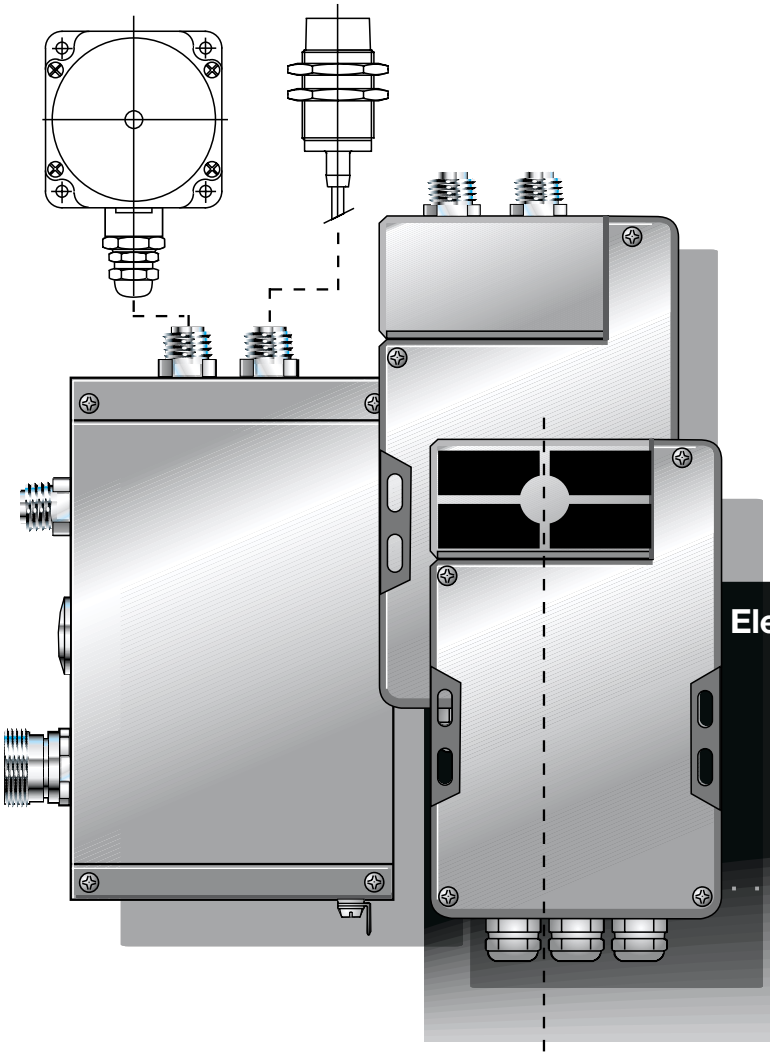
BALLUFF



Handbuch

Elektronische Identifikations-Systeme BIS
Server für Windows "BIS022SV.EXE"
Auswerteeinheit BIS C-6_0-022-...

English – please turn over!

Nr. 815 558 D/E • Ausgabe 0101
Änderungen vorbehalten.
Ersetzt Ausgabe 9903

Schreibweise:

Zu sendende Steuerzeichen sind in spitze Klammern gesetzt.

Im ASCII-Code zu übertragende Zeichen sind in Hochkomma gesetzt.

Beispiel: <STX> '1 2 3 4 5 6' BCC

http://www.balluff.de

Balluff GmbH
Schurwaldstraße 9
73765 Neuhausen a.d.F.
Deutschland
Telefon +49 (0) 71 58/1 73-0
Telefax +49 (0) 71 58/50 10
E-Mail: balluff@balluff.de

Inhaltsverzeichnis

Sicherheitshinweise

Bestimmungs-gemäßer Betrieb

Auswerteeinheiten BIS C-600 und BIS C-620 bilden zusammen mit den anderen Bausteinen des Systems BIS C das Identifikations-System und dürfen nur für diese Aufgabe im industriellen Bereich entsprechend Klasse A des EMV-Gesetzes eingesetzt werden.

Installation und Betrieb

Installation und Betrieb sind nur durch geschultes Fachpersonal zulässig. Unbefugte Eingriffe und unsachgemäße Verwendung führen zum Verlust von Garantie- und Haftungsansprüchen.

Bei der Installation der Auswerteeinheit sind die Kapitel mit den Anschlußplänen genau zu beachten. Besondere Sorgfalt erfordert der Anschluß der Auswerteeinheit an externe Steuerungen, speziell bezüglich Auswahl und Polung der Verbindungen und der Stromversorgung.

Für die Stromversorgung der Auswerteeinheit dürfen nur zugelassene Stromversorgungen benutzt werden. Einzelheiten enthält das Kapitel Technische Daten.

Einsatz und Prüfung

Für den Einsatz des Identifikations-Systems sind die einschlägigen Sicherheitsvorschriften zu beachten. Insbesondere müssen Maßnahmen getroffen werden, daß bei einem Defekt des Identifikations-Systems keine Gefahren für Personen und Sachen entstehen können.

Hierzu gehören die Einhaltung der zulässigen Umgebungsbedingungen und die regelmäßige Überprüfung der Funktionsfähigkeit des Identifikations-Systems mit allen damit verbundenen Komponenten.

Funktionsstörungen

Wenn Anzeichen erkennbar sind, daß das Identifikations-System nicht ordnungsgemäß arbeitet, ist es außer Betrieb zu nehmen und gegen unbefugte Benutzung zu sichern.

Gültigkeit

Diese Beschreibung gilt für die Auswerteeinheiten der Baureihe BIS C-600-022-...-00-KL1, BIS C-600-022-...-01-KL1 und BIS C-620-022-050-00-ST2, BIS C-620-022-050-01-ST2 in Verbindung mit der Server-Software BIS022SV.EXE mit DDEML-Schnittstelle unter Windows 3.1, Windows95/98 oder Windows NT.

Einführung
Identifikations-System BIS C

Dieses Handbuch soll den Anwender beim Erstellen des Steuerprogramms, der Installation und der Inbetriebnahme der Komponenten des Identifikations-Systems BIS C-6_0 anleiten, so daß sich ein sofortiger, reibungsloser Betrieb anschließt.

Prinzip

Das Identifikations-System BIS C-6_0 gehört zur Kategorie der

berührungslos arbeitenden Systeme, die sowohl lesen als auch schreiben können.

Diese Doppelfunktion ermöglicht Einsätze, bei denen nicht nur fest in den Codeträger programmierte Informationen transportiert, sondern auch aktuelle Informationen gesammelt und weitergegeben werden.

Einsatzgebiete

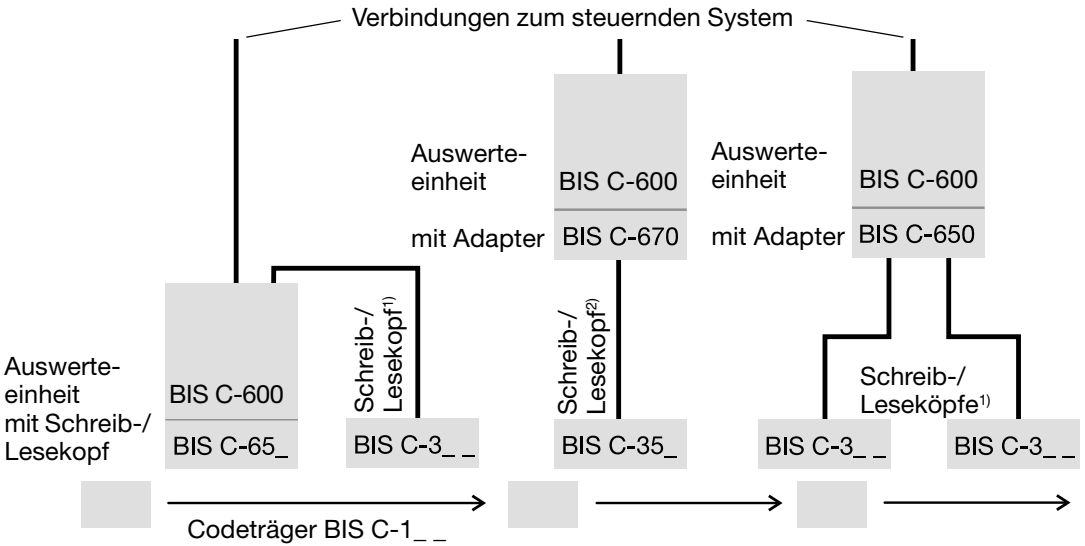
Einige der wesentlichen Einsatzgebiete finden sich

- **in der Produktion zur Steuerung des Materialflusses**
  (z. B. bei variantenspezifischen Prozessen)**,**
  beim Werkstücktransport mit Förderanlagen,
  zur Datengewinnung für die Qualitätssicherung,
  zur Erfassung sicherheitsrelevanter Daten,
- **in der Werkzeugcodierung und -überwachung;**
- **in der Betriebsmittelorganisation;**
- **im Lagerbereich zur Kontrolle der Lagerbewegungen und -bestände;**
- **im Transportwesen und in der Fördertechnik;**
- **in der Entsorgung zur mengenabhängigen Erfassung.**

System-komponenten

Die Hauptbestandteile des Identifikationssystems BIS C-600 sind

- **Auswerteeinheit,**
- **Schreib-/Leseköpfe und**
- **Codeträger.**



Schematische Darstellung eines Identifikations-Systems (Beispiel)

[1]) ausgenommen BIS C-350 und -352

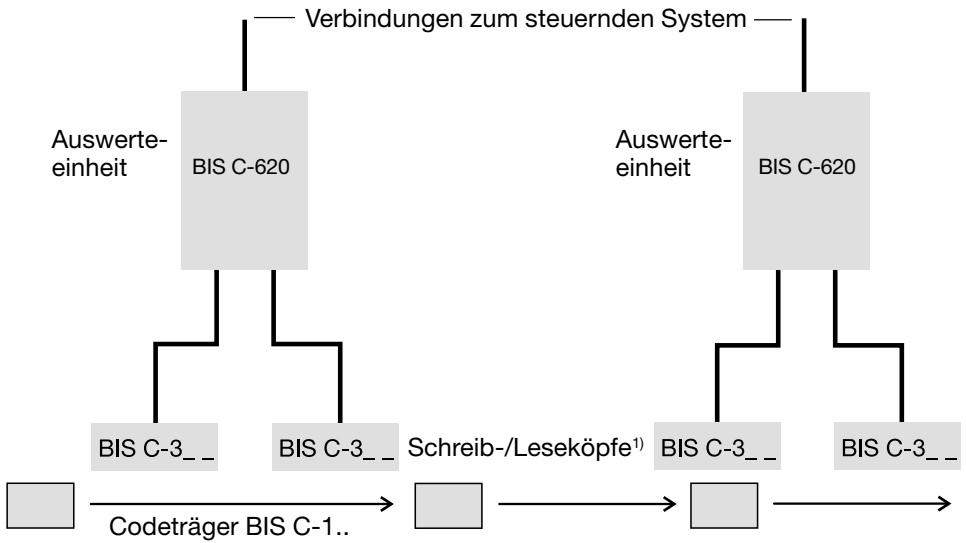
[2]) nur BIS C-350 oder -352

Einführung
Identifikations-System BIS C-620

System-komponenten

Die Hauptbestandteile des Identifikationssystems BIS C-620 sind

- **Auswerteeinheit,**
- **Schreib-/Leseköpfe und**
- **Codeträger**.



Schematische Darstellung eines Identifikations-Systems (Beispiel)

1) ausgenommen BIS C-350 und -352

Auswahl der Systemkomponenten

Die Auswerteeinheit **BIS C-600** besitzt ein Kunststoffgehäuse. Der Anschluß erfolgt über eine Klemmleiste, wobei die Kabel mittels PG-Verschraubungen gesichert werden. An die Auswerteeinheit BIS C-600 kann ein einzelner Schreib-/Lesekopf der Baureihe BIS C-65_ direkt montiert werden, wodurch eine kompakte Einheit entsteht. Über den Adapter BIS C-650, der anstatt des Schreib-/Lesekopfes BIS C-65_ montiert wird, können alternativ zwei Schreib-/Leseköpfe abgesetzt über Kabel angeschlossen werden. Über den Adapter BIS C-670 kann ein Schreib-/Lesekopf abgesetzt über Kabel angeschlossen werden.

Die Auswerteeinheit **BIS C-620** besitzt ein Metallgehäuse. Der Anschluß der Kabel erfolgt über Rundsteckverbinder. An die Auswerteeinheit BIS C-620 können zwei Schreib-/Leseköpfe abgesetzt über Kabel angeschlossen werden.

Weitere Informationen zu den Schreib-/Leseköpfen der Baureihe BIS C-65_ bzw. der Baureihe BIS C3_ _ mit sämtlichen Kombinationen der passenden Codeträger finden Sie in den zugehörigen Handbüchern der Schreib-/Leseköpfe.

Ob die Kompaktlösung der Auswerteeinheit mit integriertem Schreib-/Lesekopf oder die abgesetzte Lösung sinnvoll ist, richtet sich im wesentlichen nach der räumlichen Anordnung der Bausteine. Funktionale Einschränkungen sind nicht gegeben. Alle Schreib-/Leseköpfe sind für statisches und dynamisches Lesen geeignet. Abstand und Relativgeschwindigkeit richten sich nach der Wahl des Codeträgers. In den jeweiligen Handbüchern zu den Schreib-/Leseköpfen der Baureihe BIS C-65_ sowie der Baureihe BIS C-3_ _ finden Sie sämtliche Kombinationen von Schreib-/Lesekopf und passenden Codeträgern.

Die Systemkomponenten werden von der Auswerteeinheit elektrisch versorgt. Der Codeträger stellt eine eigenständige Einheit dar, benötigt also keine leitungsgebundene Stromzuführung. Er bekommt seine Energie vom Schreib-/Lesekopf. Dieser sendet ständig ein Trägersignal aus, das den Codeträger versorgt, sobald der notwendige Abstand erreicht ist. In dieser Phase findet der Schreib-/Lesevorgang statt. Dieser kann statisch oder dynamisch erfolgen.

Dialogmodus

Über den Schreib-/Lesekopf schreibt die Auswerteeinheit Daten vom steuernden System auf den Codeträger oder liest sie vom Codeträger und stellt sie dem steuernden System zur Verfügung. Steuernde Systeme können sein:

– **ein Steuerrechner (z.B. Industrie-PC) oder**
– **eine speicherprogrammierbare Steuerung (SPS)**

Anwendung mit der Server-Software 022

Bei der Version BIS C-6_0-022 ermöglicht die Server-Software BIS022SV.EXE die Anbindung an jede Steuerungssoftware (Client unter Windows 3.1, Windows95 oder Windows NT), indem er die Client-Einstellungen über die DDE-Schnittstelle (DDEML) an das Identifikations-System anpaßt. Der Server übernimmt die Umsetzung des Protokollablaufs, der über eine Client-Server-Schnittstelle abgewickelt wird (siehe Kapitel 'Anwendung mit Server-Software 022' ab Siete 30). Der Anwender arbeitet ausschließlich mit der Client-Software.

Der Server BIS022SV.EXE ist für den Einsatz mit der SINUMERIK 840D von Siemens zugelassen. Die manuelle Eingabe der Werkzeugdaten wird durch das automatische Lesen und Beschreiben des am Wekzeug angebrachten Codeträgers ersetzt. Damit ist die Kopplung und Erweiterung des Be- und Entladedialogs einer Maschinensteurung mit einem Werkzeug-Identifikationssystem geschaffen.

Anwendung mit dem Standard-Protokoll 007

Die Auswerteeinheit steuert und verwaltet die Datenkommunikation zwischen Codeträgern und Schreib-/Leseköpfen. Über die serielle Schnittstelle verbindet sie das Identifikations-System BIS C-6_0 mit einer externen Steuereinrichtung.

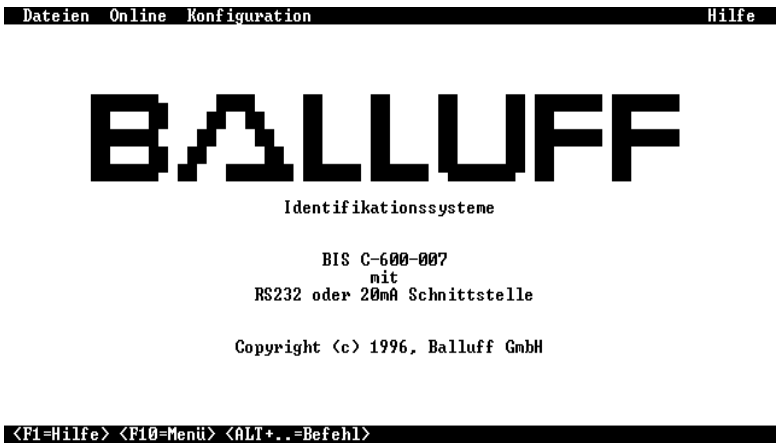
Zur Auswahl stehen entweder die Schnittstelle RS 232 (V.24) oder die 20-mA-Stromschnittstelle (TTY).

Der Datenverkehr zwischen der Auswerteeinheit und dem steuernden System geschieht über festgelegte Telegramme. Der Protokollablauf wird in Form von Funktionsblöcken dargestellt. Im Kapitel 'Anwendung mit dem Standard-Protokoll 007' ab Seite 46 werden der Protokollablauf gezeigt und die Telegramminhalte präzisiert.

Konfiguration

Vor Beginn ist die Konfiguration der Auswerteeinheit durchzuführen, falls nicht mit der Werkseinstellung gearbeitet werden soll.

Die Konfiguration wird mittels Computer und der Balluff-Software BISC600A.EXE vorgenommen und in der Auswerteeinheit gespeichert. Sie kann jederzeit überschrieben werden. Die Konfiguration kann in einer Datei gespeichert werden und ist so jederzeit wieder verfügbar.



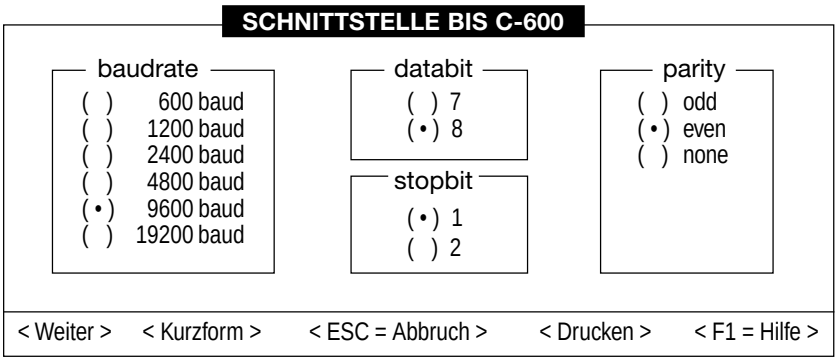
Wichtig. **Bitte beachten.**
Bitte dokumentieren Sie die gewählten Einstellungen auf den mitgelieferten Aufklebern (auf die Innenseite des Gerätedeckels kleben) sowie auf den Seiten 26 und 28 in der Kundenkonfiguration. Damit kann einerseits bei einer Reparatur der Auswerteeinheit Ihre Konfiguration gesichert werden und andererseits für weitere Einheiten benutzt werden.

Menü Schnittstelle BIS C-6_0

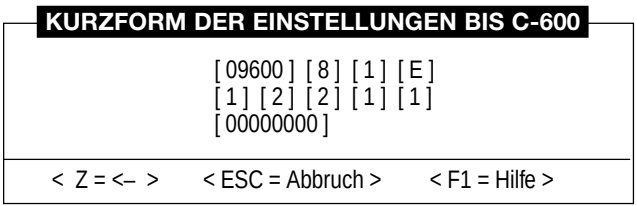
In der ersten Maske werden die Parameter Übertragungsrate, Anzahl der Daten- und Stopbits sowie die Parity-Art für die serielle Schnittstelle eingestellt.

Die Abbildung zeigt die Werkseinstellung.

Diese Einstellung ist auch zwingend für die Client-Server-Anwendung!



Wenn die Initialisierungsdaten in Kurzform vorliegen (z.B. auf dem Gehäusedeckel nach einem Gerätetausch), dann kann die Eingabe direkt in die Maske "Kurzform der Einstellungen BIS C-600" eingetragen werden (siehe auch Kundenkonfiguration auf Seite 26/27).



Die weiteren Einstellungen werden in den Masken vorgenommen, die auf den folgenden Seiten abgebildet sind.

Menü Einstellungen BIS C-6_0

EINSTELLUNGEN BIS C-600

Parameter

[ ] CT-Present auf RTS
[ ] CT-Daten sofort senden
[ ] Dynamik-Betrieb
[ ] CT-Present auf Ausgang 1
[ ] Ausgänge bei CT-Present bearb.
[ ] Schnelle Codeträgererkennung
[ ] BIS C-1../02B [x] = ja
[ ] CRC_16 Datenprüfung

Protokolltyp

(•) BCC
( ) CR als Endekennung
( ) CR
( ) LFCR als Endekennung

Seitengröße

(•) 32 Byte ( ) 64 Byte

Eingang

(•) Reset
( ) Kopfanwahl
( ) Datenbit auf Codeträger
( ) Keine Funktion

< Z = <– > < ESC = Abbruch > < Daten an BIS > < Speichern > < F1 = Hilfe >

Menü Einstellungen BIS C-6_0, Feld Protokolltyp

Feld Protokolltyp:

Werkseitig ist auf Betrieb mit Blockcheck BCC eingestellt. Für Steuergeräte, die ein Endekennungszeichen benötigen, kann die zusätzliche Verwendung von Carriage Return 'CR' oder Linefeed mit Carriage Return 'LF CR' eingerichtet werden. Auf der folgenden Seite finden Sie Beispiele für die verschiedenen Möglichkeiten.

Client-Server-Anwendung:
- **Die Einstellung des Protokolltyps BCC ist zwingend.**
- **'CT-Daten sofort senden' ist nicht zulässig!**
- **'CRC_16 Datenprüfung' ist nicht zulässig.**

Menü Einstellungen BIS C-6_0, Feld Protokolltyp (Fortsetzung)

Beispiele für den Abschluß der Telegramme:

Protokollvarianten	Telegramm mit Befehl, Adresse und Anzahl Bytes	Abschluß	Quittung	Ende-kennung
mit Blockcheck BCC	'R 0000 0001'	BCC	<ACK> '0'	
mit Carriage Return	'R 0000 0001'	'CR'	<ACK> '0'	
mit Endekennung Carriage Return	'R 0000 0001'	'CR'	<ACK> '0'	'CR'
mit Endekennung Carriage Return und Line Feed	'R 0000 0001'	'LF CR'	<ACK> '0'	'LF CR'

Menü Einstellungen BIS C-6_0, Feld Parameter

Feld Parameter:

– **CT-Present auf RTS**
Der Parameter CT-Present auf RTS (entspricht der LED-Anzeige Codetag Present am Gehäuse des BIS C-6_0) kann dazu benutzt werden, um CT-Present als Hardwaresignal von einem PC abfragen zu können. Sie können die RTS-Leitung z.B. auf den freien Eingang RI (Ring Indicator) am PC legen, da dieser nur bei Modem-Betrieb verwendet wird.

– **CT-Daten sofort senden**
Bei jedem Neuerkennen eines Codeträgers wird dieser je nach Konfiguration ausgelesen und die Daten werden an die Schnittstelle ausgegeben. Mit dieser Einstellung erübrigt sich der Lesebefehl im Dialogmodus.

