Beschreibung der Software
LONMPM 8DI/8DO Anwendung Version 1691111a

Schnittstellenbeschreibung und Softwaredokumentation

1 Revisionsdokumentation

Auf dieser Seite sind die freigegebenen und die zur Freigabe verbreiteten Softwareversionen aufgeführt.

Diese Dokumentation gilt für folgende Softwareversionen:

1.1 Version 1691111a

2 Allgemeines

2.1 Gerätebeschreibung

Die Steuerung LON –MPM 8DI/8DO stellt ein multifunktionales Modul für unterschiedlichste Steueraufgaben dar. Sie kann 8 potentialfreie Schalter, Taster oder andere Kontakte einlesen. An ihren Steuerausgängen stellt sie 8 potentialfreie Kontakte zur Verfügung. Abhängig von der eingesetzten Software kann dieses Modul beispielsweise für die Ansteuerung von Licht, die Überwachung von Fensterkontakten, oder andere Steueraufgaben genutzt werden, die potentialfreie Ein- und Ausgänge benötigt. Die Hardware verfügt über eine Notbedienebene, mit deren Hilfe die Steuerung der Ausgänge durch die Neuron-Software übergangen werden kann. Im ungebundenen Zustand wirkt jeder Schaltereingang direkt auf den Ausgang mit der gleichen Nummer. Damit ist es dem Installateur möglich die Verdrahtung zu testen.
Die Kommunikation über das LON erfolgt ausschließlich über Netzwerkvariablen bzw. LONTALK Network Management Messages.

2.2 Typografie

Standardtext in Arial Standard 10 pt
Dokumentenverweise in Arial Kursiv 10 pt
`Auszüge aus NEURON-C-Quelltexten in Courier New Standard 10 pt`
Hintergrundschattierungen in umfangreichen Aufzählungen oder Tabellen sollen deren Lesbarkeit erhöhen, oder bestimmte Begriffe hervorheben.
Das Layout dieses Dokuments ist an das Layout der *LONMARK Application Layer Interoperability Guidelines Version 3.2* [2] angelehnt.
Derzeit stellen die am Markt üblichen LONWORKS-Inbetriebnahme-Tools die Daten der Netzwerkvariablen unterschiedlich dar. Zur größtmöglichen Transparenz werden diese Daten hier weitestgehend hexadezimal codiert dargestellt.

2.3 Nomenklatur

Variablen und Funktionen der Neuron-C-Quelltexte sind zur einfacheren Typzuordnung und zur Erkennung des gültigen Wertebereichs mit Präfixen versehen:

Präfix	Item	Beispiel
us	unsigned short	`unsigned short usTempSwitches;`
s	signed short	`signed short sTemp;`
ul	unsigned long	`unsigned long  ulongTemp;`
l	signed long	`signed long lTemp;`
str	string	`char strDescriptionSwitch[8];`
t	Variablentyp	`typedef enum {...}tConfType;`
UNVT	herstellerdefinierter NV-Typ	`typedef struct{...}UNVT_cntrl_config;`
p	Zeiger	`tMiscConfData *pData;`
fp	function pointer	`tFpConfigItem fpSwitchConfigItem;`
io	Port-Pin des 3150	`IO_8 input bit ioManOvr;`
mt	Millisekunden-Timer	`mtimer repeating mt10ms=10;`
st	Software-Timer	`unsigned short stResetPWM;`
nvo	Ausgangs-Netzwerkvariable	`network output SNVT_obj_status nvoStatus;`
nvi	Eingangs-Netzwerkvariable	`network input SNVT_time_stamp nviTimeSet;`

Tabelle 1: Nomenklatur

2.4 Weitere Dokumentationen

[1] *SNVT Master List and Programmers Guide May 1997*
Sie beschreibt Wertebereiche, Einheiten, und Auflösung aller Standard Netzwerkvariablen Typen (SNVT's). Zur Erstellungszeit war die SNVT Master List Version 10 aktuell.
[2] LONMARK Application Layer Interoperability Guidelines Version 3.2
Sie liefern Design-Richtlinien für Objekte auf LONWORKS-Geräten.

[3] LONMARK Functional Profile #3200 Switch Version 1.0
[4] LONMARK Functional Profile #3040 Lamp Actuator 1.0

2.5 Einschränkungen

Soweit nicht anders beschrieben, sind alle hier aufgeführten Datenformate und Funktionalitäten entsprechend LONMARK-Konventionen bzw. *SNVT Master List and Programmers Guide May 1997* [1]zu verstehen.
Grundkenntnisse der Programmiersprachen C bzw. Neuron C sind zum Verständnis einzelner Dokumentationsteile erforderlich.

2.6 Objekte

Ein Objekt stellt eine geschlossene abstrakte Einheit aus Verhalten (Funktionalität) und Eigenschaften (Daten) dar; es kann Schnittstellen zur Kommunikation mit der Außenwelt (z.B. andere Objekte) zur Verfügung stellen. Als Basis der Objektdefinitionen dienen die generischen Objekte, wie sie in den *LONMARK Application Layer Interoperability Guidelines Version 3.2* [2] beschrieben sind.
In der Applikation 1691111a sind mehrere Objekte implementiert:

- 1 Node Objekt (Object Type #0) [2]
- 8 Switch Objekte (Functional Profile #3200, modifiziert im Funktionsumfang) [3]
- 8 Lamp Actuator Objekte (Functional Profile #3040, modifiziert im Funktionsumfang) [4]

Alle hier genannten Objekte sind gleichzeitig auf einem Gerät implementiert. Die Objekte können untereinander und/oder auf Objekte entfernter Geräte gebunden werden.

2.7 Herstellerdefinierte Netzwerkvariablenformate

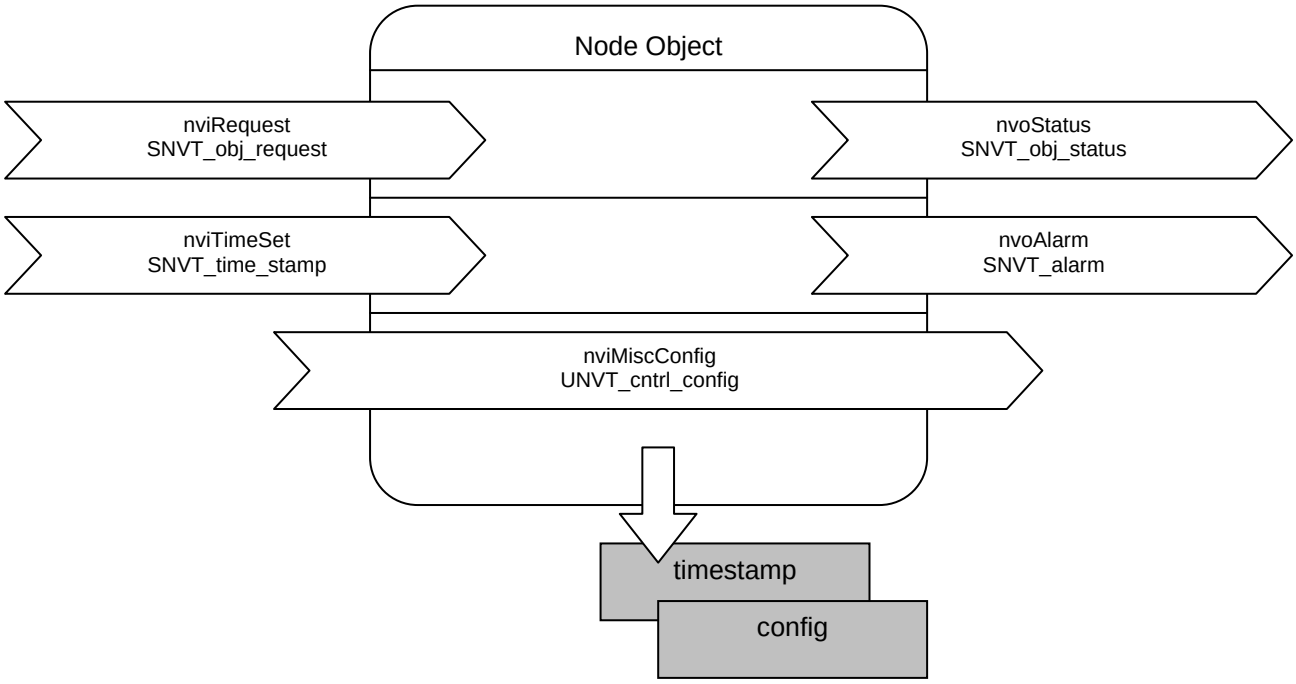
```
typedef enum
 { ConfigSave=0,
   ConfigClear=1,
   ConfigReport=2
      ...
}tConfRequest;

typedef enum
 {
  CFG_FAILED=-1,                //=255
  INVALID_CFGREQUEST=-2,        //=254
  CFG_SUCCESS=-3,               //=253
  CFG_REPORT=-4,                //=252

  //for switches
  OFFSET_CFGTYPE_SWITCH=1,
  CFG_SWITCH_FUNCTION=2,        //Arrayelement 0 im Functionpointerarray
  CFG_SWITCH_HBTIMER=3,
  CFG_SWITCH_TIMER1=4,
  CFG_SWITCH_DESCRIPTION=5,     //letztes ConfigItem
  MAX_CFGTYPE_SWITCH=6,

  //for actuators
  OFFSET_CFGTYPE_ACTR=50,
  CFG_ACTR_FUNCTION=51,         //Arrayelement 0 im Functionpointerarray
  CFG_ACTR_TIMER1=52,
  CFG_ACTR_DESCRIPTION=53,
  CFG_ACTR_COMMON_PARAM=54,
  MAX_CFGTYPE_ACTR=55,

  //for node
  OFFSET_CFGTYPE_NODE=70,
```

```
  CFG_NODE_IO=71,
  CFG_NODE_DELAY_RELAYS=72,
  CFG_NODE_DELAY_INPUT=73,
  CFG_NODE_PWM_RELAYS=74,
  CFG_NODE_ACTR_OVERRIDE=75,
  MAX_CFGTYPE_NODE=76
  }tConfType;

typedef union
{
  unsigned short  ushort;
  signed short    sshort;
  unsigned long   ulong;
  signed long     slong;
  unsigned short  raw[28];
}tMiscConfData;


typedef struct
 {
  tConfRequest usConfRequest;
  signed short sObjectIndex;
  tConfType    ConfType;
  tMiscConfData data;
 }UNVT_cntrl_config;
```

3 Node Object

Das Node Object [2] stellt Mechanismen und Schnittstellen zum Ändern und Abfragen von Zuständen aller Objekte eines Gerätes zur Verfügung. Es verfügt außerdem über Netzwerkvariablen, die nicht nur einem einzelnen Objekt, sondern mehreren oder allen Objekten eines Gerätes zugeordnet sein können.

3.1 Schnittstellenbeschreibung



3.1.1 Object Request

```
network input sd_string("@0|1") SNVT_obj_request nviRequest;
```

NOCH NICHT IMPLEMENTIERT

Objekt	**Objektindex** `nviRequest.object_id`	**Bezeichnung gemäß** `ObjectEnumeration`
Node	0	`ObjEnum_Node`
Switch Object 0	1	`ObjEnum_Switch_0`
Switch Object 1	2	`ObjEnum_Switch_1`
Switch Object 2	3	`ObjEnum_Switch_2`
Switch Object 3	4	`ObjEnum_Switch_3`
Switch Object 4	5	`ObjEnum_Switch_4`
Switch Object 5	6	`ObjEnum_Switch_5`
Switch Object 6	7	`ObjEnum_Switch_6`
Switch Object 7	8	`ObjEnum_Switch_7`
Actuator Object 0	9	`ObjEnum_Actr_0`
Actuator Object 1	A	`ObjEnum_Actr_1`
Actuator Object 2	B	`ObjEnum_Actr_2`
Actuator Object 3	C	`ObjEnum_Actr_3`
Actuator Object 4	D	`ObjEnum_Actr_4`
Actuator Object 5	E	`ObjEnum_Actr_5`
Actuator Object 6	F	`ObjEnum_Actr_6`
Actuator Object 7	10	`ObjEnum_Actr_7`

Tabelle 2: Objektindizes

3.1.2 Object Status

```
network output sd_string("@0|2") SNVT_obj_status nvoStatus;
```

NOCH NICHT IMPLEMENTIERT

3.1.3 Eingang Zeit und Datum

```
network input sd_string("@0|3") SNVT_time_stamp nviTimeSet;
```

Diese NV wird zur Zeit in dieser Softwareversion nicht verwendet.

