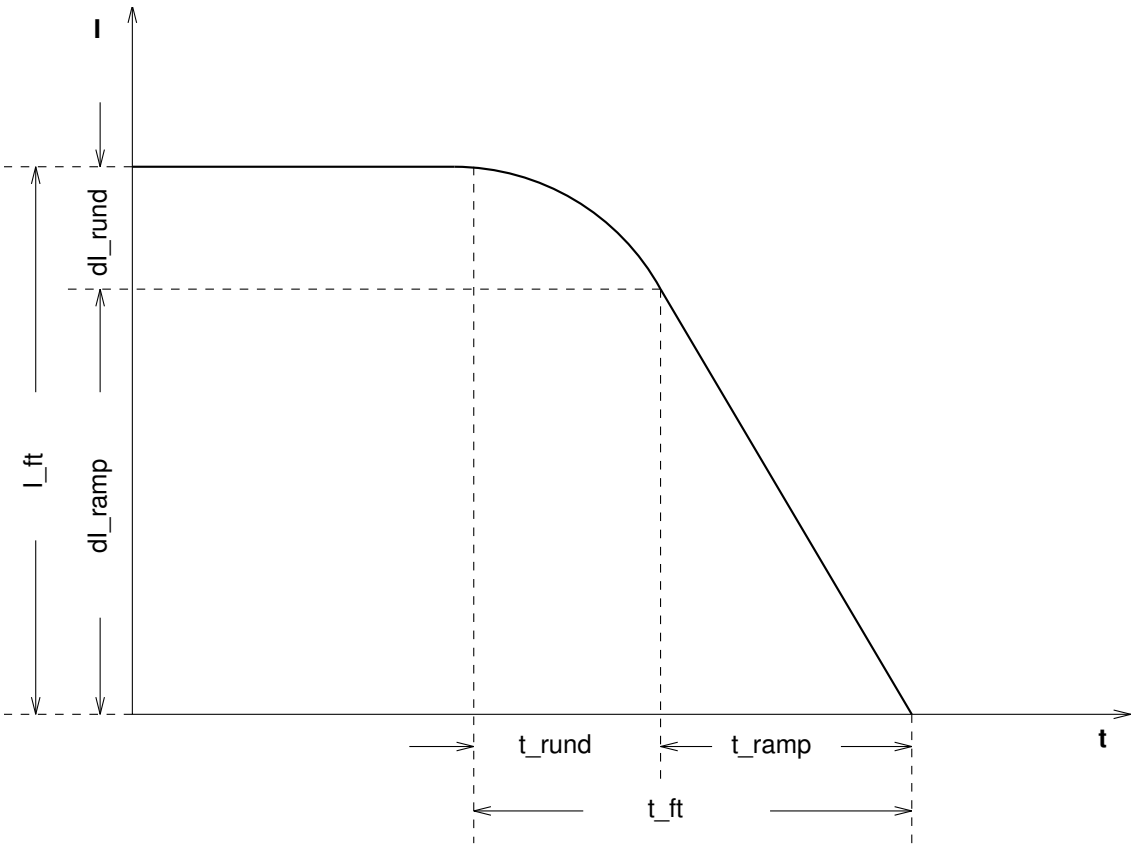

Abbildung 7: Verrundung der Rampe

Aus Abbildung 7 ergeben sich weitere Elemente:

t_{rund} Die Verrundungszeit.

t_{ramp} Die Zeit der linearen Rampe.

ΔI_{rund} Die Stromabnahme während der Verrundungszeit.

ΔI_{ramp} Die Stromabnahme während der linearen Rampe.

Da die Stützpunkte mit der halben Quarzfrequenz generiert werden, ist der Stützpunktabstand damit

$$T = \frac{2}{f_{\mathsf{Q}}}$$

und die Anzahl der Stützpunkte innerhalb t_{ramp}

$$n_{\mathsf{ramp}} = \frac{t_{\mathsf{ramp}}}{T}$$

Die Anzahl der Rundungsschritte ist konstant. Damit ergibt sich die Rundungszeit zu

$$t_{\mathsf{rund}} = n_{\mathsf{rund}} T = 10.667\,\mu\mathsf{s}$$

Die Hardware generiert die Rundung folgendermaßen: ΔI_{step} wird um den Faktor $1/n_{\mathsf{rund}}$ entwertet. Dieser Wert wird beim ersten Stützpunkt von I_{ft} abgezogen. Beim zweiten Stützpunkt wird der doppelte, beim dritten der dreifache Wert und beim i-ten der Wert $i(\Delta I_{\mathsf{step}}/n_{\mathsf{rund}})$ vom momentanen Strom abgezogen. Nach n_{rund} Stützpunkten ist die Kurve so glatt wie möglich in die lineare Rampe übergegangen.

Die gesamte Stromabnahme während der Verrundungszeit ergibt sich daraus zu

$$\begin{aligned}
\Delta I_{\mathsf{rund}} &= \frac{\Delta I_{\mathsf{step}}}{n_{\mathsf{rund}}} + 2\frac{\Delta I_{\mathsf{step}}}{n_{\mathsf{rund}}} + 3\frac{\Delta I_{\mathsf{step}}}{n_{\mathsf{rund}}} + \cdots + n_{\mathsf{rund}}\frac{\Delta I_{\mathsf{step}}}{n_{\mathsf{rund}}} \\
\Delta I_{\mathsf{rund}} &= \frac{\Delta I_{\mathsf{step}}}{n_{\mathsf{rund}}}(1 + 2 + \cdots + n_{\mathsf{rund}}) \\
\Delta I_{\mathsf{rund}} &= \frac{\Delta I_{\mathsf{step}}}{n_{\mathsf{rund}}}\frac{n_{\mathsf{rund}}}{2}(n_{\mathsf{rund}} + 1) \\
\Delta I_{\mathsf{rund}} &= \Delta I_{\mathsf{step}}\frac{n_{\mathsf{rund}} + 1}{2} \qquad (1)
\end{aligned}$$

Für die Stromabnahme zwischen zwei Stützpunkten während der linearen Rampe gilt

$$\begin{aligned}
\Delta I_{\mathsf{step}} &= \frac{\Delta I_{\mathsf{ramp}}}{n_{\mathsf{ramp}}} \\
\Delta I_{\mathsf{step}} &= \Delta I_{\mathsf{ramp}}\frac{T}{t_{\mathsf{ramp}}} \\
\Delta I_{\mathsf{step}} &= (I_{\mathsf{ft}} - \Delta I_{\mathsf{rund}})\frac{T}{t_{\mathsf{ramp}}} \qquad (2)
\end{aligned}$$

Gleichung 1 in 2 eingesetzt und einige Umstellungen ergeben die Stromdifferenz ΔI_{step} zwischen zwei Stützpunkten.

$$\Delta I_{\mathsf{step}} = (I_{\mathsf{ft}} - \Delta I_{\mathsf{step}}\frac{n_{\mathsf{rund}} + 1}{2})\frac{T}{t_{\mathsf{ramp}}}$$

$$
\begin{aligned}
\Delta I_{\mathsf{step}} \frac{t_{\mathsf{ramp}}}{T} &= I_{\mathsf{ft}} - \Delta I_{\mathsf{step}} \frac{n_{\mathsf{rund}} + 1}{2} \\
\Delta I_{\mathsf{step}} (\frac{t_{\mathsf{ramp}}}{T} + \frac{n_{\mathsf{rund}} + 1}{2}) &= I_{\mathsf{ft}} \\
\Delta I_{\mathsf{step}} &= I_{\mathsf{ft}} (\frac{t_{\mathsf{ramp}}}{T} + \frac{n_{\mathsf{rund}} + 1}{2})^{-1} \\
\Delta I_{\mathsf{step}} &= I_{\mathsf{ft}} (\frac{t_{\mathsf{ft}} - t_{\mathsf{rund}}}{T} + \frac{n_{\mathsf{rund}} + 1}{2})^{-1} \\
\Delta I_{\mathsf{step}} &= I_{\mathsf{ft}} (\frac{t_{\mathsf{ft}} - n_{\mathsf{rund}} T}{T} + \frac{n_{\mathsf{rund}} + 1}{2})^{-1} \\
\Delta I_{\mathsf{step}} &= I_{\mathsf{ft}} (\frac{t_{\mathsf{ft}}}{T} - n_{\mathsf{rund}} + \frac{n_{\mathsf{rund}} + 1}{2})^{-1} \\
\Delta I_{\mathsf{step}} &= I_{\mathsf{ft}} (\frac{f_{\mathsf{Q}}}{2} t_{\mathsf{ft}} + \frac{1 - n_{\mathsf{rund}}}{2})^{-1} \qquad (3)
\end{aligned}
$$

Die Konstanten $f_{\mathsf{Q}} = 12\,\mathsf{MHz}$ und $n_{\mathsf{rund}} = 64$ eingesetzt ergibt

$$\Delta I_{\mathsf{step}} = I_{\mathsf{ft}} (\frac{6,0}{\mu s} t_{\mathsf{ft}} - 31,5)^{-1}$$

4.1.3 Normierung der Vorgabewerte

Die Vorgabewerte für den Rampengenerator haben folgende Kenngrößen:

Flattop:	16 Bits Breite, vorzeichenbehaftet. Der 12 Bits breite DAC ist mit den oberen 12 Bits des Flattop-Wertes verbunden.
Stromdifferenz:	12 Bits Breite, vorzeichenlos.
Startverzögerung:	12 Bits Breite, vorzeichenlos.

Das interne Rechenwerk ist 21 Bits breit und vorzeichenbehaftet.

Die Zuordnung zwischen Flattop (`I_ft`), Stromdifferenz (`I_step`) und internem Rechenwerk ist in Abbildung 8 dargestellt.

