

# Anhang II

## Quellcode der Testverfahren in AWL

Programmiert von:

**Björn Ostermann**

Benutzte Programmierumgebung:

**CoDeSys Version 2.3.4.1 (Build Mar 9 2005)**  
**by 3S – Smart Software Solutions GmbH**

Sankt Augustin, ?? August 2006

## Inhaltsverzeichnis

<b>1</b>	<b>Globale Variablen.....</b>	<b>4</b>
<b>2</b>	<b>Reaktion auf einen erkannten Fehler – ERROR_FB .....</b>	<b>6</b>
<b>3</b>	<b>Fehlererkennung durch Ablaufkontrolle.....</b>	<b>7</b>
<b>3.1</b>	<b>Selbst programmierte Watchdog Funktion mit Ablaufkontrolle.....</b>	<b>7</b>
3.1.1	ABLAUF_UEBERWACHUNG_OHNE_ZEIT.....	8
3.1.2	ABLAUF_UEBERWACHUNG.....	10
3.1.3	ABLAUF_UEBERWACHUNG_RESET.....	13
3.1.4	ABLAUF_UEBERWACHUNG_SCHLEIFE.....	14
<b>4</b>	<b>Fehlererkennung durch Prozessortest .....</b>	<b>17</b>
<b>4.1</b>	<b>Test der bedingten Sprünge – JMP_TEST.....</b>	<b>17</b>
<b>4.2</b>	<b>Test des Akkumulators – ACC_TEST .....</b>	<b>18</b>
<b>4.3</b>	<b>Test der Logischen Operatoren .....</b>	<b>20</b>
4.3.1	LO_AND_TEST.....	21
4.3.2	LO_ANDN_TEST.....	22
4.3.3	LO_NOT_TEST.....	23
4.3.4	LO_OR_TEST.....	24
4.3.5	LO_ORN_TEST.....	25
4.3.6	LO_XOR_TEST.....	26
4.3.7	LO_XORN_TEST.....	27
<b>4.4</b>	<b>Test der Arithmetischen Operatoren .....</b>	<b>28</b>
4.4.1	AR_ADD_TEST.....	29
4.4.2	AR_DIV_TEST.....	30
4.4.3	AR_MOD_TEST.....	31
4.4.4	AR_MUL_TEST.....	32
4.4.5	AR_SUB_TEST.....	33
<b>4.5</b>	<b>Test der Komparatoren .....</b>	<b>34</b>
4.5.1	CO_EQ_TEST.....	35
4.5.2	CO_GE_TEST.....	36
4.5.3	CO_GT_TEST.....	37
4.5.4	CO_LE_TEST.....	38
4.5.5	CO_LT_TEST.....	39
4.5.6	CO_NE_TEST.....	40
<b>4.6</b>	<b>Test des Ladens und Speicherns von Daten – LOAD_STORE_TEST .....</b>	<b>41</b>
<b>5</b>	<b>Fehlererkennung durch Speichertest .....</b>	<b>43</b>
<b>5.1</b>	<b>Test der benutzten Variablen.....</b>	<b>43</b>
5.1.1	VAR_TEST_BOOL.....	44
5.1.2	VAR_TEST_BYTE.....	46
5.1.3	VAR_TEST_DINT.....	49
5.1.4	VAR_TEST_DWORD.....	52

5.1.5	VAR_TEST_GLOBALS.....	55
5.1.6	VAR_TEST_INT.....	66
5.1.7	VAR_TEST_RESET .....	69
5.1.8	VAR_TEST_SINT.....	73
5.1.9	VAR_TEST_UDINT .....	76
5.1.10	VAR_TEST_UINT.....	79
5.1.11	VAR_TEST_USINT.....	82
5.1.12	VAR_TEST_WORD .....	85
<b>5.2</b>	<b>Test des Merkerbereichs mittels eines Arrays.....</b>	<b>88</b>
5.2.1	VAR_TEST_GLOBAL.....	89
5.2.2	VAR_TEST_GLOBAL_CHKSUM.....	97
5.2.3	VAR_TEST_GLOBAL_CHKSUM_SAVEVAR.....	99
5.2.4	VAR_TEST_GLOBAL_SNS.....	101
5.2.5	VAR_TEST_GLOBAL_TEST.....	104
<b>5.3</b>	<b>Test des Speichers mittels eines Zeigers.....</b>	<b>106</b>
5.3.1	VAR_TEST_POINTER.....	107
5.3.2	VAR_TEST_POINTER_ASYNCHONOUSFIELD .....	116
5.3.3	VAR_TEST_POINTER_FIELD .....	124
5.3.4	VAR_TEST_POINTER_OWN .....	131
<b>6</b>	<b>Fehlererkennung durch Test der Ein- und Ausgabe .....</b>	<b>142</b>
<b>6.1</b>	<b>Test des Speichers der Ein- und Ausgänge – O_TEST .....</b>	<b>142</b>
<b>6.2</b>	<b>Test der redundanten Ausgänge über rückgekoppelte Eingänge.....</b>	<b>144</b>
6.2.1	OO_TEST .....	145
6.2.2	OO_TEST_FALSE .....	148
6.2.3	OO_TEST_TRUE.....	150
<b>7</b>	<b>Fehlererkennung durch gegenseitige Überwachung mit einer zweiten SPS .....</b>	<b>155</b>
<b>7.1</b>	<b>Kontrolle der Ablaufkontrolle durch zweite SPS.....</b>	<b>155</b>
7.1.1	ABLAUF_UEBERWACHUNG_INIT.....	156
7.1.2	ABLAUF_UEBERWACHUNG_SEND .....	157
<b>7.2</b>	<b>Test der zweiten Möglichkeit Ausgänge abzuschalten – OO_TEST_ZWEITE_SPS .....</b>	<b>160</b>

# 1 Globale Variablen

## VAR\_GLOBAL

```

Aktueller_Var_Test AT %MD4 : UDINT;          (*used in Var_Test_*)
Start_Var_Test AT %MD5: UDINT;              (*used in Var_Test_*)
Ende_Var_Test AT %MD6: UDINT;              (*used in Var_Test_*)
Begonnen_Var_Test AT %MX0.1: BOOL;         (*used in Var_Test_*)
Beendet_Var_Test AT %MX0.0 : BOOL;         (*used in Var_Test_*)
Hilfs_Variable_BOOL AT %MX0.2 : BOOL;     (*used in Var_Test_Reset &
                                           BOOL*)
Hilfs_Variable_BYTE AT %MB1 : BYTE;       (*used in Var_Test_Reset &
                                           BYTE*)
Hilfs_Variable_WORD AT %MW3 :
WORD;                                       (*used in Var_Test_Reset &
                                           WORD*)
Hilfs_Variable_DWORD AT %MD7 :
DWORD;                                       (*used in Var_Test_Reset &
                                           DWORD*)
Hilfs_Variable_SINT AT %MB2: SINT;        (*used in Var_Test_Reset &
                                           SINT*)
Hilfs_Variable_USINT AT %MB3: USINT;      (*used in Var_Test_Reset &
                                           USINT*)
Hilfs_Variable_INT AT %MW4 : INT;         (*used in Var_Test_Reset &
                                           INT*)
Hilfs_Variable_UINT AT %MW5: UINT;       (*used in Var_Test_Reset &
                                           UINT*)
Hilfs_Variable_DINT AT %MD8: DINT;       (*used in Var_Test_Reset &
                                           DINT*)
Hilfs_Variable_UDINT AT %MD9: UDINT;     (*used in Var_Test_Reset &
                                           UDINT*)

LetzteProgrammstelle AT %MD10
:DWORD;

```

```
LetzterPruefzeitpunkt AT %MD11 : TIME;
SendStatus AT %MD12 : DWORD;
LastSend AT %MD13 : TIME;
ListenStatus AT %MD14 : DWORD;
ListenStatus_last AT %MD15 : DWORD;

SaveVar AT %MD0 : DWORD;
SignatureVar AT %MD1 : DWORD;
TestFeld AT %MD2 : ARRAY [1..4094] OF DWORD;

Sim_Input AT %MD20 : DWORD;
Sim_Output AT %MD20: DWORD;

Output_2_3_Merker AT %MX50.0 : BOOL;
```

```
END_VAR
```

## 2 Reaktion auf einen erkannten Fehler – ERROR\_FB

Variablen:

Keine
-------

Funktion:

LD	0	
ST	Sim_Output	(*Die im Test simulierten Ausgänge werden auf Null gesetzt.*)
Error:		
JMP	Error	(*Schleife, die so lange läuft, bis der Watchdog abschaltet.*)

### 3 Fehlererkennung durch Ablaufkontrolle

#### 3.1 Selbst programmierte Watchdog Funktion mit Ablaufkontrolle

Dieser Test besteht aus den Funktionsbausteinen:

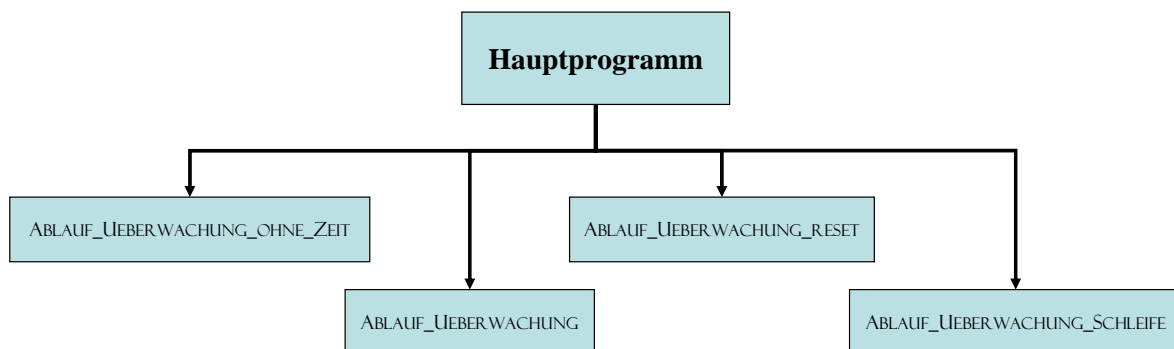
- ABLAUF\_UEBERWACHUNG\_OHNE\_ZEIT
- ABLAUF\_UEBERWACHUNG
- ABLAUF\_UEBERWACHUNG\_RESET
- ABLAUF\_UEBERWACHUNG\_SCHLEIFE

Die Bausteine:

- ABLAUF\_UEBERWACHUNG\_INIT
- ABLAUF\_UEBERWACHUNG\_SEND

gehören zur Kontrolle der Ablaufkontrolle durch eine zweite SPS und werden in Kapitel 7.1 dargestellt.

Die Aufrufhierarchie gliedert sich wie folgt:



Dies bedeutet, dass alle Bausteine aus dem Hauptprogramm heraus aufgerufen werden.

### 3.1.1 ABLAUF\_UEBERWACHUNG\_OHNE\_ZEIT

Variablen:

VAR_INPUT		
GeringsterVorgaenger :DWORD;		Die Bausteinnummer die mindestens vor diesem Baustein aufgerufen werden musste.
Programmstelle: DWORD;		Eigene Bausteinnummer
END_VAR		
VAR		
FB_Error:Error_FB;		Fehlerbaustein aus Kapitel 1
END_VAR		

Funktion:

JMP	Start	
Error:		(*Bei einem erkannten Fehler, Aufruf Fehler Baustein (inklusive Endlosschleife) und zweite Endlosschleife*)
CAL	FB_Error	
CalError:		
JMP	CalError	
Start:		
LD	GeringsterVorgaenger	(*vergleiche Geringsten Vorgänger mit der Letzten Programmstelle*)
GT	LetzteProgrammstelle	
JMPC	Error	(*Fehleransprung, wenn zuletzt gespeicherte Programmstelle kleiner ist als der erwartetet Geringste Vorgänger*)
LD	Programmstelle	(*vergleich aktuelle Programmstelle und letzte Programmstelle*)
LE	LetzteProgrammstelle	

JMPC	Error	(*Fehleransprung, wenn aktuelle Programmstelle kleiner oder gleich der letzten Programmstelle ist*)
LD	Programmstelle	(*Speichern aktuelle Position*)
ST	LetzteProgrammstelle	

### 3.1.2 ABLAUF\_UEBERWACHUNG

Variablen:

VAR_INPUT	
GeringsterVorgaenger :DWORD;	Die Bausteinnummer die mindestens vor diesem Baustein aufgerufen werden musste.
Programmstelle: DWORD;	Eigene Bausteinnummer
minZeit:TIME;	Zeit die mindestens seit der letzten Zeitmessung vergangen sein muss
maxZeit:TIME;	Zeit die maximal seit der letzten Zeitmessung vergangen sein darf
END_VAR	
VAR	
Jetzt:TIME;	Hilfsvariable in der die Systemzeit abgelegt wird
Zeit_zwischen_Pruefungen:TIME;	Errechneter Zeitunterschied zwischen der aktuellen und der letzten Zeitüberwachung
FB_Error:Error_FB;	Fehlerbaustein aus Kapitel 1
END_VAR	

Funktion:

JMP	Start	
Error:		(*Bei einem erkannten Fehler, Aufruf Fehler Baustein (inklusive Endlosschleife) und zweite Endlosschleife*)
CAL	FB_Error	
CalError:		
JMP	CalError	
Start:		
LD	GeringsterVorgaenger	(*vergleiche Geringsten Vorgänger mit der Letzten Programmstelle*)

GT	LetzteProgrammstelle	
JMPC	Error	(*Fehleransprung, wenn zuletzt gespeicherte Programmstelle kleiner ist als der erwartete Geringste Vorgänger*)
LD	Programmstelle	(*vergleich aktuelle Programmstelle und letzte Programmstelle*)
LE	LetzteProgrammstelle	
JMPC	Error	(*Fehleransprung, wenn aktuelle Programmstelle kleiner oder gleich der letzten Programmstelle ist*)
LD	Programmstelle	(*Speichern aktuelle Position*)
ST	LetzteProgrammstelle	
TIME		(*Aktuelle Zeit wird nach "Jetzt" geschrieben, da TIME selbst lange dauert, und so außerdem während der Routine keine Änderung der Systemzeit Fehler produzieren kann*)
ST	Jetzt	
SUB	LetzterPruefzeitpunkt	
ST	Zeit_zwischen_Pruefungen	(*Zeit_zwischen_Pruefungen wird mit der Zeit zwischen den Prüfaufrufen belegt. Damit wird das Problem des Überlaufs beseitigt, der auftritt, wenn die Zeiterfassung die maximale Zeitgröße überschreitet.*)
LD	Jetzt	
SUB	minZeit	
GT	Zeit_zwischen_Pruefungen	(*Wenn die minimal erlaubte Zeit seit der letzten Prüfung größer ist als die wirklich vergangene Zeit, ist ein Fehler aufgetreten.*)
JMPC	Error	

LD	Jetzt	
SUB	maxZeit	
LT	Zeit_zwischen_Pruefungen	(*Wenn die maximal erlaubte Zeit seit der letzten Prüfung kleiner ist als die wirklich vergangene Zeit, ist ein Fehler aufgetreten.*)
JMPC	Error	
LD	Jetzt	
ST	LetzterPruefzeitpunkt	(*Speichern der Systemzeit*)

### 3.13 ABLAUF\_UEBERWACHUNG\_RESET

Variablen:

Keine
-------

Funktion:

		(*Aufruf vor jedem Zyklus um Variablen zurückzusetzen*)
LD	0	
ST	LetzteProgrammstelle	
TIME		
ST	LetzterPruefzeitpunkt	

### 3.14 ABLAUF\_UEBERWACHUNG\_SCHLEIFE

Variablen:

VAR_INPUT	
GeringsterVorgaenger :DWORD;	Die Bausteinnummer die mindestens vor diesem Baustein aufgerufen werden musste.
Programmstelle: DWORD;	Eigene Bausteinnummer
minZeit:TIME;	Zeit die mindestens seit der letzten Zeitmessung vergangen sein muss
maxZeit:TIME;	Zeit die maximal seit der letzten Zeitmessung vergangen sein darf
END_VAR	
VAR	
Jetzt:TIME;	Hilfsvariable in der die Systemzeit abgelegt wird
Zeit_zwischen_Pruefungen:TIME;	Errechner Zeitunterschied zwischen der aktuellen und der letzten Zeitüberwachung
FB_Error:Error_FB;	Fehlerbaustein aus Kapitel 1
END_VAR	

Funktion:

JMP	Start	
Error:		(*Bei einem erkannten Fehler, Aufruf Fehler Baustein (inklusive Endlosschleife) und zweite Endlosschleife*)
CAL	FB_Error	
CalError:		
JMP	CalError	
Start:		
LD	GeringsterVorgaenger	(*vergleiche Geringsten Vorgänger mit der Letzten Programmstelle*)

GT	LetzteProgrammstelle	
JMPC	Error	(*Fehleransprung, wenn zuletzt gespeicherte Programmstelle kleiner ist als der erwartete Geringste Vorgänger*)
LD	Programmstelle	(*vergleich aktuelle Programmstelle und letzte Programmstelle*)
LT	LetzteProgrammstelle	(*Es wird Gleichheit erlaubt. Damit kann diese Routine in einer Schleife verwendet werden, ohne einen Fehler auszulösen. Es wird immernoch die Zeit geprüft, und ein Gefangenbleiben in einer Schleife wird mittels Watchdog abgefangen.*)
JMPC	Error	(*Fehleransprung, wenn aktuelle Programmstelle kleiner der letzten Programmstelle ist*)
LD	Programmstelle	(*Speichern aktuelle Position*)
ST	LetzteProgrammstelle	
TIME		(*Aktuelle Zeit wird nach "Jetzt" geschrieben, da TIME selbst lange dauert, und so außerdem während der Routine keine Änderung der Systemzeit Fehler produzieren kann*)
ST	Jetzt	
SUB	LetzterPruefzeitpunkt	
ST	Zeit_zwischen_Pruefungen	(*Zeit_zwischen_Pruefungen wird mit der Zeit zwischen den Prüfaufrufen belegt. Damit wird das Problem des Überlaufs beseitigt, der auftritt, wenn die Zeiterfassung die maximale Zeitgröße überschreitet.*)
LD	Jetzt	
SUB	minZeit	

GT	Zeit_zwischen_Pruefungen	(*Wenn die minimal erlaubte Zeit seit der letzten Prüfung größer ist als die wirklich vergangene Zeit, ist ein Fehler aufgetreten.*)
JMPC	Error	
LD	Jetzt	
SUB	maxZeit	
LT	Zeit_zwischen_Pruefungen	(*Wenn die maximal erlaubte Zeit seit der letzten Prüfung kleiner ist als die wirklich vergangene Zeit, ist ein Fehler aufgetreten.*)
JMPC	Error	
LD	Jetzt	
ST	LetzterPruefzeitpunkt	(*Speichern der Systemzeit*)

## 4 Fehlererkennung durch Prozessortest

Die hier beschriebenen Tests können alle einzeln oder Blockweise aus dem Hauptprogramm ausgeführt werden. Die einzelnen Tests selbst sind so kurz, dass sich ein Aufteilen einzelner Tests auf mehrere Zyklen nicht lohnt. Möglich wäre eine Abarbeitung, bei der in jedem Zyklus nur einer der Tests dieses Kapitels ausgeführt wird, oder ein Aussetzen des Speichertests für ein Zyklus und dafür eine Komplettüberwachung des Prozessors.