3.1.4 Alarmzustand

NOCH NICHT IMPLEMENTIERT

3.1.5 Gerätekonfiguration

```
network input sd_string("@0|9") UNVT_cntrl_config nviMiscCfg;
```

Diese Netzwerkvariable dient der Parametrierung aller im Gerät enthaltenen Objekte. Diese NV ist insgesamt 31 Bytes lang und kann bis zu 28 Bytes Daten zur Parametrierung übertragen. Beim Auslesen von Konfigurationseinstellungen wird nur die erforderliche Anzahl an Datenbytes aktualisiert. Weitere Bytes, die nach dem Auslesen noch Daten enthalten, sind bedeutungslos.

Valid Range

Element	Wertebereich / Bedeutung	
`.request`	0x00:	Parameter überschreiben mit neuen Daten
	0x01:	Bedeutung siehe folgende Abschnitte
	0x02:	Parameterdaten auslesen.
		weitere Dienste werden nicht unterstützt
`.objectIndex`	Index des Objektes, auf den `.request` angewendet werden soll Siehe Tabelle 1	
`.configType`	Auswahl Parameterart:	
	0x02:	Funktion des Switch Objektes
	0x03:	Heartbeattimer des Switch Objektes
	0x04:	Timer 1 des Switch Objektes
	0x05:	Description String des Switchobjektes
	0x33:	Funktion des Aktor Objektes
	0x34:	Timer 1 des Aktor Objektes
	0x35:	Description String des Aktor Objektes
	0x36:	Für alle Aktoren gültige Parameter
	0x47:	Auslesen der digitalen Ein-/Ausgänge des Knotens
	0x48:	Verzögerung für kaskadiertes Schalten
	0x49:	Zeitkonstante für das Entprellen der Kontakte
	0x4A:	Anzugspannung für die Relais im Stromsparmodus
	0x4B:	temporärer Override Modus für einzelne Aktoren
	Beim Rücklesen der nviMiscConfig nach vorhergehendem Beschreiben:	
	0xFC:	ausgelesene Daten werden angezeigt
	0xFD:	Konfiguration erfolgreich
	0xFE:	unzulässige Konfigurationsanfrage
	0xFF:	Konfiguration fehlgeschlagen
`.data`	bis zu 28 Byte Daten, die abhängig von `.request, .objectIndex, .configType` interpretiert werden.	

Tabelle 3: Aufbau der Konfigurationsvariable nviMiscCfg

Die Bedeutung der Parametrierdaten und die Länge der gültigen Daten kann den Kapiteln der einzelnen Objekte entnommen werden.
Im weiteren wird diese Netzwerkvariable folgendermaßen dargestellt:

02	06	FC	00	B4	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Die einzelnen Elemente gemäß obiger Tabelle werden farblich getrennt. Felder, deren Daten zur beschriebenen Parametrierung keinen Beitrag leisten, oder deren Inhalt irrelevant ist werden mit XX beschrieben.

Der prinzipielle Vorgang der Parametrierung sieht vor, daß zuerst die `nviMiscCfg` beschrieben wird. Die ersten drei Byte der geschriebenen Daten bestimmen die Aktion, die ausgeführt wird. Das erste Byte (`.request`) stellt die Anforderung des Dienstes entsprechend obiger Tabelle dar. Das zweite Byte (`.objectIndex`) bestimmt das angesprochene Objekt. Das dritte Byte (`.configType`) bestimmt die Art der Parametrierdaten, die gelesen, oder geschrieben werden sollen. Abhängig vom angeforderten Dienst werden Daten im Gerät gespeichert, oder ausgelesen. Nachdem das Gerät die Anforderung erhalten hat, führt es die gewünschte Aktion durch und berichtet über Erfolg, oder Mißerfolg der Aktion. Dies geschieht indem das Feld `.configType` verändert wird. Damit wird der

parametrierenden Instanz kenntlich gemacht, ob ein Polling der `nviMiscCfg` zu früh erfolgt ist und die Parametrierung noch andauert, oder ob der Vorgang abgeschlossen wurde.
Hier soll anhand von konkreten Beispielen die Vorgehensweise der Parametrierung gezeigt werden.

Beispiel 1:
Das dritte Switch Objekt soll als UND-Glied parametriert werden.
`.request = 0x00`;
`.objectIndex = 0x03`; für das dritte Switch Objekt
`.configType = 0x02`; damit die Funktion des Switch Objektes parametriert wird.
`.data = 0x02`; damit das Switch Objekt als UND-Glied funktioniert.
Setze `nviMiscCfg` auf :

00	03	02	02	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Nach dem Rücklesen der `nviMiscCfg` befindet sich im Feld `.configType` eine Rückmeldung über den Erfolg der Parametrierung. Es wird folgendes gelesen:

00	03	FD	02	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Beispiel 2:
Die Parametrierung des Timer 1 vom zweiten Aktor Objekt soll ausgelesen werden.
`.request = 0x02`;
`.objectIndex = 0x0A`; für das zweite Aktor Objekt
`.configType = 0x34`; damit der Timer 1 des Aktor Objekts ausgelesen wird.
Setze `nviMiscCfg` auf :

02	0A	34	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Nach dem Rücklesen der `nviMiscCfg` befindet sich im Feld `.configType` eine Rückmeldung über die ausgeführte Aktion. Es wird folgendes gelesen:

02	0A	FC	YY	YY	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

... wobei YY YY die ausgelesenen Daten darstellen und XX irrelevante Daten für diese Aktion sind.

3.2 Parametrierung

3.2.1 Unterstützte Dienste

Über die Parametrierschnittstelle, der Netzwerkvariablen `nviMiscCfg`, werden folgende Dienste unterstützt:

Dienst	`nviMiscCfg.request`	Bedeutung
SCF_SAVE	`0x00`	Konfiguration überschreiben mit neuen Daten
SCF_REPORT	`0x02`	Abrufen der Konfiguration

Tabelle 4: Unterstützte Dienste des Node Objekts

3.2.2 Unterstützte Objekte

Die Software 1691111a unterstützt ein einziges Node Objekt.

Objektbezeichnung	**Objektindex** `nviRequest.object_id`
Node Object	`0x00`

Tabelle 5: Objektindex des Node Objekts

3.2.3 Parametrierbare Optionen

Folgenden Tabelle faßt alle Konfigurationsmöglichkeiten des Sensor Objekts zusammen.

Element	nviMiscCfg. configType	Bedeutung
NodeIORead	0x47	aktueller Zustand von Ein-/Ausgängen auslesen
stDigitalOutputsReload	0x48	Kaskadiertes Schalten der Ausgänge
stDigitalInputsReload	0x49	Entprellen der Eingangskontakte
NodePWMRelaysConfig	0x4A	Stromsparmodus für die Relais

Tabelle 6: Parametrierbare Optionen des Sensor Objekts

3.2.4 Ein- und Ausgänge auslesen

Über die Netzwerkvariable nviMiscCfg kann der Zustand der Kontaktsätze am Eingang und der Zustand der Relais am Ausgang ausgelesen werden. Damit kann unabhängig von Parametrierung und Binding der tatsächliche Zustand des gesamten Knotens ermittelt werden. Zusätzlich kann ermittelt werden, ob der Knoten durch die Notbedienebene übersteuert wird oder noch eine Ausschaltverzögerung abläuft.
Diese Konfigurationsart unterstützt nur lesenden Zugriff. Ein schreibender Zugriff ist nicht möglich.
Beim Auslesen werden in MiscCfg.data 3 Bytes übermittelt die folgendermaßen zugeordnet werden können:
1. Byte: Eingänge
2. Byte: Notbedienebene
3. Byte: Ausgänge
4. Byte: Temporäre Übersteuerung der Relaisausgänge durch Parametrierung (vgl.3.2.8)
5. Byte: Sollzustand der Relais
6. Byte: Istzustand der Relais

Mit Hilfe nachfolgender Tabelle können diese Werte interpretiert werden. Der angezeigte Wert entspricht der Summe der Wertigkeiten in nachstehender Tabelle.

Kontakt	Wertigkeit	Klemme
Eingang 1 Kontakt geschlossen	0x01	Schalter 1: X9.1 – X9.2
Eingang 2 Kontakt geschlossen	0x02	Schalter 2: X9.3 – X9.4
Eingang 3 Kontakt geschlossen	0x04	Schalter 3: X9.5 – X9.6
Eingang 4 Kontakt geschlossen	0x08	Schalter 4: X9.7 – X9.8
Eingang 5 Kontakt geschlossen	0x10	Schalter 5: X9.9 – X9.10
Eingang 6 Kontakt geschlossen	0x20	Schalter 6: X9.11 - X9.12
Eingang 7 Kontakt geschlossen	0x40	Schalter 7: X9.13 - X9.14
Eingang 8 Kontakt geschlossen	0x80	Schalter 8: X9.15 - X9.16
Notbedienebene aktiv	0x01	Schalter 4 des 4-fach DIP-Schalters
Ausgang 1 Relais angesteuert	0x01	K 1: X1.1 – X1.2
Ausgang 2 Relais angesteuert	0x02	K 2: X1.3 – X1.4
Ausgang 3 Relais angesteuert	0x04	K 3: X4.1 – X4.2
Ausgang 4 Relais angesteuert	0x08	K 4: X4.3 – X4.4
Ausgang 5 Relais angesteuert	0x10	K 5: X6.1 – X6.2
Ausgang 6 Relais angesteuert	0x20	K 6: X6.3 – X6.4
Ausgang 7 Relais angesteuert	0x40	K 7: X7.1 – X7.2
Ausgang 8 Relais angesteuert	0x80	K 8: X7.3 – X7.4

Tabelle 7: Interpretation der Schalterstellung

Die Wertigkeit der Bytes vier bis sechs wird analog zu den Ausgängen interpretiert. Byte vier gibt an, ob der derzeitige Zustand des Relais mittels Übersteuerung durch Parametrierung hervorgerufen wurde. Byte fünf gibt den Sollzustand der Relais an, wie er durch die Software bestimmt wird. Byte 6 gibt den Istzustand der Relais an, wie er durch die Software bestimmt wird. Bestehen Abweichungen zwischen Byte fünf und Byte sechs, so kann dies z.B. auf eine parametrierte Kaskadierung der Aktoren zurückgeführt werden, die verhindert, daß mehrere Relais gleichzeitig anziehen. Nach Ablauf der entsprechenden Zeiten geht der Istzustand in den Sollzustand über.

Beispiel 1:
Die Zustände der Ein- und Ausgänge eines Gerätes sollen ermittelt werden.
Setze `nviMiscCfg` auf :

02	00	47	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Nach dem Rücklesen der `nviMiscCfg` befindet sich im Feld `.configType` eine Rückmeldung über den Erfolg des Vorgangs. Es wird folgendes gelesen:

02	00	FC	2A	00	08	00	08	08	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

`0x2A= 0x20 + 0x08 + 0x02`
D.h. daß die Kontakte an X9.3 – X9.4 und X9.7 – X9.8 und X9.11 - X9.12 geschlossen sind.
`0x00` im zweiten Datenbyte sagt aus, daß die Notbedienebene nicht aktiv ist.
`0x08` im dritten Datenbyte bedeutet, daß das Relais welches mit K 4: X4.3 – X4.4 verbunden ist, angesteuert wird.
`0x00` im vierten Datenbyte besagt, daß kein Aktor durch Parametrierung übersteuert wurde.
`0x08` im fünften und sechsten Datenbyte besagen, daß der Istzustand der Relais gleichzeitig der Sollzustand ist.

