c't-Lab UNI-C Syntax Stand 15.11.2011

c't-Lab liefert kein Echo, bei einem Terminal-Programm deshalb ggf. lokales Echo einstellen. Nur ein Befehl pro Zeile. Befehle werden erst nach dem Empfang von CR oder CR/LF verarbeitet.
Schnittstellen-Parameter: 38400 Bd, 8n1. Backspace (#8) löscht letztes Zeichen aus dem Befehlszeilenpuffer, andere Control-Zeichen werden ignoriert.
Es besteht die Option, dem Befehl/der Abfrage eine XOR8-Prüfsumme anzuhängen, in der Form $<HEXbyte> mit der XOR-Summe über den gesamten Befehlsstring
also z.B. 8:VAL 20=1.234!$45
Das fordert das jeweilige Modul auf, die Prüfsumme gegen die errechnete zu checken und im Fehlerfall mit dem Fehlercode 7 abzubrechen.
Lässt man bei Ausgabe-Befehlen das "!" weg, erfolgt keine Ausgabe des "#8:255=0 [OK]" Prompts (vermindert Datenmenge bei kritischen Anwendungen)

allgemein

Cmd	Argument	SubCh	Wertebereich	Beispiel-Befehle	Beispiel-Antwort	Erläuterung
IDN	--	254		IDN?, *:IDN?	#8:255=1.01b [UNIC by c't]	Identifizierung, '*' als Moduladresse gilt für alle Slave-Channels
VAL	0..255	0..255	div.	8:VAL 20=5.0!, VAL 10?, 8:10?, 20=5.0!	#8:20=5.0000, #8:2=0.0	Allg. Form, Messwert-Spannung in Volt, VAL kann auch weggelassen werden
STR**	--	255		*:STR?, 255?	#8:255=0 [OK]	Status-Request, Bit 7 (+128)=Busy, 6 (+64)=UserSRQ, 5 (+32)=OverLoad, 4 (+16)=!!! Bit 3..0=Fehler- oder Button-Nr. (bei UserSRQ)
ERC	--	251	Integer	ERC?	#8:251=0	Fehler-Zähler auslesen oder setzen, Zähler wird bei jedem Empfangs-Übertragungsfehler um 1 erhöht
SBD*	--	252	Word	252=38400!, 252?	#8:252=57600	Serielle Baudraten-Einstellung, erst nach Reset gültig, Schreibfreigabe mit WEN=1 erforderlich
WEN	--	250	0..1	WEN=1!, 1:250=1!	#8:255=16 [OK]	EEPROM-Write-Enable-Bit, vor dem Beschreiben von Werten mit Schreibsperre auf 1 setzen, wird danach wieder automatisch auf 0 gesetzt