### 4.1 Test der bedingten Sprünge – JMP\_TEST

Variablen:

VAR	
FB_Error:Error_FB;	Fehlerbaustein aus Kapitel 1
END_VAR	

Funktion:

LD	TRUE	
JMPCN	Error	(*Sprung bei FALSE im Akku wird getestet. Bei Auslösen von Sprung Error.*)
LD	TRUE	
JMPC	Weiter1	
JMP	Error	
Weiter1:		
LD	FALSE	
JMPC	Error	(*Sprung bei TRUE im Akku wird getestet. Bei Auslösen von Sprung Error.*)
LD	FALSE	
JMPCN	Weiter2	
JMP	Error	
Weiter2:		
RET		
Error:		
CAL	FB_Error	

CalError:

JMP	CalError	(*Schleife wartet auf den Watchdog*)
-----	----------	--------------------------------------

## 4.2 Test des Akkumulators – ACC\_TEST

Variablen:

VAR

FB\_Error:Error\_FB;

Fehlerbaustein aus Kapitel 1

END\_VAR

Funktion:

LD	32768	(*1000 0000 0000 0000(B) wird in den Akku geladen*)
SHR	1	(*nach 15 Rechtsshift sollte 0000 0000 0000 0001(B) im Akku stehen*)
SHR	1	
SHR	1	
SHR	1	
SHR	1	
SHR	1	
SHR	1	
SHR	1	
SHR	1	
SHR	1	
SHR	1	
SHR	1	
SHR	1	
SHR	1	
SHR	1	
SHR	1	
SHR	1	
EQ	1	(*verwandlung in TRUE*)

JMPCN	Error	(*WennTRUE ankommt ist alles OK, verlassen des FB. Wenn FALSE hat der Akku eine hängende Eins oder Null. Dann weiter zu Error*)
LD	1	(*0000 0000 0000 0001(B) wird in den Akku geladen*)
SHL	1	(*nach 15 Rechtsshift sollte 0000 0000 0000 0000(B) im Akku stehen*)
SHL	1	
SHL	1	
SHL	1	
SHL	1	
SHL	1	
SHL	1	
SHL	1	
SHL	1	
SHL	1	
SHL	1	
SHL	1	
SHL	1	
SHL	1	
EQ	32768	
RETC		(*WennTRUE ankommt ist alles OK, verlassen des FB. Wenn FALSE hat der Akku eine hängende Eins oder Null. Dann weiter zu Error*)
Error:		
CAL	FB_Error	
CalError:		
JMP	CalError	(*Schleife wartet auf den Watchdog*)

### 4.3 Test der Logischen Operatoren

Unter diesem Titel sind folgende Testbausteine zusammengefasst:

- LO\_AND\_TEST
- LO\_ANDN\_TEST
- LO\_NOT\_TEST
- LO\_OR\_TEST
- LO\_ORN\_TEST
- LO\_XOR\_TEST
- LO\_XORN\_TEST

Wie zum Beginn des Kapitel 4 erwähnt arbeiten alle Tests auf gleicher Ebene.

### 4.3.1 LO\_AND\_TEST

Variablen:

VAR

FB\_Error:Error\_FB;

Fehlerbaustein aus Kapitel 1

END\_VAR

Funktion:

LD	FALSE	(*Prüfung AND*)
AND	FALSE	
JMPC	Error	
LD	FALSE	
AND	TRUE	
JMPC	Error	
LD	TRUE	
AND	FALSE	
JMPC	Error	
LD	TRUE	
AND	TRUE	
JMPCN	Error	
RET		
Error:		
CAL	FB_Error	
CalError:		
JMP	CalError	(*Schleife wartet auf den Watchdog*)

## 4.3.2 LO\_ANDN\_TEST

Variablen:

VAR

FB\_Error:Error\_FB;

Fehlerbaustein aus Kapitel 1

END\_VAR

Funktion:

LD	FALSE	(*Prüfung ANDN*)
ANDN	FALSE	
JMPC	Error	
LD	FALSE	
ANDN	TRUE	
JMPC	Error	
LD	TRUE	
ANDN	FALSE	
JMPCN	Error	
LD	TRUE	
ANDN	TRUE	
JMPC	Error	
RET		
Error:		
CAL	FB_Error	
CalError:		
JMP	CalError	(*Schleife wartet auf den Watchdog*)

### 4.3.3 LO\_NOT\_TEST

Variablen:

VAR

FB\_Error:Error\_FB;

Fehlerbaustein aus Kapitel 1

END\_VAR

Funktion:

LD FALSE (\*Prüfung NOT\*)

NOT

JMPCN Error

LD TRUE

NOT

JMPC Error

RET

Error:

CAL FB\_Error

CalError:

JMP CalError (\*Schleife wartet auf den Watchdog\*)

### 4.3.4 LO\_OR\_TEST

Variablen:

VAR

FB\_Error:Error\_FB;

Fehlerbaustein aus Kapitel 1

END\_VAR

Funktion:

LD	FALSE	(*Prüfung OR*)
OR	FALSE	
JMPC	Error	
LD	FALSE	
OR	TRUE	
JMPCN	Error	
LD	TRUE	
OR	FALSE	
JMPCN	Error	
LD	TRUE	
OR	TRUE	
JMPCN	Error	
RET		
Error:		
CAL	FB_Error	
CalError:		
JMP	CalError	(*Schleife wartet auf den Watchdog*)

### 4.3.5 LO\_ORN\_TEST

Variablen:

VAR

FB\_Error:Error\_FB;

Fehlerbaustein aus Kapitel 1

END\_VAR

Funktion:

LD	FALSE	(*Prüfung OR*)
ORN	FALSE	
JMPCN	Error	
LD	FALSE	
ORN	TRUE	
JMPC	Error	
LD	TRUE	
ORN	FALSE	
JMPCN	Error	
LD	TRUE	
ORN	TRUE	
JMPCN	Error	
RET		
Error:		
CAL	FB_Error	
CalError:		
JMP	CalError	(*Schleife wartet auf den Watchdog*)

### 4.3.6 LO\_XOR\_TEST

Variablen:

VAR

FB\_Error:Error\_FB;

Fehlerbaustein aus Kapitel 1

END\_VAR

Funktion:

LD	FALSE	(*Prüfung XOR*)
XOR	FALSE	
JMPC	Error	
LD	FALSE	
XOR	TRUE	
JMPCN	Error	
LD	TRUE	
XOR	FALSE	
JMPCN	Error	
LD	TRUE	
XOR	TRUE	
JMPC	Error	
RET		
Error:		
CAL	FB_Error	
CalError:		
JMP	CalError	(*Schleife wartet auf den Watchdog*)

## 4.3.7 LO\_XORN\_TEST

Variablen:

VAR

FB\_Error:Error\_FB;

Fehlerbaustein aus Kapitel 1

END\_VAR

Funktion:

LD	FALSE	(*Prüfung XORN*)
XORN	FALSE	
JMPCN	Error	
LD	FALSE	
XORN	TRUE	
JMPC	Error	
LD	TRUE	
XORN	FALSE	
JMPC	Error	
LD	TRUE	
XORN	TRUE	
JMPCN	Error	
RET		
Error:		
CAL	FB_Error	
CalError:		
JMP	CalError	(*Schleife wartet auf den Watchdog*)

#### **4.4 Test der Arithmetischen Operatoren**

Unter diesem Titel sind folgende Testbausteine zusammengefasst:

- AR\_ADD\_TEST
- AR\_DIV\_TEST
- AR\_MOD\_TEST
- AR\_MUL\_TEST
- AR\_SUB\_TEST

Wie zum Beginn des Kapitel 4 erwähnt arbeiten alle Tests auf gleicher Ebene.

## 4.4.1 AR\_ADD\_TEST

Variablen:

VAR

FB\_Error:Error\_FB;

Fehlerbaustein aus Kapitel 1

END\_VAR

Funktion:

LD	170	(*1010 1010(B)*)
ADD	85	(*0101 0101(B)*)
EQ	255	(*1111 1111(B)*)
JMPCN	Error	
LD	255	(*0000 0000 1111 1111(B)*)
ADD	65280	(*1111 1111 0000 0000(B)*)
EQ	65535	(*1111 1111 1111 1111(B)*)
RETC		
Error:		
CAL	FB_Error	
CalError:		
JMP	CalError	(*Schleife wartet auf den Watchdog*)

## 4.4.2 AR\_DIV\_TEST

Variablen:

VAR	
FB_Error:Error_FB;	Fehlerbaustein aus Kapitel 1
END_VAR	

Funktion:

LD	170	(*1010 1010(B)*)
DIV	10	(*0000 1010(B)*)
EQ	17	(*0001 0001(B)*)
JMPCN	Error	
LD	65534	(*1111 1111 1111 1110(B)*)
DIV	255	(*0000 0000 1111 1111(B)*)
EQ	256	(*0000 0001 0000 0000(B)*)
RETC		
Error:		
CAL	FB_Error	
CalError:		
JMP	CalError	(*Schleife wartet auf den Watchdog*)

### 4.4.3 AR\_MOD\_TEST

Variablen:

VAR	
FB_Error:Error_FB;	Fehlerbaustein aus Kapitel 1
END_VAR	

Funktion:

LD	170	(*1010 1010(B)*)
MOD	10	(*0000 1010(B)*)
EQ	0	(*0000 0000(B)*)
JMPCN	Error	
LD	65534	(*1111 1111 1111 1110(B)*)
MOD	255	(*0000 0000 1111 1111(B)*)
EQ	254	(*0000 0000 1111 1110(B)*)
RETC		
Error:		
CAL	FB_Error	
CalError:		
JMP	CalError	(*Schleife wartet auf den Watchdog*)

#### 4.4.4 AR\_MUL\_TEST

Variablen:

VAR	
FB_Error:Error_FB;	Fehlerbaustein aus Kapitel 1
END_VAR	

Funktion:

LD	17	(*0001 0001(B)*)
MUL	10	(*0000 1010(B)*)
EQ	170	(*1010 1010(B)*)
JMPCN	Error	
LD	256	(*0000 0001 0000 0000(B)*)
MUL	255	(*0000 0000 1111 1111(B)*)
EQ	65280	(*1111 1111 0000 0000(B)*)
RETC		
Error:		
CAL	FB_Error	
CalError:		
JMP	CalError	(*Schleife wartet auf den Watchdog*)

#### 4.4.5 AR\_SUB\_TEST

Variablen:

VAR		
FB_Error:Error_FB;		Fehlerbaustein aus Kapitel 1
END_VAR		

Funktion:

LD	255	(*1111 1111(B)*)
SUB	85	(*0101 0101(B)*)
EQ	170	(*1010 1010(B)*)
JMPCN	Error	
LD	65535	(*1111 1111 1111 1111(B)*)
SUB	65280	(*1111 1111 0000 0000(B)*)
EQ	255	(*0000 0000 1111 1111(B)*)
RETC		
Error:		
CAL	FB_Error	
CalError:		
JMP	CalError	(*Schleife wartet auf den Watchdog*)

## 4.5 Test der Komparatoren

Unter diesem Titel sind folgende Testbausteine zusammengefasst:

- CO\_EQ\_TEST
- CO\_GE\_TEST
- CO\_GT\_TEST
- CO\_LE\_TEST
- CO\_LT\_TEST
- CO\_NE\_TEST

Wie zum Beginn des Kapitel 4 erwähnt arbeiten alle Tests auf gleicher Ebene.

## 4.5.1 CO\_EQ\_TEST

Variablen:

VAR

FB\_Error:Error\_FB;

Fehlerbaustein aus Kapitel 1

END\_VAR

Funktion:

LD 170 (\*1010 1010(B)\*)

EQ 170

JMPCN Error

LD 85 (\*0101 0101(B)\*)

EQ 170

RETCN

Error:

CAL FB\_Error

CalError:

JMP CalError (\*Schleife wartet auf den Watchdog\*)

## 4.5.2 CO\_GE\_TEST

Variablen:

VAR

FB\_Error:Error\_FB;

Fehlerbaustein aus Kapitel 1

END\_VAR

Funktion:

LD	15	(*0000 1111(B)*)
GE	240	(*1111 0000(B)*)
JMPC	Error	
LD	240	
GE	240	
JMPCN	Error	
LD	240	
GE	15	
RETC		
Error:		
CAL	FB_Error	
CalError:		
JMP	CalError	(*Schleife wartet auf den Watchdog*)

### 4.5.3 CO\_GT\_TEST

Variablen:

VAR

FB\_Error:Error\_FB;

Fehlerbaustein aus Kapitel 1

END\_VAR

Funktion:

LD            15            (\*0000 1111(B)\*)

LT            240            (\*1111 0000(B)\*)

JMPCN        Error

LD            240

LT            15

RETCN

Error:

CAL           FB\_Error

CalError:

JMP           CalError            (\*Schleife wartet auf den Watchdog\*)

## 4.5.4 CO\_LE\_TEST

Variablen:

VAR

FB\_Error:Error\_FB;

Fehlerbaustein aus Kapitel 1

END\_VAR

Funktion:

LD 15 (\*0000 1111(B)\*)

LE 240 (\*1111 0000(B)\*)

JMPCN Error

LD 240

LE 240

JMPCN Error

LD 240

LE 15

RETCN

Error:

CAL FB\_Error

CalError:

JMP CalError (\*Schleife wartet auf den Watchdog\*)

## 4.5.5 CO\_LT\_TEST

Variablen:

VAR

FB\_Error:Error\_FB;

Fehlerbaustein aus Kapitel 1

END\_VAR

Funktion:

LD 7

GT 248

JMPC Error

LD 240

GT 15

RETC

Error:

CAL FB\_Error

CalError:

JMP CalError (\*Schleife wartet auf den Watchdog\*)

## 4.5.6 CO\_NE\_TEST

Variablen:

VAR		
FB_Error:Error_FB;		Fehlerbaustein aus Kapitel 1
END_VAR		

Funktion:

LD	85	(*0101 0101(B)*)
NE	170	(*1010 1010(B)*)
JMPCN	Error	
LD	170	(*1010 1010(B)*)
NE	170	(*1010 1010(B)*)
RETCN		
Error:		
CAL	FB_Error	
CalError:		
JMP	CalError	(*Schleife wartet auf den Watchdog*)

## 4.6 Test des Ladens und Speicherns von Daten – LOAD\_STORE\_TEST

Variablen:

```

VAR
Test_Var_b1: BYTE := 170;           (*1010 1010(B)*)
Test_Var_b2: BYTE := 0;            (*0000 0000(B)*)
Test_Var_i1 : INT := 21845;         (*0101 0101 0101 0101(B)*)
Test_Var_i2 : INT := 512;           (*0000 0010 0000 0000(B)*)
Test_Var_dw1: DWORD := 4294967295; (*11111111111111111111111111111111(B)*)
Test_Var_dw2: DWORD := 2863311530; (*10101010101010101010101010101010(B)*)
FB_Error:Error_FB;                 Fehlerbaustein aus Kapitel 1
END_VAR

```

Funktion:

```

LD      Test_var_b1
ST      Test_var_b2
LD      Test_var_b1
EQ      Test_var_b2
JMPCN   Error
LD      Test_var_i1
ST      Test_var_i2
LD      Test_var_i1
EQ      Test_var_i2
JMPCN   Error
LD      Test_var_dw1
ST      Test_var_dw2
LD      Test_var_dw1
EQ      Test_var_dw2
RETC

```

Error:

CAL            FB\_Error

CalError:

JMP            CalError            (\*Schleife wartet auf den Watchdog\*)

## 5 Fehlererkennung durch Speichertest

### 5.1 Test der benutzten Variablen

Dieser Test besteht aus den Funktionsbausteinen:

- VAR\_TEST\_BOOL
- VAR\_TEST\_BYTE
- VAR\_TEST\_DINT
- VAR\_TEST\_DWORD
- VAR\_TEST\_GLOBALS
- VAR\_TEST\_INT
- VAR\_TEST\_RESET
- VAR\_TEST\_SINT
- VAR\_TEST\_UDINT
- VAR\_TEST\_UINT
- VAR\_TEST\_USINT
- VAR\_TEST\_WORD

Dabei wird der Test VAR\_TEST\_GLOBALS aus der Routine VAR\_TEST\_RESET aufgerufen. Diese wird jeweils einmal pro Zyklus aufgerufen. Die restlichen Tests werden je einmal pro genutzter Variable vom entsprechenden Variablentyp aufgerufen. Es wird empfohlen alle Funktionen im Hauptprogramm global zu deklarieren und nicht erneut in jeder Unterfunktion.