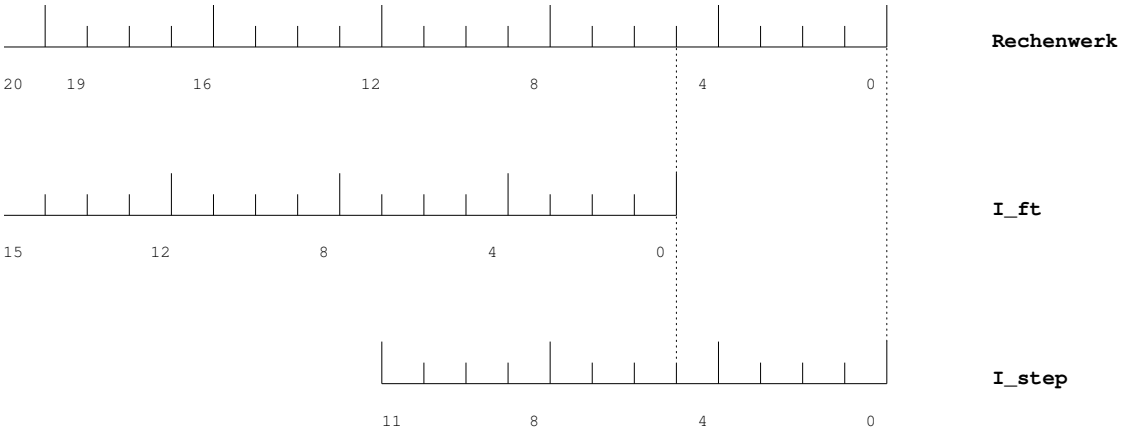

Abbildung 8: Vorgabewerte und Rechenwerk

Der Flattop-Wert wird in die oberen 16 Bits des Rechenwerks geladen. Die unteren 5 Bits werden vom Interface (EPLD) zu Null gesetzt. Daraus ergibt sich, dass man für die Rechnung den Flattop-Strom zunächst wie folgt normieren kann:

$$\mathsf{FFFE0_{hex}} \widehat{=} I_{\mathsf{Nenn}}$$

Mit diesem Wert bleibt das oberste Bit im Rechenwerk Null und der Vorgabewert positiv. Da nur die obersten 16 Bits an das Gerät geschickt werden können, muss der binäre Wert noch um 5 Bits, also 2^5, entwertet werden. Die Normierung von I_{ft} zur Programmierung des Interfaces ergibt damit den binären Wert

$$I_{\mathsf{ft,bin}} = 2^{-5} I_{\mathsf{ft}} \frac{\mathsf{FFFE0_{hex}}}{I_{\mathsf{Nenn}}}$$

Für die Stromdifferenz stehen 12 Bits (ohne Vorzeichen) zur Verfügung, die die gleiche Wertigkeit haben wie die entsprechenden Bits des Rechenwerkes - die jeweiligen Bits 0 bis 11 liegen „übereinander“. Damit kann ΔI_{step} genau so normiert werden wie der Flattop-Strom.

$$\Delta I_{\mathsf{step,bin}} = \Delta I_{\mathsf{step}} \frac{\mathsf{FFFE0_{hex}}}{I_{\mathsf{Nenn}}}$$

Da die beiden Werte direkt vergleichbar sind, kann man übrigens die maximale Steigung der Rampe leicht ausrechnen. Die maximal mögliche Stromdifferenz zwischen zwei Stützpunkten beträgt

$$\Delta I_{\mathsf{step,max}} = \frac{\mathsf{FFF_{hex}}}{\mathsf{FFFE0_{hex}}} I_{Nenn}$$

Das bedeutet für einen Nennstrom von 3000 A einen maximalen Steigungswert von

$$\begin{aligned} \dot{I}_{\mathsf{max}} &= \frac{1}{2} f_{\mathsf{Q}} \Delta I_{\mathsf{step,max}} \\ \dot{I}_{\mathsf{max}} &\simeq 70\,\mathsf{kA}/\mu\mathsf{s} \end{aligned} \tag{4}$$

Die Startverzögerung t_{D} wird, im Gegensatz zum Stützpunktabstand, direkt mit der Quarzfrequenz f_{Q} generiert. Es stehen 12 Bits (ohne Vorzeichen) zur Verfügung. Die Normierung von t_{D} zur Programmierung des Interfaces ergibt damit den binären Wert

$$t_{\mathsf{D,bin}} = t_{\mathsf{D}} f_{\mathsf{Q}}$$

4.1.4 Rückrechnung der Vorgabewerte

Aus den Vorgabewerten lassen sich durch Umstellen obiger Formeln die Sollwerte zurückrechnen. Der Flattop-Strom (entsprechend um 2^5 aufgewertet):

$$I_{\mathsf{ft}} = 2^5 I_{\mathsf{ft,bin}} \frac{I_{\mathsf{Nenn}}}{\mathsf{FFFE0_{hex}}}$$

Die Stromdifferenz und daraus die Rampendauer:

$$\Delta I_{\mathsf{step}} = \Delta I_{\mathsf{step,bin}} \frac{I_{\mathsf{Nenn}}}{\mathsf{FFFE0_{hex}}}$$

$$t_{\mathsf{ft}} = \frac{2}{f_{\mathsf{Q}}} \left(\frac{I_{\mathsf{ft}}}{\Delta I_{\mathsf{step}}} - \frac{1 - n_{\mathsf{rund}}}{2} \right)$$

Und die Startverzögerung:

$$t_{\mathsf{D}} = \frac{t_{\mathsf{D,bin}}}{f_{\mathsf{Q}}}$$

4.1.5 Fehlerbetrachtung

Die Entwertung der Stromdifferenz zur Generierung der Verrundung führt nicht zum Abschneiden der unteren 6 Bits. Das interne Addierwerk ist tief genug, um die geshifteten Bits aufzunehmen.

Hier ausrechnen, wie groß der Fehler ist a) allgemein und b) bei reellen Werten. tbs

4.2 Sollwerte

Die Geräte-Software erhält drei Sollwerte.

Der Wert für das Flattop-Feld wird über ein Polynom dritten Grades in den Flattopstrom umgerechnet. Die Minimal- und Maximalwerte für den Flattop-Strom betragen:

$$0 \leq I_{\mathsf{ft,TK2MW1}} \leq 3000\,A$$

$$0 \leq I_{\mathsf{ft,TK3MW2}} \leq 1500\,A$$

Die Rampendauer kann im Bereich

$$t_{\mathsf{ft}} = 0$$

$$120\,\mu\mathsf{s} \leq t_{\mathsf{ft}} \leq 1\,\mathsf{ms}$$

eingestellt werden.

Die Startverzögerung kann im Bereich

$$0\,\mu\mathsf{s} \leq t_{\mathsf{D}} <= 340\,\mu\mathsf{s}$$

eingestellt werden. Der genaue Maximalwert ergibt sich zu

$$t_{\mathsf{D,max}} = \frac{\mathsf{FFF}_{\mathsf{hex}}}{f_{\mathsf{Q}}}$$

Werte außerhalb dieser Bereiche werden von der Gerätesoftware abgewiesen.

4.3 Istwerte

Die Geräte-Software liefert zwei Istwerte: Das Feld des Gerätes, gemessen zum Zeitpunkt des Events `Prep_Beam_On` sowie das Feld, gemessen zum Zeitpunkt des Events `Beam_Off` - beide Felder in der Einheit Teslameter. Die Felder werden mittels Polynom dritten Grades aus dem Strom berechnet.

Die Istwerte werden mit Hardware-Triggern, die von `Prep_Beam_On` und `Beam_Off` abgeleitet werden, in die Interface-Hardware eingelatched. Diese Trigger können nicht überwacht werden. Es ist nicht sichergestellt, dass der gelesene (gelatchte) Istwert aus dem letzten virtuellen Beschleuniger stammt.

4.4 Einschalten

Folgende Sequenzen zum Einschalten beachten:

1. Power einschalten. Die notwendige Einschaltpulslänge von 200 ms wird vom Interface garantiert.
2. Nach 2 s prüfen, ob eingeschaltet ist.

4.5 Ausschalten

Folgende Sequenzen zum Ausschalten beachten:

1. Power ausschalten. Die notwendige Ausschaltpulslänge von 200 ms wird vom Interface garantiert.
2. Nach 2 s prüfen, ob ausgeschaltet ist.

4.6 Genauigkeitsanforderungen

?

Wie hoch sind die Genauigkeitsanforderungen bezüglich des Gleichlaufes der beiden Sweeper? Also um wieviel dürfen die Rampen voneinander abweichen?

Die Genauigkeitsanforderungen bezüglich des Gleichlaufes der beiden Sweeper muss ≤ 1 Promille betragen[1]. Das entspricht bei einer minimalen Rampendauer von 120 μs einer Zeit von ≤ 120 ns. Die interne Quarzfrequenz des Rampengenerators von 12 MHz ist dafür ausreichend. Das EPLD taktet zwar nur mit der halben Quarzfrequenz, der einlaufende Trigger wird aber mit der vollen Taktfrequenz synchronisiert. Der Jitter zwischen den Rampengeneratoren kann also maximal $\pm 83{,}3$ ns betragen, was einer Gleichlaufgenauigkeit von 0,694 Promille entspricht.

Die Eventdekoder in den Sweeper-Netzgeräten jittern mit etwa 600 ns. Das ist nicht hinreichend genau. Der Starttrigger muss für *beide* Sweeper aus *einem* Dekoder abgeleitet werden.

4.7 Zeitkritische Anforderungen

4.7.1 Sollwerte

Der Rampengenerator muss mit einem Trigger, der hardware-mäßig (mittels TIF) von Event `Prep_Beam_On` abgeleitet wird, gestartet werden, damit die Synchronität der beiden Sweeper gewährleistet ist. Ein Software-Trigger ist zu ungenau.

4.7.2 Istwerte

?

Die A/D-Wandler brauchen zur Konvertierung des Messwertes ca. 50 μs! Das kann schon die Hälfte der Strahlpulslänge sein. Entsprechend ungenau ist der vom ADC gelieferte Istwert. Was tun?

4.8 Einordnung in das Timing

Die Sweeper müssen ca. 2 ms vor dem Strahlpuls auf ihren Flattop-Wert fahren, um rechtzeitig eingeschwungen zu sein. Dazu wird das EQM zum Setzen der Vorgabewerte mit entsprechender

[1] Laut J. Glatz, 16. Nov. 1999

Verzögerung vom Event `Ready_to_SIS` gestartet. Dieses hat einen konstanten Abstand von 10 ms zum Event `Beam_On`, das den Beginn des Srahlpulses markiert.

Der Start des Rampengenerators erfolgt hardware-mäßig mit dem Event `Prep_Beam_On`.

Die Istwerte werden hardware-mäßig mit den Events `Prep_Beam_On` und `Beam_Off` von den ADCs eingelatched.

Das EQM zum Lesen der Istwerte und des dynamischen Status des Rampengenerators wird mit dem Event `Beam_Off` gestartet.

