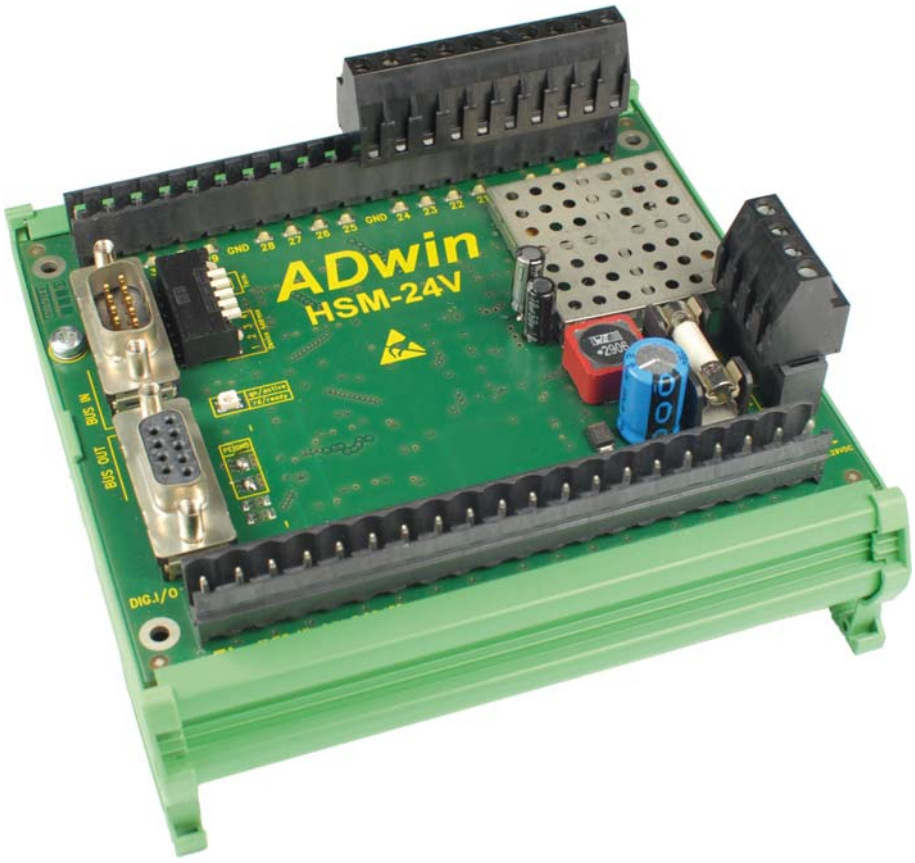
JÄGER
Computergesteuerte
Messtechnik GmbH

ADwin HSM-24V

Modul für LS-Bus

Handbuch



ADwin HSM-24V, Handbuch Feb. 2017

Hier finden Sie immer einen Ansprechpartner für Ihre Fragen:

Hotline:	(0 62 51) 9 63 20
Fax:	(0 62 51) 5 68 19
E-Mail:	info@ADwin.de
Internet	www.ADwin.de

JÄGER
Computergesteuerte
Messtechnik GmbH

Jäger Computergesteuerte
Messtechnik GmbH
Rheinstraße 2-4
D-64653 Lorsch

Inhaltsverzeichnis

Typografische Konventionen

Das „Achtung“-Zeichen steht bei Informationen, die auf Folgeschäden durch Fehlbedienung an der Hard- oder Software, am Messaufbau oder an Personen hinweisen.

Einen „Hinweis“ finden Sie bei

– Informationen, die für einen fehlerfreien Betrieb unbedingt beachtet werden müssen.

– Tipps und Ratschlägen für einen effizienten Betrieb.

Das Zeichen „Information“ verweist auf weiterführende Informationen in dieser Dokumentation oder andere Quellen wie Handbücher, Datenblätter, Literatur etc.

<C:\ADwin\…>

Dateinamen und -verzeichnisse sind in spitzen Klammern und im Schrifttyp Courier New angeben.

Programmtext

Programmanweisungen und Benutzer-Eingaben sind durch den Schrifttyp Courier New gekennzeichnet.

```
Var_1
```

Elemente eines Quelltextes wie Befehle, Variablen, Kommentar und sonstiger Text werden im Schrifttyp Courier New und farbig dargestellt.

In einem Datenwort (hier: 16 Bit) werden die Bits wie folgt nummeriert:

Bit-Nr.	15	14	13	…	1	0
Wert des Bits	2^{15}	2^{14}	2^{13}	…	2^1=2	2^0=1
Bezeichnung	MSB	-	-	-	-	LSB

1 Zu diesem Handbuch

Dieses Handbuch beschreibt das Modul HSM-24V, das am LS-Bus betrieben wird. Es wird ergänzt durch

– die Beschreibung der LS-Bus-Schnittstelle im Hardware-Handbuch für *ADwin-light-16*, *ADwin-Gold* oder *ADwin-Pro*.

– das Handbuch *ADbasic*, das grundlegende Befehle für den gleichnamigen Compiler enthält sowie das Funktionsprinzip von *ADwin*-Systemen näher erläutert.

  Die Online-Hilfe hat den gleichen Inhalt wie das Handbuch *ADbasic* und enthält zusätzlich die Hardware-bezogenen Befehle.

Bitte beachten Sie folgende Hinweise

Damit Ihr *ADwin*-System sicher arbeitet, halten Sie sich an die Informationen dieser und weiterführender Dokumentationen, auf die hier verwiesen wird.

Einschränkung der Anwendergruppe

Der Hersteller des in dieser Dokumentation beschriebenen Systems geht davon aus, dass an dem Gerät nur qualifiziertes Personal arbeitet.

> *Qualifiziertes Personal sind Personen, die aufgrund ihrer Ausbildung, Erfahrung und Unterweisung sowie ihrer Kenntnisse über einschlägige Normen, Bestimmungen, Unfallverhütungsvorschriften und Betriebsverhältnisse von dem für die Sicherheit der Anlage Verantwortlichen berechtigt worden sind, die jeweils erforderlichen Tätigkeiten auszuführen und die dabei mögliche Gefahren erkennen und vermeiden können. (Definition für Fachkräfte nach VDE 105 und IEC 60364)*.

Verfügbarkeit der Unterlagen

Diese Produktdokumentation und Unterlagen, auf die verwiesen wird, müssen stets verfügbar sein und konsequent beachtet werden. Für Schäden, die durch Missachtung der Informationen in dieser bzw. der weiterführenden Dokumentation entstehen, übernimmt die Firma *Jäger Computergesteuerte Messtechnik GmbH*, Lorsch, keine Haftung.

Rechtliche Grundlagen

Fremdprodukte werden ohne Vermerk auf mögliche Patentrechte genannt, deren Existenz nicht auszuschließen ist.

Änderungen vorbehalten.

Hotline-Adresse siehe vordere Umschlagseite, innen.

2 Der LS-Bus

Der LS-Bus ist ein serieller, bidirektionaler Bus mit einer Taktrate von 5MHz. Der Bus verbindet ein *ADwin*-System über dessen LS-Bus-Schnittstelle mit bis zu 15 LS-Bus-Modulen.

Abbildung 1 zeigt die Standard-Anschlüsse eines LS-Bus-Moduls: Bus-Ein-/Ausgang mit LED, Schutzleiter PE, DIP-Schalter für Busabschluss und Busadresse.

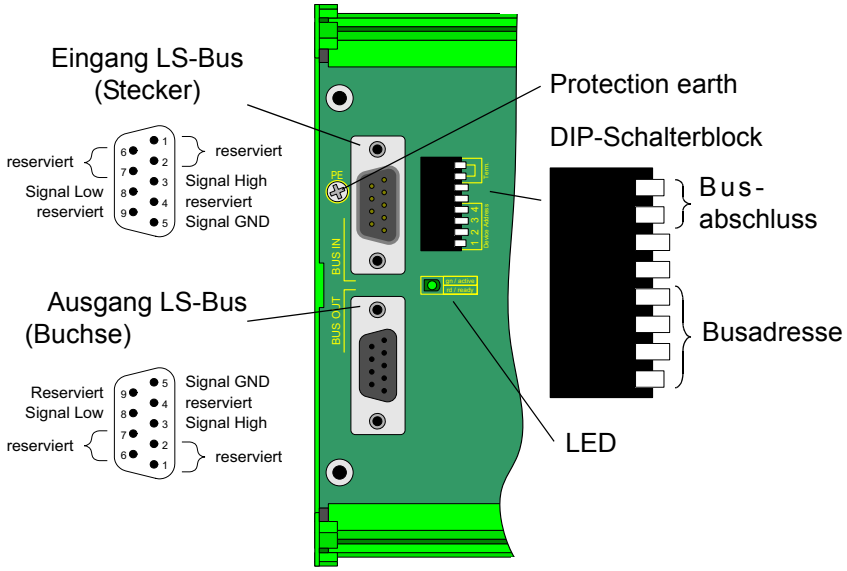


Abb. 1 – Standard-Anschlüsse

Der Bus ist als Linienverbindung aufgebaut, d.h. die Schnittstelle und die LS-Bus-Module sind jeweils über Zweipunktverbindungen miteinander verbunden. An jedem Modul sind eine Sub-D-Buchse (9-polig) als Bus-Eingang und ein Sub-D-Stecker (9-polig) als Bus-Ausgang vorhanden. Die maximale Buslänge beträgt 5m.

Die LED neben dem Bus-Eingang und -Ausgang zeigt den Datenverkehr auf dem LS-Bus an:

– LED leuchtet grün: Das Modul empfängt oder sendet Daten.

– LED leuchtet rot: Auf dem Bus werden Daten für andere Module gesendet.

– LED ist aus: Kein Datenverkehr.

Busabschluss

Am letzten Modul des LS-Bus muss der Busabschluss mit den DIP-Schaltern `Term.`, aktiviert sein, an allen anderen Modulen deaktiviert. Zum Aktivieren des Busabschlusses werden die beiden DIP-Schalter nach unten gestellt.

Busadresse

Jedes Modul am LS-Bus wird über seine Busadresse angesprochen; die Adresse muss daher für jedes Modul eindeutig sein.

Die Busadresse wird manuell an einem DIP-Schalterblock auf der Platine neben den Bussteckern eingestellt. Mit den 4 DIP-Schaltern `Device Address` sind die Adressen 1…15 einstellbar (siehe Abb. 2). Es gilt: 1 = DIP-Schalter unten, 0 = DIP-Schalter oben.

Die Adresse 0 deaktiviert das Modul. Auch wenn ein Modul deaktiviert ist, erhalten nachfolgende Module am LS-Bus alle Busnachrichten.

Modul-adresse	Einstellung der DIP-Schalter				
	1	2	3	4	
0	0	0	0	0	Modul deaktiviert
1	1	0	0	0	
2	0	1	0	0	
3	1	1	0	0	
4	0	0	1	0	
5	1	0	1	0	
…		…			
15	1	1	1	1	

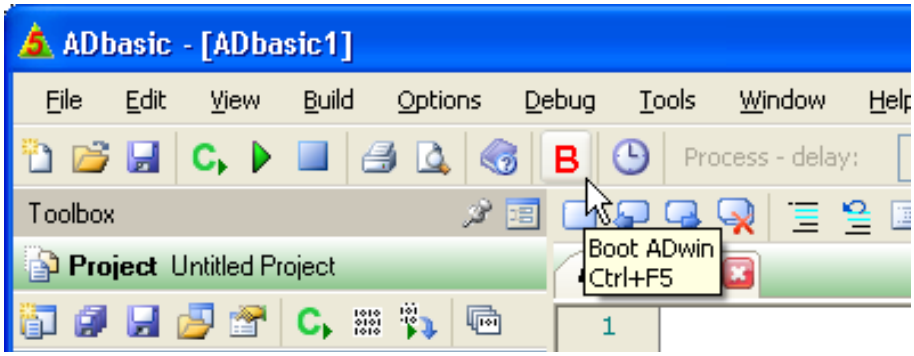
Abb. 2 – Moduladressierung mit DIP-Schaltern

Schutzleiter

Der GND-Pegel des Moduls ist mit der Hutschiene verbunden, die als Schutzleiter (PE) dient. Der Schutzleiter ist mit der Schraube `PE` auf der Oberseite des Moduls verbunden.

Booten

Starten Sie *ADbasic* und booten das *ADwin*-System durch Anklicken der Boot-Schaltfläche `B`.



3 HSM-24V

Das Modul HSM-24V hat 32 digitale Kanäle, die 24 V-Signale verarbeiten, und wird am LS-Bus betrieben.

3.1 Hardware

Das Modul benötigt eine externe Betriebsspannung von 19 V… 29 V. Der steckbare Versorgungsanschluss liegt rechts unten (siehe Abb. 3) auf dem Modul und hat je 2 Pins für V_{CC} und GND. Die Versorgungsspannung wird über eine Sicherung (5 A, träge) geführt. Beide Masseleitungen sind mit dem Schutzleiter PE verbunden.
Das Modul hat einen integrierten Verpolungsschutz.

Alle Anschlüsse sind auf der Oberseite der Platine beschriftet.

Die 32 Kanäle können in Gruppen von jeweils 8 als Eingänge oder Ausgänge geschaltet werden. Nach dem Einschalten sind alle Kanäle als Eingänge geschaltet.

Die Kanäle verarbeiten typischerweise 24 V-Signale und sind kurzschlussfest. Ein Signal über 82% der Betriebsspannung wird als High-Level verarbeitet, ein Signal unter 66% als Low-Level. Für jeden Kanal zeigt eine LED den Ist-Zustand an: LED ein entspricht High-Level.

Die Kanäle haben einen zulässigen Betriebsstrom von 0 mA…150 mA. Wenn der Strom 500 mA an einem Kanal übersteigt, wird der Kanal automatisch abgeschaltet. Ein Überstrom im Bereich von 150 mA…500 mA kann den Überhitzungsschutz des Treibers auslösen, d. h. der Treiber wird abgeschaltet, mitsamt den zugehörigen 16 Kanälen.

Jeweils 4 Kanäle haben eine gemeinsame GND-Leitung. Die Anschlussleisten sind steckbar und haben Schraubanschlüsse, über welche die Ein- und Ausgänge beschaltet werden.

Die Eingänge haben einen Filter mit ca. 12 µs Verzögerung.

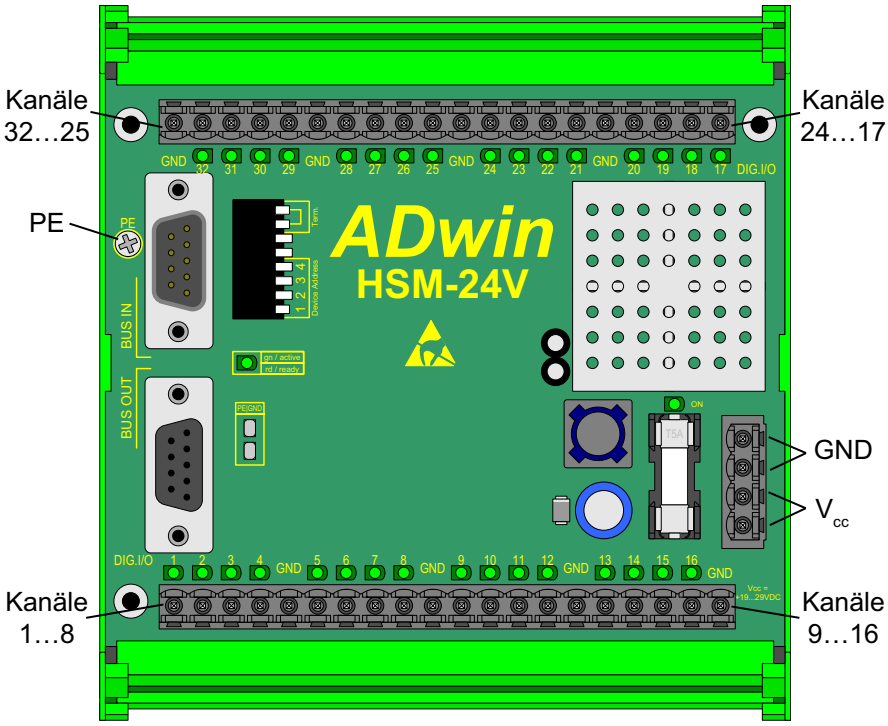
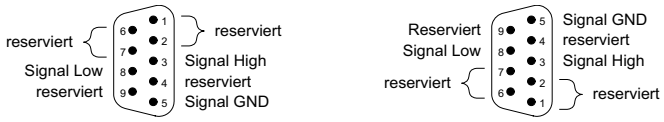


Abb. 3 – Platine HSM-24V

Das Modul ist in eine Montagewanne integriert, über die eine Schnapp-Befestigung auf einer DIN-Hutschiene möglich ist. Das Modul darf nur im Schaltschrank betrieben werden (Industrieeinsatz).



Eingang LS-Bus (Stecker) Ausgang LS-Bus (Buchse)

Abb. 4 – Pinbelegung LS-Bus HSM-24V

Das Modul ist für den Betrieb in trockenen Räumen konzipiert. Am Einbauort sollen eine Umgebungstemperatur von +5 °C … +50 °C und eine relative Luftfeuchte von 0 … 80 % (nicht kondensierend) vorhanden sein.

3.2 RoHS Konformitätserklärung

Die Richtlinie 2002/95/EG der Europäischen Union zur Beschränkung und Verwendung gefährlicher Stoffe in elektrischen und elektronischen Geräten (RoHS-Richtlinie) ist am 1. Juli 2006 in Kraft getreten.

Dabei handelt es sich um folgende Substanzen:

– Blei (Pb)
– Cadmium (Cd)
– Hexavalentes Chrom (Cr VI)
– Polybromierte Biphenyle (PBB)
– Polybromierte Diphenylether (PBDE)
– Quecksilber (Hg)

Das Modul HSM-24V erfüllt die Voraussetzungen der RoHS-Richtlinie.

3.3 Software

Die Funktionen des Moduls HSM-24V werden mit *ADbasic*-Befehlen komfortabel programmiert. Beachten Sie bitte, dass die Befehle für die verschiedenen *ADwin*-Systeme unterschiedlich sind.

Ein *ADwin*-Prozess, der auf das Modul HSM-24V zugreift, soll unbedingt mit niedriger Priorität arbeiten.
Wenn der (relativ langsame) Zugriff auf das Modul dagegen mit hoher Priorität stattfindet, müssen alle anderen Prozesse warten und ihr Echtzeitverhalten geht eventuell verloren. Außerdem kann die Datenverbindung zwischen PC und *ADwin*-System unterbrochen werden.

Die Befehle für das Modul HSM-24V haben folgende Funktion:

Befehl	Funktion
LS_DIO_INIT	HSM-Modul initialisieren.
LS_DIGPROG	Kanäle als Ein- oder Ausgänge programmieren.
LS_DIG_IO	Digital-Ausgänge setzen und die aktuellen Zustände zurückgeben. Der Befehl gilt nur für ein einzelnes Modul am LS-Bus. Keine Fehlerbehandlung.
LS_DIGOUT_LONG	Digital-Ausgänge setzen.
LS_DIGIN_LONG	Pegel der Digital-Eingänge lesen.
LS_GET_OUTPUT_STATUS	Überstrom-Status der digitalen Ausgänge lesen.
LS_WATCHDOG_INIT	Watchdog-Zähler deaktivieren oder aktivieren und die Watchdog-Zeit setzen.
LS_WATCHDOG_RESET	Watchdog-Zähler auf allen Modulen am LS-Bus auf den Startwert zurücksetzen.

Abb. 5 – Befehle für HSM-24V, Kurzübersicht

Die Befehle sind jeweils in einer Include-Datei enthalten; binden Sie die zum *ADwin*-System passende Datei am Beginn des Programms ein. Die folgende Liste für die *ADwin*-Systeme die passende Programmzeile und die Seite in dieser Dokumentation, auf der die Befehlsbeschreibung beginnt.

ADwin-light-16:	`#INCLUDE ADWL16.INC`	Seite 7
ADwin-Gold II:	`#INCLUDE ADwinGoldII.INC`	Seite 28
ADwin-Pro:	`#INCLUDE ADwinPro_All.INC`	Seite 49

Bei *ADwin-Gold II* können alle Befehle auch in *TiCoBasic* für den Zugriff auf das HSM-Modul genutzt werden. Binden Sie dafür aber die Include-Datei `GoldIITiCo.inc` in das Programm ein.

3.3.1 LS-Bus + *ADwin-light-16*

Dieser Abschnitt beschreibt Befehle für die LS-Bus-Schnittstelle an *ADwin-light-16*:

LS_DIO_Init

`LS_DIO_Init` initialisiert ein Modul vom Typ HSM-24V am LS-Bus und gibt den Fehlerstatus zurück.