Beispiel 2:
Die Relais verhalten sich nicht so wie es die Netzwerkvariablen an den Aktor Objekten vermuten lassen. Es soll getestet werden, ob die Notbedienebene aktiv ist.
Setze `nviMiscCfg` auf :

02	00	47	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Nach dem Rücklesen der `nviMiscCfg` befindet sich im Feld `.configType` eine Rückmeldung über den Erfolg des Vorgangs. Es wird folgendes gelesen:

02	00	FC	00	01	FF	00	40	40	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Da das erste Byte `0x00` ist, ist kein Kontakt an den Eingängen geschlossen.
Das zweite Byte ist `0x01` also nicht `0x00`. Damit ist ersichtlich, daß die Notbedienebene aktiviert wurde. Die Relais können dann nur noch über den 8-fach DIP-Schalter am Gerät bedient werden.
Im dritten Byte steht:
`0xFF= 0x80 + 0x40 + 0x20 + 0x10 + 0x08 + 0x04 + 0x02 + 0x01`
D.h. alle 8 DIP-Schalter wurden geschlossen.
Am vierten Byte `0x00` kann man erkennen, daß keine Übersteuerung durch Parametrierung stattgefunden hat. Das fünfte und sechste Byte zeigen an, daß nur das Relais des Ausgang 7 angezogen sein sollte.

Beispiel 3:
Es wurde eine Übersteuerung durch Parametrierung vorgenommen. Es soll geprüft werden, wie die Relais jetzt stehen.
Setze `nviMiscCfg` auf :

02	00	47	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Nach dem Rücklesen der `nviMiscCfg` befindet sich im Feld `.configType` eine Rückmeldung über den Erfolg des Vorgangs. Es wird folgendes gelesen:

02	00	FC	00	00	09	8F	89	09	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Da das erste Byte `0x00` ist, ist kein Kontakt an den Eingängen geschlossen.
Das zweite Byte zeigt an, daß die Notbedienebene nicht aktiviert wurde.
Im dritten Byte steht: `0x09= 0x08 + 0x01` D.h. Ausgang 4 und Ausgang 1 wurden angesteuert.
Am vierten Byte `0x8F` kann man erkennen, daß die Ausgänge 8,4,3,2,1 durch Parametrierung übersteuert wurden.
Das fünfte Byte zeigt den Sollwert der Relais. Es sollen die Ausgänge 8,4 und 1 angesteuert werden.
Am sechsten Byte erkennt man, daß bisher nur die Ausgänge 4 und 1 aktiviert wurden. Ausgang 8 muß aufgrund einer Kaskadierung noch mit dem Schließen seines Kontaktes warten.

3.2.5 Kaskadierung der Aktoren

Wenn mehrere große Lasten gleichzeitig angeschaltet werden, so kann es zu Spannungseinbrüchen im Versorgungsnetz kommen. Es empfiehlt sich die Lasten nicht alle gleichzeitig zu schalten. Die Software 1691111a realisiert eine solche Funktion. Kommt es zu einem Einschaltbefehl für mehrere Aktoren, so werden diese in einem parametrierbaren Intervall, beginnend beim Aktor mit dem kleinsten Objektindex eingeschaltet. Nach der konfigurierten Zeit schaltet der nächste Aktor sein Relais zu. Das Abschalten der Relais erfolgt sofort und ohne Berücksichtigung der Verzögerungszeit. Ist für den Aktor eine bestimmte Anschaltzeit programmiert, so läuft diese auch wenn die Kaskadierung das Relais noch nicht angezogen hat. Die Kaskadierung kann in 10ms Schritten konfiguriert werden. Es sind 65535 Schritte möglich. D.h. insgesamt kann die Kaskadierung bis knapp 11 Minuten erweitert werden. Eine derart große Kaskadierung zu parametrieren ist jedoch nicht ratsam. Erhalten beispielsweise 3 Relais gleichzeitig den Einschaltbefehl, so dauert es mindestens 22

Minuten, bis der dritte Aktor anzieht. Würde in dieser Zeit bereits der Ausschaltbefehl erfolgt sein, so hätte der Aktor nie angezogen. Sinnvoll erscheinen Werte bis zu 5 Sekunden.
Die Möglichkeit die Kaskadierung zu verändern sollte nur in Ausnahmefällen eingesetzt werden.

Beispiel 1:
Die Kaskadierung der Relais soll auf 3 Sekunden eingestellt werden.
3s=300*10ms; 300=0x012C
Setze `nviMiscCfg` auf :

00	00	48	01	2C	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Nach dem Rücklesen der `nviMiscCfg` befindet sich im Feld `.configType` eine Rückmeldung über den Erfolg des Vorgangs. Es wird folgendes gelesen:

00	00	FD	01	2C	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Beispiel 2:
Die Kaskadierung ist für das Überwachen von Fensterkontakten nicht praktikabel. Sie soll ausgeschaltet werden.
Setze `nviMiscCfg` auf :

00	00	48	00	00	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Nach dem Rücklesen der `nviMiscCfg` befindet sich im Feld `.configType` eine Rückmeldung über den Erfolg des Vorgangs. Es wird folgendes gelesen:

00	00	FD	00	00	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

3.2.6 Entprellen der Eingänge

Mechanische Kontakte müssen entprellt werden, da sie beim Öffnen oder Schließen prellen, bevor sie in ihrer Endlage verharren. Die Zeit hierfür kann ebenfalls parametriert werden. Diese Parametrierung sollte nur in Ausnahmefällen geändert werden. Die Werkseinstellung erlaubt komfortable Bedienung bei ausreichender Entprellung der Kontakte. Es kann ein Wert für das Entprellen konfiguriert werden, der sich im Bereich von 0 (keine Entprellung) bis 255 (Entprellung dauert ca. 5 Sekunden) eingestellt werden. Bei Auslieferung des Gerätes ist der Wert 2 voreingestellt.

Beispiel 1:
Die Entprellung soll im Vergleich zur Werkseinstellung etwas erhöht werden, da die eingesetzten Fensterkontakte sehr lange prellen. Wir wählen den Wert von 5.
Setze `nviMiscCfg` auf :

00	00	49	05	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Nach dem Rücklesen der `nviMiscCfg` befindet sich im Feld `.configType` eine Rückmeldung über den Erfolg des Vorgangs. Es wird folgendes gelesen:

00	00	FD	05	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

3.2.7 Ansteuerung der Relais

Die Hardware auf der die Software 1691111a läuft, bietet verschiedene Mechanismen um Strom zu sparen. Wenn ein Wechsel der Relaisausgänge ansteht, werden diese mit der vollen Betriebsspannung angesteuert, danach wird die Ansteuerspannung reduziert. Abhängig von den eingesetzten Relais müssen verschiedene Einstellungen für die Dauer der Ansteuerung und die reduzierte Spannung in der Software vorgenommen werden. Werkseitig ist die Software schon auf die eingesetzten Relais konfiguriert. Über die `nviMiscCfg` können zwei 8-Bit Werte konfiguriert werden.

Der erste der Werte legt die Zeit fest, in der die Relais mit der vollen Leistung angezogen wird. Es können Werte bis 255 konfiguriert werden. Pro Inkrement verlängert sich Zeit um 10ms. Maximal sind also 2,55s möglich. Der zweite Wert legt die Spannung fest, mit der nach dem Ablauf dieser Zeit die Relais angesteuert werden. Maximal ist wieder der Wert 255 möglich. Je geringer der Wert, desto geringer ist die Spannung mit der die Relais gehalten werden.
Im Allgemeinen sollte von einer Veränderung dieser Werte abgesehen werden.

Beispiel 1:
Die Zeit für das Ansteuern der Relais mit voller Leistung soll auf 2 Sekunden parametriert werden.
2s=200*10ms ; 200=0xC8
Gleichzeitig soll nach dieser Zeit die Ansteuerspannung auf 60% der maximal möglichen reduziert werden.
255*60%=153; 153=0x99

Setze `nviMiscCfg` auf :

00	00	4A	C8	99	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Nach dem Rücklesen der `nviMiscCfg` befindet sich im Feld `.configType` eine Rückmeldung über den Erfolg des Vorgangs. Es wird folgendes gelesen:

00	00	FD	C8	99	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

3.2.8 Temporäre Übersteuerung der Relaisausgänge

Die Parametrierung über die `nviMiscCfg` bietet auch die Möglichkeit die Ausgänge der Aktoren zu übersteuern. Damit kann über ein Plug in der Zustand der Relais gesteuert werden, ohne auf die eigentlichen Netzwerkvariablen der Aktoren zugreifen zu müssen. Außerdem muß für diesen Vorgang noch kein Binding vorliegen. Im ungebundenen Zustand wirken bei der Software 1691111a die Eingänge direkt auf die Ausgänge des entsprechenden Aktors um dem Installateur die Überprüfung seiner Verdrahtung zu ermöglichen. Wird jedoch über diese Parametrierung ein Eingriff vorgenommen, so wird dieses Verhalten bis zum nächsten Reset unterbunden. Passiert ein Eingriff während des normalen Betriebs, so bleibt der eingestellte Zustand an den geänderten Relais bestehen, bis der zugehörige Aktor das nächste mal eine Aktualisierung seiner Eingangsvariablen erfährt. Im Falle von Closed Loops kann dies abhängig vom Binding und der Konfiguration unmittelbar nach dem Ändern des Relaiszustandes erfolgen, so daß eine Änderung sehr schnell wieder zurück genommen wird. Die `nviMiscCfg` stellt hierfür nur schreibende Parametrierung bereit. Es werden 2 Bytes erwartet, die entsprechend der Tabelle 7 interpretiert werden müssen. Das erste Byte legt fest, welche Ausgänge übersteuert werden sollen. Das zweite Byte bestimmt den Zustand der Relaisausgänge. Soll das Ergebnis dieser Parametrierung überprüft werden, so bietet sich eine Vorgehensweise gemäß 3.2.4 an. Nach Beendigung der Übersteuerung wird ausdrücklich empfohlen einen Reset am Gerät durchzuführen.

Beispiel 1:
Die Relais der Ausgänge 1,2,3,4 sollen übersteuert werden. Ausgänge 1 und 4 sollen anziehen, Ausgänge 2 und 3 sollen abschalten.
Das erste Byte bestimmt die zu übersteuernden Ausgänge gemäß Tabelle 7:
`0x01 + 0x02 + 0x04 + 0x08 = 0x0F`
Das zweite Byte legt fest welches der übersteuerten Relais anzieht, und welches abfällt.
`1*0x01 + 0*0x02 + 0*0x04 + 1*0x08 = 0x09`

Setze `nviMiscCfg` auf :

00	00	4B	0F	09	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Nach dem Rücklesen der `nviMiscCfg` befindet sich im Feld `.configType` eine Rückmeldung über den Erfolg des Vorgangs. Es wird folgendes gelesen:

00	00	FD	0F	09	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Um zu überprüfen wie die Relais tatsächlich stehen (vgl. 3.2.4) setze `nviMiscCfg` auf :

02	00	47	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Nach dem Rücklesen der `nviMiscCfg` befindet sich im Feld `.configType` eine Rückmeldung über den Erfolg des Vorgangs. Es wird folgendes gelesen:

02	00	FC	42	00	49	0F	49	49	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Da das erste Byte `0x42` ist, ist sind die Kontakte 2 und 7 an den Eingängen geschlossen.

Das zweite Byte zeigt an, daß die Notbedienebene nicht aktiviert wurde.
Im dritten Byte steht: `0x49=0x40 + 0x08 + 0x01` D.h. Ausgang 7,4 und 1 wurden angesteuert.
Am vierten Byte `0x0F` kann man erkennen, daß die Ausgänge 4,3,2,1 durch Parametrierung übersteuert wurden.
Das fünfte Byte zeigt den Sollwert der Relais. Es sollen die Ausgänge 7,4 und 1 angesteuert werden.
Am sechsten Byte erkennt man, daß alle anzusteuernden Ausgänge bereits angesteuert wurden.

4 Sensor Object

Das Sensor Objekt liefert Werte aus den digitalen Eingängen. Jedem Sensor Objekt ist genau ein digitaler Eingang zugeordnet. Die Zuordnung kann softwareseitig nicht geändert werden. Die Sensor Objekte der Software 1691111a sind als modifizierte Functional Profiles #3200 – Switch implementiert. Im Folgenden wird für das Sensor Objekt auch der Begriff Switch Objekt verwendet.