4.9 Festlegung von Startwerten

4.9.1 Kaltstarts

Bei einem Kaltstart werden folgende Aktionen durchgeführt:

- Es wird ein Gerätereset durchgeführt.
- Der Rampengenerator auf der IFK wird enabled.
- Die Gerätekonstanten in Dualport-RAM werden initialisiert.
- Alle Vorgabewerte werden für alle virtuellen Beschleuniger auf Null gesetzt.
- Die Istwerte, sowie Stamps und EFICD-Information werden auf Null gesetzt.
- Das Gerät wird für alle virtuellen Beschleuniger inaktiv geschaltet.
- Die Interlockbehandlung wird aktiviert.
- Die SE wird in den Eventmode-Betrieb geschaltet.
- Die Standard-Eventkonnektierungen werden gesetzt.

4.9.2 Warmstarts

Bei einem Warmstart werden folgende Aktionen durchgeführt:

- Es wird ein Gerätereset durchgeführt.
- Der Rampengenerator auf der IFK wird enabled.
- Die Gerätekonstanten in Dualport-RAM werden initialisiert.
- Die Istwerte, sowie Stamps und EFICD-Information werden auf Null gesetzt.
- Die Interlockbehandlung wird aktiviert.

4.10 Handbetrieb

Das Gerät kann auf Handbetrieb (local) geschaltet werden. Der Zustand wird im Status angezeigt.

4.11 Verhalten bei Störungen

4.11.1 Geräteinterlock

Bei einem Interlock geht die Gerätesoftware in den internen Zustand `dev_interlock` und liest den aktuellen Gerätestatus. Weitere Aktionen sind nicht notwendig.

4.11.2 Event-Sequenzfehler

Event-Sequenzfehler werden gemeldet, der aktuelle Zyklus aber nicht abgebrochen.

4.11.3 Event-Overrun

Bei Overruns kann das erforderliche Timing nicht garantiert werden. Der Zyklus wird abgebrochen.

4.11.4 Emergency-Event

Im Emergency-Fall geht die Gerätesoftware in den internen Zustand `emergency` und liest den aktuellen Gerätestatus. Weitere Aktionen sind nicht notwendig.

4.11.5 Ausfall der Kommunikation EC – Gerät

Der Ausfall der Kommunikation zwischen EC und Gerät führt zu Timeouts. Tritt ein Timeout auf, wird das EQM beendet. Tritt der Timeout im Sollwert-EQM auf, wird im Istwert-EQM trotzdem versucht, die Istwerte zu lesen.

4.12 Bedienungsfehler vom Operating

Sollwerte außerhalb des zulässigen Bereichs werden zurückgewiesen.

Inkonsistente Sollwert-Kombinationen können auftreten, wenn für einen sehr kleinen Flattop-Wert eine lange Rampendauer eingestellt werden soll. Der Rampengenerator ist dann nicht in der Lage, die erforderliche flache Rampe zu generieren. In diesem Fall wird der Rampengenerator so programmiert, dass *keine* Rampe generiert wird. Der Flattop-Wert bleibt bis zum Timeout des Rampengenerators erhalten.

5 Therapiebetrieb

Im Therapiebetrieb darf das Gerät ein oder aus sein. Ist es ein, dann muss es inaktiv sein.

Beim ECC `Prep_Med` wird das Gerät inaktiv gesetzt.

6 Die Repräsentation des Gerätes

Dieses Kapitel beschreibt, wie das Gerät nach höheren Ebenen hin abgebildet wird.

6.1 Kennzeichnung des Gerätemodells

Das Gerätemodell hat die Bezeichnung **MS**.

Die Gerätemodellnummer ist 59_{dez}.

6.2 Kompatibilität mit MX und MD

Damit die Sweeper wie ein Magnet bedient werden können, sind die Magnet-Properties `FIELDS`, `FIELDI`, `CURRENTS`, `CURRENTI`, `VOLTS`, `VOLTI`, `MAGNINFO` und `CALC` implementiert. Diese Properties beziehen sich hier jeweils nur auf den Flattop-Wert des Sweepers. Startverzögerung und Rampendauer bleiben unbeeinflusst.

Aus Konsistenzgründen werden zudem die Properties `DELTA` und `RAMPTIME` zur Verfügung gestellt, damit auch die Sollwerte Startverzögerung und Rampendauer einzeln gesetzt bzw. gelesen werden können. Für Startverzögerung und Rampendauer gibt es keine Istwerte.

Beim Setzen von einzelnen Sollwerten können inkonsistente Sollwert-Kombinationen auftreten. Siehe dazu Kapitel 4.12.

Beim Lesen der Sollwerte werden diese aus den Vorgabewerten für den Rampengenerator *berechnet*. Es werden *nicht* die im Dualport-RAM liegenden sog. originalen Sollwerte geliefert.

6.3 Die Master-Properties

In der Tabelle angegeben sind Name und Klasse der Property, Anzahl und Typ der Parameter, Anzahl und Typ der Daten sowie die physikalisch technische Einheit der Daten und der zugehörige Exponent zur Basis 10.

INFOSTAT, INIT, POWER, RESET, STATUS und VERSION sind obligatorische Properties, die jedes Gerätemodell hat.

Property	Klasse	Parameter		Daten		Größe	
		Anz.	Typ	Anz.	Typ	Einh.	Exp.
`INFOSTAT`	RA	0	–	25	BitSet32	1	0
`INIT`	N	0	–	0	–	–	–
`POWER`	R/W	0	–	1	BitSet16	1	0
`RESET`	N	0	–	0	–	–	–
`STATUS`	R	0	–	1	BitSet32	1	0
`VERSION`	RA	0	–	36	BitSet8	1	0
`CALC`	RA	2	Structure	3	Structure	div.	div.
`CONSTANT`	RA	0	–	62	RealF	div.	div.

6.3.1 INFOSTAT

Bedeutung: Diese Property liefert einige wichtige Geräteinformationen in einem Zugriff. Die Informationen werden direkt aus dem Dualport-RAM gelesen, also ohne den expliziten Aufruf eines EQMs, und sind daher in der Abarbeitung nicht abhängig von Kommandoevents.

Parameter: Keine.

Daten: Die 25 Langworte enthalten im einzelnen:

1: Gerätestatus (wie in der Property STATUS)

2: Gibt in den oberen 16 Bits an, welcher virtuelle Beschleuniger aktiv gesetzt ist (ein Bit pro Beschleuniger). Das niederwertigste Bit (Bit 16) gibt den Beschleuniger 15 an, das Bit 31 den Beschleuniger 0. Die unteren 16 Bit sind nicht verwendet. Dabei bedeutet Null, dass der Beschleuniger inaktiv ist und Eins, dass der Beschleuniger aktiv ist.

3: Master-Fehler. Hier ist derjenige Master-Gerätefehlercode mit dem schwersten Fehlergrad eingetragen. Bei mehreren Fehlern mit dem gleichen Fehlergrad wird der erste eingetragen, der gefunden wurde.

4: Slave-Fehler für virtuellen Beschleuniger 0. Entsprechend dem Master-Fehler wird hier der nach dem Fehlergrad schwerste Slave-Gerätefehlercode für den Beschleuniger 0 eingetragen.

5: Entsprechend Punkt 4, aber für virtuellen Beschleuniger 1.

⋮

19: Entsprechend Punkt 4, aber für virtuellen Beschleuniger 15.

20: EC-Mode. In den oberen 16 Bit des Langwortes steht der von der Gerätesoftware eingestellte Default-EC-Mode, in den unteren 16 Bit der aktuelle EC-Mode. Folgende Modi sind definiert:

0 = not_set	Der Initwert.
1 = Preset_Command	Der ECM hat das Umschalten in Command-Mode vorbereitet aber noch nicht beendet.
2 = Command	Der ECM läuft im Command-Mode.
3 = Preset_Event	Der ECM hat das Umschalten in Event-Mode vorbereitet aber noch nicht beendet.
4 = Event	Der ECM läuft im Event-Mode.

21: EC-Performance-Mode. In den oberen 16 Bit des Langwortes steht der von der Gerätesoftware eingestellte Default-Performance-Mode, in den unteren 16 Bit der aktuelle Performance-Mode. Folgende Modi sind definiert:

0 = not_set	Initwert.
1 = Display	Der ECM läuft im Display-Mode.
2 = Preset_Turbo	Der ECM hat das Umschalten in den Turbo-Mode vorbereitet aber noch nicht beendet.
3 = Turbo	Der ECM läuft im Turbo-Mode.

22: HW-Warning-Maske. Die 32 Bits geben an aus welchen Bits im Gerätestatus das HW-Warning-Bit im Status abgeleitet wird.

23: Pulszentralen-Identifikation:

0:	TIF
1:	SIS
2:	ESR
3...6:	undefiniert
7:	Software-Pulszentrale
8:	UNILAC Master-PZ
9:	UNILAC Quelle rechts
10:	UNILAC Quelle links
11:	UNILAC Quelle HLI
12:	UNILAC HLI
13:	UNILAC RFQ und IH
14:	UNILAC Alvarez
15:	UNILAC Transferkanal

24: Reserviert für Erweiterungen.

25: Reserviert für Erweiterungen.

6.3.2 INIT

Bedeutung: Initialisierung des Gerätes (Kaltstart). Für die dabei durchzuführenden Aktionen siehe Kapitel 4.9.1 auf Seite 21.

Parameter: Keine.

Daten: Keine.

6.3.3 POWER

Bedeutung: Gibt an, ob der Leistungsteil des Gerätes ein- oder ausgeschaltet ist bzw. werden soll. Hat bei diesem Gerät keine reale Bedeutung und ist nur aus Kompatibilitätsgründen vorhanden.

Parameter: Keine.

Daten: Beim Lesen immer 1, d. h. Gerät ist eingeschaltet. Jeder Schreibzugriff wird mit einer Fehlermeldung abgewiesen.

6.3.4 RESET

Bedeutung: Reset des Gerätes (Warmstart). Für die dabei durchzuführenden Aktionen siehe Kapitel 4.9.2 auf Seite 21.

Parameter: Keine.

Daten: Keine.

6.3.5 STATUS

Bedeutung: Auslesen des Gerätestatus.

Parameter: Keine.

Daten: Das 32 Bits breiten Statuswort. Die Bits entsprechen den Statusbits, wie sie in Kapitel 3.3 auf Seite 11 und in der Tabelle 1 auf Seite 11 erklärt sind.

6.3.6 VERSION

Bedeutung: Lesen der Versionskennung der Gerätesoftware.

Parameter: Keine.

Daten: Versionskennung als ASCII-String, pro Datum ein ASCII-Zeichen.