Syntax

```
#Include ADWL16.Inc
ret_val = LS_DIO_Init(ls_module)
```

Parameter

`ls_module`	Eingestellte Moduladresse (1…15) am HSM-Modul auf dem LS-Bus.	LONG
`ret_val`	Rückgabewert, das den Fehlerstatus angibt: -1: Keine Kommunikation mit dem Modul möglich. >0: Bitmuster mit mehreren Fehlerbits. Bit = 0: kein Fehler. Bit = 1: Fehler aufgetreten.	LONG

Bit-Nr.	31…8	7	6	5…4	3	2	1	0
Status	–	Temp2	Temp1	–	WD	Time	Ovr	Par

- :don't care (mit `0CFh` ausmaskieren)
Par:Parity-Fehler bei der Datenübertragung auf dem LS-Bus.
Ovr: Overrun-Fehler bei der Datenübertragung auf dem LS-Bus.
Time: Timeout-Fehler bei der Datenübertragung auf dem LS-Bus.
WD: Watchdog hat ausgelöst. Die Kanaltreiber sind deaktiviert.
Temp1: Übertemperatur am Treiber für Kanäle 1…16. Treiber ist deaktiviert.
Temp2: Übertemperatur am Treiber für Kanäle 17…32. Treiber ist deaktiviert.

Bemerkungen

Die Anweisung soll nur im Abschnitt `INIT:` verwendet werden, weil sie eine lange Ausführungszeit hat.

Die Initialisierung setzt folgende Einstellungen:

- Alle DIO-Kanäle werden als Eingang programmiert.
  Andere Einstellungen siehe `LS_DigProg`.
- Der Status für Überstrom (> ca. 500 mA) wird zurückgesetzt.
- Der Fehlerstatus für Übertemperatur wird zurückgesetzt.
- Der Fehlerstatus für Timeout auf dem LS-Bus wird zurückgesetzt.
- Der Status des Watchdog-Zählers bleibt unverändert.

Das Modul speichert auftretende Fehler unabhängig vom *ADwin*-System. Fehlerbits im Rückgabewert können sich daher auf einen Fehler beziehen, der zu einem früheren Zeitpunkt aufgetreten ist.

Beachten Sie: Ignorieren Sie beim Programmstart das Fehlerbit WD so lange, bis der Watchdog-Zähler mit `LS_Watchdog_Init` konfiguriert ist. Der Watchdog-Zähler startet mit dem Einschalten des Moduls und hat bis zum Programmstart in der Regel schon einen Fehler ausgelöst.

Der Fehler „Übertemperatur“ eines Treibers kann nur auftreten, wenn auf mehreren Kanälen gleichzeitig ein Überstrom im Bereich von 150…500 mA anliegt. Unabhängig davon wird bei einem Überstrom über 500 mA der betroffene Kanal automatisch abgeschaltet.

Die Kanäle des Moduls HSM-24V dürfen nur im Bereich von 0…150 mA betrieben werden. Damit ist sichergestellt, dass das Modul HSM-24V dauerhaft ohne Unterbrechung arbeitet, selbst wenn alle Kanäle gleichzeitig aktiv sind.

Siehe auch

LS_DigProg, LS_Dig_IO, LS_Digout_Long, LS_Digin_Long, LS_Get_Output_Status, LS_Watchdog_Init, LS_Watchdog_Reset

Gültig für

HSM-24V + L16

Pinbelegungen

, HSM-24V

Beispiel

```
REM Example process for one module HSM-24V And ADwin-L16
#Include ADwL16.inc

Init:
  Processdelay = 4000000    '10Hz HP
  Par_1 = LS_DIO_Init(1)
  REM enable watchdog time with 1.1 sec
  Par_2 = LS_Watchdog_Init(1, 1, 1100)
  REM LS channels 1…32 as outputs
  Par_3 = LS_Digprog(1, 0Fh)

Event:
  REM set one LS channel to high, rotating from 1 to 32
  Inc Par_10
  If (Par_10 >= 32) Then Par_10 = 0
  Par_11 = Shift_Left(1, Par_10)
  REM reset watchdog, set LS channels, and read back real state
  REM (note: LS_Dig_IO uses LS address 1)
  Par_12 = LS_Dig_IO(1, Par_11)
```

LS_DigProg

LS_DigProg programmiert die digitalen Kanäle 1…32 eines Moduls vom Typ HSM-24V am LS-Bus in Gruppen zu 8 als Ein- oder Ausgang.

Syntax

```
#Include ADWL16.Inc
ret_val = LS_DigProg(ls_module, pattern)
```

Parameter

`ls_module` Eingestellte Moduladresse (1…15) am HSM-Modul auf dem LS-Bus. LONG

`pattern` Bitmuster, nach dem die Kanäle als Ein- oder Ausgang gesetzt werden: LONG
Bit = 0: Kanal als Eingang setzen.
Bit = 1: Kanal als Ausgang setzen.

Bitnr.	31…4	3	2	1	0
Kanalnr.	–	32:25	24:17	16:9	8:1

`ret_val` Rückgabewert, das den Fehlerstatus angibt: LONG
-1: Keine Kommunikation mit dem Modul möglich.
>0: Bitmuster mit mehreren Fehlerbits.
Bit = 0: kein Fehler.
Bit = 1: Fehler aufgetreten.

Bit-Nr.	31…8	7	6	5…4	3	2	1	0
Status	–	Temp2	Temp1	–	WD	Time	Ovr	Par

- :don't care (mit `0CFh` ausmaskieren)
Par:Parity-Fehler bei der Datenübertragung auf dem LS-Bus.
Ovr: Overrun-Fehler bei der Datenübertragung auf dem LS-Bus.
Time: Timeout-Fehler bei der Datenübertragung auf dem LS-Bus.
WD: Watchdog hat ausgelöst. Die Kanaltreiber sind deaktiviert.
Temp1: Übertemperatur am Treiber für Kanäle 1…16. Treiber ist deaktiviert.
Temp2: Übertemperatur am Treiber für Kanäle 17…32. Treiber ist deaktiviert.

Bemerkungen

Die Anweisung soll nur im Abschnitt **Init:** verwendet werden, weil sie eine lange Ausführungszeit hat.

Nach der Initialisierung mit **LS_DIO_Init** sind alle Kanäle als Eingänge konfiguriert.

Die Kanäle können nur in Gruppen zu je 8 als Ein- oder Ausgang gesetzt werden (nur 4 relevante Bits, die anderen Bits werden ignoriert).

Siehe auch

LS_DIO_Init, LS_Dig_IO, LS_Digout_Long, LS_Digin_Long, LS_Get_Output_Status, LS_Watchdog_Init, LS_Watchdog_Reset

Gültig für

HSM-24V + L16

Pinbelegungen

, HSM-24V

Beispiel

```
REM Example process for one module HSM-24V And ADwin-L16
#Include ADwL16.inc

Init:
  Processdelay = 4000000    '10Hz HP
  Par_1 = LS_DIO_Init(1)
  REM enable watchdog time with 1.1 sec
  Par_2 = LS_Watchdog_Init(1, 1, 1100)
  REM LS channels 1…32 as outputs
  Par_3 = LS_Digprog(1, 0Fh)

Event:
  REM set one LS channel to high, rotating from 1 to 32
  Inc Par_10
  If (Par_10 >= 32) Then Par_10 = 0
  Par_11 = Shift_Left(1, Par_10)
  REM reset watchdog, set LS channels, and read back real state
  REM (note: LS_Dig_IO uses LS address 1)
  Par_12 = LS_Dig_IO(1, Par_11)
```

LS_Dig_IO

LS_Dig_IO setzt alle Digital-Ausgänge des Moduls HSM-24V am LS-Bus auf den Pegel High oder Low und gibt den Zustand aller Kanäle als Bitmuster zurück.

Syntax

```
#Include ADWL16.Inc
ret_val = LS_Dig_IO(pattern)
```

Parameter

pattern	Bitmuster, mit dem die digitalen Ausgänge gesetzt werden (siehe Tabelle). Bit = 0: Ausgang auf Pegel Low setzen. Bit = 1: Ausgang auf Pegel High setzen.	LONG
ret_val	Bitmuster mit dem Ist-Zustand aller digitalen Kanäle (siehe Tabelle). Bit = 0: Pegel Low liegt an. Bit = 1: Pegel High liegt an.	LONG

Bitnr.	31	30	29	…	2	1	0
Eingang	32	31	30	…	3	2	1

Bemerkungen

LS_Dig_IO arbeitet nur korrekt, wenn folgende Voraussetzungen erfüllt sind:

- Am LS-Bus ist nur ein Modul angeschlossen.
- Das Modul ist vom Typ HSM-24V.
- Am Modul ist die Moduladresse 1 eingestellt.
- Nach Aufruf des Befehls **LS_Dig_IO** wird kein anderer LS-Bus-Befehl mehr verwendet.

Die Kanäle werden mit **LS_DigProg** als Ein- oder Ausgänge programmiert.

Das Bitmuster pattern wird nur für die Kanäle angewendet, die als Ausgang programmiert sind. Bits für Eingänge werden ignoriert.

Der Rückgabewert enthält den tatsächlichen Schaltzustand sowohl von Eingängen wie von Ausgängen. Beachten Sie: Die Eingänge haben einen Filter mit ca. 12µs Verzögerung.

LS_Dig_IO setzt den Watchdog-Zähler des Moduls auf den Startwert zurück. Der Zähler bleibt aktiv. Der Startwert wird mit **LS_Watchdog_Init** eingestellt.

Setzen Sie den aktiven Watchdog-Zähler während seines Zählvorgangs mindestens einmal zurück, um die Funktion des Moduls zu gewährleisten.

Gültig für

HSM-24V + L16

Siehe auch

LS_DIO_Init, LS_DigProg, LS_Digout_Long, LS_Digin_Long, LS_Get_Output_Status, LS_Watchdog_Init, LS_Watchdog_Reset

Pinbelegungen

, HSM-24V

Beispiel

```
REM Example process for one module HSM-24V And ADwin-L16
#Include ADwL16.inc

Init:
  Processdelay = 4000000    '10Hz HP
  Par_1 = LS_DIO_Init(1)
  REM enable watchdog time with 1.1 sec
  Par_2 = LS_Watchdog_Init(1, 1, 1100)
  REM LS channels 1…32 as outputs
  Par_3 = LS_Digprog(1, 0Fh)

Event:
  REM set one LS channel to high, rotating from 1 to 32
  Inc Par_10
  If (Par_10 >= 32) Then Par_10 = 0
  Par_11 = Shift_Left(1, Par_10)
  REM reset watchdog, set LS channels, and read back real state
  REM (note: LS_Dig_IO uses LS address 1)
  Par_12 = LS_Dig_IO(1, Par_11)
```

LS_Digout_Long

LS_Digout_Long setzt oder löscht durch den übergebenen 32 Bit-Wert alle Ausgänge auf einem Modul vom Typ HSM-24V am LS-Bus.

Syntax

```
#Include ADWL16.Inc
LS_Digout_Long(ls_module,pattern)
```

Parameter

ls_module	Eingestellte Moduladresse (1…15) am HSM-Modul auf dem LS-Bus.	LONG
pattern	Bitmuster, nach dem die digitalen Ausgänge gesetzt werden: Bit = 0: Ausgang auf Pegel Low setzen. Bit = 1: Ausgang auf Pegel High setzen.	LONG

Bitnr.	31	30	…	2	1	0
Ausgang	32	31	…	3	2	1

Bemerkungen

Die Kanäle werden mit **LS_DigProg** als Ein- oder Ausgänge programmiert.

Das Bitmuster pattern wird nur für die Kanäle angewendet, die als Ausgang programmiert sind. Bits für Eingänge werden ignoriert.

Siehe auch

LS_DIO_Init, LS_DigProg, LS_Dig_IO, LS_Digin_Long, LS_Digout_Long_BS, LS_Get_Output_Status, LS_Watchdog_Init, LS_Watchdog_Reset

Gültig für

HSM-24V + L16

Pinbelegungen

, HSM-24V

Beispiel

```
REM Example process for ADwin-L16 and 2 modules HSM-24V
REM Set process to low priority!
#Include ADwL16.inc

Init:
  Processdelay = 4000000    '10Hz HP
  REM settings for LS module 1
  Par_1 = LS_DIO_Init(1)
  REM enable watchdog with 1.1 s
  Par_2 = LS_Watchdog_Init(1, 1, 1100)
  REM set LS channels 1…32 as outputs
  Par_3 = LS_Digprog(1, 01111b)

  REM settings for LS module 3
  Par_11 = LS_DIO_Init(3)
  REM enable watchdog with 1.1 s
  Par_12 = LS_Watchdog_Init(3, 1, 1100)
  REM set LS channels 1…32 as inputs
  Par_13 = LS_Digprog(3, 0h)

Event:
  REM set one LS channel to high, rotating from 1 to 32
  Inc Par_15
  If (Par_15 >= 32) Then Par_15 = 0
  Par_16 = Shift_Left(1, PAR_15)
  REM set LS channels of module 1
  LS_Digout_Long(1, PAR_16)
  REM read LS channels of module 3
  Par_17 = LS_Digin_Long(3)
  REM reset watchdog
  LS_Watchdog_Reset()
```

LS_Digout_Long_BS

LS_Digout_Long_BS setzt oder löscht durch den übergebenen 32 Bit-Wert alle Ausgänge auf einem Modul vom Typ HSM-24V am LS-Bus und gibt den Fehlerstatus zurück.

Syntax

```
#Include ADWL16.Inc
LS_Digout_Long_BS(ls_module,pattern,status)
```

Parameter

`ls_module` Eingestellte Moduladresse (1…15) am HSM-Modul auf dem LS-Bus. LONG

`pattern` Bitmuster, nach dem die digitalen Ausgänge gesetzt werden: LONG
Bit = 0: Ausgang auf Pegel Low setzen.
Bit = 1: Ausgang auf Pegel High setzen.

Bitnr.	31	30	…	2	1	0
Ausgang	32	31	…	3	2	1

`status` Fehlerstatus: LONG
0: O.k.
-1: Keine Kommunikation mit dem Modul möglich.
>0: Bitmuster mit mehreren Fehlerbits.
Bit = 0: kein Fehler.
Bit = 1: Fehler aufgetreten.

Bitnr.	31…8	7	6	5…4	3	2	1	0
Status	–	Temp2	Temp1	–	WD	Time	Ovr	Par

- :don't care (mit `0CFh` ausmaskieren)
Par:Parity-Fehler bei der Datenübertragung auf dem LS-Bus.
Ovr: Overrun-Fehler bei der Datenübertragung auf dem LS-Bus.
Time: Timeout-Fehler bei der Datenübertragung auf dem LS-Bus.
WD: Watchdog hat ausgelöst. Die Kanaltreiber sind deaktiviert.
Temp1: Übertemperatur am Treiber für Kanäle 1…16. Treiber ist deaktiviert.
Temp2: Übertemperatur am Treiber für Kanäle 17…32. Treiber ist deaktiviert.

Bemerkungen

Die Kanäle werden mit **LS_DigProg** als Ein- oder Ausgänge programmiert.

Das Bitmuster `pattern` wird nur für die Kanäle angewendet, die als Ausgang programmiert sind. Bits für Eingänge werden ignoriert.

Siehe auch

LS_DIO_Init, LS_DigProg, LS_Dig_IO, LS_Digin_Long_BS, LS_Digout_Long, LS_Get_Output_Status, LS_Watchdog_Init, LS_Watchdog_Reset

Gültig für

HSM-24V + L16

Pinbelegungen

, HSM-24V

Beispiel

```
REM Example process for ADwin-L16 and 2 modules HSM-24V
REM Set process to low priority!
#Include ADwL16.inc

Init:
  Processdelay = 4000000    '10Hz HP
  REM settings for LS module 1
  Par_1 = LS_DIO_Init(1)
  REM enable watchdog with 1.1 s
  Par_2 = LS_Watchdog_Init(1, 1, 1100)
  REM set LS channels 1…32 as outputs
  Par_3 = LS_Digprog(1, 01111b)

  REM settings for LS module 3
  Par_11 = LS_DIO_Init(3)
  REM enable watchdog with 1.1 s
  Par_12 = LS_Watchdog_Init(3, 1, 1100)
  REM set LS channels 1…32 as inputs
  Par_13 = LS_Digprog(3, 0h)

Event:
  REM set one LS channel to high, rotating from 1 to 32
  Inc Par_15
  If (Par_15 >= 32) Then Par_15 = 0
  Par_16 = Shift_Left(1, PAR_15)
  REM set LS channels of module 1
  LS_Digout_Long_BS(1, PAR_16, Par_5)
  If (Par_5 <> 0) Then End 'exit on error)
  REM read LS channels of module 3
  Par_17 = LS_Digin_Long_BS(3, Par_6)
  If (Par_6 <> 0) Then End 'exit on error)
  REM reset watchdog
  LS_Watchdog_Reset()
```

LS_Digin_Long

LS_Digin_Long gibt den Zustand aller Kanäle auf einem Modul vom Typ HSM-24V am LS-Bus als Bitmuster zurück.

Syntax

```
#Include ADWL16.Inc
ret_val = LS_Digin_Long(ls_module)
```

Parameter

ls_module	Eingestellte Moduladresse (1…15) am HSM-Modul auf dem LS-Bus.	LONG
ret_val	Bitmuster. Jedes Bit (31…0) entspricht dem Zustand eines digitalen Kanals (siehe Tabelle). Bit = 0: Pegel Low liegt an. Bit = 1: Pegel High liegt an.	LONG

Bitnr.	31	30	…	2	1	0
Kanalnr.	32	31	…	3	2	1

Bemerkungen

Wir empfehlen, die angesprochenen Leitungen zunächst mit der Anweisung **LS_DigProg** als Eingänge zu programmieren.

Siehe auch

LS_DIO_Init, LS_DigProg, LS_Dig_IO, LS_Digin_Long_BS, LS_Digout_Long, LS_Get_Output_Status, LS_Watchdog_Init, LS_Watchdog_Reset

Gültig für

HSM-24V + L16

Pinbelegungen

, HSM-24V

Beispiel

```
REM Example process for ADwin-L16 and 2 modules HSM-24V
REM Set process to low priority!
#Include ADwL16.inc

Init:
  Processdelay = 4000000    '10Hz HP
  REM settings for LS module 1
  Par_1 = LS_DIO_Init(1)
  REM enable watchdog with 1.1 s
  Par_2 = LS_Watchdog_Init(1, 1, 1100)
  REM set LS channels 1…32 as outputs
  Par_3 = LS_Digprog(1, 01111b)

  REM settings for LS module 3
  Par_11 = LS_DIO_Init(3)
  REM enable watchdog with 1.1 s
  Par_12 = LS_Watchdog_Init(3, 1, 1100)
  REM set LS channels 1…32 as inputs
  Par_13 = LS_Digprog(3, 0h)

Event:
  REM set one LS channel to high, rotating from 1 to 32
  Inc Par_15
  If (Par_15 >= 32) Then Par_15 = 0
  Par_16 = Shift_Left(1, PAR_15)
  REM set LS channels of module 1
  LS_Digout_Long(1, PAR_16)
  REM read LS channels of module 3
  Par_17 = LS_Digin_Long(3)
  REM reset watchdog
  LS_Watchdog_Reset()
```

LS_Digin_Long_BS

LS_Digin_Long_BS gibt den Zustand aller Kanäle auf einem Modul vom Typ HSM-24V am LS-Bus als Bitmuster zurück sowie den Fehlerstatus.

Syntax

```
#Include ADWL16.Inc
ret_val = LS_Digin_Long_BS(ls_module)
```

Parameter

`ls_module` Eingestellte Moduladresse (1…15) am HSM-Modul auf dem LS-Bus. LONG

`status` Fehlerstatus: LONG
0: O.k.
-1: Keine Kommunikation mit dem Modul möglich.
>0: Bitmuster mit mehreren Fehlerbits.
Bit = 0: kein Fehler.
Bit = 1: Fehler aufgetreten.

Bitnr.	31…8	7	6	5…4	3	2	1	0
Status	–	Temp2	Temp1	–	WD	Time	Ovr	Par

- :don't care (mit 0CFh ausmaskieren)
Par:Parity-Fehler bei der Datenübertragung auf dem LS-Bus.
Ovr: Overrun-Fehler bei der Datenübertragung auf dem LS-Bus.
Time: Timeout-Fehler bei der Datenübertragung auf dem LS-Bus.
WD: Watchdog hat ausgelöst. Die Kanaltreiber sind deaktiviert.
Temp1: Übertemperatur am Treiber für Kanäle 1…16. Treiber ist deaktiviert.
Temp2: Übertemperatur am Treiber für Kanäle 17…32. Treiber ist deaktiviert.