4.1 Schnittstellenbeschreibung

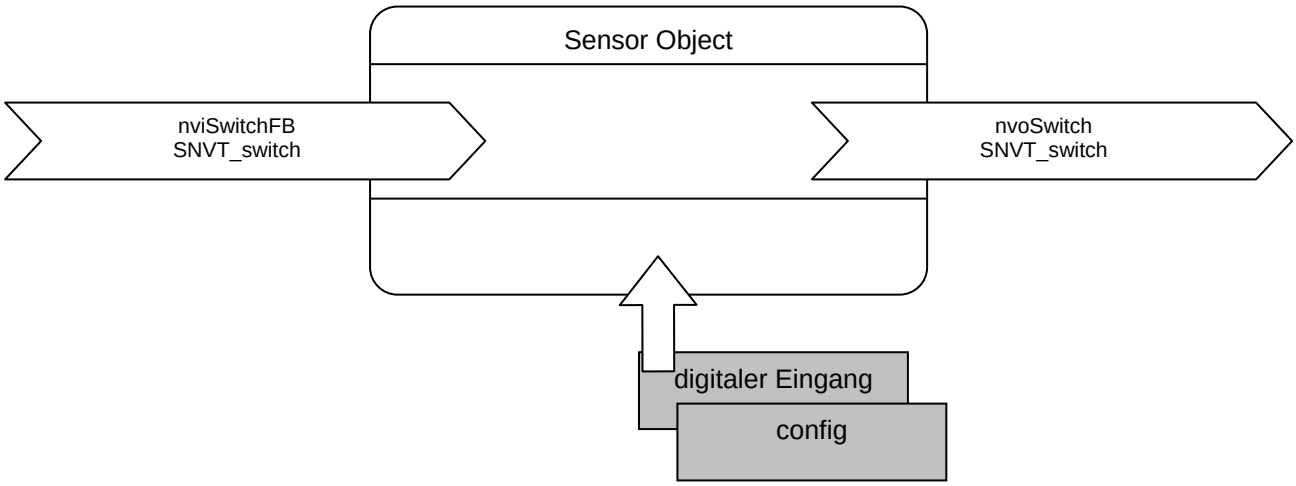


Abbildung 1 (Sensor Objekt)

4.1.1 Value Output

`network output SNVT_switch nvoSwitch;`
Diese Variable liefert die Werte des Sensor Objekts in Abhängigkeit von seiner Konfiguration, dem Zustand des zugeordneten digitalen Eingangs und der zugeordneten Feedbackvariablen.

Valid Range

Der Wert dieser Variablen wird durch die *SNVT Master List* festgelegt.
Im Folgenden werden die Begriffe „AN“ und „AUS“ verwendet.
Sie entsprechen einer `SNVT_switch` Ausgangsvariablen hier beispielsweise `nvoSwitch` mit folgenden Werten:

verwendete Abkürzung	`nvoSwitch.value`	`nvoSwitch.state`
AN	`0xC8 (dezimal:200)`	`0x01 (dezimal: 1)`
AUS	`0x00 (dezimal:0)`	`0x00 (dezimal:0)`

Tabelle 8: Definition von AN und AUS für Value Output

Default Value

Nach dem Reset oder dem Anlegen der Betriebsspannung werden alle Sensor Objekte in Abhängigkeit von digitalem Eingang und der Feedbackvariable neu berechnet. Der ermittelte Wert wird dann über das Netzwerk gesendet.

When Transmitted

Die Daten werden übertragen sobald sich am digitalen Eingang, oder am Eingang der Feedbackvariablen eine Änderung ergibt. Ist eine Ausschaltverzögerung parametriert, so wird auch die Netzwerkvariable verzögert aktualisiert. Abhängig von der Konfiguration der Funktion, die später beschrieben wird, führt eine Änderung an der Feedbackvariablen nicht zwangsläufig zu einem erneuten Senden der Ausgangsvariablen. Ein Timer zur Sendewiederholung (Heartbeat) kann parametriert werden.

Default Service Type

Der Default Service Type ist acknowledged.

4.1.2 Value Feedback

`network input SNVT_switch nviSwitchFB;`
Diese Variable dient der Rückführung eines Aktorzustandes auf das Sensor Objekt. Abhängig davon wird dann der Wert der Ausgangsvariable neu berechnet. Bei der Parametrierung als logisches Objekt, z.B. als UND-Glied, wird auf diesen Feedback-Eingang der Ausgang des vorhergehenden Gliedes gebunden.

Valid Range

Der Wert dieser Variablen wird durch die *SNVT Master List* festgelegt.
Eine Feedbackvariable vom Typ `SNVT_switch`, wird in vielen v.a. logischen Funktionen des Sensor Objekts genutzt. Wie schon bei dem Value Output wird auch hier die Abkürzung „AN“ und „AUS“ verwendet. Eingelesene Zustände werden folgendermaßen interpretiert:

verwendete Abkürzung	`nvoSwitch.value`	`nvoSwitch.state`
AN	`0xC8 (dezimal:200)`	`0x01 (dezimal: 1)`
AUS	alle anderen Zustände	alle anderen Zustände

Tabelle 9: Definition von AN und AUS für Feedback Eingang

Default Value

Beim Reset werden alle Feedback-Eingänge folgendermaßen gesetzt:
`nviSwitchFB.value=0`
`nviSwitchFB.state=0`

4.2 Parametrierung

4.2.1 Unterstützte Dienste

Über die Parametrierschnittstelle, der Netzwerkvariablen `nviMiscCfg`, werden folgende Dienste unterstützt:

Dienst	`nviMiscCfg.request`	Bedeutung
SCF_SAVE	`0x00`	Konfiguration überschreiben mit neuen Daten
SCF_REPORT	`0x02`	Abrufen der Konfiguration

Tabelle 10: Unterstützte Dienste des Sensor Objekts

4.2.2 Unterstützte Objekte

Die Software 1691111a unterstützt bis zu 8 Sensor Objekte, deren Objektnummern in folgender Tabelle dargestellt sind.

Objektbezeichnung	Objektindex nviRequest.object_id	Klemmenpaar
Switch Object 0	0x01	Schalter 1: X9.1 – X9.2
Switch Object 1	0x02	Schalter 2: X9.3 – X9.4
Switch Object 2	0x03	Schalter 3: X9.5 – X9.6
Switch Object 3	0x04	Schalter 4: X9.7 – X9.8
Switch Object 4	0x05	Schalter 5: X9.9 – X9.10
Switch Object 5	0x06	Schalter 6: X9.11 - X9.12
Switch Object 6	0x07	Schalter 7: X9.13 - X9.14
Switch Object 7	0x08	Schalter 8: X9.15 - X9.16

Tabelle 11: Unterstützte Sensor Objekte

4.2.3 Parametrierbare Optionen

Folgende Tabelle faßt alle Konfigurationsmöglichkeiten des Sensor Objekts zusammen.

Element	nviMiscCfg. configType	Bedeutung
SwitchConfig	0x02	Funktion des Sensor Objekts als Schalterinterface, Tasterinterface, UND-Glied, ODER-Glied ...
ulTimerHBSwitchConfig	0x03	Sendewiederholung
ulTimer1SwitchConf	0x04	Ausschaltverzögerung
strDescription	0x05	frei wählbarer Text z.B. für Montageort

Tabelle 12: Parametrierbare Optionen des Sensor Objekts

4.2.4 Funktion - SwitchConfig

Die parametrierbare Konfiguration SwitchConfig bestimmt das grundlegende Verhalten des Sensor Objekts. Mit ihr kann das Verhalten des Sensor Objekts auf den Anschluß eines Schalters, eines Tasters, ... festgelegt werden. Außerdem stehen verschiedene logische Funktionen zur Auswahl. Mit Hilfe dieser Funktionen können die verschiedensten Aufgaben realisiert werden. Einige Beispiele hierzu werden später folgen.

Funktion	nviMiscCfg .data	Bedeutung
ModeSWITCH	0x00	Zum Anschluß eines Schalters
ModePUSHBUTTON	0x01	Zum Anschluß eines Tasters
ModeSWITCH_DAISYCHAIN_AND	0x02	logisches UND-Glied
ModeSWITCH_DAISYCHAIN_OR	0x03	logisches ODER-Glied
ModeSWITCH_DAISYCHAIN_XOR	0x04	logisches EXKLUSIV-ODER-Glied
ModeSWITCH_DAISYCHAIN_NAND	0x05	logisches negiertes UND-Glied
ModeSWITCH_DAISYCHAIN_NOR	0x06	logisches negiertes ODER-Glied
ModeSWITCH_DAISYCHAIN_NXOR	0x07	logisches negiertes EXKLUSIV-ODER-Glied

Tabelle 13: Funktionen des Sensor Objekts

4.2.4.1 modeSWITCH(0x00)

Dieser Modus ist vorgesehen für den Anschluß eines Schalters. Der Ausgang verhält sich wie der eines Wechselschalters. Der Wert der Ausgangsvariablen nvoSwitch wird bei jeder Änderung des Schaltzustandes neu überprüft und gegebenenfalls geändert. Eine Änderung am Feedback-Eingang führt nicht zu einem erneuten Versenden des Ausgangswertes. Intern wird in diesem Fall Ausgangsvariable jedoch verändert, so daß das Polling der Ausgangsvariable bereits einen

veränderten aber nicht gesendeten Wert anzeigen kann. Der Vorgang des Pollens wird z.B. durch Variablenbrowser von Integrationstools durchgeführt. Bei der Fehlersuche in einem gebundenen Netzwerk sollte deshalb besser die Eingangsvariable des gebundenen Objekts betrachtet werden. Wird der Ausgang durch die Software umgeschaltet, so wird intern auch die Feedbackvariable geändert, ohne daß dies über das Netzwerk übertragen wird. Durch das Polling der Feedbackvariablen, kann man erkennen ob eine Aktualisierung der Feedbackvariablen zwischenzeitlich erfolgt ist. Solange dies nicht geschehen ist, wird:

`nviSwitchFB.state = 0xFF (dezimal -1)`
`nviSwitchFB.value` enthält bis zum Eintreffen des Feedbacks den Wert von `nvoSwitch.value`.

Eine ungebundene Feedbackvariable bewirkt, daß mit jeder Änderung des Schalterzustands auch die Ausgangsvariable `nvoSwitch` geändert wird. Das Gleiche passiert, wenn der Feedbackeingang zwar gebunden ist, aber keine Aktualisierung der Feedbackvariable erfolgt.
Wird die Feedbackvariable jedoch aktualisiert, so wird der neue Zustand nach Änderung des Schalterzustands abhängig von `nviSwitchFB.value` dieser Variable bestimmt. Enthält sie einen Wert größer 0 so wird der Zustand AUS ausgegeben im anderen Fall AN.

HINWEIS:
Ist in diesem Modus der Heartbeat aktiviert, so ist es erforderlich, daß die Feedbackvariable auf den entsprechenden Feedbackausgang des angesteuerten Aktors gebunden wird. Ist dies nicht der Fall, so kann im Falle von mehreren Sensor Objekten, die einen Aktor ansteuern ein unerwünschtes Verhalten auftreten, das sich darin äußert, daß ein Sensor Objekt den Aktor ansteuert, ein anderes das den Befehl dieses ersten Sensor Objekts nicht wahrgenommen hat, den Aktor aber nach Ablauf seines Heartbeattimers wieder abschaltet. Erlangt dieses zweite Sensor Objekt über seinen Feedbackeingang jedoch Kenntnis von dem neuen Zustand des Aktors, so wird es diesen nach Ablauf seines Heartbeattimers genauso anschalten, bzw. abschalten, wie es das erste Sensor Objekt getan hat.
Im Allgemeinen ist von der Verwendung des Heartbeattimers abzuraten, wenn es sich bei dem Binding um eine Closed Loop handelt. Durch die Laufzeiten, die eine Netzwerknachricht hat, kann es unter Verwendung von automatischer Sendewiederholung durch Heartbeats und Feedbacks wie sie in Closed Loops eingesetzt werden zu ungewünschten Seiteneffekten kommen, vor allem dann, wenn mehrere Sensorobjekte auf einen oder mehrere Aktoren gebunden werden.

4.2.4.2 modePUSHBUTTON(0x01)

Dieser Modus ist vorgesehen für den Anschluß eines Tasters. Der Ausgang verhält sich analog zum `modeSWITCH(0x00)`. Der Wert der Ausgangsvariablen `nvoSwitch` wird jedoch bei jedem Druck des Tasters neu überprüft und gegebenenfalls geändert. Ein Loslassen des Tasters bewirkt keine Änderung.