*** nichtflüchtige EEPROM-Werte mit Schreibsperre**
**** nur Lesen**

neue/geänderte Befehle ROT

UNI-C 1.01c

Cmd	Argument	SubCh	Wertebereich	Beispiele (hier für Adresse 8)	Beispiel-Antwort	Erläuterung
VAL**	4	4		4?	#8:4=48	FPGA-Readflags auf SPI-Register 0
VAL**	5	5		8:5?	#8:5=3	FPGA-Writeflags
VAL**	8	8		8?	#8:8=0.00000 [Hz]	Frequenzzähler/Counter (Eingang T1/PB1 vom ATmega644)
VAL	9	9		8:8=0		1 MHz Frequenzzähler-Bereich
VAL	9	9		8:8=1		100 kHz Frequenzzähler-Bereich
VAL	9	9		8:8=2		10 kHz Frequenzzähler-Bereich
VAL	9	9		8:8=3		1 kHz Frequenzzähler-Bereich
VAL	9	9		8:8=4		100 Hz Frequenzzähler-Bereich
VAL	9	9		8:8=5		Counter-Modus, fallende Flanke an T1-Pin
VAL	9	9		8:8=6		Counter-Modus, steigende Flanke an T1-Pin
						DAC-, PIO- und DDR-Werte: Mit vorheriger Schreibfreigabe WEN=1 erfolgt dauerhafte Abspeicherung im EEPROM als Einschaltwert. Wegen begrenzter Anzahl EEPROM-Schreibvorgänge Einschaltwert nicht ständig ändern!
VAL	10..17	10..17	-10 bis +10	10?	#8:10=10.002	MCP3208, 12-Bit-ADC, 8 Kanäle, Spannungswert 0..+10 (V).
VAL	20..23	20..23	-10 bis +10	8:VAL 20=5.0!, 24=1.2345!		Ausgabewerte DAC 12 Bit 4 Kanäle, Spannungswert -10..+10 (V)
PIO	0..3	30..33				Relais-Ports 0 bis 3, Ausgabewerte
PIO	4..5	34..35				Ports auf IO16-Bridge, Aus- und Eingabewerte
DDR	0..5	44..45				Ports auf IO16-Bridge, Datenrichtung (Bit = 1 -> Ausgang)
RAW**	0..7	50..57				8 ADC-Rohwerte ohne Skalierung/Offset
VAL	60..79	60..79	Integer/LongInt	8:VAL 20=4654!, 24=-16518!		SPI-Register 0 bis 19 im FPGA, sofern vorhanden
DSP	80	80	0..41			Angezeigter Messkanal oder File (8:80=0!) auf PM8 Panel
UDI	89	89	0..1	89=1!		FPGA Update Disable, stellt automatischen Datenaustausch FPGA <-> DAC/ADC/Ports ein
CLK	0..2	90..92	0..23, 0..59	90=23, 90?	#8:90=23	RTC Echtzeituhr Stunden, Minuten, Sekunden
CLK	3..5	93..95	1..31, 1..12	CLK 3=30	#8:93=30	RTC Echtzeituhr Tag, Monat, Jahr
CLK**	6	96		96?, CLK 6?	#8:96=0 [08:55:23]	Uhrzeit als formatierter Antwort-String
CLK**	7	97		97?, CLK 7?	#8:97=0 [15.08.11]	Datum als formatierter Antwort-String
						Die interne Uhr des AVR-Controllers läuft auch ohne externe RTC, sollte dann aber nach dem Einschalten auf die/das richtige Uhrzeit/Datum gesetzt werden, damit angelegte Dateien das korrekte Erstellungsdatum erhalten.
						Defaults: Skalierungen und Offsets für DACs und ADCs. Mit vorheriger Schreibfreigabe WEN=1 erfolgt dauerhafte Abspeicherung im EEPROM als Einschaltwert.
DEF*	0..7	100..107	Integer	8:DEF 0=0!, DEF 1?	#8:255=0 [OK], #8:101=23	Offsets ADC-Wert in ADC-Rohwerten