Client-Server-Anwendung:
– **Die Einstellung des Protokolltyps BCC ist zwingend.**
– **'CT-Daten sofort senden' ist nicht zulässig!**

Menü Einstellungen BIS C-6_0, Feld Parameter (Fortsetzung)

– **Dynamikbetrieb**
Diese Funktion schaltet die Fehlermeldung "Kein Codeträger vorhanden" aus, d.h.:
-> Im Dynamikbetrieb wird ein Lese- oder Schreibtelegramm so lange gespeichert, bis ein Codeträger in den Arbeitsbereich des betreffenden Schreib-/Lesekopfs kommt.
-> Ohne Dynamikbetrieb wird ein Lese- oder Schreibbefehl mit der Fehlermeldung <NAK> '1' abgelehnt, wenn sich kein Codeträger im Arbeitsbereich des Schreib-/Lesekopfs befindet; die Auswerteeinheit geht in den Ruhezustand.

– **CT-Present auf Ausgang 1**
Ist CT Present auf Ausgang 1 gewählt, wird die LED-Meldung Codetag Present auf den Ausgang 1 gegeben. Man kann so das Vorhandensein eines Codeträgers direkt als Hardwaresignal digital weiterverarbeiten.

– **Ausgänge bei CT-Present bearbeiten**
Die Ausgangsfunktionen werden normalerweise nur nach einem Lesebefehl bearbeitet. Da jedoch die Codeträgererkennung ebenfalls ein Lesen des Codeträgers ist (Lesen der ersten Codeträgerseite, je nach Typ 32 oder 64 Byte), kann auch bereits mit dem Codetag Present die Ausgangsbearbeitung stattfinden. Liegen die zur Ausgangsbearbeitung verwendeten Adressen innerhalb dieser ersten Seite, kann die Auswerteeinheit ohne Befehl über die serielle Schnittstelle kleine Steuerbefehle ausführen.
-> Für sehr schnelle Vorgänge siehe auch unter Schnelle Codeträgererkennung.

– **Schnelle Codeträgererkennung**
Für sehr schnelle Vorgänge kann die Anzahl der verwendeten Codeträgeradressen für die Codeträgererkennung von 32 bzw. 64 Byte auf 4 Byte reduziert werden. Die Codeträgererkennung verkürzt sich dadurch auf ca. 50 ms (anstatt ca. 150 ms bei Codeträgern < 2 kByte bzw. ca. 250 ms bei Codeträgern ≥ 2 kByte).
-> Beachten Sie dies bei Verwendung von Parameter "Ausgänge bei CT-Present bearbeiten".

– **BIS C-1../02B [x] = ja**
Der Parameter ist einzuschalten, wenn ein Codeträger vom Typ BIS C-1../02B verwendet wird.

– **CRC_16 Datenprüfung:** Bei Client-Server-Betrieb nicht zulässig.

Menü Einstellungen BIS C-6_0, Feld Seitengröße

Die Codeträger sind je nach Speichergröße unterschiedlich organisiert. Die Auswerteeinheit benötigt den Typ zur korrekten Bearbeitung. Werkseinstellung: 32 Byte. Es gilt:

Codeträger mit 32 Byte Seitengröße	BIS C-1_ _-02/_
	BIS C-1_ _-03/_
	BIS C-1_ _-04/_
	BIS C-1_ _-05/_

Codeträger mit 64 Byte Seitengröße	BIS C-1_ _-10/_
	BIS C-1_ _-11/_
	BIS C-1_ _-30/_

Menü Einstellungen BIS C-6_0, Feld Eingang

Funktion für den digitalen Steuereingang des BIS C-6_0.

– **Reset** (Werkseinstellung)
Ist Reset ausgewählt, bewirkt ein High-Signal an diesem Eingang einen Reset der Auswerteeinheit BIS C-600 aus. Alle anstehenden Befehle werden gelöscht.

– **Kopfanwahl**
Ist Kopfanwahl ausgewählt, erfolgt die Kopfanwahl über diesen Eingang.

Eingang auf Low: Kopf 1 angewählt.
Eingang auf High: Kopf 2 angewählt.

-> Diese Funktion hat stets Vorrang. So ist z.B. die Funktion "Beide Schreib-/Leseköpfe aktiv" deaktiviert, die über den Befehl 'HT' angewählt wird.

– **Datenbit auf Codeträger**
Mit dem Erkennen eines neuen Codeträgers wird ein frei definiertes Bit einer anzugebenden Adresse direkt oder invertiert auf den Codeträger geschrieben. Nach erfolgreichem Schreiben wird der ebenfalls zu definierende Ausgang so lange gesetzt, bis der Codeträger den aktiven Schreib-/Lesebereich verläßt.
-> Der Parameter Dynamik-Betrieb wird automatisch zurückgesetzt.

– **Keine Funktion**
Der Eingang wird nicht bearbeitet.

Menü Eingang/ Ausgänge zuweisen (nur BIS C-600)

Den Ausgängen können verschiedene Funktionen zugewiesen werden. Die Ausgangsfunktionen werden immer beim Lesen bearbeitet. Bedingung für die Ausführung ist, daß die jeweilige Adresse beim vorangegangenen Leseauftrag gelesen wurde.

Die Abbildung zeigt die Werkseinstellung.

Diese Einstellung ist zwingend für die Client-Server-Anwendung oder bei BIS C-620, d.h. Ausgänge dürfen nicht zugewiesen werden!

EINGANG/AUSGÄNGE ZUWEISEN

(•) Keine Verwendung der Ausgänge.
( ) Halbbyte des Dateninhalts einer Adresse ausgeben
( ) Inhalt mehrerer Adressen mit einem Festwert vergleichen.
( ) Inhalt einer Adresse mit unterschiedlichen Festwerten vergleichen.
( ) Inhalt mehrerer Adressen mit dem Inhalt einer Adresse vergleichen.
( ) Datenbits variabler Adressen ausgeben.

( ) Eingang als Datenbit auf Codeträger programmieren.
( ) Codeträgerdaten ohne direkten Befehl lesen und senden.

< OK > < Kurzform > < ESC = Abbruch > < Drucken . . . . > < F1 = Hilfe >

Bei "Keine Verwendung der Ausgänge" ist die Bearbeitung der Ausgänge deaktiviert.

KEINE AUSGäNGE ZUWEISEN

<Daten an BIS> <Speichern> <ESC = Abbruch> <F1 = Hilfe>

"Daten an BIS" überträgt die Daten an die Auswerteeinheit. "Speichern" legt die Daten in der Konfigurationsdatei auf Ihrem Computer ab.

Menü Eingang/ Ausgänge zuweisen (nur BIS C-600) (Fortsetzung)

Halbbyte des Dateninhalts einer Adresse ausgeben:

EINGANG/AUSGÄNGE ZUWEISEN

( ) Keine Verwendung der Ausgänge.
(•) Halbbyte des Dateninhalts einer Adresse ausgeben
( ) Inhalt mehrerer Adressen mit einem Festwert vergleichen.
( ) Inhalt einer Adresse mit unterschiedlichen Festwerten vergleichen.
( ) Inhalt mehrerer Adressen mit dem Inhalt einer Adresse vergleichen.
( ) Datenbits variabler Adressen ausgeben.

( ) Eingang als Datenbit auf Codeträger programmieren.
( ) Codeträgerdaten ohne direkten Befehl lesen und senden.

< OK > < ESC = Abbruch > < Drucken . . . . > < F1 = Hilfe >

HALBBYTE AUSGABE

Codeträgeradresse [9999]

( ) High Nibble ausgeben ?
(•) Low Nibble ausgeben ?

< Daten an BIS > < Speichern > < ESC = Abbruch > < F1 = Hilfe >

Der 8 Bit umfassende Dateninhalt einer Adresse wird in Nibble zu jeweils 4 Bit an den 4 Ausgängen ausgegeben (Bit 0 an Ausgang 1, Bit 1 an Ausgang 2 usw.). Je nach Einstellung sind es entweder die oberen (High Nibble) oder die unteren (Low Nibble). Die Adresse wird dezimal vorgegeben.

Menü Eingang/ Ausgänge zuweisen (nur BIS C-600)
(Fortsetzung)

Inhalt mehrerer Adressen mit einem Festwert vergleichen:
Die Dateninhalte von bis zu 4 dezimal angegebenen Adressen werden mit einem dezimalen Festwert verglichen. Zu jeder Adresse kann angegeben werden, welcher der Ausgänge 1 bis 4 bei einem positiven Ergebnis des Vergleichs gesetzt oder gelöscht werden soll und ob der Ausgang bei einem negativen Ergebnis des Vergleichs nicht verändert oder im Gegensatz zur Definition bei positivem Vergleich geschaltet werden soll (inverses Verhalten). Es wird jede Adresse bearbeitet, die sich innerhalb des Lesebefehls befindet.

Adressen mit einem Festwert vergleichen

Festwert: [000]

Adresse	Ausgang	Vergleich positiv	Vergleich negativ
[9999]	[1]	(•) setzen ( ) löschen	( ) nicht verändern (•) invertieren
[9999]	[2]	(•) setzen ( ) löschen	( ) nicht verändern (•) invertieren
[9999]	[3]	(•) setzen ( ) löschen	( ) nicht verändern (•) invertieren
[9999]	[4]	(•) setzen ( ) löschen	( ) nicht verändern (•) invertieren

< Daten an BIS > < Speichern > < ESC = Abbruch > < F1 = Hilfe >

Ist in der Initialisierung der Parameter "Ausgänge bei CT-Present bearbeiten" angegeben, dann wird diese Funktion auch bei einer Neuerkennung eines Codeträgers ausgeführt (eine oder mehrere der angegebenen Adressen sollten sich dann in der ersten Codeträgerseite befinden).

Menü Eingang/ Ausgänge zuweisen (nur BIS C-600)
(Fortsetzung)

Inhalt einer Adresse mit unterschiedlichen Festwerten vergleichen:

Der Dateninhalt einer dezimal angegebenen Adresse wird mit 4 Dezimal-Festwerten verglichen. Zu jedem Festwert kann angegeben werden, welcher der Ausgänge 1 bis 4 bei einem positiven Ergebnis des Vergleichs gesetzt oder gelöscht und ob der Ausgang bei einem negativen Ergebnis des Vergleichs nicht verändert oder im Gegensatz zur Definition bei positivem Vergleich geschaltet werden soll (inverses Verhalten).

Adressen mit unterschiedlichen Festwerten vergleichen

Adresse: [9999]

Festwert	Ausgang	Vergleich positiv	Vergleich negativ
[000]	[1]	(•) setzen ( ) löschen	( ) nicht verändern (•) invertieren
[000]	[2]	(•) setzen ( ) löschen	( ) nicht verändern (•) invertieren
[000]	[3]	(•) setzen ( ) löschen	( ) nicht verändern (•) invertieren
[000]	[4]	(•) setzen ( ) löschen	( ) nicht verändern (•) invertieren

< Daten an BIS > < Speichern > < ESC = Abbruch > < F1 = Hilfe >

Ist in der Initialisierung der Parameter "Ausgänge bei CT-Present bearbeiten" angegeben, dann wird diese Funktion auch bei einer Neuerkennung eines Codeträgers ausgeführt (eine oder mehrere der angegebenen Adressen sollten sich dann in der ersten Codeträgerseite befinden).

Menü Eingang/ Ausgänge zuweisen (nur BIS C-600)
(Fortsetzung)

Inhalt mehrerer Adressen mit mit dem Inhalt einer Adresse vergleichen:

Die Dateninhalte von bis zu 4 dezimal angegebenen Adressen werden mit dem Dateninhalt einer weiteren Adresse verglichen. Zu jeder Adresse kann angegeben werden, welcher der Ausgänge 1 bis 4 bei einem positiven Ergebnis des Vergleichs gesetzt oder rückgesetzt werden soll und ob der Ausgang bei einem negativen Ergebnis des Vergleichs nicht verändert oder im Gegensatz zur Definition bei positivem Vergleich geschaltet werden soll. Es wird jede Adresse bearbeitet, die sich innerhalb des Lesebefehls befindet, vorausgestzt, die Vergleichsadresse befindet sich innerhalb des Bereichs.

Inhalt mehrerer Adressen mit Inhalt anderer Adresse vergleichen

Adresse: [9999]

Adresse	Ausgang	Vergleich positiv	Vergleich negativ
[9999]	[1]	(•) setzen ( ) löschen	( ) nicht verändern (•) invertieren
[9999]	[2]	(•) setzen ( ) löschen	( ) nicht verändern (•) invertieren
[9999]	[3]	(•) setzen ( ) löschen	( ) nicht verändern (•) invertieren
[9999]	[4]	(•) setzen ( ) löschen	( ) nicht verändern (•) invertieren

< Daten an BIS > < Speichern > < ESC = Abbruch > < F1 = Hilfe >

Ist in der Initialisierung der Parameter "Ausgänge bei CT-Present bearbeiten" angegeben, dann wird diese Funktion auch bei einer Neuerkennung eines Codeträgers ausgeführt (eine oder mehrere der angegebenen Adressen sollten sich dann in der ersten Codeträgerseite befinden).

Menü Eingang/ Ausgänge zuweisen (nur BIS C-600)
(Fortsetzung)

Datenbits variabler Adressen ausgeben:

Jeweils 1 Datenbit von bis zu 4 dezimal angegebenen Adressen kann an einen der 4 Ausgänge ausgegeben und dabei invertiert oder nicht invertiert werden.

Datenbits variabler Adressen ausgeben

Adresse	Bit-Nummer	Ausgang	invertieren
[9999]	[1]	[1]	[ ] ja
[9999]	[2]	[1]	[ ] ja
[9999]	[3]	[1]	[ ] ja
[9999]	[4]	[1]	[ ] ja

< Daten an BIS > < Speichern > < ESC = Abbruch > < F1 = Hilfe >

Ist in der Initialisierung der Parameter "Ausgänge bei CT-Present bearbeiten" angegeben, dann wird diese Funktion auch bei einer Neuerkennung eines Codeträgers ausgeführt (eine oder mehrere der angegebenen Adressen sollten sich dann in der ersten Codeträgerseite befinden).

Menü Eingang/ Ausgänge zuweisen (Fortsetzung)

Eingang als Datenbit auf Codeträger programmieren:

Beim Erkennen eines neuen Codeträgers wird der Zustand des digitalen Eingangs direkt oder invertiert als Bit auf den Codeträger geschrieben.

Zulässiger Adreßbereich: 0...31! Bitnummer der Adresse: 1...8.

Die Ausgänge für das Quittungs- und das Freigabesignal sind ebenfalls anzugeben. Bei Freigabeausgang = 0 wird das Freigabesignal nicht verwendet. Nachfolgend wird der Ablauf beschrieben.

Eingang als Datenbit auf Codeträger programmieren

Adresse	[00]	[ ] Eingang invertieren ?
Bitnummer	[1]	
Quittungsausgang	[1]	
Freigabeausgang	[0]	

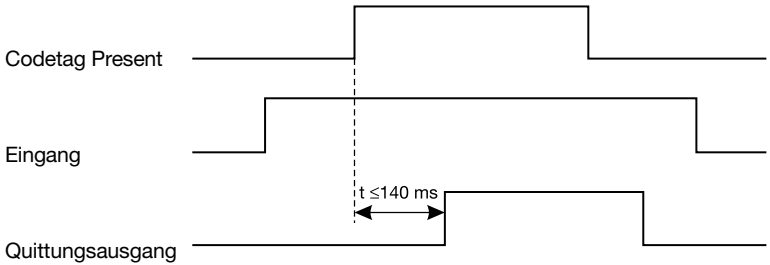
< Daten an BIS > < Speichern > < ESC = Abbruch > < F1 = Hilfe >

Ablauf ohne Freigabesignal

Mit dem Erkennen eines neuen Codeträgers wird das definierte Bit der angegebenen Adresse direkt oder invertiert programmiert. Nach erfolgreichem Schreiben wird der angegebene Quittungsausgang so lange gesetzt, bis der Codeträger den aktiven Schreib-/Lesebereich verläßt. Der Eingang muß so lange seinen Zustand beibehalten, bis der Quittungsausgang gesetzt ist.

Der Eingangszustand, der als Information auf den Codeträger geschrieben werden soll, muß bereits vor dem Erkennen des Codeträgers anstehen.

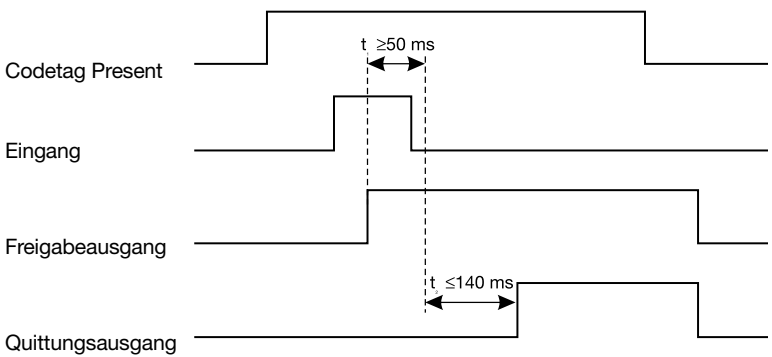
Menü Eingang/ Ausgänge zuweisen (Fortsetzung)



In diesem Ablaufbeispiel wird das Bit auf 1 gesetzt.

Ablauf mit Freigabesignal

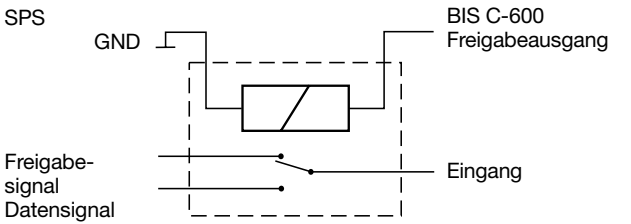
Nach dem Erkennen eines Codeträgers wird der Eingang so lange abgefragt, bis er gesetzt ist (Freigabe erteilt). Die Auswerteeinheit setzt den Freigabeausgang und wartet 50 ms. Danach wird der Zustand des Eingangs abgefragt und als Datenwert übernommen. Dieser wird je nach Vorgabe direkt oder invertiert auf den Codeträger geschrieben. Nach dem Schreiben wird der Quittungsausgang so lange gesetzt, bis der Codeträger den Schreib-/Lesebereich verlassen hat. Jetzt schaltet der Freigabeausgang wieder auf Low.



In diesem Beispiel wird das Bit auf 0 rückgesetzt.

Menü Eingang/ Ausgänge zuweisen (Fortsetzung)

Den Freigabeausgang kann man zum Schalten eines Relais verwenden, um damit das Freigabesignal und das Datensignal auf den Eingang zu schalten.



Als Eingangssignal in eine SPS kann dieser Ausgang anzeigen, daß als nächstes das Datensignal auf den Eingang des BIS C-600 geschaltet werden muß. Erforderlich ist dieser Ausgang überall, wo der Codeträger in den Bereich des Schreib-/Lesekopfs kommen kann, bevor das Datensignal am Eingang der Auswerteeinheit steht.

Diese Funktion muß zusätzlich in der Initialisierung freigegeben werden.

Menü Eingang/ Ausgänge zuweisen (Fortsetzung)

Codeträgerdaten ohne direkten Befehl lesen und senden:

Die vorgegebene Datenmenge (Anzahl Byte ab Startadresse) wird vom neu erkannten Codeträger ausgelesen.

Nach dem Lesen werden die Daten automatisch an die Schnittstelle gesendet.

Wahlweise können zusätzlich als Abschluß ein BBC und/oder 1 bzw. 2 frei definierbare Abschlußzeichen gesendet werden.

Bei Client-Server-Anwendung nicht zulässig!

Daten nach Codeträgererkennung ausgeben

Datenmenge

Startadresse:	[0000] Dezimal
Anzahl Byte:	[0000] Dezimal

Endekennung

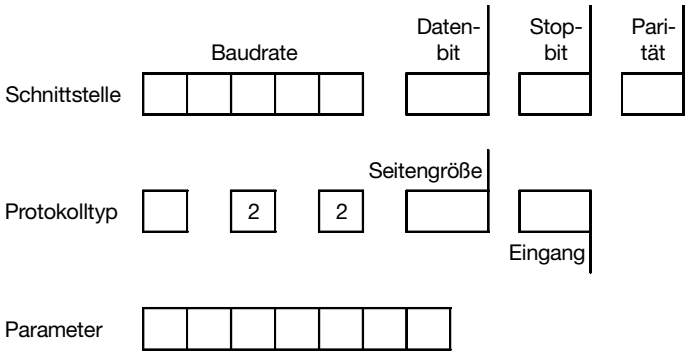
BCC	[ ] ja			
1. Abschlußzeichen:	[ ] ja	Wert:	[000]	Dez.
2. Abschlußzeichen:	[ ] ja	Wert:	[000]	Dez.

< Daten an BIS > < Speichern > < ESC = Abbruch > < F1 = Hilfe >

Kundenkonfiguration

Initialisierung

Bitte tragen Sie die Einstellungen in die Felder auf der Innenseite des Gerätedeckels ein, damit die Einstellungen bei einer Reparatur vom Werk wieder eingegeben werden können. Tragen Sie sie auch in die nachstehenden Felder ein, damit Sie identische Einstellungen z.B. bei weiteren Geräten vornehmen können.



Auf der folgenden Seite finden Sie ein Beispiel wiedergegeben, wie Sie es sich nach der Initialisierung ausdrucken lassen können. Übertragen Sie die Einstellungen in die obigen Felder, damit Sie die Werte jederzeit zur Hand haben und reproduzieren können. Sie können diese Daten dann in Kurzform in die Maske eingeben (siehe auch Seite 11).

KURZFORM DER EINSTELLUNGEN BIS C-600

[19200] [8] [1] [N]
[4] [2] [2] [2] [1]
[01101100]

< Z = <– > < ESC = Abbruch > < F1 = Hilfe >

Initialisierung
(Fortsetzung)

Beispiel eines Ausdrucks, wie Sie ihn durch das Programm BISC600A.EXE nach der Initialisierung anfertigen können.

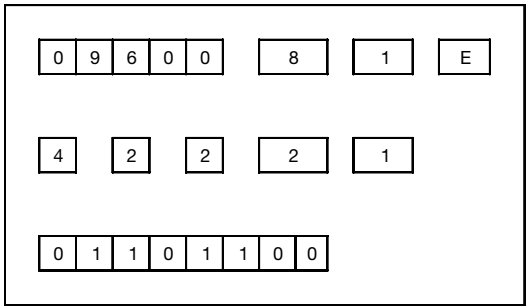
Schnittstelleneinstellung
Übertragungsrate : 9600 baud
Datenbit : 8
Stopbit : 1
Parität : Even

Protokolltyp
( ) BCC
( ) CR als Endekennung
( ) CR
(*) LF CR als Endekennung

Parameter
[ ] Codetag-Present auf RTS-Leitung ausgeben.
[X] CT-Daten sofort senden
[X] Dynamik-Betrieb
[ ] Codetag-Present auf Ausgang 1
[X] Ausgänge bei Codetag-Present bearbeiten
[X] Schnelle Codeträgererkennung
[ ] BIS C-1../02B [X] = ja
[ ] CRC_16 Datenprüfung

Seitengröße
( ) 32 Byte Seitengröße
(*) 64 Byte Seitengröße

Eingang
(*) Eingang = RESET
( ) Eingang = Kopfanwahl
( ) Eingang = Datenbitinformation auf Codeträger
( ) Eingang = Keine Funktion

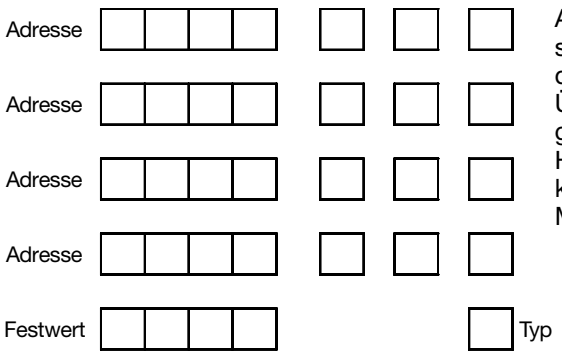


Die Eintragung in die Felder erfolgt entweder als Klartext (wie bei Schnittstelle) oder es wird die Nummer der markierten Zeile eingetragen. Bei Parameter wird die markierte Position durch eine 1 gekennzeichnet.

