LabScript 1.0

Script-Implementation für FPGA ab 1.2

INI-Files sind auf der SD-Karte gespeicherte Textdateien, die beliebige c't-Lab-Befehle enthalten dürfen. Diese werden abgearbeitet, als wären sie vom PC über die OptoBus-Schnittstelle an das Modul gelangt. Zu erstellen mit Wordpad o.dgl.

Für Scripte, die als INI-Datei auf SD-Karte gespeichert sind

Achtung: Script-Befehle sind nicht rekursiv, deshalb sind CFG, LST und FNM in einem Script nicht erlaubt (es sei denn, sie sprechen ein anderes Modul an). Branches "vorwärts" verlangsamen den ersten Durchlauf, und zwar um so mehr, je weiter "unten" sie im Script stehen.

Erweiterung auf Integer-SubCh 0..32767
Für die Script-Verarbeitung stehen 10 Fließkomma-Register zur Verfügung, von denen das erste (0) als Fließkomma-Akkumulator dient.
Load- und Store-Befehle funktionieren mit jedem Register (0..9 einschl. Akkus)

Cmd	Argument	SubCh <opcode>	<format>	Syntax/Beispiele	Beispiel-Antwort	Erläuterung
//	-	-	-	// Kommentarzeile		Ignorierte Kommentarzeile
REG		300..309	Float	REG <n>=<wert>, REG <n>?		Load Register 0..9 immediately mit Wert, auch Abfrage
ACC		300		ACC=<wert>, ACC?, 300?		Load Akku A (Register 0) immediately mit Wert, auch Abfrage
MOV		310..319		MOV <zielregister>=<quellregister>		Move Registerinhalt <zielregister> <= <quellregister>
DEC		320..329		DEC <register>		320 Decrement Register (0..9), Ergebnis für Branches merken
INC		330..339		INC <register>		330 Increment Register (0..9), Ergebnis für Branches merken
CPZ		340..349		CPZ <register>		340 Compare Register (0..9) mit "0", Ergebnis für Branches merken
XCH		350..359		XCH <register>=<register>, XCH = 4, XCH 2=6		350 Exchange 0..9 <=> 0..9, ohne Argument: Akku A mit Register (0..9)
MUL		600..609		MUL <register>		600 Multiplikation Akku A = Akku A * Register (0..9)
DIV		610..619		DIV <register>		610 Division Akku A = Akku A / Register (0..9)
ADD		620..629		ADD <register>		620 Addition Akku A = Akku A + Register (0..9)
SUB		630..639		SUB <register>		630 Subtraktion Akku A = Akku A - Register (0..9)
SQR		640..649		SQR <register>		640 Quadratwurzel Register (0..9) <= sqr(Register (0..9))
SQU		650..659		SQU <register>		650 Quadrat Register (0..9) <= Register (0..9) * Register (0..9)
NEG		660..669		NEG <register>		660 Negiere Register (*-1)
GET		400..409		GET <zielregister>?		Warte auf Ergebnis, speichere in Registerinhalt 0..9. Funktioniert wegen nötigem OptoBus-"Kreisverkehr" nur unzureichend oder gar nicht. (#X:XXX gleich Kanal-Match)
PUT		500..509		PUT <register>		Sende Wert in Register auf MCM/SCM
FWR		900..999		FWR <dateinummer> = <register>, FWR 75=2!, FWR = 0!	angehängt an Datei DATA75.XLS, DATA0.XLS	FileWrite, Register 0..9 in Datei DATA0..DATA99.XLS schreiben (reine Textdatei, kann mit Excel geöffnet werden)
LBL		1000..1099		LBL <labelnr>		Label 0..99 setzen (hier begrenzt auf 0..31)
GTO		1100..1199		GTO <labelnr>		Goto <labelnr>
BRA		1100..1199		BRA <labelnr>		Branch always to <labelnr>, wie GTO
BRG		1200..1299		BRG <labelnr>		Branch if greater 0 to Label <labelnr>, wenn Akku A größer als B
BGE		1300..1399		BGE <labelnr>		Branch if greater or equal 0 to Label <labelnr>, wenn Akku A größer oder gleich B
BEQ		1400..1499		BEQ <labelnr>		Branch if equal 0 to Label <labelnr>, wenn Akku A gleich B
BLE		1500..1599		BLE <labelnr>		Branch if lower or equal 0 to Label <labelnr>, wenn Akku A kleiner oder gleich B
BRL		1600..1699		BRL <labelnr>		Branch if lower 0 to Label <labelnr>, wenn Akku A kleiner als B
INP		2000..2255		INP <SubCh>?, INP 248? INP 0?		Akku (Reg. 0) Input, mit Ergebnis von moduleigenem SubCh 0..255 füllen
OUT		2000..2255		OUT <SubCh>=<register>, OUT 128=3		Register Output, Ausgabe <register> auf moduleigenen SubCh 0..255
WTH		290		WTH		Wait Tick Hour
WTM		291		WTM		Wait Tick Minute
WTS		292		WTS		Wait Tick Second, warte auf nächsten Sekundenwechsel
DLY		299		DLY=<ms_zeit>, 299=1500!		Delay/Pause, Ausführung wird <ms_zeit> Millisekunden angehalten. Nur in Scripts sinnvoll, da OptoBus während dieser Zeit blockiert ist. Liefert beim Lesen letzten Pausenwert.
MCH		270		MCM=<Moduladresse>		Main-Channel für GET/PUT, Script-gesendete/empfangene Daten, Moduladresse
SCH		271		SCM=<SubCh>		SubChannel für GET/PUT, Script-gesendete/empfangene Daten

Display-Befehle QVGA-Treiber FPGA (nur bis Firmware #2.33) — **obsolet ab FW 2.4**

DPV	700	DPV=1.2345		Display Value Immediate, Wert DPV=1.2345 @ DPX, DPY
DPR	710..719			Display Inhalt Register 0..9 DPR=<register> @ DPX, DPY
DPS	720	DPS=HALLO_FPGA		Display String DPS=HALLO_FPGA @ DPX, DPY
DPC	730			Display RTC, DPC=0 Zeit, DPC=1 Datum @ DPX, DPY
DRW	740..742			740 DrawRect 741 FillRect 742 Drawline mit XY-Param @ DPX, DPY
DRI	750	DRI=1		Draw Instrument, Bargraph usw.
DBG	770	DBG=0, DBG=demo.bmp		Display-Background, DBG=0 keiner, DBG=<alphastr>=BMP-Name
DPX	780	DPX=80		Display-Koordinaten, EndX, vorheriger Wert wird StartX für Linien/Rechtecke
DPY	781	DPY=200		Display-Koordinaten, EndY, vorheriger Wert wird StartY
DFX	782	DFX=1		Font-Skalierung X-Richtung (vertikal), 1=klein, 2=mittel, 3=groß
DFY	783			Font-Skalierung Y (horizontal)
DCO	790	DCO=42		Block-Farbe 6-Bit-Darstellung für nächste Anzeige-Operation, Bitfolge RRGGBB

Wünsche und Anregungen bitte an cm@ctmagazin.de