Portierung von ST-Programmen

EX_Press 4 nach EX_Press 5

ST-Element	EX_PRESS 4	EX_PRESS 5

Deklaration von Input-Variablen	VAR_INPUT
	  Start AT E1;
	  Stop AT E2;
	  Reset AT E3;
	  …
	END_VAR;
		VAR_INPUT
		  Start AT In_01;
		  Stop AT In_02;
		  Reset AT In_03;
		  …
		END_VAR

Deklaration von Output-Variablen	VAR_OUTPUT
	  Motor1 AT A1;
	  Motor2 AT A2;
	  Signal AT A3;
	  …
	END_VAR;
		VAR_OUTPUT
		  Motor1 AT Out_01;
		  Motor2 AT Out_02;
		  Signal AT Out_03;
		  …
		END_VAR;

Deklaration des Power-On-Reset Signals	VAR_INPUT
	  …
	  PwrOnRes AT POR;
	END_VAR;
		VAR_INPUT
		  …
		  PwrOnRes AT POR_delayed;
		END_VAR;

Zusammenfassung von Bit-Variablen zu Mehrbit-Variablen	VAR_ARRAY
	  Counter[C3, C2, C1, C0];
	END_VAR;
		VAR_ALIAS
		  Counter[C3, C2, C1, C0];
		END_VAR;

Deklaration von Timern	VAR_INPUT
	  Timer1 AT T1;
	  Timer2 AT T2;
	END_VAR;
		VAR_TIMER
		  Timer1;
		  Timer2;
		END_VAR;

Parametrierung von Timern	VAR_OUTPUT
	  …
	  Timer1_Par_A AT T1A;
	  Timer1_Par_B AT T1B;
	  Timer1_Res AT T1R;
	END_VAR;
	
	(* Zeitbereich: *)
	Timer1_Par_A := 0;
	Timer1_Par_B := 1;
	
	(* Start/Reset *)
	Timer1_Res := NOT Start;
		VAR_TIMER
		  Timer1;
		END_VAR;
		
		(* Zeit/Periodendauer *)
		Timer1 := 500 ms;
		
		(* Start/Reset *)
		Timer1.RESET := NOT Start;
		
		(* Weitere Parameter *)
		Timer1.ENABLE := TRUE;
		Timer1.POL := LOW;
		Timer1.MODE := CONT;

IF-Anweisungen (1)	Bei nicht getakteten Variablen sind unvollständige IF-Anweisungen erlaubt:
	
	IF (Start=1) THEN
	  Motor1 := 1;
	END_IF;
		Bei nicht getakteten Variablen ist in EX_PRESS 5 ein vollständiges IF-Konstrukt erforderlich:
		
		IF (Start=1) THEN
		  Motor1 := 1;
		ELSE
		  Motor1 := 0;
		END_IF;

H09

ST-Element	EX_PRESS 4	EX_PRESS 5

IF-Anweisungen (2)	Folgende IF-Anweisung ist erlaubt:
	
	`IF (Start AND NOT Stop) THEN`
	`  Motor1 := 1;`
	`ELSE`
	`  Motor1 := 0;`
	`END_IF;`
	
	Die Werte von “Start” und “Stop” werden hier als Boole’sche Werte interpretiert.
		Variablen vom Typ „BIT“ werden nicht als Boole’sche Werte interpretiert, so dass die IF-Anweisung mit Vergleichs-operatoren geschrieben werden muss:
		
		`IF (Start=1 AND Stop=0) THEN`
		`  Motor1 := 1;`
		`ELSE`
		`  Motor1 := 0;`
		`END_IF;`

Selbsthaltung (R-S-Flipflop)	Uneingeschränkt erlaubt:
	`Motor1 := Start`
	`OR (Motor1 AND`
	`NOT Stop);`
		Ebenfalls erlaubt:
		`Motor1 := Start`
		`OR (Motor1 AND`
		`NOT Stop);`
		
		Es erscheint jedoch eine “wesentliche Warnung” im Fenster „Fehlermeldun-gen“, weil es sich bei diesem Konstrukt um eine „asynchrone zyklische Zuweisung“ an eine nicht getaktete Variable handelt. Bei der hier programmierten Selbsthaltung (R-S-Flipflop) ist das unproblematisch, der Fitter läuft anschließend ohne Fehler durch.
		
		Nicht erlaubt:
		
		`Signal := NOT Signal;`
		
		Wenn „Signal“ ein nicht getakteter, also asynchroner Ausgang ist, erscheint ebenfalls die „wesentliche Warnung“ im Fenster „Fehlermeldun-gen“. Ein anschließender Fitter-Lauf wird mit einer Fehlermeldung abgebrochen.

H. ZANDER GmbH & Co. KG • Am Gut Wolf 15 • 52070 Aachen • Deutschland
Tel +49 (0)241 9105010 • Fax +49 (0)241 91050138 • info@zander-aachen.de • www.zander-aachen.de