## 5.11 VAR\_TEST\_BOOL

Variablen:

VAR_IN_OUT	
Test_Variable:BOOL;	Pointer auf die Speicherstelle, an der die zu testende Variable steht.
END_VAR	
VAR	
FB_Error:Error_FB;	Fehlerbaustein aus Kapitel 1
END_VAR	

Funktion:

LD	Aktueller_Var_Test	(*Incrementieren der Zählvariable*)
ADD	1	(*Incrementieren der Zählvariable*)
ST	Aktueller_Var_Test	(*Incrementieren der Zählvariable*)
LD	Beendet_Var_Test	(*Prüfen ob Variablen Test schon beendet ist, d.h. ob die zu testenden Variablen getestet wurden*)
RETC		(*Rücksprung wenn Test schon beendet*)
LD	Begonnen_Var_Test	(*Prüfen ob Test schon begonnen hat*)
JMPC	Beendet_Test	(*Test ob dies die letzte zu testende Variable des Zyklus ist*)
LD	Start_Var_Test	(*Test ob dies die erste zu testende Variable ist*)
EQ	Aktueller_Var_Test	
RETCN		(*Rücksprung wenn Test noch nicht begonnen*)
ST	Begonnen_Var_Test	
JMP	Testroutine	
Beendet_Test :		
LD	Ende_Var_Test	(*Test ob dies die letzte zu testende Variable ist*)

EQ	Aktueller_Var_Test	
ST	Beendet_Var_Test	
Testroutine:		
LD	Test_Variable	(*Test_Variable wegspeichern*)
ST	Hilfs_Variable_BOOL	
LD	FALSE	(*Beginn eigendlicher Test: Zahl laden, Zahl speichern, Speicher laden, Speicher prüfen*)
ST	Test_Variable	
LD	Test_Variable	
JMPC	Error	
LD	TRUE	
ST	Test_Variable	
LD	Test_Variable	
JMPC	NoError	
Error:		
CAL	FB_Error	
CalError:		
JMP	CalError	
NoError:		
LD	Hilfs_Variable_BOOL	(*Getestete Variable zurückschreiben*)
ST	Test_Variable	

## 5.12 VAR\_TEST\_BYTE

Variablen:

VAR_IN_OUT	
Test_Variable:BYTE;	Pointer auf die Speicherstelle, an der die zu testende Variable steht.
END_VAR	
VAR	
FB_Error:Error_FB;	Fehlerbaustein aus Kapitel 1
END_VAR	

Funktion:

LD	Aktueller_Var_Test	(*Incrementieren der Zählvariable*)
ADD	1	(*Incrementieren der Zählvariable*)
ST	Aktueller_Var_Test	(*Incrementieren der Zählvariable*)
LD	Beendet_Var_Test	(*Prüfen ob Variablen Test schon beendet ist, d.h. ob die zu testenden Variablen getestet wurden*)
RETC		(*Rücksprung wenn Test schon beendet*)
LD	Begonnen_Var_Test	(*Prüfen ob Test schon begonnen hat*)
JMPC	Beendet_Test	(*Test ob dies die letzte zu testende Variable des Zyklus ist*)
LD	Start_Var_Test	(*Test ob dies die erste zu testende Variable ist*)
EQ	Aktueller_Var_Test	
RETCN		(*Rücksprung wenn Test noch nicht begonnen*)
ST	Begonnen_Var_Test	
JMP	Testroutine	
Beendet_Test :		
LD	Ende_Var_Test	(*Test ob dies die letzte zu testende Variable ist*)

EQ	Aktueller_Var_Test	
ST	Beendet_Var_Test	
Testroutine:		
LD	Test_Variable	(*Test_Variable wegspeichern*)
ST	Hilfs_Variable_BYTE	
LD	255	(*Beginn eigendlicher Test: Zahl laden, Zahl speichern, Speicher laden, Speicher prüfen*)
ST	Test_Variable	
LD	Test_Variable	
EQ	255	(*1111 1111(B)*)
JMPCN	Error	
LD	0	(*0000 0000(B)*)
ST	Test_Variable	
LD	Test_Variable	
EQ	0	(*0000 0000(B)*)
JMPCN	Error	
LD	85	(*0101 0101(B)*)
ST	Test_Variable	
LD	Test_Variable	
EQ	85	(*0101 0101(B)*)
JMPCN	Error	
LD	170	(*1010 1010(B)*)
ST	Test_Variable	
LD	Test_Variable	
EQ	170	(*1010 1010(B)*)
JMPC	NoError	
Error:		

CAL	FB_Error	
CalError:		
JMP	CalError	
NoError:		
LD	Hilfs_Variable_BYTE	(*Getestete Variable zurückschreiben*)
ST	Test_Variable	

### 5.13 VAR\_TEST\_DINT

Variablen:

VAR_IN_OUT	
Test_Variable:DINT;	Pointer auf die Speicherstelle, an der die zu testende Variable steht.
END_VAR	
VAR	
FB_Error:Error_FB;	Fehlerbaustein aus Kapitel 1
END_VAR	

Funktion:

LD	Aktueller_Var_Test	(*Incrementieren der Zählvariable*)
ADD	1	(*Incrementieren der Zählvariable*)
ST	Aktueller_Var_Test	(*Incrementieren der Zählvariable*)
LD	Beendet_Var_Test	(*Prüfen ob Variablen Test schon beendet ist, d.h. ob die zu testenden Variablen getestet wurden*)
RETC		(*Rücksprung wenn Test schon beendet*)
LD	Begonnen_Var_Test	(*Prüfen ob Test schon begonnen hat*)
JMPC	Beendet_Test	(*Test ob dies die letzte zu testende Variable des Zyklus ist*)
LD	Start_Var_Test	(*Test ob dies die erste zu testende Variable ist*)
EQ	Aktueller_Var_Test	
RETCN		(*Rücksprung wenn Test noch nicht begonnen*)
ST	Begonnen_Var_Test	
JMP	Testroutine	
Beendet_Test :		
LD	Ende_Var_Test	(*Test ob dies die letzte zu testende Variable ist*)

EQ	Aktueller_Var_Test	
ST	Beendet_Var_Test	
Testroutine:		
LD	Test_Variable	(*Test_Variable wegspeichern*)
ST	Hilfs_Variable_DINT	
LD	-1	(*Beginn eigendlicher Test: Zahl laden, Zahl speichern, Speicher laden, Speicher prüfen*)
ST	Test_Variable	
LD	Test_Variable	
EQ	-1	(*1111 1111 1111 1111 1111 1111 1111 1111(B)*)
JMPCN	Error	
LD	0	(*0000 0000 0000 0000 0000 0000 0000 0000(B)*)
ST	Test_Variable	
LD	Test_Variable	
EQ	0	(*0000 0000 0000 0000 0000 0000 0000 0000(B)*)
JMPCN	Error	
LD	1431655765	(*0101 0101 0101 0101 0101 0101 0101 0101(B)*)
ST	Test_Variable	
LD	Test_Variable	
EQ	1431655765	(*0101 0101 0101 0101 0101 0101 0101 0101(B)*)
JMPCN	Error	
LD	-1431655766	(*1010 1010 1010 1010 1010 1010 1010 1010(B)*)
ST	Test_Variable	

LD	Test_Variable	
EQ	-1431655766	(*1010 1010 1010 1010 1010 1010 1010 1010(B)*)
JMPC	NoError	
Error:		
CAL	FB_Error	
CalError:		
JMP	CalError	
NoError:		
LD	Hilfs_Variable_DINT	(*Getestete Variable zurückschreiben*)
ST	Test_Variable	

## 5.14 VAR\_TEST\_DWORD

Variablen:

VAR_IN_OUT	
Test_Variable:DWORD;	Pointer auf die Speicherstelle, an der die zu testende Variable steht.
END_VAR	
VAR	
FB_Error:Error_FB;	Fehlerbaustein aus Kapitel 1
END_VAR	

Funktion:

LD	Aktueller_Var_Test	(*Incrementieren der Zählvariable*)
ADD	1	(*Incrementieren der Zählvariable*)
ST	Aktueller_Var_Test	(*Incrementieren der Zählvariable*)
LD	Beendet_Var_Test	(*Prüfen ob Variablen Test schon beendet ist, d.h. ob die zu testenden Variablen getestet wurden*)
RETC		(*Rücksprung wenn Test schon beendet*)
LD	Begonnen_Var_Test	(*Prüfen ob Test schon begonnen hat*)
JMPC	Beendet_Test	(*Test ob dies die letzte zu testende Variable des Zyklus ist*)
LD	Start_Var_Test	(*Test ob dies die erste zu testende Variable ist*)
EQ	Aktueller_Var_Test	
RETCN		(*Rücksprung wenn Test noch nicht begonnen*)
ST	Begonnen_Var_Test	
JMP	Testroutine	
Beendet_Test :		

LD	Ende_Var_Test	(*Test ob dies die letzte zu testende Variable ist*)
EQ	Aktueller_Var_Test	
ST	Beendet_Var_Test	
Testroutine:		
LD	Test_Variable	(*Test_Variable wegspeichern*)
ST	Hilfs_Variable_DWORD	
LD	4294967295	(*Beginn eigendlicher Test: Zahl laden, Zahl speichern, Speicher laden, Speicher prüfen*)
ST	Test_Variable	
LD	Test_Variable	
EQ	4294967295	(*1111 1111 1111 1111 1111 1111 1111 1111(B)*)
JMPCN	Error	
LD	0	(*0000 0000 0000 0000 0000 0000 0000 0000(B)*)
ST	Test_Variable	
LD	Test_Variable	
EQ	0	(*0000 0000 0000 0000 0000 0000 0000 0000(B)*)
JMPCN	Error	
LD	1431655765	(*0101 0101 0101 0101 0101 0101 0101 0101(B)*)
ST	Test_Variable	
LD	Test_Variable	
EQ	1431655765	(*0101 0101 0101 0101 0101 0101 0101 0101(B)*)
JMPCN	Error	

LD	2863311530	(*1010 1010 1010 1010 1010 1010 1010 1010 1010(B)*)
ST	Test_Variable	
LD	Test_Variable	
EQ	2863311530	(*1010 1010 1010 1010 1010 1010 1010 1010 1010(B)*)
JMPC	NoError	
Error:		
CAL	FB_Error	
CalError:		
JMP	CalError	
NoError:		
LD	Hilfs_Variable_DWORD	(*Getestete Variable zurückschreiben*)
ST	Test_Variable	

## 5.15 VAR\_TEST\_GLOBALS

Variablen:

VAR_IN_OUT		
Test_Var_Anzahl_pro_Durchlauf :UDINT;		Pointer auf die Speicherstelle, an der die zu testende Variable steht.
END_VAR		
VAR		
Test_Var_UDINT: Var_Test_UDINT;		Testroutine aus Kapitel 5.1.9
Test_Var_BOOL: Var_Test_BOOL;		Testroutine aus Kapitel 5.1.1
FB_Error:Error_FB;		Fehlerbaustein aus Kapitel 1
END_VAR		

Funktion:

CAL	Test_Var_UDINT (Test_Variable:=Aktueller_Var_Test)	
CAL	Test_Var_UDINT (Test_Variable:=Start_Var_Test)	
CAL	Test_Var_UDINT (Test_Variable:=Ende_Var_Test)	
CAL	Test_Var_BOOL (Test_Variable:=Begonnen_Var_Test)	
CAL	Test_Var_BOOL (Test_Variable:=Beendet_Var_Test)	
LD	Aktueller_Var_Test	(*Incrementieren der Zählvariable*)
ADD	1	(*Incrementieren der Zählvariable*)
ST	Aktueller_Var_Test	(*Incrementieren der Zählvariable*)
LD	Beendet_Var_Test	(*Prüfen ob Variablen Test schon beendet ist, d.h. ob die zu testenden Variablen getestet wurden*)
RETC		(*Rücksprung wenn Test schon beendet*)
LD	Begonnen_Var_Test	(*Prüfen ob Test schon begonnen hat*)
JMPC	Beendet_Test	(*Test ob dies die letzte zu testende Variable des Zyklus ist*)

LD	Start_Var_Test	(*Test ob dies die erste zu testende Variable ist*)
EQ	Aktueller_Var_Test	
RETCN		(*Rücksprung wenn Test noch nicht begonnen*)
ST	Begonnen_Var_Test	
JMP	Testroutine	
Beendet_Test :		
LD	Ende_Var_Test	(*Test ob dies die letzte zu testende Variable ist*)
EQ	Aktueller_Var_Test	
ST	Beendet_Var_Test	
Testroutine:		
LD	FALSE	(*Beginn eigendlicher Test: Zahl laden, Zahl speichern, Speicher laden, Speicher prüfen*)
ST	Hilfs_Variable_BOOL	(*Alle HilfsVariablen werden jede Runde getestet, da ein Fehler hier sich in alle im Zyklus getesteten Variablen übertragen würde.*)
LD	Hilfs_Variable_BOOL	
JMPC	Error	
LD	TRUE	
ST	Hilfs_Variable_BOOL	
LD	Hilfs_Variable_BOOL	
JMPCN	Error	
LD	255	(*Beginn eigendlicher Test: Zahl laden, Zahl speichern, Speicher laden, Speicher prüfen*)

ST	Hilfs_Variable_BYTE	
LD	Hilfs_Variable_BYTE	
EQ	255	(*1111 1111(B)*)
JMPCN	Error	
LD	0	(*0000 0000(B)*)
ST	Hilfs_Variable_BYTE	
LD	Hilfs_Variable_BYTE	
EQ	0	(*0000 0000(B)*)
JMPCN	Error	
LD	85	(*0101 0101(B)*)
ST	Hilfs_Variable_BYTE	
LD	Hilfs_Variable_BYTE	
EQ	85	(*0101 0101(B)*)
JMPCN	Error	
LD	170	(*1010 1010(B)*)
ST	Hilfs_Variable_BYTE	
LD	Hilfs_Variable_BYTE	
EQ	170	(*1010 1010(B)*)
JMPCN	Error	
LD	-1	(*Beginn eigendlicher Test: Zahl laden, Zahl speichern, Speicher laden, Speicher prüfen*)
ST	Hilfs_Variable_DINT	
LD	Hilfs_Variable_DINT	
EQ	-1	(*1111 1111 1111 1111 1111 1111 1111 1111(B)*)
JMPCN	Error	

LD	0	(*0000 0000 0000 0000 0000 0000 0000 0000(B)*)
ST	Hilfs_Variable_DINT	
LD	Hilfs_Variable_DINT	
EQ	0	(*0000 0000 0000 0000 0000 0000 0000 0000(B)*)
JMPCN	Error	
LD	1431655765	(*0101 0101 0101 0101 0101 0101 0101 0101(B)*)
ST	Hilfs_Variable_DINT	
LD	Hilfs_Variable_DINT	
EQ	1431655765	(*0101 0101 0101 0101 0101 0101 0101 0101(B)*)
JMPCN	Error	
LD	-1431655766	(*1010 1010 1010 1010 1010 1010 1010 1010(B)*)
ST	Hilfs_Variable_DINT	
LD	Hilfs_Variable_DINT	
EQ	-1431655766	(*1010 1010 1010 1010 1010 1010 1010 1010(B)*)
JMPCN	Error	
LD	4294967295	(*Beginn eigendlicher Test: Zahl laden, Zahl speichern, Speicher laden, Speicher prüfen*)
ST	Hilfs_Variable_DWORD	
LD	Hilfs_Variable_DWORD	
EQ	4294967295	(*1111 1111 1111 1111 1111 1111 1111 1111(B)*)
JMPCN	Error	

LD	0	(*0000 0000 0000 0000 0000 0000 0000 0000(B)*)
ST	Hilfs_Variable_DWORD	
LD	Hilfs_Variable_DWORD	
EQ	0	(*0000 0000 0000 0000 0000 0000 0000 0000(B)*)
JMPCN	Error	
LD	1431655765	(*0101 0101 0101 0101 0101 0101 0101 0101(B)*)
ST	Hilfs_Variable_DWORD	
LD	Hilfs_Variable_DWORD	
EQ	1431655765	(*0101 0101 0101 0101 0101 0101 0101 0101(B)*)
JMPCN	Error	
LD	2863311530	(*1010 1010 1010 1010 1010 1010 1010 1010(B)*)
ST	Hilfs_Variable_DWORD	
LD	Hilfs_Variable_DWORD	
EQ	2863311530	(*1010 1010 1010 1010 1010 1010 1010 1010(B)*)
JMPCN	Error	
LD	-1	(*Beginn eigendlicher Test: Zahl laden, Zahl speichern, Speicher laden, Speicher prüfen*)
ST	Hilfs_Variable_INT	
LD	Hilfs_Variable_INT	
EQ	-1	(*1111 1111 1111 1111(B)*)
JMPCN	Error	
LD	0	(*0000 0000 0000 0000(B)*)

ST	Hilfs_Variable_INT	
LD	Hilfs_Variable_INT	
EQ	0	(*0000 0000 0000 0000(B)*)
JMPCN	Error	
LD	21845	(*0101 0101 0101 0101(B)*)
ST	Hilfs_Variable_INT	
LD	Hilfs_Variable_INT	
EQ	21845	(*0101 0101 0101 0101(B)*)
JMPCN	Error	
LD	-21846	(*1010 1010 1010 1010(B)*)
ST	Hilfs_Variable_INT	
LD	Hilfs_Variable_INT	
EQ	-21846	(*1010 1010 1010 1010(B)*)
JMPCN	Error	
LD	-1	(*Beginn eigendlicher Test: Zahl laden, Zahl speichern, Speicher laden, Speicher prüfen*)
ST	Hilfs_Variable_SINT	
LD	Hilfs_Variable_SINT	
EQ	-1	(*1111 1111(B)*)
JMPCN	Error	
LD	0	(*0000 0000(B)*)
ST	Hilfs_Variable_SINT	
LD	Hilfs_Variable_SINT	
EQ	0	(*0000 0000(B)*)
JMPCN	Error	
LD	85	(*0101 0101(B)*)
ST	Hilfs_Variable_SINT	

LD	Hilfs_Variable_SINT	
EQ	85	(*0101 0101(B)*)
JMPCN	Error	
LD	-86	(*1010 1010(B)*)
ST	Hilfs_Variable_SINT	
LD	Hilfs_Variable_SINT	
EQ	-86	(*1010 1010(B)*)
JMPCN	Error	
LD	4294967295	(*Beginn eigendlicher Test: Zahl laden, Zahl speichern, Speicher laden, Speicher prüfen*)
ST	Hilfs_Variable_UDINT	
LD	Hilfs_Variable_UDINT	
EQ	4294967295	(*1111 1111 1111 1111 1111 1111 1111 1111(B)*)
JMPCN	Error	
LD	0	(*0000 0000 0000 0000 0000 0000 0000 0000(B)*)
ST	Hilfs_Variable_UDINT	
LD	Hilfs_Variable_UDINT	
EQ	0	(*0000 0000 0000 0000 0000 0000 0000 0000(B)*)
JMPCN	Error	
LD	1431655765	(*0101 0101 0101 0101 0101 0101 0101 0101(B)*)
ST	Hilfs_Variable_UDINT	
LD	Hilfs_Variable_UDINT	
EQ	1431655765	(*0101 0101 0101 0101 0101 0101 0101 0101(B)*)

JMPCN	Error	
LD	2863311530	(*1010 1010 1010 1010 1010 1010 1010 1010(B)*)
ST	Hilfs_Variable_UDINT	
LD	Hilfs_Variable_UDINT	
EQ	2863311530	(*1010 1010 1010 1010 1010 1010 1010 1010(B)*)
JMPCN	Error	
LD	Test_Var_Anzahl_pro_Durchlauf	
ST	Hilfs_Variable_UDINT	
LD	4294967295	(*Beginn eigendlicher Test: Zahl laden, Zahl speichern, Speicher laden, Speicher prüfen*)
ST	Test_Var_Anzahl_pro_Durchlauf	
LD	Test_Var_Anzahl_pro_Durchlauf	
EQ	4294967295	(*1111 1111 1111 1111 1111 1111 1111 1111(B)*)
JMPCN	Error	
LD	0	(*0000 0000 0000 0000 0000 0000 0000 0000(B)*)
ST	Test_Var_Anzahl_pro_Durchlauf	
LD	Test_Var_Anzahl_pro_Durchlauf	
EQ	0	(*0000 0000 0000 0000 0000 0000 0000 0000(B)*)
JMPCN	Error	
LD	1431655765	(*0101 0101 0101 0101 0101 0101 0101 0101(B)*)
ST	Test_Var_Anzahl_pro_Durchlauf	
LD	Test_Var_Anzahl_pro_Durchlauf	