Kundenkonfiguration

Ein-/Ausgangs-konfiguration

Bitte tragen Sie die Einstellungen in die Felder auf der Innenseite des Gerätedeckels ein, damit die Einstellungen bei einer Reparatur vom Werk wieder eingegeben werden können. Tragen Sie sie auch in die nachstehenden Felder ein, damit Sie identische Einstellungen z.B. bei weiteren Geräten vornehmen können.



Auf der folgenden Seite finden Sie ein Beispiel wiedergegeben, wie Sie es sich nach der Initialisierung ausdrucken lassen können. Übertragen Sie die Einstellungen in die obigen Felder, damit Sie die Werte jederzeit zur Hand haben und reproduzieren können. Sie können diese Daten dann in Kurzform in die Maske eingeben (siehe unten).

KURZFORM DER E/A EINSTELLUNGEN BIS C-600

[0000] [0] [0] [0]
[0000] [0] [0] [0]
[0000] [0] [0] [0]
[0000] [0] [0] [0]
[0000] [1]

< Z = <– > < ESC = Abbruch > < F1 = Hilfe >

Wenn die Initialisierungsdaten in Kurzform vorliegen (z.B. auf dem Gehäusedeckel nach einem Gerätetausch), dann kann die Eingabe direkt in die Maske "Kurzform der E/A Einstellungen BIS C-600" eingetragen werden.

Kundenkonfiguration

Ein-/Ausgangs-konfiguration (Fortsetzung)

Beispiel eines Ausdrucks, wie Sie ihn durch das Programm BISC600A.EXE nach der Ein-/Ausgangskonfiguration anfertigen können.

Zuweisung

( ) Keine Verwendung der Ausgänge.
( ) Halbbyte des Dateninhalts einer Adresse ausgeben.
(•) Inhalt mehrer Adressen mit einem Festwert vergleichen.
( ) Inhalt einer Adresse mit unterschiedlichen Festwerten vergleichen.
( ) Inhalt mehrer Adressen mit dem Inhalt einer Adresse vergleichen.
( ) Datenbits variabler Adressen ausgeben.

( ) Eingang als Datenbit auf Codeträger programmieren.
( ) Codeträgerdaten ohne direkten Befehl lesen und senden.

Definition

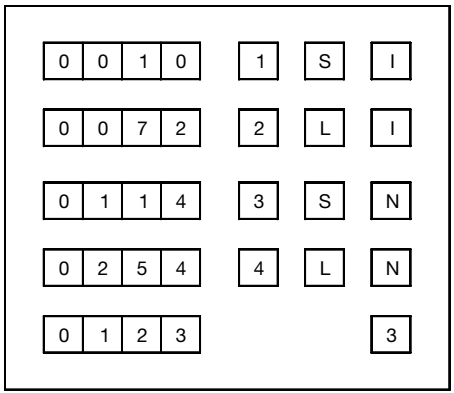
Festwert: 123

Adresse: 0010
Ausgang: 1
Vergleich positiv: (•) setzen ( ) nicht verändern
Vergleich negativ: ( ) löschen (•) invertieren

Adresse: 0072
Ausgang: 2
Vergleich positiv: ( ) setzen ( ) nicht verändern
Vergleich negativ: (•) löschen (•) invertieren

Adresse: 0114
Ausgang: 3
Vergleich positiv: (•) setzen (•) nicht verändern
Vergleich negativ: ( ) löschen ( ) invertieren

Adresse: 0254
Ausgang: 4
Vergleich positiv: ( ) setzen (•) nicht verändern
Vergleich negativ: (•) löschen ( ) invertieren



Anwendung mit Server-Software 022

Übersicht über die Funktion

Der Server stellt folgende Dienste bereit:

- Daten vom Codeträger lesen
- Daten auf den Codeträger schreiben
- Identifikations-System rücksetzen
- Initialisierungsinformation übergeben
- Schreib-/Lesevorgang abbrechen
- Fehlernummer lesen (für synchrone Bearbeitung)
- Fehlertext lesen

Synchrone und asynchrone Bearbeitung

Der Server bearbeitet jeweils einen Auftrag. Das Lesen und Beschreiben des Codeträgers ist in der Regel zeitintensiv. Deshalb bietet die Software für unterschiedliche Anwendersituationen eine Schnittstelle zur synchronen und zur asynchronen Auftragsbearbeitung.

In Anwendungen ohne zeitkritische Situationen kann die synchrone Bearbeitung gewählt werden. Sie kann nicht abgebrochen werden; der Server läßt jedoch einen Task-Wechsel im Betriebssystem zu (siehe Beispiel auf der folgenden Seite).

Um das steuernde System bei zeitkritischen Aufgaben nicht zu blockieren, muß man einen laufenden Auftrag jederzeit abbrechen können. Dies läßt die asynchrone Bearbeitung zu. Einzelheiten zu den beiden Varianten finden Sie auf den folgenden Seiten.

Task-Wechsel zulassen

Beispiel einer synchronen REQUEST-Transaktion, die einen Task-Wechsel in der Schleife zuläßt.

Client		Server
DdeClientTransaction (..XTYP_REQUEST.. ..<timeout>..)	→	XTYP_REQUEST: Schleife (Teil-)Bearbeitung durchführen if (PeekMessage(...)) TranslateMessage (..) DispatchMessage (..) bis Bearbeitung beendet Ergebnis → globales Datenobjekt return (Ergebnis)
	←	

Datenformate

Für den Datenaustausch an der DDE-Schnittstelle im CF_TEXT-Format wird folgende Umsetzung vorgenommen:

Zeichen	dezimal	hexadezimal	"Intel-Hex"
<CR>	13	0D	'0' 'D'
A	65	41	'4' '1'

Anwendung mit Server-Software 022

Installation

Die Server-Software zusammen mit allen DLL-Dateien und der INI-Datei müssen in ein gemeinsames Verzeichnis kopiert werden.

Initialisierungsdatei des Servers

Die Initialisierungsdatei enthält neben den Server-spezifischen Parametern folgende Sektionen für die Kommunikation mit dem Client über die DDE-Schnittstelle:

[Services]
; Auswerteeinheit des Identifikations-Systems
Service1 = ToolIdentSystem

[Topics]
; Nummer des Schreib-/Lesekopfes des Identifikations-Systems
Topic1=Unit1

[Misc]
; Verschiedenes
; maximale Codeträger-Kapazität in Byte
MAX_CC_CAPACITY=<n>

[Ports]
; verwendeter COM-Port
Port=COM<n>

Für <n> muß der entsprechende Wert eingegeben werden. Benutzen Sie dazu das Programm INISET.EXE, das Sie auf der Install-Diskette finden. Kopieren Sie die INISET.EXE in das Verzeichnis, in dem sich die BIS022SV.INI bzw. die BALLU.INI befindet, und führen Sie sie aus.

Service

Über **Service** wird die Auswerteeinheit des Identifikations-Systems angesprochen. Die vom Server angebotenen Services sind in der Sektion [Services] der Server-Initialisierungsdatei hinterlegt.

Topic

Über **Topic** wird die Nummer des Schreib-/Lesekopfes des Identifikations-Systems festgelegt. Die Topics sind in der Sektion [Topics] der Server-Initialisierungsdatei hinterlegt.

Misc

Über **Misc** wird die maximale Codeträgerkapazität (in Byte) angegeben. Diesen Wert können Sie aus dem Codeträgerdatenblatt entnehmen.

Port

Über **Port** wird die COM-Schnittstelle angegeben, an der die Auswerteeinheit angeschlossen ist.

Status

Bei der asynchronen Bearbeitung fragt der Client den Bearbeitungsstatus **Status** über einen "hot link" ab. Der Bearbeitungsstatus kann folgende Werte annehmen:

"000" Bearbeitung wurde fehlerfrei abgeschlossen.
Bei "Lesen vom Codeträger" folgen die Nettodaten.

"-01" Auftrag in Arbeit.

"-02" Bearbeitung wurde abgebrochen.

"nnn" Fehlernummer vom Identifikations-System.
Die Fehlernummern haben einen Wert größer "000".
Der zugehörige Fehlertext kann mit dem Auftrag "Fehlertext lesen" ermittelt werden.
Er wird in der eingestellten Sprache ausgegeben.

Transaktionen (Übersicht)

Der Server bietet folgende Dienste für die synchrone und asynchrone Bearbeitung an:

	Synchrone Bearbeitung		**Asynchrone Bearbeitung**		
	Transaktion	Item / Execute-String	Transaktion	Item / Execute-String	
Lesen vom Codeträger	REQUEST	"Data[<offset>, <count>]"	EXECUTE	"CMD=Read	OFFSET=<offset> COUNT=<count>"
			ADVISE	"Status"	
Schreiben auf den Codeträger	POKE	"Data[<offset>, <count]"	EXECUTE	"CMD=Write	OFFSET=<offset> COUNT=<count> DATA=<data>"
			ADVISE	"Status"	
Identifikations-System rücksetzen	EXECUTE	"CMD=Reset"	EXECUTE ADVISE	"CMD=Reset "Status"	ASYNC"
Initialisierungs-information übergeben	EXECUTE	"CMD=Init LANGUAGE=<path> [EOT=<e>]"	EXECUTE	"CMD=Init	LANGUAGE=<path> [EOT=<e>] ASYNC"
			ADVISE	"Status"	
Schreib-/Lesevorgang abbrechen	EXECUTE	"CMD=Abort"			
Fehlernummer lesen	REQUEST	"GetError"			
Fehlertext lesen	REQUEST	"GetErrortext"			

Daten vom Codeträger lesen

Synchrone Bearbeitung		Asynchrone Bearbeitung	
Transaktion:	Request-Item:	Transaktion:	Kommandostring:
XTYP_REQUEST	"Data[<offset>, <count>]"	XTYP_EXECUTE	"CMD=Read OFFSET=<offset> COUNT=<count>"
		ADVISE	"Status"

Der Lesevorgang beginnt bei der mit <offset> bestimmten Adresse.
Es wird die mit <count> angegebene Anzahl Bytes gelesen.
Eckige Klammern [ ] sind zwingend. <offset> und <count> sind 4stellig anzugeben.
Beispiel: Startadresse = 0, Anzahl Byte = 25.
Synchrone Bearbeitung: "Data[0000, 0025]"
Asynchrone Bearbeitung: "CMD=Read OFFSET=0000 COUNT=0025"
Die Daten werden im "Intel-Hex"-Format an den Client übergeben.

Bei der Initialisierung mit Datenendekennnug:
Wurde als Initialisierungsinformation eine Datenendekennung angegeben, wird maximal die mit <count> angeforderte Anzahl Bytes gelesen. Wird beim Lesen vorher die Endekennung erkannt, wird an dieser Stelle (einschl. Endekennung) das Lesen beendet.

Die Anzahl der gelieferten Bytes ergibt sich aus der Länge des vom Server übergebenen Datenobjekts.

Der Lesevorgang kann vom Client nicht abgebrochen werden. Task-Wechsel des Betriebssystems möglich.	Der Lesevorgang kann vom Client abgebrochen werden.

Daten vom Codeträger lesen, Reaktion des Servers

Synchrone Bearbeitung

In der Callback-Routine werden folgende Events bearbeitet:

XTYP_REQUEST

- Daten vom Codeträger lesen.
- Fehlerstatus des Identifikations-Systems auswerten.

Wenn Datenübertragung fehlerfrei abgeschlossen:

- Daten vom Binär- ins "Intel-Hex"-Format konvertieren.
- Daten in globalem Datenobjekt ablegen.
- Rückgabewert = Handle auf globales Datenobjekt.

Wenn Fehler vom Identifikations-System:

- Rückgabewert = DDE_FNOTPROCESSED
  Client erhält Rückgabewert und ermittelt Fehlernummer mit Auftrag "Fehlernummer lesen".
  Wert größer "000" zeigt Fehler des Identifikations-Systems.
  Fehlertext über Auftrag "Fehlertext lesen" ermitteln.

Asynchrone Bearbeitung

In der Callback-Routine werden folgende Events bearbeitet:

XTYP_EXECUTE (vom Client aufgerufen)
- Quittung: DDE_FACK, DDE_FBUSY oder DDE_FNOTPROCESSED *.

XTYP_ADVSTART (vom Client aufgerufen)
- "Status" = "-01", Quittung mit TRUE.

XTYP_REQUEST (vom Client aufgerufen)
- Item ist "Status", Timer starten.

XTYP_TIMER (bei Timerablauf)
- Bearbeitungsfunktion aufrufen (PostMessage).

XTYP_ADVREQ (vom Server aufgerufen)
- "Status" in globalem Datenobjekt ablegen.
- Rückgabewert = Handle auf globales Datenobjekt.

Bearbeitungsfunktion:
- Daten vom Codeträger lesen.
- Bearbeitungsstatus "Status" aktualisieren:
  - Bearbeitung nicht abgeschlossen: "Status" = "-01", PostMessage (Bearbeitungsfunktion)
  - Abbruch vom Client: "Status" = "-02", DdePostAdvise
  - Bearbeitung fehlerfrei abgeschlossen: Daten von Binär nach "Intel-Hex" konvertieren und nach "Status" kopieren, "Status" = "000<..daten..>", DdePostAdvise
  - Bearbeitung fehlerhaft abgeschlossen: "Status" = "<fehlernr wz-ident-system>", DdePostAdvise.
    Fehlertext über Auftrag "Fehlertext lesen" ermitteln.

* Client schickt DDE-Request mit Item "lasterror", Server antwortet mit "-03".

Daten auf den Codeträger schreiben

Synchrone Bearbeitung		Asynchrone Bearbeitung	
Transaktion:	Request-Item:	Transaktion:	Kommandostring:
XTYP_POKE	"Data[<offset>, <count>]"	XTYP_EXECUTE	"CMD=Write OFFSET=<offset> COUNT=<count> DATA=<data>"
		ADVISE	"Status"

Der Schreibvorgang beginnt bei der mit <offset> bestimmten Adresse.
Es wird die mit <count> angegebene Anzahl Bytes geschrieben.
Eckige Klammern [ ] sind zwingend. <offset> und <count> sind 4stellig anzugeben.
Beispiel: Startadresse = 0, Anzahl Byte = 25.
Synchrone Bearbeitung: "Data[0000, 0025]"
Asynchrone Bearbeitung: "CMD=Read OFFSET=0000 COUNT=0025"
Die Daten werden vom Client im "Intel-Hex"-Format übergeben.

Der Schreibvorgang kann vom Client nicht abgebrochen werden. Task-Wechsel des Betriebssystems möglich.	Der Schreibvorgang kann vom Client abgebrochen werden.

Daten auf den Codeträger schreiben, Reaktion des Servers

Synchrone Bearbeitung

In der Callback-Routine werden folgende Events bearbeitet:

XTYP_POKE

- Daten aus globalem Datenobjekt lesen.
- Daten von "Intel-Hex" nach Binär konvertieren.
- Daten auf Codeträger schreiben.
- Fehlerstatus des Identifikations-Systems auswerten.

Wenn Datenübertragung fehlerfrei abgeschlossen:

- Rückgabewert = DDE_FACK

Wenn Fehler vom Identifikations-System:

- Rückgabewert = DDE_FNOTPROCESSED
  Client erhält Rückgabewert DDE_FNOTPROCESSED und ermittelt Fehlernummer mit Auftrag "Fehlernummer lesen". Wert größer "000" zeigt Fehler des Identifikations-Systems. Fehlertext über Auftrag "Fehlertext lesen" ermitteln.

Asynchrone Bearbeitung

In der Callback-Routine werden folgende Events bearbeitet:

XTYP_EXECUTE (vom Client aufgerufen)
- Quittung: DDE_FACK, DDE_FBUSY oder DDE_FNOTPROCESSED *.

XTYP_ADVSTART (vom Client aufgerufen)
- "Status" = "-01", Quittung mit TRUE.

XTYP_REQUEST (vom Client aufgerufen)
- Item ist "Status", Timer starten.

WM_TIMER (bei Timerablauf)
- Bearbeitungsfunktion aufrufen (PostMessage).

XTYP_ADVREQ (vom Server aufgerufen)
- "Status" in globalem Datenobjekt ablegen.
- Rückgabewert = Handle auf globales Datenobjekt.

Bearbeitungsfunktion:
- Daten von "Intel-Hex" nach Binär konvertieren.
- Daten auf Codeträger schreiben.
- Bearbeitungsstatus "Status" aktualisieren:
  - Bearbeitung nicht abgeschlossen: "Status" = "-01", PostMessage (Bearbeitungsfunktion)
  - Abbruch vom Client: "Status" = "-02", DdePostAdvise
  - Bearbeitung fehlerfrei abgeschlossen: "Status" = "000", DdePostAdvise
  - Bearbeitung fehlerhaft abgeschlossen: "Status" = "<fehlernr wz-ident-system>", DdePostAdvise. Fehlertext über Auftrag "Fehlertext lesen" ermitteln.

* Client schickt DDE-Request mit Item "lasterror", Server antwortet mit "-03".

Identifikations-System rücksetzen

Synchrone Bearbeitung		Asynchrone Bearbeitung	
Transaktion:	Kommandostring:	Transaktion:	Kommandostring:
XTYP_EXECUTE	"CMD=Reset"	XTYP_EXECUTE ADVISE	"CMD=Reset ASYNC" "Status"

Das Identifikations-System wird in den Grundzustand gebracht.

Die Aktion kann vom Client nicht abgebrochen werden. Task-Wechsel des Betriebssystems möglich.	Die Aktion kann vom Client abgebrochen werden.

Identifikations-System rücksetzen, Reaktion des Servers

Synchrone Bearbeitung

In der Callback-Routine werden folgende Events bearbeitet:

XTYP_EXECUTE

- Reset-Telegramm an das Identifikations-System senden.
- Fehlerstatus des Identifikations-Systems auswerten.

Wenn Datenübertragung fehlerfrei abgeschlossen:

- Rückgabewert = DDE_FACK

Wenn Fehler aufgetreten:

- Rückgabewert = DDE_FNOTPROCESSED
  Client erhält Rückgabewert DDE_FNOTPROCESSED und ermittelt Fehlernummer mit Auftrag "Fehlernummer lesen". Fehlertext über Auftrag "Fehlertext lesen" ermitteln.

Asynchrone Bearbeitung

In der Callback-Routine werden folgende Events bearbeitet:

XTYP_EXECUTE (vom Client aufgerufen)
- Quittung: DDE_FACK, DDE_FBUSY oder DDE_FNOTPROCESSED *.

XTYP_ADVSTART (vom Client aufgerufen)
- "Status" = "-01", Quittung mit TRUE.

XTYP_REQUEST (vom Client aufgerufen)
- Item ist "Status", Timer starten.

WM_TIMER (bei Timerablauf)
- Bearbeitungsfunktion aufrufen (PostMessage).

XTYP_ADVREQ (vom Server aufgerufen)
- "Status" in globalem Datenobjekt ablegen.
- Rückgabewert = Handle auf globales Datenobjekt.

Bearbeitungsfunktion:
- Reset-Telegramm an das Identifikations-System senden.
- Bearbeitungsstatus "Status" aktualisieren:
  - Bearbeitung nicht abgeschlossen: "Status" = "-01", PostMessage (Bearbeitungsfunktion)
  - Abbruch vom Client: "Status" = "-02", DdePostAdvise
  - Bearbeitung fehlerfrei abgeschlossen: "Status" = "000", DdePostAdvise
  - Bearbeitung fehlerhaft abgeschlossen: "Status" = "<fehlernr>", DdePostAdvise.
    Fehlertext über Auftrag "Fehlertext lesen" ermitteln.

* Client schickt DDE-Request mit Item "lasterror", Server antwortet mit "-03".

Initialisierungs-information übergeben

Synchrone Bearbeitung		Asynchrone Bearbeitung	
Transaktion:	Kommandostring:	Transaktion:	Kommandostring:
XTYP_EXECUTE	"CMD=Init LANGUAGE=<path> [EOT=0x<h>]"	XTYP_EXECUTE	"CMD=Init LANGUAGE=<path> [EOT=0x<h>] ASYNC"
		ADVISE	"Status"

<path> kompletter Pfad und Dateiname der Sprach-DLL für Fehlermeldung des Servers.

<h> Endekennung für das Lesen der Daten vom Codeträger als HEX-String (z.B. EOT=0x2F2F definiert das Endezeichen "//")

Die Angabe [EOT=0x<h>] ist optional und kann komplett weggelassen werden, wenn kein EOT gewünscht wird.

Vor dem Datenaustausch mit dem Identifikations-System übergibt der Client an den Server Pfad und Name der Sprach-DLL und optional eine Datenendekennung fürs Lesen als Initialisierungsinformation. Die Sprach-DLL wird vom Server geladen. Ist eine Datenendekennung angegeben, wird maximal die mit <count> angeforderte Anzahl Bytes gelesen. Wird während des Lesens die Endekennung erkannt, wird an dieser Stelle (einschl. Endekennung) das Lesen beendet.

Die Aktion kann vom Client nicht abgebrochen werden. Task-Wechsel des Betriebssystems möglich.	Die Aktion kann vom Client abgebrochen werden.

Initialisierungs-information übergeben, Reaktion des Servers

Synchrone Bearbeitung

In der Callback-Routine werden folgende Events bearbeitet:

XTYP_EXECUTE

- Initialisierungsinformation auswerten, Sprach-DLL entsprechend <path> laden.
- Fehlerstatus des Identifikations-Systems auswerten.

Wenn fehlerfrei abgeschlossen:

- Rückgabewert = DDE_FACK

Wenn Fehler aufgetreten:

- Rückgabewert = DDE_FNOTPROCESSED
  Client erhält Rückgabewert DDE_FNOTPROCESSED und ermittelt Fehlernummer mit Auftrag "Fehlernummer lesen". Fehlertext über Auftrag "Fehlertext lesen" ermitteln.

Asynchrone Bearbeitung

In der Callback-Routine werden folgende Events bearbeitet:

XTYP_EXECUTE (vom Client aufgerufen)
- Quittung: DDE_FACK, DDE_FBUSY oder DDE_FNOTPROCESSED *.

XTYP_ADVSTART (vom Client aufgerufen)
- "Status" = "-01", Quittung mit TRUE.

XTYP_REQUEST (vom Client aufgerufen)
- Item ist "Status", Timer starten.

WM_TIMER (bei Timerablauf)
- Bearbeitungsfunktion aufrufen (PostMessage).

XTYP_ADVREQ (vom Server aufgerufen)
- "Status" in globalem Datenobjekt ablegen.
- Rückgabewert = Handle auf globales Datenobjekt.

Bearbeitungsfunktion:
- Initialisierungsinformation auswerten.
- Bearbeitungsstatus "Status" aktualisieren:
  - Bearbeitung nicht abgeschlossen: "Status" = "-01", PostMessage (Bearbeitungsfunktion)
  - Abbruch vom Client: "Status" = "-02", DdePostAdvise
  - Bearbeitung fehlerfrei abgeschlossen: "Status" = "000", DdePostAdvise
  - Bearbeitung fehlerhaft abgeschlossen: "Status" = "<fehlernr>", DdePostAdvise.
    Fehlertext über Auftrag "Fehlertext lesen" ermitteln.