4.2.4.3 modeSWITCH_DAISYCHAIN_AND(0x02)

Dieser Modus gibt den Zustand AN aus, wenn sowohl der Taster / Schalter /Fensterkontakt geschlossen ist, als auch der Feedbackeingang `nviSwitchFB` den Zustand AN empfangen hat. In allen anderen Fällen wird AUS ausgegeben.
Tabellarisch dargestellt ergibt sich die Funktion eines UND-Gatters:

Schalter- Tasterzustand	Wert von: `NviSwitchFB`	Wert von: `nvoSwitch`
Offen	AUS	AUS
Offen	AN	AUS
Geschlossen	AUS	AUS

Geschlossen	AN	AN

Tabelle 14: Wahrheitstabelle logische UND- Funktion

4.2.4.4 modeSWITCH_DAISYCHAIN_OR(0x03)

Dieser Modus gibt den Zustand AN aus, wenn entweder der Taster / Schalter geschlossen ist, oder/und der Feedbackeingang `nviSwitchFB` den Zustand AN empfangen hat. Ist weder der Taster / Schalter geschlossen, noch die Feedbackvariable AN, wird AUS ausgegeben. Tabellarisch dargestellt ergibt sich die Funktion eines ODER-Gatters:

Schalter- Tasterzustand	Wert von: `nviSwitchFB`	Wert von: `nvoSwitch`
Offen	AUS	AUS
Offen	AN	AN
Geschlossen	AUS	AN
Geschlossen	AN	AN

Tabelle 15: Wahrheitstabelle logische ODER- Funktion

4.2.4.5 modeSWITCH_DAISYCHAIN_XOR(0x04)

Dieser Modus gibt den Zustand AN aus, wenn entweder der Taster / Schalter geschlossen ist und der Feedbackeingang `nviSwitchFB` den Zustand AUS empfangen hat, oder der Taster / Schalter offen ist und der Feedbackeingang `nviSwitchFB` den Zustand AN empfangen hat. In allen anderen Fällen wird AUS ausgegeben.
Tabellarisch dargestellt ergibt sich die Funktion eines EXKLUSIV-ODER-Gatters:

Schalter- Tasterzustand	Wert von: `nviSwitchFB`	Wert von: `nvoSwitch`
Offen	AUS	AUS
Offen	AN	AN
Geschlossen	AUS	AN
Geschlossen	AN	AUS

Tabelle 16: Wahrheitstabelle der logischen EXKLUSIV-ODER-Funktion

4.2.4.6 modeSWITCH_DAISYCHAIN_NAND(0x05)

Dieser Modus verhält sich ähnlich wie der Modus `modeSWITCH_DAISYCHAIN_AND(0x02)`. Die Ausgangsvariable nimmt jedoch den gegenteiligen Zustand an.
Der Modus gibt den Zustand AUS aus, wenn sowohl der Taster / Schalter geschlossen ist, als auch der Feedbackeingang `nviSwitchFB` den Zustand AN empfangen hat. In allen anderen Fällen wird AN ausgegeben.
Tabellarisch dargestellt ergibt sich die Funktion eines NEGIERT-UND-Gatters:

Schalter- Tasterzustand	Wert von: `nviSwitchFB`	Wert von: `nvoSwitch`
Offen	AUS	AN
Offen	AN	AN

Geschlossen	AUS	AN
Geschlossen	AN	AUS

Tabelle 17: Wahrheitstabelle der logischen NEGIERT-UND-Funktion

4.2.4.7 modeSWITCH_DAISYCHAIN_NOR(0x06)

Dieser Modus verhält sich ähnlich wie der Modus `modeSWITCH_DAISYCHAIN_OR(0x03)`. Die Ausgangsvariable nimmt jedoch den gegenteiligen Zustand an.
Der Modus gibt den Zustand AUS aus, wenn entweder der Taster / Schalter geschlossen ist, oder/und der Feedbackeingang `nviSwitchFB` den Zustand AN empfangen hat. Ist weder der Taster / Schalter geschlossen, noch die Feedbackvariable AN, wird AN ausgegeben.
Tabellarisch dargestellt ergibt sich die Funktion eines NEGIERT-ODER-Gatters:

Schalter- Tasterzustand	Wert von: `nviSwitchFB`	Wert von: `nvoSwitch`
Offen	AUS	AN
Offen	AN	AUS
Geschlossen	AUS	AUS
Geschlossen	AN	AUS

Tabelle 18: Wahrheitstabelle der NEGIERT-ODER-Funktion

4.2.4.8 modeSWITCH_DAISYCHAIN_NXOR(0x07)

Dieser Modus verhält sich ähnlich wie der Modus `modeSWITCH_DAISYCHAIN_XOR(0x04)`. Die Ausgangsvariable nimmt jedoch den gegenteiligen Zustand an.
Der Modus gibt den Zustand AUS aus, wenn entweder der Taster / Schalter geschlossen ist und der Feedbackeingang `nviSwitchFB` den Zustand AUS empfangen hat, oder der Taster / Schalter offen ist und der Feedbackeingang `nviSwitchFB` den Zustand AN empfangen hat. In allen anderen Fällen wird AN ausgegeben.
Tabellarisch dargestellt ergibt sich die Funktion eines NEGIERT-EXKLUSIV-ODER-Gatters:

Schalter- Tasterzustand	Wert von: `nviSwitchFB`	Wert von: `nvoSwitch`
Offen	AUS	AN
Offen	AN	AUS
Geschlossen	AUS	AUS
Geschlossen	AN	AN

Tabelle 19: Wahrheitstabelle der NEGIERT-EXKLUSIV-ODER-Funktion

4.2.4.9 Beispiele zur Parametrierung der Funktion

An dieser Stelle soll die Konfiguration der Sensor Objekte in einigen Beispielen durchgeführt werden. Bezugnehmend auf die Beschreibung des Node Objekts mit der zur Parametrierung verwendeten Netzwerkvariable `nviMiscCfg` können die Beispiele hier auf die Auswahl der zu sendenden Parameter beschränkt werden.

Beispiel 1:
An Sensor Objekte 2 bis 4 sollten Taster für eine Wechselschaltung angeschlossen werden. Die entsprechenden Objekte sollen also als Tasterinterface konfiguriert werden.
Gemäß Tabelle 10 wählen wir den Dienst: `0x00`
Gemäß Tabelle 11 ermitteln wir für die Sensor Objekte die Objektnummern: `0x03,0x04,0x05`
Gemäß Tabelle 12 wählen wir die Parametrierung der Funktion: `0x02`
Gemäß Tabelle 13 wählen wir die Funktion des Tasters: `0x01`
Wir setzen `nviMiscCfg` auf :

00	03	02	01	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Um den Erfolg der Parametrierung zu testen lesen wir `nviMiscCfg` zurück und erhalten:

00	03	FD	01	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Ebenso verfahren wir mit den anderen beiden Sensor Objekten.
Schreibe `nviMiscCfg` mit:

00	04	02	01	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Lese `nviMiscCfg:`

00	04	FD	01	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Schreibe `nviMiscCfg` mit:

00	05	02	01	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Lese `nviMiscCfg:`

00	05	FD	01	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Beispiel 2:
Bei der lange zurückliegenden Inbetriebnahme wurde das Sensor Objekt 0 konfiguriert. Nun soll überprüft werden welche Funktion parametriert wurde.
Gemäß Tabelle 10 wählen wir den Dienst: `0x02` um die Daten auszulesen
Gemäß Tabelle 11 ermitteln wir für das Sensor Objekt 0 die Objektnummer: `0x01`
Gemäß Tabelle 12 wählen wir die Parametrierung der Funktion: `0x02`
Alle weiteren Daten der `nviMiscCfg` müssen nicht belegt werden.
Wir schreiben in die `nviMiscCfg`:

02	01	02	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Als nächsten Schritt pollen wir die `nviMiscCfg` und erhalten:

02	01	FC	04	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Zum Entschlüsseln verwenden wir wieder obige Tabellen:
02: Die Anforderung zum Lesen der Konfiguration gemäß Tabelle 10
01: Die Objektnummer gemäß Tabelle 11
FC: Die Bestätigung, daß das Auslesen der Konfiguration erfolgreich war, gemäß Tabelle 3
04: Die konfigurierte Funktion als EXKLUSIV-ODER-Glied gemäß Tabelle 13

4.2.5 Senden mit Heartbeat - ulTimerHBSwitchConfig

Jedem Sensor Objekt ist ein Timer zugeordnet, der es ermöglicht in zyklischen Abständen den Ausgangswert automatisch zu senden, ohne daß dafür eine Änderung des zugeordneten Schalters, Tasters, oder der Feedbackvariablen zugrunde liegen muß. Es existiert eine Konfigurationsvariable, die das Intervall dieser zyklischen Sendevorgänge festlegt.
Ist der Timer für den Heartbeat auf einen Wert größer als `0x0000` konfiguriert, so gibt dieser Wert den Zeitabschnitt in Sekunden an, der maximal verstreicht, bis der Ausgangswert erneut gesendet wird. Wird der Wert des Ausgangs aufgrund einer geänderten Schalterstellung, eines Tastendrucks, oder einer zwischenzeitlich empfangenen Feedbackvariablen erneut berechnet und gesendet, so wird die Zeit bis zum nächsten automatischen Senden neu gestartet. Wird der Wert auf `0x0000` konfiguriert (Werkseinstellung), so ist das automatisch wiederholte Senden abgeschaltet. Der Wertebereich ist der einer vorzeichenlosen 16-Bit Integerzahl. Der maximale Wert beträgt damit 65535s (`0xFFFF`). Beim Schreiben auf die `nviMiscCfg` wird immer ein 16-Bit Wert erwartet. Das höherwertige Byte muß zuerst bereit gestellt werden.
Mit der Möglichkeit Heartbeats zu senden sollte sehr sparsam umgegangen werden, damit die Netzbelastung gering bleibt. Sinnvoll ist diese Einstellung für Sensoren, die zyklisch Meßwerte an viele andere Objekte liefern sollen, oder für Objekte, die sicherheitsrelevante Daten zur Verfügung stellen. Es sollte in Betracht gezogen werden, ob in diesem Fall der Default Service Type der Ausgangsvariablen auf unacknowledged geändert werden kann.

Beispiel 1:
Das Sensor Objekt 7 (Objektindex `0x08`) soll mindestens alle 10 Minuten seine Ausgangsvariable senden. 10 Minuten sind 600s (`0x258`).
Wir beschreiben `nviMiscCfg` mit:

00	08	03	02	58	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Wir lesen `nviMiscCfg` aus und erhalten:

00	08	FD	02	58	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Beispiel 2:
Der Heartbeat von Sensor Objekt 0 (Objektindex `0x01`) soll abgeschaltet werden.
Wir beschreiben `nviMiscCfg` mit:

00	01	03	00	00	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Wir lesen `nviMiscCfg` aus und erhalten:

00	01	FD	00	00	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Beispiel 3:
Es soll die Zeit für die automatische Sendewiederholung des Sensor Objektes 5 (Objektindex `0x06`) ermittelt werden.
Wir beschreiben `nviMiscCfg` mit:

02	06	03	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Wir lesen `nviMiscCfg` aus und erhalten:

02	06	FC	00	B4	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Die ausgelesenen Daten sind: 0x00B4 (180 dezimal). Das entspricht 3 Minuten.