EQ	1431655765	(*0101 0101 0101 0101 0101 0101 0101 0101(B)*)
JMPCN	Error	
LD	2863311530	(*1010 1010 1010 1010 1010 1010 1010 1010(B)*)
ST	Test_Var_Anzahl_pro_Durchlauf	
LD	Test_Var_Anzahl_pro_Durchlauf	
EQ	2863311530	(*1010 1010 1010 1010 1010 1010 1010 1010(B)*)
JMPCN	Error	
LD	Hilfs_Variable_UDINT	
ST	Test_Var_Anzahl_pro_Durchlauf	
LD	65535	(*Beginn eigendlicher Test: Zahl laden, Zahl speichern, Speicher laden, Speicher prüfen*)
ST	Hilfs_Variable_UINT	
LD	Hilfs_Variable_UINT	
EQ	65535	(*1111 1111 1111 1111(B)*)
JMPCN	Error	
LD	0	(*0000 0000 0000 0000(B)*)
ST	Hilfs_Variable_UINT	
LD	Hilfs_Variable_UINT	
EQ	0	(*0000 0000 0000 0000(B)*)
JMPCN	Error	
LD	21845	(*0101 0101 0101 0101(B)*)
ST	Hilfs_Variable_UINT	
LD	Hilfs_Variable_UINT	
EQ	21845	(*0101 0101 0101 0101(B)*)

JMPCN	Error	
LD	43690	(*1010 1010 1010 1010(B)*)
ST	Hilfs_Variable_UINT	
LD	Hilfs_Variable_UINT	
EQ	43690	(*1010 1010 1010 1010(B)*)
JMPCN	Error	
		(*Beginn eigendlicher Test: Zahl laden, Zahl speichern, Speicher laden, Speicher prüfen*)
LD	255	
ST	Hilfs_Variable_USINT	
LD	Hilfs_Variable_USINT	
EQ	255	(*1111 1111(B)*)
JMPCN	Error	
LD	0	(*0000 0000(B)*)
ST	Hilfs_Variable_USINT	
LD	Hilfs_Variable_USINT	
EQ	0	(*0000 0000(B)*)
JMPCN	Error	
LD	85	(*0101 0101(B)*)
ST	Hilfs_Variable_USINT	
LD	Hilfs_Variable_USINT	
EQ	85	(*0101 0101(B)*)
JMPCN	Error	
LD	170	(*1010 1010(B)*)
ST	Hilfs_Variable_USINT	
LD	Hilfs_Variable_USINT	
EQ	170	(*1010 1010(B)*)
JMPCN	Error	

LD	65535	(*Beginn eigendlicher Test: Zahl laden, Zahl speichern, Speicher laden, Speicher prüfen*)
ST	Hilfs_Variable_WORD	
LD	Hilfs_Variable_WORD	
EQ	65535	(*1111 1111 1111 1111(B)*)
JMPCN	Error	
LD	0	(*0000 0000 0000 0000(B)*)
ST	Hilfs_Variable_WORD	
LD	Hilfs_Variable_WORD	
EQ	0	(*0000 0000 0000 0000(B)*)
JMPCN	Error	
LD	21845	(*0101 0101 0101 0101(B)*)
ST	Hilfs_Variable_WORD	
LD	Hilfs_Variable_WORD	
EQ	21845	(*0101 0101 0101 0101(B)*)
JMPCN	Error	
LD	43690	(*1010 1010 1010 1010(B)*)
ST	Hilfs_Variable_WORD	
LD	Hilfs_Variable_WORD	
EQ	43690	(*1010 1010 1010 1010(B)*)
RETC		
Error:		
CAL	FB_Error	
CalError:		
JMP	CalError	

## 5.1.6 VAR\_TEST\_INT

Variablen:

VAR_IN_OUT	
Test_Variable:INT;	Pointer auf die Speicherstelle, an der die zu testende Variable steht.
END_VAR	
VAR	
FB_Error:Error_FB;	Fehlerbaustein aus Kapitel 1
END_VAR	

Funktion:

LD	Aktueller_Var_Test	(*Incrementieren der Zählvariable*)
ADD	1	(*Incrementieren der Zählvariable*)
ST	Aktueller_Var_Test	(*Incrementieren der Zählvariable*)
LD	Beendet_Var_Test	(*Prüfen ob Variablen Test schon beendet ist, d.h. ob die zu testenden Variablen getestet wurden*)
RETC		(*Rücksprung wenn Test schon beendet*)
LD	Begonnen_Var_Test	(*Prüfen ob Test schon begonnen hat*)
JMPC	Beendet_Test	(*Test ob dies die letzte zu testende Variable des Zyklus ist*)
LD	Start_Var_Test	(*Test ob dies die erste zu testende Variable ist*)
EQ	Aktueller_Var_Test	
RETCN		(*Rücksprung wenn Test noch nicht begonnen*)
ST	Begonnen_Var_Test	
JMP	Testroutine	
Beendet_Test :		
LD	Ende_Var_Test	(*Test ob dies die letzte zu testende Variable ist*)

EQ	Aktueller_Var_Test	
ST	Beendet_Var_Test	
Testroutine:		
LD	Test_Variable	(*Test_Variable wegspeichern*)
ST	Hilfs_Variable_INT	
LD	-1	(*Beginn eigendlicher Test: Zahl laden, Zahl speichern, Speicher laden, Speicher prüfen*)
ST	Test_Variable	
LD	Test_Variable	
EQ	-1	(*1111 1111 1111 1111(B)*)
JMPCN	Error	
LD	0	(*0000 0000 0000 0000(B)*)
ST	Test_Variable	
LD	Test_Variable	
EQ	0	(*0000 0000 0000 0000(B)*)
JMPCN	Error	
LD	21845	(*0101 0101 0101 0101(B)*)
ST	Test_Variable	
LD	Test_Variable	
EQ	21845	(*0101 0101 0101 0101(B)*)
JMPCN	Error	
LD	-21846	(*1010 1010 1010 1010(B)*)
ST	Test_Variable	
LD	Test_Variable	
EQ	-21846	(*1010 1010 1010 1010(B)*)
JMPC	NoError	
Error:		

CAL	FB_Error
-----	----------

CalError:

JMP	CalError
-----	----------

NoError:

LD	Hilfs_Variable_INT	(*Getestete Variable zurückschreiben*)
----	--------------------	--

ST	Test_Variable
----	---------------

## 5.17 VAR\_TEST\_RESET

Variablen:

VAR_INPUT	
Anzahl_Durchläufe_bis_Gesamttest :BYTE;	Anzahl der Zyklen die durchlaufen werden sollen, bevor alle Variablen einmal getestet wurden.
END_VAR	
VAR	
Test_Var_Globals: Var_Test_Globals;	Testroutine aus Kapitel 5.1.5
Firststart : BOOL := TRUE;	Merker, ob die Routine das erste Mal aufgerufen wird
Warmup : BOOL := FALSE;	Merker, ob die Zählung der Testbausteine beendet ist und das zyklische Testen begonnen werden kann.
FB_Error:Error_FB;	Fehlerbaustein aus Kapitel 1
END_VAR	

Funktion:

LD	Warmup	(*Prüfung ob Warmup (2. Durchlauf) bereits geschehen*)
JMPC	Testroutine	(*Sprung in den Selbsttest*)
LD	Firststart	(*Prüfung ob Firststart (1. Durchlauf) noch ansteht*)
JMPCN	Warmupmarke	(*Bei 2. Durchlauf Berechnung interner Variablen*)
LD	FALSE	(*Rücksetzen von Firststart -> 1. Durchlauf und Begonnen_Var_Test (erst bei 2. Durchlauf benötigt)*)
ST	Firststart	
LD	Firststart	(*Selbsttest, ob Firststart auch FALSE enthalten kann*)
JMPC	Error	
LD	TRUE	(*Begonnen Beendet Start und Ende werden so gesetzt, dass alles geprüft wird.*)

ST	Begonnen_Var_Test	
LD	FALSE	
ST	Beendet_Var_Test	
LD	1	(*Initialisieren von Start_Var_Test und Ende_Var_Test, so dass jeder Variablen test ausgelöst wird.*)
ST	Start_Var_Test	
LD	0	
ST	Ende_Var_Test	
ST	Aktueller_Var _Test	(*Reset Aktueller_Var_Test. Ab hier wird gezählt*)
RET		(*Nach erstem Durchlauf kein Selbsttest. Erstmal Testbausteine zählen.*)
Warmupmarke:		(*Bei 2. Durchlauf Berechnung interner Variablen*)
LD	TRUE	(*Selbsttest, ob Firststart auch TRUE enthalten kann*)
ST	Firststart	
LD	Firststart	
JMPCN	Error	
LD	TRUE	(*2. Durchlauf merken*)
ST	Warmup	
LD	Warmup	(*Selbsttest ob Warmup TRUE enthalten kann*)
JMPCN	Error	
LD	FALSE	(*Beendet_Var_Test False setzen, dadurch ab jetzt Tests möglich*)
ST	Beendet_Var_Test	
LD	Aktueller_Var _Test	(*Berechnen der nötigen Variablen tests pro Zyklus, um in Anzahl_Durchlaeufe_bis_Gesamttest den gesamten Speicher einmal geprüft zu haben.*)

ADD	6	(*Var_Test_Globals und Untertests noch nicht gezählt*)
DIV	Anzahl_Durchlaeufe_bis_Gesamttest	
ADD	1	
ST	Anzahl_Tests_pro_Durchlauf	
Testroutine:		
LD	FALSE	(*Selbsttest ob Warmup TRUE enthalten kann*)
ST	Warmup	
LD	Warmup	
JMPC	Error	
LD	TRUE	(*Kein Selbsttest, da oben schon auf TRUE getestet. Wenn dies nicht möglich wäre würde dieser Teil garnicht ausgeführt.*)
ST	Warmup	
CAL	Test_Var_Glob als (Test_Var_Anz ahl_pro_Durch lauf := Anzahl_Tests_ pro_Durchlauf)	(*Aufrufen des Selbsttests der Globalen Variablen*)
		(*Errechnen von Anfang und Ende nächster Testlauf*)
LD	Ende_Var_Test	
ADD	1	
ST	Start_Var_Test	
ADD	Anzahl_Tests_pro_Durchlauf	

```
ST      Ende_Var_Test
GE      Aktueller_Var      (*Prüfen ob Ende des Speichers erreicht ist*)
      _Test
JMPC    Test_am_Ende_vom_Speicher
JMP     NoError
Error:
CAL     FB_Error
CalError:
JMP     CalError
Test_am_Ende_vom_Speicher:

LD      0
ST      Ende_Var_Tes
      t
NoError:
LD      0
ST      Aktueller_Var_Test
LD      FALSE
ST      Begonnen_Var_Test
ST      Beendet_Var_Test
```

## 5.1.8 VAR\_TEST\_SINT

Variablen:

VAR_IN_OUT	
Test_Variable:SINT;	Pointer auf die Speicherstelle, an der die zu testende Variable steht.
END_VAR	
VAR	
FB_Error:Error_FB;	Fehlerbaustein aus Kapitel 1
END_VAR	

Funktion:

LD	Aktueller_Var_Test	(*Incrementieren der Zählvariable*)
ADD	1	(*Incrementieren der Zählvariable*)
ST	Aktueller_Var_Test	(*Incrementieren der Zählvariable*)
LD	Beendet_Var_Test	(*Prüfen ob Variablen Test schon beendet ist, d.h. ob die zu testenden Variablen getestet wurden*)
RETC		(*Rücksprung wenn Test schon beendet*)
LD	Begonnen_Var_Test	(*Prüfen ob Test schon begonnen hat*)
JMPC	Beendet_Test	(*Test ob dies die letzte zu testende Variable des Zyklus ist*)
LD	Start_Var_Test	(*Test ob dies die erste zu testende Variable ist*)
EQ	Aktueller_Var_Test	
RETCN		(*Rücksprung wenn Test noch nicht begonnen*)
ST	Begonnen_Var_Test	
JMP	Testroutine	
Beendet_Test		
:		

LD	Ende_Var_Test	(*Test ob dies die letzte zu testende Variable ist*)
EQ	Aktueller_Var_Test	
ST	Beendet_Var_Test	
Testroutine:		
LD	Test_Variable	(*Test_Variable wegspeichern*)
ST	Hilfs_Variable_SINT	
LD	-1	(*Beginn eigendlicher Test: Zahl laden, Zahl speichern, Speicher laden, Speicher prüfen*)
ST	Test_Variable	
LD	Test_Variable	
EQ	-1	(*1111 1111(B)*)
JMPCN	Error	
LD	0	(*0000 0000(B)*)
ST	Test_Variable	
LD	Test_Variable	
EQ	0	(*0000 0000(B)*)
JMPCN	Error	
LD	85	(*0101 0101(B)*)
ST	Test_Variable	
LD	Test_Variable	
EQ	85	(*0101 0101(B)*)
JMPCN	Error	
LD	-86	(*1010 1010(B)*)
ST	Test_Variable	
LD	Test_Variable	
EQ	-86	(*1010 1010(B)*)
JMPC	NoError	

Error:

CAL                   FB\_Error

CalError:

JMP                   CalError

NoError:

LD                   Hilfs\_Variable\_SINT       (\*Getestete Variable zurückschreiben\*)

ST                   Test\_Variable

## 5.1.9 VAR\_TEST\_UDINT

Variablen:

VAR_IN_OUT	
Test_Variable:UDINT;	Pointer auf die Speicherstelle, an der die zu testende Variable steht.
END_VAR	
VAR	
FB_Error:Error_FB;	Fehlerbaustein aus Kapitel 1
END_VAR	

Funktion:

LD	Aktueller_Var_Test	(*Incrementieren der Zählvariable*)
ADD	1	(*Incrementieren der Zählvariable*)
ST	Aktueller_Var_Test	(*Incrementieren der Zählvariable*)
LD	Beendet_Var_Test	(*Prüfen ob Variablen Test schon beendet ist, d.h. ob die zu testenden Variablen getestet wurden*)
RETC		(*Rücksprung wenn Test schon beendet*)
LD	Begonnen_Var_Test	(*Prüfen ob Test schon begonnen hat*)
JMPC	Beendet_Test	(*Test ob dies die letzte zu testende Variable des Zyklus ist*)
LD	Start_Var_Test	(*Test ob dies die erste zu testende Variable ist*)
EQ	Aktueller_Var_Test	
RETCN		(*Rücksprung wenn Test noch nicht begonnen*)
ST	Begonnen_Var_Test	
JMP	Testroutine	
Beendet_Test :		

LD	Ende_Var_Test	(*Test ob dies die letzte zu testende Variable ist*)
EQ	Aktueller_Var_Test	
ST	Beendet_Var_Test	
Testroutine:		
LD	Test_Variable	(*Test_Variable wegspeichern*)
ST	Hilfs_Variable_UDINT	
LD	4294967295	(*Beginn eigendlicher Test: Zahl laden, Zahl speichern, Speicher laden, Speicher prüfen*)
ST	Test_Variable	
LD	Test_Variable	
EQ	4294967295	(*1111 1111 1111 1111 1111 1111 1111 1111(B)*)
JMPCN	Error	
LD	0	(*0000 0000 0000 0000 0000 0000 0000 0000(B)*)
ST	Test_Variable	
LD	Test_Variable	
EQ	0	(*0000 0000 0000 0000 0000 0000 0000 0000(B)*)
JMPCN	Error	
LD	1431655765	(*0101 0101 0101 0101 0101 0101 0101 0101(B)*)
ST	Test_Variable	
LD	Test_Variable	
EQ	1431655765	(*0101 0101 0101 0101 0101 0101 0101 0101(B)*)
JMPCN	Error	

LD	2863311530	(*1010 1010 1010 1010 1010 1010 1010 1010(B)*)
ST	Test_Variable	
LD	Test_Variable	
EQ	2863311530	(*1010 1010 1010 1010 1010 1010 1010 1010(B)*)
JMPC	NoError	
Error:		
CAL	FB_Error	
CalError:		
JMP	CalError	
NoError:		
LD	Hilfs_Variable_UDINT	(*Getestete Variable zurückschreiben*)
ST	Test_Variable	

## 5.1.10 VAR\_TEST\_UINT

Variablen:

VAR_IN_OUT	
Test_Variable:UINT;	Pointer auf die Speicherstelle, an der die zu testende Variable steht.
END_VAR	
VAR	
FB_Error:Error_FB;	Fehlerbaustein aus Kapitel 1
END_VAR	

Funktion:

LD	Aktueller_Var_Test	(*Incrementieren der Zählvariable*)
ADD	1	(*Incrementieren der Zählvariable*)
ST	Aktueller_Var_Test	(*Incrementieren der Zählvariable*)
LD	Beendet_Var_Test	(*Prüfen ob Variablen Test schon beendet ist, d.h. ob die zu testenden Variablen getestet wurden*)
RETC		(*Rücksprung wenn Test schon beendet*)
LD	Begonnen_Var_Test	(*Prüfen ob Test schon begonnen hat*)
JMPC	Beendet_Test	(*Test ob dies die letzte zu testende Variable des Zyklus ist*)
LD	Start_Var_Test	(*Test ob dies die erste zu testende Variable ist*)
EQ	Aktueller_Var_Test	
RETCN		(*Rücksprung wenn Test noch nicht begonnen*)
ST	Begonnen_Var_Test	
JMP	Testroutine	
Beendet_Test		
:		

LD	Ende_Var_Test	(*Test ob dies die letzte zu testende Variable ist*)
EQ	Aktueller_Var_Test	
ST	Beendet_Var_Test	
Testroutine:		
LD	Test_Variable	(*Test_Variable wegspeichern*)
ST	Hilfs_Variable_UINT	
LD	65535	(*Beginn eigendlicher Test: Zahl laden, Zahl speichern, Speicher laden, Speicher prüfen*)
ST	Test_Variable	
LD	Test_Variable	
EQ	65535	(*1111 1111 1111 1111(B)*)
JMPCN	Error	
LD	0	(*0000 0000 0000 0000(B)*)
ST	Test_Variable	
LD	Test_Variable	
EQ	0	(*0000 0000 0000 0000(B)*)
JMPCN	Error	
LD	21845	(*0101 0101 0101 0101(B)*)
ST	Test_Variable	
LD	Test_Variable	
EQ	21845	(*0101 0101 0101 0101(B)*)
JMPCN	Error	
LD	43690	(*1010 1010 1010 1010(B)*)
ST	Test_Variable	
LD	Test_Variable	
EQ	43690	(*1010 1010 1010 1010(B)*)
JMPC	NoError	

Error:	
CAL	FB_Error
CalError:	
JMP	CalError
NoError:	
LD	Hilfs_Variable_UINT (*Getestete Variable zurückschreiben*)
ST	Test_Variable