`ret_val` Bitmuster. Jedes Bit (31…0) entspricht dem Zustand eines digitalen Kanals (siehe Tabelle). LONG
Bit = 0: Pegel Low liegt an.
Bit = 1: Pegel High liegt an.

Bitnr.	31	30	…	2	1	0
Kanalnr.	32	31	…	3	2	1

Bemerkungen

Wir empfehlen, die angesprochenen Leitungen zunächst mit der Anweisung **LS_DigProg** als Eingänge zu programmieren.

Siehe auch

LS_DIO_Init, LS_DigProg, LS_Dig_IO, LS_Digin_Long, LS_Digout_Long_BS, LS_Get_Output_Status, LS_Watchdog_Init, LS_Watchdog_Reset

Gültig für

HSM-24V + L16

Pinbelegungen

, HSM-24V

Beispiel

```
REM Example process for ADwin-L16 and 2 modules HSM-24V
REM Set process to low priority!
#Include ADwL16.inc

Init:
  Processdelay = 4000000    '10Hz HP
  REM settings for LS module 1
  Par_1 = LS_DIO_Init(1)
  REM enable watchdog with 1.1 s
  Par_2 = LS_Watchdog_Init(1, 1, 1100)
  REM set LS channels 1…32 as outputs
  Par_3 = LS_Digprog(1, 01111b)

  REM settings for LS module 3
  Par_11 = LS_DIO_Init(3)
  REM enable watchdog with 1.1 s
  Par_12 = LS_Watchdog_Init(3, 1, 1100)
  REM set LS channels 1…32 as inputs
  Par_13 = LS_Digprog(3, 0h)

Event:
  REM set one LS channel to high, rotating from 1 to 32
  Inc Par_15
  If (Par_15 >= 32) Then Par_15 = 0
  Par_16 = Shift_Left(1, PAR_15)
  REM set LS channels of module 1
  LS_Digout_Long_BS(1, PAR_16, Par_5)
  If (Par_5 <> 0) Then End 'exit on error)
  REM read LS channels of module 3
  Par_17 = LS_Digin_Long_BS(3, Par_6)
  If (Par_6 <> 0) Then End 'exit on error)
  REM reset watchdog
  LS_Watchdog_Reset()
```

LS_Get_Output_Status

LS_Get_Output_Status gibt den Überstrom-Status der Ausgänge auf einem Modul vom Typ HSM-24V am LS-Bus als Bitmuster zurück.

Syntax

```
#Include ADWL16.Inc
ret_val = LS_Get_Output_Status(ls_module)
```

Parameter

ls_module	Eingestellte Moduladresse (1…15) am HSM-Modul auf dem LS-Bus.	LONG
ret_val	Bitmuster. Jedes Bit (31…0) entspricht dem Überstrom-Status eines digitalen Ausgangs (siehe Tabelle). Bit = 0: Normalzustand. Bit = 1: Ausgang wegen Überstrom deaktiviert.	LONG

Bitnr.	31	30	…	2	1	0
Ausgang	32	31	…	3	2	1

Bemerkungen

Der Fehler „Übertemperatur“ eines Treibers kann nur auftreten, wenn auf mehreren Kanälen gleichzeitig ein Überstrom im Bereich von 150…500 mA anliegt. Unabhängig davon wird bei einem Überstrom über 500 mA der betroffene Kanal automatisch abgeschaltet.

Nach dem Fehler „Übertemperatur“ wird das Modul mit **LS_DIO_Init** wieder zurückgesetzt.

Die Kanäle des Moduls HSM-24V dürfen nur im Bereich von 0…150 mA betrieben werden. Damit ist sichergestellt, dass das Modul HSM-24V dauerhaft ohne Unterbrechung arbeitet, selbst wenn alle Kanäle gleichzeitig aktiv sind.

Siehe auch

LS_DIO_Init, LS_DigProg, LS_Dig_IO, LS_Digout_Long, LS_Digin_Long, LS_Watchdog_Init, LS_Watchdog_Reset

Gültig für

HSM-24V + L16

Pinbelegungen

, HSM-24V

Beispiel

```
REM Example process for ADwin-L16 and 2 modules HSM-24V
REM Set process to low priority!
#Include ADwL16.inc

Init:
  Processdelay = 4000000    '10Hz HP
  REM settings for LS module 1
  Par_1 = LS_DIO_Init(1)
  REM enable watchdog with 1.1 s
  Par_2 = LS_Watchdog_Init(1, 1, 1100)
  REM set LS channels 1…32 as outputs
  Par_3 = LS_Digprog(1, 01111b)

  REM settings for LS module 3
  Par_11 = LS_DIO_Init(3)
  REM enable watchdog with 1.1 s
  Par_12 = LS_Watchdog_Init(3, 1, 1100)
  REM set LS channels 1…32 as inputs
  Par_13 = LS_Digprog(3, 0h)

Event:
  REM check for over-current
  Par_5 = LS_Get_Output_Status(1) + LS_Get_Output_Status(3)
  If (Par_5 > 0) Then End  'over-current: Exit program

  REM set one channel to high, rotating from 1 To 32
  Inc Par_15
  If (Par_15 >= 32) Then Par_15 = 0
  Par_16 = Shift_Left(1, PAR_15)
  REM set LS channels of module 1
  LS_Digout_Long(1, Par_16)
  REM read LS channels of module 3
  Par_17 = LS_Digin_Long(3)
  REM reset watchdog
  LS_Watchdog_Reset()
```

LS_Watchdog_Init

LS_Watchdog_Init aktiviert oder deaktiviert den Watchdog-Zähler eines Moduls am LS-Bus. Beim Aktivieren erhält der Zähler seinen Startwert und wird gestartet.

Syntax

```
#Include ADWL16.Inc
ret_val = LS_Watchdog_Init(ls_module,enable,time)
```

Parameter

ls_module	Eingestellte Moduladresse (1…15) am HSM-Modul auf dem LS-Bus.	LONG
enable	Status des Watchdog-Zählers einstellen: 0 : Watchdog-Zähler deaktivieren. 1 : Watchdog-Zähler aktivieren.	LONG
time	Auslösezeit (0…107374) des Zählers in Millisekunden.	LONG
ret_val	Rückgabewert, das den Fehlerstatus angibt: -1: Keine Kommunikation mit dem Modul möglich. >0: Bitmuster mit mehreren Fehlerbits. Bit = 0: kein Fehler. Bit = 1: Fehler aufgetreten.	LONG

Bit-Nr.	31…8	7	6	5…4	3	2	1	0
Status	–	Temp2	Temp1	–	WD	Time	Ovr	Par

- :don't care (mit 0CFh ausmaskieren)
Par:Parity-Fehler bei der Datenübertragung auf dem LS-Bus.
Ovr: Overrun-Fehler bei der Datenübertragung auf dem LS-Bus.
Time: Timeout-Fehler bei der Datenübertragung auf dem LS-Bus.
WD: Watchdog hat ausgelöst. Die Kanaltreiber sind deaktiviert.
Temp1: Übertemperatur am Treiber für Kanäle 1…16. Treiber ist deaktiviert.
Temp2: Übertemperatur am Treiber für Kanäle 17…32. Treiber ist deaktiviert.

Bemerkungen

Die Anweisung soll nur im Abschnitt **INIT:** verwendet werden, weil sie eine lange Ausführungszeit hat.

Solange der Watchdog-Zähler aktiv ist, dekrementiert er den Zählerstand kontinuierlich. Nach der eingestellten Auslösezeit erreicht der Zählerstand 0 (Null). Das Modul nimmt nun eine Fehlfunktion an und wird gestoppt; dadurch werden alle Ausgangssignale zurückgesetzt.

Nach dem Einschalten des Moduls ist der Zähler auf den Startwert 10 ms eingestellt und der Watchdog ist aktiv.

Setzen Sie den aktiven Watchdog-Zähler während seines Zählvorgangs mindestens einmal zurück, um die Funktion des Moduls zu gewährleisten. Zum Zurücksetzen können modulspezifische Befehle oder **LS_Watchdog_Reset** verwendet werden.

Die Watchdog-Funktion dient zur Verbindungsüberwachung zwischen *ADwin*-System und LS-Bus-Modul.

Siehe auch

LS_DIO_Init, LS_DigProg, LS_Dig_IO, LS_Digout_Long, LS_Digin_Long, LS_Get_Output_Status, LS_Watchdog_Reset

Gültig für

HSM-24V + L16

Pinbelegungen

, HSM-24V

Beispiel

```
REM Example process for one module HSM-24V And ADwin-L16
#Include ADwL16.inc

Init:
  Processdelay = 4000000    '10Hz HP
  Par_1 = LS_DIO_Init(1)
  REM enable watchdog time with 1.1 sec
  Par_2 = LS_Watchdog_Init(1, 1, 1100)
  REM LS channels 1…32 as outputs
  Par_3 = LS_Digprog(1, 0Fh)

Event:
  REM set one LS channel to high, rotating from 1 to 32
  Inc Par_10
  If (Par_10 >= 32) Then Par_10 = 0
  Par_11 = Shift_Left(1, Par_10)
  REM reset watchdog, set LS channels, and read back real state
  REM (note: LS_Dig_IO uses LS address 1)
  Par_12 = LS_Dig_IO(1, Par_11)
```

LS_Watchdog_Reset

`LS_Watchdog_Reset` setzt die Watchdog-Zähler auf allen Modulen am LS-Bus auf den jeweiligen Startwert zurück. Die Zähler bleiben aktiv.

Syntax

```
#Include ADWL16.Inc
LS_Watchdog_Reset()
```

Parameter

- / -

Bemerkungen

Solange der Watchdog-Zähler aktiv ist, dekrementiert er den Zählerstand kontinuierlich. Wenn der Zählerstand 0 (Null) erreicht, nimmt das Modul eine Fehlfunktion an und wird gestoppt. Dadurch werden alle Ausgangssignale zurückgesetzt (pull-down Stromsenke).

Setzen Sie den aktiven Watchdog-Zähler während seines Zählvorgangs mindestens einmal zurück auf den Startwert, um die Funktion des Moduls zu gewährleisten. Zum Zurücksetzen können auch modulspezifische Befehle verwendet werden.

Die Watchdog-Funktion dient zur Überwachung des LS-Bus-Moduls.

Siehe auch

LS_DIO_Init, LS_DigProg, LS_Dig_IO, LS_Digout_Long, LS_Digin_Long, LS_Get_Output_Status, LS_Watchdog_Init

Gültig für

HSM-24V + L16

Pinbelegungen

, HSM-24V

Beispiel

```
REM Example process for ADwin-L16 and 2 modules HSM-24V
REM Set process to low priority!
#Include ADwL16.inc

Init:
  Processdelay = 4000000    '10Hz HP
  REM settings for LS module 1
  Par_1 = LS_DIO_Init(1)
  REM enable watchdog with 1.1 s
  Par_2 = LS_Watchdog_Init(1, 1, 1100)
  REM set LS channels 1…32 as outputs
  Par_3 = LS_Digprog(1, 01111b)

  REM settings for LS module 3
  Par_11 = LS_DIO_Init(3)
  REM enable watchdog with 1.1 s
  Par_12 = LS_Watchdog_Init(3, 1, 1100)
  REM set LS channels 1…32 as inputs
  Par_13 = LS_Digprog(3, 0h)

Event:
  REM set one LS channel to high, rotating from 1 to 32
  Inc Par_15
  If (Par_15 >= 32) Then Par_15 = 0
  Par_16 = Shift_Left(1, PAR_15)
  REM set LS channels of module 1
  LS_Digout_Long(1, PAR_16)
  REM read LS channels of module 3
  Par_17 = LS_Digin_Long(3)
  REM reset watchdog
  LS_Watchdog_Reset()
```

3.3.2 LS-Bus + *ADwin-Gold II*

Dieser Abschnitt beschreibt Befehle für die LS-Bus-Schnittstelle an *ADwin-Gold II*:

LS_DIO_Init

T11 TiCo

`LS_DIO_Init` initialisiert ein Modul vom Typ HSM-24V am LS-Bus und gibt den Fehlerstatus zurück.

Syntax

```
#Include ADwinGoldII.inc / GoldIITiCo.inc
ret_val = LS_DIO_Init(channel,ls_module)
```

Parameter

`channel`	Nummer (1, 2) der LS-Bus-Schnittstelle.	LONG
`ls_module`	Eingestellte Moduladresse (1…15) am HSM-Modul auf dem LS-Bus.	LONG
`ret_val`	Rückgabewert, das den Fehlerstatus angibt: -1: Keine Kommunikation mit dem Modul möglich. >0: Bitmuster mit mehreren Fehlerbits. Bit = 0: kein Fehler. Bit = 1: Fehler aufgetreten.	LONG

Bit-Nr.	31…8	7	6	5…4	3	2	1	0
Status	–	Temp2	Temp1	–	WD	Time	Ovr	Par

- :don't care (mit `0CFh` ausmaskieren)
Par:Parity-Fehler bei der Datenübertragung auf dem LS-Bus.
Ovr: Overrun-Fehler bei der Datenübertragung auf dem LS-Bus.
Time: Timeout-Fehler bei der Datenübertragung auf dem LS-Bus.
WD: Watchdog hat ausgelöst. Die Kanaltreiber sind deaktiviert.
Temp1: Übertemperatur am Treiber für Kanäle 1…16. Treiber ist deaktiviert.
Temp2: Übertemperatur am Treiber für Kanäle 17…32. Treiber ist deaktiviert.

Bemerkungen

Die Anweisung soll nur im Abschnitt `INIT:` verwendet werden, weil sie eine lange Ausführungszeit hat.

Die Initialisierung setzt folgende Einstellungen:

- Alle DIO-Kanäle werden als Eingang programmiert.
  Andere Einstellungen siehe `LS_DigProg`.
- Der Status für Überstrom (> ca. 500 mA) wird zurückgesetzt.
- Der Fehlerstatus für Übertemperatur wird zurückgesetzt.
- Der Fehlerstatus für Timeout auf dem LS-Bus wird zurückgesetzt.
- Der Status des Watchdog-Zählers bleibt unverändert.

Das Modul speichert auftretende Fehler unabhängig vom *ADwin*-System. Fehlerbits im Rückgabewert können sich daher auf einen Fehler beziehen, der zu einem früheren Zeitpunkt aufgetreten ist.

Beachten Sie: Ignorieren Sie beim Programmstart das Fehlerbit WD so lange, bis der Watchdog-Zähler mit `LS_Watchdog_Init` konfiguriert ist. Der Watchdog-Zähler startet mit dem Einschalten des Moduls und hat bis zum Programmstart in der Regel schon einen Fehler ausgelöst.

Der Fehler „Übertemperatur“ eines Treibers kann nur auftreten, wenn auf mehreren Kanälen gleichzeitig ein Überstrom im Bereich von 150…500 mA anliegt. Unabhängig davon wird bei einem Überstrom über 500 mA der betroffene Kanal automatisch abgeschaltet.

Die Kanäle des Moduls HSM-24V dürfen nur im Bereich von 0…150 mA betrieben werden. Damit ist sichergestellt, dass das Modul HSM-24V dauerhaft ohne Unterbrechung arbeitet, selbst wenn alle Kanäle gleichzeitig aktiv sind.

Gültig für

HSM-24V + Gold II

Siehe auch

LS_DigProg, LS_Dig_IO, LS_Digout_Long_BS, LS_Digin_Long_BS, LS_Get_Output_Status, LS_Watchdog_Init, LS_Watchdog_Reset

Pinbelegungen

Gold II-LS-Bus, HSM-24V

Beispiel

```
REM Example process for one module HSM-24V and ADwin-Gold II
Rem Wählen Sie das passende Include für ADbasic / TiCoBasic
#Include ADwinGoldII.inc'für ADbasic
Rem #Include GoldIITiCo.incfür TiCoBasic

Init:
  Processdelay = 4000000   '10Hz HP
  Par_1 = LS_DIO_Init(,1)
  REM enable watchdog time with 1.1 sec
  Par_2 = LS_Watchdog_Init(,1,1,1100)
  REM set LS channels 1…32 as outputs
  Par_3 = LS_DigProg(,1,0Fh)

Event:
  REM set one LS channel to high, rotating from 1 to 32
  Inc Par_10
  If (Par_10 >= 32) Then Par_10 = 0
  Par_11 = Shift_Left(1, Par_10)
  REM reset watchdog, set LS channels, and read back real state
  REM (note: LS_Dig_IO uses LS address 1)
  Par_12 = LS_Dig_IO(, Par_11)
```

LS_DigProg

T11 TiCo

`LS_DigProg` programmiert die digitalen Kanäle 1…32 eines Moduls vom Typ HSM-24V am LS-Bus in Gruppen zu 8 als Ein- oder Ausgang.

Syntax

```
#Include ADwinGoldII.inc / GoldIITiCo.inc
ret_val = LS_DigProg(channel,ls_module,pattern)
```

Parameter

`channel`	Nummer (1, 2) der LS-Bus-Schnittstelle.	LONG
`ls_module`	Eingestellte Moduladresse (1…15) am HSM-Modul auf dem LS-Bus.	LONG
`pattern`	Bitmuster, nach dem die Kanäle als Ein- oder Ausgang gesetzt werden: Bit = 0: Kanal als Eingang setzen. Bit = 1: Kanal als Ausgang setzen.	LONG

Bitnr.	31…4	3	2	1	0
Kanalnr.	–	32:25	24:17	16:9	8:1

`ret_val`	Rückgabewert, das den Fehlerstatus angibt: -1: Keine Kommunikation mit dem Modul möglich. >0: Bitmuster mit mehreren Fehlerbits. Bit = 0: kein Fehler. Bit = 1: Fehler aufgetreten.	LONG

Bit-Nr.	31…8	7	6	5…4	3	2	1	0
Status	–	Temp2	Temp1	–	WD	Time	Ovr	Par

- :don't care (mit `0CFh` ausmaskieren)
Par:Parity-Fehler bei der Datenübertragung auf dem LS-Bus.
Ovr: Overrun-Fehler bei der Datenübertragung auf dem LS-Bus.
Time: Timeout-Fehler bei der Datenübertragung auf dem LS-Bus.
WD: Watchdog hat ausgelöst. Die Kanaltreiber sind deaktiviert.
Temp1: Übertemperatur am Treiber für Kanäle 1…16. Treiber ist deaktiviert.
Temp2: Übertemperatur am Treiber für Kanäle 17…32. Treiber ist deaktiviert.

Bemerkungen

Die Anweisung soll nur im Abschnitt `Init:` verwendet werden, weil sie eine lange Ausführungszeit hat.

Nach der Initialisierung mit `LS_DIO_Init` sind alle Kanäle als Eingänge konfiguriert.

Die Kanäle können nur in Gruppen zu je 8 als Ein- oder Ausgang gesetzt werden (nur 4 relevante Bits, die anderen Bits werden ignoriert).

Gültig für

HSM-24V + Gold II

Siehe auch

LS_DIO_Init, LS_Dig_IO, LS_Digout_Long_BS, LS_Digin_Long_BS, LS_Get_Output_Status, LS_Watchdog_Init, LS_Watchdog_Reset

Pinbelegungen

Gold II-LS-Bus, HSM-24V

Beispiel

```
REM Example process for one module HSM-24V and ADwin-Gold II
Rem Wählen Sie das passende Include für ADbasic / TiCoBasic
#Include ADwinGoldII.inc'für ADbasic
Rem #Include GoldIITiCo.incfür TiCoBasic

Init:
  Processdelay = 4000000    '10Hz HP
  Par_1 = LS_DIO_Init(,1)
  REM enable watchdog time with 1.1 sec
  Par_2 = LS_Watchdog_Init(,1,1,1100)
  REM set LS channels 1…32 as outputs
  Par_3 = LS_DigProg(,1,0Fh)

Event:
  REM set one LS channel to high, rotating from 1 to 32
  inc Par_10
  if (Par_10 >= 32) then Par_10 = 0
  Par_11 = shift_left(1,Par_10)
  REM reset watchdog, set LS channels, and read back real state
  REM (note: LS_Dig_IO uses LS address 1)
  Par_12 = LS_Dig_IO(,Par_11)
```

LS_Dig_IO

T11 TiCo

LS_Dig_IO setzt alle Digital-Ausgänge des Moduls HSM-24V am LS-Bus auf den Pegel High oder Low und gibt den Zustand aller Kanäle als Bitmuster zurück.