* Client schickt DDE-Request mit Item "lasterror", Server antwortet mit "-03".

Schreib-/ Lesevorgang abbrechen

Nur bei asynchroner Bearbeitung möglich

Transaktion:	Kommandostring:
XTYP_EXECUTE	"CMD=Abort"

Ein Schreib-/Lesevorgang soll auf Anforderung des Anwenders abgebrochen werden.

Reaktion Server In der Callback-Routine wird Event XTYP_EXECUTE bearbeitet:

– Globale Abbruchkennung setzen (wird in der Bearbeitungsfunktion abgefragt).
– Rückgabewert = DDE_FACK

Fehlernummer lesen

Nur bei synchroner Bearbeitung sinnvoll
(bei asynchroner Bearbeitung wird die Fehlernummer im Bearbeitungsstatus übergeben)

Transaktion:	Request-Item:
XTYP_REQUEST	"GetError"

Nach einem fehlerhaften Lese- oder Schreibauftrag soll die Fehlernummer des Identifikations-Systems gelesen werden.

Reaktion Server In der Callback-Routine wird Event XTYP_REQUEST bearbeitet:

– Globale Fehlernummer (wird in der Bearbeitungsfunktion versorgt).

Fehlertext lesen

Transaktion:	Request-Item:
XTYP_REQUEST	"GetErrortext"

Nach einem fehlerhaften Auftrag soll der Fehlertext des Identifikations-Systems gelesen werden. Die Fehlertexte sind in mehreren Sprachen in Sprach-DLLs hinterlegt. Die aktuelle DLL wird bei der Übergabe der Initialisierungsinformation geladen. Der Fehlertext erscheint in der angeforderten Sprache. Die maximale Länge des Fehlertextes beträgt 60 Zeichen.

Reaktion Server

In der Callback-Routine wird Event XTYP_REQUEST bearbeitet:

– Rückgabewert = Fehlertext zu der zuletzt aufgetretenen Fehlernummer aus der Sprach-DLL in der angeforderten Sprache.

Verfügbare Sprach-DLLs:	deutsch	(Ballu_gr.dll)
	englisch	(Ballu_uk.dll)
	französisch	(Ballu_fr.dll)
	italienisch	(Ballu_it.dll)
	spanisch	(Ballu_sp.dll)
	portugiesisch	(Ballu_po.dll)

Fehlermeldungen
Server-Software 022

Fehlernummern und Fehlertexte

Fehler-Nr.	Fehlertexte
000	"Kein Fehler, Funktion korrekt durchgeführt."
001	"Kein Codeträger vorhanden."
002	"Fehler beim Codeträger lesen."
003	"Lesen abgebrochen, da Codeträger entfernt wurde."
004	"Fehler beim Codeträger beschreiben."
005	"Schreiben abgebrochen, da Codeträger entfernt wurde."
006	"Schnittstellenfehler vom Toolidentsystem erkannt."
007	"Telegrammformatfehler zum Toolidentsystem."
008	"Prüfsummenfehler zwischen Server und Toolidentsystem."
009	"Kabelbruch des Schreib/Lesekopfes."
010	"Sprach-DLL (Fehlertexte) konnte nicht geladen werden."
011	"COM-Port kann nicht geöffnet werden."
012	"Kommandostring unbekannt."
013	"Anzahl Byte unzulässig."
014	"Nicht-BCD-Zeichen in empfangenen Schreibdaten gefunden."
015	"COM-Port kann nicht geschlossen werden."
016	"Fehler beim Parametrieren des COM-Ports."
017	"Fehler beim Empfangen vom COM-Port."
018	"Fehler beim Senden zum COM-Port."
019	"Startadresse + Anzahl Byte > MAX_CC_CAPACITY in BIS022SV.INI."

Anwendung mit Standard-Protokoll 007

Nach der Übersicht über die Telegrammarten werden die Bildung des Blockchecks und die Varianten der Endekennung beschrieben.
Auf den Seiten 50/51 wird der Telegrammablauf schematisch dargestellt.
Ab Seite 52 folgen die detailierten Informationen zum korrekten Aufbau der Telegramme .

Für die einzelnen Aufgaben im Identifikations-System BIS C existieren spezifische Telegramme. Sie beginnen stets mit dem Befehl, der der Telegrammart zugeordnet ist:

Telegrammarten mit zugehörigem Befehl (ASCII-Zeichen)

- **'L'** Lesen des Codeträgers mit Anwahl des Schreib-/Lesekopfes und der Blockgröße
- **'P'** Schreiben auf den Codeträger mit Anwahl des Schreib-/Lesekopfes und der Blockgröße
- **'C'** Schreiben eines konstanten Wertes auf den Codeträger mit Anwahl des Schreib-/Lesekopfes und der Blockgröße
- **'R'** Lesen des Codeträgers
- **'W'** Schreiben auf den Codeträger
- **'H'** Anwahl des Schreib-/Lesekopfes und der Blockgröße mit den Varianten
  - '?' Suchen des nächsten Codeträgers (einmal)
  - '!' oder Suchen des nächsten Codeträgers (ständig)
- **'B'** Ausgänge bearbeiten
- **'Q'** Neustart der Auswerteeinheit (Quit)
- **'S'** Abfrage der Statusmeldung

Bitte beachten Sie:
– Eine Dauerabfrage auf der Schnittstelle ist nicht zulässig!
– Die Mindestwartezeit zwischen zwei Befehlen beträgt 300 ms!

Anwendung mit Standard-Protokoll 007

Erklärung einiger Telegramminhalte

Startadresse und Anzahl Bytes	Die Startadresse (A3, A2, A1, A0) und die Anzahl der zu übertragenden Bytes (L3, L2, L1, L0) werden dezimal als ASCII-Zeichen übertragen. Für die Startadresse kann der Bereich 0000 bis 8191 und für die Anzahl Byte 0001 bis 8192 verwendet werden. A3 ... L0 stehen für je ein ASCII-Zeichen. **Bitte beachten Sie:** Startadresse + Anzahl Byte dürfen die Codeträger-kapazität nicht überschreiten.
Kopfnummer und Blockgröße	Bei den Befehlen 'L' (Lesen mit Kopfanwahl und Blockgröße) und 'P' (Schreiben mit Kopfanwahl und Blockgröße) wird zuerst die Nummer des Schreib-/Lesekopfes K ('1' oder '2') und danach die Blockgröße B ('0', '1') des Codeträgers übertragen. B = '0' entspricht 64 Byte, B = '1' entspricht 32 Byte.
Quittung	Die Quittung <ACK> '0' wird vom Identifikations-System gesendet, wenn die seriell übertragenen Zeichen als richtig erkannt wurden und sich ein Codeträger im Arbeitsbereich eines Schreib-/Lesekopfs befindet. Beim Befehl 'R' wird <ACK> '0' erst gegeben, wenn die Daten zur Übertragung bereit sind. Mit <NAK> + 'Fehlernr.' wird quittiert, wenn ein Fehler erkannt wurde oder wenn sich kein Codeträger im Arbeitsbereich des Schreib-/Lesekopfs befindet.
Start	Mit <STX> wird die Datenübertragung gestartet.
Übertragene Bytes	Die Daten werden codetransparent (ohne Datenwandlung) übertragen.

Bildung des Blockchecks BCC

Der Blockcheck BCC wird als EXOR-Verknüpfung aus den seriell übertragenen Binärzeichen des Telegrammblocks gebildet. Beispiel: Lesen ab Adresse 13, 128 Byte sind zu lesen.
Die Befehlszeile ohne BCC lautet: 'L 0013 0128 20'. BCC wird gebildet:

'L	=	0100 1100	EXOR
0	=	0011 0000	EXOR
0	=	0011 0000	EXOR
1	=	0011 0001	EXOR
3	=	0011 0011	EXOR
0	=	0011 0000	EXOR
1	=	0011 0001	EXOR
2	=	0011 0010	EXOR
8	=	0011 1000	EXOR
2	=	0011 0010	EXOR
0'	=	0011 0000	EXOR
ergibt als Blockcheck: BCC	=	0100 0111	= 'G'

Variante bei Abschluß mit BCC, Endekennung

Bei Bedarf kann der Abschluß mittels Blockcheck BCC durch ein spezielles ASCII-Zeichen ersetzt werden. Dies ist:

– Carriage Return 'CR'

Für Steuereinheiten, die immer ein Endekennungszeichen benötigen, muß dieses überall in die Telegramme eingefügt werden. Zur Verfügung stehen:

– Carriage Return 'CR' oder
– Line Feed mit Carriage Return 'LF CR'.

Auf der folgenden Seite werden die verschiedenen Protokollvarianten dargestellt.
Siehe auch: Konfiguration ab Seite 11.

Darstellung der verschiedenen Protokollvarianten

Von der vorangegangenen Seite stammt die Befehlszeile 'L 0013 0128 20 G' mit 'G' als BCC. Diese Befehlszeile wird hier in den möglichen Varianten gegenübergestellt; dabei werden auch die verschiedenen Formen der Quittung mit und ohne Endekennung dargestellt:

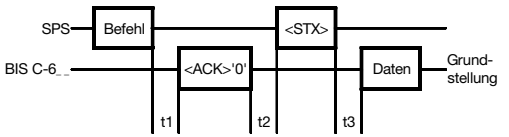
Befehlszeile vom steuernden System zum BIS	Quittung vom BIS bei korrektem Empfang	Quittung vom BIS bei inkorrektem Empfang
mit BCC als Abschluß, ohne Endekennung 'L 0013 0128 20 G'	ohne Endekennung <ACK> '0'	ohne Endekennung <NAK> '1'
mit 'CR' anstatt BCC, ohne Endekennung 'L 0013 0128 20 CR'	ohne Endekennung <ACK> '0'	ohne Endekennung <NAK> '1'
ohne BCC, mit Endekennung 'CR' 'L 0013 0128 20 CR'	mit Endekennung 'CR' <ACK> '0 CR'	mit Endekennung 'CR' <NAK> '1 CR'
ohne BCC, mit Endekennung 'LF CR' 'L 0013 0128 20 LF CR'	mit Endekennung 'LF CR' <ACK> '0 LF CR'	mit Endekennung 'LF CR' <NAK> '1 LF CR'

In der Tabelle ist als Fehlerbeispiel <NAK> '1' (= kein Codeträger vorhanden) angegeben.

Die jeweiligen Positionen für die zusätzliche Endekennung sind in den tabellarischen Darstellungen kursiv abgesetzt.

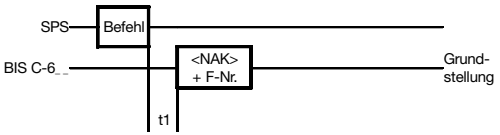
Protokollablauf: Dialogmodus ohne Kopfumschaltung

Lesen: a) Es tritt kein Fehler auf:



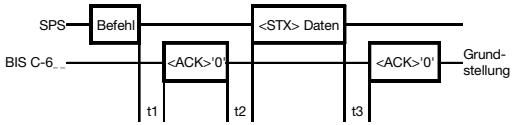
t1 je nach Anzahl zu lesender Bytes (siehe Seite 68/69)
t2 ≥ 0 (wird von der Auswerteeinheit nicht überwacht)
t3 = max. 50 ms

b) Es tritt ein Fehler auf:



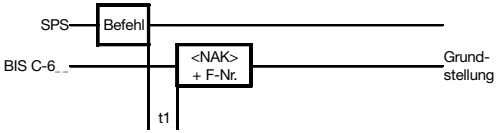
t1 je nach Anzahl zu lesender Bytes (siehe Seite 68/69) und Fehlerart (empfohlene Überwachungszeit: 15 s)

Schreiben: a) Es tritt kein Fehler auf:



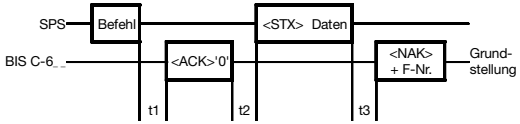
t1 = max. 50 ms
t2 ≥ 0 (wird von der Auswerteeinheit nicht überwacht)
t3 je nach Anzahl zu schreibender Bytes (siehe Seite 68/69)

b) Es tritt ein Fehler im Befehl auf:



t1 = max. 50 ms
t2 ≥ 0 (wird von der Auswerteeinheit nicht überwacht)
t3 je nach Anzahl zu schreibender Bytes (siehe Seite 68/69) und Fehlerart (empfohlene Überwachungszeit: 30 s bei Codeträgern mit 32 Byte/Block, 60 s bei Codeträgern mit 64 Byte/Block)

c) Es tritt ein Fehler beim Schreiben auf:



t1 = max. 50 ms

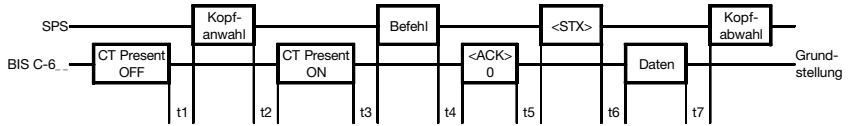
Voraussetzung für die Gültigkeit der Darstellungen:
- Die Auswerteeinheit muß sich in Grundstellung befinden.
- Vor dem Schreib-/Lesekopf befindet sich ein Codeträger.

Anwendung mit Standard-Protokoll 007

Protokollablauf: Dialogmodus mit Kopfumschaltung

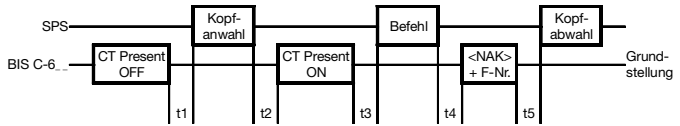
Lesen:

a) Es tritt kein Fehler auf:



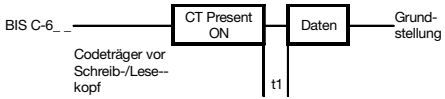
t1, t3, t7 ≥ 0
t2 = max. 500 ms
t4 je nach Anzahl zu lesender Bytes (siehe Seite 68/69)
t5 ≥ 0 (wird von der Auswerteeinheit nicht überwacht)
t6 = max. 50 ms

b) Es tritt ein Fehler auf:



t1, t3, t5 ≥ 0
t2 = max. 500 ms
t4 je nach Anzahl zu lesender Bytes (siehe Seite 68/69) und Fehlerart (empfohlene Überwachungszeit: 15 s)

Protokollablauf: Direkt-Lesemodus



t1 je nach Anzahl zu lesender Byte (siehe Seite 68/69)

Voraussetzung für die Gültigkeit der Darstellungen:

– Die Auswerteeinheit muß sich in Grundstellung befinden.
– Vor dem Schreib-/Lesekopf befindet sich ein Codeträger.

Lesen vom Codeträger mit Anwahl des Schreib-/Lesekopfs und der Blockgröße
Schreiben auf den Codeträger mit Anwahl des Schreib-/Lesekopfs und der Blockgröße

Task	Datenfluß	Befehl	Startadresse des ersten zu übertragenden Byte	Anzahl der zu übertragenden Bytes	Kopfnummer	Blockgröße	Abschluß 2)	Quittung 3)	*Endekennung 4)*	Start zur Übertragung	*Endekennung 4)*	Daten (von Startadresse bis Startadresse + Anzahl Bytes)	Abschluß 2)	Quittung 3)	*Endekennung 4)*
Lesen	vom steuernden System zum BIS	**'L'**	A3 A2 A1 A0 '0 0 0 0' bis '8 1 9 1'	L3 L2 L1 L0 '0 0 0 1' bis '8 1 9 2'	K '1' oder '2'	B '0' oder '1'	BCC oder siehe 2)			<STX>	*'CR' oder 'LF CR'*				
	vom BIS zum steuernden System							<ACK>'0' oder <NAK> + Fehler-Nr.	*'CR' oder 'LF CR'*			D1 D2 D3 ... Dn	BCC oder siehe 2)		
			1)								1)				
Schreiben	vom steuernden System zum BIS	**'P'**	A3 A2 A1 A0 '0 0 0 0' bis '8 1 9 1'	L3 L2 L1 L0 '0 0 0 1' bis '8 1 9 2'	K '1' oder '2'	B '0' oder '1'	BCC oder siehe 2)			<STX>		D1 D2 D3 ... Dn	BCC oder siehe 2)		
	vom BIS zum steuernden System							<ACK>'0' oder <NAK> + Fehler-Nr.	*'CR' oder 'LF CR'*					<ACK>'0' oder <NAK> + Fehler-Nr.	*'CR' oder 'LF CR'*
			1)									1)			

1) Die Befehle Status und/oder Quit sind an dieser Stelle nicht zugelassen.
2) Statt Blockcheck BCC kann je nach Protokollvariante entweder Carriage Return 'CR' oder Line Feed mit Carriage Return 'LF CR' verwendet werden.
3) Als Quittung kommt <ACK> '0', wenn kein Fehler aufgetreten ist, oder <NAK> + 'Fehlernr.', wenn ein Fehler aufgetreten ist.
4) Bei Protokollvarianten, die immer eine Endekennung benötigen, muß hier eines der Abschlußzeichen 'CR' oder 'LF CR' eingefügt werden.

Angaben in spitzen Klammern stellen ein Steuerzeichen dar.
Angaben in Hochkommata stellen das/die jeweilige/n Zeichen im ASCII-Code dar.

Telegrammbeispiel zu Seite 52:
Lesen vom Codeträger mit Anwahl des Schreib-/Lesekopfs und der Blockgröße
mit Blockcheck (BCC)

-> Kopf 1 ist angewählt. Es sollen 10 Byte ab Adresse 50 vom Codeträger am Schreib-/Lesekopf 2 gelesen werden. Der Codeträger vor Kopf 4 hat eine Blockgröße von 64 Byte.

Das Steuersystem sendet 'L 0050 0010 20 J' BCC ($4A_{HEX}$)

Adresse des ersten zu lesenden Byte
Anzahl der zu lesenden Byte
Schreib-/Lesekopf Nr. 2
Blockgröße 0 = 64 Byte

Die BIS-Auswerteeinheit quittiert mit <ACK> '0'
Das Steuersystem gibt den Startbefehl <STX>
Die BIS-Auswerteeinheit liefert die Daten vom Codeträger '1 2 3 4 5 6 7 8 9 A F' BCC (70_{HEX})

Nach Ablauf des Telegrammverkehrs bleibt Kopf 2 mit 64 Byte Blockgröße angewählt.

Telegrammbeispiel zu Seite 52:
Schreiben auf den Codeträger mit Anwahl des Schreib-/Lesekopfs und der Blockgröße
mit Blockcheck (BCC)

-> Kopf 1 ist angewählt. Es sollen 5 Byte ab Adresse 500 auf den Codeträger am Schreib-/Lesekopf 2 geschrieben werden. Der Codeträger vor Kopf 2 hat 64 Byte Blockgröße.

Das Steuersystem sendet 'P 0500 0005 20 R' BCC (52_{HEX})

Adresse des ersten zu schreibenden Byte
Anzahl der zu schreibenden Byte
Schreib-/Lesekopf Nr. 2
Blockgröße 0 = 64 Byte

Die BIS-Auswerteeinheit quittiert mit <ACK> '0'
Das Steuersystem gibt den Startbefehl und die Daten <STX> '1 2 3 4 5 3' BCC (33_{HEX})
Die Auswerteeinheit quittiert mit <ACK> '0'

Nach Ablauf des Telegrammverkehrs bleibt Kopf 2 mit 64 Byte Blockgröße angewählt.

Angaben in spitzen Klammern stellen ein Steuerzeichen dar.
Angaben in Hochkommata stellen das/die jeweilige/n Zeichen im ASCII-Code dar.

Schreiben eines konstanten Wertes auf den Codeträger mit Anwahl des Schreib-/Lesekopfs und der Blockgröße

Dieser Befehl kann zum Löschen eines Codeträgers verwendet werden. Man spart die Zeit zur Übertragung der zu schreibenden Byte.

Task	Datenfluß	Befehl	Startadresse des ersten zu übertragenden Byte	Anzahl der zu übertragenden Bytes	Kopfnummer	Blockgröße	Abschluß 2)	Quittung 3)	*Endekennung 4)*	Start zur Übertragung	*Endekennung 4)*	Daten (von Startadresse bis Startadresse + Anzahl Bytes)	Abschluß 2)	Quittung 3)	*Endekennung 4)*
Schreiben	vom steuernden System zum BIS	**'C'**	A3 A2 A1 A0 '0 0 0 0' bis '8 1 9 1'	L3 L2 L1 L0 '0 0 0 1' bis '8 1 9 2'	K '1' oder '2'	B '0' oder '1'	BCC oder siehe 2)			<STX>		D	BCC oder siehe 2)		
	vom BIS zum steuernden System							<ACK>'0' oder <NAK> + F-Nr.	*'CR' oder 'LF CR'*					<ACK>'0' oder <NAK> + F-Nr.	*'CR' oder 'LF CR'*
			1)									1)			

1) Die Befehle Status und/oder Quit sind an dieser Stelle nicht zugelassen.
2) Statt Blockcheck BCC kann je nach Protokollvariante entweder Carriage Return 'CR' oder Line Feed mit Carriage Return 'LF CR' verwendet werden.
3) Als Quittung kommt <ACK> '0', wenn kein Fehler aufgetreten ist, oder <NAK> + 'Fehlernr.', wenn ein Fehler aufgetreten ist.
4) Bei Protokollvarianten, die immer eine Endekennung benötigen, muß hier eines der Abschlußzeichen 'CR' oder 'LF CR' eingefügt werden.

Angaben in spitzen Klammern stellen ein Steuerzeichen dar.
Angaben in Hochkommata stellen das/die jeweilige/n Zeichen im ASCII-Code dar.

Telegrammbeispiel zu Seite 54:
Schreiben auf den Codeträger mit Anwahl des Schreib-/ Lesekopfs und der Blockgröße
mit Blockcheck (BCC)

-> Kopf 1 ist angewählt. Es sollen 500 Byte ab Adresse 20 auf den Codeträger am Schreib-/ Lesekopf 2 mit dem ASCII Datenwert 0 (30_{HEX}) geschrieben werden. Der Codeträger vor Kopf 2 hat eine Blockgröße 64 Byte.

Das Steuersystem sendet 'C 0020 0500 20 F' BCC (46_{HEX})

Adresse des ersten zu schreibenden Byte
Anzahl der zu schreibenden Byte
Schreib-/Lesekopf Nr. 2
Blockgröße 0 = 64 Byte

Die BIS-Auswerteeinheit quittiert mit <ACK> '0'
Das Steuersystem gibt den Startbefehl und die Daten <STX> '0 2' BCC (32_{HEX})
Die Auswerteeinheit quittiert mit <ACK> '0'

Nach Ablauf des Telegrammverkehrs bleibt Kopf 2 mit 64 Byte Blockgröße angewählt.

Angaben in spitzen Klammern stellen ein Steuerzeichen dar.
Angaben in Hochkommata stellen das/die jeweilige/n Zeichen im ASCII-Code dar.