### 5.1.11 VAR\_TEST\_USINT

Variablen:

VAR_IN_OUT	
Test_Variable:USINT;	Pointer auf die Speicherstelle, an der die zu testende Variable steht.
END_VAR	
VAR	
FB_Error:Error_FB;	Fehlerbaustein aus Kapitel 1
END_VAR	

Funktion:

LD	Aktueller_Var_Test	(*Incrementieren der Zählvariable*)
ADD	1	(*Incrementieren der Zählvariable*)
ST	Aktueller_Var_Test	(*Incrementieren der Zählvariable*)
LD	Beendet_Var_Test	(*Prüfen ob Variablen Test schon beendet ist, d.h. ob die zu testenden Variablen getestet wurden*)
RETC		(*Rücksprung wenn Test schon beendet*)
LD	Begonnen_Var_Test	(*Prüfen ob Test schon begonnen hat*)
JMPC	Beendet_Test	(*Test ob dies die letzte zu testende Variable des Zyklus ist*)
LD	Start_Var_Test	(*Test ob dies die erste zu testende Variable ist*)
EQ	Aktueller_Var_Test	
RETCN		(*Rücksprung wenn Test noch nicht begonnen*)
ST	Begonnen_Var_Test	
JMP	Testroutine	
Beendet_Test		
:		

LD	Ende_Var_Test	(*Test ob dies die letzte zu testende Variable ist*)
EQ	Aktueller_Var_Test	
ST	Beendet_Var_Test	
Testroutine:		
LD	Test_Variable	(*Test_Variable wegspeichern*)
ST	Hilfs_Variable_USINT	
LD	255	(*Beginn eigendlicher Test: Zahl laden, Zahl speichern, Speicher laden, Speicher prüfen*)
ST	Test_Variable	
LD	Test_Variable	
EQ	255	(*1111 1111(B)*)
JMPCN	Error	
LD	0	(*0000 0000(B)*)
ST	Test_Variable	
LD	Test_Variable	
EQ	0	(*0000 0000(B)*)
JMPCN	Error	
LD	85	(*0101 0101(B)*)
ST	Test_Variable	
LD	Test_Variable	
EQ	85	(*0101 0101(B)*)
JMPCN	Error	
LD	170	(*1010 1010(B)*)
ST	Test_Variable	
LD	Test_Variable	
EQ	170	(*1010 1010(B)*)
JMPC	NoError	

Error:		
CAL	FB_Error	
CalError:		
JMP	CalError	
NoError:		
LD	Hilfs_Variable_USINT	(*Getestete Variable zurückschreiben*)
ST	Test_Variable	

## 5.1.12 VAR\_TEST\_WORD

Variablen:

VAR_IN_OUT	
Test_Variable:WORD;	Pointer auf die Speicherstelle, an der die zu testende Variable steht.
END_VAR	
VAR	
FB_Error:Error_FB;	Fehlerbaustein aus Kapitel 1
END_VAR	

Funktion:

LD	Aktueller_Var_Test	(*Incrementieren der Zählvariable*)
ADD	1	(*Incrementieren der Zählvariable*)
ST	Aktueller_Var_Test	(*Incrementieren der Zählvariable*)
LD	Beendet_Var_Test	(*Prüfen ob Variablen Test schon beendet ist, d.h. ob die zu testenden Variablen getestet wurden*)
RETC		(*Rücksprung wenn Test schon beendet*)
LD	Begonnen_Var_Test	(*Prüfen ob Test schon begonnen hat*)
JMPC	Beendet_Test	(*Test ob dies die letzte zu testende Variable des Zyklus ist*)
LD	Start_Var_Test	(*Test ob dies die erste zu testende Variable ist*)
EQ	Aktueller_Var_Test	
RETCN		(*Rücksprung wenn Test noch nicht begonnen*)
ST	Begonnen_Var_Test	
JMP	Testroutine	
Beendet_Test :		

LD	Ende_Var_Test	(*Test ob dies die letzte zu testende Variable ist*)
EQ	Aktueller_Var_Test	
ST	Beendet_Var_Test	
Testroutine:		
LD	Test_Variable	(*Test_Variable wegspeichern*)
ST	Hilfs_Variable_WORD	
LD	65535	(*Beginn eigendlicher Test: Zahl laden, Zahl speichern, Speicher laden, Speicher prüfen*)
ST	Test_Variable	
LD	Test_Variable	
EQ	65535	(*1111 1111 1111 1111(B)*)
JMPCN	Error	
LD	0	(*0000 0000 0000 0000(B)*)
ST	Test_Variable	
LD	Test_Variable	
EQ	0	(*0000 0000 0000 0000(B)*)
JMPCN	Error	
LD	21845	(*0101 0101 0101 0101(B)*)
ST	Test_Variable	
LD	Test_Variable	
EQ	21845	(*0101 0101 0101 0101(B)*)
JMPCN	Error	
LD	43690	(*1010 1010 1010 1010(B)*)
ST	Test_Variable	
LD	Test_Variable	
EQ	43690	(*1010 1010 1010 1010(B)*)
JMPC	NoError	

Error:

CAL                   FB\_Error

CalError:

JMP                   CalError

NoError:

LD                   Hilfs\_Variable\_WORD           (\*Getestete Variable zurückschreiben\*)

ST                   Test\_Variable

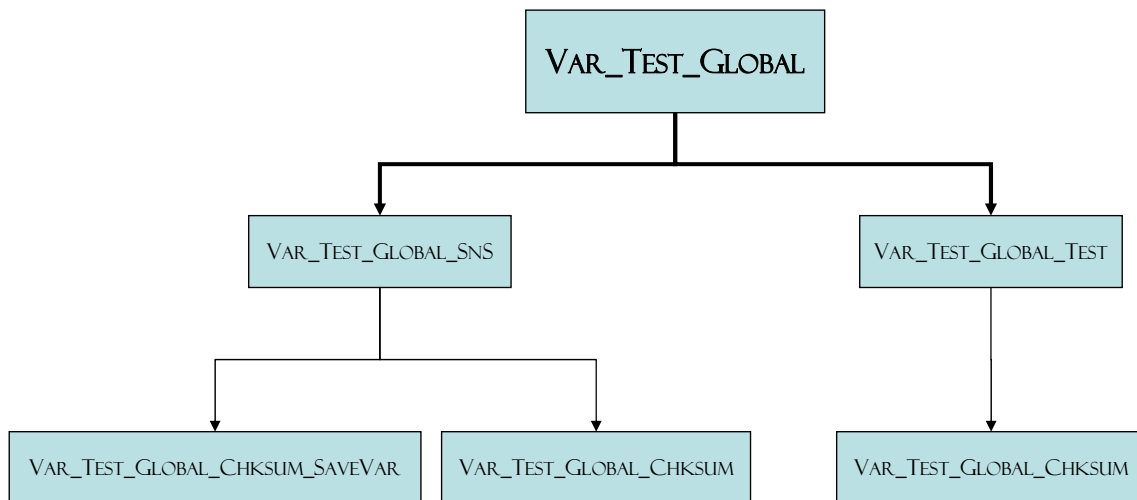
## 5.2 Test des Merkerbereichs mittels eines Arrays

Unter diesem Titel sind folgende Testbausteine zusammengefasst:

- VAR\_TEST\_GLOBAL
- VAR\_TEST\_GLOBAL\_CHKSUM
- VAR\_TEST\_GLOBAL\_CHKSUM\_SAVEVAR
- VAR\_TEST\_GLOBAL\_SNS
- VAR\_TEST\_GLOBAL\_TEST

Der Test VAR\_TEST\_GLOBALS stammt aus der Testsammlung in Kapitel 5.1.

Die Aufrufhierarchie gliedert sich wie folgt:



Das bedeutet, dass der Nutzer den Baustein VAR\_TEST\_GLOBAL aufruft, welcher die Bausteine VAR\_TEST\_GLOBAL\_SNS und VAR\_TEST\_GLOBAL\_TEST aufruft. Diese nutzen beide den Baustein VAR\_TEST\_GLOBAL\_CHKSUM und VAR\_TEST\_GLOBAL\_SNS nutzt zusätzlich den Baustein VAR\_TEST\_GLOBAL\_CHKSUM\_SAVEVAR.

## 5.2.1 VAR\_TEST\_GLOBAL

Variablen:

VAR\_INPUT

Feldbreite :DWORD;

Breite des vorbelegten Merkerbereichs

AnzahlTestsproDurchlauf :WORD;

Anzahl der Variablen die pro Zyklus getestet werden

Testbreite :DWORD;

Breite der Signatur für die Suche nach Übertragungsfehlern

END\_VAR

VAR\_OUTPUT

finished :BOOL;

Gibt zurück, ob ein kompletter Durchlauf abgeschlossen wurde. Kann z. B. genutzt werden, wenn eine komplette Prüfung des Speichers vorgenommen werden soll.

END\_VAR

VAR

ThirdStart : BOOL := TRUE;

Firststart : BOOL := TRUE;

AktuelleVar : DWORD;

WordLauf1 : DWORD;

WordLauf2 : DWORD;

WordLauf2Hilf : DWORD;

T\_Start : TIME := T#0ms;

T\_Start\_help : TIME := T#0ms;

T\_Cycle : TIME := T#0ms;

Test\_SnS : Var\_Test\_Global\_SnS;

Test\_Var\_Global\_Test:

Var\_Test\_Global\_Test;

END\_VAR

Funktion:

LD	FALSE	
ST	finished	
LD	Testbreite	(*Durch Runden kann der Anwender nun Signaturbreite mit oder ohne TestVariable eingeben.*)
SHR	1	(*DIV 2*)
ST	Testbreite	
LD	ThirdStart	(*Prüfen, ob der aktuelle Durchgang mindestens der dritte ist.*)
JMPCN	Run	
LD	Firststart	(*Prüfen, ob der aktuelle Durchgang der Erste oder der Zweite ist.*)
JMPCN	EingangsPruefung	
LD	FALSE	
ST	Firststart	(*Marker setzen: Erster Durchgang geschehen.*)
TIME		(*Zeitnahme: T_Cycle enthält die Zeit eines Durchlaufs*)
ST	T_Start_help	
SUB	T_Start	
ST	T_Cycle	
LD	T_Start_help	
ST	T_Start	(*Ende Zeitnahme*)
LD	Testbreite	
ST	WordLauf1	(*LaufVariable auf Testbreite vorbelegen*)
LD	0	

ST	AktuelleVar	(*Adresse wird auf erste freie Speicherstelle gesetzt.*)
EingangsPruefung:		(*Bei jedem ersten Durchlauf pro Testzyklus aufgerufen - Asynchrone Prüfung am Anfang des Speichers, um Pointerüberlauf nach vorne zu verhindern*)
LD	WordLauf1	(*Anzahl der noch zu prüfenden DWords*)
EQ	Testbreite	
JMPCN	OberePruefung	(*Bei Abweichung läuft obere Prüfung, bei gleichheit Prüfung der festgelegten Variablen am Speicheranfang (SaveVar, SignatureVar, etc.)*)
CAL	Test_SnS (feldbreiteoben:= Testbreite)	
LD	WordLauf1	(*WordLauf1 verringern. Ab hier Prüfung der freien Variablen.*)
SUB	1	
ST	WordLauf1	
LD	1	
ST	AktuelleVar	
RET		
OberePruefung:		(* Asynchrone Prüfung: Die zur Signatur verwendeten Variablen sind unten weniger als oben, da man hier am Speicheranfang ist.*)

LD	WordLauf1	(*WordLauf2 auf das kleinere von WordLauf1 (Verbleibende Starttests) und AnzahlTestsproDurchlauf setzen.*)
ST	WordLauf2	
GT	AnzahlTestsproDurchlauf	
JMPCN	Weiter1	
LD	AnzahlTestsproDurchlauf	
ST	WordLauf2	
Weiter1:		
Schleife1:		(*In dieser Schleife wird der Anfang des Speichers getestet.*)
LD	Testbreite	(*WordLauf2Hilf wird von 0 auf Testbreite hochgezählt.*)
SUB	1	
SUB	WordLauf1	
ST	WordLauf2Hilf	
CAL	Test_Var_Global_Test	(TestVariable:= AktuelleVar, feldbreiteunten:=WordLauf2Hilf , feldbreiteoben:=Testbreite )
LD	AktuelleVar	
ADD	1	
ST	AktuelleVar	
LD	WordLauf2	(*Prüfen, ob aktueller Durchgang beendet ist.*)
EQ	0	
JMPC	EndeSchleife1	
LD	WordLauf2	(*Wordlauf 1 und 2 decrementieren*)
SUB	1	

ST	WordLauf2	
LD	WordLauf1	
SUB	1	
ST	WordLauf1	
JMP	Schleife1	
EndeSchleife1:		
LD	WordLauf1	(*Prüfen ob die Prüfung des Anfangssegments fertig ist.*)
EQ	0	
RETCN		(*Falls Nein, Rücksprung und Weiterprüfen beim nächsten Durchgang*)
EndeOberePruefung:		
LD	FALSE	(*Falls Ja, ThirdStart auf false setzen, wodurch im nächsten Durchgang der mittlere bis hintere Speicherprüfzyklus aufgerufen wird.*)
ST	ThirdStart	
RET		
Run:		(*Mittlere oder hintere Speicherprüfung*)
LD	AnzahlTestsproDurchlauf	
ST	WordLauf1	
		(*Hier kommt eine Abfrage, ob das Speicherende erreicht ist.*)
LD	AktuelleVar	(*Aktuelle Adresse laden.*)

ADD	Testbreite	(*Die Breite der Signatur aufaddieren*)
ADD	AnzahlTestsproDurchlauf	(*Die Breite eines kompletten Durchlaufs aufaddieren*)
GT	Feldbreite	(*wenn das Ergebnis größer ist als das Ende des Speichers, Sprung in die Prüfroutine fürs Ende.*)
JMPC	EndeErreicht	(*Ende der Abfrage*)
Schleife2:		(*Speichertest in der Mitte vom Speicher*)
CAL	Test_Var_Global_Test(TestVariable:= feldbreiteunten:=Testbreite , feldbreiteoben:=Testbreite )	AktuelleVar,
LD	AktuelleVar	(*nächste Prüfstelle anwählen*)
ADD	1	
ST	AktuelleVar	
LD	WordLauf1	(*Schleifenzähler verringern*)
SUB	1	
ST	WordLauf1	
EQ	0	(*Prüfen ob Schleife beendet*)
JMPCN	Schleife2	
RET		
EndeErreicht:		
LD	Feldbreite	
SUB	AktuelleVar	
ST	WordLauf2	(*WordLauf2 enthält die Anzahl der noch zu prüfenden DWords*)

ST	WordLauf2Hilf	(*WordLauf2Hilf hat später die Anzahl der möglichen SignaturDwords hinter WordLauf2*)
GT	AnzahlTestsproDurchlauf	(*WordLauf2 wird, falls es größer als AnzahlTestsproDurchlauf ist auf AnzahlTestsproDurchlauf gesetzt.*)
JMPCN	Weiter2	
LD	AnzahlTestsproDurchlauf	
ST	WordLauf2	
Weiter2:		
Schleife3:		(*Sleife zur Prüfung des Speichers, falls der Pointer so nah am Ende ist, dass innerhalb des Durchlaufs ein Zugriff ausserhalb des Speichers geschehen könnte.*)
LD	WordLauf2	(*Wenn Ende Speicher, oder Anzahl an Durchläufen pro Test, erreicht: Schleife verlassen*)
EQ	0	
JMPC	EndeSchleife3	(*EndBedingung schon am Anfang geprüft. Daher ist dies eine WHILE Schleife.*)
LD	WordLauf2Hilf	(*WordLauf2Hilf decrementieren*)
SUB	1	
ST	WordLauf2Hilf	
LT	Testbreite	

JMPCN	SprungInSchleife3_1	(*Auswahl Synchroner oder Asynchroner Test. AsynchronerTest wenn die verbleibenden DWords für die Signatur weniger sind als von Testbreite verlangt.*)
CAL	Test_Var_Global_Test(TestVariable:=AktuelleVar, feldbreiteunten:=Testbreite , feldbreiteoben:=WordLauf2Hilf )	
JMP	SprungInSchleife3_2	(*Noch Auswahl Synchron-Asynchron*)
SprungInSchleife3_1:		
CAL	Test_Var_Global_Test(TestVariable:=AktuelleVar, feldbreiteunten:=Testbreite , feldbreiteoben:=Testbreite )	
SprungInSchleife3_2:		
LD	AktuelleVar	(*nächste Prüfstelle anwählen*)
ADD	1	
ST	AktuelleVar	
LD	WordLauf2	(*WordLauf2 decrementieren*)
SUB	1	
ST	WordLauf2	
JMP	Schleife3	
EndeSchleife3:		
LD	WordLauf2Hilf	
EQ	0	
RETCN		
LD	TRUE	(*Beim nächsten Start alles von Neuem*)
ST	ThirdStart	
ST	Firststart	
ST	finished	

## 5.2.2 VAR\_TEST\_GLOBAL\_CHKSUM

Variablen:

VAR_INPUT	
TestVariable:DWORD;	Speicherstelle, an der die zu testende Variable steht.
feldbreiteunten:DWORD;	Anzahl der Speicherstellen unterhalb die zur Signaturbildung mit einbezogen werden sollen
feldbreiteoben:DWORD;	Anzahl der Speicherstellen oberhalb die zur Signaturbildung mit einbezogen werden sollen
END_VAR	
VAR	
Schleifenende : DWORD;	
END_VAR	

Funktion:

LD	0
ST	SignatureVar
LD	feldbreiteunten
EQ	0
JMPC	Weiter1
LD	TestVariable
ST	Schleifenende
SUB	feldbreiteunten
ST	TestVariable
Schleife1:	
LD	TestFeld[TestVariable]
XOR	SignatureVar

ST	SignatureVar
LD	TestVariable
ADD	1
ST	TestVariable
EQ	Schleifenende
JMPCN	Schleife1
Weiter1:	
LD	TestVariable
ADD	1
ST	TestVariable
ADD	feldbreiteoben
ST	Schleifenende
LD	feldbreiteoben
EQ	0
RETC	
Schleife2:	
LD	TestFeld[TestVariable]
XOR	SignatureVar
ST	SignatureVar
LD	TestVariable
ADD	1
ST	TestVariable
EQ	Schleifenende
JMPCN	Schleife2

### 5.2.3 VAR\_TEST\_GLOBAL\_CHKSUM\_SAVEVAR

Variablen:

VAR_INPUT	
TestVariable:DWORD;	Speicherstelle, an der die zu testende Variable steht.
feldbreiteunten:DWORD;	Anzahl der Speicherstellen unterhalb die zur Signaturbildung mit einbezogen werden sollen
feldbreiteoben:DWORD;	Anzahl der Speicherstellen oberhalb die zur Signaturbildung mit einbezogen werden sollen
END_VAR	
VAR	
Schleifenende : DWORD;	
END_VAR	