Syntax

```
#Include ADwinGoldII.inc / GoldIITiCo.inc
ret_val = LS_Dig_IO(channel, pattern)
```

Parameter

channel	Nummer (1, 2) der LS-Bus-Schnittstelle.	LONG
pattern	Bitmuster, mit dem die digitalen Ausgänge gesetzt werden (siehe Tabelle). Bit = 0: Ausgang auf Pegel Low setzen. Bit = 1: Ausgang auf Pegel High setzen.	LONG
ret_val	Bitmuster mit dem Ist-Zustand aller digitalen Kanäle (siehe Tabelle). Bit = 0: Pegel Low liegt an. Bit = 1: Pegel High liegt an.	LONG

Bitnr.	31	30	29	…	2	1	0
Eingang	32	31	30	…	3	2	1

Bemerkungen

LS_Dig_IO arbeitet nur korrekt, wenn folgende Voraussetzungen erfüllt sind:

- Am LS-Bus ist nur ein Modul angeschlossen.
- Das Modul ist vom Typ HSM-24V.
- Am Modul ist die Moduladresse 1 eingestellt.
- Nach Aufruf des Befehls **LS_Dig_IO** wird kein anderer LS-Bus-Befehl mehr verwendet.

Die Kanäle werden mit **LS_DigProg** als Ein- oder Ausgänge programmiert.

Das Bitmuster pattern wird nur für die Kanäle angewendet, die als Ausgang programmiert sind. Bits für Eingänge werden ignoriert.

Der Rückgabewert enthält den tatsächlichen Schaltzustand sowohl von Eingängen wie von Ausgängen. Beachten Sie: Die Eingänge haben einen Filter mit ca. 12 µs Verzögerung.

LS_Dig_IO setzt den Watchdog-Zähler des Moduls auf den Startwert zurück. Der Zähler bleibt aktiv. Der Startwert wird mit **LS_Watchdog_Init** eingestellt.

Setzen Sie den aktiven Watchdog-Zähler während seines Zählvorgangs mindestens einmal zurück, um die Funktion des Moduls zu gewährleisten.

Gültig für

HSM-24V + Gold II

Siehe auch

LS_DIO_Init, LS_DigProg, LS_Digout_Long_BS, LS_Digin_Long_BS, LS_Get_Output_Status, LS_Watchdog_Init, LS_Watchdog_Reset

Pinbelegungen

Gold II-LS-Bus, HSM-24V

Beispiel

```
REM Example process for one module HSM-24V and ADwin-Gold II
Rem Wählen Sie das passende Include für ADbasic / TiCoBasic
#Include ADwinGoldII.inc'für ADbasic
Rem #Include GoldIITiCo.incfür TiCoBasic

Init:
  Processdelay = 4000000    '10Hz HP
  Par_1 = LS_DIO_Init(,1)
  REM enable watchdog time with 1.1 sec
  Par_2 = LS_Watchdog_Init(,1,1,1100)
  REM set LS channels 1…32 as outputs
  Par_3 = LS_DigProg(,1,0Fh)

Event:
  REM set one LS channel to high, rotating from 1 to 32
  Inc Par_10
  If (Par_10 >= 32) Then Par_10 = 0
  Par_11 = Shift_Left(1, Par_10)
  REM reset watchdog, set LS channels, and read back real state
  REM (note: LS_Dig_IO uses LS address 1)
  Par_12 = LS_Dig_IO(channel, Par_11)
```

LS_Digout_Long

T11 TiCo

LS_Digout_Long setzt oder löscht durch den übergebenen 32 Bit-Wert alle Ausgänge auf einem Modul vom Typ HSM-24V am LS-Bus.

Syntax

```
#Include ADwinGoldII.inc / GoldIITiCo.inc
LS_Digout_Long(channel, ls_module, pattern)
```

Parameter

channel	Nummer (1, 2) der LS-Bus-Schnittstelle.	LONG
ls_module	Eingestellte Moduladresse (1…15) am HSM-Modul auf dem LS-Bus.	LONG
pattern	Bitmuster, nach dem die digitalen Ausgänge gesetzt werden: Bit = 0: Ausgang auf Pegel Low setzen. Bit = 1: Ausgang auf Pegel High setzen.	LONG

Bitnr.	31	30	…	2	1	0
Ausgang	32	31	…	3	2	1

Bemerkungen

Die Kanäle werden mit **LS_DigProg** als Ein- oder Ausgänge programmiert.

Das Bitmuster pattern wird nur für die Kanäle angewendet, die als Ausgang programmiert sind. Bits für Eingänge werden ignoriert.

Gültig für

HSM-24V + Gold II

Siehe auch

LS_DIO_Init, LS_DigProg, LS_Dig_IO, LS_Digout_Long_BS, LS_Digin_Long_BS, LS_Digin_Long, LS_Get_Output_Status, LS_Watchdog_Init, LS_Watchdog_Reset

Pinbelegungen

Gold II-LS-Bus, HSM-24V

Beispiel

```
REM Example process for ADwin-Gold II and 2 modules HSM-24V
REM Set process to low priority!
Rem Wählen Sie das passende Include für ADbasic / TiCoBasic
#Include ADwinGoldII.inc'für ADbasic
Rem #Include GoldIITico.incfür TiCoBasic

Init:
  Processdelay = 4000000   '10Hz HP
  REM settings for LS module 1
  Par_1 = LS_DIO_Init(,1)
  REM enable watchdog time with 1.1 sec
  Par_2 = LS_Watchdog_Init(,1,1,1100)
  REM set LS channels 1…32 as outputs
  Par_3 = LS_Digprog(,1,0Fh)

  REM settings for LS module 3
  Par_11 = LS_DIO_Init(,3)
  REM enable watchdog time with 1.1 sec
  Par_12 = LS_Watchdog_Init(,3,1,1100)
  REM set LS channels 1…32 as inputs
  Par_13 = LS_DigProg(,3,0h)

Event:
  REM set one LS channel to high, rotating from 1 to 32
  Inc Par_15
  If (Par_15 >= 32) Then Par_15 = 0
  Par_16 = Shift_Left(1,Par_15)
  REM set LS channels of module 1
  LS_Digout_Long(,1,Par_16)
  REM read LS channels of module 3
  Par_17 = LS_Digin_Long(,3)
  REM reset watchdog
  LS_Watchdog_Reset()
```

LS_Digout_Long_BS

T11 TiCo

LS_Digout_Long_BS setzt oder löscht durch den übergebenen 32 Bit-Wert alle Ausgänge auf einem Modul vom Typ HSM-24V am LS-Bus und gibt den Fehlerstatus zurück.

Syntax

```
#Include ADwinGoldII.inc / GoldIITiCo.inc
LS_Digout_Long_BS(channel, ls_module, pattern,
    status)
```

Parameter

channel	Nummer (1, 2) der LS-Bus-Schnittstelle.	LONG
ls_module	Eingestellte Moduladresse (1…15) am HSM-Modul auf dem LS-Bus.	LONG
pattern	Bitmuster, nach dem die digitalen Ausgänge gesetzt werden: Bit = 0: Ausgang auf Pegel Low setzen. Bit = 1: Ausgang auf Pegel High setzen.	LONG

Bitnr.	31	30	…	2	1	0
Ausgang	32	31	…	3	2	1

status	Fehlerstatus: 0: O.k. -1: Keine Kommunikation mit dem Modul möglich. >0: Bitmuster mit mehreren Fehlerbits. Bit = 0: kein Fehler. Bit = 1: Fehler aufgetreten.	LONG

Bitnr.	31…8	7	6	5…4	3	2	1	0
Status	–	Temp2	Temp1	–	WD	Time	Ovr	Par

- :don't care (mit 0CFh ausmaskieren)
Par:Parity-Fehler bei der Datenübertragung auf dem LS-Bus.
Ovr: Overrun-Fehler bei der Datenübertragung auf dem LS-Bus.
Time: Timeout-Fehler bei der Datenübertragung auf dem LS-Bus.
WD: Watchdog hat ausgelöst. Die Kanaltreiber sind deaktiviert.
Temp1: Übertemperatur am Treiber für Kanäle 1…16. Treiber ist deaktiviert.
Temp2: Übertemperatur am Treiber für Kanäle 17…32. Treiber ist deaktiviert.

Bemerkungen

Die Kanäle werden mit **LS_DigProg** als Ein- oder Ausgänge programmiert.

Das Bitmuster pattern wird nur für die Kanäle angewendet, die als Ausgang programmiert sind. Bits für Eingänge werden ignoriert.

Gültig für

HSM-24V + Gold II

Siehe auch

LS_DIO_Init, LS_DigProg, LS_Dig_IO, LS_Digin_Long_BS, LS_Digout_Long, LS_Get_Output_Status, LS_Watchdog_Init, LS_Watchdog_Reset

Pinbelegungen

Gold II-LS-Bus, HSM-24V

Beispiel

```
REM Example process for ADwin-Gold II and 2 modules HSM-24V
REM Set process to low priority!
Rem Wählen Sie das passende Include für ADbasic / TiCoBasic
#Include ADwinGoldII.inc'für ADbasic
Rem #Include GoldIITico.incfür TiCoBasic

Init:
  Processdelay = 4000000   '10Hz HP
  REM settings for LS module 1
  Par_1 = LS_DIO_Init(,1)
  REM enable watchdog time with 1.1 sec
  Par_2 = LS_Watchdog_Init(,1,1,1100)
  REM set LS channels 1…32 as outputs
  Par_3 = LS_Digprog(,1,0Fh)

  REM settings for LS module 3
  Par_11 = LS_DIO_Init(,3)
  REM enable watchdog time with 1.1 sec
  Par_12 = LS_Watchdog_Init(,3,1,1100)
  REM set LS channels 1…32 as inputs
  Par_13 = LS_DigProg(,3,0h)

Event:
  REM set one LS channel to high, rotating from 1 to 32
  Inc Par_15
  If (Par_15 >= 32) Then Par_15 = 0
  Par_16 = Shift_Left(1,Par_15)
  REM set LS channels of module 1
  LS_Digout_Long_BS(,1,Par_16,Par_5)
  If (Par_5 <> 0) Then End 'exit on error
  REM read LS channels of module 3
  Par_17 = LS_Digin_Long_BS(,3,Par_6)
  If (Par_6 <> 0) Then End 'exit on error
  REM reset watchdog
  LS_Watchdog_Reset()
```

LS_Digin_Long

T11 TiCo

LS_Digin_Long gibt den Zustand aller Kanäle auf einem Modul vom Typ HSM-24V am LS-Bus als Bitmuster zurück.

Syntax

```
#Include ADwinGoldII.inc / GoldIITiCo.inc
ret_val = LS_Digin_Long(module, channel, ls_module)
```

Parameter

channel	Nummer (1, 2) der LS-Bus-Schnittstelle.	LONG
ls_module	Eingestellte Moduladresse (1…15) am HSM-Modul auf dem LS-Bus.	LONG
ret_val	Bitmuster. Jedes Bit (31…0) entspricht dem Zustand eines digitalen Kanals (siehe Tabelle). Bit = 0: Pegel Low liegt an. Bit = 1: Pegel High liegt an.	LONG

Bitnr.	31	30	…	2	1	0
Kanalnr.	32	31	…	3	2	1

Bemerkungen

Wir empfehlen, die angesprochenen Leitungen zunächst mit der Anweisung **LS_DigProg** als Eingänge zu programmieren.

Gültig für

HSM-24V + Gold II

Siehe auch

LS_DIO_Init, LS_DigProg, LS_Dig_IO, LS_Digin_Long_BS, LS_Digout_Long_BS, LS_Get_Output_Status LS_Digout_Long, LS_Get_Output_Status, LS_Watchdog_Init, LS_Watchdog_Reset

Pinbelegungen

Gold II-LS-Bus, HSM-24V

Beispiel

```
REM Example process for ADwin-Gold II and 2 modules HSM-24V
REM Set process to low priority!
Rem Wählen Sie das passende Include für ADbasic / TiCoBasic
#Include ADwinGoldII.inc'für ADbasic
Rem #Include GoldIITiCo.incfür TiCoBasic

Init:
  Processdelay = 4000000   '10Hz HP
  REM settings for LS module 1
  Par_1 = LS_DIO_Init(,1)
  REM enable watchdog time with 1.1 sec
  Par_2 = LS_Watchdog_Init(,1,1,1100)
  REM set LS channels 1…32 as outputs
  Par_3 = LS_Digprog(,1,0Fh)

  REM settings for LS module 3
  Par_11 = LS_DIO_Init(,3)
  REM enable watchdog time with 1.1 sec
  Par_12 = LS_Watchdog_Init(,3,1,1100)
  REM set LS channels 1…32 as inputs
  Par_13 = LS_DigProg(,3,0h)

Event:
  REM set one LS channel to high, rotating from 1 to 32
  Inc Par_15
  If (Par_15 >= 32) Then Par_15 = 0
  Par_16 = Shift_Left(1, Par_15)
  REM set LS channels of module 1
  LS_Digout_Long(,1,Par_16)
  REM read LS channels of module 3
  Par_17 = LS_Digin_Long(,3)
  REM reset watchdog
  LS_Watchdog_Reset()
```

LS_Digin_Long_BS

T11 TiCo

LS_Digin_Long_BS gibt den Zustand aller Kanäle auf einem Modul vom Typ HSM-24V am LS-Bus als Bitmuster zurück sowie den Fehlerstatus.

Syntax

```
#Include ADwinGoldII.inc / GoldIITiCo.inc
ret_val = LS_Digin_Long_BS(channel, ls_module,
    status)
```

Parameter

channel	Nummer (1, 2) der LS-Bus-Schnittstelle.	LONG
ls_module	Eingestellte Moduladresse (1…15) am HSM-Modul auf dem LS-Bus.	LONG
status	Fehlerstatus: 0: O.k. -1: Keine Kommunikation mit dem Modul möglich. >0: Bitmuster mit mehreren Fehlerbits. Bit = 0: kein Fehler. Bit = 1: Fehler aufgetreten.	LONG

Bitnr.	31…8	7	6	5…4	3	2	1	0
Status	–	Temp2	Temp1	–	WD	Time	Ovr	Par

- :don't care (mit 0CFh ausmaskieren)
Par:Parity-Fehler bei der Datenübertragung auf dem LS-Bus.
Ovr: Overrun-Fehler bei der Datenübertragung auf dem LS-Bus.
Time: Timeout-Fehler bei der Datenübertragung auf dem LS-Bus.
WD: Watchdog hat ausgelöst. Die Kanaltreiber sind deaktiviert.
Temp1: Übertemperatur am Treiber für Kanäle 1…16. Treiber ist deaktiviert.
Temp2: Übertemperatur am Treiber für Kanäle 17…32. Treiber ist deaktiviert.

ret_val	Bitmuster. Jedes Bit (31…0) entspricht dem Zustand eines digitalen Kanals (siehe Tabelle). Bit = 0: Pegel Low liegt an. Bit = 1: Pegel High liegt an.	LONG

Bitnr.	31	30	…	2	1	0
Kanalnr.	32	31	…	3	2	1

Bemerkungen

Wir empfehlen, die angesprochenen Leitungen zunächst mit der Anweisung **LS_DigProg** als Eingänge zu programmieren.

Gültig für

HSM-24V + Gold II

Siehe auch

LS_DIO_Init, LS_DigProg, LS_Dig_IO, LS_Digin_Long, LS_Digout_Long_BS, LS_Digout_Long, LS_Get_Output_Status, LS_Watchdog_Init, LS_Watchdog_Reset

Pinbelegungen

Gold II-LS-Bus, HSM-24V

Beispiel

```
REM Example process for ADwin-Gold II and 2 modules HSM-24V
REM Set process to low priority!
Rem Wählen Sie das passende Include für ADbasic / TiCoBasic
#Include ADwinGoldII.inc'für ADbasic
Rem #Include GoldIITico.incfür TiCoBasic

Init:
  Processdelay = 4000000   '10Hz HP
  REM set LS module a
  Par_1 = LS_DIO_Init(,)
  REM enable watchdog time with 1.1 sec
  Par_2 = LS_Watchdog_Init(,,1,1100)
  REM set LS channels 1…32 as outputs
  Par_3 = LS_Digprog(,,0Fh)

  REM set LS module b
  Par_11 = LS_DIO_Init(,)
  REM enable watchdog time with 1.1 sec
  Par_12 = LS_Watchdog_Init(,,1,1100)
  REM set LS channels 1…32 as inputs
  Par_13 = LS_DigProg(,,0h)

Event:
  REM set one LS channel to high, rotating from 1 to 32
  Inc Par_15
  If (Par_15 >= 32) Then Par_15 = 0
  Par_16 = Shift_Left(1,Par_15)
  REM set LS channels of module a
  LS_Digout_Long_BS(,,Par_16,Par_5)
  If (Par_5 <> 0) Then End 'exit on error
  REM read LS channels of module b
  Par_17 = LS_Digin_Long_BS(,,Par_6)
  If (Par_6 <> 0) Then End 'exit on error
  REM reset watchdog
  LS_Watchdog_Reset()
```

LS_Get_Output_Status

T11 TiCo

`LS_Get_Output_Status` gibt den Überstrom-Status der Ausgänge auf einem Modul vom Typ HSM-24V am LS-Bus als Bitmuster zurück.

Syntax

```
#Include ADwinGoldII.inc / GoldIITiCo.inc
ret_val = LS_Get_Output_Status(channel, ls_module)
```

Parameter

`channel`	Nummer (1, 2) der LS-Bus-Schnittstelle.	LONG
`ls_module`	Eingestellte Moduladresse (1…15) am HSM-Modul auf dem LS-Bus.	LONG
`ret_val`	Bitmuster. Jedes Bit (31…0) entspricht dem Überstrom-Status eines digitalen Ausgangs (siehe Tabelle). Bit = 0: Normalzustand. Bit = 1: Ausgang wegen Überstrom deaktiviert.	LONG

Bitnr.	31	30	…	2	1	0
Ausgang	32	31	…	3	2	1

Bemerkungen

Der Fehler „Übertemperatur“ eines Treibers kann nur auftreten, wenn auf mehreren Kanälen gleichzeitig ein Überstrom im Bereich von 150…500 mA anliegt. Unabhängig davon wird bei einem Überstrom über 500 mA der betroffene Kanal automatisch abgeschaltet.

Nach dem Fehler „Übertemperatur“ wird das Modul mit `LS_DIO_Init` wieder zurückgesetzt.

Die Kanäle des Moduls HSM-24V dürfen nur im Bereich von 0…150 mA betrieben werden. Damit ist sichergestellt, dass das Modul HSM-24V dauerhaft ohne Unterbrechung arbeitet, selbst wenn alle Kanäle gleichzeitig aktiv sind.

Gültig für

HSM-24V + Gold II

Siehe auch

LS_DIO_Init, LS_DigProg, LS_Dig_IO, LS_Digout_Long_BS, LS_Digin_Long_BS, LS_Watchdog_Init, LS_Watchdog_Reset

Pinbelegungen

Gold II-LS-Bus, HSM-24V

Beispiel

```
REM Example process for ADwin-Gold II and 2 modules HSM-24V
REM Set process to low priority!
Rem Wählen Sie das passende Include für ADbasic / TiCoBasic
#Include ADwinGoldII.inc'für ADbasic
Rem #Include GoldIITiCo.incfür TiCoBasic

Init:
  Processdelay = 4000000    '10Hz HP
  REM set LS module 1
  Par_1 = LS_DIO_Init(,1)
  REM enable watchdog time with 1.1 sec
  Par_2 = LS_Watchdog_Init(,1,1,1100)
  REM set LS channels 1…32 as outputs
  Par_3 = LS_Digprog(,1,0Fh)

  REM set LS module 3
  Par_11 = LS_DIO_Init(,3)
  REM enable watchdog time with 1.1 sec
  Par_12 = LS_Watchdog_Init(,3,1,1100)
  REM set LS channels 1…32 as inputs
  Par_13 = LS_Digprog(,3,0h)

Event:
  REM check for over-current
  Par_5 = LS_Get_Output_Status(,1)
  Par_5 = Par_5 + LS_Get_Output_Status(,3)
  If (Par_5 > 0) Then End   'over-current: Exit program

  REM set one LS channel to high, rotating from 1 to 32
  Inc Par_15
  If (Par_15 >= 32) Then Par_15 = 0
  Par_16 = Shift_Left(1,Par_15)
  REM set LS channels of module 1
  LS_Digout_Long(,1,Par_16)
  REM read LS channels of module 3
  Par_17 = LS_Digin_Long(,3)
  REM reset watchdog
  LS_Watchdog_Reset()
```

LS_Watchdog_Init

T11 TiCo

`LS_Watchdog_Init` aktiviert oder deaktiviert den Watchdog-Zähler eines Moduls am LS-Bus. Beim Aktivieren erhält der Zähler seinen Startwert und wird gestartet.