Bytes	Inhalt
1...12	Version der USRs
13...24	Version der EQMs
25...36	Version des Standard-MIL-Treibers
37...48	Variante der EQMs

6.3.7 CALC

Bedeutung: Feld, Strom und Spannung aus gegebenem Feld, Strom oder Spannung berechnen. Dies ist eine reine Rechen-Property. Sie beeinflusst in keiner Weise das Gerät.

Parameter: Die 2 Parameter haben folgende Struktur:

Vorgabetyp: Integer32
Vorgabe: RealF

Der Vorgabetyp kann 3 Werte annehmen:

Vorgabetyp = 1 ⇒ Vorgabe ist Feld in Tm
Vorgabetyp = 2 ⇒ Vorgabe ist Strom in A
Vorgabetyp = 3 ⇒ Vorgabe ist Spannung in mV

Daten: Die zurück gelieferten Daten haben folgende Struktur und Bedeutung:

Komponente	Einheit	Format
Feld	Tm	RealF
Strom	A	RealF
Spannung	mV	Integer32

6.3.8 CONSTANT

Bedeutung: Lesen der gerätespezifischen Konstanten.

Parameter: Keine.

Daten: Die 62 Realwerte bedeuten im einzelnen:

1: Minimal einstellbarer Wert des Stroms in Ampère.
2: Maximal einstellbarer Wert des Stroms in Ampère.
3: Minimal einstellbare Rampenzeit in μs.
4: Maximal einstellbare Rampenzeit in μs.
5: Minimale Rampensteilheit in A/μs.
6: Maximale Rampensteilheit in A/μs.
7: Minimale Startverzögerung in μs.
8: Maximale Startverzögerung in μs.
9...26: Struktur zur Beschreibung der Polynome zur Berechnung von $Bl(I)$.
27...44: Struktur zur Beschreibung der Polynome zur Berechnung von $I(Bl)$.
45...62: Struktur zur Beschreibung der Polynome zur Berechnung von $Bl(U)$.

Eine Struktur zur Beschreibung der Polynome besteht aus drei gleichen Substrukturen:

1...6: Substruktur 1.
7...12: Substruktur 2.
12...18: Substruktur 3.

Eine Substruktur beschreibt jeweils *ein* Polynom dritten Grades mit dessen Gültigkeitsbereich:

1: Anfang des Gültigkeitsbereichs.
2: Ende des Gültigkeitsbereichs.
3: Polynomkoeffizient $a0$.
4: Polynomkoeffizient $a1$.
5: Polynomkoeffizient $a2$.
6: Polynomkoeffizient $a3$.

Wobei ein Polynom die Form ($y = a_3x^3 + a_2x^2 + a_1x + a_0$) hat.

Bemerkung: Die Property liefert mehr Daten als in der Konstantentabelle der lokalen Datenbasis vorhanden sind. Zusätzlich geliefert werden die aus Hardware-Konstanten des Rampengenerators berechneten Werte für die minimale und maximale Rampensteilheit sowie für die minimale und maximale Startverzögerung.

6.4 Die Slave-Properties

In der Tabelle angegeben sind Name und Klasse der Property, Anzahl und Typ der Parameter, Anzahl und Typ der Daten sowie die physikalisch technische Einheit der Daten und der zugehörige Exponent zur Basis 10.

Liefert eine Property z. B. Milliampere (mA), dann ist die physikalisch technische Einheit „A“ und der Exponent „-3“.

ACTIV, COPYSET und EQMERROR sind obligatorische Properties, die jedes Gerätemodell hat.

Property	Klasse	Parameter		Daten		Größe	
		Anz.	Typ	Anz.	Typ	Einh.	Exp.
ACTIV	R/W	0	–	1	BitSet16	1	0
COPYSET	W	0	–	1	BitSet16	1	0
EQMERROR	RA	217	Integer32	348	Integer32	1	0
CURRENTI	R	1	Integer16	1	RealF	A	0
CURRENTS	R/W	0	–	1	RealF	A	0
DELAY	R/W	0	–	1	RealF	s	-6
DYNSTAT	R	0	–	1	BitSet16	1	0
FIELDI	R	1	Integer16	1	RealF	Tm	0
FIELDS	R/W	0	–	1	RealF	Tm	0
MAGNINFO	RA	1	Integer16	80	Structure	div.	0
MEDDATAI	RA	3	BitSet32	2	RealF	Tm	0
MEDDATAS	WA	3	BitSet16	3	RealF	Tm, s	0, -6
MEDDATAS	RA	4	BitSet32	3	RealF	Tm, s	0, -6
RAMPI	RA	0	–	2	RealF	A	0
RAMPS	RA/WA	0	–	3	RealF	A, s	0, -6
RAMPTIME	R/W	0	–	1	RealF	s	-6
VOLTI	R	1	Integer16	1	Integer32	V	-3
VOLTS	R/W	0	–	1	Integer32	V	-3

6.4.1 ACTIV

Bedeutung: Gibt an, ob das Gerät für den zugehörigen virtuellen Beschleuniger an der Puls-zu-Puls-Modulation teilnehmen soll bzw teilnimmt.

Parameter: Keine.

Daten: Das Datum kann nur zwei Werte annehmen. Null heißt, das Gerät nimmt für den zugeordneten Beschleuniger *nicht* an der PPM teil bzw. soll *nicht* an der PPM teilnehmen. Eins heißt, das Gerät nimmt für den zugeordneten Beschleuniger an der PPM teil bzw. soll an der PPM teilnehmen.

6.4.2 COPYSET

Bedeutung: Kopiert alle Geräteeinstellungen (Sollwerte) eines („fremden“) virtuellen Beschleunigers in den zugehörigen („eigenen“) virtuellen Beschleuniger.

Parameter: Keine.

Daten: Nummer des („fremden“) virtuellen Beschleunigers, von dem die Einstellungen (Sollwerte) kopiert werden sollen.

Bemerkung: Die Nummer des zugehörigen („eigenen“) virtuellen Beschleunigers, in den die Geräteeinstellungen kopiert werden sollen, wird dem Standard entsprechend im XSR-Header geliefert.

6.4.3 EQMERROR

Bedeutung: Fehlermeldungen der auf der SE installierten Gerätesoftware. Es werden die aktuellen Fehlermeldungen sowohl für die Masterfehler als auch für die Slavefehler der Geräteebene geliefert. Dazu wird auch der Inhalt des Fehlerpuffers zurückgegeben, in dem die letzten aufgetretenen Fehler abgespeichert wurden.

Parameter: Hier hat nur der erste der 217 Parameter eine Bedeutung.

1: Wird bei konnektierten Aufträgen ausgewertet. 0: Es wird bei jeder Ausführung des Auftrages eine Antwort verschickt. 1: Es wird bei jeder Ausführung des Auftrages nur dann eine Antwort verschickt, wenn sich seit dem letzten Aufruf der Inhalt der Daten geändert hat.

2...217: Dummy, sie werden vom MOPS intern verwendet und können vom Benutzer beliebig gesetzt werden.

Daten: Die Anzahl der Fehlermeldungen sei bezeichnet durch:

m Zahl der Master-Fehlermeldungen
s Zahl der Slave-Fehlermeldungen
b Größe des Fehlerpuffers

Weiterhin soll gelten:

$l = m + s$
$t = m + s + b$

Die Daten im Einzelnen:

1: In den unteren beiden Bytes sind die Anzahl der Master-Fehlermeldungen m und die Anzahl der Slave-Fehlermeldungen s angegeben:

0	0	s	m

2: erste Master-Fehlermeldung

⋮

$m+1$: letzte Master-Fehlermeldung

$m+2$: erste Slave-Fehlermeldung

⋮

$l+1$: letzte Slave-Fehlermeldung

$l+2$: Länge b des Fehlerpuffers

$l+3$: Zahl der Einträge im Fehlerpuffer

$l+4$: Index des ersten freien Platzes im Fehlerpuffer (der Fehlerpuffer ist ein Ringpuffer)

$l+5$: Erster Speicherplatz im Fehlerpuffer

⋮

$t+4$: Letzter Speicherplatz im Fehlerpuffer

6.4.4 CURRENTI

Bedeutung: Lese den Istwert eines Gerätes.

Parameter: 1 Parameter mit den möglichen Werten:

1 = Istwert lesen, der bei `Prep_Beam_On` gemessen wurde
2 = Istwert lesen, der bei `Beam_Off` gemessen wurde

Der Parameter kann weggelassen werden. In diesem Fall wird der Istwert gelesen, der bei `Prep_Beam_On` gemessen wurde.

Daten: Der gelesene Istwert in Ampère.

6.4.5 CURRENTS

Bedeutung: Setze oder lese den Sollwert eines Gerätes.

Parameter: Keine.

Daten: Der zu setzende bzw. gelesene Sollwert in Ampère.

6.4.6 DELAY

Bedeutung: Setzen oder Lesen des Sollwertes für die Startverzögerung.

Parameter: Keine.

Daten: 1 Realwert.

Bemerkung: Es gibt keinen Istwert für die Startverzögerung.

6.4.7 DYNSTAT

Bedeutung: Lesen des beschleuniger-spezifischen, dynamischen Gerätestatus.

Parameter: Keine.

Daten: 1 BitSet16

6.4.8 FIELDI

Bedeutung: Lese den Istwert eines Gerätes.

Parameter: 1 Parameter mit den möglichen Werten:

1 = Istwert lesen, der bei `Prep_Beam_On` gemessen wurde
2 = Istwert lesen, der bei `Beam_Off` gemessen wurde

Der Parameter kann weggelassen werden. In diesem Fall wird der Istwert gelesen, der bei `Prep_Beam_On` gemessen wurde.

Daten: Der gelesene Istwert in Teslameter.

6.4.9 FIELDS

Bedeutung: Setze oder lese den Sollwert eines Gerätes.

Parameter: Keine.

Daten: Der zu setzende bzw. gelesene Sollwert in Teslameter.

6.4.10 MAGNINFO

Bedeutung: Diese Property ist für spezielle Operatingprogramme designed. Sie liefert mit einem Lesezyklus verschiedene, ständig benötigte Informationen an diese Programme.

Parameter: 1 Parameter mit den möglichen Werten:

1 = Istwert liefern, der bei `Prep_Beam_On` gemessen wurde

2 = Istwert liefern, der bei `Beam_Off` gemessen wurde

Der Parameter kann weggelassen werden. In diesem Fall wird der Istwert gelesen, der bei `Prep_Beam_On` gemessen wurde.