Funktion:

LD	0
ST	SaveVar
LD	feldbreiteunten
EQ	0
JMPC	Weiter1
LD	TestVariable
ST	Schleifenende
SUB	feldbreiteunten
ST	TestVariable
Schleife1:	
LD	TestFeld[TestVariable]
XOR	SaveVar

ST	SaveVar
LD	TestVariable
ADD	1
ST	TestVariable
EQ	Schleifenende
JMPCN	Schleife1

Weiter1:

LD	TestVariable
ADD	1
ST	TestVariable
ADD	feldbreiteoben
ST	Schleifenende

LD	feldbreiteoben
EQ	0
RETC	

Schleife2:

LD	TestFeld[TestVariable]
XOR	SaveVar
ST	SaveVar
LD	TestVariable
ADD	1
ST	TestVariable
EQ	Schleifenende
JMPCN	Schleife2

## 5.2.4 VAR\_TEST\_GLOBAL\_SNS

Variablen:

```

VAR_INPUT

feldbreiteoben:DWORD;                Anzahl der Speicherstellen oberhalb die
                                      zur Signaturbildung mit einbezogen
                                      werden sollen

END_VAR

VAR

SignatureVarSave : DWORD;

Chksum: VarTestGlobal_Chksum;

Chksum_SaveVar:
VarTestGlobal_Chksum_SaveVar;

FB_Error:Error_FB;                    Fehlerbaustein aus Kapitel 1

END_VAR

```

Funktion:

```

JMP      Start

Error:

CAL      FB_Error

CalError:

JMP      CalError

Start:

CAL      Chksum(TestVariable:=0      ,      (*BEGIN Test SaveVar*)
          feldbreiteunten:=0        ,
          feldbreiteoben:= feldbreiteoben )

LD       SignatureVar

ST       SignatureVarSave

LD       1431655765                    (*0101 0101 0101 0101 0101
          0101 0101 0101(B)*)

ST       SaveVar

```

CAL	Chksum(TestVariable:=0 , feldbreiteunten:=0 , feldbreiteoben:=feldbreiteoben )	
LD	SignatureVarSave	
EQ	SignatureVar	
JMPCN	Error	
LD	SaveVar	
EQ	1431655765	(*0101 0101 0101 0101 0101 0101 0101 0101(B)*)
JMPCN	Error	
LD	2863311530	(*1010 1010 1010 1010 1010 1010 1010 1010(B)*)
ST	SaveVar	
CAL	Chksum(TestVariable:=0 , feldbreiteunten:=0 , feldbreiteoben:=feldbreiteoben )	
LD	SignatureVarSave	
EQ	SignatureVar	
JMPCN	Error	
LD	SaveVar	
EQ	2863311530	(*1010 1010 1010 1010 1010 1010 1010 1010(B)*)
JMPCN	Error	(*END Test SaveVar*)
CAL	Chksum_SaveVar(TestVariable:=0 , feldbreiteunten:=0 , feldbreiteoben:= feldbreiteoben )	(*BEGIN Test SignatureVar*)
LD	SaveVar	
ST	SignatureVarSave	

LD	1431655765	(*0101 0101 0101 0101 0101 0101 0101 0101(B)*)
ST	SignatureVar	
CAL	Chksum_SaveVar(TestVariable:=0 feldbreiteoben:=feldbreiteoben )	, feldbreiteunten:=0 ,
LD	SignatureVarSave	
EQ	SaveVar	
JMPCN	Error	
LD	SignatureVar	
EQ	1431655765	(*0101 0101 0101 0101 0101 0101 0101 0101(B)*)
JMPCN	Error	
LD	2863311530	(*1010 1010 1010 1010 1010 1010 1010 1010(B)*)
ST	SignatureVar	
CAL	Chksum_SaveVar(TestVariable:=0 feldbreiteoben:=feldbreiteoben )	, feldbreiteunten:=0 ,
LD	SignatureVarSave	
EQ	SaveVar	
JMPCN	Error	
LD	SignatureVar	
EQ	2863311530	(*1010 1010 1010 1010 1010 1010 1010 1010(B)*)
JMPCN	Error	(*END Test SignatureVar*)

## 5.2.5 VAR\_TEST\_GLOBAL\_TEST

Variablen:

VAR\_INPUT

TestVariable:DWORD;

Speicherstelle, an der die zu testende Variable steht.

feldbreiteunten:DWORD;

Anzahl der Speicherstellen unterhalb die zur Signaturbildung mit einbezogen werden sollen

feldbreiteoben:DWORD;

Anzahl der Speicherstellen oberhalb die zur Signaturbildung mit einbezogen werden sollen

END\_VAR

VAR

SignatureVarSave : DWORD;

Chksum: VarTestGlobal\_Chksum;

FB\_Error:Error\_FB;

Fehlerbaustein aus Kapitel 1

END\_VAR

Funktion:

JMP Start

Error:

CAL FB\_Error

CalError:

JMP CalError

Start:

LD TestFeld[TestVariable] (\*BEGIN Test TestVar5\*)

ST SaveVar

CAL Chksum(TestVariable:=TestVariable , feldbreiteunten:=feldbreiteunten , feldbreiteoben:=feldbreiteoben )

LD SignatureVar

ST	SignatureVarSave	
LD	1431655765	(*0101 0101 0101 0101 0101 0101 0101 0101(B)*)
ST	TestFeld[TestVariable]	
CAL	Chksum(TestVariable:=TestVariable , feldbreiteunten:=feldbreiteunten , feldbreiteoben:=feldbreiteoben )	
LD	SignatureVar	
EQ	SignatureVarSave	
JMPCN	Error	
LD	TestFeld[TestVariable]	
EQ	1431655765	(*0101 0101 0101 0101 0101 0101 0101 0101(B)*)
JMPCN	Error	
LD	2863311530	(*1010 1010 1010 1010 1010 1010 1010 1010(B)*)
ST	TestFeld[TestVariable]	
CAL	Chksum(TestVariable:=TestVariable , feldbreiteunten:=feldbreiteunten , feldbreiteoben:=feldbreiteoben )	
LD	SignatureVar	
EQ	SignatureVarSave	
JMPCN	Error	
LD	TestFeld[TestVariable]	
EQ	2863311530	(*1010 1010 1010 1010 1010 1010 1010 1010(B)*)
JMPCN	Error	
LD	SaveVar	
ST	TestFeld[TestVariable]	(*END Test TestVar5*)

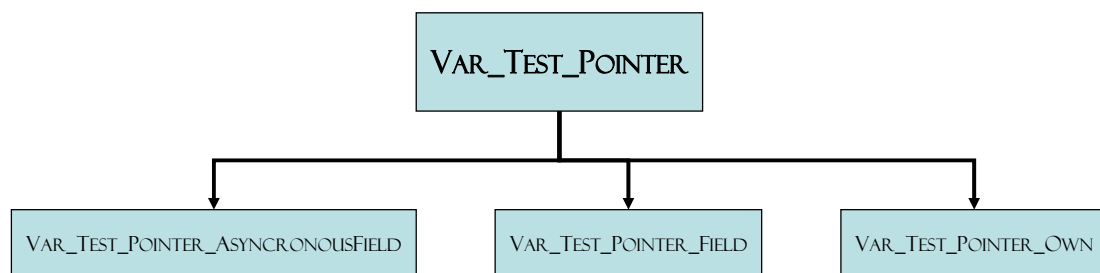
### 5.3 Test des Speichers mittels eines Zeigers

Unter diesem Titel sind folgende Testbausteine zusammengefasst:

- VAR\_TEST\_POINTER
- VAR\_TEST\_POINTER\_ASYNCHONOUSFIELD
- VAR\_TEST\_POINTER\_FIELD
- VAR\_TEST\_POINTER\_OWN

Der Test VAR\_TEST\_GLOBALS stammt aus der Testsammlung in Kapitel 5.1.

Die Aufrufhierarchie gliedert sich wie folgt:



### 5.3.1 VAR\_TEST\_POINTER

Variablen:

VAR\_INPUT

Anfangsstelle :POINTER TO DWORD;

Zeiger auf die erste zu prüfende Stelle

Feldbreite :DWORD;

Breite des Speicherbereichs der getestet werden soll

AnzahlTestsproDurchlauf :WORD;

Anzahl der Variablen die pro Zyklus getestet werden

Testbreite :DWORD;

Breite der Signatur für die Suche nach Übertragungsfehlern

END\_VAR

VAR\_OUTPUT

finished :BOOL;

Gibt zurück, ob ein kompletter Durchlauf abgeschlossen wurde. Kann z. B. genutzt werden, wenn eine komplette Prüfung des Speichers vorgenommen werden soll.

END\_VAR

VAR

SaveVar AT %MD0: DWORD;

SignatureVar AT %MD1 : DWORD;

SignatureSaveVar AT %MD2 : DWORD;

TestPointer AT %MD3 : POINTER TO  
DWORD;

HilfPointer : POINTER TO DWORD;

ThirdStart : BOOL := TRUE;

Firststart : BOOL := TRUE;

WordLauf1 AT %MD4: DWORD;

WordLauf2 : DWORD;

WordLauf2Hilf : DWORD;

T\_Start : TIME := T#0ms;

```

T_Cycle : TIME := T#0ms;

Test_Var_Pointer_Own:
Var_Test_Pointer_Own;

Test_Var_Pointer_Field:
Var_Test_Pointer_Field;

Test_Var_Pointer_AsyncronousField:
Var_Test_Pointer_AsyncronousField;

END_VAR

```

Funktion: (\* Diese Funktion benutzt MD0..5 für interne Zähler und Speicher. Diese dürfen nicht normal mitgetestet werden, sondern werden separat in Var\_Test\_Pointer\_Own getestet.\*)

LD	FALSE	
ST	finished	
LD	Testbreite	(*Durch Runden kann der Anwender nun die Signaturbreite mit oder ohne TestVariable eingeben.*)
SHR	1	(*DIV 2*)
ST	Testbreite	
LD	ThirdStart	(*Prüfen, ob der aktuelle Durchgang mindestens der dritte ist.*)
JMPCN	Run	
LD	Firststart	(*Prüfen, ob der aktuelle Durchgang der Erste oder der Zweite ist.*)
JMPCN	EingangsPruefung	
LD	FALSE	
ST	Firststart	(*Marker setzen: Erster Durchgang geschehen.*)
TIME		(*Zeitnahme: T_Cycle enthält die Zeit eines Durchlaufs*)
SUB	T_Start	

ST	T_Cycle	
LD	Testbreite	
ST	WordLauf1	(*LaufVariable auf Testbreite vorbelegen*)
TIME		
ST	T_Start	
LD	Anfangsstelle	
ST	TestPointer	(*Adresse wird auf erste freie Speicherstelle gesetzt.*)
EingangsPruefung:		(*Bei jedem ersten Durchlauf pro Testzyklus aufgerufen - Asynchrone Prüfung am Anfang des Speichers, um Pointerüberlauf nach vorne zu verhindern*)
LD	WordLauf1	(*Anzahl der noch zu prüfenden DWords*)
EQ	Testbreite	
JMPCN		(*Bei Abweichung läuft obere Prüfung, bei gleichheit Prüfung der festgelegten Variablen am Speicheranfang (SaveVar, SignatureVar, etc.)*)
CAL	Test_Var_Pointer_Own	
LD	WordLauf1	(*WordLauf1 verringern. Ab hier Prüfung der freien Variablen.*)
SUB	1	
ST	WordLauf1	
EQ	0	(*Wenn WordLauf1 hier Null ist, ist keine Asynchrone Prüfung mehr nötig.*)

JMPC	EndeOberePruefung	
RET		
OberePruefung:		(* Asynchrone Prüfung: Die zur Signatur verwendeten Variablen sind unten weniger als oben, da man hier am Speicheranfang ist.*)
LD	WordLauf1	(*WordLauf2 auf das kleinere von WordLauf1 (Verbleibende Starttests) und AnzahlTestsproDurchlauf setzen.*)
ST	WordLauf2	
GT	AnzahlTestsproDurchlauf	
JMPCN	Weiter1	
LD	AnzahlTestsproDurchlauf	
ST	WordLauf2	
Weiter1:		
Schleife1:		(*In dieser Schleife wird der Anfang des Speichers getestet.*)
LD	Testbreite	(*WordLauf2Hilf wird von 0 auf Testbreite hochgezählt.*)
SUB	1	
SUB	WordLauf1	
ST	WordLauf2Hilf	
CAL	Test_Var_Pointer_AsynchronousField	(TestbreiteOben:=Testbreite, TestbreiteUnten:=WordLauf2Hilf)
LD	TestPointer	(*Testpointer nach hinten verschieben*)
ADD	4	

ST	TestPointer	
LD	WordLauf2	(*Prüfen, ob aktueller Durchgang beendet ist.*)
EQ	0	
JMPC	EndeSchleife1	
LD	WordLauf2	(*Wordlauf 1 und 2 decrementieren*)
SUB	1	
ST	WordLauf2	
LD	WordLauf1	
SUB	1	
ST	WordLauf1	
JMP	Schleife1	
EndeSchleife1:		
LD	WordLauf1	(*Prüfen ob die Prüfung des Anfangssegments fertig ist.*)
EQ	0	
RETCN		(*Falls Nein, Rücksprung und Weiterprüfen beim nächsten Durchgang*)
EndeOberePruefung:		
LD	FALSE	(*Falls Ja, ThirdStart auf false setzen, wodurch im nächsten Durchgang der mittlere bis hintere Speicherprüfzyklus aufgerufen wird.*)
ST	ThirdStart	
RET		

Run:		(*Mittlere oder hintere Speicherprüfung*)
LD	AnzahlTestsproDurchlauf	
ST	WordLauf1	
LD	Anfangsstelle	
ADD	Feldbreite	(*16384 ist die Größe des virtuellen Speichers in CoDeSys*)
ST	HilfPointer	(*Erste Speicherstelle wird nach HilfPointer geschrieben*)
(*Hier kommt eine Abfrage, ob das Speicherende erreicht ist.*)		
LD	TestPointer^	(*Aktuelle Adresse laden.*)
ADR		
ADD	(Testbreite	(*Die Breite der Signatur aufaddieren*)
MUL	4	
)		
ADD	(AnzahlTestsproDurchlauf	(*Die Breite eines kompletten Durchlaufs aufaddieren*)
MUL	4	
)		
GT	HilfPointer	(*wenn das Ergebnis größer ist als das Ende des Speichers, Sprung in die Prüfroutine fürs Ende.*)
JMPC	EndeErreicht	
(*Ende der Abfrage*)		
Schleife2:		(*Speichertest in der Mitte vom Speicher*)
CAL	Test_Var_Pointer_Field (Testbreite:=Testbreite)	

LD	TestPointer	(*nächste Prüfstelle anwählen*)
ADD	4	
ST	TestPointer	
LD	WordLauf1	(*Schleifenzähler verringern*)
SUB	1	
ST	WordLauf1	
EQ	0	(*Prüfen ob Schleife beendet*)
JMPCN	Schleife2	
RET		
EndeErreicht:		
LD	Anfangsstelle^	
ADR		
ST	WordLauf2Hilf	(*WordLauf2Hilf enthält den Wert des Anfangspointers... nur den Wert, mit Pointern kann man nicht rechnen*)
LD	TestPointer^	
ADR		
SUB	WordLauf2Hilf	
ST	WordLauf2	(*WordLauf2 enthält die Differenz zwischen dem Aktuellen Pointer und dem Anfangspointer*)
LD	Feldbreite	
SUB	WordLauf2	(*WordLauf2 enthält die Anzahl der noch zu Prüfenden Bytes*)
DIV	4	
ST	WordLauf2	(*WordLauf2 enthält die Anzahl der noch zu prüfenden DWords*)

ST	WordLauf2Hilf	(*WordLauf2Hilf hat später die Anzahl der möglichen SignaturDwords hinter WordLauf2*)
GT	AnzahlTestsproDurchlauf	(*WordLauf2 wird, falls es größer als AnzahlTestsproDurchlauf ist auf AnzahlTestsproDurchlauf gesetzt.*)
JMPCN	Weiter2	
LD	AnzahlTestsproDurchlauf	
ST	WordLauf2	
Weiter2:		
Schleife3:		(*Sleife zur Prüfung des Speichers, falls der Pointer so nah am Ende ist, dass innerhalb des Durchlaufs ein Zugriff ausserhalb des Speichers geschehen könnte.*)
LD	WordLauf2	(*Wenn Ende Speicher, oder Anzahl an Durchläufen pro Test, erreicht: Schleife verlassen*)
EQ	0	
JMPC	EndeSchleife3	(*EndBedingung schon am Anfang geprüft. Daher ist dies eine WHILE Schleife.*)
LD	WordLauf2Hilf	(*WordLauf2Hilf decrementieren*)
SUB	1	
ST	WordLauf2Hilf	
LT	Testbreite	
JMPCN	SprungInSchleife3_1	(*Auswahl Synchroner oder Asynchroner Test. AsynchronerTest wenn die verbleibenden DWords für die Signatur weniger sind als von Testbreite verlangt.*)

CAL	Test_Var_Pointer_AsynchronousField (TestbreiteOben:=WordLauf2Hilf, TestbreiteUnten:=Testbreite)	
JMP	SprungInSchleife3_2	(*Noch Auswahl Synchron-Asynchron*)
SprungInSchleife3_1:		
CAL	Test_Var_Pointer_Field (Testbreite:=Testbreite)	
SprungInSchleife3_2:		
LD	TestPointer	(*TestPointer um ein DWord nach hinten setzen*)
ADD	4	
ST	TestPointer	
LD	WordLauf2	(*WordLauf2 decrementieren*)
SUB	1	
ST	WordLauf2	
JMP	Schleife3	
EndeSchleife3:		
LD	WordLauf2Hilf	
EQ	0	
RETCN		
LD	TRUE	(*Beim nächsten Start alles von Neuem*)
ST	ThirdStart	
ST	Firststart	
ST	finished	

### 5.3.2 VAR\_TEST\_POINTER\_ASYNCHONOUSFIELD

Variablen:

VAR\_INPUT

TestbreiteUnten :DWORD;

Breite der Signatur unterhalb der Testvariablen für die Suche nach Übertragungsfehlern

TestbreiteOben :DWORD;

Breite der Signatur oberhalb der Testvariablen für die Suche nach Übertragungsfehlern

END\_VAR

VAR

SaveVar AT %MD0: DWORD;

SignatureVar AT %MD1 : DWORD;

SignatureSaveVar AT %MD2 : DWORD;

TestPointer AT %MD3 : POINTER TO  
DWORD;

ByteLauf AT %MD5 : DWORD;

TestVarPointer: POINTER TO DWORD;