Syntax

```
#Include ADwinGoldII.inc / GoldIITiCo.inc
ret_val = LS_Watchdog_Init(channel,ls_module,enable,
    time)
```

Parameter

channel	Nummer (1, 2) der LS-Bus-Schnittstelle.	LONG
ls_module	Eingestellte Moduladresse (1…15) am HSM-Modul auf dem LS-Bus.	LONG
enable	Status des Watchdog-Zählers einstellen: 0 : Watchdog-Zähler deaktivieren. 1 : Watchdog-Zähler aktivieren.	LONG
time	Auslösezeit (0…107374) des Zählers in Millisekunden.	LONG
ret_val	Rückgabewert, das den Fehlerstatus angibt: -1: Keine Kommunikation mit dem Modul möglich. >0: Bitmuster mit mehreren Fehlerbits. Bit = 0: kein Fehler. Bit = 1: Fehler aufgetreten.	LONG

Bit-Nr.	31…8	7	6	5…4	3	2	1	0
Status	–	Temp2	Temp1	–	WD	Time	Ovr	Par

- :don't care (mit `0CFh` ausmaskieren)
Par:Parity-Fehler bei der Datenübertragung auf dem LS-Bus.
Ovr: Overrun-Fehler bei der Datenübertragung auf dem LS-Bus.
Time: Timeout-Fehler bei der Datenübertragung auf dem LS-Bus.
WD: Watchdog hat ausgelöst. Die Kanaltreiber sind deaktiviert.
Temp1: Übertemperatur am Treiber für Kanäle 1…16. Treiber ist deaktiviert.
Temp2: Übertemperatur am Treiber für Kanäle 17…32. Treiber ist deaktiviert.

Bemerkungen

Die Anweisung soll nur im Abschnitt `INIT:` verwendet werden, weil sie eine lange Ausführungszeit hat.

Solange der Watchdog-Zähler aktiv ist, dekrementiert er den Zählerstand kontinuierlich. Nach der eingestellten Auslösezeit erreicht der Zählerstand 0 (Null). Das Modul nimmt nun eine Fehlfunktion an und wird gestoppt; dadurch werden alle Ausgangssignale zurückgesetzt.

Nach dem Einschalten des Moduls ist der Zähler auf den Startwert 10 ms eingestellt und der Watchdog ist aktiv.

Setzen Sie den aktiven Watchdog-Zähler während seines Zählvorgangs mindestens einmal zurück, um die Funktion des Moduls zu gewährleisten. Zum Zurücksetzen können modulspezifische Befehle oder `LS_Watchdog_Reset` verwendet werden.

Die Watchdog-Funktion dient zur Verbindungsüberwachung zwischen *ADwin*-System und LS-Bus-Modul.

Gültig für

HSM-24V + Gold II

Siehe auch

LS_DIO_Init, LS_DigProg, LS_Dig_IO, LS_Digout_Long_BS, LS_Digin_Long_BS, LS_Get_Output_Status, LS_Watchdog_Reset

Pinbelegungen

Gold II-LS-Bus, HSM-24V

Beispiel

```
REM Example process for one module HSM-24V and ADwin-Gold II
Rem Wählen Sie das passende Include für ADbasic / TiCoBasic
#Include ADwinGoldII.inc'für ADbasic
Rem #Include GoldIITiCo.incfür TiCoBasic

Init:
  Processdelay = 4000000    '10Hz HP
  Par_1 = LS_DIO_Init(,1)
  REM enable watchdog time with 1.1 sec
  Par_2 = LS_Watchdog_Init(,1,1,1100)
  REM set LS channels 1…32 as outputs
  Par_3 = LS_Digprog(,1,0Fh)

Event:
  REM set one LS channel to high, rotating from 1 to 32
  Inc Par_10
  If (Par_10 >= 32) Then Par_10 = 0
  Par_11 = Shift_Left(1, Par_10)
  REM reset watchdog, set LS channels, and read back real state
  REM (note: LS_Dig_IO uses LS address 1)
  Par_12 = LS_Dig_IO(, Par_11)
```

LS_Watchdog_Reset

T11 TiCo

`LS_Watchdog_Reset` setzt die Watchdog-Zähler auf allen Modulen am LS-Bus auf den jeweiligen Startwert zurück. Die Zähler bleiben aktiv.

Syntax

```
#Include ADwinGoldII.inc / GoldIITiCo.inc
LS_Watchdog_Reset(channel)
```

Parameter

channel	Nummer (1, 2) der LS-Bus-Schnittstelle.	LONG

Bemerkungen

Solange der Watchdog-Zähler aktiv ist, dekrementiert er den Zählerstand kontinuierlich. Wenn der Zählerstand 0 (Null) erreicht, nimmt das Modul eine Fehlfunktion an und wird gestoppt. Dadurch werden alle Ausgangssignale zurückgesetzt (pull-down Stromsenke).

Setzen Sie den aktiven Watchdog-Zähler während seines Zählvorgangs mindestens einmal zurück auf den Startwert, um die Funktion des Moduls zu gewährleisten. Zum Zurücksetzen können auch modulspezifische Befehle verwendet werden.

Die Watchdog-Funktion dient zur Überwachung des LS-Bus-Moduls.

Gültig für

HSM-24V + Gold II

Siehe auch

LS_DIO_Init, LS_DigProg, LS_Dig_IO, LS_Digout_Long_BS, LS_Digin_Long_BS, LS_Get_Output_Status, LS_Watchdog_Init

Pinbelegungen

Gold II-LS-Bus, HSM-24V

Beispiel

```
REM Example process for ADwin-Gold II and 2 modules HSM-24V
REM Set process to low priority!
Rem Wählen Sie das passende Include für ADbasic / TiCoBasic
#Include ADwinGoldII.inc'für ADbasic
Rem #Include GoldIITiCo.incfür TiCoBasic

Init:
  Processdelay = 4000000   '10Hz HP
  REM set LS module 1
  Par_1 = LS_DIO_Init(,1)
  REM enable watchdog time with 1.1 sec
  Par_2 = LS_Watchdog_Init(,1,1,1100)
  REM set LS channels 1…32 as outputs
  Par_3 = LS_Digprog(,1,01111b)

  REM set LS module 3
  Par_11 = LS_DIO_Init(,3)
  REM enable watchdog time with 1.1 sec
  Par_12 = LS_Watchdog_Init(,3,1,1100)
  REM set LS channels 1…32 as inputs
  Par_13 = LS_Digprog(,3,0h)

Event:
  REM set one LS channel to high, rotating from 1 to 32
  Inc Par_15
  If (Par_15 >= 32) Then Par_15 = 0
  Par_16 = Shift_Left(1, Par_15)
  REM set LS channels of module 1
  LS_Digout_Long(,1,Par_16)
  REM read LS channels of module 3
  Par_17 = LS_Digin_Long(,3)
  REM reset watchdog
  LS_Watchdog_Reset()
```

3.3.3 LS-Bus + HSM-24V

Dieser Abschnitt beschreibt folgende Befehle:

LS_DIO_Init

`LS_DIO_Init` initialisiert ein Modul vom Typ HSM-24V am LS-Bus und gibt den Fehlerstatus zurück.

Syntax

```
#Include ADwinPro_All.Inc
ret_val = LS_DIO_Init(module,channel,ls_module)
```

Parameter

`module`	Eingestellte Adresse des Pro-Moduls (1…255).	LONG
`channel`	Nummer (1, 2) der LS-Bus-Schnittstelle auf dem Pro-Modul.	LONG
`ls_module`	Eingestellte Moduladresse (1…15) am HSM-Modul auf dem LS-Bus.	LONG
`ret_val`	Rückgabewert, das den Fehlerstatus angibt: -1: Keine Kommunikation mit dem Modul möglich. >0: Bitmuster mit mehreren Fehlerbits. Bit = 0: kein Fehler. Bit = 1: Fehler aufgetreten.	LONG

Bit-Nr.	31…8	7	6	5…4	3	2	1	0
Status	–	Temp2	Temp1	–	WD	Time	Ovr	Par

- :don't care (mit `0CFh` ausmaskieren)
Par:Parity-Fehler bei der Datenübertragung auf dem LS-Bus.
Ovr: Overrun-Fehler bei der Datenübertragung auf dem LS-Bus.
Time: Timeout-Fehler bei der Datenübertragung auf dem LS-Bus.
WD: Watchdog hat ausgelöst. Die Kanaltreiber sind deaktiviert.
Temp1: Übertemperatur am Treiber für Kanäle 1…16. Treiber ist deaktiviert.
Temp2: Übertemperatur am Treiber für Kanäle 17…32. Treiber ist deaktiviert.

Bemerkungen

Die Anweisung soll nur im Abschnitt `INIT:` verwendet werden, weil sie eine lange Ausführungszeit hat.

Die Initialisierung setzt folgende Einstellungen:

- Alle DIO-Kanäle werden als Eingang programmiert.
  Andere Einstellungen siehe `LS_DigProg`.
- Der Status für Überstrom (> ca. 500 mA) wird zurückgesetzt.
- Der Fehlerstatus für Übertemperatur wird zurückgesetzt.
- Der Fehlerstatus für Timeout auf dem LS-Bus wird zurückgesetzt.
- Der Status des Watchdog-Zählers bleibt unverändert.

Das Modul speichert auftretende Fehler unabhängig vom *ADwin*-System. Fehlerbits im Rückgabewert können sich daher auf einen Fehler beziehen, der zu einem früheren Zeitpunkt aufgetreten ist.

Beachten Sie: Ignorieren Sie beim Programmstart das Fehlerbit WD so lange, bis der Watchdog-Zähler mit `LS_Watchdog_Init` konfiguriert ist. Der Watchdog-Zähler startet mit dem Einschalten des Moduls und hat bis zum Programmstart in der Regel schon einen Fehler ausgelöst.

Der Fehler „Übertemperatur“ eines Treibers kann nur auftreten, wenn auf mehreren Kanälen gleichzeitig ein Überstrom im Bereich von 150…500 mA anliegt. Unabhängig davon wird bei einem Überstrom über 500 mA der betroffene Kanal automatisch abgeschaltet.

Die Kanäle des Moduls HSM-24V dürfen nur im Bereich von 0…150 mA betrieben werden. Damit ist sichergestellt, dass das Modul HSM-24V dauerhaft ohne Unterbrechung arbeitet, selbst wenn alle Kanäle gleichzeitig aktiv sind.

Gültig für

LS-2 Rev. A

Siehe auch

LS_DigProg, LS_Dig_IO, LS_Digout_Long, LS_Digin_Long, LS_Get_Output_Status, LS_Watchdog_Init, LS_Watchdog_Reset

Beispiel

```
Rem Example prozess for one module HSM-24V and ADwin-Pro-LS2
#Include ADwinPro_All.Inc

Init:
  Processdelay = 4000000    '10Hz HP
  Par_1 = LS_DIO_Init(,,1)
  REM enable watchdog time with 1.1 sec on LS module 1
  Par_2 = LS_Watchdog_Init(,,1,1,1100)
  REM set LS channels 1…32 as outputs
  Par_3 = LS_DigProg(,,1,0Fh)

Event:
  Rem set one LS channel to high, rotating from 1 to 32
  Inc Par_10
  If (Par_10 >= 32) then Par_10 = 0
  Par_11 = Shift_Left(1,Par_10)
  REM reset watchdog, set LS channels, and read back real state
  REM (note: LS_Dig_IO uses LS address 1)
  Par_12 = LS_Dig_IO(,,Par_11)
```

LS_DigProg

LS_DigProg programmiert die digitalen Kanäle 1…32 eines Moduls vom Typ HSM-24V am LS-Bus in Gruppen zu 8 als Ein- oder Ausgang.

Syntax

```
#Include ADwinPro_All.Inc
ret_val = LS_DigProg(module,channel,ls_module,
    pattern)
```

Parameter

module	Eingestellte Adresse des Pro-Moduls (1…255).	LONG
channel	Nummer (1, 2) der LS-Bus-Schnittstelle auf dem Pro-Modul.	LONG
ls_module	Eingestellte Moduladresse (1…15) am HSM-Modul auf dem LS-Bus.	LONG
pattern	Bitmuster, nach dem die Kanäle als Ein- oder Ausgang gesetzt werden: Bit = 0: Kanal als Eingang setzen. Bit = 1: Kanal als Ausgang setzen.	LONG

Bitnr.	31…4	3	2	1	0
Kanalnr.	–	32:25	24:17	16:9	8:1

ret_val	Rückgabewert, das den Fehlerstatus angibt: -1: Keine Kommunikation mit dem Modul möglich. >0: Bitmuster mit mehreren Fehlerbits. Bit = 0: kein Fehler. Bit = 1: Fehler aufgetreten.	LONG

Bit-Nr.	31…8	7	6	5…4	3	2	1	0
Status	–	Temp2	Temp1	–	WD	Time	Ovr	Par

- :don't care (mit 0CFh ausmaskieren)
Par:Parity-Fehler bei der Datenübertragung auf dem LS-Bus.
Ovr: Overrun-Fehler bei der Datenübertragung auf dem LS-Bus.
Time: Timeout-Fehler bei der Datenübertragung auf dem LS-Bus.
WD: Watchdog hat ausgelöst. Die Kanaltreiber sind deaktiviert.
Temp1: Übertemperatur am Treiber für Kanäle 1…16. Treiber ist deaktiviert.
Temp2: Übertemperatur am Treiber für Kanäle 17…32. Treiber ist deaktiviert.

Bemerkungen

Die Anweisung soll nur im Abschnitt **Init:** verwendet werden, weil sie eine lange Ausführungszeit hat.

Nach der Initialisierung mit **LS_DIO_Init** sind alle Kanäle als Eingänge konfiguriert.

Die Kanäle können nur in Gruppen zu je 8 als Ein- oder Ausgang gesetzt werden (nur 4 relevante Bits, die anderen Bits werden ignoriert).

Gültig für

LS-2 Rev. A

Siehe auch

LS_DIO_Init, LS_Dig_IO, LS_Digout_Long, LS_Digin_Long, LS_Get_Output_Status, LS_Watchdog_Init, LS_Watchdog_Reset

Beispiel

```
Rem Example prozess for one module HSM-24V and ADwin-Pro-LS2
#Include ADwinPro_All.Inc

Init:
  Processdelay = 4000000    '10Hz HP
  Par_1 = LS_DIO_Init(,,1)
  REM enable watchdog time with 1.1 sec on LS module 1
  Par_2 = LS_Watchdog_Init(,,1,1,1100)
  Rem LS channels 1…32 as outputs
  Par_3 = LS_DigProg(,,1,0Fh)

Event:
  Rem set LS one channel to high, rotating from 1 to 32
  Inc Par_10
  If (Par_10 >= 32) then Par_10 = 0
  Par_11 = Shift_Left(1,Par_10)
  REM reset watchdog, set LS channels, and read back real state
  REM (note: LS_Dig_IO uses LS address 1)
  Par_12 = LS_Dig_IO(1,2,Par_11)
```

LS_Dig_IO

LS_Dig_IO setzt alle Digital-Ausgänge des Moduls HSM-24V am LS-Bus auf den Pegel High oder Low und gibt den Zustand aller Kanäle als Bitmuster zurück.

Syntax

```
#Include ADwinPro_All.Inc
ret_val = LS_Dig_IO(module, channel, pattern)
```

Parameter

module	Eingestellte Adresse des Pro-Moduls (1…255).	LONG
channel	Nummer (1, 2) der LS-Bus-Schnittstelle auf dem Pro-Modul.	LONG
pattern	Bitmuster, mit dem die digitalen Ausgänge gesetzt werden (siehe Tabelle). Bit = 0: Ausgang auf Pegel Low setzen. Bit = 1: Ausgang auf Pegel High setzen.	LONG
ret_val	Bitmuster mit dem Ist-Zustand aller digitalen Kanäle (siehe Tabelle). Bit = 0: Pegel Low liegt an. Bit = 1: Pegel High liegt an.	LONG

Bitnr.	31	30	29	…	2	1	0
Eingang	32	31	30	…	3	2	1

Bemerkungen

LS_Dig_IO arbeitet nur korrekt, wenn folgende Voraussetzungen erfüllt sind:

- Am LS-Bus ist nur ein Modul angeschlossen.
- Das Modul ist vom Typ HSM-24V.
- Am Modul ist die Moduladresse 1 eingestellt.
- Nach Aufruf des Befehls **LS_Dig_IO** wird kein anderer LS-Bus-Befehl mehr verwendet.

Die Kanäle werden mit **LS_DigProg** als Ein- oder Ausgänge programmiert.

Das Bitmuster pattern wird nur für die Kanäle angewendet, die als Ausgang programmiert sind. Bits für Eingänge werden ignoriert.

Der Rückgabewert enthält den tatsächlichen Schaltzustand sowohl von Eingängen wie von Ausgängen. Beachten Sie: Die Eingänge haben einen Filter mit ca. 12µs Verzögerung.

LS_Dig_IO setzt den Watchdog-Zähler des Moduls auf den Startwert zurück. Der Zähler bleibt aktiv. Der Startwert wird mit **LS_Watchdog_Init** eingestellt.

Setzen Sie den aktiven Watchdog-Zähler während seines Zählvorgangs mindestens einmal zurück, um die Funktion des Moduls zu gewährleisten.

Gültig für

LS-2 Rev. A

Siehe auch

LS_DIO_Init, LS_DigProg, LS_Digout_Long, LS_Digin_Long, LS_Get_Output_Status, LS_Watchdog_Init, LS_Watchdog_Reset

Beispiel

```
Rem Example prozess for one module HSM-24V and ADwin-Pro-LS2
#Include ADwinPro_All.Inc

Init:
  Processdelay = 4000000    '10Hz HP
  Par_1 = LS_DIO_Init(,,1)
  REM enable watchdog time with 1.1 sec on LS module 1
  Par_2 = LS_Watchdog_Init(,,1,1,1100)
  Rem LS channels 1…32 as outputs
  Par_3 = LS_DigProg(,,1,0Fh)

Event:
  Rem check for over-current
  Par_5 = LS_Get_Output_Status(,,1)
  If (Par_5>0) Then End     'over-current: Exit program

  Rem set one LS channel to high, rotating from 1 to 32
  Inc Par_10
  If (Par_10 >= 32) Then Par_10 = 0
  Par_11 = Shift_Left(1,Par_10)
  REM reset watchdog, set LS channels, and read back real state
  REM (note: LS_Dig_IO uses LS address 1)
  Par_12 = LS_Dig_IO(,,Par_11)
```

LS_Digout_Long

LS_Digout_Long setzt oder löscht durch den übergebenen 32 Bit-Wert alle Ausgänge auf einem Modul vom Typ HSM-24V am LS-Bus.

Syntax

```
#Include ADwinPro_All.Inc
LS_Digout_Long(module, channel, ls_module, pattern)
```

Parameter

module	Eingestellte Adresse des Pro-Moduls (1…255).	LONG
channel	Nummer (1, 2) der LS-Bus-Schnittstelle auf dem Pro-Modul.	LONG
ls_module	Eingestellte Moduladresse (1…15) am HSM-Modul auf dem LS-Bus.	LONG
pattern	Bitmuster, nach dem die digitalen Ausgänge gesetzt werden: Bit = 0: Ausgang auf Pegel Low setzen. Bit = 1: Ausgang auf Pegel High setzen.	LONG

Bitnr.	31	30	…	2	1	0
Ausgang	32	31	…	3	2	1

Bemerkungen

Die Kanäle werden mit **LS_DigProg** als Ein- oder Ausgänge programmiert.

Das Bitmuster pattern wird nur für die Kanäle angewendet, die als Ausgang programmiert sind. Bits für Eingänge werden ignoriert.