Lesen vom Codeträger, Schreiben auf den Codeträger

Task	Datenfluß	Befehl	Startadresse des ersten zu übertragenden Byte	Anzahl zu übertragender Bytes	Abschluß 2)	Quittung 3)	*Endekennung 4)*	Start zur Übertragung	*Endekennung 4)*	Daten (von Startadresse bis Startadresse + Anzahl Bytes)	Abschluß 2)	Quittung 3)	*Endekennung 4)*
Lesen	vom steuernden System zum BIS	**'R'**	A3 A2 A1 A0 '0 0 0 0' bis '8 1 9 1'	L3 L3 L1 L0 '0 0 0 1' bis '8 1 9 2'	BCC oder siehe 2)			<STX>	*'CR' oder 'LF CR'*				
	vom BIS zum steuernden System					<ACK>'0' oder <NAK> + Fehler-Nr.	*'CR' oder 'LF CR'*			D1 D2 D3 ... Dn	BCC oder siehe 2)		
			1)										
Schreiben	vom steuernden System zum BIS	**'W'**	A3 A2 A1 A0 '0 0 0 0' bis '8 1 9 1'	L3 L2 L1 L0 '0 0 0 1' bis '8 1 9 2'	BCC oder siehe 2)			<STX>		D1 D2 D3 ... Dn	BCC oder siehe 2)		
	vom BIS zum steuernden System					<ACK>'0' oder <NAK> + Fehler-Nr.	*'CR' oder 'LF CR'*					<ACK>'0' oder <NAK> + Fehler-Nr.	*'CR' oder 'LF CR'*
			1)							1)			

1) Die Befehle Status und/oder Quit sind an dieser Stelle nicht zugelassen.
2) Statt Blockcheck BCC kann je nach Protokollvariante Carriage Return 'CR' oder Line Feed mit Carriage Return 'LF CR' verwendet werden.
3) Als Quittung kommt <ACK> '0', wenn kein Fehler aufgetreten ist, oder <NAK> + 'Fehlernr.', wenn ein Fehler aufgetreten ist.
4) Bei Protokollvarianten, die immer eine Endekennung benötigen, muß hier eines der Abschlußzeichen 'CR' oder 'LF CR' eingefügt werden.

Angaben in spitzen Klammern stellen ein Steuerzeichen dar.
Angaben in Hochkommata stellen das/die jeweilige/n Zeichen im ASCII-Code dar.

Anwendung mit Standard-Protokoll 007

Telegrammbeispiel zu Seite 56:
Lesen vom Codeträger
mit Blockcheck (BCC)

Lesen vom Codeträger: -> Es sollen 10 Byte ab Adresse 50 gelesen werden.

Das Steuersystem sendet	'R 0 0 5 0 0 0 1 0 V' BCC (56HEX)
Adresse des ersten zu lesenden Byte	
Anzahl der zu lesenden Byte	
Die BIS-Auswerteeinheit quittiert mit	<ACK> '0'
Das Steuersystem gibt den Startbefehl	<STX>
Die BIS-Auswerteeinheit liefert die Daten vom Codeträger	'1 2 3 4 5 6 7 8 9 0 SOH' BCC (01HEX)

Telegrammbeispiel zu Seite 56:
Schreiben auf den Codeträger
mit Blockcheck (BCC)

Schreiben auf den Codeträger: -> Es sollen 5 Byte ab Adresse 500 geschrieben werden.

Das Steuersystem sendet	'W 0 5 0 0 0 0 0 5 W' BCC (57HEX)
Die BIS-Auswerteeinheit quittiert mit	<ACK> '0'
Das Steuersystem sendet die Daten	<STX> '1 2 3 4 5 3' BCC (33HEX)
Die BIS-Auswerteeinheit quittiert mit	<ACK> '0'

Die Befehle 'R' und 'W' stellen eine Untermenge der Befehle 'L' und 'P' dar.

Angaben in spitzen Klammern stellen ein Steuerzeichen dar.
Angaben in Hochkommata stellen das/die jeweilige/n Zeichen im ASCII-Code dar.

Anwahl des Schreib-/Lesekopfs

Mit dem Befehl 'H1' kann der Schreib-/Lesekopf 1, mit 'H2' der Schreib-/Lesekopf 2 und mit 'HT' (Head Twin) können beide Köpfe angewählt werden.

Sind beide Köpfe angewählt, ist zu beachten:

1. Es darf immer nur an einem Schreib-/Lesekopf ein Codeträger vorhanden sein.
2. Die Schreib- oder Lesezeit verlängert sich um ca. 40 ms, unabhängig von der Datenanzahl, die gelesen oder geschrieben werden soll. (Dies gilt nicht für die Codeträgererkennung).
3. Die positive Quittung bei einem Schreib- oder Leseauftrag heißt nicht mehr <ACK> '0' sondern <ACK> '1' oder <ACK> '2' je nachdem, vor welchem Schreib-/Lesekopf sich gerade ein Codeträger befindet, der gelesen oder beschrieben wurde.

Task	Datenfluß	Befehl	Kopfnummer	Abschluß 2)	Quittung 3)	*Endekennung 4)*
Anwahl Schreib-/ Lesekopf	vom steuernden System zum BIS	**'H'**	'1', '2' oder 'T'	BCC oder siehe 2)		
	vom BIS zum steuernden System				<ACK>'1' oder <ACK>'2' bzw. <NAK> + Fehler-Nr.	*'CR' oder 'LF CR'*
			1)			

1) Die Befehle Status und/oder Quit sind an dieser Stelle nicht zugelassen.
2) Statt Blockcheck BCC kann je nach Protokollvariante Carriage Return 'CR' oder Line Feed mit Carriage Return 'LF CR' verwendet werden.
3) Als Quittung kommt <ACK>'1' oder <ACK>'2', wenn kein Fehler aufgetreten ist, oder <NAK> + 'Fehlernr.', wenn ein Fehler aufgetreten ist.
4) Bei Protokollvarianten, die immer eine Endekennung benötigen, muß hier eines der Abschlußzeichen 'CR' oder 'LF CR' eingefügt werden.

Telegrammbeispiel:
Anwahl des Schreib-/ Lesekopfs
mit Blockcheck (BCC)

-> Es soll auf Kopf 1 umgeschaltet werden.

Das Steuersystem sendet	'H 1	y'	BCC (79_{HEX})
Die BIS-Auswerteeinheit quittiert mit	<ACK>'1'	⌊_____⌋	

Angaben in spitzen Klammern stellen ein Steuerzeichen dar.
Angaben in Hochkommata stellen das/die jeweilige/n Zeichen im ASCII-Code dar.

Anwendung mit Standard-Protokoll 007

Nächsten Codeträger suchen (einmal)

Mit dem nachfolgend angegebenen Telegramm wird der nächste Codeträger gesucht. Dabei wird zum nächstfolgenden Schreib-/Lesekopf weitergeschaltet und geprüft, ob sich ein Codeträger vor diesem Schreib-/Lesekopf befindet. Wenn ja, enthält die Telegrammrückmeldung die zugehörige Nummer des Schreib-/Lesekopfs und die ersten 4 Byte des Codeträgers. Wenn nein, wird der ursprüngliche Schreib-/Lesekopf wieder angewählt und geprüft. Wird auch hier kein Codeträger gefunden, dann lautet die Telegrammrückmeldung: 'H ? 0000 w'.

'H ?' erkennt jeden Codeträger, unabhängig von der eingestellten Blockgröße, vorausgesetzt, Schreib-/Lesekopf und Codeträger sind kompatibel.

Task	Datenfluß	Befehl	Kennung	Abschluß 2)	Quittung	*Endekennung 3)*	Rückmeldung	Kopfnummer	Daten vom Codeträger	Abschluß 2)
Nächsten Codeträger suchen (einmal)	vom steuernden System zum BIS	**'H'**	'?'	BCC oder siehe 2)						
	vom BIS zum steuernden System				<ACK>'0'	*'CR' oder 'LF CR'*	'H'	'1', '2' oder '?'	D1 D2 D3 D4	BCC oder siehe 2)
			1)							

1) Die Befehle Status und/oder Quit sind an dieser Stelle nicht zugelassen.
2) Statt Blockcheck BCC kann je nach Protokollvariante entweder Carriage Return 'CR' oder Line Feed mit Carriage Return 'LF CR' verwendet werden.
3) Bei Protokollvarianten, die immer eine Endekennung benötigen, muß hier eines der Abschlußzeichen 'CR' oder 'LF CR' eingefügt werden.

Telegrammbeispiel:
Nächsten Codeträger suchen (einmal)
mit Blockcheck (BCC)

-> Kopf 1 ist angewählt. Es befindet sich nur vor Schreib-/Lesekopf 2 ein Codeträger, dessen erste vier Byte mit 9876 beschrieben sind.

Das Steuersystem sendet	'H ?	w' BCC (77_{HEX})
Die BIS-Auswerteeinheit quittiert mit	<ACK> '0'	
und sendet die Daten	'H 2 9 8 7 6	z' BCC ($7A_{HEX}$)

Angaben in spitzen Klammern stellen ein Steuerzeichen dar.
Angaben in Hochkommata stellen das/die jeweilige/n Zeichen im ASCII-Code dar.

Nächsten Codeträger suchen (ständig)

Mit dem nachfolgend angegebenen Telegramm wird der nächste Codeträger gesucht. Dabei wird zum nächstfolgenden Schreib-/Lesekopf weitergeschaltet und geprüft, ob sich ein Codeträger vor diesem Schreib-/Lesekopf befindet. Wenn ja, enthält die Telegrammrückmeldung die zugehörige Nummer des Schreib-/Lesekopfs und die ersten 4 Byte des Codeträgers. Wenn nein, wird zum angewählten Kopf zurückgeschaltet und geprüft, ob sich ein Codeträger vor diesem Schreib-/Lesekopf befindet. Dies wiederholt sich so lange, bis ein Codeträger erkannt wird.
'H !' erkennt jeden Codeträger, unabhängig von der eingestellten Blockgröße, vorausgesetzt, Schreib-/Lesekopf und Codeträger sind kompatibel.

Task	Datenfluß	Befehl	Kennung	Abschluß 2)	Quittung	Endekennung 3)	Rückmeldung	Kopfnummer	Daten vom Codeträger	Abschluß 2)
Nächsten Codeträger suchen (ständig)	vom steuernden System zum BIS	'H'	'!'	BCC oder siehe 2)						
	vom BIS zum steuernden System				<ACK>'0'	'CR' oder 'LF CR'	'H'	'1' oder '2'	D1 D2 D3 D4	BCC oder siehe 2)
			1)							

1) Die Befehle Status und/oder Quit sind an dieser Stelle nicht zugelassen.
2) Statt Blockcheck BCC kann je nach Protokollvariante entweder Carriage Return 'CR' oder Line Feed mit Carriage Return 'LF CR' verwendet werden.
3) Bei Protokollvarianten, die immer eine Endekennung benötigen, muß hier eines der Abschlußzeichen 'CR' oder 'LF CR' eingefügt werden.

Telegrammbeispiel:
Nächsten Codeträger suchen (ständig)
mit Blockcheck (BCC)

-> Es befindet sich vor Schreib-/Lesekopf 2 ein Codeträger, dessen erste vier Byte mit 9876 beschrieben sind.

Das Steuersystem sendet	'H !	i' BCC (69_{HEX})
Die BIS-Auswerteeinheit quittiert mit	<ACK> '0'	
und sendet die Daten	'H 2 9 8 7 6	z' BCC ($7A_{HEX}$)

Angaben in spitzen Klammern stellen ein Steuerzeichen dar.
Angaben in Hochkommata stellen das/die jeweilige/n Zeichen im ASCII-Code dar.

Bearbeiten der Ausgänge

Durch das Absenden des Telegramms können die vier Ausgänge gesetzt oder gelöscht werden.

Task	Datenfluß	Befehl	Kennung	Abschluß 2)	Quittung	Ende-kennung 3)
Ausgänge bearbeiten (setzen oder löschen)	vom steuernden System zum BIS	**'B'**	'00' bis 'A1' (siehe unten)	BCC oder siehe 2)		
	vom BIS zum steuernden System				<ACK>'0'	'CR' oder 'LF CR'
			1)			

Bedeutung der Kennung:

	0	1	2	3	alle Ausgänge
Ausgang Nr.					
Ausgang löschen	00	10	20	30	A0
Ausgang setzen	01	11	21	31	A1

1) Die Befehle Status und/oder Quit sind an dieser Stelle nicht zugelassen.
2) Statt Blockcheck BCC kann je nach Protokollvariante entweder Carriage Return 'CR' oder Line Feed mit Carriage Return 'LF CR' verwendet werden.
3) Bei Protokollvarianten, die immer eine Endekennung benötigen, muß hier eines der Abschlußzeichen 'CR' oder 'LF CR' eingefügt werden.

Telegrammbeispiel:
Bearbeiten der Ausgänge
mit Blockcheck (BCC)

Das Steuersystem sendet 'B 21 A' BCC (41HEX)
Die BIS-Auswerteeinheit quittiert mit <ACK> '0'

Nach Ablauf des Telegramms ist Ausgang 2 gesetzt.

Angaben in spitzen Klammern stellen ein Steuerzeichen dar.
Angaben in Hochkommata stellen das/die jeweilige/n Zeichen im ASCII-Code dar.

Alle Ausgänge ausgeben

Durch das Absenden des Telegramms können die Zustände aller vier Ausgänge ausgegeben werden.

Task	Datenfluß	Befehl	Kennung	Abschluß 2)	Quittung	Zustand der vier Ausgänge	Ende-kennung 3)	Abschluß 2)
alle Ausgänge ausgeben	vom steuernden System zum BIS	**'B'**	'AO'	BCC oder siehe 2)				
	vom BIS zum steuernden System				<ACK>'0'	'XXXX' '0' =gelöscht, '1' = gesetzt	'CR' oder 'LF CR'	BCC oder siehe 2)
			1)					

Die Ausgabe des Zustands erfolgt in der Reihenfolge der Ausgangs-Nr. 0 1 2 3

1) Die Befehle Status und/oder Quit sind an dieser Stelle nicht zugelassen.
2) Statt Blockcheck BCC kann je nach Protokollvariante entweder Carriage Return 'CR' oder Line Feed mit Carriage Return 'LF CR' verwendet werden.
3) Bei Protokollvarianten, die immer eine Endekennung benötigen, muß hier eines der Abschlußzeichen 'CR' oder 'LF CR' eingefügt werden.

Telegrammbeispiel:
Ausgeben aller Ausgänge
mit Blockcheck (BCC)

-> Die Ausgänge 0 und 1 sind gesetzt, die Ausgänge 2 und 3 gelöscht.

Das Steuersystem sendet	'BAO	L'	BCC ($4C_{HEX}$)
Die BIS-Auswerteeinheit quittiert mit	<ACK> '0'		
und sendet die Daten	'1100	NUL'	BCC (00_{HEX})

Angaben in spitzen Klammern stellen ein Steuerzeichen dar.
Angaben in Hochkommata stellen das/die jeweilige/n Zeichen im ASCII-Code dar.

Neustart der Auswerteeinheit (Quit)

Durch das Absenden des Telegramms Neustart wird ein in Arbeit befindliches Telegramm abgebrochen und die Auswerteeinheit in den Grundzustand gebracht.
Nach der Quittierung dieses Telegramms sind mindestens 1600 ms Pause vorzusehen, bevor ein neues Telegramm gestartet wird.

Wichtig! Der Befehl Quit ist nicht zugelassen, während die Auswerteeinheit auf ein Abschlußzeichen wartet (BCC, 'CR' oder 'LF CR'). In dieser Situation würde Quit als Abschluß- oder Nutzzeichen fehlinterpretiert.

Task	Datenfluß	Befehl	Abschluß 2)	Quittung	Abschluß 2)
Neustart (Quit)	vom steuernden System zum BIS	**'Q'**	BCC oder siehe 2)		
	vom BIS zum steuernden System			'Q'	BCC oder siehe 2)
			1)		

1) Die Befehle Status und/oder Quit sind an dieser Stelle nicht zugelassen.
2) Statt Blockcheck BCC kann je nach Protokollvariante entweder Carriage Return 'CR' oder Line Feed mit Carriage Return 'LF CR' verwendet werden.

Telegrammbeispiel mit Blockcheck (BCC):

Das System BIS soll in den Grundzustand gebracht werden.

Das Steuersystem sendet 'Q Q' BCC (51_{HEX})

Die BIS-Auswerteeinheit quittiert mit 'Q Q' BCC (51_{HEX})

Angaben in spitzen Klammern stellen ein Steuerzeichen dar.
Angaben in Hochkommata stellen das/die jeweilige/n Zeichen im ASCII-Code dar.

Abfrage der Statusmeldung

Mit dem Statustelegramm wird abgefragt, was für ein Telegramm sich in Arbeit befindet.

Wichtig! Der Befehl Status ist nicht zugelassen, während die Auswerteeinheit auf ein Abschlußzeichen wartet (BCC, 'CR' oder 'LF CR'). In dieser Situation würde Status als Abschluß- oder Nutzzeichen fehlinterpretiert.

Wichtig: Eine Statusabfrage während eines Lese- oder Schreibzugriffs auf einen Codeträger (LED Codetag Operating leuchtet) verlängert die Lese- oder Schreibzeit. Dies kann besonders beim dynamischen Betrieb dazu führen, daß die Zeit, in der sich der Codeträger im Arbeitsbereich des Schreib-/Lesekopfs befindet, zum vollständigen Lesen oder Schreiben nicht mehr ausreicht. Die dauernde Statusabfrage stört die Bearbeitung des Codeträgers; evtl. wird der Codeträger nicht erkannt!

Die Angaben zwischen den Hochkommata stellen das/die jeweilige/n Zeichen im ASCII-Code dar. '_' = Leertaste (Space) = ASCII-Zeichen 20HEX.

Task	Datenfluß	Befehl	Abschluß 2)	Statusmeldung	Abschluß 2)
Abfrage der Statusmeldung	vom steuernden System zum BIS	**'S'**	BCC oder siehe 2)		
	vom BIS zum steuernden System			'S' '_', 'R', 'W', 'L', 'P' oder 'H'	BCC oder siehe 2)
			1)		

1) Die Befehle Status und/oder Quit sind an dieser Stelle nicht zugelassen.
2) Statt Blockcheck BCC kann je nach Protokollvariante entweder Carriage Return 'CR' oder Line Feed mit Carriage Return 'LF CR' verwendet werden.

Statusmeldungen und ihre Bedeutung:

'S L'	=	Lesen vom Codeträger mit Anwahl des Schreib-/Lesekopfs und der Blockgröße des Codeträgers
'S P'	=	Schreiben auf den Codeträger mit Anwahl des Schreib-/Lesekopfs und der Blockgröße des Codeträgers
'S R'	=	Lesen vom Codeträger
'S W'	=	Schreiben auf den Codeträger
'S H'	=	Anwahl des Schreib-/Lesekopfs und der Blockgröße des Codeträgers
'S _'	=	kein Telegramm in Arbeit

Telegrammbeispiele zu Seite 64:
Abfrage der Statusmeldung
mit Blockcheck (BCC)

-> Es soll der Status im BIS abgefragt werden, nachdem kurz zuvor ein **Lesetelegramm** abgesandt worden war.

Das Steuersystem sendet	'S	S'	BCC (53_{HEX})
Die BIS-Auswerteeinheit quittiert mit	'S L	US'	BCC ($1F_{HEX}$)

-> Es soll der Status im BIS abgefragt werden, nachdem kurz zuvor ein **Schreibtelegramm** abgesandt worden war.

Das Steuersystem sendet	'S	S'	BCC (53_{HEX})
Die BIS-Auswerteeinheit quittiert mit	'S P	ETX'	BCC (03_{HEX})

-> Es soll der Status im BIS abgefragt werden, nachdem kurz zuvor ein **Telegramm zur Umschaltung des Schreib-/Lesekopfs** abgesandt worden war.

Das Steuersystem sendet	'S	S'	BCC (53_{HEX})
Die BIS-Auswerteeinheit quittiert mit	'S H	ESC'	BCC ($1B_{HEX}$)

-> Es soll der Status im BIS abgefragt werden. Zuvor wurde **kein Telegramm** abgesandt.

Das Steuersystem sendet	'S	S'	BCC (53_{HEX})
Die BIS-Auswerteeinheit quittiert mit	'S	_'	BCC (20_{HEX})

Fehlernummern

BIS C-6_0 gibt immer eine Fehlernummer aus. Deren Bedeutung zeigt nachfolgende Tabelle.

Nr.	Fehlerbeschreibung	Auswirkung
1	Kein Codeträger vorhanden	Telegrammabbruch, Auswerteeinheit geht in den Grundzustand.
2	Fehler beim Lesen	Lesetelegrammabbruch, Auswerteeinheit geht in den Grundzustand.
3	Lesen abgebrochen, da der Codeträger entfernt wurde.	Auswerteeinheit geht in den Grundzustand.
4	Fehler beim Schreiben	Schreibtelegrammabbruch, Auswerteeinheit geht in den Grundzustand. BEACHTE: Es könnten bereits neue Daten auf den Codeträger geschrieben worden sein!
5	Schreiben abgebrochen, da der Codeträger entfernt wurde.	Auswerteeinheit geht in den Grundzustand. ACHTUNG: Es könnten bereits neue Daten auf den Codeträger geschrieben worden sein!
6	Fehler auf der Schnittstelle	Auswerteeinheit geht in den Grundzustand. (Paritäts- oder Stoppbitfehler)
7	Telegramm-Formatfehler	Auswerteeinheit geht in den Grundzustand. Mögliche Formatfehler: - Befehl ist kein 'L'/'P'/'C'/'R'/'W'/'H'/'B'/'Q' oder 'S'. - Startadresse oder Anzahl der Bytes außerhalb des zugelassenen Bereichs.

Fehlernummern
(Fortsetzung)

8	BCC-Fehler, der übertragene BCC ist falsch.	Telegrammabbruch, Auswerteeinheit geht in den Grundzustand.
9	Kabelbruch, Codetag Present LED blinkt.	Telegrammabbruch, Auswerteeinheit geht in den Grundzustand. Kabelbruch zum angewählten Schreib-/Lesekopf oder nicht angeschlossen. Wurden beide Schreib-/Leseköpfe über den Befehl 'HT' angewählt, könnte ein Kopf nicht angeschlossen sein. Sind beide Schreib-/Leseköpfe angewählt, wird die Kabelbruchmeldung nur angezeigt, wenn sich vor dem angeschlossenen, nicht defekten Kopf kein Codeträger befindet.
A	Neuer Befehl nicht möglich, da bereits ein Lesebefehl in Arbeit ist.	Nach Fehlermeldung wird Lesebefehl intern beendet, aber nicht mehr quittiert. Auswerteeinheit geht in den Grundzustand.
B	Neuer Befehl nicht möglich, da bereits ein Schreibbefehl in Arbeit ist.	Nach Fehlermeldung wird Schreibbefehl intern beendet, aber nicht mehr quittiert. Auswerteeinheit geht in den Grundzustand. ACHTUNG: Treten bei erneutem Schreiben auf den Codeträger weitere Fehler auf, folgen keine weiteren Fehlermeldungen.
C	Neuer Befehl nicht möglich, da bereits eine Kopfumschaltung in Arbeit ist.	Nach Fehlermeldung wird nicht mehr positiv quittiert, obwohl die Kopfumschaltung ausgeführt wurde. Auswerteeinheit geht in den Grundzustand.