TestSignaturUntenPointer: POINTER TO  
DWORD;

TestSignaturObenPointer: POINTER TO  
DWORD;

FB\_Error:Error\_FB;

Fehlerbaustein aus Kapitel 1

END\_VAR

Funktion:

```
LD      0
ST      SignatureVar
ST      SignatureSaveVar
JMP     Start
Error:
```

CAL	FB_Error	
CalError:		
JMP	CalError	
Start:		
LD	TestbreiteUnten	(*Schleifendurchläufe = Halbe Anzahl der mit XOR zu verknüpfenden Variablen*)
ST	ByteLauf	
LD	TestPointer^	(*Inhalt der testvariablen sichern*)
ST	SaveVar	
LD	TestPointer	(*TestPointer sichern*)
ST	TestVarPointer	
ST	TestSignaturObenPointer	
LD	TestbreiteOben	
EQ	0	
JMPC	Jump1	(*Wenn Oben nicht getestet wird, wird auch kein Pointer hoch gesetzt. Schutz vor falschem Speicherzugriff.*)
LD	TestPointer	
ADD	4	
ST	TestSignaturObenPointer	(*Begin Oberes Segment sichern*)
Jump1:		
LD	TestPointer^	(*Direkt vom Pointer ein WORD abziehen geht nicht. Darum hier der Umweg über ADR*)
ADR		
SUB	( TestbreiteUnten	(*4 mal Testbreite abziehen -> Anfang des unteren zu prüfenden Segments*)
MUL	4	
)		

ST	TestSignaturUntenPointer	(*Begin Unteres Segment sichern*)
ST	TestPointer	(*Lauf Pointer auf unteres Segment setzen*)

Schleife1:

LD	ByteLauf
EQ	0
JMPC	Ausgang1
LD	TestPointer^
XOR	SignatureVar
ST	SignatureVar
LD	TestPointer
ADD	4
ST	TestPointer
LD	ByteLauf
SUB	1
ST	ByteLauf
JMP	Schleife1

Ausgang1:

LD	TestSignaturObenPointer
ST	TestPointer
LD	TestbreiteOben
ST	ByteLauf

Schleife2:

LD	ByteLauf
EQ	0

JMPC	Ausgang2	
LD	TestPointer^	
XOR	SignatureVar	
ST	SignatureVar	
LD	TestPointer	
ADD	4	
ST	TestPointer	
LD	ByteLauf	
SUB	1	
ST	ByteLauf	
JMP	Schleife2	
Ausgang2:		
LD	TestVarPointer	
ST	TestPointer	
LD	1431655765	(*0101 0101 0101 0101 0101 0101 0101 0101 0101(B)*)
ST	TestPointer^	
LD	TestSignaturUntenPointer	
ST	TestPointer	
LD	TestbreiteUnten	
ST	ByteLauf	
Schleife3:		
LD	ByteLauf	
EQ	0	
JMPC	Ausgang3	

LD	TestPointer^
XOR	SignatureSaveVar
ST	SignatureSaveVar
LD	TestPointer
ADD	4
ST	TestPointer
LD	ByteLauf
SUB	1
ST	ByteLauf
JMP	Schleife3
Ausgang3:	
LD	TestSignaturObenPointer
ST	TestPointer
LD	TestbreiteOben
ST	ByteLauf
Schleife4:	
LD	ByteLauf
EQ	0
JMPC	Ausgang4
LD	TestPointer^
XOR	SignatureSaveVar
ST	SignatureSaveVar
LD	TestPointer
ADD	4
ST	TestPointer
LD	ByteLauf

```
SUB      1
ST      ByteLauf
JMP     Schleife4
Ausgang4:

LD      TestVarPointer
ST      TestPointer
LD      SignatureSaveVar
EQ      SignatureVar
JMPCN   Error
LD      TestPointer^
EQ      1431655765      (*0101 0101 0101 0101 0101 0101 0101
                        0101(B)*)
JMPCN   Error

(*2ter Durchgang!!*)

LD      0
ST      SignatureSaveVar
LD      2863311530     (*1010 1010 1010 1010 1010 1010 1010
                        1010(B)*)
ST      TestPointer^

LD      TestSignaturUntenPointer
ST      TestPointer
LD      TestbreiteUnten
ST      ByteLauf

Schleife5:
```

LD	ByteLauf
EQ	0
JMPC	Ausgang5
LD	TestPointer^
XOR	SignatureSaveVar
ST	SignatureSaveVar
LD	TestPointer
ADD	4
ST	TestPointer
LD	ByteLauf
SUB	1
ST	ByteLauf
JMP	Schleife5
Ausgang5:	
LD	TestSignaturObenPointer
ST	TestPointer
LD	TestbreiteOben
ST	ByteLauf
Schleife6:	
LD	ByteLauf
EQ	0
JMPC	Ausgang6
LD	TestPointer^
XOR	SignatureSaveVar
ST	SignatureSaveVar
LD	TestPointer

ADD	4	
ST	TestPointer	
LD	ByteLauf	
SUB	1	
ST	ByteLauf	
JMP	Schleife6	
Ausgang6:		
LD	TestVarPointer	
ST	TestPointer	
LD	SignatureSaveVar	
EQ	SignatureVar	
JMPCN	Error	
LD	TestPointer^	
EQ	2863311530	(*1010 1010 1010 1010 1010 1010 1010 1010 1010(B)*)
JMPCN	Error	
LD	SaveVar	
ST	TestVarPointer^	

### 5.3.3 VAR\_TEST\_POINTER\_FIELD

Variablen:

VAR_INPUT	
Testbreite :DWORD;	Breite der Signatur für die Suche nach Übertragungsfehlern
END_VAR	
VAR	
SaveVar AT %MD0: DWORD;	
SignatureVar AT %MD1 : DWORD;	
SignatureSaveVar AT %MD2 : DWORD;	
TestPointer AT %MD3 : POINTER TO DWORD;	
ByteLauf AT %MD5 : DWORD;	
TestVarPointer: POINTER TO DWORD;	
TestSignaturUntenPointer: POINTER TO DWORD;	
TestSignaturObenPointer: POINTER TO DWORD;	
FB_Error:Error_FB;	Fehlerbaustein aus Kapitel 1
END_VAR	

Funktion:

LD	0
ST	SignatureVar
ST	SignatureSaveVar
JMP	Start
Error:	
CAL	FB_Error
CalError:	
JMP	CalError

Start:

LD            Testbreite            (\*Schleifendurchläufe = Halbe Anzahl der mit XOR zu verknüpfenden Variablen\*)

ST            ByteLauf

LD            TestPointer^            (\*Inhalt der testvariablen sichern\*)

ST            SaveVar

LD            TestPointer            (\*TestPointer sichern\*)

ST            TestVarPointer

ADD          4

ST            TestSignaturObenPointer            (\*Begin Oberes Segment sichern\*)

LD            TestPointer^            (\*Direkt vom Pointer ein WORD abziehen geht nicht. Darum hier der Umweg über ADR\*)

ADR

SUB          ( Testbreite            (\*4 mal Testbreite abziehen -> Anfang des unteren zu prüfenden Segments\*)

MUL          4

)

ST            TestSignaturUntenPointer            (\*Begin Unteres Segment sichern\*)

ST            TestPointer            (\*Lauf Pointer auf unteres Segment setzen\*)

Schleife1:

LD            TestPointer^

XOR          SignatureVar

ST            SignatureVar

LD            TestPointer

ADD          4

ST            TestPointer

LD	ByteLauf	
SUB	1	
ST	ByteLauf	
EQ	0	
JMPCN	Schleife1	
LD	TestSignaturObenPointer	
ST	TestPointer	
LD	Testbreite	
ST	ByteLauf	
Schleife2:		
LD	TestPointer^	
XOR	SignatureVar	
ST	SignatureVar	
LD	TestPointer	
ADD	4	
ST	TestPointer	
LD	ByteLauf	
SUB	1	
ST	ByteLauf	
EQ	0	
JMPCN	Schleife2	
LD	TestVarPointer	
ST	TestPointer	
LD	1431655765	(*0101 0101 0101 0101 0101 0101 0101 0101(B)*)

ST	TestPointer^
LD	TestSignaturUntenPointer
ST	TestPointer
LD	Testbreite
ST	ByteLauf

Schleife3:

LD	TestPointer^
XOR	SignatureSaveVar
ST	SignatureSaveVar
LD	TestPointer
ADD	4
ST	TestPointer
LD	ByteLauf
SUB	1
ST	ByteLauf
EQ	0
JMPCN	Schleife3

LD	TestSignaturObenPointer
ST	TestPointer
LD	Testbreite
ST	ByteLauf

Schleife4:

LD	TestPointer^
XOR	SignatureSaveVar

ST	SignatureSaveVar	
LD	TestPointer	
ADD	4	
ST	TestPointer	
LD	ByteLauf	
SUB	1	
ST	ByteLauf	
EQ	0	
JMPCN	Schleife4	
LD	TestVarPointer	
ST	TestPointer	
LD	SignatureSaveVar	
EQ	SignatureVar	
JMPCN	Error	
LD	TestPointer^	
EQ	1431655765	(*0101 0101 0101 0101 0101 0101 0101 0101(B)*)
JMPCN	Error	
(*2ter Durchgang!!*)		
LD	0	
ST	SignatureSaveVar	
LD	2863311530	(*1010 1010 1010 1010 1010 1010 1010 1010(B)*)
ST	TestPointer^	
LD	TestSignaturUntenPointer	

ST	TestPointer
LD	Testbreite
ST	ByteLauf
Schleife5:	
LD	TestPointer^
XOR	SignatureSaveVar
ST	SignatureSaveVar
LD	TestPointer
ADD	4
ST	TestPointer
LD	ByteLauf
SUB	1
ST	ByteLauf
EQ	0
JMPCN	Schleife5
LD	TestSignaturObenPointer
ST	TestPointer
LD	Testbreite
ST	ByteLauf
Schleife6:	
LD	TestPointer^
XOR	SignatureSaveVar
ST	SignatureSaveVar
LD	TestPointer
ADD	4

ST	TestPointer	
LD	ByteLauf	
SUB	1	
ST	ByteLauf	
EQ	0	
JMPCN	Schleife6	
LD	TestVarPointer	
ST	TestPointer	
LD	SignatureSaveVar	
EQ	SignatureVar	
JMPCN	Error	
LD	TestPointer^	
EQ	2863311530	(*1010 1010 1010 1010 1010 1010 1010 1010 1010(B)*)
JMPCN	Error	
LD	SaveVar	
ST	TestPointer^	

### 5.3.4 VAR\_TEST\_POINTER\_OWN

Variablen:

VAR

SaveVar AT %MD0: DWORD;

SignatureVar AT %MD1 : DWORD;

TestVar1 AT %MD2 : DWORD;

TestVar2 AT %MD3 : DWORD;

TestVar3 AT %MD4 : DWORD;

TestVar4 AT %MD5 : DWORD;

TestVar5 AT %MD6 : DWORD;

TestVar6 AT %MD7 : DWORD;

TestVar7 AT %MD8 : DWORD;

TestVar8 AT %MD9 : DWORD;

TestVar9 AT %MD10 : DWORD;

FB\_Error:Error\_FB;

Fehlerbaustein aus Kapitel 1

END\_VAR

Funktion:

JMP           Start

Error:

CAL           FB\_Error

CalError:

JMP           CalError

Start:

LD            TestVar1           (\*BEGIN Test SaveVar\*)

XOR           TestVar2

XOR           TestVar3

XOR           TestVar4

XOR           TestVar5

XOR	TestVar6	
XOR	TestVar7	
XOR	TestVar8	
XOR	TestVar9	
ST	SignatureVar	
LD	1431655765	(*0101 0101 0101 0101 0101 0101 0101 0101(B)*)
ST	SaveVar	
LD	TestVar1	
XOR	TestVar2	
XOR	TestVar3	
XOR	TestVar4	
XOR	TestVar5	
XOR	TestVar6	
XOR	TestVar7	
XOR	TestVar8	
XOR	TestVar9	
EQ	SignatureVar	
JMPCN	Error	
LD	SaveVar	
EQ	1431655765	(*0101 0101 0101 0101 0101 0101 0101 0101(B)*)
JMPCN	Error	
LD	2863311530	(*1010 1010 1010 1010 1010 1010 1010 1010(B)*)
ST	SaveVar	
LD	TestVar1	
XOR	TestVar2	
XOR	TestVar3	

XOR	TestVar4	
XOR	TestVar5	
XOR	TestVar6	
XOR	TestVar7	
XOR	TestVar8	
XOR	TestVar9	
EQ	SignatureVar	
JMPCN	Error	
LD	SaveVar	
EQ	2863311530	(*1010 1010 1010 1010 1010 1010 1010 1010(B)*)
JMPCN	Error	(*END Test SaveVar*)
LD	TestVar1	(*BEGIN Test SignatureVar*)
XOR	TestVar2	
XOR	TestVar3	
XOR	TestVar4	
XOR	TestVar5	
XOR	TestVar6	
XOR	TestVar7	
XOR	TestVar8	
XOR	TestVar9	
ST	SaveVar	
LD	1431655765	(*0101 0101 0101 0101 0101 0101 0101 0101(B)*)
ST	SignatureVar	
LD	TestVar1	
XOR	TestVar2	
XOR	TestVar3	

XOR	TestVar4	
XOR	TestVar5	
XOR	TestVar6	
XOR	TestVar7	
XOR	TestVar8	
XOR	TestVar9	
EQ	SaveVar	
JMPCN	Error	
LD	SignatureVar	
EQ	1431655765	(*0101 0101 0101 0101 0101 0101 0101 0101(B)*)
JMPCN	Error	
LD	2863311530	(*1010 1010 1010 1010 1010 1010 1010 1010(B)*)
ST	SignatureVar	
LD	TestVar1	
XOR	TestVar2	
XOR	TestVar3	
XOR	TestVar4	
XOR	TestVar5	
XOR	TestVar6	
XOR	TestVar7	
XOR	TestVar8	
XOR	TestVar9	
EQ	SaveVar	
JMPCN	Error	
LD	SignatureVar	
EQ	2863311530	(*1010 1010 1010 1010 1010 1010 1010 1010(B)*)
JMPCN	Error	(*END Test SignatureVar*)

LD	TestVar1	(*BEGIN Test TestVar1*)
ST	SaveVar	
LD	TestVar2	
XOR	TestVar3	
XOR	TestVar4	
XOR	TestVar5	
XOR	TestVar6	
XOR	TestVar7	
XOR	TestVar8	
XOR	TestVar9	
ST	SignatureVar	
LD	1431655765	(*0101 0101 0101 0101 0101 0101 0101 0101(B)*)
ST	TestVar1	
LD	TestVar2	
XOR	TestVar3	
XOR	TestVar4	
XOR	TestVar5	
XOR	TestVar6	
XOR	TestVar7	
XOR	TestVar8	
XOR	TestVar9	
EQ	SignatureVar	
JMPCN	Error	
LD	TestVar1	
EQ	1431655765	(*0101 0101 0101 0101 0101 0101 0101 0101(B)*)
JMPCN	Error	

LD	2863311530	(*1010 1010 1010 1010 1010 1010 1010 1010(B)*)
ST	TestVar1	
LD	TestVar2	
XOR	TestVar3	
XOR	TestVar4	
XOR	TestVar5	
XOR	TestVar6	
XOR	TestVar7	
XOR	TestVar8	
XOR	TestVar9	
EQ	SignatureVar	
JMPCN	Error	
LD	TestVar1	
EQ	2863311530	(*1010 1010 1010 1010 1010 1010 1010 1010(B)*)
JMPCN	Error	
LD	SaveVar	
ST	TestVar1	(*END Test TestVar1*)
LD	TestVar2	(*BEGIN Test TestVar2*)
ST	SaveVar	
LD	TestVar1	
XOR	TestVar3	
XOR	TestVar4	
XOR	TestVar5	
XOR	TestVar6	
XOR	TestVar7	
XOR	TestVar8	

XOR	TestVar9	
ST	SignatureVar	
LD	1431655765	(*0101 0101 0101 0101 0101 0101 0101 0101(B)*)
ST	TestVar2	
LD	TestVar1	
XOR	TestVar3	
XOR	TestVar4	
XOR	TestVar5	
XOR	TestVar6	
XOR	TestVar7	
XOR	TestVar8	
XOR	TestVar9	
EQ	SignatureVar	
JMPCN	Error	
LD	TestVar2	
EQ	1431655765	(*0101 0101 0101 0101 0101 0101 0101 0101(B)*)
JMPCN	Error	
LD	2863311530	(*1010 1010 1010 1010 1010 1010 1010 1010(B)*)
ST	TestVar2	
LD	TestVar1	
XOR	TestVar3	
XOR	TestVar4	
XOR	TestVar5	
XOR	TestVar6	
XOR	TestVar7	
XOR	TestVar8	

XOR	TestVar9	
EQ	SignatureVar	
JMPCN	Error	
LD	TestVar2	
EQ	2863311530	(*1010 1010 1010 1010 1010 1010 1010 1010(B)*)
JMPCN	Error	
LD	SaveVar	
ST	TestVar2	(*END Test TestVar2*)
LD	TestVar3	(*BEGIN Test TestVar3*)
ST	SaveVar	
LD	TestVar1	
XOR	TestVar2	
XOR	TestVar4	
XOR	TestVar5	
XOR	TestVar6	
XOR	TestVar7	
XOR	TestVar8	
XOR	TestVar9	
ST	SignatureVar	
LD	1431655765	(*0101 0101 0101 0101 0101 0101 0101 0101(B)*)
ST	TestVar3	
LD	TestVar1	
XOR	TestVar2	
XOR	TestVar4	
XOR	TestVar5	
XOR	TestVar6	

XOR	TestVar7	
XOR	TestVar8	
XOR	TestVar9	
EQ	SignatureVar	
JMPCN	Error	
LD	TestVar3	
EQ	1431655765	(*0101 0101 0101 0101 0101 0101 0101 0101(B)*)
JMPCN	Error	
LD	2863311530	(*1010 1010 1010 1010 1010 1010 1010 1010(B)*)
ST	TestVar3	
LD	TestVar1	
XOR	TestVar2	
XOR	TestVar4	
XOR	TestVar5	
XOR	TestVar6	
XOR	TestVar7	
XOR	TestVar8	
XOR	TestVar9	
EQ	SignatureVar	
JMPCN	Error	
LD	TestVar3	
EQ	2863311530	(*1010 1010 1010 1010 1010 1010 1010 1010(B)*)
JMPCN	Error	
LD	SaveVar	
ST	TestVar3	(*END Test TestVar3*)