Gültig für

- / -

Siehe auch

LS_DIO_Init, LS_DigProg, LS_Dig_IO, LS_Digin_Long, LS_Get_Output_Status, LS_Watchdog_Init, LS_Watchdog_Reset

Beispiel

```
Rem Example process for ADwin-Pro and 2 modules HSM-24V
Rem Set process to low priority!
#Include ADwinPro_All.Inc

Init:
  Processdelay = 4000000    '10Hz HP
  Rem Settings for LS module 2
  Par_1 = LS_DIO_Init(,,2)
  Rem watchdog time 1.1 sec
  Par_2 = LS_Watchdog_Init(,,2,1,1100)
  Rem LS channels 1…32 as outputs
  Par_3 = LS_DigProg(,,2,0Fh)

  Rem Settings for LS module 4
  Par_11 = LS_DIO_Init(,,4)
  Rem watchdog time 1.1 sec
  Par_12 = LS_Watchdog_Init(,,4,1,1100)
  Rem LS channels 1…32 as inputs
  Par_13 = LS_DigProg(,,4,0h)

Event:
  Rem set one LS channel to high, rotating from 1 to 32
  Inc Par_15
  If (Par_15 >= 32) then Par_15 = 0
  Par_16 = Shift_Left(1,Par_15)
  Rem set channels of LS module 2
  LS_Digout_Long(,,2,Par_16)
  Rem read channels of LS module 4
  Par_17 = LS_Digin_Long(,,4)
  Rem reset watchdog
  LS_Watchdog_Reset(,)
```

LS_Digout_Long_BS

LS_Digout_Long_BS setzt oder löscht durch den übergebenen 32 Bit-Wert alle Ausgänge auf einem Modul vom Typ HSM-24V am LS-Bus und gibt den Fehlerstatus zurück.

Syntax

```
#Include ADwinPro_All.Inc
LS_Digout_Long_BS(module, channel, ls_module,
    pattern, status)
```

Parameter

module	Eingestellte Adresse des Pro-Moduls (1…255).	LONG
channel	Nummer (1, 2) der LS-Bus-Schnittstelle auf dem Pro-Modul.	LONG
ls_module	Eingestellte Moduladresse (1…15) am HSM-Modul auf dem LS-Bus.	LONG
pattern	Bitmuster, nach dem die digitalen Ausgänge gesetzt werden: Bit = 0: Ausgang auf Pegel Low setzen. Bit = 1: Ausgang auf Pegel High setzen.	LONG

Bitnr.	31	30	…	2	1	0
Ausgang	32	31	…	3	2	1

status	Fehlerstatus: 0: O.k. -1: Keine Kommunikation mit dem Modul möglich. >0: Bitmuster mit mehreren Fehlerbits. Bit = 0: kein Fehler. Bit = 1: Fehler aufgetreten.	LONG

Bitnr.	31…8	7	6	5…4	3	2	1	0
Status	–	Temp2	Temp1	–	WD	Time	Ovr	Par

- :don't care (mit 0CFh ausmaskieren)
Par:Parity-Fehler bei der Datenübertragung auf dem LS-Bus.
Ovr: Overrun-Fehler bei der Datenübertragung auf dem LS-Bus.
Time: Timeout-Fehler bei der Datenübertragung auf dem LS-Bus.
WD: Watchdog hat ausgelöst. Die Kanaltreiber sind deaktiviert.
Temp1: Übertemperatur am Treiber für Kanäle 1…16. Treiber ist deaktiviert.
Temp2: Übertemperatur am Treiber für Kanäle 17…32. Treiber ist deaktiviert.

Bemerkungen

Die Kanäle werden mit **LS_DigProg** als Ein- oder Ausgänge programmiert.

Das Bitmuster pattern wird nur für die Kanäle angewendet, die als Ausgang programmiert sind. Bits für Eingänge werden ignoriert.

Gültig für

- / -

Siehe auch

LS_DIO_Init, LS_DigProg, LS_Dig_IO, LS_Digin_Long, LS_Get_Output_Status, LS_Watchdog_Init, LS_Watchdog_Reset

Beispiel

```
Rem Example process for ADwin-Pro and 2 modules HSM-24V
Rem Set process to low priority!
#Include ADwinPro_All.Inc

Init:
  Processdelay = 4000000    '10Hz HP
  Rem Settings for LS module 2
  Par_1 = LS_DIO_Init(,,2)
  Rem watchdog time 1.1 sec
  Par_2 = LS_Watchdog_Init(,,2,1,1100)
  Rem LS channels 1…32 as outputs
  Par_3 = LS_DigProg(,,2,0Fh)

  Rem Settings for LS module 4
  Par_11 = LS_DIO_Init(,,4)
  Rem watchdog time 1.1 sec
  Par_12 = LS_Watchdog_Init(,,4,1,1100)
  Rem LS channels 1…32 as inputs
  Par_13 = LS_DigProg(,,4,0h)

Event:
  Rem set one LS channel to high, rotating from 1 to 32
  Inc Par_15
  If (Par_15 >= 32) then Par_15 = 0
  Par_16 = Shift_Left(1,Par_15)
  Rem set channels of LS module 2
  LS_Digout_Long_BS(,,2,Par_16,Par_5)
  If (Par_5 <> 0) Then End 'exit on error
  Rem read channels of LS module 4
  Par_17 = LS_Digin_Long_BS(,,4,Par_6)
  If (Par_6 <> 0) Then End 'exit on error
  Rem reset watchdog
  LS_Watchdog_Reset(,)
```

LS_Digin_Long

LS_Digin_Long gibt den Zustand aller Kanäle auf einem Modul vom Typ HSM-24V am LS-Bus als Bitmuster zurück.

Syntax

```
#Include ADwinPro_All.Inc
ret_val = LS_Digin_Long(module, channel, ls_module)
```

Parameter

module	Eingestellte Adresse des Pro-Moduls (1…255).	LONG
channel	Nummer (1, 2) der LS-Bus-Schnittstelle auf dem Pro-Modul.	LONG
ls_module	Eingestellte Moduladresse (1…15) am HSM-Modul auf dem LS-Bus.	LONG
ret_val	Bitmuster. Jedes Bit (31…0) entspricht dem Zustand eines digitalen Kanals (siehe Tabelle). Bit = 0: Pegel Low liegt an. Bit = 1: Pegel High liegt an.	LONG

Bitnr.	31	30	…	2	1	0
Kanalnr.	32	31	…	3	2	1

Bemerkungen

Wir empfehlen, die angesprochenen Leitungen zunächst mit der Anweisung **LS_DigProg** als Eingänge zu programmieren.

Gültig für

- / -

Siehe auch

LS_DIO_Init, LS_DigProg, LS_Dig_IO, LS_Digout_Long, LS_Get_Output_Status, LS_Watchdog_Init, LS_Watchdog_Reset

Beispiel

```
Rem Example process for ADwin-Pro and 2 modules HSM-24V
Rem Set process to low priority!
#Include ADwinPro_All.Inc

Init:
  Processdelay = 4000000    '10Hz HP
  Rem Settings for LS module 2
  Par_1 = LS_DIO_Init(,,2)
  Rem watchdog time 1.1 sec
  Par_2 = LS_Watchdog_Init(,,2,1,1100)
  Rem LS channels 1…32 as outputs
  Par_3 = LS_DigProg(,,2,0Fh)

  Rem Settings for LS module 4
  Par_11 = LS_DIO_Init(,,4)
  Rem watchdog time 1.1 sec
  Par_12 = LS_Watchdog_Init(,,4,1,1100)
  Rem LS channels 1…32 as inputs
  Par_13 = LS_DigProg(,,4,0h)

Event:
  Rem set one LS channel to high, rotating from 1 to 32
  Inc Par_15
  If (Par_15 >= 32) then Par_15 = 0
  Par_16 = Shift_Left(1,Par_15)
  Rem set channels of LS module 2
  LS_Digout_Long(,,2,Par_16)
  Rem read channels of LS module 4
  Par_17 = LS_Digin_Long(,,4)
  Rem reset watchdog
  LS_Watchdog_Reset(,)
```

LS_Digin_Long_BS

LS_Digin_Long_BS gibt den Zustand aller Kanäle auf einem Modul vom Typ HSM-24V am LS-Bus als Bitmuster zurück sowie den Fehlerstatus.

Syntax

```
#Include ADwinPro_All.Inc
ret_val = LS_Digin_Long_BS(module, channel,
    ls_module, status)
```

Parameter

module	Eingestellte Adresse des Pro-Moduls (1…255).	LONG
channel	Nummer (1, 2) der LS-Bus-Schnittstelle auf dem Pro-Modul.	LONG
ls_module	Eingestellte Moduladresse (1…15) am HSM-Modul auf dem LS-Bus.	LONG
status	Fehlerstatus: 0: O.k. -1: Keine Kommunikation mit dem Modul möglich. >0: Bitmuster mit mehreren Fehlerbits. Bit = 0: kein Fehler. Bit = 1: Fehler aufgetreten.	LONG

Bitnr.	31…8	7	6	5…4	3	2	1	0
Status	–	Temp2	Temp1	–	WD	Time	Ovr	Par

- :don't care (mit 0CFh ausmaskieren)
Par:Parity-Fehler bei der Datenübertragung auf dem LS-Bus.
Ovr: Overrun-Fehler bei der Datenübertragung auf dem LS-Bus.
Time: Timeout-Fehler bei der Datenübertragung auf dem LS-Bus.
WD: Watchdog hat ausgelöst. Die Kanaltreiber sind deaktiviert.
Temp1: Übertemperatur am Treiber für Kanäle 1…16. Treiber ist deaktiviert.
Temp2: Übertemperatur am Treiber für Kanäle 17…32. Treiber ist deaktiviert.

ret_val	Bitmuster. Jedes Bit (31…0) entspricht dem Zustand eines digitalen Kanals (siehe Tabelle). Bit = 0: Pegel Low liegt an. Bit = 1: Pegel High liegt an.	LONG

Bitnr.	31	30	…	2	1	0
Kanalnr.	32	31	…	3	2	1

Bemerkungen

Wir empfehlen, die angesprochenen Leitungen zunächst mit der Anweisung **LS_DigProg** als Eingänge zu programmieren.

Gültig für

- / -

Siehe auch

LS_DIO_Init, LS_DigProg, LS_Dig_IO, LS_Digout_Long, LS_Get_Output_Status, LS_Watchdog_Init, LS_Watchdog_Reset

Beispiel

```
Rem Example process for ADwin-Pro and 2 modules HSM-24V
Rem Set process to low priority!
#Include ADwinPro_All.Inc

Init:
  Processdelay = 4000000    '10Hz HP
  Rem Settings for LS module 2
  Par_1 = LS_DIO_Init(,,2)
  Rem watchdog time 1.1 sec
  Par_2 = LS_Watchdog_Init(,,2,1,1100)
  Rem LS channels 1…32 as outputs
  Par_3 = LS_DigProg(,,2,0Fh)

  Rem Settings for LS module 4
  Par_11 = LS_DIO_Init(,,4)
  Rem watchdog time 1.1 sec
  Par_12 = LS_Watchdog_Init(,,4,1,1100)
  Rem LS channels 1…32 as inputs
  Par_13 = LS_DigProg(,,4,0h)

Event:
  Rem set one LS channel to high, rotating from 1 to 32
  Inc Par_15
  If (Par_15 >= 32) then Par_15 = 0
  Par_16 = Shift_Left(1,Par_15)
  Rem set channels of LS module 2
  LS_Digout_Long_BS(,,2,Par_16,Par_5)
  If (Par_5 <> 0) Then End 'exit on error
  Rem read channels of LS module 4
  Par_17 = LS_Digin_Long_BS(,,4,Par_6)
  If (Par_6 <> 0) Then End 'exit on error
  Rem reset watchdog
  LS_Watchdog_Reset(,)
```

LS_Get_Output_Status

LS_Get_Output_Status gibt den Überstrom-Status der Ausgänge auf einem Modul vom Typ HSM-24V am LS-Bus als Bitmuster zurück.

Syntax

```
#Include ADwinPro_All.Inc
ret_val = LS_Get_Output_Status(module, channel,
    ls_module)
```

Parameter

module	Eingestellte Adresse des Pro-Moduls (1…255).	LONG
channel	Nummer (1, 2) der LS-Bus-Schnittstelle auf dem Pro-Modul.	LONG
ls_module	Eingestellte Moduladresse (1…15) am HSM-Modul auf dem LS-Bus.	LONG
ret_val	Bitmuster. Jedes Bit (31…0) entspricht dem Überstrom-Status eines digitalen Ausgangs (siehe Tabelle). Bit = 0: Normalzustand. Bit = 1: Ausgang wegen Überstrom deaktiviert.	LONG

Bitnr.	31	30	…	2	1	0
Ausgang	32	31	…	3	2	1

Bemerkungen

Der Fehler „Übertemperatur“ eines Treibers kann nur auftreten, wenn auf mehreren Kanälen gleichzeitig ein Überstrom im Bereich von 150…500 mA anliegt. Unabhängig davon wird bei einem Überstrom über 500 mA der betroffene Kanal automatisch abgeschaltet.

Nach dem Fehler „Übertemperatur“ wird das Modul mit **LS_DIO_Init** wieder zurückgesetzt.

Die Kanäle des Moduls HSM-24V dürfen nur im Bereich von 0…150 mA betrieben werden. Damit ist sichergestellt, dass das Modul HSM-24V dauerhaft ohne Unterbrechung arbeitet, selbst wenn alle Kanäle gleichzeitig aktiv sind.

Gültig für

- / -

Siehe auch

LS_DIO_Init, LS_DigProg, LS_Dig_IO, LS_Digout_Long, LS_Digin_Long, LS_Watchdog_Init, LS_Watchdog_Reset

Beispiel

```
Rem Example process for ADwin-Pro and 2 modules HSM-24V
Rem Set process to low priority!
#Include ADwinPro_All.Inc

Init:
  Processdelay = 4000000    '10Hz HP
  Rem Settings for LS module 2
  Par_1 = LS_DIO_Init(,,2)
  Rem watchdog time 1.1 sec
  Par_2 = LS_Watchdog_Init(,,2,1,1100)
  Rem LS channels 1…32 as outputs
  Par_3 = LS_DigProg(,,2,0Fh)

  Rem Settings for LS module 4
  Par_11 = LS_DIO_Init(,,4)
  Rem watchdog time 1.1 sec
  Par_12 = LS_Watchdog_Init(,,4,1,1100)
  Rem LS channels 1…32 as inputs
  Par_13 = LS_DigProg(,,4,0h)

Event:
  Rem check for over-current
  Par_5 = LS_Get_Output_Status(,,2)
  Par_5 = Par_5 + LS_Get_Output_Status(,,4)
  If (Par_5 > 0) Then End  'over-current: Exit program

  Rem set one LS channel to high, rotating from 1 to 32
  Inc Par_15
  If (Par_15 >= 32) then Par_15 = 0
  Par_16 = Shift_Left(1,Par_15)
  Rem set channels of LS module 2
  LS_Digout_Long(,,2,Par_16)
  Rem read channels of LS module 4
  Par_17 = LS_Digin_Long(,,4)
  Rem reset watchdog
  LS_Watchdog_Reset(,)
```

LS_Watchdog_Init

`LS_Watchdog_Init` aktiviert oder deaktiviert den Watchdog-Zähler eines Moduls am LS-Bus. Beim Aktivieren erhält der Zähler seinen Startwert und wird gestartet.

Syntax

```
#Include ADwinPro_All.Inc
ret_val = LS_Watchdog_Init(module, channel,
    ls_module, enable, time)
```

Parameter

`module`	Eingestellte Adresse des Pro-Moduls (1…255).	LONG
`channel`	Nummer (1, 2) der LS-Bus-Schnittstelle auf dem Pro-Modul.	LONG
`ls_module`	Eingestellte Moduladresse (1…15) am HSM-Modul auf dem LS-Bus.	LONG
`enable`	Status des Watchdog-Zählers einstellen: 0 : Watchdog-Zähler deaktivieren. 1 : Watchdog-Zähler aktivieren.	LONG
`time`	Auslösezeit (0…107374) des Zählers in Millisekunden.	LONG
`ret_val`	Rückgabewert, das den Fehlerstatus angibt: -1: Keine Kommunikation mit dem Modul möglich. >0: Bitmuster mit mehreren Fehlerbits. Bit = 0: kein Fehler. Bit = 1: Fehler aufgetreten.	LONG

Bit-Nr.	31…8	7	6	5…4	3	2	1	0
Status	–	Temp2	Temp1	–	WD	Time	Ovr	Par

- :don't care (mit `0CFh` ausmaskieren)
Par:Parity-Fehler bei der Datenübertragung auf dem LS-Bus.
Ovr: Overrun-Fehler bei der Datenübertragung auf dem LS-Bus.
Time: Timeout-Fehler bei der Datenübertragung auf dem LS-Bus.
WD: Watchdog hat ausgelöst. Die Kanaltreiber sind deaktiviert.
Temp1: Übertemperatur am Treiber für Kanäle 1…16. Treiber ist deaktiviert.
Temp2: Übertemperatur am Treiber für Kanäle 17…32. Treiber ist deaktiviert.

Bemerkungen

Die Anweisung soll nur im Abschnitt `INIT:` verwendet werden, weil sie eine lange Ausführungszeit hat.

Solange der Watchdog-Zähler aktiv ist, dekrementiert er den Zählerstand kontinuierlich. Nach der eingestellten Auslösezeit erreicht der Zählerstand 0 (Null). Das Modul nimmt nun eine Fehlfunktion an und wird gestoppt; dadurch werden alle Ausgangssignale zurückgesetzt.

Nach dem Einschalten des Moduls ist der Zähler auf den Startwert 10 ms eingestellt und der Watchdog ist aktiv.

Setzen Sie den aktiven Watchdog-Zähler während seines Zählvorgangs mindestens einmal zurück, um die Funktion des Moduls zu gewährleisten. Zum Zurücksetzen können modulspezifische Befehle oder `LS_Watchdog_Reset` verwendet werden.

Die Watchdog-Funktion dient zur Verbindungsüberwachung zwischen *ADwin*-System und LS-Bus-Modul.

Gültig für

- / -

Siehe auch

LS_DIO_Init, LS_DigProg, LS_Dig_IO, LS_Digout_Long, LS_Digin_Long, LS_Get_Output_Status, LS_Watchdog_Reset

Beispiel

```
Rem Example prozess for one module HSM-24V and ADwin-Pro-LS2
#Include ADwinPro_All.Inc

Init:
  Processdelay = 4000000    '10Hz HP
  Par_1 = LS_DIO_Init(,,1)
  REM enable watchdog time with 1.1 sec on LS module 1
  Par_2 = LS_Watchdog_Init(,,1,1,1100)
  Rem LS channels 1…32 as outputs
  Par_3 = LS_DigProg(,,1,0Fh)

Event:
  Rem check for over-current
  Par_5 = LS_Get_Output_Status(,,1)
  If (Par_5>0) Then End     'over-current: Exit program

  Rem set one LS channel to high, rotating from 1 to 32
  Inc Par_10
  If (Par_10 >= 32) Then Par_10 = 0
  Par_11 = Shift_Left(1,Par_10)
  REM reset watchdog, set LS channels, and read back real state
  REM (note: LS_Dig_IO uses LS address 1)
  Par_12 = LS_Dig_IO(,,Par_11)
```

LS_Watchdog_Reset

`LS_Watchdog_Reset` setzt die Watchdog-Zähler auf allen Modulen am LS-Bus auf den jeweiligen Startwert zurück. Die Zähler bleiben aktiv.

Syntax

```
#Include ADwinPro_All.Inc
LS_Watchdog_Reset(module, channel)
```

Parameter

module	Eingestellte Adresse des Pro-Moduls (1…255).	LONG
channel	Nummer (1, 2) der LS-Bus-Schnittstelle auf dem Pro-Modul.	LONG

Bemerkungen

Solange der Watchdog-Zähler aktiv ist, dekrementiert er den Zählerstand kontinuierlich. Wenn der Zählerstand 0 (Null) erreicht, nimmt das Modul eine Fehlfunktion an und wird gestoppt. Dadurch werden alle Ausgangssignale zurückgesetzt (pull-down Stromsenke).

Setzen Sie den aktiven Watchdog-Zähler während seines Zählvorgangs mindestens einmal zurück auf den Startwert, um die Funktion des Moduls zu gewährleisten. Zum Zurücksetzen können auch modulspezifische Befehle verwendet werden.

Die Watchdog-Funktion dient zur Überwachung des LS-Bus-Moduls.

Gültig für

- / -

Siehe auch

LS_DIO_Init, LS_DigProg, LS_Dig_IO, LS_Digout_Long, LS_Digin_Long, LS_Get_Output_Status, LS_Watchdog_Init

Beispiel

```
Rem Example process for ADwin-Pro and 2 modules HSM-24V
Rem Set process to low priority!
#Include ADwinPro_All.Inc

Init:
  Processdelay = 4000000    '10Hz HP
  Rem Settings for LS module 2
  Par_1 = LS_DIO_Init(,,2)
  Rem watchdog time 1.1 sec
  Par_2 = LS_Watchdog_Init(,,2,1,1100)
  Rem LS channels 1…32 as outputs
  Par_3 = LS_DigProg(,,2,0Fh)

  Rem Settings for LS module 4
  Par_11 = LS_DIO_Init(,,4)
  Rem watchdog time 1.1 sec
  Par_12 = LS_Watchdog_Init(,,4,1,1100)
  Rem LS channels 1…32 as inputs
  Par_13 = LS_DigProg(,,4,0h)

Event:
  Rem set one LS channel to high, rotating from 1 to 32
  Inc Par_15
  If (Par_15 >= 32) then Par_15 = 0
  Par_16 = Shift_Left(1,Par_15)
  Rem set channels of LS module 2
  LS_Digout_Long(,,2,Par_16)
  Rem read channels of LS module 4
  Par_17 = LS_Digin_Long(,,4)
  Rem reset watchdog
  LS_Watchdog_Reset(,)
```

3.3.4 LS-Bus + Pro II

Dieser Abschnitt beschreibt folgende Befehle:

P2_LS_DIO_Init

LS_DIO_Init initialisiert ein Modul vom Typ HSM-24V am LS-Bus und gibt den Fehlerstatus zurück.

