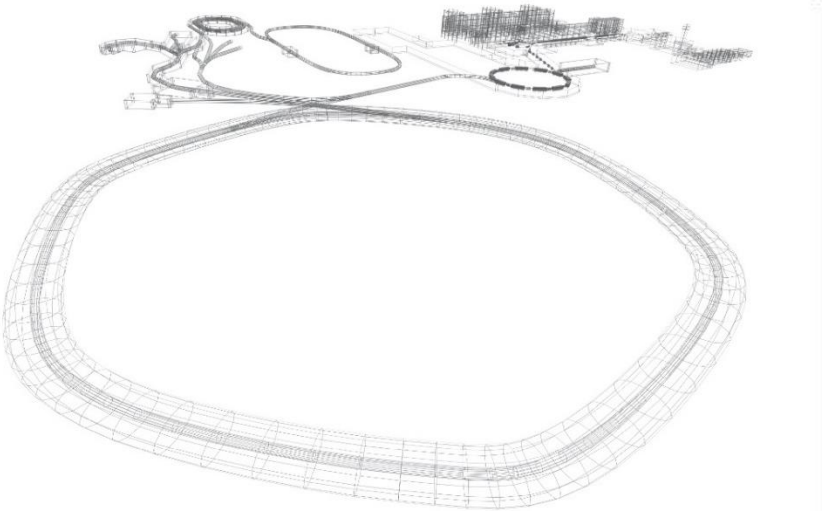
Elektrische
Energie
Technik

GSI Darmstadt



FSPs der Multi Function Unit Second Edition (MFU SE)

(ab FW 7.5)

Version vom: Freitag, 1. März 2024, 12:14:00

Aktuelle Versionen immer unter:

https://wiki.gsi.de/EPS/ACUModuleDescriptions

Inhaltsverzeichnis

1. Änderungsliste

Datum	Name	Kommentar
01.07.2021	D. Schupp	Dokument erstellt aus ACU-FSP mUSIc TFT
11.11.2022	D. Schupp	Überabeitet für MFU FW 7.5
15.02.2023	D. Schupp	Überabeitet für MFU FW 7.5
17.03.2023	D. Schupp	Überabeitet für MFU FW 7.5
05.04.2023	D. Schupp	FSP110, DAC4 Eingänge korrigiert
01.03.2024	D. Schupp	[FSP252_UpdateMFU_CFI_SoftwareViaRemote] Info über „Bulkerase“ des CFI hinzugefügt. FSP242_SW_CPU_Status[29] ergänzt.

2. FSPs MFU

Dieses Dokument behandelt modulspezifische FSPs der MultiFunctionUnit (MFU).

FSP beginnend mit „_SW_“ sind nicht in der MFU Hardware implementiert, sondern Bestandteil der Nios-Software.

Name	**FSP001_ModuleStatus**
Adresse	**$0x01_H/1_D/0x3031_{ASCII}$**
Tiefe	3 Byte / 24 Bit
I/O	lesen
Reset	$0x(\text{siehe Beschreibung})_H$

[23..22] n.u., immer ‚0'

[21] Zustand Remote/Local Schalter: '0' ACU System im Local-Betrieb '1', ACU System im Remote-Betrieb

[20] Controller Enabled
wenn 1' ist Regler freigegeben

[19..16] Gerätestatus (4 Bit)

[3..0]	Status
0x0	Kein Status lesbar
0x1	*cSTATUSSetDefaults/ cSTATSUWaitForParameters* keine definierter Status
0x2	*cSTATUSUnitOff* Gerät ausgeschaltet
0x3	*cSTATUSLoadingBank* Bank laden
0x4	*cSTATUSSwitchingUnitOn* Gerät einschalten
0x5	*cSTATUSUnitOn* Gerät eingeschaltet
0x6	*cSTATUSControllerDisabledByFPGAInternalCause* FPGA interne Gründe (des Status erzeugenden Moduls) sperren den Regler
0x7	*cSTATUSControllerEnabled* Regler freigegeben
0x8	*cSTATUSSwitchingUnitOff* Gerät ausschalten
0x9	*cSTATUSControllerDisabledByCommand* Das Kommando *cCMDDisableController* sperrt den Regler
0xA	*cSTATUSControllerDisabledByFPGAExternalCause* FPGA externe Gründe (des Status erzeugenden Moduls) sperren den Regler
0xB	*cSTATUSResetInterlocks*
0xC	*cSTATUSMachineProtection*
0xD	n.u.
0xE	*cSTATUSPowerOnReset*
0xF	*cSTATUSWhenOthers* keine definierter Status

[15..12] Kommando (lokales oder „remote" Kommando)
→ siehe FSP010_ModuleCommands

[11..9] n.u. immer 0

[8] USIIsHighSpeed
wenn ‚1' ist mindestens eine der USI im Highspeed Mode

[7..6] immer 0

[5] NoInterlocks
wenn ‚1' stehen keine Interlocks an
Im Modul sind keine Interlocks gespeichert und es stehen auch keine Interlocks an.

[4] NoErrors
wenn ‚1' ist Modul fehlerfrei
Im Modul sind keine Fehler gespeichert die den Betrieb stören.

[3] NoWarnings
wenn ‚1' ist Modul ohne Warnungen
Im Modul sind keine Warnmeldungen vorhanden die den Betrieb zwar nicht stören aber trotzdem überprüft werden müssten (Details im FSP für die Warnungsbits) z.B. Temperatur zu hoch.

[2] ModuleReady
wenn ‚1' ist Modul betriebsbereit
Das Modul ist voll betriebsbereit, nachdem der Nios2 die Bootsequenz abgearbeitet hat.

[1] ChecksumOK
wenn ‚1' Parameter Checksumme OK
Die Prüfsumme für die Modulparameter ist bestätigt.

[0] ParametersLoaded
wenn ‚1' sind die Parameter geladen
Das Modul hat seine Konfigurationsparameter geladen.
Dies ist dann der Fall, wenn entweder der Nios2 die Parameter aus dem internen Flash geladen hat oder die Parameter über einen PC mittels PCA in die ACu geladen wurden.

Name	**FSP009_ModuleSerialNumber**
Adresse	**$0x09_H/9_D/0x3039_{ASCII}$**
Tiefe	6 Byte / 48 Bit
I/O	lesen
Reset	$0x(\text{siehe Beschreibung})_H$

Der FSP enthält die MFU Serien Nummer

Die Serien Nummer ist über einen One Wire Chip von Dallas/Maxim zu erzeugen, da gewährleistet sein muss das die Serien Nummer weltweit nur einmal vergeben ist.

[47..0] MFU Seriennummer (48 Bit)

Name	**FSP010_ModuleCommands**
Adresse	**$0x0A_H/10_D/0x3041_{ASCII}$**
Tiefe	1 Byte / 8 Bit
I/O	lesen / schreiben
Reset	$0x00_H$

Wenn ein Modul Kommandos unterstützt (Einschalten, Ausschalten, Reset usw.) dann werden diese über diesen FSP gesetzt.

Ist die MFU Betriebsart „Remote“ erfolgt die Kommandoübermittlung über das Kontrollsystem-Kommandoregister und die Daten aus diesem FSP werden ignoriert.

Zu beachten ist: die MFU reagiert nur auf Änderungen der Kommandos. Soll also das gleiche Kommando mehrfach hintereinander ausgeführt werden, bedarf es jedes Mal dem Kommando `cCMDNoAction` dazwischen.

[7..4] n.u.

[3..0] Command_USB, diese Kommandos werden z.B. von PowerConfigAdvanced oder ein Terminalprogram über die MFU USB Verbindung gesetzt und steuern die Gerätefunktion, sofern sich die MFU in der Betriebsart „*Local*“ befindet.

[3..0]	Kommando
0x0	`cCMDNoAction` keine Aktion
0x1	`cCMDSwitchUnitOn` Gerät einschalten (wenn möglich)
0x2	`cCMDSwitchUnitOff` Gerät abschalten
0x3	`cCMDResetUnit` Reset durchführen (z.B. Interlocks)
0x4	`cCMDDisableController`
0x5	`cCMDTriggerSomething` hiermit lassen sich Sonderfunktionen in Modulen auslösen

Name	**FSP013_PeripheralConfig**
Adresse	**$0x0D_H/13_D/0x3044_{ASCII}$**
Tiefe	1 Byte / 8 Bit
I/O	lesen / schreiben
Reset	$0x82_H$

Auf dem Modul befindliche Peripherie kann mit diesem FSP konfiguriert werden

[7] Nach einschalten der Spannungsversorgung wird dieses Bit automatisch auf ‚1' gesetzt. Werden Parameter in die MFU geladen (per PC oder durch die MFU selbst) die zur Prüfsummenbildung beitragen sollen, muss dieses Bit gelöscht werden, bevor der erste Parameter übertragen wird. Ist das Laden der Parameter beendet, muss dieses Bit wieder auf ‚1' gesetzt werden. Im Anschluss daran wird die Vergleichs-Prüfsumme an „FSP058_ParameterChecksumValue" gesendet. Die Freigabe des Reglers erfolgt aber nur, wenn die Vergleichs-Prüfsumme auch zu der aus den restlichen Parametern gebildeten Prüfsumme passt.
Das Löschen dieses Bit löst ggf. eine gezielte Reglersperre aus und die „alte" in der MFU errechnete Prüfsumme wird gelöscht.
Die MFU führt im normalen Betrieb zyklische Zugriffe auf die FSP durch, dies auch während neue Parameter geladen werden. Lesezugriffe auf die FSP sind kein Problem, da die Prüfsumme nur bei Schreibzugriffen aufaddiert wird. Damit diese Zugriffe nicht für eine Verfälschung der Parameter Prüfsumme führen, werden Schreib-Zugriffe seitens der MFU unterbunden solange dieses Bit ‚0' ist.

Ggf. ist es sinnvoll, dieses Bit nicht mit einem direkten Zugriff auf FSP013 zu löschen, da hierdurch evtl. auch weitere Bits des FSP013 verändert werden. Soll dieses Bit nur gezielt gelöscht werden, kann auch FSP241 benutzt werden.

[6..4] n.u.

[3] Wenn ‚0' kann Bit [2] vom Anwender über das Menü NICHT geändert werden, d.h. es ist lokal am Gerät NICHT möglich zwischen Strom- und/oder Feldregelung umzuschalten. Wenn ‚1' kann Bit [2] hingegen am Gerät lokal über das Menü geändert werden. Nach dem Reset ist diese Bit immer ‚0', d.h. der Nutzer kann lokal den Regelmodus NICHT ändern.

[2] Wenn ‚0' wird Stromgeregelt, wenn ‚1' Feldgeregelt. Nach dem ‚Reset' ist immer die Stromregelung gewählt.

[1] Wenn ‚0' wird der Regler gesperrt, d.h. für eine Reglerfreigabe muss dieses Bit gesetzt werden. Dieses Bit ist UND-Verknüpft mit Bit[7].

[0] wenn '1' können über den USB Port (Front MFU) Sollwerte und Kommandos (Ein, Aus, Reset) gesetzt werden, wenn '0' über Backplane-Interface (Interface-Karte oder SCU)

Name	**FSP014_CurrentScale**
Adresse	**$0x0E_H/14_D/0x3045_{ASCII}$**
Tiefe	4 Byte / 32 Bit
I/O	lesen / schreiben
Reset	$0x0000\_000A_H$

Repräsentiert die Stromskalierung des ACU Systems und des darin zur Anwendung kommenden DCCT (10V = CurrentScale Ampere).

[31] wenn ‚1' → bipolar, dabei reicht es, wenn FSP14[31], FSP15[31] oder FSP16[31] gesetzt ist, das Gerät wird dann in jedem Fall als bipolar betrachtet

[30..24] n.u.

[23..0] Stromskalierung (24 Bit)

Name	**FSP015_VoltageScale**
Adresse	**$0x0F_H/15_D/0x3046_{ASCII}$**
Tiefe	4 Byte / 32 Bit
I/O	lesen / schreiben
Reset	Reset:$0x0000\_000A_H$

Repräsentiert die Spannungsskalierung des ACU Systems

[31] wenn ‚1' → bipolar, dabei reicht es, wenn FSP14[31], FSP15[31] oder FSP16[31] gesetzt ist, das Gerät wird dann in jedem Fall als bipolar betrachtet

[30..24] n.u.

[23..0] Spannungsskalierung (24 Bit)

Name	**FSP016_BFieldScale**
Adresse	**$0x10_H/16_D/0x3130_{ASCII}$**
Tiefe	4 Byte / 32 Bit
I/O	lesen / schreiben
Reset	Reset:$0x0000\_000A_H$

Repräsentiert die Feldskalierung des ACU Systems und der darin zur Anwendung kommenden Feldsonde.

[31] wenn ‚1' → bipolar, dabei reicht es, wenn FSP14[31], FSP15[31] oder FSP16[31] gesetzt ist, das Gerät wird dann in jedem Fall als bipolar betrachtet

[30..24] n.u.

[23..0] Feldkalibrierung (24 Bit)

Name	**FSP020_ActualValue_A**
Adresse	**$0x14_H/20_D/0x3134_{ASCII}$**
Tiefe	3 Byte / 24 Bit
I/O	lesen
Reset	$0x(\text{siehe Beschreibung})_H$

Über diesen FSP kann der Istwert_**A** gelesen werden. I.d.R. ist dies der Stromistwert des DCCT.

[23..0] Vorzeichenbehafteter 20 Bit Wert, Bit[3..0] sind immer ‚0'.

Name	**FSP021_ActualValue_B**
Adresse	**$0x15_H/21_D/0x3231_{ASCII}$**
Tiefe	3 Byte / 24 Bit
I/O	lesen
Reset	$0x(\text{siehe Beschreibung})_H$

Über diesen FSP kann der Istwert_**B** gelesen werden. I.d.R. ist dies der Feldistwert der Feldsonde.

[23..0] Vorzeichenbehafteter 20 Bit Wert, Bit[3..0] sind immer ‚0'.

Name	**FSP029_ActualValuePhysicalQuantities**
Adresse	**$0x1D_H/29_D/0x3144_{ASCII}$**
Tiefe	2 Byte / 16 Bit
I/O	lesen / schreiben
Reset	$0x0030_H$

Gibt die physikalische Größe der Istwerte an.

[15..12] (physikalische Einheit des Istwertes D) obsolet

[11..8] (physikalische Einheit des Istwertes C) obsolet

[7..4] physikalische Einheit des Istwertes **B**

[3..0] physikalische Einheit des Istwertes **A**

[3..0]	Einheit
0x0	Ampere (A)
0x1	Volt (V)
0x2	Temperatur (C)
0x3	Magnetische Flussdichte (T)
0x4	Magnetische Flussdichte (G)
0x5	n.u.
0x6	n.u.
0x7	n.u.
0x8	n.u.
0x9	n.u.
0xA	n.u.
0xB	n.u.
0xC	n.u.
0xD	n.u.
0xE	n.u.
0xF	n.u.

Name	**FSP030_SetValue_A**
Adresse	**$0x1E_H/30_D/0x3145_{ASCII}$**
Tiefe	3 Byte / 24 Bit
I/O	lesen / schreiben
Reset	$0x000000_H$

Über diesen FSP wird der Sollwert_**A** im Remotebetrieb an die MFU übermittelt.

[23..0] Vorzeichenbehafteter 20 Bit Wert (0xFFFFF bis 0x7FFFF),
wird über verschiedene Multiplexer diversen Zielen vorgegeben, Bit[3..0] sind immer ‚0‘:

- Als **Soll**wert für den PI Regler **1** mittels des Sollwert-Multiplexer, welcher über „FSP070_Controller_1_2_InputSourceSelectionMultiplexer“ konfiguriert wird.
- Als **Soll**wert für den PI Regler **2** mittels des Sollwert-Multiplexer, welcher über „FSP070_Controller_1_2_InputSourceSelectionMultiplexer“ konfiguriert wird.
- Als **Ist**wert für den PI Regler **1** mittels des Sollwert-Multiplexer, welcher über „FSP070_Controller_1_2_InputSourceSelectionMultiplexer“ konfiguriert wird.
- Als Summand **1** des Addierers **1**, welcher über „FSP090_Adder_SourceSelectionMultiplexer“ konfiguriert wird.
- Als Quelle für den „ACU_SovereignitySelector_USBorIFK_SCU“ Block, dessen Ausgang als „SetValue_**A**_MultiplexedOutput“ Verwendung findet. Abhängig von „FSPFSP013_PeripheralConfig[0]“ wird entweder dieser Sollwert(1) oder der Sollwert des Kontrollsystems(0) via Backplane für „SetValue_**A**_MultiplexedOutput“ verwendet.

Name	**FSP031_SetValue_B**
Adresse	**$0x1F_H/31_D/0x3146_{ASCII}$**
Tiefe	3 Byte / 24 Bit
I/O	lesen / schreiben
Reset	$0x000000_H$

Über diesen FSP wird der Sollwert_**B** im Remotebetrieb an die MFU übermittelt.

[23..0] Vorzeichenbehafteter 20 Bit Wert (0xFFFFF bis 0x7FFFF)
wird über verschiedene Multiplexer diversen Zielen vorgegeben, Bit[3..0] sind immer ‚0‘.:

- Als **Ist**wert für den PI Regler **2** mittels des Sollwert-Multiplexer, welcher über „FSP070_Controller_1_2_InputSourceSelectionMultiplexer“ konfiguriert wird.
- Als Summand **2** des Addierers **1**, welcher über „FSP090_Adder_SourceSelectionMultiplexer“ konfiguriert wird.
- Als Quelle für den „ACU_SovereignitySelector_USBorIFK_SCU“ Block, dessen Ausgang als „SetValue_**B**_MultiplexedOutput“ Verwendung findet. Abhängig von „FSPFSP013_PeripheralConfig[0]“ wird entweder dieser Sollwert(1) oder der Sollwert des Kontrollsystems(0) via Backplane für „SetValue_**B**_MultiplexedOutput“ verwendet.

Name	**FSP039_SetValuePhysicalQuantities**
Adresse	**$0x27_H/39_D/0x3237_{ASCII}$**
Tiefe	2 Byte / 16 Bit
I/O	lesen / schreiben
Reset	$0x0030_H$

Gibt die physikalische Größe der Sollwerte an.

[15..12] (physikalische Einheit des Sollwertes D) obsolet

[11..8] (physikalische Einheit des Sollwertes C) obsolet

[7..4] physikalische Einheit des Sollwertes **B**

[3..0] physikalische Einheit des Sollwertes **A**

[3..0]	Einheit
0x0	Ampere (A)
0x1	Volt (V)
0x2	Temperatur (C)
0x3	Magnetische Flussdichte (T)
0x4	Magnetische Flussdichte (G)
0x5	n.u.
0x6	n.u.
0x7	n.u.
0x8	n.u.
0x9	n.u.
0xA	n.u.
0xB	n.u.
0xC	n.u.
0xD	n.u.
0xE	n.u.
0xF	n.u.

Name	**FSP045_AlteraRemoteUpdateCmd**
Adresse	**$0x2D_H/45_D/0x3244_{ASCII}$**
Tiefe	7 Byte / 56 Bit
I/O	lesen / schreiben
Reset	Reset:$0x00800000\_00\_0\_0\_0\_0_H$

Dieser FSP dient als Kommando FSP für die Altera Remote Update Funktion

Imagetyp lesen

Bit[4] = ‚0‘ (Read)
Bit[8] = ‚1‘ (steigende Flanke startet lesen des Imagetyps)
FSP046[1..0] enthält nun den aktuellen Imagetyp.

Imagetyp wechseln

Bit[4] = ‚1‘ (Write)
Bit[12] = ‚1‘ (steigende Flanke wechselt das Image)

[55..24] Flash Start Address (ab dieser Adresse wird das Image geschrieben)

[23..17] n.u.

[16] Reset WD Disable (only for debug)

[15..13] n.u.

[12] Start Write (steigende Flanke an diesem Bit startet die FSM zum Imagetyp-Wechsel)

[11..9] n.u.

[8] Start Read (steigende Flanke an diesem Bit startet die FSM zum lesen des Image-Type)

[7..5] n.u.

[4] Read_n_Write_Enable (muss ‚0‘ sein damit ‚Start Read‘ überhaupt ausgeführt wird, muss ‚1‘ sein damit ‚Start Write’überhaupt ausgeführt wird)

[3..2] n.u.

[1..0] Read Source

Name	**FSP046_AlteraRemoteUpdateStatus**
Adresse	**$0x2E_H/46_D/0x3245_{ASCII}$**
Tiefe	10 Byte / 80 Bit
I/O	lesen
Reset	Reset:0x(siehe Beschreibung)$_H$

Dieser FSP dient als Status FSP für die Altera Remote Update Funktion

[79..72] ReconfTriggerCondition

[71..69] Force Osc_int n.u.

[68] Force Osc_int

[67..44] Boot Address

[43..41] Wachdog Enable n.u.

[40] Wachdog Enable

[39..8] Wachdog timeout

[7..5] Cd_early n.u.

[4] Cd_early, wenn ‚1' ist ein gültiges Application-Image an der Bootadresse zu finden

[3..2] MSM State n.u.

[1..0] MSM State ('00' = Factory Image, '11' = Application Image)

Name	**FSP050_ModuleSupplyValues**
Adresse	**$0x32_H/50_D/0x3332_{ASCII}$**
Tiefe	16 Byte / 128 Bit
I/O	lesen
Reset	0x(siehe Beschreibung)

Liefert die vorzeichenbehafteten Betriebsspannungen des Moduls. Immer 2 Byte stehen für eine Spannung, d.h. bei 8 Spannungen ist dieses FSP 16 Byte tief. Die Spannungen sind vorzeichenbehaftet und wie folgt sortiert.

[127..112] vorzeichenbehaftete -12Volt (13 Bit)

[111..96] vorzeichenbehaftete Pad1 (13 Bit) (floatet, da Pad offen)

[95..80] vorzeichenbehaftete Volt (13 Bit)

[79..64] vorzeichenbehaftete Pad2 (13 Bit) (floatet, da Pad offen)

[63..48] vorzeichenbehaftete 5 Volt (13 Bit)

[47..32] vorzeichenbehaftete 3,3 Volt (13 Bit)

[31..16] vorzeichenbehaftete 2,5 Volt (13 Bit)

[15..0] vorzeichenbehaftete 1,2 Volt (13 Bit)

Die Bitwertigkeiten sind wie folgt:

bei 1V2, 2V5, 3V3 und 5V0 werden die gemessenen Werte direkt mit dem LSB Wert (SUP_LSB_Size = 0.002441) multipliziert und ergeben so die Spannung. Bei -12V0 und 12V0 erfolgt ebenfalls zuerst die LSB Wert Multiplikation und anschließend die Multiplikation mit dem Faktor 11, der dem vorgeschalteten Spannungsteiler entspricht. Die tatsächlich gemessenen Spannungen liegen nämlich bei 1/11 der Originalspannung.

Name	**FSP053_ModuleTemperatures**
Adresse	**$0x35_H/53_D/0x3335_{ASCII}$**
Tiefe	4 Byte / 32 Bit
I/O	lesen
Reset	$0x(\text{siehe Beschreibung})_H$

Je Temperatur 2 Byte ASCII, also je 1 Byte Vorzeichen behaftetes HEX Zeichen.

[31..24] Alarm_Info

[7..0]	Ausgang des Multiplexers
0x00	n.u., Resetzustand
0x01	Device **1** nicht bereit
0x02	Device **2** nicht bereit
0x04	Device **3** nicht bereit
0x08	DeviceSearchRunDone
0x10	Device 1 Grenze überschritten
0x20	Device 2 Grenze überschritten
0x40	Device 3 Grenze überschritten
0x80	Alarm Interrupt wenn Grenze bei einem Device überschritten

[23..16] Device **3**: Temperatur Modul Mitte (8 Bit)

[15..8] Device **2**: Temperatur Längsregler (8 Bit)

[7..0] Device **1**: Temperatur FPGA (8 Bit)

Die Schwellen der Temperaturgrenzen werden im „FSP054_ModuleTemperaturesComparationThresholds“ festgelegt.

Name	**FSP054_ModuleTemperaturesComparationThresholds**
Adresse	**$0x36_H/54_D/0x3336_{ASCII}$**
Tiefe	3 Byte / 24 Bit
I/O	lesen / schreiben
Reset	$0x46\_4646_H$

Stellt die Vergleichswerte zur Verfügung bei denen die Temperatursensoren Alarm auslösen sollen, sofern die Temperatur überschritten wurde.

Je Temperatur 2 Byte ASCII, also je 1 Byte Vorzeichen behaftetes HEX Zeichen.

Als Standardwert ist 70° Celsius ($70_D = 46_H$) gewählt.