LD	TestVar4	(*BEGIN Test TestVar4*)
ST	SaveVar	
LD	TestVar1	
XOR	TestVar2	
XOR	TestVar3	
XOR	TestVar5	
XOR	TestVar6	
XOR	TestVar7	
XOR	TestVar8	
XOR	TestVar9	
ST	SignatureVar	
LD	1431655765	(*0101 0101 0101 0101 0101 0101 0101 0101(B)*)
ST	TestVar4	
LD	TestVar1	
XOR	TestVar2	
XOR	TestVar3	
XOR	TestVar5	
XOR	TestVar6	
XOR	TestVar7	
XOR	TestVar8	
XOR	TestVar9	
EQ	SignatureVar	
JMPCN	Error	
LD	TestVar4	
EQ	1431655765	(*0101 0101 0101 0101 0101 0101 0101 0101(B)*)
JMPCN	Error	

LD	2863311530	(*1010 1010 1010 1010 1010 1010 1010 1010(B)*)
ST	TestVar4	
LD	TestVar1	
XOR	TestVar2	
XOR	TestVar3	
XOR	TestVar5	
XOR	TestVar6	
XOR	TestVar7	
XOR	TestVar8	
XOR	TestVar9	
EQ	SignatureVar	
JMPCN	Error	
LD	TestVar4	
EQ	2863311530	(*1010 1010 1010 1010 1010 1010 1010 1010(B)*)
JMPCN	Error	
LD	SaveVar	
ST	TestVar4	(*END Test TestVar3*)

## 6 Fehlererkennung durch Test der Ein- und Ausgabe

### 6.1 Test des Speichers der Ein- und Ausgänge – O\_TEST

Variablen:

```

VAR
  Bitshifter : BYTE;
  ByteRun1 : BYTE;
  Save_Output : BYTE;
  Save_Pruef : BYTE;

  FB_Error:Error_FB;
  END_VAR

```

Fehlerbaustein aus Kapitel 1

Funktion:

```

LD      128
ST      Bitshifter

LD      3
ST      ByteRun1

Schleife1:                                (*Einzeltests: Ein Bit wird gekippt, dann wird alles
                                          überprüft.*)

(*Ein Ausgangsbit umkippen*)

LD      Sim_Output_Array[ByteRun1]
ST      Save_Output
XOR     Bitshifter
ST      Sim_Output_Array[ByteRun1]
ST      Save_Pruef

(*Pruefen des
Ausgangsspeichers*)

LD      Sim_Output_Array[ByteRun1]
EQ      Save_Pruef

```

JMPCN	Error
(*Ausgang wieder zurücksetzen*)	
LD	Save_Output
ST	Sim_Output_Array[ByteRun1]
(*Zähler runtersetzen*)	
LD	Bitshifter
SHR	1
ST	Bitshifter
EQ	0
JMPCN	Schleife1
LD	ByteRun1
SUB	1
ST	ByteRun1
EQ	0
JMPC	Ende1
LD	128
ST	Bitshifter
JMP	Schleife1
Ende1:	
RET	
Error:	
CAL	FB_Error
CalError:	
JMP	CalError

## 6.2 Test der redundanten Ausgänge über rückgekoppelte Eingänge

Dieser Test beinhaltet die Funktionsblöcke

- OO\_TEST
- OO\_TEST\_FALSE
- OO\_TEST\_TRUE

Bei denen der Funktionsblock OO\_TEST die anderen beiden aufruft.

## 6.2.1 OO\_TEST

Variablen:

```

VAR

Neuer_Test : BOOL := FALSE;

Reset_noetig : BOOL;

TestZyklus : BYTE := 0;

Output1_Merker : BOOL;

Output2_Merker : BOOL;

Test_OO_True_2_3 : OO_Test_True;

Test_OO_False_2_3 : OO_Test_False;

END_VAR

```

Funktion:

```

(*LD      Ausgänge_die_nicht_getestet_werden_Merker
ST      Ausgänge_die_nicht_getestet_werden*)

LD      Neuer_Test      (*Bei jedem neuen Testlauf wird
                        Reset_noetig auf True gesetzt*)

ST      Reset_noetig

LD      FALSE

ST      Neuer_Test

                        (*BEGIN - Abfrage welcher Output
                        getestet werden soll*)

LD      TestZyklus

EQ      0

JMPC    Output_2_3

LD      TestZyklus

EQ      1

JMPC    Output_4_5

```

		(*Hier können weitere OutputTests aufgefufen werden*)
LD	0	(*Wenn kein Test aufgerufen wurde wird der Zykluszähler von Null gestartet*)
ST	TestZyklus	
RET		
		(*END - Abfrage welcher Output getestet werden soll*)
Output_2_3:		
LD	Sim_Output_Bool2	(*Zwischenspeicherung der simmulierten Ausgänge, da direkte Variablen nicht per VAR_IN_OUT übergeben werden können werden hier Kopien übergeben*)
ST	Output1_Merker	
LD	Sim_Output_Bool3	
ST	Output2_Merker	
CAL	Test_OO_True_2_3( ZyklenBisFehler:= 100, Reset:= Reset_noetig, Input1:= Sim_Input_Bool2, Input2:= Sim_Input_Bool3, Output1:= Output1_Merker, Output2:= Output2_Merker )	
LD	Output1_Merker	(*Rückspeicherung der bearbeiteten Kopien in die Originale*)
ST	Sim_Output_Bool2	
LD	Output2_Merker	
ST	Sim_Output_Bool3	
LD	Test_OO_True_2_3.Gepueft	

RETCN

Ende\_Output\_2\_3:

LD TRUE

ST Neuer\_Test (\*beim nächsten Durchlauf wird ein  
Reset ausgelöst\*)

LD 1

ST TestZyklus

RET

Output\_4\_5:

(\*Hier kann eine weitere Testroutine  
eingefügt werden\*)

LD 2

ST TestZyklus

RET

## 6.2.2 OO\_TEST\_FALSE

Variablen:

```
VAR_INPUT
```

```
ZyklenBisFehler : BYTE;
```

```
Reset : BOOL;
```

```
Input1 : BOOL;
```

```
Input2 : BOOL;
```

```
Output1 : BOOL;
```

```
Output2 : BOOL;
```

```
END_VAR
```

```
VAR_OUTPUT
```

```
Gepueft : BOOL;
```

```
END_VAR
```

```
VAR
```

```
Fehlerzyklen : DWORD := 0;
```

```
FB_Error:Error_FB;
```

Fehlerbaustein aus Kapitel 1

```
END_VAR
```

Funktion:

```
JMP          StartTest
```

```
Error:
```

```
CAL          FB_Error
```

```
CalError:
```

```
JMP          CalError
```

```
StartTest:
```

(\*alle Eingänge und Ausgänge werden überprüft.  
Bei Ungleichheit wird eine Variable erhöht. Falls  
diese ZyklenBisFehler erreicht wird die SPS  
angehalten.\*)

```

(*Test0:*)
LD      FALSE
ST      Geprueft
LD      Reset      (*bei Reset Fehlerzyklen zurücksetzen*)
JMPCN   weiter1
LD      0
ST      Fehlerzyklen
weiter1:
LDN     Output1
ANDN    Output2
ANDN    Input1
ANDN    Input2     (*TRUE wenn alle 4 Ausgänge False sind*)
JMPC    TestOK     (*Wenn TRUE Test erfolgreich, wenn False
                   Anzahl Fehlerzyklen erhöhen*)

LD      Fehlerzyklen
ADD     1
ST      Fehlerzyklen
EQ      ZyklenBisFehler
JMPC    Error
JMP     TestEnde
TestOK:
LD      0
ST      Fehlerzyklen
LD      TRUE
ST      Geprueft
TestEnde:

```

## 6.2.3 OO\_TEST\_TRUE

Variablen:

VAR\_INPUT

ZyklenBisFehler : DWORD := 0;

Reset : BOOL;

Input1 : BOOL;

Input2 : BOOL;

END\_VAR

VAR\_IN\_OUT

Output1 : BOOL;

Output2 : BOOL;

END\_VAR

VAR\_OUTPUT

Gepueft : BOOL;

END\_VAR

VAR

Fehlerzyklen : DWORD := 0;

FB\_Error:Error\_FB;

Fehlerbaustein aus Kapitel 1

END\_VAR

Funktion:

JMP                   Start

Error:

CAL                   FB\_Error

CalError:

JMP                   CalError

Start:

LD	Reset	(*Bei jedem Reset werden die Fehlerzyklen zurückgesetzt und die Pruefung des ersten eingangs aktiviert.*)
JMPCN	StartTest	
LD	0	
ST	Fehlerzyklen	
LD	TRUE	
ST	PruefeEins	
StartTest:		
LD	FALSE	
ST	Geprueft	
LD	PruefeEins	(*Auswahl des zu prüfenden Ausgangs*)
JMPC	Eins	
LD	Output2	
JMPCN	PruefeZwei	(*Wenn der 2te Ausgang nicht gesetzt ist startet die Überprüfung des verknüpften Eingangs*)
LD	Input1	
AND	Output1	
JMPC	EinsGesetzt	(*Andernfalls wird überwacht, ob der erste Ausgang gesetzt ist, damit es zu keiner Doppelabschaltung kommt.*)
LD	TRUE	(*ist der erste Ausgang nicht gesetzt, wird dies hier erledigt.*)
ST	Output1	

JMP	Ende	(*danach wird ans Ende gesprungen. Ein Fehler wird also auch ausgelöst, wenn der Ausgang gesetzt ist, dies aber nicht am Eingang bemerkt wird.*)
EinsGesetzt:		
LD	FALSE	(*wenn Ausgang 1 gesetzt ist, wird hier Ausgang 2 abgeschaltet*)
ST	Output2	
JMP	Ende	(*Da am Eingang wegen des EVA Prinzip noch keine Reaktion anliegen kann, wird hier ans Ende gesprungen*)
PruefeZwei:		
LD	Input2	
JMPC	Ende	(*Ist der Eingang2 noch auf wahr wird ans Ende gesprungen. Dies bedeutet ein hochzählen der Fehler.*)
LD	TRUE	(*Im anderen Fall wird die Routine als geprüft verlassen und die Fehlerzyklen zurückgesetzt*)
ST	Geprueft	
ST	PruefeEins	
LD	0	
ST	Fehlerzyklen	
RET		
Eins:		
LD	Output1	
JMPCN	PruefeEins	(*Wenn der erste Ausgang nicht gesetzt ist startet die Überprüfung des verknüpften Eingangs*)
LD	Input2	

AND	Output2	
JMPC	ZweiGesetzt	(*Andernfalls wird überwacht, ob der zweite Ausgang gesetzt ist, damit es zu keiner Doppelabschaltung kommt.*)
LD	TRUE	(*ist der zweite Ausgang nicht gesetzt, wird dies hier erledigt.*)
ST	Output2	
JMP	Ende	(*danach wird ans Ende gesprungen. Ein Fehler wird also auch ausgelöst, wenn der ausgang gesetzt ist, dies aber nicht am Eingang bemerkt wird.*)
ZweiGesetzt:		
LD	FALSE	(*wenn Ausgang 2 gesetzt ist, wird hier Ausgang 1 abgeschaltet*)
ST	Output1	
JMP	Ende	(*Da am Eingang wegen des EVA Prinzip noch keine Reaktion anliegen kann, wird hier ans Ende gesprungen*)
PruefeEins:		
LD	Input1	
JMPC	Ende	(*Ist der Eingang1 noch auf wahr wird ans Ende gesprungen. Dies bedeutet ein hochzählen der Fehler.*)
LD	FALSE	(*Im anderen Fall wird mit der Überprüfung von Ausgang 2 fortgeführt.*)
ST	PruefeEins	
RET		
Ende:		
LD	Fehlerzyklen	(*Fehlerzyklen zählt die Zyklen bis zur erfolgreichen Prüfung*)

ADD	1
ST	Fehlerzyklen
GE	ZyklenBisFehler
JMPC	Error

## **7 Fehlererkennung durch gegenseitige Überwachung mit einer zweiten SPS**

### **7.1 Kontrolle der Ablaufkontrolle durch zweite SPS**

Die Bausteine:

- ABLAUF\_UEBERWACHUNG\_INIT
- ABLAUF\_UEBERWACHUNG\_SEND

gehören zur Kontrolle der Ablaufkontrolle durch eine zweite SPS.

Dabei ist der Baustein ABLAUF\_UEBERWACHUNG\_INIT für die Einmalige Aufnahme der Verbindung gedacht und der Baustein ABLAUF\_UEBERWACHUNG\_SEND für den kontinuierlichen Austausch von Daten.

## 7.1.1 ABLAUF\_UEBERWACHUNG\_INIT

Variablen:

```

VAR_OUTPUT

andereSPSlaueft : BOOL;

END_VAR

```

Funktion:

```

(*Die SPS fängt an zu zählen und zu senden.
Gleichzeitig wartet sie darauf, dass die 2te SPS eine
andere Zahl als Null sendet.*)

LD      LastSend
ADD     1
ST      LastSend
ST      SendStatus
LD      FALSE
ST      andereSPSlaueft    (*Wert mit False vorbelegt*)

LD      ListenStatus
EQ      0
RETC    (*Rücksprung, wenn 2te SPS nicht läuft*)

LD      TRUE
ST      andereSPSlaueft    (*2te SPS läuft ausgeben*)

LD      ListenStatus    (*aktuellen Wert der 2ten SPS merken. - Wichtig
wenn dies die 2te SPS ist, die anläuft, da die andere
schon weit gezählt haben kann.*)

ST      ListenStatus_last

```



LD	ListenStatus	(*Empfangene Variable mit letzter gemerkter empfangener Variable prüfen*)
SUB	ListenStatus_last	
ST	ListenStatus_last_differenz	(*Differenz merken*)
EQ	0	
JMPCN	Weiter1	(*Wenn beide gleich sind nicht springen, wenn nicht prüfen ob Differenz OK ist*)
LD	Equals_occured	(*Anzahl Gleichheiten erhöhen und bei mehr als erlaubt in Fehleroutine springen*)
ADD	1	
ST	Equals_occured	
GE	Equals_till_Error	
RETCN		
JMP	Error	
Weiter1:		
LD	ListenStatus_last_differenz	(*Differenz mit maximal erlaubter Differenz prüfen. Da Differenzen verglichen werden führt ein Rückwärtszählen der anderen SPS zu einer großen Differenz (Zahlenkreis) und damit zum Fehler*)
GT	ListenStatus_last_differenz_Max	
JMPC	Error	
LD	0	
ST	Equals_occured	

LD	ListenStatus
ST	ListenStatus_last

## 7.2 Test der zweiten Möglichkeit Ausgänge abzuschalten – OO\_TEST\_ZWEITE\_SPS

Variablen:

```

VAR

Neuer_Test : BOOL := FALSE;

Reset_noetig : BOOL;

TestZyklus : BYTE := 0;

Output1_Merker : BOOL;

Output2_Merker : BOOL;

Test_OO_True_2_3 : OO_Test_True;           Aus Kapitel 6.2.3

Test_OO_False_2_3 : OO_Test_False;        Aus Kapitel 6.2.2

END_VAR

```

Funktion:

```

(*LD      Ausgänge_die_nicht_getestet_werden_Merker
ST        Ausgänge_die_nicht_getestet_werden*)

LD        Neuer_Test          (*Bei jedem neuen Testlauf wird
                               Reset_noetig auf True gesetzt*)

ST        Reset_noetig

LD        FALSE

ST        Neuer_Test

                               (*BEGIN - Abfrage welcher Output
                               getestet werden soll*)

LD        TestZyklus

EQ        0

JMPC     Output_2_3

LD        TestZyklus

EQ        1

```

JMPC	Output_4_5	(*Hier können weitere OutputTests aufgefufen werden*)
LD	0	(*Wenn kein Test aufgerufen wurde wird der Zykluszähler von Null gestartet*)
ST	TestZyklus	
RET		(*END - Abfrage welcher Output getestet werden soll*)
Output_2_3:		
(*LD	Alle_restlichen_Outputs_Merker	
ST	Alle_restlichen_Outputs*)	
LD	Output_2_3_Merker	(*Im Output_2_3_Merker steht ob die Ausgänge High oder Low sein sollen*)
JMPC	Output_2_3_High	(*Sprung in den HIGH Test*)
LD	Sim_Output_Bool2	
OR	Sim_Output_Bool3	
JMPCN		(*Wenn beide Ausgänge LOW sind, Sprung in den LOW Test*)
ST	Reset_noetig	(*Reset Noetig wird hier auf TRUE gesetzt*)
LD	FALSE	(*Bei Low keine Prüfung, nur beide zurücksetzen.*)
ST	Sim_Output_Bool2	
ST	Sim_Output_Bool3	
Output_2_3_Low_Test:		
CAL	Test_OO_False_2_3(	
ZyklenBisFehler:= 100,		

```

Reset:= Reset_noetig,
Input1:= Sim_Input_Bool2,
Input2:= Sim_Input_Bool3,
Output1:= Sim_Output_Bool2,
Output2:= Sim_Output_Bool3 )
LD          Test_OO_False_2_3.Geprueft
RETCN
JMP         Ende_Output_2_3          (*Sprung ans Ende*)
Output_2_3_High:
LDN        Sim_Output_Bool2          (*Vergleich: wenn weder der eine, noch
                                     der andere Ausgang gesetzt ist, werden
                                     hier beide gesetzt. Danach wird die
                                     Prüfung begonnen.*)
ANDN       Sim_Output_Bool3
JMPCN      Output_2_3_High_Test
LD         TRUE
ST         Reset_noetig
ST         Sim_Output_Bool2
ST         Sim_Output_Bool3
Output_2_3_High_Test:
LD         Sim_Output_Bool2          (*Zwischenspeicherung          der
                                     simmulierten Ausgänge, da direkte
                                     Variablen nicht per VAR_IN_OUT
                                     übergeben werden können werden hier
                                     Kopien übergeben*)
ST         Output1_Merker
LD         Sim_Output_Bool3
ST         Output2_Merker
CAL        Test_OO_True_2_3(
ZyklenBisFehler:= 100,

```

```
Reset:= Reset_noetig,  
Input1:= Sim_Input_Bool2,  
Input2:= Sim_Input_Bool3,  
Output1:= Output1_Merker,  
Output2:= Output2_Merker )  
  
LD          Output1_Merker          (*Rückspeicherung der bearbeiteten  
                                     Kopien in die Originale*)  
  
ST          Sim_Output_Bool2  
  
LD          Output2_Merker  
  
ST          Sim_Output_Bool3  
  
LD          Test_OO_True_2_3.Geprueft  
  
RETCN  
  
Ende_Output_2_3:  
  
LD          TRUE  
  
ST          Neuer_Test              (*beim nächsten Durchlauf wird ein  
                                     Reset ausgelöst*)  
  
LD          1  
  
ST          TestZyklus  
  
RET  
  
  
Output_4_5:  
  
(*Hier kann eine weitere Testroutine eingefügt werden*)  
  
LD          2  
  
ST          TestZyklus  
  
RET
```