Schreib-/Lesezeiten

Lesezeiten im statischen Betrieb (Konfiguration: ohne Dynamikbetrieb)

Für zweimaliges Lesen und Vergleichen:

Codeträger mit 32 Byte je Block	
Anzahl Byte	Lesezeit [ms]
von 0 bis 31	110
für jeweils weitere angebrochene 32 Byte addieren Sie weitere	120
von 0 bis 255	= 950

Codeträger mit 64 Byte je Block	
Anzahl Byte	Lesezeit [ms]
von 0 bis 63	220
für jeweils weitere angebrochene 64 Byte addieren Sie weitere	230
von 0 bis 2047	= 7350

Schreibzeiten im statischen Betrieb (Konfiguration: ohne Dynamikbetrieb)

Inclusive Rücklesen und Vergleichen:

Codeträger mit 32 Byte je Block	
Anzahl Byte	Schreibzeit [ms]
von 0 bis 31	110 + n * 10
≥ 32	y * 120 + n * 10

Codeträger mit 64 Byte je Block	
Anzahl Byte	Schreibzeit [ms]
von 0 bis 63	220 + n * 10
≥ 64	y * 230 + n * 10

n = Anzahl der zusammenhängend zu schreibenden Bytes
y = Anzahl der zu bearbeitenden Blöcke

Beispiel:
Es sollen 17 Byte ab Adresse 187 geschrieben werden. Codeträger = 32 Byte je Block.
Bearbeitet werden Block 5 und 6, da Anfangsadresse 187 in Block 5 und Endadresse 203 in Block 6 ist.

t = 2 * 120 + 17 * 10 = 410

Die angegebenen Zeiten gelten, nachdem der Codeträger erkannt wurde. Andernfalls müssen für den Energieaufbau bis zum Erkennen des Codeträgers 45 ms hinzugerechnet werden.

Schreib-/Lesezeiten

Lesezeiten im dynamischen Betrieb (Konfiguration: mit Dynamikbetrieb)

Lesezeiten innerhalb des 1. Blocks für zweimaliges Lesen und Vergleichen:

Codeträger mit 32 Byte je Block		
Anzahl Byte		Lesezeit [ms]
von 0 bis 3	=	14
für jedes weitere Byte		3,5
von 0 bis 31	=	112

Codeträger mit 64 Byte je Block		
Anzahl Byte		Lesezeit [ms]
von 0 bis 3	=	14
für jedes weitere Byte		3,5
von 0 bis 63	=	224

m = größte zu lesende Adresse

Formel: $t = (m + 1) * 3{,}5$ ms

Beispiel: Es sollen 11 Byte ab Adresse 9 gelesen werden. D.h. die größte zu lesende Adresse ist 19. Dies ergibt 70 ms.

Schreibzeiten im dynamischen Betrieb (Konfiguration: mit Dynamikbetrieb)

Inklusive Rücklesen und Vergleichen:

Codeträger mit 32 Byte je Block		
Anzahl Byte		Schreibzeit [ms]
von 0 bis 3	=	14 + n * 10
für jedes weitere Byte		3,5

Codeträger mit 64 Byte je Block		
Anzahl Byte		Schreibzeit [ms]
von 0 bis 3	=	14 + n * 10
für jedes weitere Byte		3,5

n = Anzahl der zusammenhängend zu schreibenden Bytes

Die angegebenen Zeiten gelten, nachdem der Codeträger erkannt wurde. Andernfalls müssen für den Energieaufbau bis zum Erkennen des Codeträgers 45 ms hinzugerechnet werden.

LED-Anzeige

LED-Anzeige:
System Ready
Codetag Present
Codetag Operating

Die Auswerteeinheit BIS C-600 meldet die wichtigsten Betriebszustände über drei LED's auf der Gehäuseseite.

Betriebs-zustand	LED	Bedeutung
System Ready	an (grün)	Betriebsspannung in Ordnung; kein Hardwarefehler
	aus	Betriebsspannung oder Hardware nicht in Ordnung oder Kabelbruch zum angewählten Schreib-/Lesekopf oder nicht angeschlossen.
Codetag Present	an (gelb)	Codeträger schreib-/lesebereit. (Tritt während des Schreibens/ Lesens ein Schreib-/Lesefehler auf, erlischt System Ready, wenn Protokollvariante "ohne Fehlernummer" eingestellt ist!)
	blinkt	Kabelbruch zum angewählten Schreib-/Lesekopf oder nicht angeschlossen. Wurden beide Schreib-/Leseköpfe über den Befehl 'HT' angewählt, könnte ein Kopf nicht angeschlossen sein. Sind beide Schreib-/Leseköpfe angewählt, wird die Kabelbruch-meldung nur angezeigt, wenn sich vor dem angeschlossenen, nicht defekten Kopf kein Codeträger befindet.
	aus	Kein Codeträger im Schreib-/Lesebereich
Codetag Operating	an (gelb)	Befehl wird bearbeitet
	aus	Kein Befehl in Arbeit

Wenn alle drei LEDs gleichzeitig synchron blinken, muß die Auswerteeinheit zur Reparatur ins Werk.

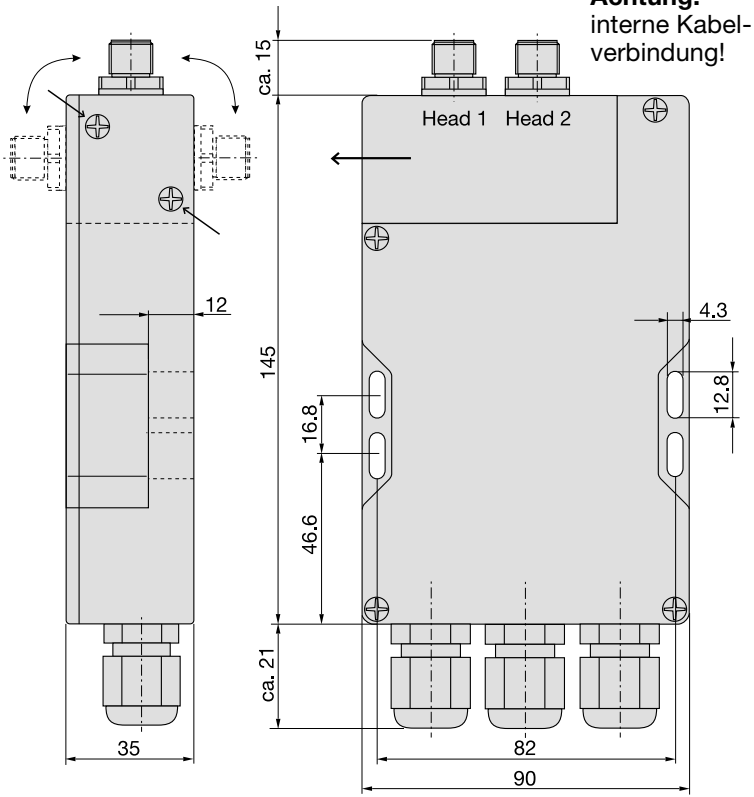
BIS C-600
Montage Auswerteeinheit / Kopf

Montage der Auswerteeinheit BIS C-600 und Anordnung des Schreib-/Lesekopfes bzw. des Adapters

Die Auswerteeinheit wird an den 4 seitlichen Langlöchern befestigt.

Je nach Ausführung ist die Auswerteeinheit mit einem Schreib-/Lesekopf oder dem Adapter für abgesetzte Schreib-/Leseköpfe ausgestattet. Sowohl der Schreib-/Lesekopf als auch der Adapter können vom Anwender durch Umsetzen um + oder –90° in die gewünschte Lage gebracht werden (siehe Bild). Sorgen Sie dafür, daß das Gerät spannungsfrei geschaltet ist. Öffnen Sie die beiden Schrauben (im Bild durch Pfeile gekennzeichnet). Ziehen Sie den Kopf bzw. den Adapter vorsichtig nach der Seite heraus (Pfeilrichtung, rechtes Bild). **Achtung: interne Kabelverbindung!** Montieren Sie ihn in der gewünschten Lage und schrauben Sie ihn wieder an.

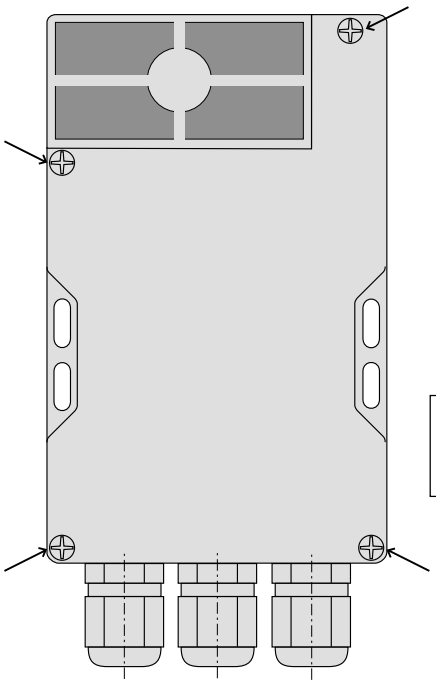
Achtung: interne Kabelverbindung!



Öffnen der Auswerteeinheit BIS C-600

Um die Verbindungen herzustellen, ist die Auswerteeinheit BIS C-600 zu öffnen.

Sorgen Sie dafür, daß das Gerät spannungsfrei geschaltet ist. Öffnen Sie die 4 Schrauben am BIS C-600 und entfernen Sie den Deckel. Führen Sie die Anschlußkabel durch die Klemmverschraubungen. Weitere Einzelheiten zur Verdrahtung siehe folgende Seiten.



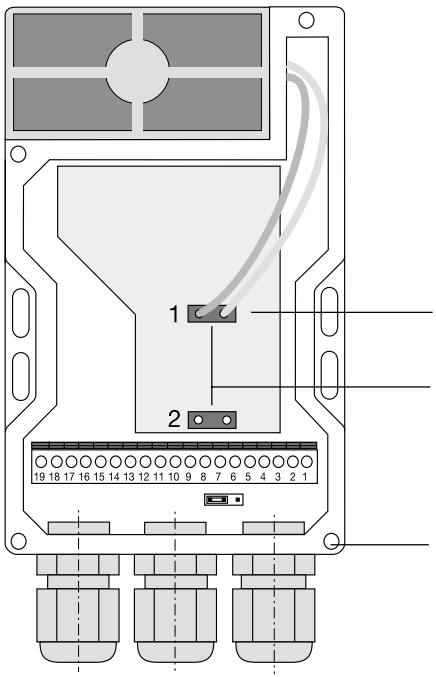
Befestigung des Deckels (4 Schrauben),
max. zulässiges Anzugsdrehmoment: 0,15 Nm

Versehen Sie die mitgelieferten Aufkleber mit Ihren Konfiguratuonsdaten und kleben Sie sie auf die Innenseite des Gerätedeckels.

Öffnen der Auswerteeinheit

Montage eines Schreib-/Lesekopfes bzw. Adapters BIS C-650/-670

Wenn Sie einen Schreib-/Lesekopf wechseln wollen: Auswerteeinheit spannungsfrei schalten und öffnen. Befestigungsschrauben des Schreib-/Lesekopfs lösen (siehe Seite 71) und Deckel der Auswerteeinheit abschrauben (siehe Seite 72). Lösen Sie die Steckverbindung des Schreib-/Lesekopfs von der Platine und ziehen Sie das Verbindungskabel durch den Kabelschacht heraus. Für die Montage des neuen Kopfs verfahren Sie in umgekehrter Reihenfolge.



Wenn Sie einen Adapter montieren wollen, verfahren Sie wie oben beschrieben. Bei BIS C-650 müssen beide Verbindungskabel auf der Platine gesteckt werden.

Achtung! Bei Verwendung des Adapters BIS C-650 oder BIS C-670 darf an der Klemmleiste kein weiterer Schreib-/Lesekopf angeschlossen werden.

Anschluß des integrierten Schreib-/Lesekopfs oder des Adapters BIS C-670

Anschlüsse für den Adapter BIS C-650
1 = Head 1
2 = Head 2

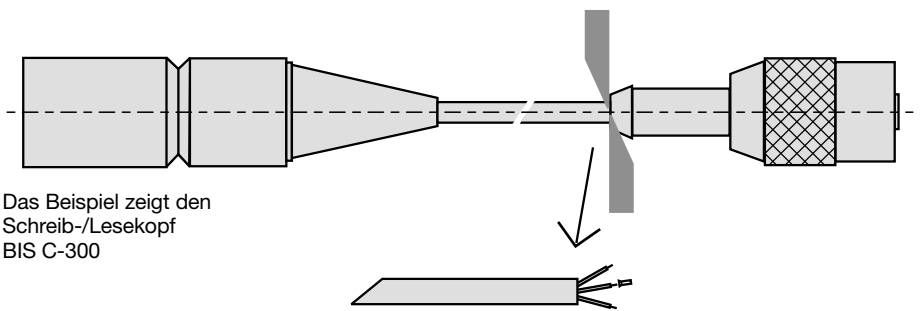
Befestigung des Deckels (4 Schrauben), max. zulässiges Anzugsdrehmoment: 0,15 Nm

BIS C-600
Montage externer Schreib-/Lesekopf

Schreib-/Lesekopf für den Anschluß an der Klemmleiste der Auswerteeinheit BIS C-600 vorbereiten

Um einen Schreib-/Lesekopf der Baureihe BIS C-3_ _ (ausgenommen BIS C-350 und -352) an der Klemmleiste der Auswerteeinheit BIS C-600 anschließen zu können, ist der am Kabel angebrachte Steckverbinder zu entfernen.

Bitte beachten Sie, daß das Kabel wie unten gezeigt direkt am Steckverbinder abgeschnitten werden muß, da die Länge des Kabels auf die Funktion des Schreib-/Lesekopfs abgestimmt ist. Bei veränderter Kabellänge können die Daten nicht mehr gewährleistet sein.
Das Kabel des Schreib-/Lesekopfs darf maximal 5 m lang sein.



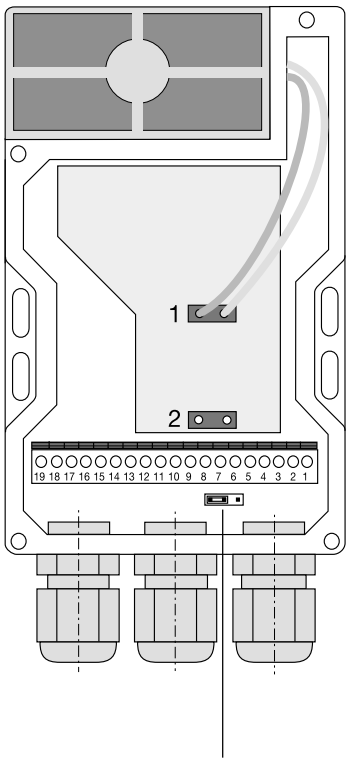
Das Beispiel zeigt den Schreib-/Lesekopf BIS C-300

Anschlußplan für den 2. Schreib-/Lesekopf an die Auswerteeinheit BIS C-600

Adern-farbe	Funktion	BIS C-60_ Klemmleiste
BU	AH	15
BN	EH	14
Schirm	GND	16

Der Kabelmantel ist auf eine Länge von ca. 5 cm zu entfernen. Die Kabeladern sind auf einer Länge von ca. 5 mm abzuisolieren und mit Aderendhülsen für 0,25 ... 0,34 mm² Querschnitt zu versehen.

Anschlußplan für Auswerteeinheiten BIS C-600 mit integriertem Schreib-/Lesekopf



Lage und Bezeichnung der Anschlüsse

Shunt-Stecker für die Handshake-Einstellung bei RS 232 (siehe Anschlußpläne auf den folgenden Seiten)

19	18	17	16	15	14
+VS	–VS	⏚	⏚	AH	EH
POWER			HEAD #2		

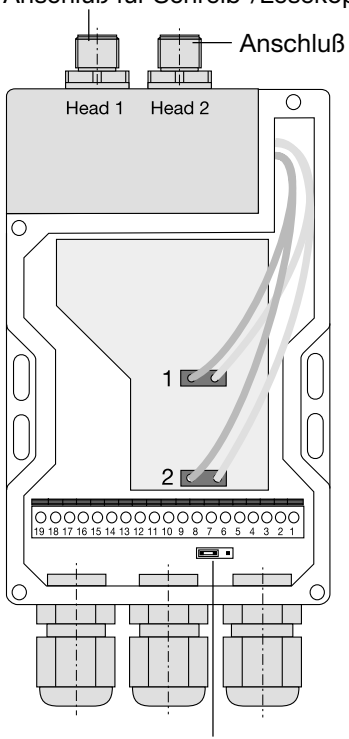
13	12	11	10	9	8	7	6
+VS	–VS	1	2	3	4	+IN	–IN
OUTPUT						INPUT	

5	4	3	2	1	
COM	RxD	CTS	TxD	RTS	
RS 232					BIS C-600...00

5	4	3	2	1	
n.c.	RxD+	RxD-	TxD+	TxD-	
	20 mA (TTY)				BIS C-600...01

Belegung der Klemmleiste
n.c. = nichts anschließen!

Anschlußplan für Auswerteeinheiten BIS C-601 mit Adapter BIS C-650



Klemmleiste

Shunt-Stecker für die Handshake-Einstellung bei RS 232 (siehe Anschlußpläne auf den folgenden Seiten)

Lage und Bezeichnung der Anschlüsse

19	18	17	16...14
+VS	–VS	⏚	n.c.
POWER			

13	12	11	10	9	8	7	6
+VS	–VS	1	2	3	4	+IN	–IN
OUTPUT						INPUT	

5	4	3	2	1
COM	RxD	CTS	TxD	RTS
RS 232				

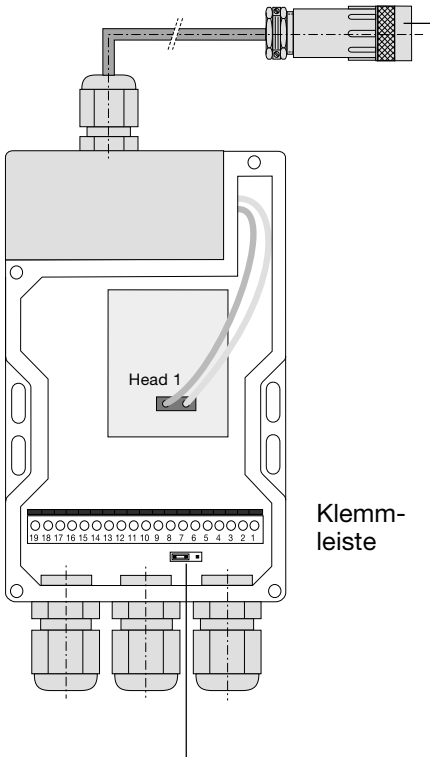
BIS C-600...00

5	4	3	2	1
n.c.	RxD+	RxD-	TxD+	TxD-
	20 mA (TTY)			

BIS C-600...01

Belegung der Klemmleiste
n.c. = nichts anschließen!

Anschlußplan für Auswerteeinheiten BIS C-600 mit Adapter BIS C-670



Lage und Bezeichnung der Anschlüsse

Shunt-Stecker für die Handshake-Einstellung bei RS 232 (siehe Anschlußpläne auf den folgenden Seiten)

19	18	17	16...14
+VS	–VS	⏚	n.c.
POWER			

13	12	11	10	9	8	7	6
+VS	–VS	1	2	3	4	+IN	–IN
OUTPUT						INPUT	

5	4	3	2	1	
COM	RxD	CTS	TxD	RTS	
RS 232					BIS C-600...00

5	4	3	2	1	
n.c.	RxD+	RxD-	TxD+	TxD-	
	20 mA (TTY)				BIS C-600...01

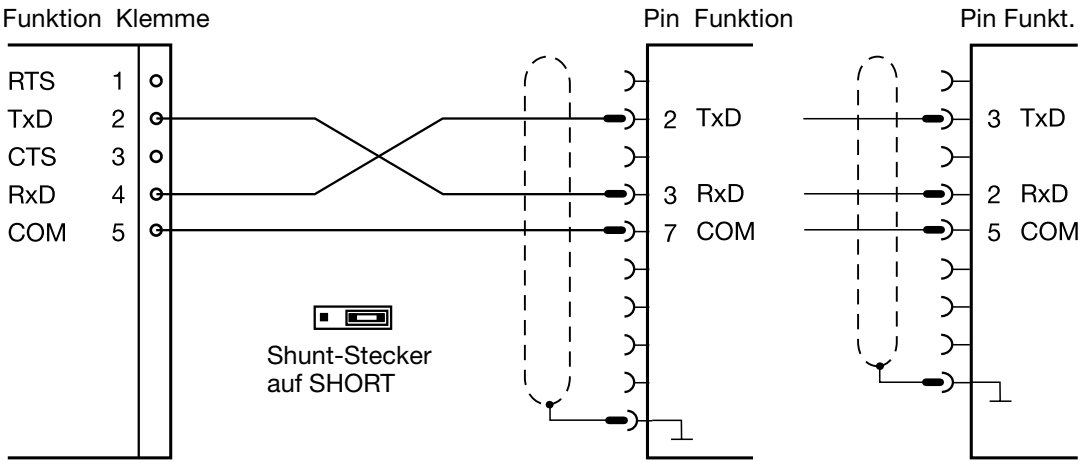
Belegung der Klemmleiste
n.c. = nichts anschließen!

Funktion der Schnittstellen

Zum Anschluß der Auswerteeinheit BIS C-600 an ein steuerndes System (z.B. PC oder SPS) ist eine serielle Schnittstelle vorgesehen. Je nach Ausführung stehen zur Verfügung:

BIS C-600...00 die Schnittstelle RS 232 (V.24) oder
BIS C-600...01 die 20-mA-Stromschnittstelle (TTY).

Schnittstelle RS 232 (V.24) ohne Hardware-Handshake



Anschluß an Klemmleiste der Auswerteeinheit

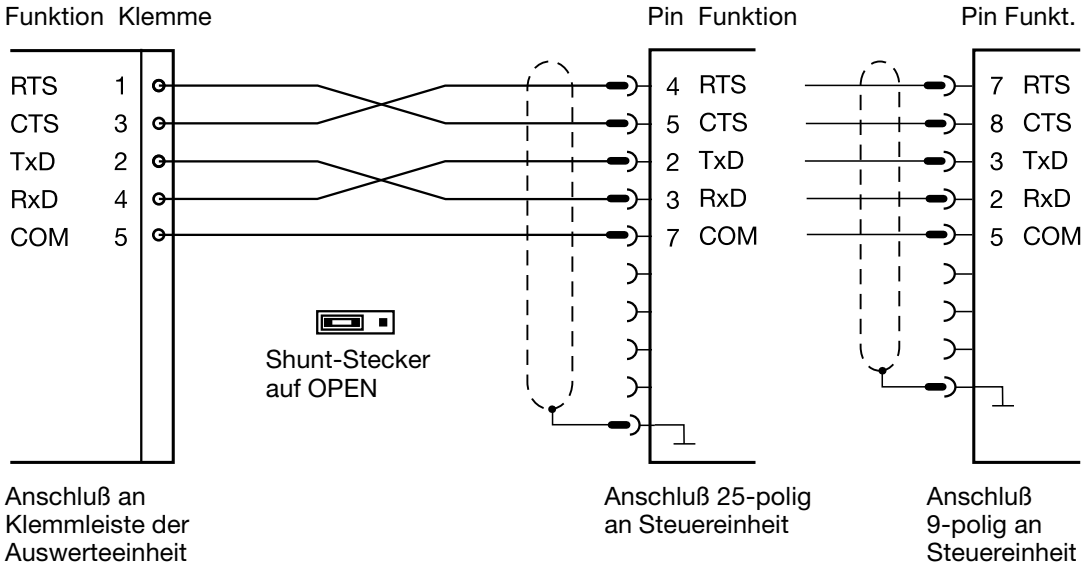
Anschluß 25-polig an Steuereinheit

Anschluß 9-polig an Steuereinheit

OPEN / SHORT

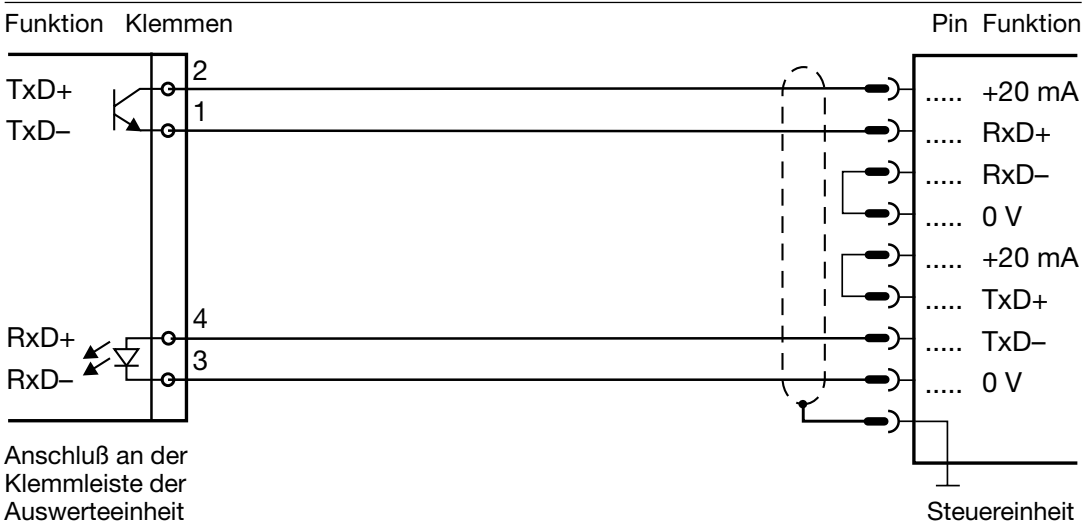
CTS  Werkseinstellung = SHORT.
Da das Steuersignal CTS nicht verwendet wird, bleibt der Shunt-Stecker in Position SHORT.