4.2.6 Zeitverzögerung mit Timer1 - ulTimer1SwitchConf

Der Einsatz des Timers1 ermöglicht das verzögerte Abschalten der `nvoSwitch`. Diese Funktion kann in vielen Anwendungen eingesetzt werden. Denkbar ist beispielsweise die akustische Meldung von kurzzeitigen Fehlerzuständen, oder die Anzeige von Fensterkontakten. Soll ein geöffneter Fensterkontakt mindestens für 30s angezeigt werden, auch wenn das Fenster nur für wenige Sekunden geöffnet wurde, so kann das Sensor Objekt durch die Verwendung von Timer1 so konfiguriert werden, daß die Anzeige noch 30s nach dem Schließen des Fensters aktiv bleibt. Stellt sich ein erneuter Einschaltzustand an den Eingängen des Sensor Objektes ein, während die Ausschaltverzögerung noch läuft, so wird die Ausschaltverzögerung gestoppt und erst beim nächsten Ausschaltzustand erneut mit der vollen Verzögerungszeit gestartet. Die Verzögerung bezieht sich immer auf den Übergang vom aktiven in den inaktiven Zustand. Sind logische Funktionen mit invertiertem Verhalten (NAND, NOR und NXOR) paramteriert, so bewirkt die Parametrierung eines Wertes ungleich 0 für Timer1 ein verzögertes Schalten in den Zustand AN, welcher bei diesen Funktionen dem inaktiven Zustand entspricht.
Die Ausschaltverzögerung kann in Sekundenintervallen parametriert werden. Die maximale Dauer beträgt: 65535s (`0xFFFF`). Soll die Ausschaltverzögerung deaktiviert werden, so muß der Wert in `Timer1` auf die Werkseinstellung 0 (`0x0000`) konfiguriert werden. Beim Schreiben auf die `nviMiscCfg` wird immer ein 16-Bit Wert erwartet. Das höherwertige Byte muß zuerst bereit gestellt werden.
Abbildung 2 soll das an einem als Schalter konfigurierten Sensor Objekt grafisch darstellen:

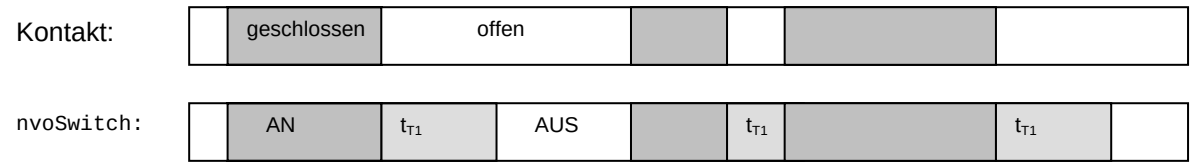


Abbildung 2: Ausschaltverzögerung am Sensor Objekt mit Timer1

In der Abblidung ist von links nach rechts die Zeit aufgetragen. Die grau hinterlegten Balken stellen verschiedene Zustände des Schalters und der `nvoSwitch` dar. Nachdem der Schaltkontakt geöffnet wurde, beginnt die Ausschaltverzögerung t_{T1} abzulaufen. Während dieser Zeit hat `nvoSwitch` noch den Wert AN. Erst wenn die parametrierte Zeit in `Timer1` abgelaufen ist, kippt der Ausgangszustand auf AUS. Zwischen der zweiten und dritten Schließung des Kontaktes kann der Timer1 nicht vollständig ablaufen, also erreicht die Variable `nvoSwitch` nie den Wert AUS. Nach dem letzten Öffnen des Kontaktes verstreicht die volle Periode von Timer1 bevor der Wert AUS ausgegeben wird.

HINWEIS:
Soll eine Ausschaltverzögerung an einem Aktor realisiert werden, so ist es geschickter diesem die Ausschaltverzögerung zu parametrieren. Soll eben dieser Aktor von mehreren Sensoren angesprochen werden, so muß die Zeit für die Ausschaltverzögerung nur ein einziges Mal konfiguriert werden. Wirkt nur ein Sensor auf einen, oder mehrere Aktoren, so spielt es keine Rolle welches Objekt die Ausschaltverzögerung realisiert.

Beispiel 1:
Das Sensor Objekt 7 (Objektindex `0x08`) sei als Schalter konfiguriert und soll nach dem Öffnen des Kontaktes noch 6 Minuten warten, bevor die Ausgangsvariable AUS sendet. 6 Minuten sind 360s (`0x168`).
Wir beschreiben `nviMiscCfg` mit:

00	08	04	01	68	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Wir lesen `nviMiscCfg` aus und erhalten:

00	08	FD	01	68	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Beispiel 2:
Der `Timer1` von Sensor Objekt 0 (Objektindex `0x01`) soll abgeschaltet werden.
Wir beschreiben `nviMiscCfg` mit:

00	01	04	00	00	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Wir lesen `nviMiscCfg` aus und erhalten:

00	01	FD	00	00	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Beispiel 3:
Es soll die Zeit für die Ausschaltverzögerung des Sensor Objektes 5 (Objektindex `0x06`) ermittelt werden.
Wir beschreiben `nviMiscCfg` mit:

02	06	04	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Wir lesen `nviMiscCfg` aus und erhalten:

02	06	FC	00	0A	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Die ausgelesenen Daten sind: 0x000A (10 dezimal). Das entspricht 10 Sekunden.

4.2.7 Einspeichern eines Textes – strDescription

Die Software 1691111a erlaubt es, jedem Sensor Objekt einen beliebigen Text bis zu 20 Zeichen, ohne terminierendes NULL-Zeichen, zuzuordnen. Dieser Text wird im Gerät gespeichert. Üblicherweise werden dort Informationen über den Installationsort des Bedienelements gespeichert. Bei der Auswertung von Fensterkontakten kann darin beispielsweise die Nummer des Fensters gespeichert werden.
Normalerweise werden die Ziffern im ASCII-Format codiert. Zeichenfolgen müssen mit einem NULL-Zeichen (0x00) terminiert werden um das Ende der Zeichenkette anzuzeigen. Man beachte, daß das NULL-Zeichen nicht der Ziffer „0“ im ASCII-Code entspricht, welche den Code (0x30) besitzt.
Beim Auslesen der Information werden alle 20 möglichen Zeichen und das terminierende NULL-Zeichen ausgegeben. Wurde eine Zeichenkette vorzeitig beendet, erkennt man das an einem NULL-Zeichen vor der 21. Stelle im Datenbereich. Es steht dem Benutzer frei auf die ASCII-Codierung zu verzichten und beliebige Hexadezimalzahlen zu verwenden. Allerdings kann man dann das Ende der Zeichenfolge nicht mehr eindeutig erkennen.

Beispiel 1:
Es soll im Sensor Objekt 4 hinterlegt werden, daß sich ein zugehöriger Fensterkontakt im Ostflügel, in der 3. Etage am Fenster mit der Nummer 207 befindet. Folgender Text soll im ASCII-Format abgespeichert werden: „Fenster@Ost#3-207“. Abgeschlossen wird der Text mit einem NULL-Zeichen.
Im ASCII-Format ergibt sich:
0x46 0x65 0x6E 0x73 0x74 0x65 0x72 0x40 0x4F 0x73 0x74 0x23 0x33 0x2D 0x32 0x30 0x37 0x00
Wir beschreiben `nviMiscCfg` mit:

00	05	05	46	65	6E	73	74	65	72	40	4F	73	74	23	33	2D	32	30	37	00	XX	XX	XX	XX	XX	XX	XX	XX	XX

Wir lesen `nviMiscCfg` aus und erhalten:

00	05	FD	46	65	6E	73	74	65	72	40	4F	73	74	23	33	2D	32	30	37	00	XX	XX	XX	XX	XX	XX	XX	XX	XX

Beispiel 2:
Der Text, der dem Sensor Objekt 6 als ASCII-Text hinterlegt wurde, soll ausgelesen werden.
Wir beschreiben `nviMiscCfg` mit:

02	07	05	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Wir lesen `nviMiscCfg` aus und erhalten:

02	07	FC	57	30	32	45	38	33	30	30	00	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	00	XX	XX	XX	XX	XX	XX	XX

Nach der Rückkonvertierung aus dem ASCII-Code ergibt sich: „W02E8300“.

Anmerkung: könnte man nicht davon ausgehen, daß es sich um einen NULL-Zeichen terminierten ASCII-Code handelt, müßten alle Zeichen bis zu 0x00 an der 21. Stelle des Datenbereiches interpretiert werden.

5 Actuator Object

Das Aktor Objekt steuert das ihm zugehörige Relais an. Die Zuordnung zwischen Relais und Aktor Objekt ist fix und kann nicht durch Parametrieren geändert werden. Die Aktor Objekte der Software 1691111a sind als modifizierte Functional Profiles #3040 – LampActuator implementiert. Aufgrund der Modifizierung können diese Aktor Objekte sehr universell, also auch für andere Steueraufgaben eingesetzt werden.

5.1 Schnittstellenbeschreibung

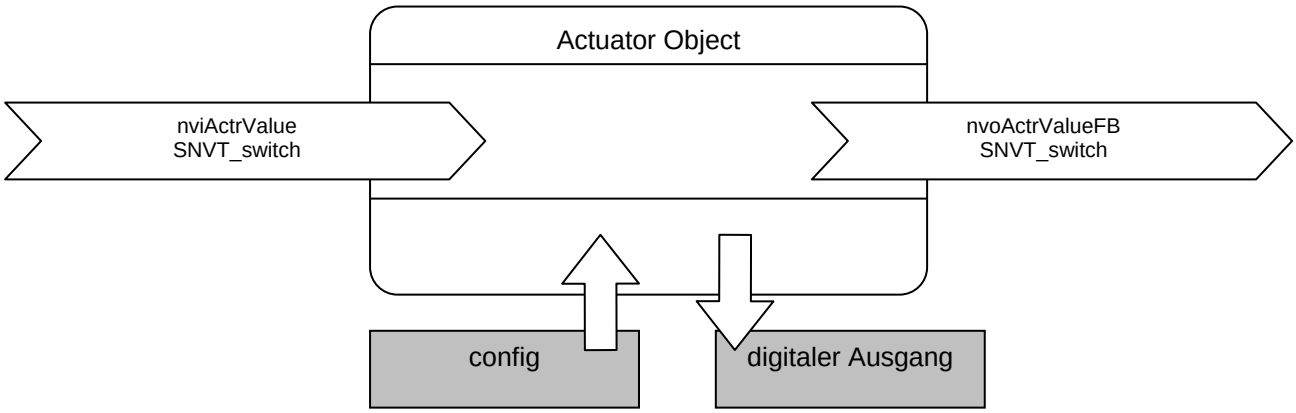


Abbildung 3 (Aktor Objekt)

5.1.1 Value Input

`network input SNVT_switch nviActrValue;`

Diese Variable stellt den Steuereingang des Aktor Objekts dar. Mit Hilfe der Konfigurationsparameter, die für dieses Objekt zur Verfügung stehen, kann Einfluß auf das Schaltverhalten genommen werden. Das resultierende Verhalten wird in den Kapiteln zu den einzelnen Parametern genauer erklärt. Abhängig von diesen Parametern und den empfangenen Werten an dieser Variablen wird das zugehörige Relais angesteuert und die Feedbackvariable entsprechend gesetzt.

Valid Range

Der Wert dieser Variablen wird durch die *SNVT Master List* festgelegt.
Im Folgenden werden die Begriffe „AN“ und „AUS“ verwendet.
Sie entsprechen einer `SNVT_switch` Eingangsvariablen hier beispielsweise `nviActrValue` mit folgenden Werten:

verwendete Abkürzung	`nviActrValue.value`	`nviActrValue.state`
AN	`größer 0x00`	`größer 0x00`
AUS	`Dont Care`	`0x00`
AUS	`0x00`	`egal`

Tabelle 20: Definition von AN und AUS für Value Eingang

Default Value

Nach dem Reset oder dem Anlegen der Betriebsspannung werden alle Aktorwerte folgendermaßen gesetzt:
`nviActrValue.value=0`
`nviActrValue.state=0`

Dementsprechend fallen die Relais ab. Erst wenn ein neuer Wert empfangen wird, werden die Relais entsprechend angesteuert.

5.1.2 Value Feedback

`network output SNVT_switch nvoActrValueFB;`
Diese Variable dient der Rückführung eines Aktorzustandes, z.B. auf das Sensor Objekt. Wenn z.B. mehrere Taster auf einen Aktor wirken, können sich diese mit Hilfe des Feedbacks synchronisieren. Vorsicht ist geboten, wenn der Feedbackausgang im Zusammenhang mit verzögernden Timern, sei es im Aktor oder in den Sensoren, eingesetzt wird. Laufzeiteffekte können ein unbeabsichtigtes Verhalten bewirken. Dies tritt vor allem dann auf, wenn mehrere Sensor Objekte noch nicht vollständig synchronisiert sind, bevor ein neuer Schaltbefehl ausgelöst wird.