[23..16] Vergleichswert Device **3**: Temperatur Modul Mitte (8 Bit)

[15..8] Vergleichswert Device **2**: Temperatur Längsregler (8 Bit)

[7..0] Vergleichswert Device **1**: Temperatur FPGA (8 Bit)

Name	**FSP058_ParameterChecksumValue**
Adresse	**$0x3A_H/58_D/0x3341_{ASCII}$**
Tiefe	3 Byte / 24 Bit
I/O	lesen / schreiben
Reset	$0x00\_0000_H$

Repräsentiert die Vergleichs-Prüfsumme der vom Module empfangenen Parameter. Dieser Wert dient zum Vergleich der im Modul errechneten Prüfsumme.

Die Modul-Prüfsumme wird dabei aus den empfangenen Datenbytes durch aufaddieren gebildet und abschließend mit dem Eintrag von „FSP058_ParameterChecksumValue“ verglichen.

[23..0] Prüfsumme der Datenübertragung vom PC zur MFU.

Die Prüfsumme wird im Modul ChecksumBuilder der Teil von mUISc (modular-USI-control) ist aus den Daten der beschriebenen FSP gebildet und abschließend mit dem Wert dieses FSP verglichen.

Name	**FSP059_ParameterChecksumValueCalculated**
Adresse	**$0x3B_H/59_D/0x3342_{ASCII}$**
Tiefe	3 Byte / 24 Bit
I/O	lesen
Reset	$0x(\text{siehe Beschreibung})_H$

Repräsentiert die errechnete Prüfsumme innerhalb des Moduls. Gibt die aktuell im Modul errechnete Prüfsumme zurück. Dadurch kann der Fortschritt der Prüfsummenbildung jederzeit verifiziert werden.

[23..0] errechnete Checksumme der Datenübertragung vom PC zur MFU.

Name	**FSP060_SlopeLimiter_C1**
Adresse	**$0x3C_H/60_D/0x3343_{ASCII}$**
Tiefe	30 Byte / 240 Bit
I/O	lesen / schreiben
Reset	0x00_000000_000000_000000_000000_000000_0000_0000_0000_0000_ $745D17\_8BA2E8_H$

Repräsentiert Parameter des Steilheitsbegrenzers.

Das Äquivalent für den Steilheitsbegrenzer für den Regler **2** findet sich im „FSP080_SlopeLimiter_C2“.

[239..234] n.u.

[233] Aktiviert den Steilheitbegrenzer,
wenn ‚1‘ UND ‚ACU_ControllerEnable‘

[232] Wenn ‚0‘ wird der Startwert des Steilheitbegrenzers mit 0 geladen, andernfalls der Wert des Eingangs ‚Dynamic_Threshold_RiseTime_Selection‘.
Dies ist i.d.R. ‚Controller1_ActValueMuxOut‘

[231..216] RampThresholdValue_**D** (16 Bit)

[215..200] RampThresholdValue_**C** (16 Bit)

[199..184] RampThresholdValue_**B** (16 Bit)

[183..168] RampThresholdValue_**A** (16 Bit)

[167..144] RampRiseTime_**E** (24 Bit)

[143..120] RampRiseTime_**D** (24 Bit)

[119..96] RampRiseTime_**C** (24 Bit)

[95..72] RampRiseTime_**B** (24 Bit)

[71..48] RampRiseTime_**A** (24 Bit)

[47..24] Obere Grenze (24 Bit)

[23..0] Untere Grenze (24 Bit)

Name	**FSP061_DifferenceCalculatorMultiplier**
Adresse	**$0x3D_H/61_D/0x3631_{ASCII}$**
Tiefe	6 Byte / 48 Bit
I/O	lesen / schreiben
Reset	$0x03E8\_03E8\_03E8_H$

Repräsentiert den Multiplikator für den Multiplikand Delta I der Regelabweichung der Regler **1** und **2**, bzw. den Multiplikand Delta I der Differenzen zwischen Istwert Regler **1** und Istwert Regler **2**.

[47..43] n.u.

[42..32] bis MFU_SE_7_2: Multiplikator (11 Bit) für
den Differenzbilder Regler **1** Istwert zu Regler **2** Istwert
ab MFU_SE_7_3: n.u.

[31..27] n.u.

[26..16] Multiplikator (11 Bit) für den Differenzbilder Regler **2**

[15..11] n.u.

[10..0] Multiplikator (11 Bit) für den Differenzbilder Regler **1**

Name	**FSP062_LocalSetValue**
Adresse	**$0x3E_H/62_D/0x3632_{ASCII}$**
Tiefe	3 Byte / 24 Bit
I/O	lesen / schreiben
Reset	$0x000000_H$

Repräsentiert den lokalen (Hand) StromSOLLwert.

[23..0] Vorzeichenbehafteter 20 Bit Wert (0xFFFFF bis 0x7FFFF), wird automatisch dem Sollwert_**A** zugewiesen, wenn der Schalter Local/Remote auf Local geschaltet ist. Die Bits[3..0] sind immer ‚0'.

Name	**FSP063_MPS**
Adresse	**$0x3F_H/63_D/0x3346_{ASCII}$**
Tiefe	7 Byte / 56 Bit
I/O	lesen/schreiben
Reset	$0x00\_0000\_0000\_0000_H$

With this FSP it is possible to configure the MPS (Machine Protection System) comparator used to generate the MPS_OUT signal sent via LEMO.

[55..53] n.u.

[52] CtrlSel_1L_2H: it selects one of the two available PI controller outputs as value to compare in input to the hysteresis comparator. =1 => controller2 output selected; =0=> controller1 output selected.

[51..49] n.u.

[48] MPS_outSel_DIL_DITH: it selects between the hysteresis comparator output (=0) and the OR combination of the same signal with the tripline (=1) as value of MPS_OUT signal.

[47..24] Histeresys comparator OFF threshold.(20 Bit)

[23..0] Histeresys comparator ON threshold (20 Bit)

Name	**FSP064_USIxHS_Multiplexer**
Adresse	**$0x40_H/64_D/0x3634_{ASCII}$**
Tiefe	13 Byte / 103 Bit
I/O	lesen / schreiben
Reset	$0x00\_0000\_0000\_0000\_0000\_0000\_0000_H$

Zur Nutzung der USI HighSpeed Transferkanäle darf an eine USI nur maximal ein Modul angeschlossen sein und dieses muss das USI HighSpeed Protokoll unterstützen.

Jede USI unterstützt genau einen USI HighSpeed-Transferkanal in eine Datenrichtung. D.h. ein HighSpeed-Transferkanal von der MFU zum angeschlossene Module mit HighSpeed Unterstützung und ein HighSpeed-Transferkanal vom angeschlossenen Modul mit HighSpeed-Unterstützung zur MFU.

Die Übertragung eines vollständigen Strings benötigt bei maximaler USI Bitrate von 20MBit 6us.

Mit diesem FSP ist eine Vielzahl von Multiplexer parametrierbar, die ein umfangreiches „routen“ von USI HighSpeed Transferkanal Daten von verschiedenen Quellen an verschiedene Ziele ermöglicht.

<u>Die Betrachtung ist dabei wie folgt:</u>

GEHEND[TO] die HighSpeed Daten gehen von der MFU an ein Modul

MFU interne Signale lassen sich hiermit auf USI HighSpeed-Transferkanäle legen. Es können Signale ausgewählt werden, welche via HighSpeed Transferkanal von der MFU über die gewählte USI an das daran angeschlossene Modul mit HighSpeed-Unterstützung übertragen werden.

KOMMEND[FROM] die HighSpeed Daten kommen von einem Modul zur MFU

Hiermit wird definiert von welchem Modul mit HighSpeed-Unterstützung welches Signal in die MFU zurück gelangt. Die Signale dieser USI HighSpeed-Transferkanäle müssen zuvor innerhalb des sendenden Moduls auf den USI HighSpeed-Transferkanal gelegt werden. Jedes Modul hat ggf. standardisierte Daten die im HighSpeed-Transferkanal übertragen werden. Der ADC wird z.B. immer ADC Werte im USI HighSpeed-Transferkanal senden, das ICM immer Netzgeräte-Status-Informationen. Sofern nicht alle Bits des USI HighSpeed-Transferkanals belegt sind ist es möglich wahlweise auf freie Bits zusätzliche Datensignale eines Moduls mittels eines Multiplexers zu legen. Diese zurück gesendeten Daten werden MFU intern wiederrum auf bestimmte Signalwege aufgeschaltet, die für eine einwandfreie Funktion des ACU Gesamtsystems notwendig sind.

Kommende [FROM] Signale werden dabei teilweise noch über einen weiteren nachgeschalteten Multiplexer geführt um deren Wertigkeiten und Inhalte weiter zu verfeinern.

[103..100] ControllerX_PreSelectionMux_**2** KOMMEND[FROM]

[99..96] ControllerX_PreSelectionMux_**1** KOMMEND[FROM]

[95..92] n.u.

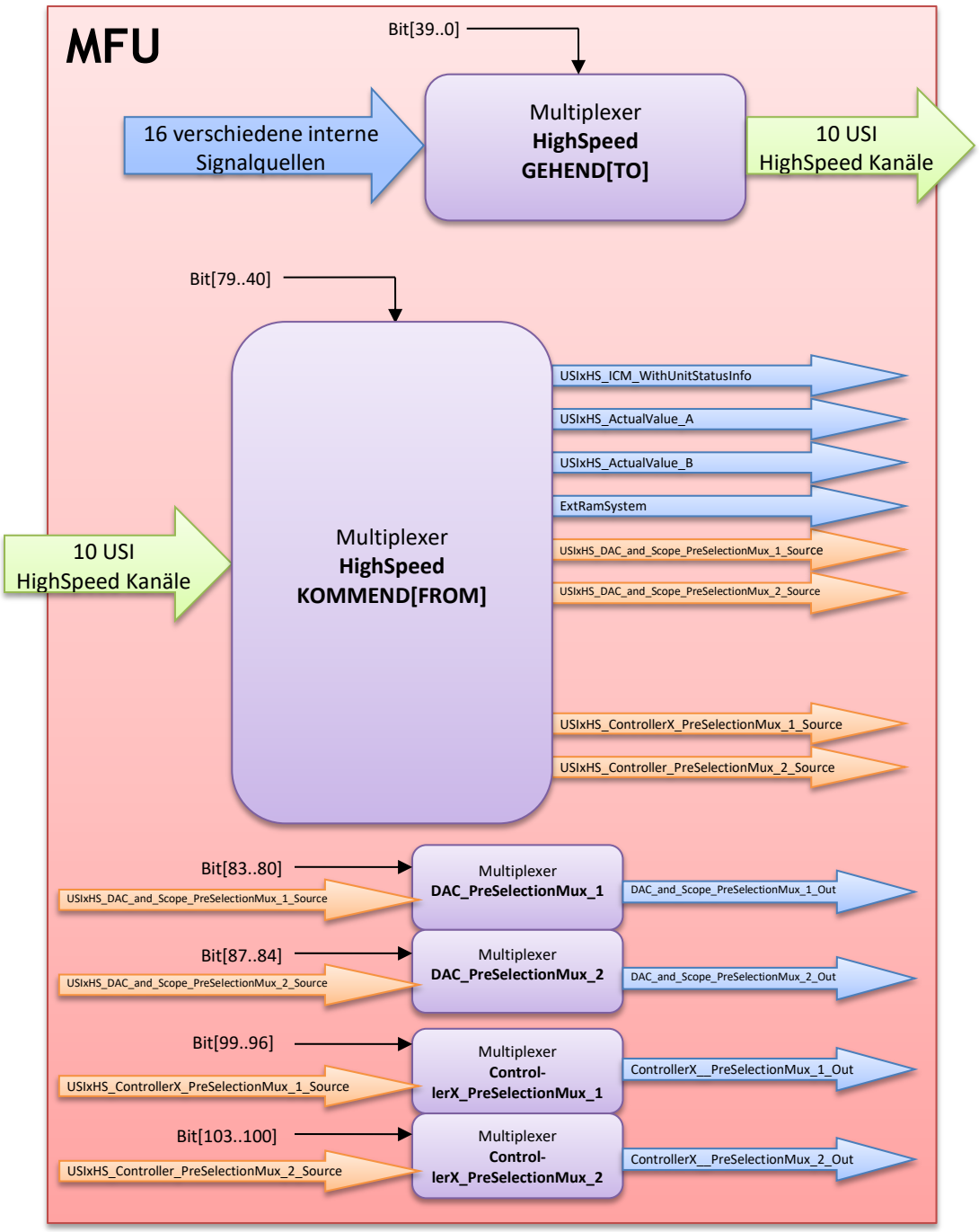
[91..88] n.u.

[87..84] DAC_and_Scope_PreSelectionMux_**2** KOMMEND[FROM]

[83..80] DAC_and_Scope_PreSelectionMux_**1** KOMMEND[FROM]

[79..40] Multiplexer HighSpeed Daten KOMMEND[FROM]

[39..0] Multiplexer HighSpeed GEHEND[TO]



PreSelectionMux ermöglichen beliebige USI HighSpeed Kanäle von den jeweiligen Quellen zum Ziel zu leiten und dabei eine Vorauswahl zu treffen, welche Bit des empfangenen Wertes dabei am Ziel wirklich ankommen sollen.

[103..100] ControllerX_PreSelectionMux_**2**

Über 4 Bits werden die Daten am Eingang dieses Multiplexers einem Ausgang (ControllerX_PreSelectionMux_**2**_Out[19..0]) zugewiesen.

[3..0]	Daten am Eingang
0	n.u
1	USIxHS_ControllerX_PreSelectionMux_**2**_Source[31..12]
2	USIxHS_ControllerX_PreSelectionMux_**2**_Source[31..18],GND_BUS[5..0]
3	USIxHS_ControllerX_PreSelectionMux_**2**_Source[17..4],GND_BUS[5..0]
4	GND_BUS[19..0]
5	GND_BUS[19..0]

6	GND_BUS[19..0]
7	GND_BUS[19..0]
8	GND_BUS[19..0]
9	GND_BUS[19..0]
A	GND_BUS[19..0]
B	GND_BUS[19..0]
C	GND_BUS[19..0]
D	GND_BUS[19..0]
E	GND_BUS[19..0]
F	GND_BUS[19..0]

[99..96] ControllerX_PreSelectionMux_**1**

Über 4 Bits werden die Daten am Eingang dieses Multiplexers einem Ausgang (ControllerX_PreSelectionMux_**1**_Out[19..0]) zugewiesen.

[3..0]	Daten am Eingang
0	n.u.
1	USIxHS_ControllerX_PreSelectionMux_**1**_Source[31..12]
2	USIxHS_ControllerX_PreSelectionMux_**1**_Source[31..18],GND_BUS[5..0]
3	USIxHS_ControllerX_PreSelectionMux_**1**_Source[17..4],GND_BUS[5..0]
4	GND_BUS[19..0]
5	GND_BUS[19..0]
6	GND_BUS[19..0]
7	GND_BUS[19..0]
8	GND_BUS[19..0]
9	GND_BUS[19..0]
A	GND_BUS[19..0]
B	GND_BUS[19..0]
C	GND_BUS[19..0]
D	GND_BUS[19..0]
E	GND_BUS[19..0]
F	GND_BUS[19..0]

[95..92] n.u.

[91..88] n.u.

[87..84] DAC_and_Scope_PreSelectionMux_**2**

Über 4 Bits werden die Daten am Eingang dieses Multiplexers einem Ausgang (DAC_and_Scope_PreSelectionMux_**2**_Out[19..0]) zugewiesen.

[3..0]	Daten am Eingang
0	n.u.
1	USIxHS_DAC_ and_Scope_PreSelectionMux_**2**_Source[31..12]
2	USIxHS_DAC_ and_Scope_PreSelectionMux_**2**_Source[31..18],GND_BUS[5..0]
3	USIxHS_DAC_ and_Scope_PreSelectionMux_**2**_Source[17..4],GND_BUS[5..0]
4	GND_BUS[19..0]
5	GND_BUS[19..0]
6	GND_BUS[19..0]
7	GND_BUS[19..0]

8	GND_BUS[19..0]
9	GND_BUS[19..0]
A	GND_BUS[19..0]
B	GND_BUS[19..0]
C	GND_BUS[19..0]
D	GND_BUS[19..0]
E	GND_BUS[19..0]
F	GND_BUS[19..0]

[83..80] DAC_PreSelectionMux_**1**

Über 4 Bits werden die Daten am Eingang dieses Multiplexers einem Ausgang (DAC_PreSelectionMux_**1**_Out[19..0]) zugewiesen.

[3..0]	Daten am Eingang
0	n.u.
1	USIxHS_DAC_PreSelectionMux_**1**_Source[31..12]
2	USIxHS_DAC_PreSelectionMux_**1**_Source[31..18],GND_BUS[5..0]
3	USIxHS_DAC_PreSelectionMux_**1**_Source[17..4],GND_BUS[5..0]
4	GND_BUS[19..0]
5	GND_BUS[19..0]
6	GND_BUS[19..0]
7	GND_BUS[19..0]
8	GND_BUS[19..0]
9	GND_BUS[19..0]
A	GND_BUS[19..0]
B	GND_BUS[19..0]
C	GND_BUS[19..0]
D	GND_BUS[19..0]
E	GND_BUS[19..0]
F	GND_BUS[19..0]

[79..40] Mux_16to10_DataFROMHighSpeedPorts

Dieser Multiplexer hat 16 Eingänge, die in Abhängigkeit der Bitnibbel 10 Ausgängen zugeordnet werden. Die Daten an diesen Ausgängen entsprechen den empfangenen HighSpeedDaten der zugehörigen USI. Z.B. Eingang 2 ist der HighSpeed Empfangskanal von USI 2.

[79..76] legt fest, welcher HighSpeedPort_x_RcvdData an den Ausgang **10** gelegt wird

[75..72] legt fest, welcher HighSpeedPort_x_RcvdData an den Ausgang **9** gelegt wird

[71..68] legt fest, welcher HighSpeedPort_x_RcvdData an den Ausgang **8** gelegt wird

[67..64] legt fest, welcher HighSpeedPort_x_RcvdData an den Ausgang **7** gelegt wird

[63..60] legt fest, welcher HighSpeedPort_x_RcvdData an den Ausgang **6** gelegt wird

[59..56] legt fest, welcher HighSpeedPort_x_RcvdData an den Ausgang **5** gelegt wird

[55..52] legt fest, welcher HighSpeedPort_x_RcvdData an den Ausgang **4** gelegt wird

[51..48] legt fest, welcher HighSpeedPort_x_RcvdData an den Ausgang **3** gelegt wird

[47..44] legt fest, welcher HighSpeedPort_x_RcvdData an den Ausgang **2** gelegt wird

[43..40] legt fest, welcher HighSpeedPort_x_RcvdData an den Ausgang **1** gelegt wird

Nachfolgend sind die Belegungen der Multiplexereingänge aufgeführt

DataIn	Daten am Eingang

1	HighSpeedPort_**1**_RcvdData [31..0]
2	HighSpeedPort_**2**_RcvdData [31..0]
3	HighSpeedPort_**3**_RcvdData [31..0]
4	HighSpeedPort_**4**_RcvdData [31..0]
5	HighSpeedPort_**5**_RcvdData [31..0]
6	HighSpeedPort_**6**_RcvdData [31..0]
7	HighSpeedPort_**7**_RcvdData [31..0]
8	HighSpeedPort_**8**_RcvdData [31..0]
9	HighSpeedPort_**9**_RcvdData [31..0]
10	HighSpeedPort_**10**_RcvdData [31..0]

Die Ausgänge erwarten folgende Infomationen

DataOut	Daten am Ausgang
1	USIxHS_ICM_WithUnitStatusInfo[31..0]
2	USIxHS_ActualValue_A[31..0]
3	USIxHS_ActualValue_B[31..0]
4	ExtRamSystem[31..0]
5	USIxHS_DAC_and_Scope_PreSelectionMux_**1**_Source[31..0]
6	USIxHS_DAC_and_Scope_PreSelectionMux_**2**_Source[31..0]
7	n.u.
8	n.u.
9	USIxHS_ControllerX_PreSelectionMux_**1**_Source[31..0]
10	USIxHS_ControllerX_PreSelectionMux_**2**_Source[31..0]

An den Bits[79..40] werden die Eingänge (DataIn) für die jeweiligen Ausgänge (DataOut) des Multiplexers gewählt.

Nachfolgend ist aufgeführt welche Bitkombination den jeweiligen Eingang auf welchen Ausgang legt.

Bitkombination	Zuweisung (Daten am Ausgang)
0x0	Eingang **1**[31..0]
0x1	Eingang **2**[31..0]
0x2	Eingang **3**[31..0]
0x3	Eingang **4**[31..0]
0x4	Eingang **5**[31..0]
0x5	Eingang **6**[31..0]
0x6	Eingang **7**[31..0]
0x7	Eingang **8**[31..0]
0x8	Eingang **9**[31..0]
0x9	Eingang**10**[31..0]
0xA..0xF	Ausgang ist 0

Beispiel:

Die Bitkombination (Ausgang n) → [(n*4)-1..(n-4)] legt fest welcher Eingang (HighSpeed-Port_x_RcvdData [31..0]) auf den Ausgang gelegt wird.

Sollen die Daten von Eingang **2** an Ausgang **2** müssen die Bit 47.4.4 auf 1 stehen

[(2 * 4)-1 .. (2 * 4) -4] = [47..44]

Sollen die Daten von Eingang **9** an Ausgnag **6** müssen die Bit 63..60 auf 8 stehen

[(6 * 4)-1 .. (6 * 4) -4] = [63..60]

[39..0] Mux_16to10_DataTOHighSpeedPorts

Dieser Multiplexer hat 16 Eingänge, die in Abhängigkeit der Bitnibbel 10 Ausgängen zugeordnet werden. Die Daten an diesen Ausgängen entsprechen den zu sendenden HighSpeedDaten der zugehörigen USI. Z.B. Ausgang **2** ist der HighSpeed Sendekanal von USI **2**.

[39..36] legt fest, welcher Eingang an HighSpeedPort_**10**_TransferData gelegt wird

[35..32] legt fest, welcher Eingang an HighSpeedPort_**9**_TransferData gelegt wird

[31..28] legt fest, welcher Eingang an HighSpeedPort_**8**_TransferData gelegt wird

[27..24] legt fest, welcher Eingang an HighSpeedPort_**7**_TransferData gelegt wird

[23..20] legt fest, welcher Eingang an HighSpeedPort_**6**_TransferData gelegt wird

[19..16] legt fest, welcher Eingang an HighSpeedPort_**5**_TransferData gelegt wird

[15..12] legt fest, welcher Eingang an HighSpeedPort_**4**_TransferData gelegt wird

[11..8] legt fest, welcher Eingang an HighSpeedPort_**3**_TransferData gelegt wird

[7..4] legt fest, welcher Eingang an HighSpeedPort_**2**_TransferData gelegt wird

[3..0] legt fest, welcher Eingang an HighSpeedPort_**1**_TransferData gelegt wird

Nachfolgend sind die Belegungen der Multiplexereingänge aufgeführt

DataIn	Daten am Eingang
1	[31..5] *GND_BUS[26..0]* [4] *IGBT_V5_Active* [3..0] *Command[3..0]*
2	[31..12] *ERLOutValue[19..0]* [11..5] *GND_BUS[6..0]* [4] *IGBT_V5_Active* [3..0] *Command[3..0]*
3	[31..12] *Controller**1**_SetValueMuxOut[19..0]* [11..5] *GND_BUS[6..0]* [4] *IGBT_V5_Active,* [3..0] *Command[3..0]*
4	[31..12] *Controller**2**_SetValueMuxOut[19..0]* [11..5] *GND_BUS[6..0]* [4] *IGBT_V5_Active* [3..0] *Command[3..0]*
5	[31..13] *GND_BUS[31..13]* [12..9] *HighSpeedPort_**3**_Out[3..0]* [8..4] *MUX_DesignatedLoad[4..0]* [3..0] *Command[3..0]*
6	[31..12] *SetValue_**A**_MuxOut[19..0]* [11..5] *GND_BUS[6..0]* [4] *IGBT_V5_Active* [3..0] *Command[3..0]*
7	[31..12] *SetValue_**B**_MuxOut[19..0]* [11..5] *GND_BUS[6..0]* [4] *IGBT_V5_Active* [3..1] *Command[3..0]*
8	[31..0] *USIxHS_ICM_WithUnitStatusInfo[31..0]*
9	[31..18] *Controller**1**_SetValueDeviation[19..6]* [17..4] *Controller**1**_ActValMuxOut[19..6]* [3..0] *Command[3..0]*
10	[31..18] *SlopeLimiterOut_C**1**[19..6]* [17..4] *Controller**1**_ActValMuxOut[19..6]* [3..0] *Command[3..0]*

11	[31..18] *ERLOutValue[19..6]* [17..5] *Controller**1**_SetValueDeviation [19..7]* [4] *IGBT_V5_Active,* [3..0] *Command[3..0]*
12	[31..18] *SlopeLimiterOut_C**2**[19..6]* [17..4] *Controller**2**_ActValMuxOut[19..6]* [3..0] *Command[3..0]*
13	[31..18] *Controller**1**_ActValMuxOut [19..6]* [17..4] *Controller**2**_ActValMuxOut [19..6]* [3..0] *Command[3..0]*
14	[31..8] *FSP20_ActualValue_**A**[23..0]* [7..5] *GND_BUS[2..0]* [4] *IGBT_V5_Active* [3..0] *Command[3..0]*
15	n.u.
16	n.u.