Daten: Insgesamt 80 Bytes vom SIS-Datentyp Structure, die in 20 Elementen wie folgt strukturiert sind:

Master-Status	(BitSet32) Gerätestatus.
Dyn. Status	(BitSet32) Dynamischer Status des Rampengenerators.
Aktivzustand	(BitSet32) Aktivzustand. 0 = inaktiv, 1 = aktiv.
Polwender	(Integer32) Polwenderstellung. 1 = normal, 0 = kein Polwender, -1 = invers.
FieldIMode	(Integer32) Berechnungsmodus des Feldistwertes. 1 = via Strom, 2 = via Hallsonde, 3 = via Dummy-Polynom.
(reserviert)	(Integer32)
(reserviert)	(Integer32)
(reserviert)	(Integer32)
VoltS	(Integer32) Flattop-Sollwert in mV.
VoltI	(Integer32) Istwert in mV (siehe Parameter).
CurrentS	(Real) Sollwert in A.
CurrentI	(Real) Istwert in A (siehe Parameter).
FieldS	(Real) Sollwert in Tm.
FieldI	(Real) Istwert in Tm (siehe Parameter).
(reserviert)	(Real)
(reserviert)	(Real)
(reserviert)	(Real)
(reserviert)	(Real)
(reserviert)	(Real)
(reserviert)	(Real)

6.4.11 MEDDATAI

Bedeutung: Lesen eines Istwertsatzes aus dem Therapiebetrieb.

Parameter: Die Struktur der Parameter ist in [1] beschrieben.

Daten: Die 2 Realwerte enthalten:

1: Das Feld in Teslameter, gemessen zum Zeitpunkt des Events `Prep_Beam_On`. Dies sollte das Flattop-Feld sein.

2: Das Feld in Teslameter, gemessen zum Zeitpunkt des Events `Beam_Off`. Dies sollte das Feld am Ende der Rampe sein, in der Regel also Null Tm.

6.4.12 MEDDATAS

Bedeutung: Schreiben oder Lesen eines Sollwertsatzes für den Therapiebetrieb.

Parameter: Die Struktur der Parameter ist unterschiedlich, je nachdem ob geschrieben oder gelesen wird. Die Strukturen sind in [1] beschrieben.

Daten: Die 3 Realwerte enthalten:

1: Das Flattop-Feld in Teslameter.
2: Die Startverzögerung der Rampe in μs, gemessen ab dem Hardware-Trigger, der vom Event `Prep_Beam_On` abgeleitet wird.
3: Die Rampendauer in μs, gemessen vom Ende der Startverzögerung bis zum Nulldurchgang.

6.4.13 RAMPI

Bedeutung: Lese die beiden Feld-Istwerte. Feld wird bei den Events `Prep_Beam_On` und `Beam_Off` gemessen. Zwischen diesen Events liegt die abfallende Rampe.

Parameter: Keine.

Daten: Die 2 Realwerte enthalten:

1: Das Feld in Teslameter, gemessen zum Zeitpunkt des Events `Prep_Beam_On`. Dies sollte das Flattop-Feld sein.
2: Das Feld in Teslameter, gemessen zum Zeitpunkt des Events `Beam_Off`. Dies sollte das Feld am Ende der Rampe sein, in der Regel also Null Tm.

6.4.14 RAMPS

Bedeutung: Setze oder lese die Sollwerte für die abfallende Rampe. Die Berechnung der Vorgabewerte für die Hardware aus den Sollwerten ist in Kapitel 4.1.2 beschrieben, die Rückrechnung der Sollwerte aus den Vorgabewerten in Kapitel 4.1.4.

Parameter: Keine.

Daten: Die 3 Realwerte enthalten:

1: Das Flattop-Feld in Teslameter.
2: Die Startverzögerung der Rampe in μs, gemessen ab dem Hardware-Trigger, der vom Event `Prep_Beam_On` abgeleitet wird.
3: Die Rampendauer in μs, gemessen vom Ende der Startverzögerung bis zum Nulldurchgang.

6.4.15 RAMPTIME

Bedeutung: Setzen oder Lesen des Sollwertes für die Rampendauer. Die Zeit zählt vom eventuell verzögerten Startzeitpunkt bis zum Nulldurchgang der Rampe.

Parameter: Keine.

Daten: 1 Realwert.

Bemerkung: Es gibt keinen Istwert für die Rampendauer.

6.4.16 VOLTI

Bedeutung: Lese den Istwert eines Gerätes. Der Istwert ist für den DAC/ADC normiert auf $10000mV$.

Parameter: 1 Parameter mit den möglichen Werten:

1 = Istwert lesen, der bei `Prep_Beam_On` gemessen wurde

2 = Istwert lesen, der bei `Beam_Off` gemessen wurde

Der Parameter kann weggelassen werden. In diesem Fall wird der Istwert gelesen, der bei `Prep_Beam_On` gemessen wurde.

Daten: Der gelesene Istwert zwischen $0mV$ und $10000mV$.

6.4.17 VOLTS

Bedeutung: Setze oder lese den Sollwert eines Gerätes. Der Sollwert ist für den DAC/ADC normiert auf $10000mV$.

Parameter: Keine.

Daten: Der zu setzende bzw. gelesene Sollwert zwischen $0mV$ und $10000mV$.

Teil II
Die Gerätesoftware

7 Softwareentwurf

Dies ist noch ein sehr allgemeiner Punkt. Hier sollte unter anderem hingehören:

- Datenstrukturen,
- Datenflussdiagramme,
- Kontrollflussdiagramme
- ...

8 Lokale Datenbasis

8.1 Tabelle der Konstanten

```
TYPE
  {-----------------------------------------------------}
  { Structure of one polynom of the device constants }
  {-----------------------------------------------------}
  poly_type = RECORD
    rng_start, { range of ... }
    rng_end,   { validation }
    a0, a1, a2, a3: real; { coefficients }
    END;

  {-----------------------------------------------------}
  { Each conversion I(Bl), Bl(I) is described with    }
  { three polynoms for three different ranges.        }
  {-----------------------------------------------------}
  poly_desc_type = ARRAY [1..3] OF poly_type;
  ptr_poly_desc_type = ^poly_desc_type;

  {-------------------------------------------------------------------}
  { Structure of the device constants read from the local database. }
  {-------------------------------------------------------------------}
  dev_const_type = RECORD
    min_current: REAL;
    max_current: REAL;
    min_ramptime: REAL;
    max_ramptime: REAL;
    bl_i_poly: poly_desc_type; { Bl(I) conversion }
    i_bl_poly: poly_desc_type; { I(Bl) conversion }
    bl_u_poly: poly_desc_type; { Bl(Uhall) conversion }
    END;
```

9 Dualport-RAM

Hierher kommt die Beschreibung des gerätespezifischen Teils des Dualport-RAM (m_data_type, s_data_type, Dev_Common_Buf_Type, ...).

```
TYPE
  s_rfc_type = RECORD
    origCurrent: real;
    origDelay: real;
    origRamptime: real;
    flattop: word;
    delay: word;
    delta: word;
    END;

  s_act_type = RECORD
    currenti_on: word;
    currenti_off: word;
    dynstat: uns_word;
    END;

  {-----------------------------------}
  { Master data type                  }
  {-----------------------------------}
  m_data_type = RECORD
    {------------------------------}
    { Status as delivered from IFB }
    {------------------------------}
    m_sts: m_status_type;

    {-------------------------}
    { Internal software state }
    {-------------------------}
    int_sts: internal_status_type;

    {-----------------------------------------------}
    { int_sts extensions for tracking the device's }
    { internal state including MMI-operation and   }
    { special handling of ESR injection septum.    }
    {-----------------------------------------------}
    svcd_vrtacc: uns_word;

    {--------------------------------------}
    { Device constants from local database }
    {--------------------------------------}
    const_actual: BOOLEAN;
    dev_const: dev_const_type;

    {---------------------------------}
    { Version of sweeper macro in EPLD }
    {---------------------------------}
    epld_version: uns_word;
```

```
  END; { record m_data_type }

{-----------------------------------}
{ Slave data type                   }
{-----------------------------------}
s_data_type = RECORD
  {-------------------------------------------------------------}
  { Sollwerte Flattopstrom, Startverzögerung und Stromdekrement   }
  { für einen Schritt. Errechnet aus Flattopstrom und Rampenzeit. }
  {-------------------------------------------------------------}
  sync_rfc: sync_type;
  rfc: ARRAY [sync_type] OF RECORD
    data: s_rfc_type;
    eficd: EFICD_Type;
    END;

  {---------------------------------------------}
  { Stromistwerte, gemessen zu den Zeitpunkten   }
  { der Events Prep_Beam_On und Beam_Off, sowie  }
  { der Zustand des Funktionsgenerators, gelesen }
  { zum Zeitpunkt des CurrentI-EQMs.             }
  {---------------------------------------------}
  sync_act: sync_type;
  act: ARRAY [sync_type] OF RECORD
    data: s_act_type;
    stamps: stamp_type;
    eficd: EFICD_Type;
    END;
  END; { record s_data_type }
```

10 USRs - User Service Routines

10.1 Obligatorische USRs

Obligatorische USRs werden hier nicht beschrieben. Sie werden als `DEFAULT_USRS.PIN` inkludiert und sind an anderer Stelle dokumentiert.

10.2 Gerätespezifische USRs

10.2.1 R_Calc

Den empfangenen Vorgabetyp testen. Er darf nur die Werte 1 (Vorgabe ist Feld), 2 (Vorgabe ist Strom) oder 3 (Vorgabe ist Spannung) annehmen.

Aus der Vorgabe Feld, Strom und Spannung berechnen und zurück liefern.

Undefinierte Vorgabetypen oder Bereichsüberschreitungen des Vorgabewertes oder der berechneten Werte werden mit einer Fehlermeldung beantwortet.

R_Calc benutzt zur Umrechnung der Werte die Funktionen aus X2Y.PIN (siehe Kapitel 10.4).

10.2.2 R_Constant

Lesen der gerätespezifischen Konstanten. Die Struktur der gelesenen Daten ist in Kapitel 6.3.8 auf Seite 26 beschrieben.

10.2.3 R_CurrentI

Den eventuell gelieferten Parameter testen.

Wurde kein Parameter oder ein Parameter mit dem Wert 1 geliefert, dann den im Dualport-RAM liegenden und beim Event `Prep_Beam_On` gemessenen digitalen Istwert in Strom umrechnen.