Valid Range

Der Wert dieser Variablen wird durch die *SNVT Master List* festgelegt.
Wie schon bei dem Value Input wird auch hier die Abkürzung „AN“ und „AUS“ verwendet. Es werden folgende Zustände gesendet:

verwendete Abkürzung	`nvoActrValueFB.value`	`nvoActrValueFB.state`
AN	`0xC8 (dezimal:200)`	`0x01 (dezimal: 1)`
AUS	`0x00 (dezimal:0)`	`0x00 (dezimal:0)`

Tabelle 21: Definition von AN und AUS für Feedback Ausgang

Default Value

Beim Reset werden alle Feedback-Ausgänge folgendermaßen gesetzt:
`nvoActrValueFB.value=0`
`nvoActrValueFB.state=0`

When Transmitted

Die Daten werden übertragen, sobald das Relais seinen Schaltzustand ändert. Das bedeutet, daß eine konfigurierte Verzögerungszeit ebenfalls die Aktualisierung des Feedbackausgangs verzögert.

Default Service Type

Der Default Service Type ist acknowledged.

5.2 Parametrierung

5.2.1 Unterstützte Dienste

Über die Parametrierschnittstelle, der Netzwerkvariablen `nviMiscCfg`, werden folgende Dienste unterstützt:

Dienst	nviMiscCfg.request	Bedeutung
SCF_SAVE	0x00	Konfiguration überschreiben mit neuen Daten
SCF_REPORT	0x02	Abrufen der Konfiguration

Tabelle 22: Unterstützte Dienste des Aktor Objekts

5.2.2 Unterstützte Objekte

Die Software 1691111a unterstützt bis zu 8 Aktor Objekte, deren Objektnummern in folgender Tabelle dargestellt sind.

Objektbezeichnung	Objektindex nviRequest.object_id	Klemmenpaar
Aktor Object 0	0x09	Kontakt 1: X1.1 – X1.2
Aktor Object 1	0x0A	Kontakt 2: X1.3 – X1.4
Aktor Object 2	0x0B	Kontakt 3: X4.1 – X4.2
Aktor Object 3	0x0C	Kontakt 4: X4.3 – X4.4
Aktor Object 4	0x0D	Kontakt 5: X6.1 – X6.2
Aktor Object 5	0x0E	Kontakt 6: X6.3 - X9.4
Aktor Object 6	0x0F	Kontakt 7: X7.1 – X7.2
Aktor Object 7	0x10	Kontakt 8: X7.3 – X7.4

Tabelle 23: Unterstützte Aktor Objekte und deren Klemmen

5.2.3 Parametrierbare Optionen

Folgende Tabelle faßt alle Konfigurationsmöglichkeiten des Aktor Objekts zusammen.

Element	nviMiscCfg.configType	Bedeutung
ActrConfig	0x33	Funktion des Aktor Objekts als einfacher Aktor, Monoflop, retriggerbares Monoflop ...
ulTimer1ActrConf	0x34	Ausschaltverzögerung
strDescription	0x35	frei wählbarer Text z.B. für Montageort
CommonParameters	0x36	gemeinsame Parameter

Tabelle 24: Parametrierbare Optionen des Aktor Objekts

5.2.4 Funktion - ActrConfig

Die parametrierbare Konfiguration `ActrConfig` bestimmt das grundlegende Verhalten des Aktor Objekts. Mit ihr kann das Verhalten des Aktor Objekts z.B. als einfaches Aktor Objekt, als Monoflop, als retriggerbares Monoflop, als stopbares Monoflop usw. parametriert werden. Mit Hilfe dieser Funktionen können die verschiedensten Aufgaben realisiert werden. Einige Beispiele hierzu werden später folgen.

Funktion	nviMiscCfg .data	Bedeutung
modeACTRSIMPLE	0x00	Einfacher Aktor
modeACTR_MONO_NORET_NOSTOP	0x01	Einfaches Monoflop
modeACTR_MONO_NORET_STOP	0x02	Stopbares einfaches Monoflop
modeACTR_MONO_RET_NOSTOP	0x03	Retriggerbares Monoflop
modeACTR_MONO_RET_STOP	0x04	Stopbares retriggerbares Monoflop

Tabelle 25: Funktionen des Aktor Objekts

5.2.4.1 modeACTRSIMPLE(0x00)

Wird das Aktor Objekt auf diese Funktion parametriert, so verhält es sich wie ein gewöhnlicher Aktor. Wird an der nviActrValue der Zustand AN empfangen, so zieht der Aktor sein Relais an. Mit dem Anziehen des Relais wird auch die nvoActrValueFB entsprechend auf AN gesetzt. Mit Hilfe des Timer1 kann eine Ausschaltverzögerung eingestellt werden. Ist Timer1 auf 0s parametriert, so ist die Ausschaltverzögerung inaktiv und das Objekt schaltet sofort nach Empfang des AUS-Befehls sein Relais ab. Steht ein Wert in Timer1, so verstreicht erst diese Zeit nach dem Empfang eines AUS-Befehls, bevor das Objekt wirklich ausschaltet.
Standardmäßig ist das Aktor Objekt als einfacher Aktor ohne Ausschaltverzögerung parametriert.
Darstellung des Schaltverhaltens bei parametriertem Timer1 mit der Zeit t_{T1}:

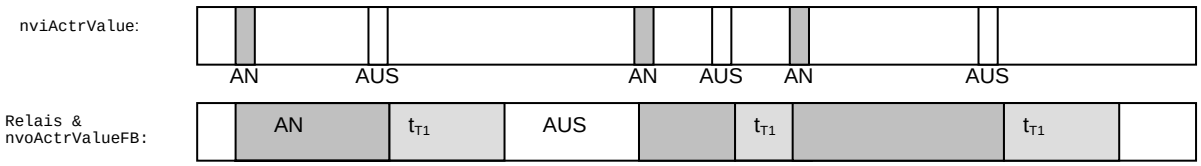


Abbildung 4: Aktor als einfacher Aktor mit parametriertem Timer 1

Mit jedem empfangenen AN-Befehl zieht auch der Aktor sein Relais an und setzt seine Feedbackvariable entsprechend. Die im Timer1 des Aktors gespeicherte Zeit (t_{T1}) muß nach einem empfangenen AUS-Befehl erst ablaufen, bevor der Aktor das Relais wirklich abfallen läßt. Wird vor Ablauf dieser Zeit jedoch ein erneuter AN-Befehl empfangen, so geht der Aktor sofort wieder in den AN-Zustand über, ohne zwischenzeitlich den AUS-Zustand einzunehmen. Die Zeit im Timer1 wird beim nächsten AUS-Befehl neu gestartet.

5.2.4.2 modeACTR_MONO_NORET_NOSTOP (0x01)

Mit dieser Einstellung läßt sich z.B. ein Treppenlicht realisieren, das eine eingestellte Laufzeit lang brennt, sich nicht vorzeitig abschalten läßt und dessen Brenndauer sich auch durch weitere Bedienungen nicht verlängern läßt. Die Brenndauer kann über Timer1 parametriert werden. Zur Auswertung werden nur empfangene AN-Befehle herangezogen. Neben dem Treppenlicht sind auch Anwendungen wie Lüftersteuerungen, die Ansteuerungen von Warnsignalen usw. denkbar. Neben diesem Monoflop gibt es weitere, die im Folgenden beschrieben werden und geringfügig anders funktionieren. Folgende Abbildung soll das zeitliche Verhalten des nicht retriggerbaren und nicht stopbaren Monoflop darstellen:

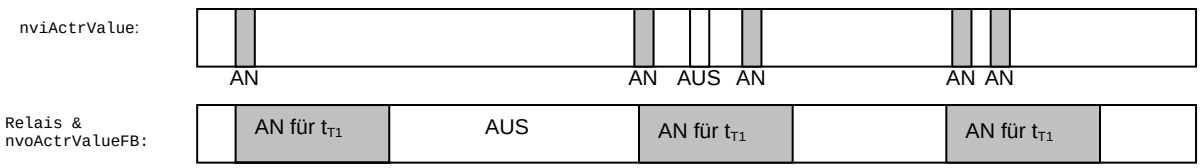


Abbildung 5: Aktor als Monoflop, nicht retriggerbar, nicht stopbar

5.2.4.3 modeACTR_MONO_NORET_STOP (0x02)

Diese Einstellung realisiert ein Monoflop, dessen Laufzeit zwar nicht verlängert, jedoch vorzeitig abgebrochen werden kann. Wie im vorherigen Fall werden auch hier nur AN-Befehle ausgewertet und die Laufzeit in Timer1 parametriert. Zu einem vorzeitigen Abbruch der Laufzeit kommt es, indem innerhalb eines kurzen Zeitraumes (werkseitig 1s) zweimal ein AN-Befehl empfangen wird. Diese Funktion kann sinnvoll für selten benutze Räume eingesetzt werden. Ein Licht schaltet sich spätestens nach Ablauf der Zeit in Timer1 aus, kann aber auch beim Verlassen des Raumes abgeschaltet werden. Folgende Abbildung soll dies grafisch darstellen. Dabei ist zu beachten, daß die beiden AN-Befehle im mittleren Teil des Diagramms zeitlich weiter auseinander liegen, als die parametrierte Zeit für einen Doppeltastendruck dessen Parametrierung später beschrieben wird. Die letzten beiden AN-Befehle des Diagramms liegen eng genug beieinander, damit diese als Doppeltastendruck interpretiert werden können. Damit wird die Laufzeit vorzeitig unterbrochen. Als Schaltelement setzt man einen Taster ein, der an den Klemmen eines als Schalter parametrierten Switch Objekts angeklemmt wird. Da in diesem Beschaltungsfall die Feedbackvariablen des Switch Objekts nicht eingesetzt werden sollen, kann es auch als ODER- Gatter parametriert sein.

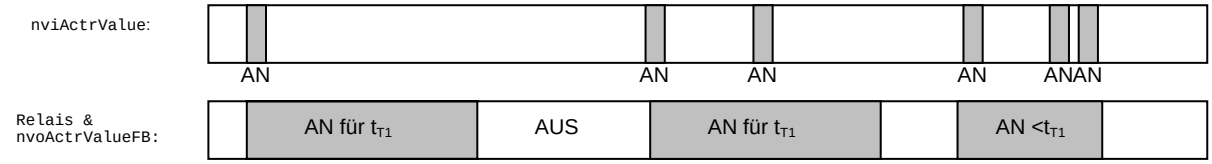


Abbildung 6: Aktor als Monoflop, nicht retriggerbar, aber stoppbar

5.2.4.4 modeACTR_MONO_RET_NOSTOP (0x03)

Soll die Laufzeit eines Treppenlichtes durch wiederholten Tastendruck neu gestartet werden, so kann das Aktor Objekt mit Hilfe einer Parametrierung als retriggerbares Monoflop entsprechend konfiguriert werden. Ebenso wie bei den nicht retriggerbaren Monoflops kann man zwischen einem wählen, dessen Laufzeit nicht vorzeitig abgebrochen werden kann, und einem stopbaren Monoflop. Bei der hier beschriebenen Parametrierung ist ein vorzeitiges Stoppen nicht möglich. Jeder empfangene AN-Befehl, egal welchen zeitlichen Abstand er vom vorhergehenden hat, startet die Laufzeit in Timer1 erneut. Das nächste Diagramm soll dies verdeutlichen.

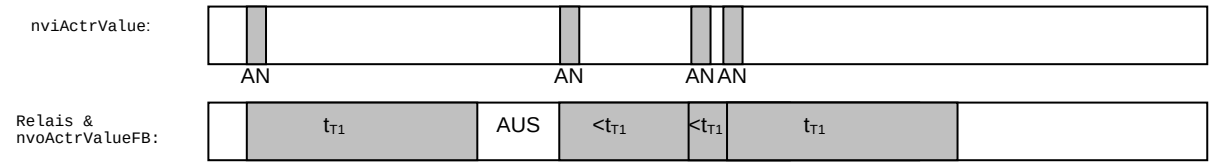


Abbildung 7: Aktor als Monoflop, retriggerbar, nicht stoppbar

Mit jedem empfangenen AN-Befehl wird die Zeit von Timer1 neu gestartet. Erst wenn diese abgelaufen ist, wird der Schaltkontakt geöffnet und die Feedbackvariable sendet AUS.