Nachfolgend ist aufgeführt welche Bitkombination den jeweiligen Eingang auf welchen Ausgang legt.

Bitkombination	Zuweisung (Daten am Ausgang)
0x0	Eingang **1**[31..0]
0x1	Eingang **2**[31..0]
0x2	Eingang **3**[31..0]
0x3	Eingang **4**[31..0]
0x4	Eingang **5**[31..0]
0x5	Eingang **6**[31..0]
0x6	Eingang **7**[31..0]
0x7	Eingang **8**[31..0]
0x8	Eingang **9**[31..0]
0x9	Eingang**10**[31..0]
0xA	Eingang**11**[31..0]
0xB	Eingang**12**[31..0]
0xC	Eingang**13**[31..0]
0xD	Eingang**14**[31..0]
0xE	Eingang**15**[31..0]
0xF	Eingang**16**[31..0]

Beispiel:

Die Bitkombination (USI n) → [(n*4)-1..(n-4)] legen fest welcher Eingang auf den Ausgang (HighSpeedPort_x_TransferData) gelegt wird.

Sollen die Daten von Eingang **2** an HighSpeedPort_**2**_TransferData (USI2) müssen die Bit 7..4 auf 1 stehen

[(2 * 4)-1 .. (2 * 4) -4] = [7..4]

Sollen die Daten von Eingang **9** an HighSpeedPort_**6**_TransferData (USI6) müssen die Bit 23..20 auf 8 stehen

[(6 * 4)-1 .. (6 * 4) -4] = [23..20]

Name	**FSP065_FrontLemoMultiplexer**
Adresse	**$0x41_H/65_D/0x3431_{ASCII}$**
Tiefe	2 Byte / 16 Bit
I/O	lesen / schreiben
Reset	$0x17\_17_H$

Repräsentiert die Einstellungen der Ausgangsmultiplexer für die Front-Lemo-Ausgangsbuchsen.

[15..13] n.u.

[12..8] Quellenwahl für Front LEMO Ausgang **2** (X2) (5 Bit)

[4..0]	Ausgang des Multiplexers
0x00	intScopeTriggerEventDetected
0x01	intScopeTimeBasePulse
0x02	ControllerStatusBit[0] (GND)
0x03	ControllerStatusBit[1] (GND)
0x04	ControllerStatusBit[2] (GND)
0x05	ControllerStatusBit[3] (GND)
0x06	ControllerStatusBit[4] (GND)
0x07	ControllerStatusBit[5] (GND)
0x08	ControllerStatusBit[6] (GND)
0x09	ControllerStatusBit[7] (ACU_Controller_Enable)
0x0A	ControllerStatusBit[8] (CurrentEqual)
0x0B	ControllerStatusBit[9] (IGBT_V5_Active)
0x0C	ControllerStatusBit[10] (Controller2_IGBT_V5_Active)
0x0D	ControllerStatusBit[11] (Controller1_IGBT_V5_Active)
0x0E	ControllerStatusBit[12] (PWM_FDRIVE_Comparator_MuxOut)
0x0F	ControllerStatusBit[13] (Controller2_PWM_FDRIVE_Comparator)
0x10	ControllerStatusBit[14] (Controller1_PWM_FDRIVE_Comparator)
0x11	ControllerStatusBit[15] (Degaussing_Active)
0x12	ControllerStatusBit[16] (Controller2_P2_Part_Active)
0x13	ControllerStatusBit[17] (Controller1_P2_Part_Active)
0x14	ControllerStatusBit[18] (Controller2_I_Part_Disable)
0x15	ControllerStatusBit[19] (Controller1_I_Part_Disable)
0x16	1 (VCC)
0x17	0 (GND)
0x18	MPS
0x19	n.u.
0x1A	n.u.
0x1B	n.u.
0x1C	n.u.
0x1D	n.u.
0x1E	n.u.
0x1F	n.u.

[7..5] n.u.

[4..0] Quellenwahl für Front LEMO Ausgang **1** (X1) (5 Bit)

[4..0]	Ausgang des Multiplexers
0x00	intScopeTriggerEventDetected

0x01	intScopeTimeBasePulse
0x02	ControllerStatusBit[0] (GND)
0x03	ControllerStatusBit[1] (GND)
0x04	ControllerStatusBit[2] (GND)
0x05	ControllerStatusBit[3] (GND)
0x06	ControllerStatusBit[4] (GND)
0x07	ControllerStatusBit[5] (GND)
0x08	ControllerStatusBit[6] (GND)
0x09	ControllerStatusBit[7] (ACU_Controller_Enable)
0x0A	ControllerStatusBit[8] (CurrentEqual)
0x0B	ControllerStatusBit[9] (IGBT_V5_Active)
0x0C	ControllerStatusBit[10] (Controller**2**_IGBT_V5_Active)
0x0D	ControllerStatusBit[11] (Controller**1**_IGBT_V5_Active)
0x0E	ControllerStatusBit[12] (PWM_FDRIVE_Comparator_MuxOut)
0x0F	ControllerStatusBit[13] (Controller**2**_PWM_FDRIVE_Comparator)
0x10	ControllerStatusBit[14] (Controller**1**_PWM_FDRIVE_Comparator)
0x11	ControllerStatusBit[15] (Degaussing_Active)
0x12	ControllerStatusBit[16] (Controller**2**_P**2**_Part_Active)
0x13	ControllerStatusBit[17] (Controller**1**_P**2**_Part_Active)
0x14	ControllerStatusBit[18] (Controller**2**_I_Part_Disable)
0x15	ControllerStatusBit[19] (Controller**1**_I_Part_Disable)
0x16	1 (VCC)
0x17	0 (GND)
0x18	MPS
0x19	n.u.
0x1A	n.u.
0x1B	n.u.
0x1C	n.u.
0x1D	n.u.
0x1E	n.u.
0x1F	n.u.

Name	**FSP067_Defined_USI**
Adresse	**$0x43_H/67_D/0x3433_{ASCII}$**
Tiefe	2 Byte / 16 Bit
I/O	lesen / schreiben
Reset	0x0000H

This FSP is used to enable the High Speed data received/sent from/to MFU. It is written during the power converter setup and it contains the information of which USI is defined in the configuration file.

[15..10] Not used.

[9] Enable High Speed data for USI**10** (active high)

[8] Enable High Speed data for USI**9** (active high)

[7] Enable High Speed data for USI**8** (active high)

[6] Enable High Speed data for USI**7** (active high)

[5] Enable High Speed data for USI**6** (active high)

[4] Enable High Speed data for USI**5** (active high)

[3] Enable High Speed data for USI**4** (active high)

[2] Enable High Speed data for USI**3** (active high)

[1] Enable High Speed data for USI**2** (active high)

[0] Enable High Speed data for USI**1** (active high)

Name	**FSP068_ButtonAndLEMOInStatus**
Adresse	**$0x44_H/68_D/0x3434_{ASCII}$**
Tiefe	1 Byte / 8 Bit
I/O	Lesen
Reset	$0x(\text{siehe Beschreibung})_H$

Stellt die Status der Tasten, Schalter und LEMO Eingänge zur Verfügung.

[7] Status LEMO_IN[2] (X**4**)

[6] Status LEMO_IN[1] (X**3**)

[5] n.u., immer ‚0‘

[4] n.u., immer ‚0‘

[3] Status Schalter REMOTE/LOCAL, ‚1‘ = REMOTE

[2] Status RESET Taste, ‚1‘ = gedrückt

[1] Status OFF Taste, ‚1‘ = gedrückt

[0] Status ON Taste, ‚1‘ = gedrückt

Name	**FSP069_ExternalTriplinesStatus**
Adresse	**$0x45_H/69_D/0x3435_{ASCII}$**
Tiefe	4 Byte / 32 Bit
I/O	Lesen
Reset	$0x(\text{siehe Beschreibung})_H$

Stellt die Status der externen Reißleinen zur Verfügung.

[31..26] n.u., immer ‚0‘

[25..16] ExtTripLinesPulled_n, tatsächlicher Zustand der externen Reißleinen. Eine Reißleine gilt als geschlossen, sofern das zugehörige Bit gesetzt (‚1‘) ist.

[15.10] n.u., immer ‚0‘

[9..0] ExtTripLinesPulled_nMem, gespeicherter Zustand der externen Reisleinen. Eine Reißleine gilt als geschlossen, sofern das zugehörige Bit gesetzt (‚1‘) ist. Wird eine Reißleine geöffnet wird sofort der Zustand aller Reißleinen gespeichert und hier ausgegeben. Die Reißleine die zuerst ausgelöst wurde ist ‚0‘, alle übrigen Bits blieben unverändert. Erst wenn alle Reißleinen wieder geschlossen sind, wird die Ausgabe von , ExtTripLinesPulled_nMem‘ sich wieder auf ‚0x11_1111_1111‘ ändern.

Name	**FSP070_Controller_1_2_InputSourceSelectionMultiplexer**
Adresse	**$0x46_H/70_D/0x3436_{ASCII}$**
Tiefe	3 Byte / 24 Bit
I/O	lesen / schreiben
Reset	$0x00\_0\_0\_0\_0_H$

Repräsentiert die Einstellungen der PI Regler **1** und **2** Eingangsmultiplexer für dessen Soll- und Istwert-quelle.

[23..20] n.u.

[19] '1' Ausgangswert des Regler **2** Istwertquellen-Multiplexers wird invertiert

[18] '1' Ausgangswert des Regler **2** Sollwertquellen-Multiplexers wird invertiert

[17] '1' Ausgangswert des Regler **1** Istwertquellen-Multiplexers wird invertiert

[16] '1' Ausgangswert des Regler **1** Sollwertquellen-Multiplexers wird invertiert

[15..12] Regler **2** Quellenwahl für das Istwertquellen-Multiplexer-Ausgangssignal (4 Bit)

[3..0]	Ausgang des Multiplexers
0x0	0
0x1	Istwert **A**, FSP020_ActualValue_A
0x2	Istwert **B**, FSP021_ActualValue_B
0x3	Ausgang interner Funktionsgenerator
0x4	Summand **1** für den Addierer, bestimmt durch "FSP090_Adder_SourceSelectionMultiplexer[8][3..0]"
0x5	Controller**1**_SetValueDeviation, analog FSP076_Controller_1_SetValueDeviation
0x6	Controller**X**_PreSelectionMux_**1**_Out, bestimmt durch FSP064_USIxHS_Multiplexer[99..96]
0x7	Controller**X**_PreSelectionMux_**2**_Out, bestimmt durch FSP064_USIxHS_Multiplexer[103..100]
0x8	Sollwert **A**, FSP030_SetValue_A
0x9	GND_BUS
0xA	GND_BUS
0xB	GND_BUS
0xC	GND_BUS
0xD	GND_BUS
0xE	GND_BUS
0xF	GND_BUS

[11..8] Regler **2** Quellenwahl für Sollwertquellen-Multiplexer-Ausgangssignal (4 Bit)

[3..0]	Ausgang des Multiplexers
0x0	0
0x1	Sollwert **A**, bestimmt durch „FSP013_PeripheralConfig[0]"
0x2	Sollwert **B**, bestimmt durch „FSP013_PeripheralConfig[0]"
0x3	Sollwert **A**, FSP030_SetValue_A
0x4	Ausgang interner Funktionsgenerator
0x5	Summand **1** für den Addierer, bestimmt durch "FSP090_Adder_SourceSelectionMultiplexer[16][3..0]"
0x6	Controller**1**_SetValueDeviation, analog FSP076_Controller_1_SetValueDeviation
0x7	Controller**X**_PreSelectionMux_**1**_Out, bestimmt durch

	FSP064_USIxHS_Multiplexer[99..96]
0x8	Controller**X**_PreSelectionMux_**2**_Out, bestimmt durch FSP064_USIxHS_Multiplexer[103..100]
0x9	Controller**1**_SetValueMuxOut[19..0], bestimmt durch FSP070_Controller_1_2_InputSourceSelectionMultiplexer[3..0]
0xA	Controller**1**_PI_Part_Output[19..0]
0xB	GND_BUS
0xC	GND_BUS
0xD	GND_BUS
0xE	GND_BUS
0xF	GND_BUS

[7..4] Regler **1** Quellenwahl für das Istwertquellen-Multiplexer-Ausgangssignal (4 Bit)

[3..0]	Ausgang des Multiplexers
0x0	0
0x1	Istwert **A**, bestimmt durch FSP020_ActualValue_A
0x2	Istwert **B**, bestimmt durch FSP021_ActualValue_B
0x3	Ausgang interner Funktionsgenerator
0x4	Controller**X**_PreSelectionMux_**1**_Out, bestimmt durch FSP064_USIxHS_Multiplexer[99..96]
0x5	Controller**X**_PreSelectionMux_**2**_Out, bestimmt durch FSP064_USIxHS_Multiplexer[103..100]
0x6	Sollwert **A**, FSP030_SetValue_A
0x7	GND_BUS
0x8	GND_BUS
0x9	GND_BUS
0xA	GND_BUS
0xB	GND_BUS
0xC	GND_BUS
0xD	GND_BUS
0xE	GND_BUS
0xF	GND_BUS

[3..0] Regler **1** Quellenwahl für Sollwertquellen-Multiplexer-Ausgangssignal (4 Bit)

[3..0]	Ausgang des Multiplexers
0x0	0
0x1	Sollwert **A**, bestimmt durch „FSP013_PeripheralConfig[0]“
0x2	Sollwert **B**, bestimmt durch „FSP013_PeripheralConfig[0]“
0x3	Ausgang interner Funktionsgenerator
0x4	Controller**X**_PreSelectionMux_**1**_Out, bestimmt durch FSP064_USIxHS_Multiplexer[99..96]
0x5	Controller**X**_PreSelectionMux_**2**_Out, bestimmt durch FSP064_USIxHS_Multiplexer[103..100]
0x6	Sollwert **A**, FSP030_SetValue_A
0x7	GND_BUS
0x8	GND_BUS
0x9	GND_BUS
0xA	GND_BUS
0xB	GND_BUS

0xC	GND_BUS
0xD	GND_BUS
0xE	GND_BUS
0xF	GND_BUS

Name	**FSP071_Controller_1_SetValue**
Adresse	**$0x47_H/71_D/0x3731_{ASCII}$**
Tiefe	3 Byte / 24 Bit
I/O	lesen
Reset	$0x(\text{siehe Beschreibung})_H$

Über diesen FSP kann der Sollwert des PI Regler **1** ausgelesen werden. Der Sollwert wird dabei über den Multiplexer „FSP070_Controller_1_2_InputSourceSelectionMultiplexer“ ausgewählt.

Das Äquivalent für Regler **2** ist der „FSP081_Controller_2_SetValue“.

[23..0] Vorzeichenbehafteter 20 Bit Sollwert des PI Regler **1**, die Bits[3..0] sind immer ‚0‘.

Name	**FSP072_Controller_1_ActualValue**
Adresse	**$0x48_H/72_D/0x3438_{ASCII}$**
Tiefe	3 Byte / 24 Bit
I/O	lesen
Reset	$0x(\text{siehe Beschreibung})_H$

Über diesen FSP kann der Istwert des PI Regler **1** ausgelesen werden. Der Istwert wird dabei über den Multiplexer „FSP070_Controller_1_2_InputSourceSelectionMultiplexer“ ausgewählt.

Das Äquivalent für Regler **2** ist der „FSP082_Controller_2_ActualValue“.

[23..0] vorzeichenbehafteter 20 Bit Istwert des PI Regler **1**, die Bits[3..0] sind immer ‚0‘.

Name	**FSP073_Controller_1_Limits**
Adresse	**$0x49_H/73_D/0x3439_{ASCII}$**
Tiefe	6 Byte / 48 Bit
I/O	lesen / schreiben
Reset	$0x000000\_000000_H$

Über diesen FSP können die Bereichsgrenzen des PI Regler **1** festgelegt werden.

Das Äquivalent für Regler **2** ist der „FSP083_Controller_2_Limits“.

[47..24] Controller_**1**_MaxVal (20 Bit), repräsentiert den oberen (maximalen) Grenzwert des PI Regler **1**, die Bits[3..0] sind immer ‚0‘.

[23..0] Controller_**1**_MinVal (20 Bit), repräsentiert den unteren (minimalen) Grenzwert des PI Regler **1**, die Bits[3..0] sind immer ‚0‘.

Name	**FSP074_Controller_1_PI_Settings**
Adresse	**$0x4A_H/60_D/0x3441_{ASCII}$**
Tiefe	13 Byte / 104 Bit
I/O	lesen / schreiben
Reset	$0x00\_00000000\_00000000\_00000000_H$

Über diesen FSP können die I und P1, P2 Anteile des Regler **1** gesetzt werden.

Das Äquivalent für Regler **2** ist der „FSP084_Controller_2_PI_Settings“.

[103..97] n.u.

[96] Controller_**1**_PI_Control - Wenn ‚1’ wird der I-Anteil des Reglers um den Faktor 1000 verlangsamt.

[95..64] Controller_**1**_I_Part (32 Bit), repräsentiert den I Anteil des PI Regler **1**.

[63..32] Controller_**1**_P2_Part (32 Bit), repräsentiert den 2. P Anteil des PI Regler **1**.
- [63..18] Vorkommastelle (Integer)
- [17..0] Nachkommastelle

[31..0] Controller_**1**_P1_Part (32 Bit), repräsentiert den 1. P Anteil des PI Regler **1**.
- [31..20] Vorkommastelle (Integer)
- [19..0] Nachkommastelle

Name	**FSP075_Controller_1_I_Part_ComparatorLimits**
Adresse	**$0x4B_H/75_D/0x3442_{ASCII}$**
Tiefe	6 Byte / 48 Bit
I/O	lesen / schreiben
Reset	$0x000000\_000000_H$

Repräsentiert die Bereichsgrenzen für Regler **1**, in denen der I Anteil bei der Regelung berücksichtig werden soll.

Das Äquivalent für Regler **2** ist der „FSP085_Controller2_I_Part_ComparatorLimits“.

[47..24] Controller_**1**_I_Part_ComparatorOFFThreshold (20 Bit), die Bits[3..0] sind immer ‚0‘.

[23..0] Controller_**1**_I_Part_ComparatorONThreshold (20 Bit), die Bits[3..0] sind immer ‚0‘.

Name	**FSP076_Controller_1_SetValueDeviation**
Adresse	**$0x4C_H/76_D/0x3443_{ASCII}$**
Tiefe	3 Byte / 24 Bit
I/O	lesen
Reset	$0x(\text{siehe Beschreibung})_H$

Über diesen FSP kann die Regeldifferenz (Delta I) des PI Regler **1** gelesen werden. Die Regeldifferenz wird dabei im Differenzbildner des Regler **1** aus dem Minuend SetValue (Ausgang: „FSP070_Controller_1_2_InputSourceSelectionMultiplexer“, bzw. „FSP071_Controller_1_SetValue“) und dem Subtrahend ActualValue (Ausgang: „FSP070_Controller_1_2_InputSourceSelectionMultiplexer“, bzw. „FSP072_Controller_1_ActualValue“).

Das Äquivalent für Regler **2** ist der „FSP086_Controller_2_SetValueDeviation“.

[23..0] SetValueDeviation (20 Bit), die Bits[3..0] sind immer ‚0‘.

Name	**FSP077_Controller_1_PI_Output**
Adresse	**$0x4D_H/77_D/0x3444_{ASCII}$**
Tiefe	9 Byte / 72 Bit
I/O	lesen
Reset	$0x(\text{siehe Beschreibung})_H$

Über diesen FSP können der PI-Anteil, I-Anteil und P-Anteil des PI Regler **1** gelesen werden

Das Äquivalent für Regler **2** ist der „FSP087_Controller_2_PI_Output“.

[71..48] Controller_**1**_P_Part_Output(20 Bit), die Bits[3..0] sind immer ‚0‘.

[47..24] Controller_**1**_I_Part_Output (20 Bit), die Bits[3..0] sind immer ‚0‘.

[23..0] Controller_**1**_PI_Output (20 Bit), die Bits[3..0] sind immer ‚0‘.

Name	**FSP078_Controller_1_P2_Part_ComparatorLimits**
Adresse	**$0x4E_H/78_D/0x3445_{ASCII}$**
Tiefe	6 Byte / 48 Bit
I/O	lesen / schreiben
Reset	$0x000000\_000000_H$

Repräsentiert die Bereichsgrenzen in denen der P2 Anteil bei der Regelung des Regler **1** berücksichtig werden soll.

Das Äquivalent für Regler **2** ist der „FSP088_Controller_2_P2_Part_ComparatorLimits“.

[47..24] Controller_**1**_P2_Part_ComparatorOFFThreshold (20 Bit), die Bits[3..0] sind immer ‘0’:

[23..0] Controller_**1**_P2_Part_ComparatorONThreshold (20 Bit), die Bits[3..0] sind immer ‘0’:

Name	**FSP079_Controller_1_SlopeLimiterOutput**
Adresse	**$0x4F_H/79_D/0x34465_{ASCII}$**
Tiefe	3 Byte / 24 Bit
I/O	lesen
Reset	$0x(\text{siehe Beschreibung})_H$

Liefert den Ausgangswert des Steilheitsbegrenzers für Regler **1**

Das Äquivalent für Regler **2** ist der „FSP089_Controller_2_SlopeLimiterOutput“.

[23..0] SlopeLimiterOutput Regler **1** (20 Bit), die Bits[3..0] sind immer ‘0’:

Name	**FSP080_SlopeLimiter_C2**
Adresse	**$0x3C_H/60_D/0x3343_{ASCII}$**
Tiefe	30 Byte / 240 Bit
I/O	lesen / schreiben
Reset	0x00_000000_000000_000000_000000_000000_0000_0000_0000_0000_ $745D17\_8BA2E8_H$

Repräsentiert Parameter des Steilheitsbegrenzers.

Das Äquivalent für den Steilheitsbegrenzer **1** findet sich im „FSP060_SlopeLimiter_C1“.

[239..234] n.u.

[233] Aktiviert den Steilheitbegrenzer,
wenn ‚1‘ UND ‚ACU_ControllerEnable‘

[232] Wenn ‚0‘ wird der Startwert des Steilheitbegrenzers mit 0 geladen, andernfalls der Wert des Eingangs ‚Dynamic_Threshold_RiseTime_Selection‘.
Dies ist i.d.R. ‚Controller1_ActValueMuxOut‘

[231..216] RampThresholdValue_**D** (16 Bit)

[215..200] RampThresholdValue_**C** (16 Bit)

[199..184] RampThresholdValue_**B** (16 Bit)

[183..168] RampThresholdValue_**A** (16 Bit)

[167..144] RampRiseTime_**E** (24 Bit)

[143..120] RampRiseTime_**D** (24 Bit)

[119..96] RampRiseTime_**C** (24 Bit)

[95..72] RampRiseTime_**B** (24 Bit)

[71..48] RampRiseTime_**A** (24 Bit)

[47..24] Obere Grenze (24 Bit)

[23..0] Untere Grenze (24 Bit)

Name	**FSP081_Controller_2_SetValue**
Adresse	**$0x51_H/81_D/0x3531_{ASCII}$**
Tiefe	3 Byte / 24 Bit
I/O	lesen
Reset	$0x(\text{siehe Beschreibung})_H$

Über diesen FSP kann der Sollwert des PI Regler **2** ausgelesen werden. Der Sollwert wird dabei über den Multiplexer „FSP070_Controller_1_2_InputSourceSelectionMultiplexer“ ausgewählt.

Das Äquivalent für Regler **1** ist der „FSP071_Controller_1_SetValue“.

[23..0] vorzeichenbehafteter 20 Bit Sollwert des PI Regler **2**, die Bits[3..0] sind immer ‚0‘.