DEF*	10..13	110..113	Float	DEF 10=1.23		Kanal-Skalierung AD-Wert; Messwert = (ADC-Rohwert + Offset) * Kanal-Skalierung * Basis-Skalierung
DEF*	20..23	120..123	Integer			Offsets DAC-Werte in DAC-Rohwerten
DEF*	30..33	130..133	Float			Kanal-Skalierungen DAC-Werte; Ausgabewert = (Kanal-Skalierung * Spannung / Basis-Skalierung) + Kanal-Offset
DEF*	40	140	Integer			DAC-Basis-Skalierung; möglichste nicht ändern!
DEF*	41	141	Integer			ADC-Basis-Skalierung; möglichste nicht ändern!
OPT	0..9	200..209	Byte	OPT 8=9	OPT 8?	Options-Tabelle, hier nur OPT 8 verwendet (**MainCh)**. Neuer MainCh wird erst beim nächsten Einschalten übernommen.
OPT	10	210	String	OPT 10 ="mystart.ini"		Default-FPGA-Konfigurationsdatei oder INI-Script, hier "STARTUP.INI"
OPT	11	211	String			Default-FileName für mit FWR und FWV angelegte/erweiterte Messdaten-Textdateien, hier "DATAFILE.XLS"
CFG	-	240	Byte	CFG=3!, CFG?	#7:255=0 [OK]	Auswahl und Laden der Datei Nummer <byte> von SD-Karte. Bei BIN- und BIT-Files wird die Datei gelesen und ins FPGA programmiert, INI-Dateien werden als Script ausgeführt. Achtung: OptoBus ist bei Konfigurationsdaten etwa 0,5 Sekunden blockiert!
LST**, DIR**	-	241	-	LST?, DIR?, 241?	#7:241=0 [<FileName1>], #7:241=1 [<FileName2>] usw.	Directory-Liste mit Dateinummern und zugehörigen Dateinamen
FNM**		242	-	FNM?, 7:242?		FileNum, Anzahl der Dateien auf der SD-Karte
CFG		240		CFG=9, CFG=DDS.BIN		File Load, BIT/BIN-Konfigurationen, aber auch INI-, DAT-, MEM-Dateien (werden je nach Suffix anders behandelt)
CFG				CFG=BINDATEI.DAT		BIN- und BIT-Dateien werden als Konfiguration ins FPGA geladen, INI-Dateien werden als LabScript ausgeführt, MEM- und DAT-Dateien auf eine SPI-AutoIncrement-Adresse ausgegeben (z.B. für BlockRAM-Inhalte, DAT immer binär, byteweise!)
CFG				CFG="PICOBLAZ1.MEM"		MEM-Dateien sind BlockRAM-Inhalte für den PicoBlaze-Core. Diese müssen dem von KCPSM3-Assembler ausgegebenen Format entsprechen (Textdatei mit einem Hex-Wert pro Zeile, max. 1024 Werte). **Auto-Increment**-Registernummer (für FPGA-SPI) für MEM- und DAT-Files (z.B. BlockRAM-Inhalte), **ist bei UNI-C immer 128**. Dieses Register ist das Daten-Empfangregister, **nach** jedem Zugriff muss sich ein Adresszähler im FPGA erhöhen. Ein Reset des Zählers erfolgt **vor und nach** jeder Datei-Übertragung durch automatischen Zugriff auf SPI-Register 129. Dieser Zugriff kann daher auch zum Reset des (PicoBlaze-)Cores dienen. Alphanumerische Befehlsargumente können nun auch in "Häkchen" gesetzt werden, somit sind z.B. auch Dateinamen zulässig, die mit Ziffern beginnen. Die interne Uhr des AVR-Controllers läuft auch ohne externe RTC, sollte dann aber nach dem Einschalten auf die/das richtige Uhrzeit/Datum gesetzt werden, damit angelegte Dateien das korrekte Erstellungsdatum erhalten.
FNA		243		FNA="<filename>"		FileName für mit FWR und FWV angelegte/erweiterte Messdaten-Textdateien temporär neu setzen, default der mit OPT 31 angelegte Dateiname.
FDL		244		FDL="<filename>"		FileDelete, beliebige Datei löschen
FQU		249		FQU="<filename>"	#7:255=31 [NOTFOUND]	File Query, liefert 0 [OK] wenn File existiert, sonst Fehlermeldung
FWR		260		FWR = <register>, FWR=3		FileWriteRegister, LabScript-Registerinhalt 0..9 in (ggf. mit FNA bestimmte) Textdatei schreiben, wird automatisch erstellt (wenn nötig), Header angelegt, geöffnet und geschlossen, Registernummer und Uhrzeit werden ebenfalls geschrieben (TAB-getrennt).
FWV		270		FWV <index>=<wert>, FWV=1.23456, FWV 3=775.0		FileWriteValue, angegebenen Wert in (ggf. mit FNA bestimmte) Textdatei schreiben, wird automatisch erstellt (wenn nötig) geöffnet und geschlossen, Header angelegt. Index und Uhrzeit werden ebenfalls geschrieben (TAB-getrennt).

* nichtflüchtige EEPROM-Werte mit Schreibsperre
** nur Lesen

LabScript 1.0

Script-Implementation für FPGA ab 1.2 und UNI-C ab 1.01b

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.

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)
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 10? INP 20?		Akku (Reg. 0) Input, mit Ergebnis von moduleigenem SubCh 0..255 füllen
						Bei UNI-C sind nur die SubCh 10..79 für INP zulässig
OUT		2000..2255		OUT <SubCh>=<register>, OUT 20=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.
END		999				Beendet Script an dieser Stelle

Wünsche und Anregungen bitte an cm@ctmagazin.de