Schnittstelle RS 232 (V.24) mit Hardware-Handshake



OPEN / SHORT
CTS Werkseinstellung = SHORT.
Da das Steuersignal CTS verwendet wird,
ist der Shunt-Stecker in Position OPEN gesteckt.

20-mA-Strom-schnittstelle (TTY), Auswerteeinheit passiv



Sie können die Pin-Nummern für die Belegung des Anschlusses an Ihrer Steuereinheit in die Zeichnung eintragen.

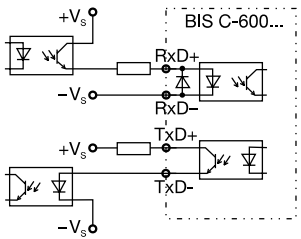
BIS C-600
Technische Daten

Abmessungen, Gewicht	**Gehäuse**	Kunststoff PS
	Abmessungen mit Schreib-/Lesekopf BIS C-652	ca. 169 x 90 x 35 mm
	Abmessungen mit Adapter BIS C-650	ca. 184 x 90 x 35 mm
	Gewicht	ca. 400 g
Temperaturbereich	Umgebungstemperatur	0 °C bis +60 °C
Anschlüsse	Klemmleiste	19-polig
	Kabeleinführung	3 x Klemmkorb PG 9
	Kabeldurchmesser	4 bis 8 mm
	Leitergrößen	0,14 bis 1 mm^2
	mit Adernhülsen	0,25 bis 0,34 mm^2
Schutzart	Schutzart	IP 65 (in angeschlossenem Zustand)
Elektrische Anschlüsse	**Betriebsspannung V_S, Eingang**	DC 24 V ± 20 %
	Restwelligkeit	≤ 10 %
	Stromaufnahme	≤ 400 mA
	Schreib-/Lesekopf	integriert, BIS C-65_ und folgende*);
	alternativ bei montiertem Adapter BIS C-650 *)	2 x Einbaustecker 4-polig (Stift) für alle Schreib-/Leseköpfe BIS C-3_ _ mit 4-poligem Stecker (Buchse), nicht BIS C-350 und BIS C-352
	alternativ bei montiertem Adapter BIS C-670 *)	1 x Anschlußstecker 8-polig (Stift) für einen der Schreib-/Leseköpfe BIS C-350 und BIS C-352
	Serielle Schnittstelle	RS 232 Schnittstelle (V.24) oder 20-mA-Stromschnittstelle (TTY)

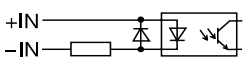
*) um ± 90° umsetzbar

Elektrische Anschlüsse (Fortsetzung)

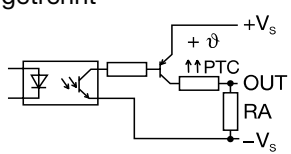
20-mA-Stromschnittstelle (TTY)	über Optokoppler galvanisch getrennt
Empfänger	Current loop, 20 mA passiv
Leitungsstrom für Ruhezustand	20 bis 50 mA
Spannungsabfall bei 20 mA	ca. 3 V
Sender	Current loop, 20 mA passiv
Leerlaufspannung V_S	max. 50 V
Spannungsabfall bei 20 mA	ca. 3 V
Leitungsstrom	max. 50 mA



Digitaler Eingang (+IN, –IN)	über Optokoppler galvanisch getrennt
Steuerspannung aktiv	4 V bis 40 V
Steuerspannung inaktiv	1,5 V bis –40 V
Eingangsstrom bei 24 V	11 mA
Verzögerungszeit typisch	5 ms



Steuerausgänge (01 bis 04)	über Optokoppler galvanisch getrennt
Ausgangsstrom	max. 20 mA
Spannungsabfall bei 20 mA	ca. 2,5 V
Ausgangswiderstand R_A	10 kΩ gegen V_S
Ausgangsschaltung	PNP (plusschaltend)
Betriebsspannung, Ausgang V_S	DC 24 V ± 20 %
Restwelligkeit	≤ 10 %



Funktionsanzeigen	
System Ready	LED grün
Codetag Present	LED gelb
Codetag Operating	LED gelb

Mit dem CE-Zeichen bestätigen wir, daß unsere Produkte den Anforderungen der EG-Richtlinie

89/336/EWG (EMV-Richtlinie)

und des EMV-Gesetzes entsprechen. In unserem EMV-Labor, das von der DATech für Prüfungen der elektromagnetischen Verträglichkeit akkreditiert ist, wurde der Nachweis erbracht, daß die Balluff-Produkte die EMV-Anforderungen der Fachgrundnorm

EN 50081-2 (Emission), EN 50082-2 (Störfestigkeit) erfüllen.

BIS C-600
Bestellinformationen

Typenschlüssel

BIS C-600-___-___-__-KL1

Balluff Identifikations-System

Baureihe C

Hardware-Typ
600 = Kunststoffgehäuse

Software-Typ
022 = Server-Software
007 = Balluff Standard-Protokoll

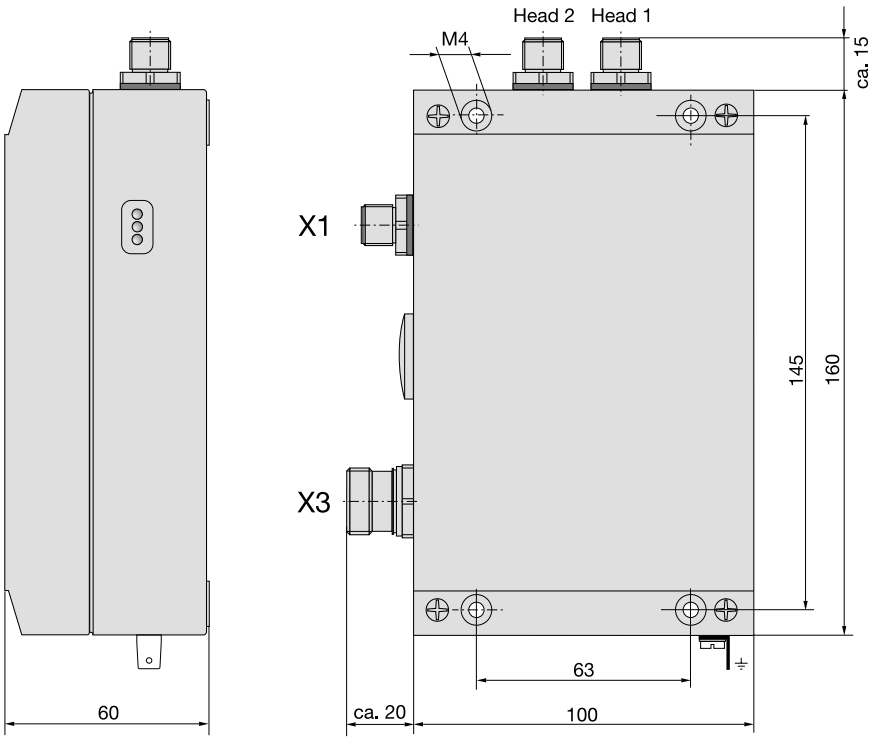
Schreib-/Lesekopf
000 = kein Schreib-/Lesekopf
651 = Schreib-/Lesekopf BIS C-651 (Rundantenne stirnseitig)
652 = Schreib-/Lesekopf BIS C-652 (Rundantenne frontseitig)
653 = Schreib-/Lesekopf BIS C-653 (Stabantenne)
650 = Adapter mit zwei Anschlüssen für externe Schreib-/Leseköpfe BIS C-3_ _ (ausgenommen BIS C-350 und -352)
670 = Adapter mit einem abgesetztem Anschlußstecker für einen externen Schreib-/Lesekopf BIS C-350 oder BIS C-352

Schnittstelle
00 = RS 232-Schnittstelle (V.24)
01 = 20-mA-Stromschnittstelle (TTY)

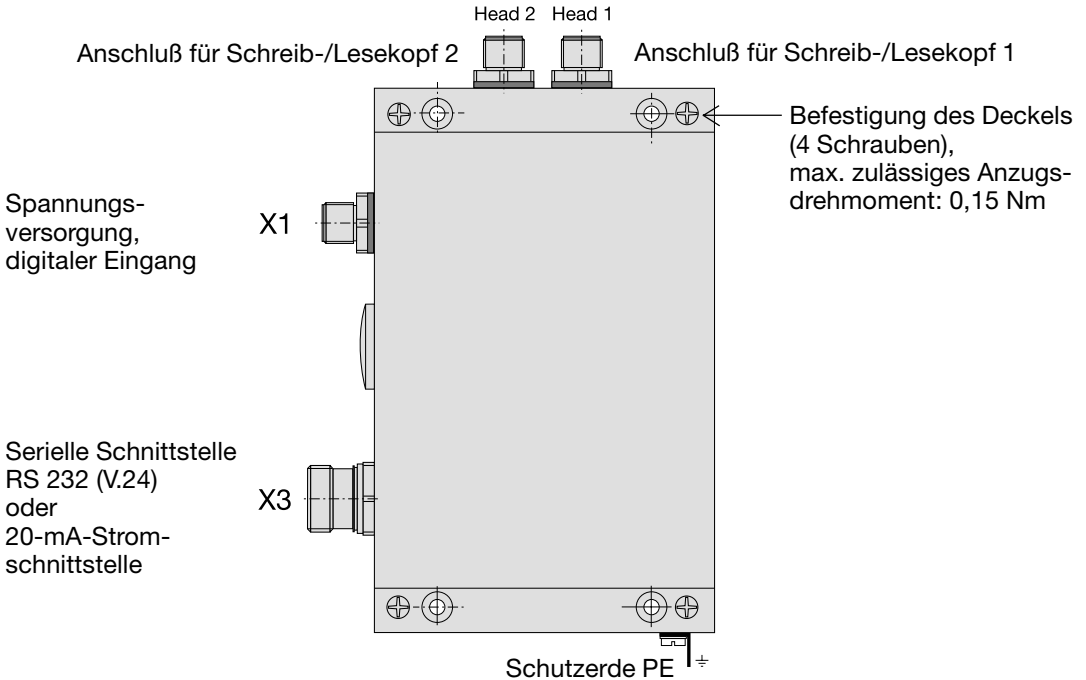
Kundenanschluß
KL1 = Klemmanschluß über 3 x PG9

Montage der Auswerteeinheit BIS C-620

Die Auswerteeinheit wird mit 4 Schrauben M4 befestigt.



Anschlußplan für Auswerteeinheit BIS C-620



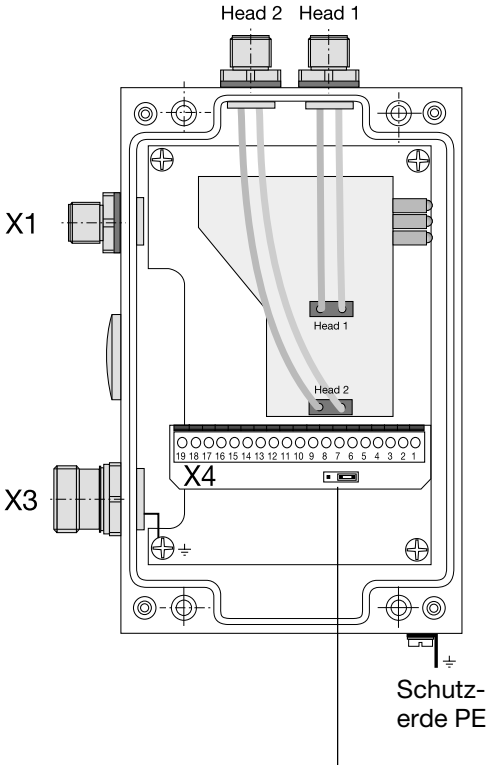
Lage und Bezeichnung der Anschlüsse

Öffnen der Auswerteeinheit BIS C-620

Um den Shunt-Stecker für die Handshake-Einstellung bei RS 232 zu stecken, ist die Auswerteeinheit BIS C-620 zu öffnen.

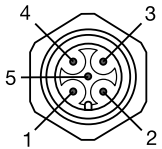
Sorgen Sie dafür, daß das Gerät spannungsfrei geschaltet ist. Öffnen Sie die 4 Schrauben am BIS C-620 und entfernen Sie den Deckel. Weitere Einzelheiten siehe folgende Seiten.

Anschlußplan für Auswerteeinheit BIS C-620



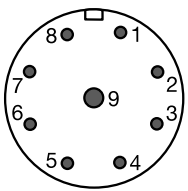
Shunt-Stecker für die Handshake-Einstellung bei RS 232 (siehe Anschlußpläne auf den folgenden Seiten)

X1, Stromversorgung und digitaler Eingang



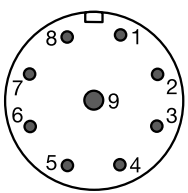
Pin	Funktion
1	+Vs
2	–IN
3	–Vs
4	+IN
5	n.c.

X3, RS 232
BIS C-620...00



Pin	Funktion
1	n.c.
2	RxD
3	TxD
4	n.c.
5	COM
6	n.c.
7	RTS
8	CTS
9	n.c.

X3, 20-mA-Strom-schnittstelle
BIS C-620...01



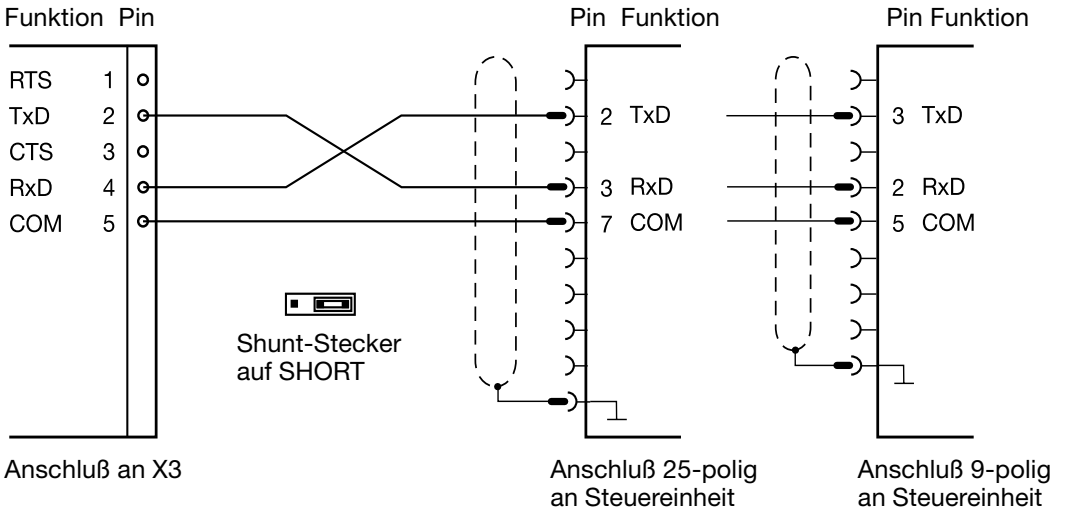
Pin	Funktion
1	TxD–
2	n.c.
3	n.c.
4	TxD+
5	n.c.
6	RxD–
7	n.c.
8	n.c.
9	RxD+

n.c. = nichts anschließen

Funktion der Schnittstellen

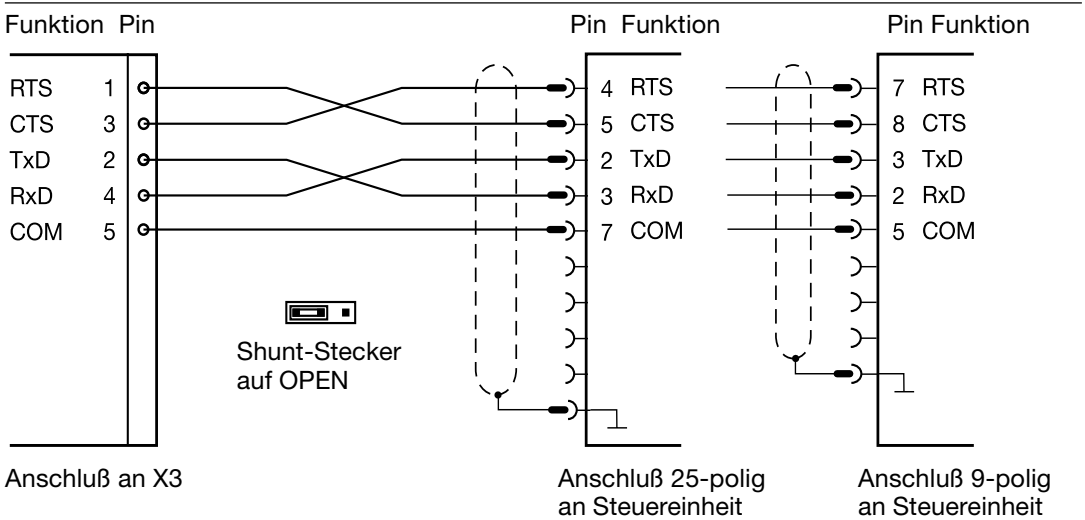
Zum Anschluß der Auswerteeinheit BIS C-620 an ein steuerndes System (z.B. PC oder SPS) ist eine serielle Schnittstelle vorgesehen. Je nach Ausführung stehen zur Verfügung:
BIS C-620...00 die Schnittstelle RS 232 (V.24) oder
BIS C-620...01 die 20-mA-Stromschnittstelle (TTY).

Schnittstelle RS 232 (V.24) ohne Hardware-Handshake



OPEN / SHORT
CTS Werkseinstellung = SHORT.
Da das Steuersignal CTS nicht verwendet wird, bleibt der Shunt-Stecker in Position SHORT.

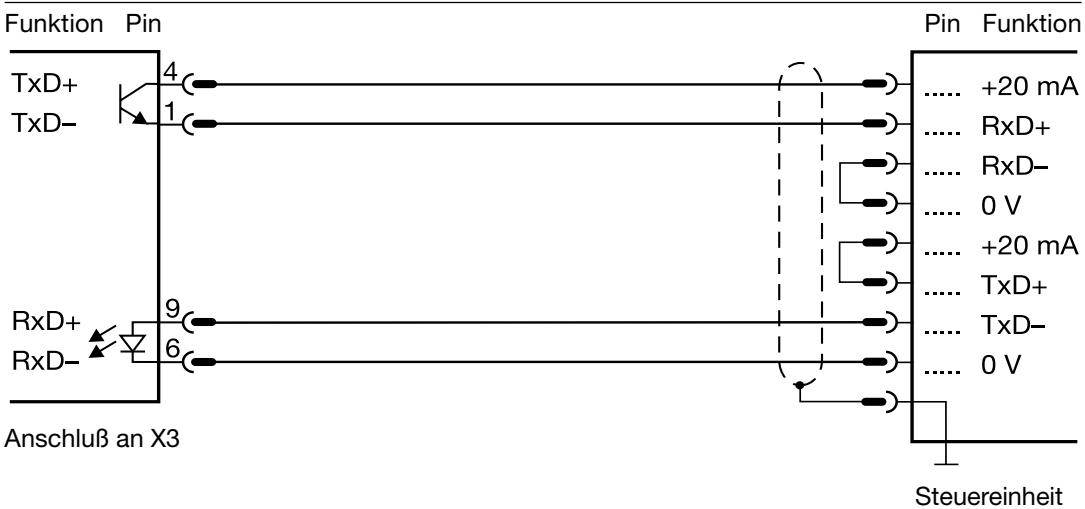
Schnittstelle RS 232 (V.24) mit Hardware-Handshake



OPEN / SHORT

CTS Werkseinstellung = SHORT.
Da das Steuersignal CTS verwendet wird,
ist der Shunt-Stecker in Position OPEN gesteckt.

20-mA-Stromschnittstelle (TTY), Auswerteeinheit passiv



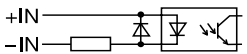
Sie können die Pin-Nummern für die Belegung des Anschlusses an Ihrer Steuereinheit in die Zeichnung eintragen.

BIS C-620
Technische Daten

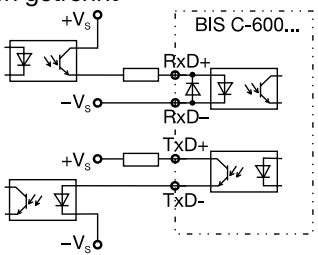
Abmessungen, Gewicht	**Gehäuse**	Metall
	Abmessungen	175 x 120 x 60 mm
	Gewicht	800 g
Temperaturbereich	Umgebungstemperatur	0 °C bis +60 °C
Anschlüsse	Einbaustecker X1	5-polig (Stift)
	Einbaustecker Head 1, Head 2	4-polig (Stift)
	Rundsteckverbinder X3	9-polig (Stift)
Schutzart	Schutzart	IP 65 (in angeschlossenem Zustand)
Elektrische Anschlüsse	**Eingang X1, Betriebsspannung V_S**	DC 24 V ± 20 %
	Restwelligkeit	≤ 10 %
	Stromaufnahme	≤ 400 mA
	Anschluß X3, serielle Schnittstelle	RS 232 (V.24) oder 20-mA-Stromschnittstelle (TTY)
	Anschlüsse Head 1, Head 2, Schreib-/Lesekopf	2 x Einbaustecker 4-polig (Stift) für alle Schreib-/Leseköpfe BIS C-3_ _ mit 4-poligem Stecker (Buchse), nicht BIS C-350 und BIS C-352

Elektrische Anschlüsse

Digitaler Eingang (+IN, –IN)	über Optokoppler galvanisch getrennt
Steuerspannung aktiv	4 V bis 40 V
Steuerspannung inaktiv	1,5 V bis –40 V
Eingangsstrom bei 24 V	11 mA
Verzögerungszeit typisch	5 ms



Serielle Schnittstelle (TTY)	über Optokoppler galvanisch getrennt
Empfänger	Current loop, 20 mA passiv
Leitungsstrom für Ruhezustand	20 bis 50 mA
Spannungsabfall bei 20 mA	ca. 3 V
Sender	Current loop, 20 mA passiv
Leerlaufspannung V_S	max. 50 V
Spannungsabfall bei 20 mA	ca. 3 V
Leitungsstrom	max. 50 mA



Funktionsanzeigen

System Ready	LED grün
Codetag Present	LED gelb
Codetag Operating	LED gelb

Mit dem CE-Zeichen bestätigen wir, daß unsere Produkte den Anforderungen der EG-Richtlinie

89/336/EWG (EMV-Richtlinie)

und des EMV-Gesetzes entsprechen. In unserem EMV-Labor, das von der DATech für Prüfungen der elektromagnetischen Verträglichkeit akkreditiert ist, wurde der Nachweis erbracht, daß die Balluff-Produkte die EMV-Anforderungen der Fachgrundnorm

EN 50081-2 (Emission), EN 50082-2 (Störfestigkeit) erfüllen.