Syntax

```
#Include ADwinPro_All.Inc
ret_val = P2_LS_DIO_Init(module,channel,ls_module)
```

Parameter

module	Eingestellte Adresse des Pro-Moduls (1…15).	LONG
channel	Nummer (1, 2) der LS-Bus-Schnittstelle auf dem Pro-Modul.	LONG
ls_module	Eingestellte Moduladresse (1…15) am HSM-Modul auf dem LS-Bus.	LONG
ret_val	Fehlerstatus als Rückgabewert: 0: O.k. -1: Keine Kommunikation mit dem Modul möglich. >0: Bitmuster mit mehreren Fehlerbits. Bit = 0: kein Fehler. Bit = 1: Fehler aufgetreten.	LONG

Bitnr.	31…8	7	6	5…4	3	2	1	0
Status	–	Temp2	Temp1	–	WD	Time	Ovr	Par

- :don't care (mit 0CFh ausmaskieren)
Par:Parity-Fehler bei der Datenübertragung auf dem LS-Bus.
Ovr: Overrun-Fehler bei der Datenübertragung auf dem LS-Bus.
Time: Timeout-Fehler bei der Datenübertragung auf dem LS-Bus.
WD: Watchdog hat ausgelöst. Die Kanaltreiber sind deaktiviert.
Temp1: Übertemperatur am Treiber für Kanäle 1…16. Treiber ist deaktiviert.
Temp2: Übertemperatur am Treiber für Kanäle 17…32. Treiber ist deaktiviert.

Bemerkungen

Die Anweisung soll nur im Abschnitt **INIT:** verwendet werden, weil sie eine lange Ausführungszeit hat.

Initialisieren Sie erst den Watchdog-Zähler, bevor Sie auf den LS-Bus zugreifen.

Die Initialisierung setzt folgende Einstellungen:

- Alle DIO-Kanäle werden als Eingang programmiert.
  Andere Einstellungen siehe **P2_LS_DigProg**.
- Der Status für Überstrom (> ca. 500 mA) wird zurückgesetzt.
- Der Fehlerstatus für Übertemperatur wird zurückgesetzt.
- Der Fehlerstatus für Timeout auf dem LS-Bus wird zurückgesetzt.
- Der Status des Watchdog-Zählers bleibt unverändert.

Das Modul speichert auftretende Fehler unabhängig vom *ADwin*-System. Fehlerbits im Rückgabewert können sich daher auf einen Fehler beziehen, der zu einem früheren Zeitpunkt aufgetreten ist.

Beachten Sie: Ignorieren Sie beim Programmstart das Fehlerbit WD so lange, bis der Watchdog-Zähler mit **P2_LS_Watchdog_Init** konfiguriert ist. Der Watchdog-Zähler startet mit dem Einschalten des Moduls und hat bis zum Programmstart in der Regel schon einen Fehler ausgelöst.

Der Fehler „Übertemperatur“ eines Treibers kann nur auftreten, wenn auf mehreren Kanälen gleichzeitig ein Überstrom im Bereich von 150…500 mA anliegt. Unabhängig davon wird bei einem Überstrom über 500 mA der betroffene Kanal automatisch abgeschaltet.

Die Kanäle des Moduls HSM-24V dürfen nur im Bereich von 0…150 mA betrieben werden. Damit ist sichergestellt, dass das Modul HSM-24V

dauerhaft ohne Unterbrechung arbeitet, selbst wenn alle Kanäle gleichzeitig aktiv sind.

Siehe auch

P2_LS_DigProg, P2_LS_Dig_IO, P2_LS_Digout_Long, P2_LS_Digin_Long, P2_LS_Get_Output_Status, P2_LS_Reset, P2_LS_Watchdog_Init, P2_LS_Watchdog_Reset

Gültig für

LS-2 Rev. E

Beispiel

```
Rem Example process for one module HSM-24V and ADwin-Pro II-LS2
#Include ADwinPro_All.Inc
#Define module 2
#Define channel 1

Init:
  Processdelay = 4000000    '10Hz HP
  Rem reset LS interface
  P2_LS_Reset(module,channel)
  Rem Settings for LS module 1
  Par_1 = P2_LS_DIO_Init(module,channel,1)
  REM enable watchdog time with 1.1 sec
  Par_2 = P2_LS_Watchdog_Init(module,channel,1,1,1100)
  REM set LS channels 1…32 as outputs
  Par_3 = P2_LS_DigProg(module,channel,1,0Fh)

Event:
  REM set one LS channel to high, rotating from 1 to 32
  Inc Par_10
  If (Par_10 >= 32) then Par_10 = 0
  Par_11 = Shift_Left(1, Par_10)
  REM reset watchdog, set LS channels, and read back real state
  REM (note: P2_LS_Dig_IO uses LS address 1)
  Par_12 = P2_LS_Dig_IO(module,channel,Par_11)
```

P2_LS_DigProg

LS_DigProg programmiert die digitalen Kanäle 1…32 eines Moduls vom Typ HSM-24V am LS-Bus in Gruppen zu 8 als Ein- oder Ausgang.

Syntax

```
#Include ADwinPro_All.Inc
ret_val = P2_LS_DigProg(module,channel,ls_module,
    pattern)
```

Parameter

module	Eingestellte Adresse des Pro-Moduls (1…15).	LONG
channel	Nummer (1, 2) der LS-Bus-Schnittstelle auf dem Pro-Modul.	LONG
ls_module	Eingestellte Moduladresse (1…15) am HSM-Modul auf dem LS-Bus.	LONG
pattern	Bitmuster, nach dem die Kanäle als Ein- oder Ausgang gesetzt werden: Bit = 0: Kanal als Eingang setzen. Bit = 1: Kanal als Ausgang setzen.	LONG

Bitnr.	31…4	3	2	1	0
Kanalnr.	–	32:25	24:17	16:9	8:1

ret_val	Fehlerstatus als Rückgabewert: 0: O.k. -1: Keine Kommunikation mit dem Modul möglich. >0: Bitmuster mit mehreren Fehlerbits. Bit = 0: kein Fehler. Bit = 1: Fehler aufgetreten.	LONG

Bitnr.	31…8	7	6	5…4	3	2	1	0
Status	–	Temp2	Temp1	–	WD	Time	Ovr	Par

- :don't care (mit **0CFh** ausmaskieren)
Par:Parity-Fehler bei der Datenübertragung auf dem LS-Bus.
Ovr: Overrun-Fehler bei der Datenübertragung auf dem LS-Bus.
Time: Timeout-Fehler bei der Datenübertragung auf dem LS-Bus.
WD: Watchdog hat ausgelöst. Die Kanaltreiber sind deaktiviert.
Temp1: Übertemperatur am Treiber für Kanäle 1…16. Treiber ist deaktiviert.
Temp2: Übertemperatur am Treiber für Kanäle 17…32. Treiber ist deaktiviert.

Bemerkungen

Die Anweisung soll nur im Abschnitt **Init:** verwendet werden, weil sie eine lange Ausführungszeit hat.

Nach der Initialisierung mit **P2_LS_DIO_Init** sind alle Kanäle als Eingänge konfiguriert.

Die Kanäle können nur in Gruppen zu je 8 als Ein- oder Ausgang gesetzt werden (nur 4 relevante Bits, die anderen Bits werden ignoriert).

Siehe auch

P2_LS_DIO_Init, P2_LS_Dig_IO, P2_LS_Digout_Long, P2_LS_Digin_Long, P2_LS_Get_Output_Status, P2_LS_Reset, P2_LS_Watchdog_Init, P2_LS_Watchdog_Reset

Gültig für

LS-2 Rev. E

Beispiel

```
Rem Example process for one module HSM-24V and ADwin-Pro II-LS2
#Include ADwinPro_All.Inc
#Define module 2
#Define channel 1

Init:
  Processdelay = 4000000    '10Hz HP
  Rem reset LS interface
  P2_LS_Reset(module,channel)
  Rem Settings for LS module 1
  Par_1 = P2_LS_DIO_Init(module,channel,1)
  REM enable watchdog time with 1.1 sec
  Par_2 = P2_LS_Watchdog_Init(module,channel,1,1,1100)
  REM set LS channels 1…32 as outputs
  Par_3 = P2_LS_DigProg(module,channel,1,0Fh)

Event:
  REM set one LS channel to high, rotating from 1 to 32
  Inc Par_10
  If (Par_10 >= 32) then Par_10 = 0
  Par_11 = Shift_Left(1, Par_10)
  REM reset watchdog, set LS channels, and read back real state
  REM (note: P2_LS_Dig_IO uses LS address 1)
  Par_12 = P2_LS_Dig_IO(module,channel,Par_11)
```

P2_LS_Dig_IO

P2_LS_Dig_IO setzt alle Digital-Ausgänge des Moduls HSM-24V am LS-Bus auf den Pegel High oder Low und gibt den Zustand aller Kanäle als Bitmuster zurück.

Syntax

```
#Include ADwinPro_All.Inc
ret_val = P2_LS_Dig_IO(module, channel, pattern)
```

Parameter

module	Eingestellte Adresse des Pro-Moduls (1…15).	LONG
channel	Nummer (1, 2) der LS-Bus-Schnittstelle auf dem Pro-Modul.	LONG
pattern	Bitmuster, mit dem die digitalen Ausgänge gesetzt werden (siehe Tabelle). Bit = 0: Ausgang auf Pegel Low setzen. Bit = 1: Ausgang auf Pegel High setzen.	LONG
ret_val	Bitmuster mit dem Ist-Zustand aller digitalen Kanäle (siehe Tabelle). Bit = 0: Pegel Low liegt an. Bit = 1: Pegel High liegt an.	LONG

Bitnr.	31	30	29	…	2	1	0
Eingang	32	31	30	…	3	2	1

Bemerkungen

P2_LS_Dig_IO arbeitet nur korrekt, wenn folgende Voraussetzungen erfüllt sind:

- Am LS-Bus ist nur ein Modul angeschlossen.
- Das Modul ist vom Typ HSM-24V.
- Am Modul ist die Moduladresse 1 eingestellt.
- Nach Aufruf des Befehls **P2_LS_Dig_IO** wird kein anderer LS-Bus-Befehl mehr verwendet.

Die Kanäle werden mit **P2_LS_DigProg** als Ein- oder Ausgänge programmiert.

Das Bitmuster pattern wird nur für die Kanäle angewendet, die als Ausgang programmiert sind. Bits für Eingänge werden ignoriert.

Der Rückgabewert enthält den tatsächlichen Schaltzustand sowohl von Eingängen wie von Ausgängen. Beachten Sie: Die Eingänge haben einen Filter mit ca. 12µs Verzögerung.

P2_LS_Dig_IO setzt den Watchdog-Zähler des Moduls auf den Startwert zurück. Der Zähler bleibt aktiv. Der Startwert wird mit **P2_LS_Watchdog_Init** eingestellt.

Setzen Sie den aktiven Watchdog-Zähler während seines Zählvorgangs mindestens einmal zurück, um die Funktion des Moduls zu gewährleisten.

P2_LS_DIO_Init, P2_LS_DigProg, P2_LS_Digout_Long, P2_LS_Digin_Long, P2_LS_Get_Output_Status, P2_LS_Reset, P2_LS_Watchdog_Init, P2_LS_Watchdog_Reset

Gültig für

LS-2 Rev. E

Beispiel

```
Rem Example process for one module HSM-24V and ADwin-Pro II-LS2
#Include ADwinPro_All.Inc
#Define module 1
#Define channel 2

Init:
  Processdelay = 4000000    '10Hz HP
  Rem reset LS interface
  P2_LS_Reset(module,channel)
  Par_1 = P2_LS_DIO_Init(module,channel,1)
  Rem watchdog time 1.1 sec
  Par_2 = P2_LS_Watchdog_Init(module,channel,1,1,1100)
  Rem LS channels 1…32 as outputs
  Par_3 = P2_LS_DigProg(module,channel,1,0Fh)

Event:
  Rem check for over-current
  Par_5 = P2_LS_Get_Output_Status(module,channel,1)
  If (Par_5 > 0) Then End   'over-current: Exit program

  Rem set one LS channel to high, rotating from 1 to 32
  Inc Par_10
  If (Par_10 >= 32) Then Par_10 = 0
  Par_11 = Shift_Left(1,Par_10)
  REM reset watchdog, set LS channels, and read back real state
  REM (note: P2_LS_Dig_IO uses LS address 1)
  Par_12 = P2_LS_Dig_IO(module,channel,Par_11)
```

P2_LS_Digout_Long

LS_Digout_Long setzt oder löscht durch den übergebenen 32 Bit-Wert alle Ausgänge auf einem Modul vom Typ HSM-24V am LS-Bus.

Syntax

```
#Include ADwinPro_All.Inc
P2_LS_Digout_Long(module, channel, ls_module,
    pattern)
```

Parameter

module	Eingestellte Adresse des Pro-Moduls (1…15).	LONG
channel	Nummer (1, 2) der LS-Bus-Schnittstelle auf dem Pro-Modul.	LONG
ls_module	Eingestellte Moduladresse (1…15) am HSM-Modul auf dem LS-Bus.	LONG
pattern	Bitmuster, nach dem die digitalen Ausgänge gesetzt werden: Bit = 0: Ausgang auf Pegel Low setzen. Bit = 1: Ausgang auf Pegel High setzen.	LONG

Bitnr.	31	30	…	2	1	0
Ausgang	32	31	…	3	2	1

Bemerkungen

Die Kanäle werden mit **P2_LS_DigProg** als Ein- oder Ausgänge programmiert.

Das Bitmuster pattern wird nur für die Kanäle angewendet, die als Ausgang programmiert sind. Bits für Eingänge werden ignoriert.

Siehe auch

P2_LS_DIO_Init, P2_LS_DigProg, P2_LS_Dig_IO, P2_LS_Digin_Long, P2_LS_Get_Output_Status, P2_LS_Reset, P2_LS_Watchdog_Init, P2_LS_Watchdog_Reset

Gültig für

LS-2 Rev. E

Beispiel

```
Rem Example process for ADwin-Pro and 2 modules HSM-24V
Rem Set process to low priority!
#Include ADwinPro_All.Inc
#Define module 3
#Define channel 2

Init:
  Processdelay = 4000000    '10Hz HP
  Rem reset LS interface
  P2_LS_Reset(module,channel)
  Rem Settings for LS module 2
  Par_1 = P2_LS_DIO_Init(module,channel,2)
  Rem set watchdog time 1.1 sec
  Par_2 = P2_LS_Watchdog_Init(module,channel,2,1,1100)
  Rem LS channels 1…32 as outputs
  Par_3 = P2_LS_DigProg(module,channel,2,01111b)

  Rem Settings for LS module 4
  Par_11 = P2_LS_DIO_Init(module,channel,4)
  Rem set watchdog time 1.1 sec
  Par_12 = P2_LS_Watchdog_Init(module,channel,4,1,1100)
  Rem LS channels 1…32 as inputs
  Par_13 = P2_LS_DigProg(module,channel,4, 0h)

Event:
  Rem set one LS channel to high, rotating from 1 to 32
  Inc Par_10
  If (Par_10 >= 32) then Par_10 = 0
  Par_11 = Shift_Left(1,Par_10)
  Rem set LS channels of LS module 2
  P2_LS_Digout_Long(module,channel,2,Par_11)
  Rem read LS channels of LS module 4
  Par_15 = P2_LS_Digin_Long(module,channel,4)
  Rem reset watchdog
  P2_LS_Watchdog_Reset(module,channel)
```

P2_LS_Digout_Long_BS

LS_Digout_Long_BS setzt oder löscht durch den übergebenen 32 Bit-Wert alle Ausgänge auf einem Modul vom Typ HSM-24V am LS-Bus und gibt den Fehlerstatus zurück.

Syntax

```
#Include ADwinPro_All.Inc
P2_LS_Digout_Long_BS(module, channel, ls_module,
    pattern, status)
```

Parameter

module	Eingestellte Adresse des Pro-Moduls (1…15).	LONG
channel	Nummer (1, 2) der LS-Bus-Schnittstelle auf dem Pro-Modul.	LONG
ls_module	Eingestellte Moduladresse (1…15) am HSM-Modul auf dem LS-Bus.	LONG
pattern	Bitmuster, nach dem die digitalen Ausgänge gesetzt werden: Bit = 0: Ausgang auf Pegel Low setzen. Bit = 1: Ausgang auf Pegel High setzen.	LONG

Bitnr.	31	30	…	2	1	0
Ausgang	32	31	…	3	2	1

status	Fehlerstatus: 0: O.k. -1: Keine Kommunikation mit dem Modul möglich. >0: Bitmuster mit mehreren Fehlerbits. Bit = 0: kein Fehler. Bit = 1: Fehler aufgetreten.	LONG

Bitnr.	31…8	7	6	5…4	3	2	1	0
Status	–	Temp2	Temp1	–	WD	Time	Ovr	Par

- :don't care (mit 0CFh ausmaskieren)
Par:Parity-Fehler bei der Datenübertragung auf dem LS-Bus.
Ovr: Overrun-Fehler bei der Datenübertragung auf dem LS-Bus.
Time: Timeout-Fehler bei der Datenübertragung auf dem LS-Bus.
WD: Watchdog hat ausgelöst. Die Kanaltreiber sind deaktiviert.
Temp1: Übertemperatur am Treiber für Kanäle 1…16. Treiber ist deaktiviert.
Temp2: Übertemperatur am Treiber für Kanäle 17…32. Treiber ist deaktiviert.

Bemerkungen

Die Kanäle werden mit **P2_LS_DigProg** als Ein- oder Ausgänge programmiert.

Das Bitmuster pattern wird nur für die Kanäle angewendet, die als Ausgang programmiert sind. Bits für Eingänge werden ignoriert.

Siehe auch

P2_LS_DIO_Init, P2_LS_DigProg, P2_LS_Dig_IO, P2_LS_Digin_Long, P2_LS_Get_Output_Status, P2_LS_Reset, P2_LS_Watchdog_Init, P2_LS_Watchdog_Reset

Gültig für

LS-2 Rev. E

Beispiel

```
Rem Example process for ADwin-Pro and 2 modules HSM-24V
Rem Set process to low priority!
#Include ADwinPro_All.Inc
#Define module 3
#Define channel 2

Init:
  Processdelay = 4000000    '10Hz HP
  Rem reset LS interface
  P2_LS_Reset(module,channel)
  Rem Settings for LS module 2
  Par_1 = P2_LS_DIO_Init(module,channel,2)
  Rem set watchdog time 1.1 sec
  Par_2 = P2_LS_Watchdog_Init(module,channel,2,1,1100)
  Rem LS channels 1…32 as outputs
  Par_3 = P2_LS_DigProg(module,channel,2,01111b)

  Rem Settings for LS module 4
  Par_11 = P2_LS_DIO_Init(module,channel,4)
  Rem set watchdog time 1.1 sec
  Par_12 = P2_LS_Watchdog_Init(module,channel,4,1,1100)
  Rem LS channels 1…32 as inputs
  Par_13 = P2_LS_DigProg(module,channel,4, 0h)

Event:
  Rem set one LS channel to high, rotating from 1 to 32
  Inc Par_10
  If (Par_10 >= 32) then Par_10 = 0
  Par_11 = Shift_Left(1,Par_10)
  Rem set LS channels of LS module 2
  P2_LS_Digout_Long_BS(module,channel,2,Par_14,Par_5)
  If (Par_5 <> 0) Then End 'exit on error
  Rem read LS channels of LS module 4
  Par_15 = P2_LS_Digin_Long_BS(,,4,Par_6)
  If (Par_6 <> 0) Then End 'exit on error
  Rem reset watchdog
  P2_LS_Watchdog_Reset(module,channel)
```

P2_LS_Digin_Long

LS_Digin_Long gibt den Zustand aller Kanäle auf einem Modul vom Typ HSM-24V am LS-Bus als Bitmuster zurück.