Name	**FSP082_Controller_2_ActualValue**
Adresse	**$0x52_H/82_D/0x3532_{ASCII}$**
Tiefe	3 Byte / 24 Bit
I/O	lesen
Reset	$0x(\text{siehe Beschreibung})_H$

Über diesen FSP kann der Istwert des PI Regler **2** ausgelesen werden. Der Istwert wird dabei über den Multiplexer „FSP070_Controller_1_2_InputSourceSelectionMultiplexer“ ausgewählt.

Das Äquivalent für Regler **1** ist der „FSP072_Controller_1_ActualValue“.

[23..0] vorzeichenbehafteter 20 Bit Istwert des PI Regler **2**, die Bits[3..0] sind immer ‚0‘.

Name	**FSP083_Controller_2_Limits**
Adresse	**$0x53_H/83_D/0x3533_{ASCII}$**
Tiefe	6 Byte / 48 Bit
I/O	lesen / schreiben
Reset	$0x000000\_000000_H$

Über diesen FSP können die Bereichsgrenzen des PI Regler **2** festgelegt werden.

Das Äquivalent für Regler **1** ist der „FSP073_Controller_1_Limits“.

[47..24] Controller2_MaxVal (20 Bit), repräsentiert den oberen (maximalen) Grenzwert des PI Regler **2**, die Bits[3..0] sind immer ‚0‘.

[23..0] Controller2_MinVal (20 Bit), repräsentiert den unteren (minimalen) Grenzwert des PI Regler **2**, die Bits[3..0] sind immer ‚0‘.

Name	**FSP084_Controller_2_PI_Settings**
Adresse	**$0x54_H/84_D/0x3534_{ASCII}$**
Tiefe	13 Byte / 104 Bit
I/O	lesen / schreiben
Reset	$0x00\_00000000\_00000000\_00000000_H$

Über diesen FSP können die I und P1, P2 Anteile des Reglers **2** gesetzt werden.

Das Äquivalent für Regler **1** ist der „FSP074_Controller_1_PI_Settings“.

[103..96] Controller_**2**_PI_Control

[103..97] n.u.

[96] Wenn ‚1’ wird der I-Anteil des Reglers um den Faktor 1000 verlangsamt.

[95..64] Controller_**2**_I_Part (32 Bit), repräsentiert den I Anteil des PI Regler **2**.

[63..32] Controller_**2**_P2_Part (32 Bit), repräsentiert den 2. P Anteil des PI Regler **2**.

[31..0] Controller_**2**_P1_Part (32 Bit), repräsentiert den 1. P Anteil des PI Regler **2**.

Name	**FSP085_Controller2_I_Part_ComparatorLimits**
Adresse	**$0x55_H/85_D/0x3535_{ASCII}$**
Tiefe	3 Byte / 24 Bit
I/O	lesen / schreiben
Reset	$0x000000\_000000_H$

Repräsentiert die Bereichsgrenzen für Regler **2**, in denen der I Anteil bei der Regelung berücksichtig werden soll.

Das Äquivalent für Regler **1** ist der „FSP075_Controller_1_I_Part_ComparatorLimits“.

[47..24] Controller_**2**_I_Part_ComparatorOFFThreshold (20 Bit), die Bits[3..0] sind immer ‚0‘.

[23..0] Controller_**2**_I_Part_ComparatorONThreshold (20 Bit), die Bits[3..0] sind immer ‚0‘.

Name	**FSP086_Controller_2_SetValueDeviation**
Adresse	**$0x56_H/86_D/0x3536_{ASCII}$**
Tiefe	3 Byte / 24 Bit
I/O	lesen
Reset	$0x(\text{siehe Beschreibung})_H$

Über diesen FSP kann die Regeldifferenz (Delta I) des PI Regler **2** gelesen werden. Die Regeldifferenz wird dabei im Differenzbildner des Regler **2** aus dem Minuend SetValue (Ausgang: „FSP070_Controller_1_2_InputSourceSelectionMultiplexer“, bzw. „FSP081_Controller_2_SetValue“) und dem Subtrahend ActualValue (Ausgang: „FSP070_Controller_1_2_InputSourceSelectionMultiplexer“, bzw. „FSP082_Controller_2_ActualValue“).

Das Äquivalent für Regler **1** ist der „FSP076_Controller_1_SetValueDeviation“.

[23..0] SetValueDeviation (20 Bit), die Bits[3..0] sind immer ‚0‘.

Name	**FSP087_Controller_2_PI_Output**
Adresse	**$0x57_H/87_D/0x3837_{ASCII}$**
Tiefe	9 Byte / 72 Bit
I/O	lesen
Reset	$0x(\text{siehe Beschreibung})_H$

Über diesen FSP können der PI-Anteil, I-Anteil und P-Anteil des PI Regler **2** gelesen werden

Das Äquivalent für Regler **1** ist der „FSP077_Controller_1_PI_Output“.

[71..48] Controller_**2**_P_Part_Output(20 Bit), die Bits[3..0] sind immer ‚0‘.

[47..24] Controller_**2**_I_Part_Output (20 Bit), die Bits[3..0] sind immer ‚0‘.

[23..0] Controller_**2**_PI_Output (20 Bit), die Bits[3..0] sind immer ‚0‘.

Name	**FSP088_Controller_2_P2_Part_ComparatorLimits**
Adresse	**$0x58_H/88_D/0x3838_{ASCII}$**
Tiefe	6 Byte / 48 Bit
I/O	lesen / schreiben
Reset	$0x000000\_000000_H$

Repräsentiert die Bereichsgrenzen in denen der P2 Anteil bei der Regelung des Regler **2** berücksichtig werden soll.

Das Äquivalent für Regler **1** ist der „FSP078_Controller_1_P2_Part_ComparatorLimits“.

[47..24] Controller_**2**_P2_Part_ComparatorOFFThreshold (20 Bit), die Bits[3..0] sind immer ‘0’:

[23..0] Controller_**2**_P2_Part_ComparatorONThreshold (20 Bit), die Bits[3..0] sind immer ‘0’:

Name	**FSP089_Controller_2_SlopeLimiterOutput**
Adresse	**$0x59_H/89_D/0x3839_{ASCII}$**
Tiefe	3 Byte / 24 Bit
I/O	lesen
Reset	$0x(\text{siehe Beschreibung})_H$

Liefert den Ausgangswert des Steilheitsbegrenzers für Regler **2**

Das Äquivalent für Regler **1** ist der „FSP079_Controller_1_SlopeLimiterOutput“.

[23..0] SlopeLimiterOutput Regler **2** (20 Bit), die Bits[3..0] sind immer ‘0’:

Name	**FSP090_Adder_SourceSelectionMultiplexer**
Adresse	**$0x5A_H/90_D/0x3541_{ASCII}$**
Tiefe	2 Byte / 16 Bit
I/O	lesen / schreiben
Reset	$0x00\_0\_0_H$

Repräsentiert die Einstellungen der Addierer Eingangsmultiplexer für dessen Summanden **1** und **2**.

[15..10] n.u.

[9] '1' Ausgangswert des Multiplexers [Adder_Summand_**2**] wird invertiert

[8] '1' Ausgangswert des Multiplexers [Adder_Summand_**1**] wird invertiert

[7..4] Quellenwahl für Multiplexer Ausgangssignal [Adder_Summand_**2**] (4 Bit)

[3..0]	Ausgang des Multiplexers
0x0	0
0x1	Sollwert **B**, bestimmt durch FSP031_SetValue_B
0x2	Controller**2**_SetValueDeviation, analog FSP086_Controller_2_SetValueDeviation
0x3	Controller**2**_PI_Part_Output
0x4	SlopeLimiterOutput_C**2**[19..0]
0x5	Controller**2**_SetValueMuxOut[19..0], bestimmt durch FSP070_Controller_1_2_InputSourceSelectionMultiplexer[18][11..8]
0x6	GND_BUS[19..0]
0x7	GND_BUS[19..0]
0x8	GND_BUS[19..0]
0x9	GND_BUS[19..0]
0xA	GND_BUS[19..0]
0xB	GND_BUS[19..0]
0xC	GND_BUS[19..0]
0xD	GND_BUS[19..0]
0xE	GND_BUS[19..0]
0xF	GND_BUS[19..0]

[3..0] Quellenwahl für Multiplexer Ausgangssignal [Adder_Summand_**1**] (4 Bit)

[3..0]	Ausgang des Multiplexers
0x0	0
0x1	Sollwert **A**, bestimmt durch FSP030_SetValue_A
0x2	SlopeLimiterOutput_C**1**[19..0]
0x3	Controller**1**_SetValueDeviation, analog FSP076_Controller_1_SetValueDeviation
0x4	Controller**1**_PI_Part_Output
0x5	Controller**1**_SetValueMuxOut[19..0], bestimmt durch FSP070_Controller_1_2_InputSourceSelectionMultiplexer[16][3..0]
0x6	GND_BUS[19..0]
0x7	GND_BUS[19..0]
0x8	GND_BUS[19..0]
0x9	GND_BUS[19..0]
0xA	GND_BUS[19..0]
0xB	GND_BUS[19..0]

0xC	GND_BUS[19..0]
0xD	GND_BUS[19..0]
0xE	GND_BUS[19..0]
0xF	GND_BUS[19..0]

Name	**FSP091_Adder_Limits**
Adresse	**$0x5B_H/91_D/0x3542_{ASCII}$**
Tiefe	6 Byte / 48 Bit
I/O	lesen / schreiben
Reset	$0x000000\_000000_H$

Über diesen FSP werden die Bereichsgrenzen des Addierers festgelegt.

[47..24] Adder_MaxVal (20 Bit), repräsentiert den oberen (maximalen) Grenzwert des Addierers. Die Bits[3..0] sind immer ‚0'.

[23..0] Adder_MinVal (20 Bit), repräsentiert den unteren (minimalen) Grenzwert des Addierers. Die Bits[3..0] sind immer ‚0'.

Name	**FSP092_Adder_SumOut**
Adresse	**$0x5C_H/92_D/0x3543_{ASCII}$**
Tiefe	3 Byte / 24 Bit
I/O	lesen
Reset	$0x(\text{siehe Beschreibung})_H$

Über diesen FSP kann die Summe des Addierwerkes gelesen werden. Die Summanden werden dabei über „FSP090_Adder_SourceSelectionMultiplexer“ die Bereichsgrenzen über „FSP091_Adder_Limits“ festgelegt.

[23..0] Adder_SumOut (20 Bit), die Bits[3..0] sind immer ‚0‘.

Name	**FSP093_CorrFactorPI_Limits**
Adresse	**$0x5D_H/93_D/0x3544_{ASCII}$**
Tiefe	6 Byte / 48 Bit
I/O	lesen / schreiben
Reset	$0x000000\_000000_H$

Über diesen FSP werden die Bereichsgrenzen des CorrectorFactor PI Regler festgelegt.

[47..24] MaxVal (20 Bit), repräsentiert den oberen (maximalen) Grenzwert.
Die Bits[3..0] sind immer ‚0'.

[23..0] MinVal (20 Bit), repräsentiert den unteren (minimalen) Grenzwert.
Die Bits[3..0] sind immer ‚0'.

Name	**FSP094_CorrFactorPI_kP**
Adresse	**$0x5E_H/94_D/0x3545_{ASCII}$**
Tiefe	4 Byte / 32 Bit
I/O	lesen / schreiben
Reset	$0x00000000_H$

Über diesen FSP können die P Anteile des CorrectionFactor Reglers gesetzt werden.

[31..0] CorrFactorPI_kP (32 Bit), repräsentiert den P Anteil des CorrFactorPI.

Name	**FSP095_ComparatorControl**
Adresse	**$0x5F_H/95_D/0x3546_{ASCII}$**
Tiefe	2 Byte / 16 Bit
I/O	lesen / schreiben
Reset	$0x0000_H$

Repräsentiert verschiedene Komparator-Kontrollzustände in der MFU.

[15..13] n.u.

[12] ‚1' aktiviert den Komparator zum Aktivieren/Deaktivieren folgender Funktion, sofern die Komparatorgrenzen „FSP103_PWM_FDrive2_ComparatorLimits" durch die Regelabweichung des Regler **2** über-, bzw. unterschritten werden und Degaussing (Entmagnetisierung) aktiviert ist:

Es wird bei Komparatorauslösung anstelle des Ergebnis der Addierstufe „FSP092_Adder_SumOut" der Wert „FSP105_IGBT_AlternateSet" als Steuerwert der Rückspeisungsbegrenzung verwendet.

[11] ‚1' aktiviert den Komparator zum Aktivieren/Deaktivieren folgender Funktion, sofern die Komparatorgrenzen „FSP102_PWM_FDrive1_ComparatorLimits" durch die Reglabweichung des Regler **1** über-, bzw. unterschritten werden und Degaussing (Entmagnetisierung) nicht aktiviert ist:

Es wird bei Komparatorauslösung anstelle des Ergebnis der Addierstufe „FSP092_Adder_SumOut" der Wert „FSP105_IGBT_AlternateSet" als Steuerwert Rückspeisungsbegrenzung verwendet.

[10] ‚1' aktiviert die Rückspeisungsbegrenzung

[9] n.u.

[8] ‚1' aktiviert den Komparator zum Aktivieren/Deaktivieren des P2-Anteils beim Regler **2**, sofern dieser die Komparatorgrenzen „FSP088_Controller_2_P2_Part_ComparatorLimits" durch die Regelabweichung des Regler **2** über-, bzw. unterschreitet.

[7] ‚1' aktiviert den Komparator zum Aktivieren/Deaktivieren des I-Anteils beim Regler **2**, sofern dieser die Komparatorgrenzen „FSP085_Controller2_I_Part_ComparatorLimits" durch die Regelabweichung des Regler **2** über-, bzw. unterschreitet.

[6] n.u.

[5] ‚1' aktiviert die Degaussing- (Entmagnetisierungs)funktion und den Komparator für deren Durchführung, sofern dieser die Komparatorgrenzen „FSP101_Degauss_ComparatorLimit" durch den Sollwert des Regler **1** über-, bzw. unterschreitet.

[4] n.u.

[3] ‚1' aktiviert den Komparator zum Aktivieren/Deaktivieren folgender Funktion, sofern die Komparatorgrenzen „FSP100_V5_ComparatorLimits" durch die Regelabweichung des Regler **1** über-, bzw. unterschritten werden und Degaussing (Entmagnetisierung) nicht aktiviert ist:

‚1' aktiviert den Komparator zum Aktivieren/Deaktivieren folgender Funktion, sofern die Komparatorgrenzen „FSP100_V5_ComparatorLimits" durch die Regelabweichung des Regler **2** über-, bzw. unterschritten werden und Degaussing (Entmagnetisierung) aktiviert ist:

Es wird bei Komparatorauslösung IGBT V5 aktiviert, abhängig vom aktiven Degausing. Ist dieses aktiv, wird der Kompartorausgang von Regler **2** benutzt, sonst von Regler **1**.

[2] n.u.

[1] ‚1' aktiviert den Komparator zum Aktivieren/Deaktivieren des P2-Anteils beim Regler **1**, sofern die Reglerabweichung des Regler **1** „FSP078_Controller_1_P2_Part_ComparatorLimits“ über-, bzw. unterschreitet.

[0] ‚1' aktiviert den Komparator zum Aktivieren/Deaktivieren des I-Anteils beim Regler **1**, sofern dieser die Komparatorgrenzen „FSP075_Controller_1_I_Part_ComparatorLimits“ über-, bzw. unterschreitet.

Name	**FSP97_SelVal2CompP2Comp**
Adresse	**$0x61_H/97_D/0x3631_{ASCII}$**
Tiefe	1 Byte / 8 Bit
I/O	lesen / schreiben
Reset	$0x00_H$

Über dieses FSP kann für die Komparatoren zum aktivieren/deaktivieren des P**2** Teils der PI-Regler **1** und **2** der zu vergleichende Wert gewählt werden.

Im normalen Modus ist dies immer die errechnete Reglabweichung zwischen Soll- und Istwert des Reglers zugehörigen Diffenzbilders.

Im ExpertMode können andere Vergleichswerte ausgewählt werden.

[7..4] Legt den Vergleichswert für den P**2** Komparator des Regler **1** fest.

[3..0]	Ausgang des Multiplexers
0x0	0
0x1	Controller**1**_SetValueDeviation[19..0]
0x2	Controller**2**_SetValueDeviation[19..0]
0x3	SlopeLimiterOutputC**1**[19..0]
0x4	Controller**2**_SetValueMuxOut[19..0]
0x5	Controller**1**_ActValueMuxOut[19..0]
0x6	Controller**2**_ActValueMuxOut[19..0]
0x7	SlopeLimiterOutput_C**2**[19..0]
0x8	n.u.
0x9	n.u.
0xA	n.u.
0xB	n.u.
0xC	n.u.
0xD	n.u.
0xE	n.u.
0xF	n.u.

[3..0] Legt den Vergleichswert für den P**2** Komparator des Regler **1** fest.

[3..0]	Ausgang des Multiplexers
0x0	0
0x1	Controller**1**_SetValueDeviation[19..0]
0x2	Controller**2**_SetValueDeviation[19..0]
0x3	SlopeLimiterOutputC**1**[19..0]
0x4	Controller**2**_SetValueMuxOut[19..0]
0x5	Controller**1**_ActValueMuxOut[19..0]
0x6	Controller**2**_ActValueMuxOut[19..0]
0x7	SlopeLimiterOutput_C**2**[19..0]
0x8	n.u.
0x9	n.u.
0xA	n.u.
0xB	n.u.
0xC	n.u.
0xD	n.u.

0xE	n.u.
0xF	n.u.

Name	**FSP098_Selectable_kIkP1**
Adresse	**$0x62_H/98_D/0x3632_{ASCII}$**
Tiefe	32 Byte / 256 Bit
I/O	lesen / schreiben
Reset	0x0000000000000000_0000000000000000_0000000000000000_0000000000000000$_H$

This FSP contains the 4 of 5 selectable coefficients (kI, kP1) for the controller number one. The first coefficients set is stored in the FSP

[255..192] Fifth Coefficients set 64=32+32 bits.
[63..32] kI_**5** coefficient
[31..0] kP**1**_**5** coefficient

[191..128] Fourth Coefficients set 64=32+32 bits.
[63..32] kI_**4** coefficient
[31..0] kP**1**_**4** coefficient

[127..64] Third Coefficients set 64=32+32 bits.
[63..32] kI_**3** coefficient
[31..0] kP**1**_**3** coefficient

[63..0] Second Coefficients set 64=32+32 bits.
[63..32] kI_**2** coefficient
[31..0] kP**1**_**2** coefficient

Name	**FSP099_Selectable_kIkP1Thresholds**
Adresse	**$0x63_H/99_D/0x3633_{ASCII}$**
Tiefe	25 Byte / 200 Bit
I/O	lesen / schreiben
Reset	$00\_0000\_0000\_0000\_0000\_0000\_0000\_0000\_0000\_0000\_0000\_0000_H$

This FSP contains the thresholds (on and off) used to generate the coefficients multiplexer selector. The SelectorSignal is compared with these thresholds and a decoded signal is generated for the purpose above descripted. The FSP contains also the Enable bit used to turn on/off the coefficientSelector module. In off mode, the first coefficients set is always selected (module back compatible)

[199..192] FSP99_Selectable_kIkP_Enable[7..0]

[191..0] FSP99_Selectable_kIkP_Thresholds

[199..193] n.u.

[192] Enabled: ACU_CoefficientSelector
0=> The first coefficients set are selected to the output (trans-parent mode).
1=> The output coefficients take one of the 5 available inputs based on the SelectorSignal value.

[191..168] Fourth Off Threshold value (20b), die unteren 4 Bit sind immer ‘0’.

[167..144] Third Off Threshold value (20b), die unteren 4 Bit sind immer ‘0’.

[143..120] Second Off Threshold value (20b), die unteren 4 Bit sind immer ‘0’.

[119..96] First Off Threshold value (20b), die unteren 4 Bit sind immer ‘0’.

[95..72] Fourth On Threshold value (20b), die unteren 4 Bit sind immer ‘0’.

[71..48] Third On Threshold value (20b), die unteren 4 Bit sind immer ‘0’.

[47..24] Second On Threshold value (20b), die unteren 4 Bit sind immer ‘0’.

[23..4] First On Threshold value (20b), die unteren 4 Bit sind immer ‘0’.

Name	**FSP100_V5_ComparatorLimits**
Adresse	**$0x64_H/100_D/0x3634_{ASCII}$**
Tiefe	6 Byte / 48 Bit
I/O	lesen / schreiben
Reset	$0x000000\_000000_H$

Legt die Grenzwerte des Komparators zur Aktivierung/Deaktivierung von IGBT V5 fest.

[47..24] V5_ComparatorOFFThreshold (20 Bit), die Bit[3..0] sind immer ‘0’.

[23..0] V5_ComparatorONThreshold (20 Bit), die Bit[3..0] sind immer ‘0’.

Name	**FSP101_Degauss_ComparatorLimit**
Adresse	**$0x65_H/101_D/0x3635_{ASCII}$**
Tiefe	6 Byte / 48 Bit
I/O	lesen / schreiben
Reset	$0xFFF6A4\_00095B_H$

Repräsentiert die Schwelle des Komparators für das Entmagnetisieren (Degaussen)

[47..24] Degauss_ComparatorOFFThreshold (20 Bit), die Bit[3..0] sind immer ‘0’.

[23..0] Degauss_ComparatorONThreshold (20 Bit), die Bit[3..0] sind immer ‘0’.

Name	**FSP102_PWM_FDrive1_ComparatorLimits**
Adresse	**$0x66_H/102_D/0x3636_{ASCII}$**
Tiefe	6 Byte / 48 Bit
I/O	lesen / schreiben
Reset	$0x000000\_000000_H$

Legt die Grenzwerte des Komparators zur Aktivierung/Deaktivierung des alternativen Steuerwerts der Rückspeisungsbegrenzung fest, sofern Degaussen (Entmagnetisieren) nicht aktiv ist.

[47..24] PWM_FDRIVE1_ComparatorOFFThreshold (20 Bit), die Bit[3..0] sind immer ‘0’.

[23..0] PWM_FDRIVE1_ComparatorONThreshold (20 Bit), die Bit[3..0] sind immer ‘0’.

Name	**FSP103_PWM_FDrive2_ComparatorLimits**
Adresse	**$0x67_H/103_D/0x3637_{ASCII}$**
Tiefe	6 Byte / 48 Bit
I/O	lesen / schreiben
Reset	$0x000000\_000000_H$

Legt die Grenzwerte des Komparators zur Aktivierung/Deaktivierung des alternativen Steuerwerts der Rückspeisungsbegrenzung fest, sofern Degaussen (Entmagnetisieren) aktiv ist.

[47..24] PWM_FDRIVE2_ComparatorOFFThreshold (20 Bit), die Bit[3..0] sind immer ‘0’.

[23..0] PWM_FDRIVE2_ComparatorONThreshold (20 Bit), die Bit[3..0] sind immer ‘0’.

Name	**FSP105_IGBT_AlternateSetValue**
Adresse	**$0x69_H/105_D/0x3639_{ASCII}$**
Tiefe	3 Byte / 24 Bit
I/O	lesen / schreiben
Reset	$0x000000_H$

Repräsentiert den alternativen Steuerwert für die Rückspeisungsbegrenzung, sofern einer der Komparatoren die Grenzen „FSP102_PWM_FDrive1_ComparatorLimits“, bei nicht aktivem Degaussing (Entmagnetisieren), bzw. „FSP103_PWM_FDrive2_ComparatorLimits“, bei aktivem Degaussing (Entmagnetisieren) unter-, bzw. überschritten hat.

[23..0] IGBT_AlternateSetValue (20 Bit), die Bit[3..0] sind immer ‘0’.

Name	**FSP106_EnergyRecoverLimitation_CurrentDriveValue**
Adresse	**$0x6A_H/106_D/0x3641_{ASCII}$**
Tiefe	6 Byte / 48 Bit
I/O	lesen / schreiben
Reset	$0x000000\_000000_H$

Funktion

[47..24] ERL_CurrentThresholdValue (20 Bit)

[23..0] ERL_CurrentDriveValue (20 Bit)

Name	**FSP107_CtrlEnable_Delay**
Adresse	**$0x6B_H/107_D/0x3642_{ASCII}$**
Tiefe	1 Byte / 8 Bit
I/O	lesen / schreiben
Reset	$0x10_H$

[7..5]	n.u.
[5]	Enable
[3..0]	DelayValue * 10us

Name	**FSP110_DACx_and_ScopeChannelx_SourceSelectionMultiplexer**
Adresse	**$0x6E_H/110_D/0x3645_{ASCII}$**
Tiefe	3 Byte / 24 Bit
I/O	lesen / schreiben
Reset	$0x00\_0\_1\_1\_1_H$

Repräsentiert die Einstellungen der DAC Ausgangs- und int. Oszilloskop Eingangsmultiplexer.