Wurde ein Parameter mit dem Wert 2 geliefert, entsprechend mit dem Istwert, der beim Event `Beam_Off` gemessen wurde, verfahren.

Den Strom-Istwert zurück liefern.

Andere Parameterkombinationen sind nicht erlaubt und werden mit einer Fehlermeldung beantwortet.

10.2.4 R_CurrentS

Aus den im Dualport-RAM gespeicherten Vorgabewerten für den Rampengenerator die Sollwerte *Strom*, *Startverzögerung* und *Rampendauer* berechnen (siehe IFCToSet in Kapitel 10.3.4) und den Strom zurück liefern.

10.2.5 W_CurrentS

Mit dem empfangenen Strom und den im Dualport-RAM gespeicherten originalen Sollwerten *Startverzögerung* und *Rampendauer* die Vorgabewerte für den Rampengenerator berechnen (siehe SetToIFC in Kapitel 10.3.3).

Test sowohl der Soll- als auch der Vorgabewerte auf Überschreitung ihrer definierten bzw. möglichen Bereich inklusive eventueller Fehlermeldung.

Speichern des Stroms als sog. Originalwert-Sollwert im Dualport-RAM.

10.2.6 R_Delay

Aus den im Dualport-RAM gespeicherten Vorgabewerten für den Rampengenerator die Sollwerte *Strom*, *Startverzögerung* und *Rampendauer* berechnen (siehe IFCToSet in Kapitel 10.3.4) und die Startverzögerung zurück liefern.

10.2.7 W_Delay

Aus der empfangenen Startverzögerung und den im Dualport-RAM gespeicherten originalen Sollwerten *Strom* und *Rampendauer* die Vorgabewerte für den Rampengenerator berechnen (siehe SetToIFC in Kapitel 10.3.3).

Test sowohl der Soll- als auch der Vorgabewerte auf Überschreitung ihrer definierten bzw. möglichen Bereich inklusive eventueller Fehlermeldung.

Speichern der Startverzögerung als sog. Originalwert-Sollwert im Dualport-RAM.

10.2.8 R_DynStat

Lesen des beschleuniger-spezifischen, dynamischen Gerätestatus. Die einzelnen Statusbits sind in Kapitel 3.4 auf Seite 12 definiert.

10.2.9 R_FieldI

Den eventuell gelieferten Parameter testen.

Wurde kein Parameter oder ein Parameter mit dem Wert 1 geliefert, dann den im Dualport-RAM liegenden und beim Event `Prep_Beam_On` gemessenen digitalen Istwert in Feld umrechnen mit Hilfe eines Polynoms dritten Grades, das die Funktion $Bl(I)$ beschreibt.

Wurde ein Parameter mit dem Wert 2 geliefert, entsprechend mit dem Istwert, der beim Event `Beam_Off` gemessen wurde, verfahren.

Den Feld-Istwert zurück liefern.

Andere Parameterkombinationen sind nicht erlaubt und werden mit einer Fehlermeldung beantwortet.

10.2.10 R_FieldS

Aus den im Dualport-RAM gespeicherten Vorgabewerten für den Rampengenerator die Sollwerte *Strom*, *Startverzögerung* und *Rampendauer* berechnen (siehe IFCToSet in Kapitel 10.3.4).

Umrechnung des Stromes in Feld mit Hilfe eines Polynoms dritten Grades, das die Funktion $Bl(I)$ beschreibt.

Die Sollwerte *Feld*, *Startverzögerung* und *Rampendauer* zurück liefern.

10.2.11 W_FieldS

Umrechnung des empfangenen Feldwertes in Strom mit Hilfe eines Polynoms dritten Grades, das die Funktion $I(Bl)$ beschreibt. Mit dem neuen Strom und den im Dualport-RAM gespeicherten originalen Sollwerten *Startverzögerung* und *Rampendauer* die Vorgabewerte für den Rampengenerator berechnen (siehe SetToIFC in Kapitel 10.3.3).

Test sowohl der Soll- als auch der Vorgabewerte auf Überschreitung ihrer definierten bzw. möglichen Bereich inklusive eventueller Fehlermeldung.

Speichern des berechneten Stroms als sog. Originalwert-Sollwert im Dualport-RAM.

10.2.12 R_MagnInfo

Lesen wichtiger Geräteinformationen und -einstellungen speziell für das Poti-Programm. Die Struktur der Daten ist in Kapitel 6.4.10 auf Seite 30 beschrieben.

R_MagnInfo erhält einen Parameter, der den gewünschten Istwert auswählt.

Wurde kein Parameter oder ein Parameter mit dem Wert 1 geliefert, dann den im Dualport-RAM liegenden und beim Event `Prep_Beam_On` gemessenen digitalen Istwert in Feld, Strom und Spannung umrechnen und zurück liefern.

Wurde ein Parameter mit dem Wert 2 geliefert, entsprechend mit dem Istwert, der beim Event `Beam_Off` gemessen wurde, verfahren.

10.2.13 R_MedDataI, R_MedDataS, W_MedDataS

Siehe die Beschreibungen der entsprechenden Properties in Kapitel 6.4.11 auf Seite 30 bzw. in Kapitel 6.4.12 auf Seite 31.

10.2.14 R_RampI

Aus den im Dualport-RAM liegenden beiden digitalen Istwerten jeweils das Feld berechnen mit Hilfe eines Polynoms dritten Grades, das die Funktion $Bl(I)$ beschreibt.

Die beiden Feld-Istwerte zurück liefern.

10.2.15 R_RampS

Aus den im Dualport-RAM gespeicherten Vorgabewerten für den Rampengenerator die Sollwerte *Strom*, *Startverzögerung* und *Rampendauer* berechnen (siehe IFCToSet in Kapitel 10.3.4) und zurück liefern.

Umrechnung des Stromes in Feld mit Hilfe eines Polynoms dritten Grades, das die Funktion $Bl(I)$ beschreibt.

Die Sollwerte *Feld*, *Startverzögerung* und *Rampendauer* zurück liefern.

10.2.16 W_RampS

Umrechnung des empfangenen Feldwertes in Strom mit Hilfe eines Polynoms dritten Grades, das die Funktion $I(Bl)$ beschreibt. Mit dem neuen Strom und den empfangenen Sollwerten *Startverzögerung* und *Rampendauer* die Vorgabewerte für den Rampengenerator berechnen (siehe SetToIFC in Kapitel 10.3.3).

Test sowohl der Soll- als auch der Vorgabewerte auf Überschreitung ihrer definierten bzw. möglichen Bereich inklusive eventueller Fehlermeldung.

10.2.17 R_RampTime

Aus den im Dualport-RAM gespeicherten Vorgabewerten für den Rampengenerator die Sollwerte *Strom*, *Startverzögerung* und *Rampendauer* berechnen (siehe IFCToSet in Kapitel 10.3.4) und die Rampendauer zurück liefern.

10.2.18 W_RampTime

Aus der empfangenen Rampendauer und den im Dualport-RAM gespeicherten originalen Sollwerten *Strom* und *Startverzögerung* die Vorgabewerte für den Rampengenerator berechnen (siehe SetToIFC in Kapitel 10.3.3).

Test sowohl der Soll- als auch der Vorgabewerte auf Überschreitung ihrer definierten bzw. möglichen Bereich inklusive eventueller Fehlermeldung.

Speichern der Rampendauer als sog. Originalwert-Sollwert im Dualport-RAM.

10.2.19 R_VoltI

Den eventuell gelieferten Parameter testen.

Wurde kein Parameter oder ein Parameter mit dem Wert 1 geliefert, dann den im Dualport-RAM liegenden und beim Event `Prep_Beam_On` gemessenen digitalen Istwert in Spannung umrechnen.

Wurde ein Parameter mit dem Wert 2 geliefert, entsprechend mit dem Istwert, der beim Event `Beam_Off` gemessen wurde, verfahren.

Den Spannungs-Istwert zurück liefern.

Andere Parameterkombinationen sind nicht erlaubt und werden mit einer Fehlermeldung beantwortet.

10.2.20 R_VoltS

Aus den im Dualport-RAM gespeicherten Vorgabewerten für den Rampengenerator die Sollwerte *Strom*, *Startverzögerung* und *Rampendauer* berechnen (siehe IFCToSet in Kapitel 10.3.4).

Umrechnung des Stromes in Spannung und diese zurück liefern.

10.2.21 W_VoltS

Aus der empfangenen Spannung den Strom berechnen. Mit dem neuen Strom und den im Dualport-RAM gespeicherten originalen Sollwerten *Startverzögerung* und *Rampendauer* die Vorgabewerte für den Rampengenerator berechnen (siehe SetToIFC in Kapitel 10.3.3).

Test sowohl der Soll- als auch der Vorgabewerte auf Überschreitung ihrer definierten bzw. möglichen Bereich inklusive eventueller Fehlermeldung.

Speichern des berechneten Stroms als sog. Originalwert-Sollwert im Dualport-RAM.

10.3 Globale Routinen

Hier werden alle Routinen aufgeführt, die im Modul USRs global definiert sind und von verschiedenen USRs benutzt werden.

10.3.1 GetConst

Vereinfachter Zugriff auf die gerätespezifischen Konstanten aus der lokalen Datenbasis. Es wird geprüft, ob die Konstanten bereits im Dualport-RAM liegen. Falls nicht, werden sie dorthin kopiert.

10.3.2 Polynom_3

Berechnung des Polynoms $y = a_3x^3+a_2x^2+a_1x+a_0$, wobei das für x passende Polynom ausgewählt wird, evtl. zwischen Null und dem Anfang des ersten Polynoms interpoliert wird und überschrittene Bereiche als Fehler gemeldet werden.

10.3.3 SetToIFC

Funktion zur Berechnung der Vorgabewerte für den Rampengenerator. SetToIFC erhält die Sollwerte *Flattop-Strom*, *Startverzögerung* und *Rampendauer* sowie die Gerätekonstanten und liefert die digitalen Vorgabewerte *Flattop-Strom*, *Startverzögerung* und *Stromdifferenz*.

Zur Berechnung der Stromdifferenz arbeitet SetToIFC mit der Formel

$$\Delta I_{\mathsf{step}} = I_{\mathsf{ft}}(\frac{f_{\mathsf{Q}}}{2}t_{\mathsf{ft}} + \frac{1 - n_{\mathsf{rund}}}{2})^{-1}$$

da in diese die Hardware-Eigenschaften des Rampengenerators mit f_{Q} und n_{rund} direkt einfließen.