Syntax

```
#Include ADwinPro_All.Inc
ret_val = P2_LS_Digin_Long(module, channel,
    ls_module)
```

Parameter

module	Eingestellte Adresse des Pro-Moduls (1…15).	LONG
channel	Nummer (1, 2) der LS-Bus-Schnittstelle auf dem Pro-Modul.	LONG
ls_module	Eingestellte Moduladresse (1…15) am HSM-Modul auf dem LS-Bus.	LONG
ret_val	Bitmuster. Jedes Bit (31…0) entspricht dem Zustand eines digitalen Kanals (siehe Tabelle). Bit = 0: Pegel Low liegt an. Bit = 1: Pegel High liegt an.	LONG

Bitnr.	31	30	…	2	1	0
Kanalnr.	32	31	…	3	2	1

Bemerkungen

Wir empfehlen, die angesprochenen Leitungen zunächst mit der Anweisung **P2_LS_DigProg** als Eingänge zu programmieren.

Siehe auch

P2_LS_DIO_Init, P2_LS_DigProg, P2_LS_Dig_IO, P2_LS_Digout_Long, P2_LS_Get_Output_Status, P2_LS_Reset, P2_LS_Watchdog_Init, P2_LS_Watchdog_Reset

Gültig für

LS-2 Rev. E

Beispiel

```
Rem Example process for ADwin-Pro and 2 modules HSM-24V
Rem Set process to low priority!
#Include ADwinPro_All.Inc
#Define module 3
#Define channel 2

Init:
  Processdelay = 4000000    '10Hz HP
  Rem reset LS interface
  P2_LS_Reset(module,channel)
  Rem Settings for LS module 2
  Par_1 = P2_LS_DIO_Init(module,channel,2)
  Rem set watchdog time 1.1 sec
  Par_2 = P2_LS_Watchdog_Init(module,channel,2,1,1100)
  Rem LS channels 1…32 as outputs
  Par_3 = P2_LS_DigProg(module,channel,2,01111b)

  Rem Settings for LS module 4
  Par_11 = P2_LS_DIO_Init(module,channel,4)
  Rem set watchdog time 1.1 sec
  Par_12 = P2_LS_Watchdog_Init(module,channel,4,1,1100)
  Rem LS channels 1…32 as inputs
  Par_13 = P2_LS_DigProg(module,channel,4, 0h)

Event:
  Rem set one LS channel to high, rotating from 1 to 32
  Inc Par_10
  If (Par_10 >= 32) then Par_10 = 0
  Par_11 = Shift_Left(1,Par_10)
  Rem set LS channels of LS module 2
  P2_LS_Digout_Long(module,channel,2,Par_11)
  Rem read LS channels of LS module 4
  Par_15 = P2_LS_Digin_Long(module,channel,4)
  Rem reset watchdog
  P2_LS_Watchdog_Reset(module,channel)
```

P2_LS_Digin_Long_BS

LS_Digin_Long_BS gibt den Zustand aller Kanäle auf einem Modul vom Typ HSM-24V am LS-Bus als Bitmuster zurück sowie den Fehlerstatus.

Syntax

```
#Include ADwinPro_All.Inc
ret_val = P2_LS_Digin_Long_BS(module, channel,
    ls_module, status)
```

Parameter

module	Eingestellte Adresse des Pro-Moduls (1…15).	LONG
channel	Nummer (1, 2) der LS-Bus-Schnittstelle auf dem Pro-Modul.	LONG
ls_module	Eingestellte Moduladresse (1…15) am HSM-Modul auf dem LS-Bus.	LONG
status	Fehlerstatus: 0: O.k. -1: Keine Kommunikation mit dem Modul möglich. >0: Bitmuster mit mehreren Fehlerbits. Bit = 0: kein Fehler. Bit = 1: Fehler aufgetreten.	LONG

Bitnr.	31…8	7	6	5…4	3	2	1	0
Status	–	Temp2	Temp1	–	WD	Time	Ovr	Par

- :don't care (mit 0CFh ausmaskieren)
Par:Parity-Fehler bei der Datenübertragung auf dem LS-Bus.
Ovr: Overrun-Fehler bei der Datenübertragung auf dem LS-Bus.
Time: Timeout-Fehler bei der Datenübertragung auf dem LS-Bus.
WD: Watchdog hat ausgelöst. Die Kanaltreiber sind deaktiviert.
Temp1: Übertemperatur am Treiber für Kanäle 1…16. Treiber ist deaktiviert.
Temp2: Übertemperatur am Treiber für Kanäle 17…32. Treiber ist deaktiviert.

ret_val	Bitmuster. Jedes Bit (31…0) entspricht dem Zustand eines digitalen Kanals (siehe Tabelle). Bit = 0: Pegel Low liegt an. Bit = 1: Pegel High liegt an.	LONG

Bitnr.	31	30	…	2	1	0
Kanalnr.	32	31	…	3	2	1

Bemerkungen

Wir empfehlen, die angesprochenen Leitungen zunächst mit der Anweisung **P2_LS_DigProg** als Eingänge zu programmieren.

Siehe auch

P2_LS_DIO_Init, P2_LS_DigProg, P2_LS_Dig_IO, P2_LS_Digout_Long, P2_LS_Get_Output_Status, P2_LS_Reset, P2_LS_Watchdog_Init, P2_LS_Watchdog_Reset

Gültig für

LS-2 Rev. E

Beispiel

```
Rem Example process for ADwin-Pro and 2 modules HSM-24V
Rem Set process to low priority!
#Include ADwinPro_All.Inc
#Define module 3
#Define channel 2

Init:
  Processdelay = 4000000    '10Hz HP
  Rem reset LS interface
  P2_LS_Reset(module,channel)
  Rem Settings for LS module 2
  Par_1 = P2_LS_DIO_Init(module,channel,2)
  Rem set watchdog time 1.1 sec
  Par_2 = P2_LS_Watchdog_Init(module,channel,2,1,1100)
  Rem LS channels 1…32 as outputs
  Par_3 = P2_LS_DigProg(module,channel,2,01111b)

  Rem Settings for LS module 4
  Par_11 = P2_LS_DIO_Init(module,channel,4)
  Rem set watchdog time 1.1 sec
  Par_12 = P2_LS_Watchdog_Init(module,channel,4,1,1100)
  Rem LS channels 1…32 as inputs
  Par_13 = P2_LS_DigProg(module,channel,4, 0h)

Event:
  Rem set one LS channel to high, rotating from 1 to 32
  Inc Par_10
  If (Par_10 >= 32) then Par_10 = 0
  Par_11 = Shift_Left(1,Par_10)
  Rem set LS channels of LS module 2
  P2_LS_Digout_Long_BS(module,channel,2,Par_14,Par_5)
  If (Par_5 <> 0) Then End 'exit on error
  Rem read LS channels of LS module 4
  Par_15 = P2_LS_Digin_Long_BS(,,4,Par_6)
  If (Par_6 <> 0) Then End 'exit on error
  Rem reset watchdog
  P2_LS_Watchdog_Reset(module,channel)
```

P2_LS_Get_Output_Status

LS_Get_Output_Status gibt den Überstrom-Status der Ausgänge auf einem Modul vom Typ HSM-24V am LS-Bus als Bitmuster zurück.

Syntax

```
#Include ADwinPro_All.Inc
ret_val = P2_LS_Get_Output_Status(module, channel,
    ls_module)
```

Parameter

module	Eingestellte Adresse des Pro-Moduls (1…15).	LONG
channel	Nummer (1, 2) der LS-Bus-Schnittstelle auf dem Pro-Modul.	LONG
ls_module	Eingestellte Moduladresse (1…15) am HSM-Modul auf dem LS-Bus.	LONG
ret_val	Bitmuster. Jedes Bit (31…0) entspricht dem Überstrom-Status eines digitalen Ausgangs (siehe Tabelle). Bit = 0: Normalzustand. Bit = 1: Ausgang wegen Überstrom deaktiviert.	LONG

Bitnr.	31	30	…	2	1	0
Ausgang	32	31	…	3	2	1

Bemerkungen

Der Fehler „Übertemperatur“ eines Treibers kann nur auftreten, wenn auf mehreren Kanälen gleichzeitig ein Überstrom im Bereich von 150…500 mA anliegt. Unabhängig davon wird bei einem Überstrom über 500 mA der betroffene Kanal automatisch abgeschaltet.

Nach dem Fehler „Übertemperatur“ wird das Modul mit **P2_LS_DIO_Init** wieder zurückgesetzt.

Die Kanäle des Moduls HSM-24V dürfen nur im Bereich von 0…150 mA betrieben werden. Damit ist sichergestellt, dass das Modul HSM-24V dauerhaft ohne Unterbrechung arbeitet, selbst wenn alle Kanäle gleichzeitig aktiv sind.

Siehe auch

P2_LS_DIO_Init, P2_LS_DigProg, P2_LS_Dig_IO, P2_LS_Digout_Long, P2_LS_Digin_Long, P2_LS_Reset, P2_LS_Watchdog_Init, P2_LS_Watchdog_Reset

Gültig für

LS-2 Rev. E

Beispiel

```
Rem Example process for one module HSM-24V and ADwin-Pro II-LS2
#Include ADwinPro_All.Inc
#Define module 1
#Define channel 2

Init:
  Processdelay = 4000000    '10Hz HP
  Rem reset LS interface
  P2_LS_Reset(module,channel)
  Rem Settings for LS module 2
  Par_1 = P2_LS_DIO_Init(module,channel,2)
  Rem set watchdog time 1.1 sec
  Par_2 = P2_LS_Watchdog_Init(module,channel,2,1,1100)
  Rem LS channels 1…32 as outputs
  Par_3 = P2_LS_DigProg(module,channel,2,01111b)

  Rem Settings for LS module 4
  Par_11 = P2_LS_DIO_Init(module,channel,4)
  Rem set watchdog time 1.1 sec
  Par_12 = P2_LS_Watchdog_Init(module,channel,4,1,1100)
  Rem LS channels 1…32 as inputs
  Par_13 = P2_LS_DigProg(module,channel,4, 0h)

Event:
  Rem check for over-current
  Par_5 = P2_LS_Get_Output_Status(module,channel,2)
  Par_5 = Par_5 + P2_LS_Get_Output_Status(module,channel,4)
  If (Par_5 > 0) Then End  'over-current: Exit program

  Rem set one LS channel to high, rotating from 1 to 32
  Inc Par_10
  If (Par_10 >= 32) then Par_10 = 0
  Par_14 = Shift_Left(1,Par_10)
  Rem set LS channels of LS module 2
  P2_LS_Digout_Long(module,channel,2,Par_14)
  Rem read LS channels of LS module 4
  Par_15 = P2_LS_Digin_Long(module,channel,4)
  Rem reset watchdog
  P2_LS_Watchdog_Reset(module,channel)
```

P2_LS_Reset

LS_Reset setzt die LS-Bus-Schnittstelle auf dem Pro-Modul zurück.

Syntax

```
#Include ADwinPro_All.Inc
P2_LS_Reset(module, channel)
```

Parameter

module	Eingestellte Adresse des Pro-Moduls (1…15).	LONG
channel	Nummer (1, 2) der LS-Bus-Schnittstelle auf dem Pro-Modul.	LONG

Bemerkungen

Setzen Sie die LS-Bus-Schnittstelle am Anfang eines Programms zurück.

Siehe auch

P2_LS_DIO_Init, P2_LS_DigProg, P2_LS_Dig_IO, P2_LS_Digout_Long, P2_LS_Digin_Long, P2_LS_Get_Output_Status, P2_LS_Watchdog_Init, P2_LS_Watchdog_Reset

Gültig für

LS-2 Rev. E

Beispiel

```
Rem Example process for ADwin-Pro and 2 modules HSM-24V
Rem Set process to low priority!
#Include ADwinPro_All.Inc
#Define module 3
#Define channel 2

Init:
  Processdelay = 4000000    '10Hz HP
  Rem reset LS interface
  P2_LS_Reset(module,channel)
  Rem Settings for LS module 2
  Par_1 = P2_LS_DIO_Init(module,channel,2)
  Rem set watchdog time 1.1 sec
  Par_2 = P2_LS_Watchdog_Init(module,channel,2,1,1100)
  Rem LS channels 1…32 as outputs
  Par_3 = P2_LS_DigProg(module,channel,2,01111b)

  Rem Settings for LS module 4
  Par_11 = P2_LS_DIO_Init(module,channel,4)
  Rem set watchdog time 1.1 sec
  Par_12 = P2_LS_Watchdog_Init(module,channel,4,1,1100)
  Rem LS channels 1…32 as inputs
  Par_13 = P2_LS_DigProg(module,channel,4, 0h)

Event:
  Rem set one LS channel to high, rotating from 1 to 32
  Inc Par_10
  If (Par_10 >= 32) then Par_10 = 0
  Par_11 = Shift_Left(1,Par_10)
  Rem set LS channels of LS module 2
  P2_LS_Digout_Long(module,channel,2,Par_11)
  Rem read LS channels of LS module 4
  Par_15 = P2_LS_Digin_Long(module,channel,4)
  Rem reset watchdog
  P2_LS_Watchdog_Reset(module,channel)
```

P2_LS_ Watchdog_Init

`LS_Watchdog_Init` aktiviert oder deaktiviert den Watchdog-Zähler eines Moduls am LS-Bus. Beim Aktivieren erhält der Zähler seinen Startwert und wird gestartet.

Syntax

```
#Include ADwinPro_All.Inc
ret_val = P2_LS_Watchdog_Init(module, channel,
    ls_module, enable, time)
```

Parameter

`module`	Eingestellte Adresse des Pro-Moduls (1…15).	LONG
`channel`	Nummer (1, 2) der LS-Bus-Schnittstelle auf dem Pro-Modul.	LONG
`ls_module`	Eingestellte Moduladresse (1…15) am HSM-Modul auf dem LS-Bus.	LONG
`enable`	Status des Watchdog-Zählers einstellen: 0: Watchdog-Zähler deaktivieren. 1: Watchdog-Zähler aktivieren.	LONG
`time`	Auslösezeit (0…107374) des Zählers in Millisekunden.	LONG
`ret_val`	Fehlerstatus als Rückgabewert: 0: O.k. -1: Keine Kommunikation mit dem Modul möglich. >0: Bitmuster mit mehreren Fehlerbits. Bit = 0: kein Fehler. Bit = 1: Fehler aufgetreten.	LONG

Bitnr.	31…8	7	6	5…4	3	2	1	0
Status	–	Temp2	Temp1	–	WD	Time	Ovr	Par

- :don't care (mit `0CFh` ausmaskieren)
Par:Parity-Fehler bei der Datenübertragung auf dem LS-Bus.
Ovr: Overrun-Fehler bei der Datenübertragung auf dem LS-Bus.
Time: Timeout-Fehler bei der Datenübertragung auf dem LS-Bus.
WD: Watchdog hat ausgelöst. Die Kanaltreiber sind deaktiviert.
Temp1: Übertemperatur am Treiber für Kanäle 1…16. Treiber ist deaktiviert.
Temp2: Übertemperatur am Treiber für Kanäle 17…32. Treiber ist deaktiviert.

Bemerkungen

Die Anweisung soll nur im Abschnitt `Init:` verwendet werden, weil sie eine lange Ausführungszeit hat.

Solange der Watchdog-Zähler aktiv ist, dekrementiert er den Zählerstand kontinuierlich. Nach der eingestellten Auslösezeit erreicht der Zählerstand 0 (Null). Das Modul nimmt nun eine Fehlfunktion an und wird gestoppt; dadurch werden alle Ausgangssignale zurückgesetzt.

Nach dem Einschalten des Moduls ist der Zähler auf den Startwert 10ms eingestellt und der Watchdog ist aktiv. Initialisieren Sie daher zuerst den Watchdog-Zähler, bevor Sie auf den LS-Bus zugreifen.

Setzen Sie den aktiven Watchdog-Zähler während seines Zählvorgangs mindestens einmal zurück, um die Funktion des Moduls zu gewährleisten. Zum Zurücksetzen können modulspezifische Befehle oder `P2_LS_Watchdog_Reset` verwendet werden.

Die Watchdog-Funktion dient zur Verbindungsüberwachung zwischen *ADwin*-System und LS-Bus-Modul.

Siehe auch

P2_LS_DIO_Init, P2_LS_DigProg, P2_LS_Dig_IO, P2_LS_Digout_Long, P2_LS_Digin_Long, P2_LS_Get_Output_Status, P2_LS_Reset, P2_LS_Watchdog_Reset

Gültig für

LS-2 Rev. E

Beispiel

```
Rem Example process for one module HSM-24V and ADwin-Pro II-LS2
#Include ADwinPro_All.Inc
#Define module 2
#Define channel 1

Init:
  Processdelay = 4000000    '10Hz HP
  Rem reset LS interface
  P2_LS_Reset(module,channel)
  Rem Settings for LS module 1
  Par_1 = P2_LS_DIO_Init(module,channel,1)
  REM enable watchdog time with 1.1 sec
  Par_2 = P2_LS_Watchdog_Init(module,channel,1,1,1100)
  REM set LS channels 1…32 as outputs
  Par_3 = P2_LS_DigProg(module,channel,1,0Fh)

Event:
  REM set one LS channel to high, rotating from 1 to 32
  Inc Par_10
  If (Par_10 >= 32) then Par_10 = 0
  Par_11 = Shift_Left(1, Par_10)
  REM reset watchdog, set LS channels, and read back real state
  REM (note: P2_LS_Dig_IO uses LS address 1)
  Par_12 = P2_LS_Dig_IO(module,channel,Par_11)
```

P2_LS_ Watchdog_Reset

`LS_Watchdog_Reset` setzt die Watchdog-Zähler auf allen Modulen am LS-Bus auf den jeweiligen Startwert zurück. Die Zähler bleiben aktiv.

Syntax

```
#Include ADwinPro_All.Inc
P2_LS_Watchdog_Reset(module, channel)
```

Parameter

`module`	Eingestellte Adresse des Pro-Moduls (1…15).	LONG
`channel`	Nummer (1, 2) der LS-Bus-Schnittstelle auf dem Pro-Modul.	LONG

Bemerkungen

Solange der Watchdog-Zähler aktiv ist, dekrementiert er den Zählerstand kontinuierlich. Wenn der Zählerstand 0 (Null) erreicht, nimmt das Modul eine Fehlfunktion an und wird gestoppt. Dadurch werden alle Ausgangssignale zurückgesetzt (pull-down Stromsenke).

Setzen Sie den aktiven Watchdog-Zähler während seines Zählvorgangs mindestens einmal zurück auf den Startwert, um die Funktion des Moduls zu gewährleisten. Zum Zurücksetzen können auch modulspezifische Befehle verwendet werden.

Die Watchdog-Funktion dient zur Überwachung des LS-Bus-Moduls.

Siehe auch

P2_LS_DIO_Init, P2_LS_DigProg, P2_LS_Dig_IO, P2_LS_Digout_Long, P2_LS_Digin_Long, P2_LS_Get_Output_Status, P2_LS_Reset, P2_LS_Watchdog_Init

Gültig für

LS-2 Rev. E

Beispiel

```
Rem Example process for ADwin-Pro and 2 modules HSM-24V
Rem Set process to low priority!
#Include ADwinPro_All.Inc
#Define module 3
#Define channel 2

Init:
  Processdelay = 4000000    '10Hz HP
  Rem reset LS interface
  P2_LS_Reset(module,channel)
  Rem Settings for LS module 2
  Par_1 = P2_LS_DIO_Init(module,channel,2)
  Rem set watchdog time 1.1 sec
  Par_2 = P2_LS_Watchdog_Init(module,channel,2,1,1100)
  Rem LS channels 1…32 as outputs
  Par_3 = P2_LS_DigProg(module,channel,2,01111b)

  Rem Settings for LS module 4
  Par_11 = P2_LS_DIO_Init(module,channel,4)
  Rem set watchdog time 1.1 sec
  Par_12 = P2_LS_Watchdog_Init(module,channel,4,1,1100)
  Rem LS channels 1…32 as inputs
  Par_13 = P2_LS_DigProg(module,channel,4, 0h)

Event:
  Rem set one LS channel to high, rotating from 1 to 32
  Inc Par_10
  If (Par_10 >= 32) then Par_10 = 0
  Par_11 = Shift_Left(1,Par_10)
  Rem set LS channels of LS module 2
  P2_LS_Digout_Long(module,channel,2,Par_11)
  Rem read LS channels of LS module 4
  Par_15 = P2_LS_Digin_Long(module,channel,4)
  Rem reset watchdog
  P2_LS_Watchdog_Reset(module,channel)
```