5.2.4.5 modeACTR_MONO_RET_STOP (0x04)

Dieses Monoflop kann sowohl neu gestartet, als auch gestoppt werden. Ein AN-Befehl startet die Laufzeit in Timer1. Wird nach einer Pause, die länger ist als die parametrierte Zeit für einen Doppeltastendruck, erneut ein AN-Befehl empfangen, so wird Timer1 neu gestartet. Folgen jedoch zwei AN-Befehle zeitlich so dicht nacheinander, daß diese als Doppeltastendruck erkannt werden, so wird die Laufzeit vorzeitig gestoppt. AUS-Befehle haben keine Wirkung. Wirken mehrere Sensoren auf diesen Aktor, so können zwei Tastendrücke z.B. in verschiedenen Etagen eines Treppenhauses den vorzeitigen Abbruch der Laufzeit bewirken, obwohl von den bedienenden Personen eine Verlängerung der Laufzeit beabsichtigt war.
Folgendes Diagramm verdeutlicht das Verhalten:

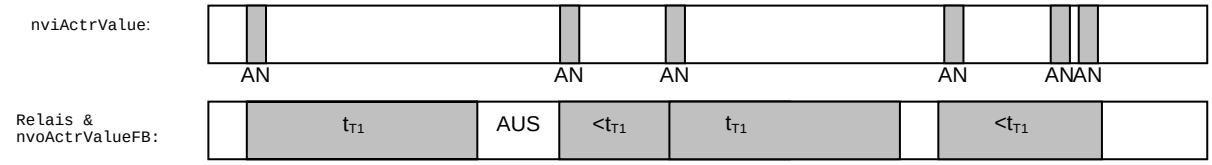


Abbildung 8: Aktor als Monoflop, nachtriggerbar und stopbar

5.2.4.6 Beispiele zur Parametrierung der Funktion

An dieser Stelle soll die Konfiguration der Aktor Objekte in einigen Beispielen durchgeführt werden. Bezugnehmend auf die Beschreibung des Node Objekts mit der zur Parametrierung verwendeten Netzwerkvariable `nviMiscCfg` können die Beispiele hier auf die Auswahl der zu sendenden Parameter beschränkt werden.

Beispiel 1:
Der Aktor 0 soll als einfacher Aktor ohne Ausschaltverzögerung parametriert werden
Wir setzen `nviMiscCfg` auf :

00	09	33	00	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Um den Erfolg der Parametrierung zu testen lesen wir `nviMiscCfg` zurück und erhalten:

00	09	FD	00	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Jetzt muß noch der Timer1 auf 0s (`0x0000`)parametriert werden.
Beispiele hierzu sind im Kapitel des Timer1 aufgeführt.

Beispiel 2:
Ein dreistöckiges Treppenhaus soll mit einer Treppenlichtschaltung ausgestattet werden. Die Lichter in den oberen beiden Stockwerken sollen jeweils 3 Minuten brennen. Das Licht im Erdgeschoß soll hingegen 5 Minuten lang brennen. Wird während der Brenndauer ein Taster erneut gedrückt, so soll die Zeit neu gestartet werden. Werden auf verschiedenen Etagen die Taster fast gleichzeitig betätigt, so darf das Licht nicht erlöschen. Es sollen die Aktoren 5 (EG), 6 (1.Stock) und 7 (2.Stock) verwendet werden. Es werden retriggerbare aber nicht stopbare Monoflops parametriert (`0x03`).
Wir setzen `nviMiscCfg` auf :

00	0E	33	03	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Um den Erfolg der Parametrierung zu testen lesen wir `nviMiscCfg` zurück und erhalten:

00	0E	FD	03	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

um die Funktion des ersten Aktors für das EG zu parametrieren. Dessen Timer1 muß noch auf 5 Minuten (=300s das entspricht 0x012C) parametriert werden. Wie die Timer1 parametriert werden wird im Kapitel zum T
Um den Aktor für den 1. Stock zu parametrieren setzen wir `nviMiscCfg` auf :

00	0F	33	03	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Um den Erfolg der Parametrierung zu testen lesen wir `nviMiscCfg` zurück und erhalten:

00	0F	FD	03	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Der Timer1 dazu muß auf 3 Minuten (=180s das entspricht `0x00B4`) gesetzt werden.
Um den Aktor für den 2. Stock zu parametrieren setzen wir `nviMiscCfg` auf :

00	10	33	03	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Um den Erfolg der Parametrierung zu testen lesen wir `nviMiscCfg` zurück und erhalten:

00	10	FD	03	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Der Timer1 dazu muß ebenfalls auf `0x00B4` gesetzt werden.

Beispiel 3:
Es soll ermittelt werden, welche Funktion Aktor 1 hat.
Wir setzen `nviMiscCfg` auf :

02	0A	33	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Wir lesen `nviMiscCfg` zurück und erhalten:

02	0A	FC	02	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Die ausgelesene Funktion entspricht einem Monoflop, das während der Laufzeit nicht neu gestartet werden kann, dessen Laufzeit jedoch vorzeitig abgebrochen werden kann

5.2.5 Timer1 - ulTimer1ActrConf

Abhängig von der parametrierten Funktion des Aktor Objekts kann der Timer1 eine andere Zeit speichern. Ist das Aktor Objekt als einfacher Aktor parametriert, so wird die Zeit in Timer1 dafür genutzt die Ausschaltverzögerung zu bestimmen. D.h. nach dem Empfang eines AUS-Befehls muß diese Zeit erst verstreichen bevor der Aktor wirklich abschaltet. Ist der Aktor jedoch ein Monoflop, dann gibt die Zeit in Timer1 die Brenndauer an. Abhängig von der Art des Monoflops wird die Brenndauer eines angeschlossenen Lichts bei jedem empfangenen AN-Befehl neu gestartet, oder unter Umständen vorzeitig abgebrochen.
Timer1 kann in Sekundenintervallen parametriert werden. Die maximale Dauer beträgt: 65535s (`0xFFFF`). Soll die Ausschaltverzögerung bei einem einfachen Aktor deaktiviert werden, so muß der Wert in `Timer1` auf die Werkseinstellung 0 (`0x0000`) konfiguriert werden. Bei einem Monoflop ist ein Wert von `0x0000` nicht sinnvoll, da er die Brenndauer darstellt.
Beim Schreiben auf die `nviMiscCfg` wird immer ein 16-Bit Wert erwartet. Das höherwertige Byte muß zuerst bereit gestellt werden.

Beispiel 1:
Das Aktor Objekt 0 (Objektindex `0x09`) ist als einfaches Aktor Objekt parametriert. Seine Ausschaltverzögerung soll abgeschaltet werden (`0x0000`).
Wir beschreiben `nviMiscCfg` mit:

00	09	34	00	00	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Wir lesen `nviMiscCfg` aus, um die Parametrierung zu überprüfen und erhalten:

00	09	FD	00	00	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Beispiel 2:
Der Timer1 von Aktor Objekt 5 (Objektindex `0x0E`) soll auf 5 Minuten parametriert werden (`0x012C`)
Die Timer1 der Aktor Objekte 6 und 7 sollen auf jeweils 3 Minuten (`0x00B4`) parametriert werden.
Wir beschreiben `nviMiscCfg` mit:

00	0E	34	01	2C	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Wir lesen `nviMiscCfg` aus und erhalten:

00	0E	FD	01	2C	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Wir beschreiben `nviMiscCfg` mit:

00	0F	34	00	B4	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Wir lesen `nviMiscCfg` aus und erhalten:

00	0F	FD	00	B4	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Wir beschreiben `nviMiscCfg` mit:

00	10	34	00	B4	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Wir lesen `nviMiscCfg` aus und erhalten:

00	10	FD	00	B4	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Beispiel 3:
Es soll die Laufzeit für das Monoflop des Aktor 1 ermittelt werden.
Wir setzen `nviMiscCfg` auf :

02	0A	34	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Wir lesen `nviMiscCfg` zurück und erhalten:

02	0A	FC	01	A1	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Die ausgelesene Laufzeit `0x01A1` entspricht 417s (= 6 Minuten 57 Sekunden)

5.2.6 Einspeichern eines Textes – strDescription

Die Software 1691111a erlaubt es, jedem Aktor Objekt einen beliebigen Text bis zu 20 Zeichen, ohne terminierendes NULL-Zeichen, zuzuordnen. Dieser Text wird im Gerät gespeichert. Üblicherweise werden dort Informationen über den Installationsort des angesteuerten Gerätes gespeichert. Bei der Ansteuerung von Licht kann darin beispielsweise die Nummer der Leuchte gespeichert werden. Normalerweise werden die Ziffern im ASCII-Format codiert. Zeichenfolgen müssen mit einem NULL-Zeichen (0x00) terminiert werden um das Ende der Zeichenkette anzuzeigen. Man beachte, daß das NULL-Zeichen nicht der Ziffer „0“ im ASCII-Code entspricht, welche den Code (0x30) besitzt.
Beim Auslesen der Information werden alle 20 möglichen Zeichen und das terminierende NULL-Zeichen ausgegeben. Wurde eine Zeichenkette vorzeitig beendet, erkennt man das an einem NULL-Zeichen vor der 21. Stelle im Datenbereich. Es steht dem Benutzer frei auf die ASCII-Codierung zu verzichten und beliebige Hexadezimalzahlen zu verwenden. Allerdings kann man dann das Ende der Zeichenfolge nicht mehr eindeutig erkennen.

Beispiel 1:
Es soll im Aktor Objekt 1 hinterlegt werden, daß sich ein zugehöriger Lüfter in Raum: Bau 1/380 befindet. Folgender Text soll im ASCII-Format abgespeichert werden: „Lüfter@B1/380“.
Abgeschlossen wird der Text mit einem NULL-Zeichen.
Im ASCII-Format ergibt sich:
0x4C 0xFC 0x66 0x74 0x65 0x72 0x40 0x42 0x31 0x2F 0x33 0x38 0x30 0x00
Wir beschreiben `nviMiscCfg` mit:

00	0A	35	4C	FC	66	74	65	72	40	42	31	2F	33	38	30	00	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Wir lesen `nviMiscCfg` aus und erhalten:

00	0A	FD	4C	FC	66	74	65	72	40	42	31	2F	33	38	30	00	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Beispiel 2:
Der Text, der dem Aktor Objekt 6 als ASCII-Text hinterlegt wurde, soll ausgelesen werden.
Wir beschreiben `nviMiscCfg` mit:

02	0F	35	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Wir lesen `nviMiscCfg` aus und erhalten:

02	0F	FC	57	30	32	45	38	33	30	30	00	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	00	XX	XX	XX	XX	XX	XX

Nach der Rückkonvertierung aus dem ASCII-Code ergibt sich: „W02E8300“.

Anmerkung: könnte man nicht davon ausgehen, daß es sich um einen NULL-Zeichen terminierten ASCII-Code handelt, müßten alle Zeichen bis zu 0x00 an der 21. Stelle des Datenbereiches interpretiert werden.

5.2.7 Gemeinsame Parameter - CommonParameters

In diesem Parametrierschritt können Parameter hinterlegt werden, die für alle Aktoren gelten. Es wird empfohlen diese Parameter nicht zu ändern, da sie werkseitig auf passende Werte eingestellt sind.
Bei der Parametrierung werden alle gemeinsamen Variablen der Aktor Objekte mit einem Schreibbefehl parametriert. Als Objektnummer kann die eines beliebigen Aktor Objekts übergeben werden. Vorzugsweise ist die des ersten Aktor Objekts zu verwenden.
Doppelclick der Aktoren:
Das erste Byte des Datenbereiches in `nviMiscCfg` parametriert die Zeitspanne für einen Doppelclick. Z.B. werden bei stopbaren Monoflops zwei empfangene AN-Befehle innerhalb der parametrierten Zeitspanne als Abbruchkriterium der Laufzeit herangezogen. Werkseitig ist dieser Wert auf 1s festgelegt. Der Wertebereich erstreckt sich von 0 bis 255. Und gibt die Zeit in Einheiten von 10ms an.

Beispiel 1:
Die Zeitspanne, in der zwei empfangene AN-Befehle als Doppelclick erkannt werden wird, soll auf 1,6 Sekunden eingestellt werden. 1,6s = 160*10ms 160 entspricht `0xA0`
Wir beschreiben `nviMiscCfg` mit:

00	09	36	A0	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Wir lesen `nviMiscCfg` aus und erhalten:

00	09	FD	A0	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

6 Inbetriebnahme

7 Index