Sowohl die Ausgabequellen der DACs, als auch die Eingangsquellen des internen Oszilloskops werden über einen einzigen Multiplexer geführt. Das interne Oszilloskop hat 4 Kanäle, die dabei jeweils den nachfolgenden Quellen entsprechen.

[23..20] n.u.

[19] '1' Ausgangswert des Multiplexers [DAC_**4**_and_ScopeChannel_**4**_Source]
→ X**8** wird invertiert

[18] '1' Ausgangswert des Multiplexers [DAC_**3**_and_ScopeChannel_**3**_Source]
→ X**7** wird invertiert

[17] '1' Ausgangswert des Multiplexers [DAC_**2**_and_ScopeChannel_**2**_Source]
→ X**6** wird invertiert

[16] '1' Ausgangswert des Multiplexers [DAC_**1**_and_ScopeChannel_**1**_Source]
→ X**5** wird invertiert

[15..12] Quellenwahl für Multiplexer Ausgangssignal [DAC_**4**_and_ScopeChannel_**4**_Source] → X**8**

[3..0]	Ausgang des Multiplexers
0x0	0
0x1	Controller**1**_SetValueMuxOut[19..0], Sollwert Regler 1, bestimmt durch „FSP070_Controller_1_2_InputSourceSelectionMultiplexer[16][3..0]“
0x2	Controller**2**_SetValueMuxOut[19..0], Sollwert Regler 2, bestimmt durch „FSP070_Controller_1_2_InputSourceSelectionMultiplexer[18][11..8]„
0x3	DAC_and_Scope_PreSelectionMux_**1**_Out[19..0], bestimmt durch “FSP064_USIxHS_Multiplexer[83..80]”
0x4	DAC_and_scope_PreSelectionMux_**2**_Out[19..0], bestimmt durch “FSP064_USIxHS_Multiplexer[87..84]”
0x5	Controller**1**_P_Part_Ouput[19..0]
0x6	Controller**1**_I_Part_Ouput[19..0]
0x7	Controller**1**_PI_Part_Ouput[19..0]
0x8	Controller**2**_P_Part_Ouput[19..0]
0x9	Controller**2**_I_Part_Ouput[19..0]
0xA	Controller**2**_PI_Part_Ouput[19..0]
0xB	Adder_SumOut[19..0]
0xC	ERLOutValue, Rückspeisungsbegrenzung[19..0]
0xD	DeterminedOutValue[19..0], festgelegter Sollwert, bestimmt durch „FSP105_IGBT_AlternateSetValue“
0xE	intFuncGenOutput[19..0]
0xF	ControllerStatusBits[19..0] entspricht dem FSP121_ControllerStatusBits LE/SE [19] Controller_**1**_I_Part_Disable [18] Controller_**2**_I_Part_Disable [17] Controller_**1**_P2_Part_Active [16] Controller_**2**_P2_Part_Active [15] Degaussing_Active [14] Controller_**1**_PWM_FDrive_Comparator

	[13] Controller_**2**_PWM_FDrive_Comparator [12] PWM_FDriveComparator_MuxOut [11] Controller_**1**_IGBT_V5_Active [10] Controller_**2**_IGBT_V5_Active [9] IGBT_V5_Active [8] n.u., immer ‚0‘ [7] ACU_Controller_Enable [6] SlopeLimiterEnable_C**1** [5] n.u., immer ‚0‘ [4] SlopeLimiterEnable_C**2** [3..0] n.u., immer ‚0‘

[11..8] Quellenwahl für Multiplexer Ausgangssignal [DAC_**3**_and_ScopeChannel_**3**_Source] → X**7**

[3..0]	Ausgang des Multiplexers
0x0	0
0x1	Controller**1**_MultipliedDeviation[19..0], Ausgang des MFU_DifferenceCalculator, Regler 1, multiplizierte Reglerdifferenz
0x2	Controller**2**_MultipliedDeviation[19..0], Ausgang des MFU_DifferenceCalculator, Regler 2, multiplizierte Reglerdifferenz
0x3	DAC_and_scope_PreSelectionMux_**1**_Out[19..0], bestimmt durch “FSP064_USIxHS_Multiplexer[83..80]”
0x4	DAC_and_Scope_PreSelectionMux_**2**_Out[19..0], bestimmt durch “FSP064_USIxHS_Multiplexer[87..84]”
0x5	Controller**1**_P_Part_Ouput[19..0]
0x6	Controller**1**_I_Part_Ouput[19..0]
0x7	Controller**1**_PI_Part_Ouput[19..0]
0x8	Controller**2**_P_Part_Ouput[19..0]
0x9	Controller**2**_I_Part_Ouput[19..0]
0xA	Controller**2**_PI_Part_Ouput[19..0]
0xB	Adder_**1**_SumOut[19..0]
0xC	ERLOutValue[19..0], Rückspeisungsbegrenzung
0xD	DeterminedOutValue[19..0], festgelegter Sollwert, bestimmt durch „FSP105_IGBT_AlternateSetValue“
0xE	intFuncGenOutput[19..0]
0xF	n.u., immer 0

[7..4] Quellenwahl für Multiplexer Ausgangssignal [DAC_**2**_and_ScopeChannel_**2**_Source] → X**6**

[3..0]	Ausgang des Multiplexers
0x0	0
0x1	Controller**1**_ActValueMuxOut[19..0], Istwert des Regler 1, bestimmt durch „FSP070_Controller_1_2_InputSourceSelectionMultiplexer[17][7..4]“
0x2	SetValue_**B**_MultiplexerOutput[19..0], bestimmt durch „FSP013_PeripheralConfig[0]“
0x3	ActualValue_**A**[19..0], analog „FSP020_ActualValue_A”
0x4	ActualValue_**B**[19..0], analog „FSP021_ActualValue_B”
0x5	SlopeLimiterOutput_C**2**[19..0]
0x6	Controller**1**_MultipliedDeviation[19..0], Ausgang des MFU_DifferenceCalculator, Regler **1**, multiplizierte Reglerdifferenz
0x7	Controller**2**_MultipliedDeviation[19..0], Ausgang des MFU_DifferenceCalculator, Regler **2**, multiplizierte Reglerdifferenz
0x8	DAC_and_Scope_PreSelectionMux_**1**_Out[19..0], bestimmt durch “FSP064_USIxHS_Multiplexer[83..80]”

0x9	DAC_and_Scope_PreSelectionMux_**2**_Out[19..0], bestimmt durch “FSP064_USIxHS_Multiplexer[87..84]”
0xA	Controller**1**_P_Part_Ouput[19..0]
0xB	Controller**1**_I_Part_Ouput[19..0]
0xC	Controller**1**_PI_Part_Ouput[19..0]
0xD	ERLOutValue[19..0], Rückspeisungsbegrenzung
0xE	DeterminedOutValue[19..0], festgelegter Sollwert, bestimmt durch „FSP105_IGBT_AlternateSetValue“
0xF	intFuncGenOutput[19..0]

[3..0] Quellenwahl für Multiplexer Ausgangssignal [DAC_**1**_and_ScopeChannel_**1**_Source] → X**5**

[3..0]	Ausgang des Multiplexers
0x0	0
0x1	Controller**1**_SetValueMuxOut[19..0], Sollwert Regler **1**, bestimmt durch „FSP070_Controller_1_2_InputSourceSelectionMultiplexer[16][3..0]“
0x2	Controller**2**_SetValueMuxOut[19..0], Sollwert Regler **2**, bestimmt durch „FSP070_Controller_1_2_InputSourceSelectionMultiplexer[18][11..8]„
0x3	SlopeLimiterOutput_C**1**[19..0]
0x4	Controller**1**_MultipliedDeviation[19..0], Ausgang des MFU_DifferenceCalculator, Regler **1**, multiplizierte Regeldifferenz
0x5	Controller**2**_MultipliedDeviation[19..0], Ausgang des MFU_DifferenceCalculator, Regler **2**, multiplizierte Regeldifferenz
0x6	intFuncGenOutput[19..0]
0x7	DAC_and_Scope_PreSelectionMux_**1**_Out[19..0], bestimmt durch “FSP064_USIxHS_Multiplexer[83..80]”
0x8	DAC_and_Scope_PreSelectionMux_**2**_Out[19..0], bestimmt durch “FSP064_USIxHS_Multiplexer[87..84]”
0x9	Controller**1**_I_Part_Ouput[19..0]
0xA	Controller**1**_PI_Part_Ouput[19..0]
0xB	ERLOutValue[19..0], Rückspeisungsbegrenzung
0xC	DeterminedOutValue[19..0], festgelegter Sollwert, bestimmt durch „FSP105_IGBT_AlternateSetValue“
0xD	n.u., immer 0
0xE	n.u., immer 0
0xF	n.u., immer 0

Name	**FSP114_intScopeTFTSettings**
Adresse	**$0x72_H/114_D/0x3732_{ASCII}$**
Tiefe	3 Byte / 24 Bit
I/O	lesen / schreiben
Reset	$0x01\_210\_0_H$

Repräsentiert die Einstellungen für die Anzeige des internen Scopes auf dem TFT Standardbildschirm.

[23..17] n.u.

[16] intScopeTFT_ShowClassicView
Wenn ‚1' wird anstelle des internen Scopes auf dem TFT in jedem Fall die klassische grafische Ansicht des Stromistwertes angezeigt.
Wenn ‚0' wird im ‚Remote'-Betrieb und wenn KEIN PC mit der MFU verbunden ist im Triggerfall das interne Scope auf dem TFT dargestellt. In allen anderen Fällen die klassiche grafische Anischt des Stromistwertes.

[15..4] intScopeTFT_Channel_Selector
[15] n.u.
[14..12] TFT Kanal **3** (grün)
[11] n.u.
[10..8] TFT Kanal **2** (rot)
[7] n.u.
[6..4] TFT Kanal **1** (blau)
mit

Bitfolge	Aufzeichnung
000	Scope Eingang Kanal **1**
001	Scope Eingang Kanal **2**
010	Scope Eingang Kanal **3**
011	Scope Eingang Kanal **4**

[3] n.u.

[2..0] intScopeTFT_Y_Scalling
Skaliert die Auflösung der aufgezeichneten Kanäle für die TFT Darstellung. Jeder Scopeeingangskanal hat eine Auflösung von 12 Bit, die intern für die Darstellung auf dem TFT auf 7 Bit reduziert werden. Von den zwölf Bit werden bei bipolaren Geräten alle 12 Bit benutzt, das MSB bildet dabei das Vorzeichen. Bei unipolaren Geräten werden hingegen nur 11 Bit benutzt, das Vorzeichen bleibt unbenutzt. Bei bipolaren Geräten hat dies zur Folge, dass 6 Bit die eigentliche Spannung und 1 Bit das Vorzeichen repräsetieren. Bei unipolaeren Geräten wird die Spannung hingegen auf allen 7 Bit abgebildet. Die Skalierung verschiebt nun diese Bitfolge nach rechts. Dies erhöht die Auflösung im kleinen Spannungsbereich, da die oberen Bit ignoriert werden. Bei bipolaren Geräten kann dadurch die Anzeige um den Faktor 25 = 32-fach und bei unipolaren Geräten 24 = 16-fach vergrößert werden.

Bitfolge	Aufzeichnung	
000	Bipolar: Unipolar:	TFT Speicherbits[Eingangsbits 11(VZ), 10..5] TFT Speicherbits[Eingangsbits 10..4]
001	Bipolar: Unipolar:	TFT Speicherbits[Eingangsbits 11(VZ), 9..4] TFT Speicherbits[Eingangsbits 9..3]
010	Bipolar: Unipolar:	TFT Speicherbits[Eingangsbits 11(VZ), 8..3] TFT Speicherbits[Eingangsbits 8..2]
011	Bipolar: Unipolar:	TFT Speicherbits[Eingangsbits 11(VZ), 7..2] TFT Speicherbits[Eingangsbits 7..1]
100	Bipolar: Unipolar:	TFT Speicherbits[Eingangsbits 11(VZ), 6..1] TFT Speicherbits[Eingangsbits 6..0]
101..111	Bipolar: Unipolar:	TFT Speicherbits[Eingangsbits 11(VZ), 5..0] TFT Speicherbits[Eingangsbits 6..0]

Name	**FSP116_intScopeSettings**
Adresse	**$0x74_H/116_D/0x3734_{ASCII}$**
Tiefe	9 Byte / 72 Bit
I/O	lesen / schreiben
Reset	$0x0000\_0\_0\_0000\_00000\_0\_0\_0_H$

Repräsentiert die interne Scope Parametrierung

[71..60] TriggerLevel (12 Bit) Repräsentiert den Triggerlevel des internen Oszilloskops

[59..56] n.u. (4 Bit), TriggerLevel unterstes Nibble zur Rundung

[55..52] External RAM extra configuration(4 bits)

[3..0]	
[0]	Debug mode enable. Active high. It selects as data to store in the RAM, the output of a 32 bit counter and as external trigger signal the bit nr [1] of this nibble.
[1]	Debug External trigger signal. It enables also the debug 32 bits counter described above.Active high
[2]	Trigger signal selector: if 0 => internal trigger signal selected; else external trigger signal selected.
[3]	Incoming data sampler signal: if 0=> the incoming data are sampled using the internal Scope timebase pulse signal; else the incoming data are sampled using the HighSpeedPort_2_NewDataStrobe pulse signal.

[51..48] TimeScale (4 Bit), gibt die Skalierung der Zeitbasis an

[3..0]	Skalierung
0x0	ns, 100ns kleinster Wert bei TimeBase = 0, bzw. 1, Schrittweite 100ns
0x1	µs, 1µs kleinster Wert bei TimeBase = 0, bzw. 1, Schrittweite 1µs
0x2	ms, 1ms kleinster Wert bei TimeBase = 0, bzw. 1, Schrittweite 1ms
0x3	s, 1s kleinster Wert bei TimeBase = 0, bzw. 1, Schrittweite 1s
0x4.. 0xF	n.u., Standardskalierung 100ns wird gewählt

[47..32] TimeBase (16 Bit), repräsentiert die Zeitbasis (horizontal) des internen Oszilloskop.
'TimeBase' gibt dabei an, wie viele male 'TimeScale' für ein Gesamtbild benötigt wird.
Bsp.: abhängig von ‚TimeScale' gilt für die zeitliche Auflösung bei 500 Bildpunkten:(TimeBase x TimeScale x 500)/10 = Zeit/DIV
TimeBase = 2, TimeScale = 1(us) => 2us => alle 2us wird ein Bildpunkt erfasst.
Bei 500 Werten bedeutet dies 2us * 500 = 1000us = 1ms für das gesamte Bild.

[31..25] n.u. (7 Bit)

[24..16] TriggerOffset (9 Bit), gibt einen Triggeroffset an

[15..12] n.u. (4 Bit)

[11..8] TriggerChannel (4 Bit)

[3..0]	Trigger Kanal
0x0	Kanal 1
0x1	Kanal 2
0x2	Kanal 3
0x3	Kanal 4
0x4	n.u.
0x5	n.u.
0x6	n.u.

0x7	n.u.
0x8	Externe Triggerquelle über LEMO Buchse X4, LEMO 1 IN
0x9.. 0xF	Interne Triggerquelle über den internen Funktionsgenerator

[7..4] Trigger Mode (4 Bit)

[3..0]	Trigger Modus
0x0	Triggerlevel > Wert, Messwertannäherung von unten durch Triggerlevel
0x1	Triggerlevel = Wert
0x2	Triggerlevel < Wert, Messwertannäherung von oben durch Triggerlevel
0x3	positive Triggerflanke über digitalen Eingang LEMO 1 oder den int. Funktions-generator
0x4	negative Triggerflanke über digitalen Eingang LEMO 1 oder den int. Funktions-generator

[3..0] TriggerConfig (4 Bit)

[3..0]	Beschreibung
0x0	Trigger Konfigurieren, keine Datenaufzeichnung
0x2	auf Triggerereignis warten (Trigger scharf)

Name	**FSP117_intScopeTriggerReadOut**
Adresse	**$0x75_H/117_D/0x3735_{ASCII}$**
Tiefe	4 Byte / 32 Bit
I/O	lesen
Reset	$0x(\text{siehe Beschreibung})_H$

Repräsentiert Informationen über den Triggerprozess des internen Oszilloskops. Die Werte in diesem FSP werden bei einem Triggerereignis aktualisiert.

[31..28] n.u. (immer 0)

[27..24] TriggerStatus[3..0]

[3..0]	Beschreibung
0x0	TriggerIdle (FSP116_intScopeSettings[3..0] FunctionMode auf 0x0, keine Datenaufzeichnung, Trigger kann konfiguriert werden)
0x1	PrefillingRAM (FSP116_intScopeSettings[3..0] FunctionMode wurde auf 0x2 gesetzt, Datenaufzeichnung noch nicht aktiv, warten bis RAM vorgefüllt ist)
0x2	WaitForTriggerEvent (RAM ist vorgefüllt, FSP116_intScopeSettings[3..0] FunctionMode ist auf 0x2, Datenaufzeichnung aktiv, warten auf Triggerereignis)
0x4	TriggerEventOccured (das in FSP116_intScopeSettings[7..4] TriggerMode gewählte Ereignis hat stattgefunden)
0x8	WaitForReadOutData (alle Daten rund um das Triggerereignis sind aufgezeichnet und zur Abholung bereit)

[23..21] n.u. (immer 0)

[20..12] TriggerOffsetEcho[8..0]
entspricht nach dem Triggerereignis dem Wert von FSP116_intScopeSettings[25..16]

[11..9] n.u. (immer 0)

[8..0] TriggerAddress[8..0]
entspricht nach dem Triggerereignis der Trigger-Adresse

Name	**FSP118_intScopeDataReadOutAddress**
Adresse	**$0x76_H$/118_D/$0x3736_{ASCII}$**
Tiefe	2 Byte / 16 Bit
I/O	lesen / schreiben
Reset	$0x0000_H$

Repräsentiert die 9 Bit Adresse des internen Scopespeichers dessen Daten gelesen werden sollen

[15..9] n.u. (7 Bit)

[8..0] ScopeReadOutRAMAddress (9 Bit), repräsentiert die RAM Speicheradresse des internen Oszilloskop die angesprochen werden soll. Das interne Oszilloskop besitzt 4 Kanäle die in 4 RAM Speicher abgelegt werden können. Dies geschieht zeitgleich, d.h. die jeweils angelegte Adresse adressiert alle 4 RAM Speicher zeitgleich.

Name	**FSP119_intScopeDataReadOut**
Adresse	**$0x77_H$/119_D/$0x3737_{ASCII}$**
Tiefe	10 Byte / 80 Bit (ab MFU FW 7.5.0)
I/O	lesen
Reset	0x(siehe Beschreibung)

Repräsentiert die 9 Bit Adresse des internen Scopespeichers, und die darin befindlichen Daten (ScopeRamData_CHx) der Kanäle 1 bis 4.

[79..76] n.u. (4 Bit, immer 0)

[75..64] ScopeRamData_CH4[11..0] (12 Bit)

[63..60] n.u. (4 Bit, immer 0)

[59..48] ScopeRamData_CH3[11..0] (12 Bit)

[47..44] n.u. (4 Bit, immer 0)

[43..32] ScopeRamData_CH2[11..0] (12 Bit)

[31..28] n.u. (4 Bit, immer 0)

[27..16] ScopeRamData_CH1[11..0] (12 Bit)

[15..9] n.u. (7 Bit, immer 0)

[8..0] ScopeReadOutRAMAddress[8..0], (9 Bit) entspricht dem Eintrag in FSP118_intScopeDataReadOutAddress und dient der Verifizierung ob die Daten auch zu dieser Adresse gehören.

Name	**FSP120_intFunctionGenerator**
Adresse	**$0x78_H/120_D/0x3738_{ASCII}$**
Tiefe	16 Byte / 128 Bit
I/O	lesen / schreiben
Reset	$0x0\_0\_000000\_000000\_000000\_000000\_000000_H$

Repräsentiert die Parameter für den internen Funktionsgenerator

[127..124] ModeSelect

[3..0]	ModeSelect
0x0	Funktionsgenerator läuft frei, unabhängig ‚ExtSyncSource‘
0x1	Eingänge , ManualDirectionUP_nDown‘ und , ManualTriggerCounter‘ sind aktiv
0x2	Rampe aufwärts/abwärts startet nur mit steigender Flanke an ‚ExternalEnableSelect‘
0x3	-
0x4	-
0x5	-
0x6	-
0x7	-
0x8	-
0x9	-
0xA	-
0xB	-
0xC	-
0xD	-
0xE	-
0xF	-

[123..120] ExternalEnableSelect

[3..0]	ExternalEnableSelect
0x0	GND
0x1	GND
0x2	GND
0x3	GND
0x4	GND
0x5	GND
0x6	GND
0x7	GND
0x8	GND
0x9	GND
0xA	GND
0xB	Front LEMO In 2
0xC	Front LEMO In 1
0xD	intScopeTriggerEventDetected
0xE	VCC
0xF	GND

[119..96] RampRiseTime (24 Bit), Wenn X“00_0000“ pulst der Funktionsgenerator, andernfalls wird gerammt. RampTime repräsentiert dabei die Pausenzeit zwischen zwei Rampenstützpunkten während der Anstiegs-, bzw. Abstiegszeit der Rampe. Gibt die Zeit der Pausenintervalle zwischen den einzelnen Inkrementen zwischen dem ‚LowermostValueRampPulse’ und dem ‚TopValueRampPulse’ an. Das Pausenintervall entspricht „(RampRise * 10ns) + 10 ns“.
Beispiel: LowermostValueRampPulse = -32768_D
TopValueRampPulse = 32767_D
RampTime = 1_D
Es liegen 65535 Stützpunkte zwischen dem Minimal und Maximalwert.
Es wird alle „(RampRise * 10ns)“ ein neuer Stützpunkt ausgegeben, d.h. alle 10ns. Eine Rampe dauert also 65535 * (1*10ns)

[95..72]

[95..89] n.u. (7 Bit)

[88..72] LowermostValueDuration (17 Bit), repräsentiert die Zeitdauer die der untere (minimale) Pulswert/Scheitelwert der Rampe (Faltbottom) des internen Funktionsgenerators anstehen soll. Ist Bit [88] gesetzt gilt der Wert von Bit[87..72] in uSekunde. Ist Bit [88] nicht gesetzt gilt der Wert von Bit[87..72] in mSekunden.

[71..48]

[71..65] n.u. (7 Bit)

[64..48] TopValueDuration (17 Bit), repräsentiert die Zeitdauer die der obere (maximale) Pulswert/Scheitlwert der Rampe (Flattop) des internen Funktionsgenerators anstehen soll. Ist Bit [64] gesetzt gilt der Wert von Bit [63..48] in uSekunde. Ist Bit [64] nicht gesetzt gilt der Wert von Bit [63..48] in mSekunden.

[47..24]

[47..28] LowermostValueRampPulse (20 Bit), repräsentiert den unteren (minimalen) Pulswert/Scheitelwert der Rampe des internen Funktionsgenerators. Vorzeichenbehafteter Wert im Bereich zwischen -11..+11 Volt.

[27..24] n.u. (4 Bit)

[23..0]

[23..4] TopValueRampPulse (20 Bit), repräsentiert den oberen (maximalen) Pulswert/Scheitelwert der Rampe des internen Funktionsgenerators. Vorzeichenbehafteter Wert im Bereich zwischen -11..+11 Volt.

[3..0] n.u. (4 BIt)

Name	**FSP121_ControllerStatusBits LE/SE**
Adresse	**$0x79_H/121_D/0x3739_{ASCII}$**
Tiefe	3 Byte / 24 Bit
I/O	lesen
Reset	$0x01_H$

Hierbei handelt es sich um ein binäres Datenwort, bestehend aus 23 Bits, die verschiedene Steuerzustände innerhalb der MFU repräsentieren.

[23..20] n.u., immer ‚0'

[19] Controller_**1**_I_Part_Disable

[18] Controller_**2**_I_Part_Disable

[17] Controller_**1**_P**2**_Part_Active

[16] Controller_**2**_P**2**_Part_Active

[15] Degaussing_Active

[14] Controller_**1**_PWM_FDrive_Comparator

[13] Controller_**2**_PWM_FDrive_Comparator

[12] PWM_FDriveComparator_MuxOut

[11] Controller_**1**_IGBT_V5_Active

[10] Controller_**2**_IGBT_V5_Active

[9] IGBT_V5_Active

[8] n.u.