Ist die berechnete Steigung nicht im Bereich, den der Rampengenerator erzeugen kann, werden die Vorgabewerte so generiert, dass *keine* Rampe gefahren, sondern der Flattop-Wert während des Strahlpulses beibehalten wird.

Zur Berechnung der digitalen Werte aus den Vorgaben der Property siehe Kapitel 4.1.

Es werden sowohl die Sollwerte als auch die Vorgabewerte auf Bereichüberschreitungen getestet.

10.3.4 IFCToSet

Funktion zur Berechnung der Sollwerte aus den Vorgabewerten des Rampengenerators.

SetToIFC erhält die digitalen Vorgabewerte *Flattop-Strom*, *Startverzögerung* und *Stromdifferenz* sowie die Gerätekonstanten und liefert die Sollwerte *Flattop-Strom*, *Startverzögerung* und *Rampendauer*.

10.3.5 I_OnOrOff

Test des optionalen Parameters der Properties `FieldI`, `CurrentI` oder `VoltI`. Es wird zurück geliefert, welcher der beiden möglichen Istwerte gewünscht ist.

10.4 Routinen zur Soll- und Istwertkonvertierung

Dies sind ebenfalls globale Routinen, die sich in der Include-Datei `X2Y.PIN` befinden.

Die Funktionen in `X2Y.PIN` benötigen die Funktion `polynom_3` sowie die Konstanten `umax_integer`, `dac_max_uns`, `dac_max_real` und `dac_fact_mvolt_real` für die Umrechnungen. Diese müssen außerhalb von `X2Y.PIN` definiert werden.

Alle Funktionen erhalten den umzurechnenden Wert und die Gerätekonstanten als Input-Parameter. Sie liefern den umgerechneten Wert als Output-Parameter sowie den Fehlerstatus als Funktionsparameter.

10.4.1 Bl2I

Funktion: Feld in Strom umrechnen.

Umrechnung des Feldes in Strom mit Hilfe eines Polynoms dritten Grades, das die Funktion $I(Bl)$ beschreibt.

Test, ob der berechnete Strom innerhalb der definierten Grenzen liegt.

10.4.2 Bl2U

Funktion: Feld in Spannung umrechnen.

Umrechnung des Feldes in Strom mit Hilfe eines Polynoms dritten Grades, das die Funktion $I(Bl)$ beschreibt. Umrechnung des Stroms in Spannung.

Test, ob die berechnete Spannung innerhalb der definierten Grenzen liegt.

10.4.3 Bl2DAC

Funktion: Feld in den Wert für den D/A-Wandler umrechnen.

Umrechnung des Feldes in Strom mit Hilfe eines Polynoms dritten Grades, das die Funktion $I(Bl)$ beschreibt.

Test, ob der berechnete Strom innerhalb der definierten Grenzen liegt.

Umrechnung des Stroms in den Wert für den D/A-Wandler.

10.4.4 I2Bl

Funktion: Strom in Feld umrechnen.

Test, ob der angegebene Strom innerhalb der definierten Grenzen liegt.

Umrechnung des Stroms in Feld mit Hilfe eines Polynoms dritten Grades, das die Funktion $Bl(I)$ beschreibt.

10.4.5 I2U

Funktion: Strom in Spannung umrechnen.

Test, ob der angegebene Strom innerhalb der definierten Grenzen liegt.

Umrechnung des Stroms in Spannung.

10.4.6 I2DAC

Funktion: Strom in den Wert für den D/A-Wandler umrechnen.

Test, ob der angegebene Strom innerhalb der definierten Grenzen liegt.

Umrechnung des Stroms in den Wert für den D/A-Wandler.

10.4.7 U2Bl

Funktion: Spannung in Feld umrechnen.

Test, ob die angegebene Spannung innerhalb der definierten Grenzen liegt.

Umrechnung der Spannung in Strom.

Umrechnung des Stroms in Feld mit Hilfe eines Polynoms dritten Grades, das die Funktion $Bl(I)$ beschreibt.

10.4.8 U2I

Funktion: Spannung in Strom umrechnen.

Test, ob die angegebene Spannung innerhalb der definierten Grenzen liegt.

Umrechnung der Spannung in Strom.

10.4.9 U2DAC

Funktion: Spannung in den Wert für den D/A-Wandler umrechnen.

Test, ob die angegebene Spannung innerhalb der definierten Grenzen liegt.

Umrechnung der Spannung in den Wert für den D/A-Wandler.

10.4.10 DAC2Bl

Funktion: Wert für den D/A-Wandler in Feld umrechnen.

Umrechnung des Wertes für den D/A-Wandler in Strom.

Umrechnung des Stroms in Feld mit Hilfe eines Polynoms dritten Grades, das die Funktion $Bl(I)$ beschreibt.

10.4.11 DAC2I

Funktion: Wert für den D/A-Wandler in Strom umrechnen.

Umrechnung des Wertes für den D/A-Wandler in Strom.

10.4.12 DAC2U

Funktion: Wert für den D/A-Wandler in Spannung umrechnen.

Umrechnung des Wertes für den D/A-Wandler in Spannung.

11 EQMs - Equipment Modules

11.1 Interne Zustände

11.1.1 Bedeutung der internen Zustände

Für die Gerätesoftware sind folgende interne Zustände definiert:

not_set	Initzustand. Dieser Zustand sollte nie auftreten.
noifc	Der Rampengenerator als Erweiterung der Interface-Karte fehlt oder konnte nicht eingeschaltet werden.
emergency	Ein Emergency-Event wurde empfangen. Dieser Zustand darf nur durch Rücksetzen vom Operating verlassen werden.
interlock	Ein Interlock wurde gemeldet. In einem periodisch ablaufenden Auftrag wird überprüft, ob die Interlock-Ursache noch vorliegt. Falls nein, Übergang nach ready.
local	Das Gerät wird mit Handsteuerung betrieben.
power_off	Das Gerät ist ausgeschaltet.
power_seq	Das Gerät schaltet gerade ein oder aus.
error	Während der Abarbeitung eines EQMs wurde ein Fehler erkannt.
ready	Das Gerät ist bereit für Aktionen. Ausgangszustand am Beginn und Endzustand am Ende eines virtuellen Beschleunigers.
busy	Dem Gerät wurden die Sollwerte geschickt und das Lesen der Istwerte ist noch nicht beendet.

11.1.2 Übergänge zwischen den Zuständen

Die Zustände und die Übergänge zwischen denselben sind aus Platzgründen in die Tabellen 6 und 7 aufgeteilt.

Der Zustand „noifc“ kann nur durch die EQMs Init und Reset eingenommen und auch wieder verlassen werden. Er wird in den Tabellen nicht noch einmal aufgeführt.

Tabelle der Zustandsübergänge

von↓ nach→		emergency	interlock	local	power_off	power_seq
emergency	U:	–	RESET, SI	RESET	RESET	–
	B:	–	–	r	Rp	–
	A:	–	Interl_EQM	Reset_EQM	Reset_EQM	–
interlock	U:	Evt_Emerg	–	RESET	RESET	–
	B:	–	–	r	Rp	–
	A:	Emerg_EQM	–	SI_Off_Chk, Reset_EQM	SI_Off_Chk, Reset_EQM	–
local	U:	Evt_Emerg	SI	–	–	–
	B:	–	–	–	Rp	–
	A:	Emerg_EQM	Interl_EQM	–	Status lesen (periodisch)	–
power_off	U:	Evt_Emerg	SI	–	–	Power=1
	B:	–	–	r	–	–
	A:	Emerg_EQM	Interl_EQM	Status lesen (periodisch)	–	Power_EQM
power_seq	U:	Evt_Emerg	SI	–	–	–
	B:	–	–	r	Rp	–
	A:	Emerg_EQM	Interl_EQM	Status lesen (periodisch)	ChkPwr_EQM	–
error	U:	Evt_Emerg	SI	–	–	–
	B:	–	–	r	Rp	–
	A:	Emerg_EQM	Interl_EQM	Status lesen (periodisch)	Status lesen (periodisch)	–
ready	U:	Evt_Emerg	SI	–	–	Power=0
	B:	–	–	r	Rp	–
	A:	Emerg_EQM	Interl_EQM	Status lesen (periodisch)	Status lesen (periodisch)	Power_EQM
busy	U:	Evt_Emerg	SI	–	–	–
	B:	–	–	–	–	–
	A:	Emerg_EQM	Interl_EQM	–	–	–

Tabelle 6: Zustandsübergangsdiagramm 1

Legende zu den Tabellen 6 und 7:

U: Auslösende Ursache.

SI	Summeninterlock des Gerätes steht an.
Evt_Emerg	Pulszentrale verschickte Emergency-Event.
RESET	Reset wird per Kommando oder Knöpfchendrücken ausgelöst.
Power=1	Power wird per Kommando eingeschaltet.
Power=0	Power wird per Kommando ausgeschaltet.

B: Abzuprüfende Bedingung.

R	Remotebit des Status steht auf Remote.
r	Remotebit des Status steht auf Local.
P	Powerbit des Status steht auf Power on.
p	Powerbit des Status steht auf Power off.

A: Ausführende Stelle des Zustandübergangs.

Status lesen (periodisch)	Beim periodischen (oder zumindest regelmäßigen) Lesen des Status.
x_EQM	Innerhalb des EQMs x_EQM.

– Die Priorität der Zustände (höchste Priorität zuerst): emergency, interlock, local, power_off und power_seq, error, ready und busy.

Liegen mehrere Bedingungen für verschiedene Zustände gleichzeitig vor (z. B. Netz aus und Gerät auf Handbetrieb), muss der jeweils wichtigste Zustand eingenommen werden.

Tabelle der Zustandsübergänge				
von↓ nach→		error	ready	busy
emergency	U:	–	RESET	–
	B:	–	–	–
	A:	–	Reset_EQM	–
interlock	U:	–	RESET	–
	B:	–	–	–
	A:	–	SI_Off_Chk, Reset_EQM	–
local	U:	–	–	–
	B:	–	RP	–
	A:	–	Status lesen (periodisch)	–
power_off	U:	–	–	–
	B:	–	RP	–
	A:	–	Status lesen (periodisch)	–
power_seq	U:	MIL timeout	–	–
	B:	–	RP	–
	A:	ChkPwr_EQM	ChkPwr_EQM	–
error	U:	–	RESET, Zyklusende	–
	B:	–	RP	–
	A:	–	Reset_EQM, RampS_EQM	–
ready	U:	overrun etc.	–	„Evt_Set_Sollwert“
	B:	–	–	–
	A:	div. EQMs	–	RampS_EQM
busy	U:	overrun etc.	Event Beam_Off	–
	B:	–	–	–
	A:	RampS_EQM RampI_EQM	RampI_EQM	–

Tabelle 7: Zustandsübergangsdiagramm 2

11.1.3 Standard-Zustandsübergänge

Zur Verdeutlichung hier einige Standard-Zustandsübergänge. Sie kommen Zustande, wenn eine Sequenz ohne Fehler abläuft.