[7] ACU_Controller_Enable

[6] SlopeLimiterEnable C**1**

[5] n.u.

[4] SlopeLimiterEnable C**2**

[3..0] immer ‚0'

Name	**FSP225_SW_READ_RECORDED_MODULE_DATA_FROM_DATA_STORAGE**
Adresse	**$0xE1_H/225_D/0x4531_{ASCII}$**
Tiefe	dynamisch
I/O	lesen / schreiben
Reset	$0x(\text{siehe Beschreibung})_H$

Für die Nutzung von FSP225 muss das datensammelnde Modul über ein [DataStorage] Modul verfügen.

Über diesen FSP werden die gesammelten Daten des [DataStorage] via HighSpeed an die MFU und von dieser weiter an den angeschlossenen Computer übertragen.

Wichtig ist, die zu lesenden Daten müssen schon im [DataStorage] gespeichert sein.

Die Konfiguration von [DataStorage] ist explizit nicht Bestandteil dieses FSP und muss zuvor extern erfolgen.

Die externe Sequenz zur Aufzeichnung der Daten in [DataStorge] ist wie folgt:

1) FSP225 zunächst schreiben:
   `STX PID PID GW MA FSP FSP [USI Number] PP PP ETX (11 Byte)`

   Parameter [ModuleNumber] ist nicht nötig, da Modul Nummer bei HighSpeed uninteressant => immer Modul 1(0).

2) `Reset` Kommando setzen/löschen in Modul FSP018[0].
3) Status lesen (Modul FSP017) und auf `0x00` testen.
4) `ExtTriggerCommand` setzen (active high) (Modul FSP018[8]).
5) Status lesen (Modul FSP017), muss nun `0x05` sein (Ereignis getriggert, Werte erfasst).
6) `ExtTriggerCommand` loeschen (low) (Modul FSP018[8]).
7) Status lesen (Modul FSP017), muss nun `0x01` sein (Werte erfasst).
8) `RdEnable` setzen (Modul FSP018[4]).

Jetzt kommt das, was diese Funktion tut:

1) Kommando `CMDTriggerSomething` absetzen durch setzen/löschen von `CPU_STATUS_CMD_TRIGGER_SOMETHING` im CPU_STATUS[30].
2) Aufgezeichnete Messwerte per HighSpeed empfangen.
   Die Antwort ist dabei wie folgt:
```
STX GW MA FSP FSP  [Byte7 Byte6 Byte5 Byte4 Byte3 Byte2 Byte1 Byte0] <- n
                   [Byte7 Byte6 Byte5 Byte4 Byte3 Byte2 Byte1 Byte0] <- n-1
                   [Byte7 Byte6 Byte5 Byte4 Byte3 Byte2 Byte1 Byte0] <- n-2
                   [Byte7 Byte6 Byte5 Byte4 Byte3 Byte2 Byte1 Byte0] <- n-3
                   [Byte7 Byte6 Byte5 Byte4 Byte3 Byte2 Byte1 Byte0] <- n-4
                   [...]
                   [Byte7 Byte6 Byte5 Byte4 Byte3 Byte2 Byte1 Byte0] <- n-n
PP PP ETX
```
   Die Anzahl (n) der gesendeten Datensätze ist abhängig von der [DataStorage] Implementierung.

Im Anschluss extern:

1) Status lesen (FSP017), muss nun `0x03` sein

Name	**FSP226_SW_FSP_FILL_FGSTIMULI_RAM**
Adresse	**$0xE2_H/226_D/0x4532_{ASCII}$**
Tiefe	dynamisch
I/O	lesen / schreiben
Reset	$0x(\text{siehe Beschreibung})_H$

FSP226 ermöglicht das RAM für den FG-Stimuli mit Daten zu füllen.

Es werden die Daten einer `.hex`-Datei ins RAM geschrieben.

```
:0C 0000 00 57170D57170C81446F860706 38
:0C 0001 00 57170D550D1C8290A8610606 D3
||  |    |  |                       \-- Pruefsumme
||  |    |  \--------------------------- Daten (13 Byte)
||  |    \------------------------------ Typ
||  \------------------------------------ Adresse
|\------------------------------------- Laenge der Daten (13 Byte)
\-------------------------------------- Start
```

Das RAM des FG-Stimuli hat eine max. Tiefe von 4096 x 96Bit.

Um den Datenbus des NIOS nicht unnötig aufzublähen, werden die Daten zu je 6-Bit ins RAM geschrieben.

Also jeweils 16 x 6 Bit mit 16 Adressen anstelle 1 x 96 Bit mit einer Adresse.

D.h.
```
96-Bit Adresse 0x0000 <=> 6-Bit Adressen 0x0000 ... 0x000F
96-Bit Adresse 0x0001 <=> 6-Bit Adressen 0x0010 ... 0x001F
96-Bit Adresse 0x0002 <=> 6-Bit Adressen 0x0020 ... 0x002F
usw.
```

Seitens der Hardware werden diese Daten aber mit 96 aus dem RAM ausgelesen.

Schreiben des FSP:

```
STX PID PID GW MA FSP FSP [.hex-Datei] PP PP ETX
```

NIOS Interface
==========

```
FG_RAM_NmbOfDataTupels_Pio(16)
```

[15..12] 0000

[11..0] NmbOfDataTupels(12 Bit)

```
FG_RAM_Ctl_Addr_Data_Pio  (32)
```

[31] Enable(1 Bit),
wenn '1' werden die o-Ausgänge des "ACU_FG_Stimuli" an "fg_quad_scu_bus" gelegt, andernfalls die i-Eingänge des "ACU_FG_Stimuli".

[30] FGEnable(1 Bit),
Entspricht Bit[1] des FG Ctrl Registers, starten/stoppen des FG.

[29] RAM_WrEnable(1 Bit),
Wenn '1' werden Daten an den Eingängen "RAM..." ins RAM übernommen.

[28] FGReset(1 Bit),
Wenn '1' erfolgt in 'ACU_FG_Stimuli' UND 'fg_quad_scu_bus' ein Reset.

[27..24] (0000)

[23..8] RAM_WrAddress(16 Bit),
Adresse (zusammen mit 'RAM_WrEnable').

[7..6] (00)

[5..0] RAM_WrData(6 Bit),
Daten (zusammen mit 'RAM_WrEnable')

Name	**FSP227_SW_FSP_BIT_MAN_ENHANCED**
Adresse	**$0xE3_H/227_D/0x4533_{ASCII}$**
Tiefe	dynamisch
I/O	lesen / schreiben
Reset	$0x(\text{siehe Beschreibung})_H$

FSP227 ermöglicht Bitmanipulationen in beliebigen FSPs, beliebiger Module.

Basiert auf dem FSP241_SW_BitManipulation, der (leider) nur MFU interne FSPs bedienen kann.

Einschränkung: es können nur die Bits zwischen 0..255 manipuliert werden.

Aufbau der Anforderung

`STX PID PID GW MA FSPhi FSPlo USINrTarget ModuleNrTarget FSPmHi FSPmLo BITPosHi BITPosLo BITVal PP PP ETX` (17 Byte)

`STX`	StartOfText (0x02)
`PID`	Schreibanforderung (0x57 0x52)
`GW`	Gateway Adresse (0x30 da Gateway die MFU ist)
`MA`	Modul Adresse (0x30 da Modul die MFU ist)
`FSPhi FSPlo`	FSP Nummer (0x45, 0x33)
`USINrtarget`	USI an der das Modul zu finden ist (0x30(MFU), 0x31..0x0A)
`ModuleNrTarget`	Modul Nummer an diesem USI (0x30(MFU), 0x31..0x38)
`FSPmHi FSPmLo`	zu manipulierendes FSP (1...255)
`BITPosHi BITPOsLo`	Bitposition (0...255, gibt die Bitposition an)
`BITVal`	Wertigkeit des Bit (gleich 0(0x30) -> Bit löschen, ungleich 0 -> Bit setzen)
`PP`	Prüfsumme (Prüfsumme der Daten in Hex als 2 Byte ASCII)
`ETX`	EndOfText (0x03)

Name	**FSP228_SW_SCOPE_HEAD**
Adresse	**$0xE4_H/228_D/0x4534_{ASCII}$**
Tiefe	dynamisch
I/O	lesen / schreiben
Reset	$0x(\text{siehe Beschreibung})_H$

FSP228 ermöglicht die Umschaltung der Soll-/Istwertanzeige auf dem TFT.

Noch nicht vollständig implmentiert.

Name	**FSP229_SW_HighSpeedStream_Synchronized**
Adresse	**$0xE5_H/229_D/0x4535_{ASCII}$**
Tiefe	dynamisch
I/O	lesen / schreiben
Reset	$0x(\text{siehe Beschreibung})_H$

FSP229 basiert auf dem „FSP237_SW_HighSpeedStream“ und gibt ebenfalls eine bestimmte Anzahl von Daten aus, die über einen wählbaren HighSpeed Kanal gesammelt werden.

Bedingung hierbei ist, dass das Zielmodul über diese Funktionalität verfügt. Dazu müssen im Zielmodul die FSP017 und FSP018 vorhanden sein.

Die Unterschiede dabei sind:

- die Datenerfassung ist synchronisiert, d.h. die erfassten Daten werden nicht nach einer voreinstellbaren Zeit gepollt, sondern es wird jeder neue Messwert erfasst
- die Anzahl lesbarer HS Datensätze ist größer 2^{16} (=> max. 2^{32}), ist wegen des begrenzten RAM im Modul/der MFU aber nicht nutzbar). Dabei ist die Anzahl lesbarer Messwerte im Modul FSP017 zu finden und wird nicht vom Nutzer gesetzt
- vor dem Lesen wird
  1.) der Regler gesperrt
  2.) im auszulesenden Modul seitens der MFU eine Initialisierung für den Ausleseprozess gestartet.

Bevor das FSP gelesen werden kann, muss es beschrieben werden mit:

```
STX PID PID GW MA FSP FSP [USINr.] PP PP ETX
```

Mit:

PID	WR
[USINr.]	USI Nummer (0x01..0x0B)

Im Anschluss wird das FSP gelesen.

Die MFU führt daraufhin die folgenden Aktionen aus:

- sämtliche USIs seitens der MFU werden deselektiert
- es wird geprüft ob die aktuelle MFU Firmware diese Funktion unterstützt
- FSP017 des Zielmoduls lesen. Dort ist die Anzahl der erfassbaren Messwerte zu finden.
- ist FSP017 des Zielmoduls lesbar. die passende Speichergröße in der MFU reservieren
- den Regler sperren
- im Zielmodul FSP018[0] einen Reset der Datenerfassung ausführen
- im Zielmodul FSP018[2] den Trigger zur Datenerfassung aktivieren
- nun wird die Anzahl von Daten laut FSP017 des Zielmoduls, im Zielmodul erfasst und auch dort gespeichert
- FSP017 des Zielmoduls lesen bis Bit[1] gesetzt ist (Messung beendet)
- Zielmodul FSP018[1] setzen, dies ermöglicht das lesen der Messdaten
- die MFU setzt das Kommando „TriggerSometing“ ab, dadurch startet das Zielmodul den Datentransfer der Messwerte über den HighSpeed Kanal. Dazu wird der „normale HighSpeed Stream“ unterbrochen. Stattdessen findet der Messwertdatentransfer statt.
- sind alle Daten übertragen werden diese an die übergeordnete Instanz (Kontrollsystem/Anwender PC) ausgegeben
- der „normale HigHSpeed Stream“ wird wieder auf der USI aktiviert

Die Daten sehen dabei wie folgt aus:

```
STX GW MA FSP FSP
[Byte7 Byte6 Byte5 Byte4 Byte3 Byte2 Byte1 Byte0] ← Datensatz n
[Byte7 Byte6 Byte5 Byte4 Byte3 Byte2 Byte1 Byte0] ← Datensatz n-1
[..]
[Byte7 Byte6 Byte5 Byte4 Byte3 Byte2 Byte1 Byte0] ← Datensatz n-n
PP PP ETX
```

Name	**FSP230_SW_intScopeHeaderReadOut**
Adresse	**$0xE6_H/230_D/0x4536_{ASCII}$**
Tiefe	6 Byte / 48 Bit
I/O	lesen
Reset	$0x(\text{siehe Beschreibung})_H$

Der interne SW FSP230 ist ein CPU Software FSP und stellt die TimeScale, TimeBase, TriggerOffsetEcho, und TriggerAddress des internen Oszilloskops über den USB→PC Anschluss zur Verfügung.

Diese Daten werden von PCA (PowerConfigAdvanced) zur Darstellung der internen Oszilloskopdaten benötigt.

FSP230 kann nur gelesen werden. Schreibanforderungen bekommen eine NACK Antwort mit Fehlercode.

Die Informationen des FSP230 werden dabei aus Inhalten der „FSP116_intScopeSettings“ und „FSP117_intScopeTriggerReadOut“ gewonnen.

Ausgabestring nach der Leseanforderung:

```
STX GW MA -FSP-  -TS--  ----TB-----  ----TOE---- ---TA--- --PP-  ETX
02  30 30 45 36  ts ts  tt tt tt tt  toe toe toe ta ta ta pp pp  03
```

TS	TimeScale	Eingestellte Zeitskalierung für TimeBase des internen Oszilloskops (2 Byte ASCII = 8 Bit HEX)
TB	Timebase	Eingestellte Zeitbasis des internen Oszilloskops (4 Byte ASCII = 16 Bit HEX)
TOE	TriggerOffsetEcho	Triggeroffset vom Nullpunkt der X-Achse (3 Byte ASCII = 12 Bit HEX)
TA	TriggerAddress	Speicheradresse an der das Triggerereignis stattgefunden hat (3 Byte ASCII = 12 Bit HEX)

Der Speicher des int. Oszilloskops ist als Ringspeicher ausgeführt. D.h. solange kein Triggerereignis erfolgt werden die Daten zyklisch in diesen Ringspeicher eingetragen.

Findet nun ein Triggerereignis statt, wird in `TriggerAddress` die Adresse gespeichert an der das Ereignis stattgefunden hat, `TriggerOffsetEcho` gibt die Trigger Offsetverschiebung vom Nullpunkt der X-Achse an.

Name	**FSP231_SW_intScopeDataStreamReadOut**
Adresse	**$0xE7_H$/231_D/$0x4537_{ASCII}$**
Tiefe	12008 Byte / 96064 Bit (bis MFU FW 7.4.x) 6008 Byte / 48064 Bit (ab MFU FW 7.5.0)
I/O	lesen
Reset	$0x(\text{siehe Beschreibung})_H$

Der interne FSP231 ist ein CPU Software FSP und stellt die eigentlichen Daten des internen Oszilloskops als Stream über den USB→PC Anschluss zur Verfügung.

Diese Daten werden von PCA (PowerConfigAdvanced) zur Darstellung der internen Oszilloskopdaten benötigt.

FSP231 kann nur gelesen werden. Schreibanforderungen bekommen eine NACK Antwort mit Fehlercode.

Sollen die Daten beginnend vom Triggeroffset und nicht vom RAM Speicher 0 ausgegeben werden bitte den FSP245_SW_intScopeDataStream benutzen.

<u>Bis MFU FW 7.4.x gilt folgendes:</u>

Ausgabestring nach der Leseanforderung:

```
STX GW MA -FSP-  -CH8(12Bit) CH7(12Bit) CH7(12Bit) CH5(12Bit)
CH4(12Bit) CH3(12Bit) CH2(12Bit) CH1(12Bit)- -PP-   ETX
```

```
02  30 30 45 37   dd dd dd   dd dd dd   dd dd dd   dd dd dd
dd dd dd   dd dd dd   dd dd dd   dd dd dd    pp pp  03
```

CHx - Daten des Kanals

Insgesamt werden 500 x **<u>8</u>** Kanäle übertragen. D.h. der gelb markierte Bereich wird also insgesamt 500-mal übertragen. Beginnend mit dem Wert 1 für Kanal 8 bis 1, Wert 2 für Kanal 8 bis 1 usw. Die Prüfsumme PP folgt nach Wert 500 für Kanal 8 bis 1 über die gesamten Daten.

<u>Ab MFU FW 7.5.0 gilt folgendes:</u>

Ausgabestring nach der Leseanforderung:

```
STX GW MA -FSP-  -CH4(12Bit) CH3(12Bit) CH2(12Bit) CH1(12Bit)- -PP-
ETX
```

```
02  30 30 45 37   dd dd dd   dd dd dd   dd dd dd   dd dd dd    pp pp
03
```

CHx - Daten des Kanals

Insgesamt werden 500 x **<u>4</u>** Kanäle übertragen. D.h. der gelb markierte Bereich wird also insgesamt 500-mal übertragen. Beginnend mit dem Wert 1 für Kanal 4 bis 1, Wert 2 für Kanal 4 bis 1 usw. Die Prüfsumme PP folgt nach Wert 500 für Kanal 4 bis 1 über die gesamten Daten.

Name	**FSP232_SW_intSystemParameters**
Adresse	**$0xE8_H/232_D/0x4538_{ASCII}$**
Tiefe	dynamisch
I/O	lesen
Reset	$0x(\text{siehe Beschreibung})_H$

Der interne FSP232 ist ein CPU Software FSP und stellt die ACU Systemparameter als Stream über den USB→PC Anschluss zur Verfügung.

FSP232 kann nur gelesen werden. Schreibanforderungen bekommen eine NACK Antwort mit Fehlercode.

Nach einer Leseanforderung werden die in der MFU gespeicherten Systemparameter übertragen. Diesen wird der reguläre USI Header vorangestellt und der Stream mit Prüfsumme und ETX abgeschlossen.

Wichtiger Hinweis:
Die Systemparameter sind als USI konforme Strings im MFU Speicher hinterlegt. d.h. jeder dieser Einzelstrings wird mit einem STX [0x02] eingeleitet und einer Prüfsumme über die Daten dieses Einzelstrings, sowie einem ETX[0x03] abgeschlossen. Die Summe aller Systemparameter wiederrum wird ebenfalls als USI konformer String ausgegeben. Dieser wird also ebenfalls durch ein ETX eingeleitet und mit Prüfsumme (diesmal über alle vorherigen Parameterstrings) und ETX abgeschlossen.

□	0	0	E	8	□	W	R	0	0	F	1	0	D	0	1	0	0	7	5	□	□	W	R
02	30	30	45	38	02	57	52	30	30	46	31	30	44	30	31	30	30	37	35	03	02	57	52

Einleitung der FSP Antwort mit STX, Gateway, Moduladresse, FSP Nummer ($E8_H = 223_D$)
erster Parameterstring
Anfang des zweiten Parameterstring
es folgen ggf. weitere Parameterstrings

0	0	0	0	0	0	0	□	4	C	□
30	30	30	30	30	30	30	03	34	43	03

Rest des letzten Parameterstring
Abschluss der FSP Antwort mit Prüfsumme (über alle Parameterstrings, inkl. deren STX, „WR“, Gateway, Modulnummer, Daten, Prüfsummen und ETX) und ETX

Das Speichern der Systemparameter wird mittels Bit[7] im „FSP242_SW_CPU_Status“ (siehe Seite 112) eingeleitet und auch bestätigt.

Name	**FSP233_SW_InterlockTexts**
Adresse	**$0xE9_H/233_D/0x4539_{ASCII}$**
Tiefe	dynamisch
I/O	lesen / schreiben
Reset	$0x(\text{siehe Beschreibung})_H$

Der interne FSP233 ist ein CPU Software FSP und beinhaltet die für das ACU System in der MFU hinterlegten Interlocktexte.

Weitere Informationen zum beschreiben dieses FSP finden Sie im Dokument "ACU-MFU-FSP233-Interlocktexte programmieren".

Nach einer Leseanforderung werden die in der MFU gespeicherten Interlocktexte übertragen. Diesen wird der reguläre USI Header vorangestellt und der Stream mit Prüfsumme und ETX abgeschlossen.

Name	**FSP234_SW_MDS**
Adresse	**$0xEA_H/234_D/0x4541_{ASCII}$**
Tiefe	dynamisch
I/O	lesen / schreiben
Reset	$0x(\text{siehe Beschreibung})_H$

Der interne FSP234 ist ein CPU Software FSP und liefert die MDS der an den USI gefundenen Module im ACU System.

Dem voran wird die MDS der MFU selbst gesendet.

Wird FSP234 geschrieben, startet dies einen USI Rescan. In diesem Fall werden geschickte Dummy-Daten und Prüfsumme nicht ausgewertet.

```
STX PID PID GW MA FSP FSP Data Data PP PP ETX
0x02 0x57 0x52 0x30 0x30 0x45 0x41 0x30 0x30 0x30 0x30 0x03
```

Nach einer Leseanforderung werden die im ACU System gefundenen Moduldeskriptoren wie folgt ausgegeben:

				Datenbytes für Prüfsumme											
Direkte Antwort des FSP234				MDS der MFU					MDS gefundener Module					Prüf-summe	
STX	GW	MA	FSP	USINr. (GW)	MA	Mod. FSP	MDS ACU	PP	USINr.	MA	Mod. FSP	MDS Modul	PP	PP	ETX
0x02	0x30	0x30	0x4541	0x30	0x30	0x3030	(...EOD)		0x3u	0x3m	0x3030	(...EOD)		0x.. 0x..	0x03
Einleitung mit STX, der Gateway- und Modul-, sowie die FSP Nummer				Die MDS der MFU wird wie die nachfolgenden Modul MDS mit der USI Nummer (in diesem Falle die Gatewaynummer 0, da es sich ja um die MFU handelt), der Modul- und FSP Nummer eingeleitet. Der Deskriptor wird mit dem End-deskriptor (…EOD) und der an-schließenden Prüfsumme (PP = 2 Byte ASCII) abgeschlossen. Es folgt KEIN ETX!					Die MDS der an der MFU ange-schlossenen und identifizierten Module. 'u' bei USINr. liegt dabei zwischen 1...10, 'm' bei Mod.Nr. zwischen 1...8. Die einzelnen Deskriptoren werden mit dem Enddeskriptor (…EOD) und der anschließenden Prüf-summe (PP = 2 Byte ASCII) abgeschlossen, es folgt erst nach der letzten MDS ein ETX!					2 Byte ASCII Prüf-summe über alle Daten-bytes	ETX nach der letzten gesende-ten MDS.

Name	**FSP235_SW_Logbook**
Adresse	**$0xEB_H/235_D/0x4542_{ASCII}$**
Tiefe	dynamisch
I/O	lesen / schreiben
Reset	$0x(\text{siehe Beschreibung})_H$

Der interne FSP235 ist ein CPU Software FSP und liefert das MFU interne Logbuch.

Jeder Eintrag des Logbuchs ist 8 Byte lang.
Die ersten 5 Bytes beinhalten das Datum und die Uhrzeit des Eintrags.
Die RTC liefert die Daten in BCD.
Im Logbuch sind diese aber aus Platzgründen in komprimiert in HEX abgelegt

Bezeichnung	Bereich	Bitdarstellung	Kurzformat
Tag	1..31	5 Bit 00001..11111	D
Monat	1..12	4 Bit 0001..1100	M
Jahr	0..99	7 Bit 0000000..1100011	Y
Stunde	0..23	5 Bit 00000..10111	H
Minute	0..59	6 Bit 000000..111011	m
Sekunde	0..59	6 Bit 000000..111011	S
Wochentag	0..6	3 Bit 000.110 (0 = Sonntag)	W

Die obige BCD Darstellung wird zur Ausgabe in die nachfolgende HEX Darstellung transferiert.

Byte	Inhalt
0	DDDD DMMM
1	MYYY YYYY
2	HHHH Hmmm
3	mmmS SSSS
4	SWWW xxxx

Die drei nachfolgenden Bytes enthalten den eigentlichen Logbucheintrag.

Byte	Inhalt	Details
5	Spezifikation	0 = normaler Eintrag
		1 = Fehler
		2 = Warnung
		3 = Interlock
6	Eintrag MSB	Nibbel 3..2
7	Eintrag LSB	Nibbel 1..0

Normal:

Nibble	Bedeutung	Bereich
3	Logbucheintrag	$(0..255)_D$
2		
1		
0		

Fehler:

Nibble	Bedeutung	Bereich
3	USI Nummer	0x0..0xB $(0..11)_D$
2	Modul Nummer	0x0..0x7 $(0..7)_D$
1	Fehlernummer	0x00...0xFF (1.127)
0		

Warnung:

Nibble	Bedeutung	Bereich
3	Bitposition der Warnung in Dezimal	$(0..31)_D$
2		
1		
0		

Interlockeintrag:

Nibble	Bedeutung	Bereich
3	USI Nummer	0x0..0xB $(0..11)_D$
2	Modul Nummer	0x0..0x7 $(0..7)_D$
1	Interlockbit	0x00...0xFF (0.255)
0		

FSP235 kann nur mit einem legitimierten USB Stick geschrieben werden. Andernfalls bekommen Schreibanforderungen eine NACK Antwort mit Fehlercode.

Ist ein legitimierter USB Stick eingesteckt und wird FSP235 geschrieben, wird das Logbuch gelöscht. In diesem Fall werden geschickte Dummy-Daten und Prüfsumme nicht ausgewertet.

STX PID PID GW MA FPS FSP Data Data PP PP ETX

0x02 0x57 0x52 0x30 0x30 0x45 0x42 0x30 0x30 0x30 0x30 0x03

Name	**FSP236_SW_Delete_Errors**
Adresse	**$0xEC_H/236_D/0x4543_{ASCII}$**
Tiefe	dynamisch
I/O	schreiben
Reset	$0x(\text{siehe Beschreibung})_H$

Der interne FSP236 ist ein CPU Software FSP und löscht den MFU internen Fehlerspeicher.

In diesem Fall werden geschickte Dummy-Daten und Prüfsumme nicht ausgewertet.

```
STX PID PID GW MA FPS FSP Data Data PP PP ETX
0x02 0x57 0x52 0x30 0x30 0x45 0x43 0x30 0x30 0x30 0x30 0x03
```

Name	**FSP237_SW_HighSpeedStream**
Adresse	**$0xED_H/237_D/0x4544_{ASCII}$**
Tiefe	dynamisch
I/O	lesen / schreiben
Reset	$0x(\text{siehe Beschreibung})_H$

Gibt eine bestimmte, frei wählbare Anzahl von USI HighSpeed Daten mit einer ebenfalls frei wählbaren Zeitverzögerung zurück.

Dazu muss der FSP zunächst beschrieben werden mit:

```
STX PID PID GW MA FSP FSP
[USINr.][Quantity][Quantity][Quantity][Quantity][Pause/Skip] PP PP ETX
```

Mit:

`PID`	WR
`[USINr.]`	USI Nummer (0x1..0xB)
`[Quantity]`	Anzahl von Werten ‚n' (1..65536)
`[Pause/Skip]`	Wartezeit zwischen zwei Werten (0..15) in uSekunden, bzw. bei neuer MFU FW die Anzahl von Werten die zwischen zwei erfassten Werte ignoriert werden sollen. Wird der FSP anschließend gelesen, werden ‚n' HighSpeed Datensätze gesendet:

```
STX GW MA FSP FSP
[Byte7 Byte6 Byte5 Byte4 Byte3 Byte2 Byte1 Byte0] ← Datensatz n
[Byte7 Byte6 Byte5 Byte4 Byte3 Byte2 Byte1 Byte0] ← Datensatz n-1
[..]
[Byte7 Byte6 Byte5 Byte4 Byte3 Byte2 Byte1 Byte0] ← Datensatz n-n
PP PP ETX
```

Ab MFU SE FW 7.4.0, bzw. MFU LE FW 7.3.0 ist die Datenerfassung synchronisiert, d.h. die erfassten Daten werden nicht mehr nach einer voreinstellbaren Zeit gepollt, sondern es wird jeder neue Messwert erfasst. `[Pause]` gibt jetzt nicht mehr die Zeitdauer zwischen zwei erfassten Messwerten in uSekunden an, sondern wird zu `[Skip]` und bestimmt die Anzahl von Messwerten die zwischen zwei erfassten Messwerten verworfen werden sollen.

Name	**FSP238_SW_SnapshotHighSpeed**
Adresse	**$0xEE_H/238_D/0x4545_{ASCII}$**
Tiefe	dynamisch
I/O	lesen
Reset	$0x(\text{siehe Beschreibung})_H$

Liefert einen Schnappschuss der Highspeed Daten aller aktiven und im HighSpeed Modus befindlichen USIs (1..11) zurück.

Eine komplette Antwort lautet:

```
STX GW MA FSP FSP
[USI01Nr. Byte7 Byte6 Byte5 Byte4 Byte3 Byte2 Byte1 Byte0]
[USI02Nr. Byte7 Byte6 Byte5 Byte4 Byte3 Byte2 Byte1 Byte0]
[USI03Nr. Byte7 Byte6 Byte5 Byte4 Byte3 Byte2 Byte1 Byte0]
[USI04Nr. Byte7 Byte6 Byte5 Byte4 Byte3 Byte2 Byte1 Byte0]
[USI05Nr. Byte7 Byte6 Byte5 Byte4 Byte3 Byte2 Byte1 Byte0]
[USI06Nr. Byte7 Byte6 Byte5 Byte4 Byte3 Byte2 Byte1 Byte0]
[USI07Nr. Byte7 Byte6 Byte5 Byte4 Byte3 Byte2 Byte1 Byte0]
[USI08Nr. Byte7 Byte6 Byte5 Byte4 Byte3 Byte2 Byte1 Byte0]
[USI09Nr. Byte7 Byte6 Byte5 Byte4 Byte3 Byte2 Byte1 Byte0]
[USI10Nr. Byte7 Byte6 Byte5 Byte4 Byte3 Byte2 Byte1 Byte0]
[USI11Nr. Byte7 Byte6 Byte5 Byte4 Byte3 Byte2 Byte1 Byte0]
PP PP ETX
```

mit:

`[ ]` sofern diese USI aktiv ist. Andernfalls wird die entsprechende Zeile bei der Ausgabe übersprungen.

Name	**FSP239_SW_Debug**
Adresse	**$0xEF_H/239_D/0x4546_{ASCII}$**
Tiefe	65536 Byte / 524288 Bit
I/O	lesen / schreiben
Reset	$0x(\text{siehe Beschreibung})_H$

Dieser FSP liefert Inhalte von Flashsektoren zurück -> für Debugzwecke

Wird der FSP beschrieben, muss der zu lesende Sektor mitgeteilt werden.

```
STX GW MA FSP FSP SectorNr.High SectorNr.Low PP PP ETX
```

Beipiel: Sektor 5 lesen

◻	W	R	0	0	E	F	0	5	0	5	◻
02	57	52	30	30	45	46	30	35	30	35	03

Wird der FSP gelesen liefert er die Daten des zuvor gewählten Sektors.

Nach dem Bootvorgang der MFU ist immer der Sektor 5 (Parameter) gewählt.

Name	**FSP240_SW_RealTimeClock**
Adresse	**$0xF0_H/240_D/0x4630_{ASCII}$**
Tiefe	7 Byte / 56 Bit
I/O	Lesen / schreiben
Reset	$0x(\text{siehe Beschreibung})_H$

Der interne FSP240 ist ein CPU Software FSP und dient dem Stellen und Auslesen der MFU internen RTC (Real Time Clock).

FSP240 kann geschrieben und gelesen werden.

Zum Beschreiben wird eine reguläre USI Schreibanforderung mit folgender Syntax an dieses FSP gesendet,

```
STX PID PID GW MA FSPhi FSPlo wDay wDay Day Day Month Month Year Year
Hour Hour Minute Minute Second Second PP PP ETX
```

OLD_RTC (RTC im FPGA)

[55..48]	`wDay`, Wochentag	(0x00..0x06, wobei Sonntag = 0)
[47..40]	`Day`, Tag	(0x01..0x1F → 1..31))
[39..32]	`Month`, Monat	(0x01..0x0C → 1..12)
[31..24]	`Year`, Jahr	(0x00..0x63 → 0..99)
[23..16]	`Hour`, Stunden	(0x00..0x17 → 0..23)
[15..8]	`Minute`, Minuten	(0x00..0x3B → 0..59)
[7..0]	`Second`, Sekunden	(0x00..0x3B → 0..59)

Jeder Einstellwert (Tag, Monat usw.) besteht aus 2 Byte ASCII mit hexadezimaler Information. Nicht benötigte Bytes wie das MSB beim Wochentag und Monat werden als 0x30 gesendet.

Wird der FSP gelesen wird die Zeitinformation in der gleichen Reihenfolge ausgegeben.

Die Millisekunden werden weder von extern geschrieben noch ausgegeben. Nach dem senden der Uhrzeitdaten vom PC werden die Millisekunden auf 0 gesetzt und die RTC gestartet.

NEW_RTC (RTC im eigenen Chip, FPGA extern)

Alle Daten sind BCD codiert.

[55..48]	`wDay`, Wochentag	(0x30, 0x30 .. 0x30, 0x36, → 0 .. 6, mit Sonntag = 0)
[47..40]	`Day`, Tag	(0x30, 0x31 .. 0x33, 0x31 → 1 .. 31))
[39..32]	`Month`, Monat	(0x30, 0x31 .. 0x31, 0x32 → 1 .. 12)
[31..24]	`Year`, Jahr	(0x30, 0x30 .. 0x39, 0x39 → 0 .. 99)
[23..16]	`Hour`, Stunden	(0x30, 0x30 .. 0x32, 0x33 → 0 .. 23)
[15..8]	`Minute`, Minuten	(0x30, 0x30 .. 0x35, 0x39 → 0 .. 59)
[7..0]	`Second`, Sekunden	(0x30, 0x30 .. 0x35, 0x39 → 0 .. 59)

Jeder Einstellwert (Tag, Monat usw.) besteht aus 2 Byte ASCII mit BCD Information. Das nicht benötigte MSB Byte beim Wochentag wird immer als 0x30 gesendet.

Wird der FSP gelesen wird die Zeitinformation in der gleichen Reihenfolge ausgegeben.

Erst nach dem erfolgreichen senden aller Uhrzeitdaten vom PC wird die RTC gesetzt und ggf. gestartet.

Im folgenden Beispiel wird die RTC auf 16:15:25 20.06.2012 gestellt.

◻	W	R	0	0	F	0	0	3	2	0	0	6	1	2	1	6	1	5	2	0	0	5	◻
02	57	52	30	30	46	30	30	33	32	30	30	36	31	32	31	36	31	35	32	30	30	35	03

Name	**FSP241_SW_BitManipulation**
Adresse	**$0xF1_H/241_D/0x4631_{ASCII}$**
Tiefe	3 Byte / 24 Bit
I/O	schreiben
Reset	$0x(\text{siehe Beschreibung})_H$

Der interne FSP241 ist ein CPU Software FSP und dient der Bitmanipulation der FSP in der MFU.

FSP241 kann nur geschrieben werden. Leseanforderungen bekommen eine NACK Antwort mit Fehlercode.

Zum Beschreiben wird eine reguläre USI Schreibanforderung mit folgender Syntax an dieses FSP gesendet,

```
STX PID PID GW MA FSPhi FSPlo [FSPmhi FSPmlo] [BIThi BITlo] [BITVal
BITVal] PP PP ETX
```

Einschränkung: es können nur die Bits 0..255 manipuliert werden.

[23..16] `FSPmhi/FSPmlo`, zu manipulierendes FSP (nur beschreibbare FSP)

[15..8] `BIThi/Bitlo`, Bitnummer (gibt die Bitposition an, beginnt bei 0…255)

[7..0] `BITVal`, Wertigkeit des Bit,
gleich = 0x00 → Bit löschen,
ungleich ! = 0x00 → Bit setzen

Name	**FSP242_SW_CPU_Status**
Adresse	**$0xF2_H/242_D/0x4632_{ASCII}$**
Tiefe	4 Byte / 32 Bit
I/O	lesen / schreiben
Reset	$0x(\text{siehe Beschreibung})_H$

Der interne FSP242 ist ein CPU Software FSP und repräsentiert 'CPU_Status' Informationen.

Es können allerdings nur die Bit[7] 'CPU_STATUS_RECEIVE_SYSPARAMETERS' und Bit[31] 'CPU_STATUS_DISABLE_CIRCULAR_INTERLOCK_CHECK' seitens des PC geschrieben werden. Alle anderen Bit sind **Read Only**.

Bit	Bezeichnung	Beschreibung
[31]	CPU_STATUS_DISABLE_CIRCULAR_INTERLOCK_CHECK	Wenn [1] wird KEINE zyklische Interlockabfrage durchgeführt. TIMER_INTERLOCK_CIRCULAR_CHECK ist gestoppt.
[30]	CPU_STATUS_CMD_TRIGGER_SOMETHING	Wenn [1] wird das Kommando '`cCMDTiggerSomething`' von 'inst_MFUSwitchingOperations' ausgegeben.
[29]	CPU_STATUS_DISABLE_ALL_TIMERS	Ab Nios 7.0801. Wenn [1] sind alle Hardware Timer gestoppt: TIMER_STD_SCREEN_UPDATE TIMER_WARNING_UPDATE TIMER_INTERLOCK_CIRCULAR_CHECK TIMER_RECORD TIMER_RETURN_TO_STD_SCREEN TIMER_TFT_DIM_BACKLIGHT TIMER_UNIVERSAL
[28..21]		n.u.
[20]	CPU_STATUS_SYSTEM_HAS_INTERLOCKS	Wenn [1] steht mindestens ein Interlock an
[19]	CPU_STATUS_RECEIVING_SYSPARAMETERS_RAM	Wenn [1] werden gerade Parameter vom PC empfangen
[18]	CPU_STATUS_MODULES_VERIFIED	Wird [1] sofern der FSP243 erfolgreich die gefundenen mit den gewünschten Modulen verglichen hat.
[17]	CPU_STATUS_PARAMETERS_VALID	Wenn [1] sind die in der MFU geladenen Parameter gültig
[16]	CPU_STATUS_LOADING_INTERNAL_PARAMETERS	Wenn [1] werden gerade die MFU internen Parameter geladen
[15]	CPU_STATUS_WATCHDOG	Perm. Wechsel zwischen [1] und [0] als Herzschlag der CPU Funktionalität
[14]	CPU_STATUS_BOOTSEQUENZ_COMPLETED	[1] wenn die MFU die Bootsequenz nach dem PowerUp abgeschlossen hat
[13]	CPU_STATUS_VNC1L_NOT_PROGRAMMED	[1] wenn der VNC1L (USB Controller) nicht programmiert ist
[12]	CPU_STATUS_USB_DEVICE_PERMITTED	Nur wenn [1] dürfen USB-, bzw. bestimmte Menüzugriffe erfolgen, wird [1] wenn ein "zugelassener" USB Stick an die MFU angeschlossen wird
[11]	CPU_STATUS_USB_DEVICE_DETECTED	Wenn [1] wurde ein eingesteckter USB

		Stick an der MFU Front entdeckt
[10]	CPU_STATUS_USING_INTERNAL_PARAMETERS	Wenn [1] werden die interen Systemparameter aus dem seriellen Flash benutzt. Wenn [0] wurden diese Parameter per PC flüchtig überschrieben.
[9]	CPU_STATUS_ERROR_OCCURED	Wenn [1] ist ein Fehler Software aufgetreten. Dieser wird sowohl im Logbuch als auch flüchtig im Speicher vermerkt.
[8]	CPU_STATUS_WARNING_OCCURED	Wenn [1] ist eine Systemwarnung aufgetreten welche nicht zwangsläufig das Gerät abgeschaltet hat, den Benutzer jedoch darauf hinweist.
[7]	CPU_STATUS_RECORDING_SYSPARAMETERS	Beim Übergang von [0] nach [1] wird der Parametersektor im seriellen Flash der MFU gelöscht. Anschließend werden alle über den USB empfangenen USI Kommandos an im seriellen Flash gespeichert. Beim Übergang von [1] nach [0] wird der Speichervorgang abgeschlossen.
[6]	CPU_STATUS_FETCHING_INTERLOCKS	Wenn [1] globale Info darüber, dass gerade Interlocks von der CPU getestet werden
[5]	MPU_STATUS_MFU_CAN_NOT_TRANSFER_ANY_DATA_RIGHT_NOW	Wenn [1] können gerade KEINE Daten an die MFU gesendet, bzw. von dieser gelesen werden, weil diese z.B. einen USI (Re)scan oder USB-Speicherzugriffe durchführt.
[4]	CPU_STATUS_RESET_BUTTON_ACTIVE	Wenn [1] ist die RESET Taste an der MFU Vorderseite gedrückt. Im Standardbildschirm kann nun anstelle 1/50 in 1/200 Schritten der Handsollwert geändert werden.
[3]	CPU_STATUS_STDSCREEN_ACTIVE	Der Standardbildschirm ist aktiv und wird angezeigt.
[2]	CPU_STATUS_PSU_IS_REMOTE	Das ACU System ist im Remote Betrieb.
[1]	-	n.u.
[0]	CPU_STATUS_PSU_IS_ON	Das ACU System ist eingeschaltet (Regler ist freigegeben)

Name	**FSP243_SW_VerifyHWConfig_ModuleClasses**
Adresse	**0xF3/243$_D$/0x4633$_{ASCII}$**
Tiefe	dynamisch
I/O	schreiben
Reset	0x(siehe Beschreibung)$_H$

Über diesen FSP lässt sich die korrekte Hardwarekonfiguration des ACU-Systems verifizieren.

Dieses FSP beinhaltet dabei die Modul(-Sub-)klassen der an den USIs erwarteten Module. Der NiosII der MFU liest dieses FSP und vergleicht dessen Inhalt mit den Modul(-Sub-)klassen der gefundenen Module. Besteht Konsistenz wird die Reglerfreigabe ermöglicht.

Die Datentiefe dieses FSP ist abhängig von der ACU Konfiguration und der Anzahl der an der MFU angebundenen Module.

Die in diesem FSP zu verarbeitenden Daten sind dabei wie folgt organisiert:

```
STX PID PID GW MA FSP FSP { [Head Head][ModuleClass Mo-
dulClass][ModuleSubClass ModulSubClass] } PP PP ETX
```

mit

`[Head]`	oberes Nibble USI Nummer, unteres Nibbel Modul Nummer Bsp.: 0x12$_H$/0x31,0x32$_{ASCII}$ für USI 1, Modul 2
`[ModuleClass]`	2 Nibble Hex (2 Byte ASCII) Modulklasse des in [Head] beschriebenen Moduls
`[ModuleSubClass]`	2 Nibble Hex (2 Byte ASCII) Modul-Sub-klasse des in [Head] beschriebenen Moduls
`{}`	stetige Wiederholung obiger Beschreibung für alle Module

Wird dieses FSP gefüllt, wird sofort die gefundene Modulkonfiguration mit der in diesem FSP gewünschten verglichen. Ist dieser Vergleich erfolgreich, wird ‚CPU_STATUS_MODULES_VERIFIED' im FSP242 gesetzt.

Wird die MFU ohne Module betrieben lässt sich das Flag ‚CPU_STATUS_MODULES_VERIFIED' setzen, indem ein Schreibkommando an FSP243 gesendet wird, bei dem alle Parameter (Head, ModulClass, ModulSubClass) auf null (0x30) stehen.

STX	PID	GW	MA	FSP	Head	ModuleClass	ModuleSubClass	PP	ETX
0x02	0x57 0x52	0x30	0x30	0x46 0x33	0x30 0x30	0x30 0x30	0x30 0x30	0x30 0x30	0x03

Name	**FSP244_SW_ChangeUSIBitrate_ChangeUSIMode**
Adresse	**0xF4/244$_D$/0x4634$_{ASCII}$**
Tiefe	2 Byte / 16 Bit
I/O	schreiben
Reset	0x(siehe Beschreibung)$_H$

Über diesen FSP kann seitens des PC die Bitrate sowie der USI Modus einzelner USI konfiguriert werden.

Die Anforderung ist dabei identisch mit einer Anforderung an das Standard FSP012 zur Konfiguration der USI. Schreibt der PC direkt an einen Modul FSP012, erfolgt keine Umschaltung der USI innerhalb der MFU, da die MFU lediglich als Gateway funktioniert und die Daten zwischen PC und Modulen nicht analysiert.

Daher erfolgt die Bitratenumschaltung der USI über diesen Software FSP.

Mit diesem FSP wird die Bitraten/USI Mode Umschaltung in der MFU vorgenommen und von dieser mit dem/mit den angeschlossenen Modul(en) verhandelt, damit auch diese die gewünschte Bitrate/den gewünschten USI Mode entsprechend der Anforderung einstellen.

WICHITG: Ist eine USI in HighSpeed, kann deren Bitrate NICHT direkt in HighSpeed geändert werden. Es muss zuvor HighSpeed an dieser USI dekaktiviert werden. Bei der HighSpeed-Deaktivierungs-Anforderung kann aber eine Wunschbitrate für die USI-Standardkommunikation mitgeteilt werden. (Bsp.: mit `Data Data` = 0x30 0x38 wird HighSpeed abgeschaltet und das USI wechselt sofort auf 10MBit). Im Anschluss wird ggf. in einer weiteren Anforderung an diesen FSP die neue Bitrate/der neue USI Mode eingestellt.

Sollen an einem USI mit mehreren Modulen alle Module zeitgleich auf eine neue Bitrate geschaltet werden, wird für den Parameter `Mod` = 0 (0x30) gewählt.

Sollen alle Module an allen USIs zeitgleich auf eine neue Bitrate geschaltet werden, wird für den Parameter `USI` = 0 (0x30) gewählt.

```
STX PID PID GW MA FSP FSP USI Mod Data Data PP PP ETX
```

`STX`	- StartOfText	(0x02)
`PID PID`	- Schreibanforderung	(0x5752)
`GW`	- Gateway Adresse	(0x30 – da Gateway die MFU ist)
`MA`	- Modul Adresse	(0x30 – da Modul die MFU ist)
`FSP FSP`	- FSP Nummer	(0x46, 0x34)
`USI`	- USI Nummer an der Bitrate/Modus geändert werden soll (0x30, 0x31...0x42)	
`Mod`	- Modulnummer an dem Bitrate/Modus geändert werden soll (0x30, 0x31...0x38)	
`Data Data`	- Bitrate/Modus (siehe Standard FSP012)	
`PP PP`	- Prüfsumme	
`ETX`	- EndOfText	(0x03)

Mit `Data` für Standrad USI:
07 `[0x30 0x37]` = 115,2kBit
06 `[0x30 0x36]` = 1Mbit
05 `[0x30 0x35]` = 2Mbit
04 `[0x30 0x34]` = 5Mbit
03 `[0x30 0x33]` = 10Mbit
02 `[0x30 0x32]` = 16,6Mbit
01 `[0x30 0x31]` = 20Mbit
00 `[0x30 0x30]` = 25MBit

Soll USI HighSpeed benutzt werden, muss zusätzlich das MSB gesetzt werden:
87 `[0x38 0x37]` = 115,2kBit HS
86 `[0x38 0x37]` = 1MBit HS
(...)

Example: switch USI 1 Module 1 to 115,2kBit, no HighSpeed

STX	W	R	0	0	F	4	1	1	0	7	0	7	ETX
02	57	52	30	30	46	34	31	31	30	37	30	37	03

Example: switch on HighSpeed for USI 1 Module 1 again and stay at 115.2kBit

```
STX W  R  0  0  F  4  1  1  8  7  0  F  ETX
02  57 52 30 30 46 34 31 31 38 37 30 46 03
```

Example: switch on HighSpeed for USI 1 Module 1 again and switch directly to 2Mbit

```
STX W  R  0  0  F  4  1  1  8  5  0  D <3>
02  57 52 30 30 46 34 31 31 38 35 30 44 03
```

Name	**FSP245_SW_intScopeDataStream**
Adresse	**0xF5/245$_D$/0x4635$_{ASCII}$**
Tiefe	12008 Byte / 96064 Bit (bis MFU FW 7.4.x) 6008 Byte / 48064 Bit (ab MFU FW 7.5.0)
I/O	lesen
Reset	0x(siehe Beschreibung)$_H$

Dieser FSP sendet Scope Kanaldaten des internen Scopes mit

8 Kanälen (bis MFU FW 7.4.x) mit je 500 Werten und 12 Bit Tiefe
→ 6000 Byte$_H$, bzw. 12000 Bytes$_A$ + einbettende STX ... PP PP ETX usw.

4 Kanälen (ab MFU FW 7.5.0) mit 500 Werten und 12 Bit Tiefe
→ 3000 Byte$_H$, bzw. 6000 Bytes$_A$ + einbettende STX ... PP PP ETX usw.

Bei der Ausgabe dieser Daten wird der Triggerpunkt und -offset berücksichtigt.

D.h. die Daten werden nicht beginnend vom RAM Speicher 0, sondern beginnend vom Triggeroffset ausgegeben. An der höchsten RAM-Stelle (500) wird mit der RAM Stelle 0 fortgefahren und die Ausgabe endet bei Triggeroffset – 1.

Sollen die Daten beginnend vom RAM Speicher 0 und nicht vom Triggeroffset ausgegeben werden bitte den FSP231_SW_intScopeDataStreamReadOut benutzen.

FSP245 kann nur gelesen werden. Schreibanforderungen bekommen eine NACK Antwort mit Fehlercode.