Das Gerät nimmt an der Puls-zu-Puls-Modulation teil. Innerhalb eines virtuellen Beschleunigers wird zum Sollwert setzen der Zustand `ready` erwartet und zum Zustand `busy` weitergeschaltet. Zum Istwert lesen wird der Zustand `busy` erwartet und zum Zustand `ready` weitergeschaltet.

```
ready  ->  busy  ->  ready
```

Das Gerät schaltet ein bzw. aus.

```
power_off  ->  power_seq  ->  ready
ready      ->  power_seq  ->  power_off
```

11.2 Eventkonnektierte EQMs

11.2.1 RampS_EQM

Event: Ready_To_SIS.

Delay: 0 ms.

Aktion: Das Bcst_EQM verzögert starten. Status lesen und, wenn nötig, den internen Zustand anpassen. Ist das Gerät erreichbar und der interne Zustand ist in Ordnung, dann die Vorgabewerte an den Rampengenerator schicken. Diese werden damit noch *nicht* realisiert (siehe Kapitel 11.2.2).

11.2.2 BcstS_EQM

Event: (Ready_To_SIS). Siehe Kapitel 11.2.1.

Delay: 7,1 ms.

Aktion: Via Broadcast *alle* Rampengeneratoren anweisen, den Flattop-Sollwert an den Geräten zu realisieren.

11.2.3 RampI_EQM

Event: Beam_Off.

Delay: 0 ms.

Aktion: Unabhängig vom internen und vom Aktivzustand die Istwerte lesen, falls das Gerät online ist.

11.2.4 Emergency_EQM

Event: Emergency.

Aktion: Internen Zustand auf „Emergency“ setzen.

11.3 Periodisch konnektierte EQMs

11.3.1 CheckPower_EQM

Zeit: $2s$

Anzahl: Einmalige Ausführung.

Aktion: Prüfen, ob Ein- oder Ausschalten erfolgt ist. Das Schalten der Power kann bis zu $2s$ dauern.

11.3.2 Update_Config_EQM

Zeit: $60s$

Anzahl: Unendlich.

Aktion: Aktualisieren der Geräteverfügbarkeit: Es wird versucht, von möglichen Geräteadressen den Status zu lesen. Erfolgt eine Reaktion, wird das Gerät als „online“ geführt.

11.4 An externe Interrupts konnektierte EQMs

11.4.1 Interlock_EQM

Interrupt: Summen-Interlock.

Aktion: Internen Zustand auf „Interlock“ setzen, falls er nicht „Emergency“ ist. Sollwert Null setzen.

11.4.2 DRD_EQM

Interrupt: Data Ready Interrupt.

Aktion: Keine. Sollte bei MS nicht vorkommen.

11.4.3 DRQ_EQM

Interrupt: Data Request Interrupt.
Aktion: Keine. Sollte bei MS nicht vorkommen.

11.5 Kommandogetriggerte EQMs

11.5.1 Dev_Init_EQM

Folgende Aktionen werden ausgeführt:

- Fehlerpuffer initialisieren.
- Reset ans Gerät schicken.
- Rampengenerator auf IFK einschalten (EnableSweeperIFC).
- Interlock-Service präparieren (Do_Intr_Service_Prep).
- Status vom Gerät lesen (Read_and_Update_Status).
- Kostanten in Dualport-RAM initialisieren.
- Internen Zustand setzen (Set_InternalState).
- Aktivzustand für alle virtuellen Beschleuniger initialisieren.
- Alle Slave-Soll- und -Istdaten initialisieren.
- Fehlerbehandlung.

11.5.2 Dev_Reset_EQM

Folgende Aktionen werden ausgeführt:

- Fehlerpuffer initialisieren.
- Reset ans Gerät schicken.
- Rampengenerator auf IFK einschalten (EnableSweeperIFC).
- Interlock-Service präparieren (Do_Intr_Service_Prep).
- Status vom Gerät lesen (Read_and_Update_Status).
- Kostanten in Dualport-RAM initialisieren.
- Internen Zustand setzen (Set_InternalState).
- Aktivzustand für alle virtuellen Beschleuniger initialisieren.
- Alle Slave-Solldaten validieren.
- Alle Slave-Istdaten initialisieren.
- Fehlerbehandlung.

11.5.3 Status_EQM

Folgende Aktionen werden ausgeführt:

- Status vom Gerät lesen (Read_and_Update_Status).
- Internen Zustand setzen (Set_InternalState).
- Fehlerbehandlung.

11.5.4 Active_EQM

Folgende Aktionen werden ausgeführt:

- Status vom Gerät lesen (Read_and_Update_Status).
- Internen Zustand setzen (Set_InternalState).
- Gerät aktiv bzw. inaktiv setzen.
- Fehlerbehandlung.

11.5.5 Power_EQM

Folgende Aktionen werden ausgeführt:

- Status vom Gerät lesen (Read_and_Update_Status).
- Internen Zustand setzen (Set_InternalState).
- Internen Zustand prüfen (int_sts_ok).
- Ist der interne Zustand ok, dann das Gerät ein bzw. ausschalten und einen Auftrag zur Prüfung der Aktion aufsetzen (switch_power).
- Fehlerbehandlung.

11.5.6 SetMedDataS_EQM

Folgende Aktionen werden ausgeführt:

- Empfangenen Therapie-Datensatz in den lokalen Speicher kopieren (SaveSlaveSet).
- Fehlerbehandlung

11.5.7 GetMedDataS_EQM

Folgende Aktionen werden ausgeführt:

- Unabhängige EFI-Parameter Null setzen
- EFI-Parameter testen
- Falls ein gültiger Datensatz angefordert wurde, diesen zurückschicken.
- Fehlerbehandlung

11.5.8 MedDataI_EQM

Folgende Aktionen werden ausgeführt:

- Angeforderten Datensatz zurückschicken (ReadActVal).
- Fehlerbehandlung

11.6 EQMs für die Diagnose vor Ort

11.6.1 Display_DPR_EQM

Parameter: Das EQM benötigt 2 Parameter.

P1: virtueller Beschleuniger (in Hex angeben)
P2: logische Gerätenummer (in Hex angeben)

Daten: Keine.

Aktion: Zeigt am Bildschirm vor Ort die wichtigsten Daten aus dem DPRAM für das gewählte Gerät und den gewählten virtuellen Beschleuniger an.

11.6.2 Display_DevErr_EQM

Parameter: Das EQM benötigt 2 Parameter.

P1: virtueller Beschleuniger (in Hex angeben)
P2: logische Gerätenummer (in Hex angeben)

Daten: Keine.

Aktion: Zeigt am Bildschirm vor Ort die Fehlerkodes aus der Datenstruktur im Dualport-RAM für das gewählte Gerät und den gewählten virtuellen Beschleuniger an.

11.7 Globale Routinen

11.7.1 Read_and_Update_Status

Liest den 24 Bits breiten Hardware-Status des Gerätes, leitet die Software-Statusbits daraus ab und schreibt den neuen Status ins Dualport-RAM. Bei einer Statusänderung wird automatisch ein Statusalarm verschickt.

11.7.2 Set_InternalState

Bestimmt aus dem Status und dem aktuellen internen Zustand den neuen internen Zustand.

11.7.3 Do_Intr_Service_Prep

Setzt das Summmeninterlock-Display im Dualport-RAM zurück und ermöglicht den Interlock-Interrupt auf der Inetrface-Karte.

11.7.4 int_sts_ok

Testet, ob der interne Zustand das Ein- oder Ausschalten der Power erlaubt.

11.7.5 EnableSweeperIFC

Testet die Version der Interface-Karte, schaltet den Rampengenerator auf der Interface-Karte ein, testet ob er eingeschaltet ist und liest dessen EPLD-Version.

11.7.6 switch_power

Schaltet die Power ein oder aus und setzt einen (periodischen) Auftrag auf, der nach 2s nachschaut, ob die Power ein- bzw. ausgeschaltet ist.

11.8 Therapie-Routinen

Werden Therapie-EQMs inkludiert, dann müssen die EQMs einige sogenannte Callback-Routinen zur Verfügung stellen, die von den Therapie-EQMs aufgerufen werden. Definitionen für den Therapiebetrieb sind in Kapitel 5 beschrieben.

11.8.1 SetActiveState

Stellt für den Medizinbeschleuniger den für den Thearpiebetrieb definierten Aktivzustand ein.

11.8.2 CheckPowerState

Testet den für den Therapiebetrieb definierten Zustand der Power des Gerätes. Sweeper dürfen ein oder aus sein.

11.8.3 SetDefaultSlaveSet

Schreibt einen Default-Slave-Datensatz als Sollwerte für den Therapiebeschleuniger ins Dualport-RAM. Diese Routine wird aufgerufen, wenn der angeforderte (EFI-) Datensatz nicht aus dem lokalen Speicher geholt werden konnte.

11.9 Sonstige Routinen

11.9.1 Startup_EQM

Installiert die Event-EQM-Konnektierung für alle virtuellen Beschleuniger (siehe hierzu auch Kapitel 4.8 auf Seite 20), konnektiert die Therapie-EQMs und schaltet die SE in den Event-Mode.

11.9.2 UserIni

Initialisiert die Dualport-RAM-Konstanten, setzt die periodischen Aufträge auf und macht die EQMs dem ECM bekannt.

11.10 MIL-Treiber

Zum Ansteuern der Interfacekarte wird der Standard-MIL-Treiber eingesetzt.

Literatur

[1] Ludwig Hechler. Therapieprojekt: Die GμP-Ebene. Accelerator Controls Documentation F-MBP-01, Gesellschaft für Schwerionenforschung, Darmstadt, April 1995. (Source: dd_gup.tex).

[2] Peter Kainberger. Therapieprojekt: Die SE-Ebene. Accelerator Controls Documentation F-MBP-02, Gesellschaft für Schwerionenforschung, Darmstadt, Mai 1995. (Source: dd_se.tex).

Index

— V —

— W —

— X —

— Z —