Bis MFU FW 7.4.x gilt folgendes:

Ausgabestring nach der Leseanforderung:

```
STX GW MA -FSP-  -CH8(12Bit) CH7(12Bit) CH7(12Bit) CH5(12Bit)
CH4(12Bit) CH3(12Bit) CH2(12Bit) CH1(12Bit)- -PP-   ETX

02  30 30 45 37   dd dd dd   dd dd dd   dd dd dd   dd dd dd
dd dd dd   dd dd dd   dd dd dd   dd dd dd    pp pp  03
```

CHx - Daten des Kanals

Insgesamt werden 500 x **8** Kanäle übertragen. D.h. der gelb markierte Bereich wird also insgesamt 500-mal übertragen. Beginnend mit dem Wert 1 für Kanal 8 bis 1, Wert 2 für Kanal 8 bis 1 usw. beginnend beim Triggerpunkt. Es bilden also immer 8 x 12Bit = 96Bit → 12 Bytes einen Datensatz. Die Prüfsumme PP folgt nach Wert 500 für Kanal 8 bis 1 über die gesamten Daten.

Ab MFU FW 7.5.0 gilt folgendes:

Ausgabestring nach der Leseanforderung:

```
STX GW MA -FSP-  -CH4(12Bit) CH3(12Bit) CH2(12Bit) CH1(12Bit)- -PP-
ETX

02  30 30 45 37   dd dd dd   dd dd dd   dd dd dd   dd dd dd    pp pp
03
```

CHx - Daten des Kanals

Insgesamt werden 500 x **4** Kanäle übertragen. D.h. der gelb markierte Bereich wird also insgesamt 500-mal übertragen. Beginnend mit dem Wert 1 für Kanal 4 bis 1, Wert 2 für Kanal 4 bis 1 usw. beginnend beim Triggerpunkt. Es bilden also immer 4 x 12Bit = 48Bit → 6 Bytes einen Datensatz. Die Prüfsumme PP folgt nach Wert 500 für Kanal 4 bis 1 über die gesamten Daten.

Name	**FSP246_SW_Recorded_Supplies**
Adresse	**0xF6/246$_D$/0x4636$_{ASCII}$**
Tiefe	10800 Byte
I/O	lesen
Reset	0x(siehe Beschreibung)$_H$

Dieser FSP liefert die in der MFU überwachten Betriebsspannungen der letzten 24 Stunden. Die Spannungen werden einmal je Minute gespeichert. Die Sortierung ist dabei wie folgt: 1V2, 2V5, 3V3, 5V0, p12V0, n12V0. Der Speicher der Betriebsspannungen ist flüchtig, d.h. nach dem Aus- und Wiedereinschalten sind alle Werte zunächst 0.

Je Spannung wird durch 12 Bit Daten + Vorzeichen repräsentiert. Bei der Ausgabe werden je Spannung 16 Bit gesendet. Das MSB ist dabei je Spannung das Vorzeichen, danach folgen die 12 Bit Spannung. Die restlichen 3 Bit je Spannungswert bleiben immer 0.

FSP246 kann nur gelesen werden. Schreibanforderungen bekommen eine NACK Antwort mit Fehlercode.

Ausgabestring nach der Leseanforderung:

```
STX GW MA -FSP-   1V2(16Bit)   2V5(16Bit)   3V3(16Bit)   5V0(16Bit)
12V0(16Bit)  -12V0(16Bit) (…) PP PP ETX

02  30 30 46 36   dd dd dd dd  dd dd dd dd  dd dd dd dd  dd dd dd dd
dd dd dd dd  dd dd dd dd              03
```

dd dd dd dd$_H$ entspricht dabei vsss ssss ssss sxxx$_B$ mit

v = Vorzeichen
s = Spannungswert
x = don’t care, immer ‚0‘

Insgesamt werden 24x60=1440 Werte je Spannung übertragen. D.h. der gelb markierte Bereich wird insgesamt 1440-mal übertragen. Beginnend mit dem Wert für 1,2 Volt. Die Prüfsumme PP folgt nach dem 1440-sten Wert für -15,0 Volt über die gesamten Daten.

1440 Spannungen * 6 Einträge mit je 16 Bit 17280 Byte Hex, bzw. 34560 Byte ASCII an Daten.

Die Bitwertigkeiten sind wie folgt:

bei 1V2, 2V5, 3V3 und 5V0 werden die gemessenen Werte direkt mit dem LSB Wert (SUP_LSB_Size = 0.002441) multipliziert und ergeben so die Spannung. Bei -12V0 und 12V0 erfolgt ebenfalls zuerst die LSB Wert Multiplikation und anschließend die Multiplikation mit dem Faktor 11, der dem vorgeschalteten Spannungsteiler entspricht. Die tatsächlich gemessenen Spannungen liegen nämlich bei 1/11 der Originalspannung.

Name	**FSP247_SW_Recorded_Temperatures**
Adresse	**0xF7/247$_D$/0x4637$_{ASCII}$**
Tiefe	8640 Byte
I/O	lesen
Reset	0x(siehe Beschreibung)$_H$

Dieser FSP liefert die in der MFU überwachten Modultemperaturen der letzten 24 Stunden. Die Temperaturen werden einmal je Minute gespeichert. Die Sortierung ist dabei wie folgt: FPGA, Längsregler, Modulmitte. Der Speicher der Temperaturenaufzeichnung ist flüchtig, d.h. nach dem Aus- und Wiedereinschalten sind alle Werte zunächst 0.

Je Temperatur steht ein 8 Bit, also 1 Byte Speicher zur Verfügung. Alle 3 Temperaturen belegen also pro Eintrag 3 Byte Hex, bzw. 6 Byte ASCII.

FSP247 kann nur gelesen werden. Schreibanforderungen bekommen eine NACK Antwort mit Fehlercode.

Ausgabestring nach der Leseanforderung:

```
STX GW MA -FSP-  T1(8Bit) T2(8Bit) T3(8Bit) (…) PP PP ETX
02  30 30 46 37   dd dd    dd dd    dd dd             03
```

Insgesamt werden 24x60=1440 Werte je Temperatur übertragen. D.h. der gelb markierte Bereich wird insgesamt 1440-mal übertragen. Beginnend mit dem Wert T1 für das FPGA. Die Prüfsumme PP folgt nach dem 1440-sten Wert für T3 über die gesamten Daten.

1440 Übertragungen mit je 3 Byte Hex sind 4320 Byte Hex, bzw. 8640 Byte ASCII an Daten.

Die Bitwertigkeiten sind wie folgt:

Aktuelle Temperatur	Gemessene Temperatur	Binär Hexadzimal
+130.00°C	+127°C	0111 1111
+127.00°C	+127°C	0111 1111
+126.50°C	+126°C	0111 1110
+25.25°C	+25°C	0001 1001
+0.50°C	0°C	0000 0000
+0.25°C	0°C	0000 0000
0.00°C	0°C	0000 0000
-0.25°C	-1°C	1111 1111
-0.50°C	-1°C	1111 1111
-0.75°C	-1°C	1111 1111
-1.00°C	-1°C	1111 1111
-25.00°C	-25°C	1110 0111
-25.25°C	-26°C	1110 0110
-54.75°C	-55°C	1100 1001
-55.00°C	-55°C	1100 1001
-65.00°C	-65°C	1011 1111

Name	**FSP248_SW_ReadExtRAMData**
Adresse	**$0xF8_H/248_D/0x4638_{ASCII}$**
Tiefe	dynamisch
I/O	lesen / schreiben
Reset	$0x(\text{siehe Beschreibung})_H$

Gibt eine bestimmte, frei wählbare Anzahl von RAM Einträgen eines ExtRAMExtention Moduls zurück.

Dazu müssen zunächst Daten gesammelt worden sein.

Die Daten können nur 1x ausgelesen werden. Danach muss ein neuer Speicherzyklus durchlaufen werden. Vor dem Starten eines neuen Speicherzyklus müssen ALLE Daten ausgelesen worden sein.

Zum Auslesen der Daten muss der FSP ggf. zunächst beschrieben werden mit:

```
STX PID PID GW MA FSP FSP [Quantity][Quantity][Quantity][Quantity] PP
PP ETX
```

Mit:

`PID PID` WR
`[Quantity]` Anzahl von Werten (1..65536)

Der Resetwert von `[Quantity]` ist 0x0000, dadurch werden alle im ExtRAMmodul befindlichen Werte zurück geliefert.

Sofern `[Quantity]` != 0x0000 gesetzt wurde, lässt `[Quantity]` sich nach jedem ausgelesenen Datenblock beliebig ändern und auch wieder auf 0x0000 setzen um die restlichen Daten in einem Aufruf zu lesen.

Wird der FSP anschließend gelesen, werden mit jeder Leseaufforderung [Quantity] Datensätze des ExtRAMExtention Moduls gesendet. Ist `[Quantity]` = 0 werden alle Daten ohne Unterbrechung gesendet:

```
STX GW MA FSP FSP
[Byte3 Byte2 Byte1 Byte0] ← Datensatz n
[Byte3 Byte2 Byte1 Byte0] ← Datensatz n-1
[..]
[Byte3 Byte2 Byte1 Byte0] ← Datensatz n-n
PP PP ETX
```

Name	**FSP249_Local_Setvalue_Scaling_Factor**
Adresse	**$0xF9_H/249_D/0x4639_{ASCII}$**
Tiefe	2 Byte /16 Bit
I/O	lesen / schreiben
Reset	$0x0002_H$

Definiert das Teilerverhältnis der Lokalen, manuellen Sollwertvorgabe.

Die Standardschrittweite bei der Sollwertvorgabe beträgt 1/50 des Nennwerts. Bei Betätigen der RESET-Taste im Local-Modus wird diese Schrittweite weiter heruntergeteilt.

:2, :4(Default), :8, :16, :32, :64, :128, ... (usw.)

Weil bei der Berechnung des Teilerfaktors dieser einfach geschoben wird, muss 'setvalue_stepwidth' die Anzahl der Bit-Schübe beinhalten.

Der Standardwert ist hier 2, dadurch wird 1/50 zu 1/200.

```
STX PID PID GW MA FSP FSP [Shift][Shift] PP PP ETX
```

Mit:

`[Shift]` Anzahl von Bitschüben

Name	**FSP250_NIOS_SW_ Version**
Adresse	**$0xFA_H/250_D/0x4641_{ASCII}$**
Tiefe	dynamisch
I/O	lesen
Reset	$0x(\text{siehe Beschreibung})_H$

Dieser SW FSP beinhaltet die NiosII SW Version für Remote Updates.

Er kann nur gelesen werden.

Nachfolgendes Beispiel zeigt den Nios Software-Stand 007.00004 an.

▫	0	0	F	A	0	0	7	.	0	0	0	0	4	2	D	▫
02	30	30	46	41	30	30	37	2E	30	30	30	30	34	32	44	03

Name	**FSP251_compressed_PCA_configuration_file**
Adresse	**$0xFB_H/252_D/0x4642_{ASCII}$**
Tiefe	dynamisch
I/O	lesen / schreiben
Reset	$0x(\text{siehe Beschreibung})_H$

Dieser SW FSP beinhaltet komprimierte PCA Konfigurationsdatei.

Um eine Datei zu speichern muss zunächst ein Schreibkommando auf den FSP erfolgen, in dem die Größe der anschließend zu übertragenden *.xpc7-Konfigurationsdatei mitgeteilt wird.

```
STX PID PID GW MA FSP FSP -[Length (8 Byte)]-       PP PP ETX
STX  W   R   0  0   F   B  Dateigroesse in Byte     PP PP ETX
```

mit

`[Length]` ist 8 Byte ASCII lang, sollen z.B. $8.977_D = 2311_H$ Byte übertragen werden ist
`Length = 0x30 0x30 0x30 0x30 0x32 0x33 0x31 0x31`

Im Falle von ACK schaltet die MFU automatisch auf einen Empfangsmodus um und erwartet nachfolgend die zuvor angekündigte Anzahl von Bytes. D.h. es werden nun solange Daten der komprimierten *.xpc7 Datei an die MFU gesendet, bis alle Daten im seriellen Flash gespeichert sind.

Die Daten werden dabei NICHT als ASCII-Zeichen, sondern wie sie vorliegen, als reine Rohdaten gesendet.

Der Empfang der Daten wird abschließend mit einem weiteren ACK, bzw. NACK quittiert.

Soll die Datei aus dem seriellen Flash gelesen werden, muss lediglich das FSP gelesen werden.

Die dabei empfangenen Daten werden in einen USI-Header, eine Prüfsumme und ein ETX eingebettet.

Die Daten an sich sind aber wie beim Senden auch, die reinen Rohdaten und NICHT ASCII kodiert.

Name	**FSP252_UpdateMFU_CFI_SoftwareViaRemote**
Adresse	**$0xFC_H/252_D/0x4643_{ASCII}$**
Tiefe	Dynamisch
I/O	lesen / schreiben
Reset	$0x(\text{siehe Beschreibung})_H$

Das CFI beinhaltet 2 Software-Versionen, die 'Factory-SW' und die 'Application-SW' und zusätzlich den Boot Copier

--------- SCHREIBEN ---------

Via FSP252 können beide Softwaren und der Boot Copier ersetzt werden.

Per Default wird aber i.d.R. nur die Application-SW ersetzt.

Dazu muss das FSP252 zunächst ein Init-Kommando erhalten:

```
STX PID PID GW MA FSP FSP -Data-                    PP PP ETX
```

(Update Boot Copier)

```
STX  W  R   0  0  F   C   UBC+Dateigroesse in Byte  PP PP ETX
```

(Update Application SW)

```
STX  W  R   0  0  F   C   UAS+Dateigroesse in Byte  PP PP ETX
```

(Update Factory SW)

```
STX  W  R   0  0  F   C   UFS+Dateigroesse in Byte  PP PP ETX
```

Die [`Dateigroesse in Byte`] muss immer 8 Zeichen lang sein. Die Angabe der Dateigröße erfolgt in Hexadezimal. Führende Stellen sind 0.

Nachfolgend soll als Beispiel das Application-Image ersetzt werden. Die dazu nötige *.s19-Datei hat eine Größe von 959232_D = $EA300_H$ Bytes.

◘	W	R	0	0	F	C	U	A	S	0	0	0	E	A	3	0	0	4	0	◘
02	57	52	30	30	46	43	55	41	53	30	30	30	45	41	33	30	30	34	30	03

Die Prüfsumme ist hierbei über den gesamten `-Data-` Bereich (`UAS000EA300`) zu bilden.

Darauf antwortet FSP252 mit einem ACK, bzw. im Fehlerfall einem NACK.

Im Falle von ACK schaltet die MFU automatisch auf einen Empfangsmodus um und erwartet nachfolgend die zuvor angekündigte Anzahl von Bytes. D.h. es werden nun solange Daten der *.s19-Datei an die MFU gesendet, bis alle Daten im CFI gespeichert sind. Die Daten werden dabei NICHT als ASCII-Zeichen, sondern wie sie vorliegen, als reine Rohdaten gesendet.

Der Empf. der *.s19-Datei wird mit ACK, bzw. NACK quittiert.

--------- LESEN ---------

Sollen Daten aus dem CFI gelesen werden, muss dies zuvor per Schreibzugriff definiert welche Daten genau wie viele Bytes davon gelesen werden sollen.

(Read Boot Copier)

```
STX  W  R   0  0  F   C   RBC+Dateigroesse in Byte  PP PP ETX
```

(Read Application SW)

```
STX  W  R   0  0  F   C   RAS+Dateigroesse in Byte  PP PP ETX
```

(Read Factory SW)

```
STX  W  R   0  0  F   C   RFS+Dateigroesse in Byte  PP PP ETX
```

Anschließend erfolgt eine Leseaufforderung. Diese liefert dann die gewünschte Anzahl an Bytes.

Bsp.: 390000_D = $5F370_H$ Byte der Application SW lesen:

◻	W	R	0	0	F	C	R	A	S	0	0	0	5	F	3	7	0	3	7	◻
02	57	52	30	30	46	43	52	41	53	30	30	30	35	46	33	37	30	33	37	03

Die Prüfsumme ist hierbei über den gesamten `-Data-` Bereich (`RAS0005F370`) zu bilden.

Anschließend erfolgt ein Lesekommando. Die dabei empfangenen Daten werden in einen USI-Header, eine Prüfsumme und ein ETX eingebettet.

Die Daten an sich sind aber wie beim Senden auch, die reinen Rohdaten und NICHT ASCII kodiert.

--------- LÖSCHEN ---------

Das CFI lässt sich auch nur löschen (vollständig):

```
STX  W   R   0  0   F   C  EBC+Dateigroesse in Byte   PP PP ETX
```

Die Daten für [`Dateigroesse in Byte`] sind dabei egal.

Name	**FSP253_UpdateMFU_EPCS_FirmwareViaRemote**
Adresse	**$0xFD_H/253_D/0x4644_{ASCII}$**
Tiefe	dynamisch
I/O	lesen / schreiben
Reset	$0x(\text{siehe Beschreibung})_H$

Das EPCS beinhaltet 2 Firmware-Versionen, das 'Factory-Image' und das 'Application-Image'.

Via FSP253 können beide Images ersetzt werden.

Per default wird aber i.d.R. nur das Application-Image ersetzt.

Dazu muss das FSP253 zunächst ein Init-Kommando erhalten:

```
STX PID PID GW MA FSP FSP -Data-                    PP PP ETX
```

(Update Application Image)

```
STX W   R   0   0   F   D   UAI+Dateigroesse in Byte  PP PP ETX
```

Beim Factory Image muss das Init-Kommando entsprechend so lauten:

(Update Factory Image)

```
STX  W  R   0   0   F   D   UFI+Dateigroesse in Byte  PP PP ETX
```

Die `Dateigroesse in Byte` muss immer 8 Zeichen lang sein. Die Angabe der Dateigröße erfolgt in Hexadezimal. Führende Stellen sind 0.

Nachfolgend soll als Beispiel das Application-Image ersetzt werden. Die dazu nötige *.rbf-Datei hat eine Größe von 988394_D = $F14EA_H$ Bytes.

¤	W	R	0	0	F	D	U	A	I	0	0	0	F	1	4	E	A	2	A	¤
02	57	52	30	30	46	44	55	41	49	30	30	30	46	31	34	45	41	32	41	03

Die Prüfsumme ist hierbei über den gesamten `-Data-` Bereich (`UAI000F14EA`) zu bilden.

Daraufhin antwortet FSP253 mit einem ACK, bzw. im Fehlerfall einem NACK.

Im Falle von ACK schaltet die MFU automatisch auf einen Empfangsmodus um und erwartet nachfolgend die zuvor angekündigte Anzahl von Bytes. D.h. es werden nun solange Daten der *rbf-Datei an die MFU gesendet, bis alle Daten im EPCS gespeichert sind. Die Daten werden dabei NICHT als ASCII-Zeichen, sondern wie sie vorliegen, als reine Rohdaten gesendet

Der Empfang der *.rbf Daten wird mit ACK, bzw. NACK quittiert.

Sollen Daten aus dem EPCS gelesen werden, muss zuvor per Schreibzugriff definiert werden ob das Application oder das Factory Image gelesen werden soll und wie viele Bytes davon gelesen werden sollen.

(Read Appilcation Image)

```
STX  W  R   0   0   F   D   RAI+Dateigroesse in Byte  PP PP ETX
```

(Read Factory Image)

```
STX  W  R   0   0   F   D   RFI+Dateigroesse in Byte  PP PP ETX
```

Bsp.: 988394_D = $F14EA_H$ Byte der Application FW lesen

¤	W	R	0	0	F	D	R	A	I	0	0	0	F	1	4	E	A	2	D	¤
02	57	52	30	30	46	44	52	41	49	30	30	30	46	31	34	45	41	32	44	03

Die Prüfsumme ist hierbei über den gesamten `-Data-` Bereich (`RAI000F14EA`) zu bilden.

Anschließend erfolgt ein Lesekommando. Die dabei empfangenen Daten werden in einen USI-Header, eine Prüfsumme und ein ETX eingebettet.

Die Daten an sich sind aber wie beim Senden auch, die reinen Rohdaten und NICHT ASCII kodiert.

Name	**FSP254_Parameter_Information_String**
Adresse	**$0xFE_H/254_D/0x4645_{ASCII}$**
Tiefe	Dynamisch, max. 256 Zeichen
I/O	lesen / schreiben
Reset	$0x(\text{siehe Beschreibung})_H$

Parameterinformationen in ASCII Klartext.

```
STX PID PID GW MA FSP FSP [Text] PP PP ETX
```

Mit:

`[Text]` max. 256 Zeichen Textinformation über die in der MFU hinterlegten Parameter

Name	**FSP255_SW_Flash_VNC2**
Adresse	**$0xFF_H/255_D/0x4646_{ASCII}$**
Tiefe	65536 Byte / 524288 Bit
I/O	schreiben
Reset	$0x(\text{siehe Beschreibung})_H$

Der interne FSP255 ist ein CPU Software FSP und wird zum programmieren des Vinculum VNC2 USB Master uController in der MFU benutzt.

FSP255 kann nur geschrieben werden. Leseanforderungen bekommen eine NACK Antwort mit Fehlercode.

Bis MFU Software 002.00002

Zum programmieren des Vinculum VNC2 wird eine reguläre USI Schreibanforderung an dieses FSP gesendet. Die nach dem Header folgenden Daten werden von der MFU bei diesem FSP ohnehin verworfen, d.h. es brauchen weder Daten noch eine Prüfsumme vor dem ETX gesendet werden.

```
STX PID PID GW MA FSP FSP ETX
```

STX	-	StartOfText	(0x02)
PID PID	-	Schreibanforderung	(0x5752)
GW	-	Gateway Adresse	(0x30 – da Gateway die MFU ist)
MA	-	Modul Adresse	(0x30 – da Modul die MFU ist)
FSP FSP	-	FSP Nummer	(0x46, 0x46)
[Daten..Prüfsumme]	- nicht notwendig, kann entfallen		
ETX	-	EndOfText	(0x03)

Wichtig: Sofern die Programmieranforderung von der MFU verstanden wurde, wird KEIN ACK gesendet. Wird diese hingegen nicht verstanden erfolgt das senden eines NACK.

Im Erfolgsfall wechselt die Anzeige des LCD und zeigt „Waiting for flash ROM data“.

Nun wird das ROM File gesendet. Wichtig ist hierbei, dass die Übertragung in 'Hex' erfolgt.

Erst nach dem erfolgreichen senden und programmieren des ROM Files wird von der MFU ein ACK, im Fehlerfall ein weiteres NCK gesendet.

Ab MFU Software 002.00003

Zum programmieren des Vinculum VNC2 wird eine reguläre USI Schreibanforderung an dieses FSP gesendet. Nach dem Header wird der MFU die Länge der folgenden Programmierdatei mitgeteilt. Dies ist notwendig, weil die MFU auf dem TFT eine Fortschrittsanzeige generiert und der VNC2 nur vollständig geschriebene Pages akzeptiert. Ist die Programmierdatei zu Ende aber die letzte Page des VNC2 noch nicht vollständig programmiert schlägt die gesamte Programmierung fehl. Die MFU generiert aus der Information der Programmierdatenlänge und den evtl. abschließend fehlenden Daten eine vollständige letzte Page für den VNC2 automatisch. Dazu benötigt sie aber die Länge der Programmierdaten. Bis V002.00002 war die Länge per #define festgelegt, was das Programmieren des VNC2 mit neueren Softwareständen deren Länge davon abweicht nicht möglich machte.

```
STX PID PID GW MA FSP FSP Length Lenght Length Length PP PP ETX
```

STX	-	StartOfText	(0x02)
PID PID	-	Schreibanforderung	(0x5752)
GW	-	Gateway Adresse	(0x30 – da Gateway die MFU ist)
MA	-	Modul Adresse	(0x30 – da Modul die MFU ist)
FSP FSP	-	FSP Nummer	(0x46, 0x46)
Length	-	Länge der nachfolgenden Programmierdatei	(0x30, 0x30, 0x30, 0x33, 0x32, 0x30, 0x39, 0x30)
PP PP	-	Prüfsumme über Length	
ETX	-	EndOfText	(0x03)

Sofern die Programmieranforderung von der MFU verstanden wurde, wird EIN ACK gesendet die Anzeige des TFT zeigt „Waiting for flash ROM data“.

Wird diese hingegen nicht verstanden erfolgt das senden eines NACK.

Nun wird das ROM File gesendet. Wichtig ist hierbei, dass die Übertragung in 'Hex' erfolgt.

Nach dem erfolgreichen senden und programmieren des ROM Files wird von der MFU ein weiteres ACK, im Fehlerfall ein NACK gesendet.