BIS C-620
Bestellinformationen

Typenschlüssel

BIS C-620-___-050-__-ST2

Balluff Identifikations-System

Baureihe C

Hardware-Typ
620 = Metallgehäuse

Software-Typ
022 = Server-Software
007 = Balluff Standard-Protokoll

Schreib-/Lesekopf
050 = mit zwei Anschlußsteckern für externe Schreib-/Leseköpfe
BIS C-3_ _ (ausgenommen BIS C-350 und -352)

Schnittstelle
00 = RS 232-Schnittstelle (V.24)
01 = 20-mA-Stromschnittstelle (TTY)

Kundenanschluß
ST2 = Steckervariante
1 Rundsteckverbinder für Stromversorgung,
1 Rundsteckverbinder für RS 232 (V.24) / 20-mA-Stromschnittstelle (TTY)

Zubehör
(optional, nicht im Lieferumfang)

Artikel		Bestellbezeichnung
Gegenstecker	zu X1	BKS-S 79-00
	zu X3	BKS-S 84-00
Verschlußkappe	zu Head 1 / Head 2	BES 12-SM-2

Anhang, ASCII-Tabelle

Decimal	Hex	Control Code	ASCII
0	00	Ctrl @	NUL
1	01	Ctrl A	SOH
2	02	Ctrl B	STX
3	03	Ctrl C	ETX
4	04	Ctrl D	EOT
5	05	Ctrl E	ENQ
6	06	Ctrl F	ACK
7	07	Ctrl G	BEL
8	08	Ctrl H	BS
9	09	Ctrl I	HT
10	0A	Ctrl J	LF
11	0B	Ctrl K	VT
12	0C	Ctrl L	FF
13	0D	Ctrl M	CR
14	0E	Ctrl N	SO
15	0F	Ctrl O	SI
16	10	Ctrl P	DLE
17	11	Ctrl Q	DC1
18	12	Ctrl R	DC2
19	13	Ctrl S	DC3
20	14	Ctrl T	DC4
21	15	Ctrl U	NAK

Decimal	Hex	Control Code	ASCII
22	16	Ctrl V	SYN
23	17	Ctrl W	ETB
24	18	Ctrl X	CAN
25	19	Ctrl Y	EM
26	1A	Ctrl Z	SUB
27	1B	Ctrl [	ESC
28	1C	Ctrl \	FS
29	1D	Ctrl]	GS
30	1E	Ctrl ^	RS
31	1F	Ctrl _	US
32	20		SP
33	21		!
34	22		"
35	23		#
36	24		$
37	25		%
38	26		&
39	27		'
40	28		(
41	29		)
42	2A		*
43	2B		+

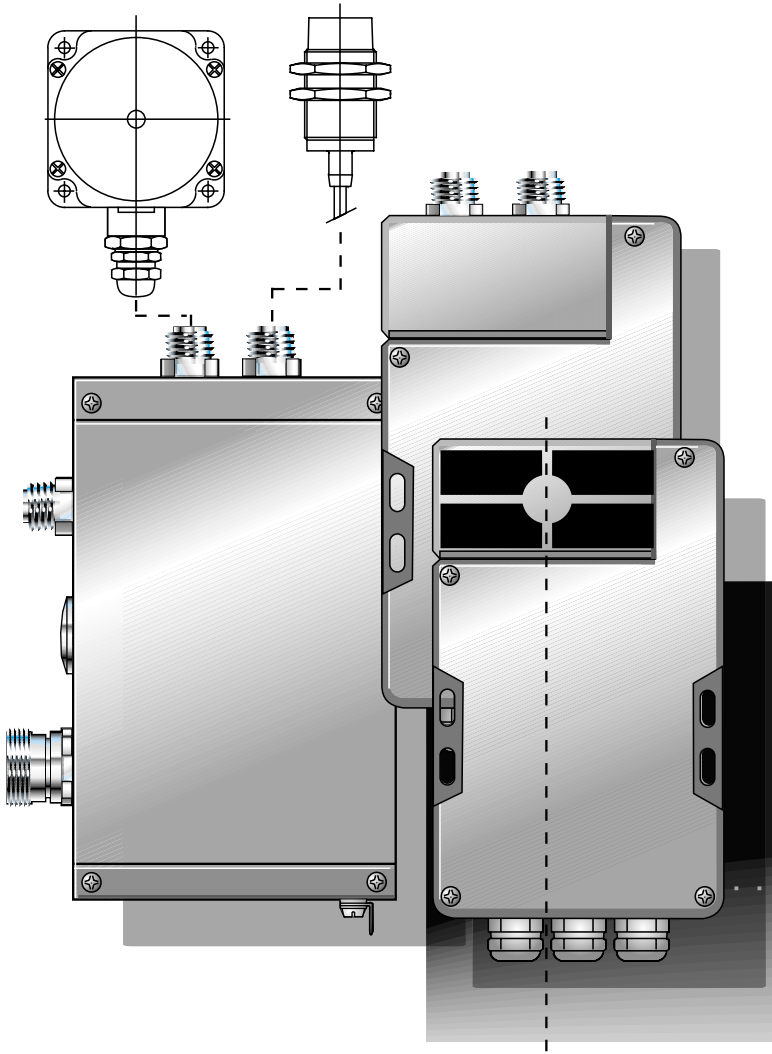
Decimal	Hex	ASCII
44	2C	,
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?
64	40	@

Decimal	Hex	ASCII
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U

Decimal	Hex	ASCII
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j

Decimal	Hex	ASCII
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	DEL

BALLUFF



Manual

Electronic Identification Systems BIS
Server for Windows "BIS022SV.EXE"
Processor BIS C-6_0-022-...

Deutsch – bitte wenden!

Nr. 815 558 D/E • Edition 0101
Subject to modification.
Replaces edition 9903

Writing convention:
Control characters to be transmitted are in angle brackets.
Characters to be transmitted in ASCII code are enclosed in apostrophes.
Example: <STX> '1 2 3 4 5 6' BCC

http://www.balluff.de

Balluff GmbH
Schurwaldstrasse 9
73765 Neuhausen a.d.F.
Germany
Phone +49 (0) 71 58/1 73-0
Fax +49 (0) 71 58/50 10
E-Mail: balluff@balluff.de

Content

Safety advisory

Approved operation

Series BIS C-6_0 processors along with the other BIS C system components comprise an identification system and may only be used for this purpose in an industrial environment in conformity with Class A of the EMC Law.

Installation and operation

Installation and operation should be carried out only by trained personnel. Unauthorized work and improper use will void the warranty and liability.

When installing the processor, follow the chapters containing the wiring diagrams closely. Special care is required when connecting the processor to external controllers, in particualy with respect to selection and polarity of the signals and power supply,

Only approved power supplies may be used for powering the processor. See Technical Data for details.

Use and testing

Prevailing safety regulations must be adhered to when using the identification system. In particular, steps must be taken to ensure that a failure of or defect in the identification system does not result in hazards to persons or equipment.

This includes maintaining the specified ambient conditions and regular testing for functionality of the identification system including all its associated components.

Function faults

Should there ever be indications that the identification system is not working properly, it should be taken out of commission and secured from unauthorized use

Scope

This manual applies to processors in the series BIS C-600-022-...-00-KL1, BIS C-600-022-...-01-KL1 and BIS C-620-022-050-00-ST2, BIS C-620-007-050-01-ST2 in conjunction with Server Software BIS022SV.EXE or BALLU.EXE using DDEML interface under Windows 3.1, Windows95 or Windows NT.

Introduction
BIS C Identification System

This manual is designed to assist the user in setting up the control program and installing and starting up the components of the BIS C-6_0 Identification System, and to assure rapid, trouble-free operation.

Principles

The BIS C-6_0 Identification System belongs in the category of

non-contact systems for reading and writing.

This dual function permits applications for not only transporting information in fixed-programmed code tags, but also for gathering and passing along up-to-date information as well.

Applications

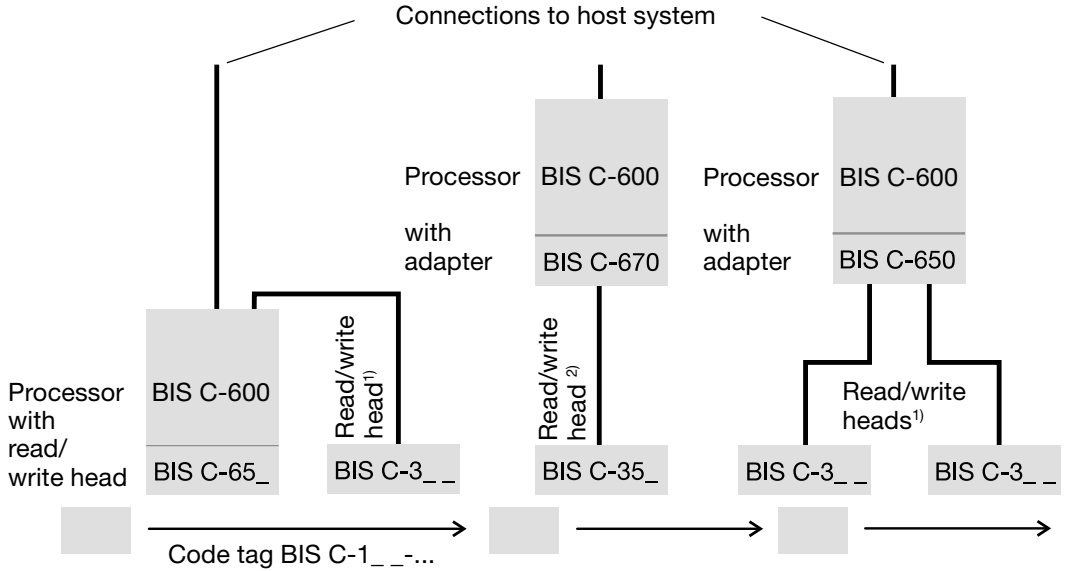
Some of the notable areas of application include

- **for controlling material flow in production processes**
  (e.g. in model-specific processes)**,**
  for workpiece conveying in transfer lines,
  in data gathering for quality assurance ,
  for gathering satety-related data,
- **in tool coding and monitoring;**
- **in equipment organization;**
- **in storage systems for monitoring inventory movement;**
- **in transporting and conveying systems;**
- **in waste management for quantity-based fee assessment.**

System components

The main components of the BIS C-600 Identification System are

- **Processor,**
- **Read/write heads, and**
- **Code tags.**



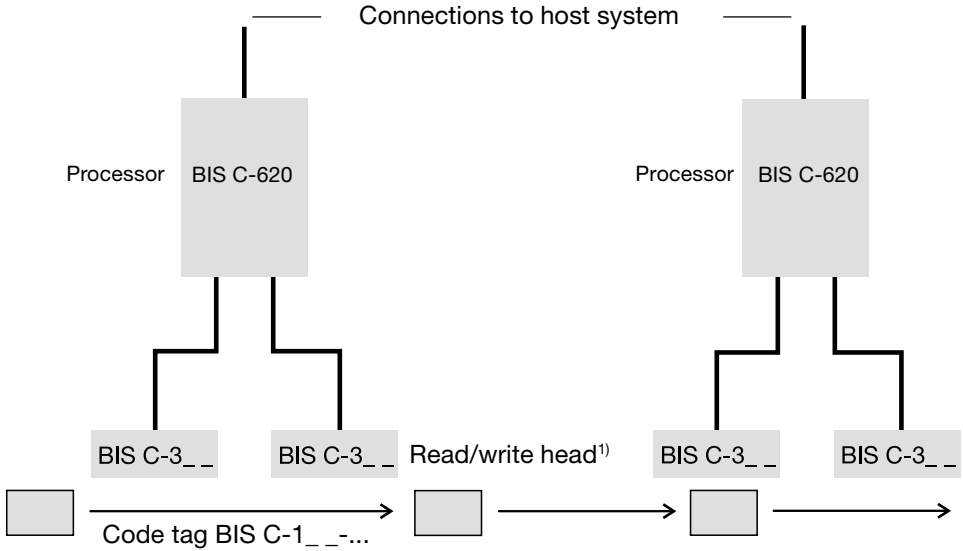
Schematic representation of an identification system (example)

[1)] except BIS C-350 and -352 [2)] only BIS C-350 or -352

System components

The main components of the BIS C-620 Identification System are

- **Processor,**
- **Read/write heads, and**
- **Code tags**.



Schematic representation of an identification system (example)

1) except BIS C-350 and -352

Selecting system components

The **BIS C-600** processor has a plastic housing. Connections are made through a terminal strip, with the cable secured by a PG fitting. A single read/write head from series BIS C-65_ can be directly connected to the processor, which creates a compact unit. Additinally a second read/write head can be connected separately via cable to the terminal strip. If the BIS C-650 adapter is attached instead of the BIS C-65_ read/write head, two read/write heads may be cable connected. If the BIS C-670 adapter is attached, one read/write head may be cable connected.

The **BIS C-620** processor has a metal housing. Connection is made through round connectors. Two read/write heads can be cable connected to the BIS C-620 processor.

Additional information on the read/write heads in series BIS C-65_ including all the possible code tag combinations can be found in the manuals for the respective read/write heads.

Whether the compact version of the processor with integrated read/write head makes sense or whether the external solution is preferred depends primarily on the spatial arrangement of the components. There are no functional limitations. All read/write heads are suitable for both static and dynamic reading. Distance and relative velocity are based on which code tag is selected. Additional information on the read/write heads in series BIS C-65_ and series BIS C-3_ _ including all the possible code tag/read-write head combinations can be found in the manuals for the respective read/write heads.

The system components are electrically supplied by the processor. The code tag represents an free-standing unit and needs no line-carried power. It receives its energy from the read/write head. The latter constantly sends out a carrier signal which supplies the code head as soon as the required distance between the two is reached. The read/write operation takes place during this phase. Reading and writing may be dynamic or static.

Dialog mode

The processor writes data from the host system to the code tag and reads it from the code tag through the read/write head, and prepares it for the host system. Host systems may be:

– **a controlling computer (e.g., industrial PC) or**
– **a programmable logic controller (PLC)**

Application with Server-Software 022

The Server Software (BIS022SV.EXE) allows the system components of the BIS C-6_0 Identification System to any control software (client under Windows 3.1, Windows95, or Windows NT) by adapting the client parameters through the DDE interface (DDEML) to the Balluff Identification System. The server automatically handles the conversion of the internal protocol of the identification system through the client-server interface (see chapter 'Application with Server-Software 022' starting on page 30). The user works exclusively with the client software.

The BIS022SV.EXE server is accepted for use with the Siemens SINUMERIK 840D. Manual entry of the tool data is replaced by automatic reading and writing of the tag attached to the tool. This creates the coupling and expansion of the load and unload dialog in a machine controller with a tool identification system.

Application with Standard Protocol 007

The processor controls and adminsters the data communication between code tags and read/write heads. The serial port connects the BIS C-6_0 identification system to the external controller.

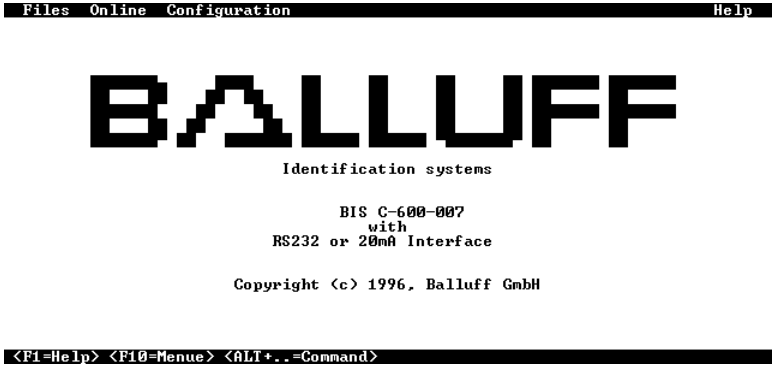
Available are an RS 232 (V.24) or 20 mA current loop interface (TTY).

Data is exchanged between the processor and the host system using specific telegrams. The protocol sequence is represented in the form of function blocks. See the chapter 'Application with Standard Protocol 007' for more details on the protocol sequence and on the telegram contents starting on page 46.

Configuration

Before programming, the processor configuration must be carried out, in case the factory settings will not be used.

Configuration is done using a computer and the Balluff BISC600A.EXE software, and it is stored in the processor. It may be overwritten at any time. The configuration can be stored in a file, making it accessible when required.



Important. **Please note.**
Please note the selected settings on the stick-on label supplied (to be pasted on the inside of the processor cover) as well as on page 26 and 28 in the customer configuration section, so that in case of repair of the processor the settings can be saved or otherwise can also be used to set other processor units.

Interface Menu BIS C-6_0

The first screen shows the parameters baud rate, number of data and stop bits, and parity type for the serial interface selected. The graphic shows the factory settings. The other settings are carried out in the corresponding masks which are illustrated in the following pages.

The menu shows the factory setting.

This setup is mandatory when using the Client-Server application!

INTERFACE BIS C-600

baudrate
( ) 600 baud
( ) 1200 baud
( ) 2400 baud
( ) 4800 baud
(•) 9600 baud
( ) 19200 baud

databit
( ) 7
(•) 8

stopbit
(•) 1
( ) 2

parity
( ) odd
(•) even
( ) none

< Next > < Shortform > < ESC = Exit > < Print . . . . > < F1 = Help >

19200 baud can only be used with the RS 232/V.24 interface, not with the TTY interface.

If the initializing data are available in short form (e.g. on the processor cover after a replacement of the unit) the data can be entered directly into the "Shortform of initialization BIS C-600" mask (see also Customer cofiguration on page 26/27).

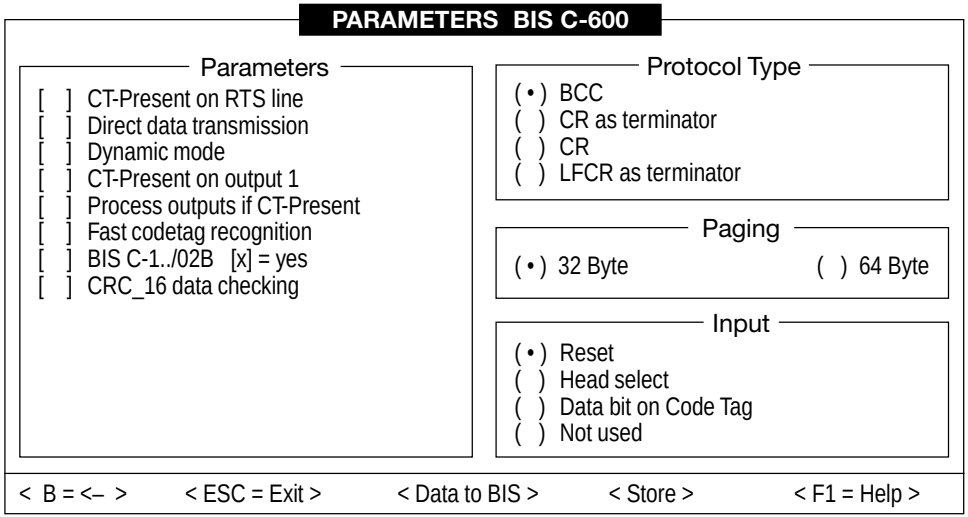
SHORTFORM OF INITIALIZATION BIS C-600

[09600] [8] [1] [E]
[1] [2] [2] [1] [1]
[00000000]

< B = <– > < ESC = Exit > < F1 = Help >

All other settings are done in the menus shown on the following pages.

Menu settings BIS C-6_0



Menu settings BIS C-6_0, Protocol Type Field

Protocol Type Field:

For host devices which require a terminator, the additional use of Carriage Return 'CR' or Line Feed with Carriage Return 'LF CR' is made available. The following page contains examples of the various possibilities.

Client-Server application:
- **Protocol type BCC is mandatory!**
- **'Direct data transmission' is not allowed!**
- **'CRC_16 Data checking' is not allowed!**

Menu settings BIS C-6_0, Protocol Type Field (continued)

Examples for terminating telegrams:

Protocol Variants	Telegram with command, Address and no. of bytes	Termi-nator	Acknow-ledge	Terminator
with Blockcheck BCC	'R 0000 0001'	BCC	<ACK> '0'	
with Carriage Return	'R 0000 0001'	'CR'	<ACK> '0'	
with Terminator Carriage Return	'R 0000 0001'	'CR'	<ACK> '0'	'CR'
with Terminator Carriage Return and Line Feed	'R 0000 0001'	'LF CR'	<ACK> '0'	'LF CR'

Menu settings BIS C-6_0, Parameter Field

Parameters Field:

– CT-Present on RTS
The parameter CT-Present on RTS (corresponds to the LED indicator Codetag Present on the BIS C-6_0 housing) can be used to allow the PC to check CT-Present using a hardware signal. You can connect the RTS signal, for example, to the free input RI (Ring Indicator) of the PC as this is only used with Modem operation.

– Direct data transmission
Each time a code tag is recognized, the data will be read out depending on the configuration and output to the interface. With this setting the read command in the dialog mode is superfluous.

Client-Server application:
– Protocol type BCC is mandatory!
– 'Direct data transmission' is not allowed!

Menu settings BIS C-6_0, Parameter Field (continued)

– **Dynamic Mode**
This function switches off the error-message "No code tag present", i.e.:
-> In dynamic mode, a read or write telegram is stored until a code tag enters the working range of the corresponding read/write head.
-> Without dynamic mode, a read or write telegram is acknowledged with an error message <NAK> '1' if there is no code tag present in front of a read/write head; the processor goes into the ground state.

– **CT-Present on Output 1**
If CT Present is selected for Output 1, the LED message Codetag Present is also output on Output 1. In this way the presence of a code tag can be directly verified as a digital hardware signal.

– **Process Outputs if CT Present**
The output functions are normally processed only after a read command. But since code tag recognition is also an automatic tag read (reads first page, either 32 or 64 bytes depending on type), the output processing can occur simultaneously with Codetag Present. If the addresses for output processing are located on the first page, then the processor can itself carry out short control commands without a seperate command.
-> For very fast transactions, see next section.

– **Fast Code Tag Recognition**
For very fast transactions, the number of code tag addresses used for code tag recognition can be reduced from 32 or 64 bytes to 4 bytes. The code tag recognition time is thereby reduced to ca. 50 ms (instead of ca. 150 ms for tags with < 2 kBytes or ca. 250 ms for code tags with ≥ 2 kBytes of memory).
-> Please note this when using the parameter "Process outputs with CT-present".

– **BIS C-1../02B [x] = yes**
This parameter should be switched on when a code tag of the type BIS C-1../02B is used.

– **CRC_16 Data Checking:** Not allowed when using the Client-Server application.

Menu settings BIS C-6_0, Page Size Field

Code tag memory is organized in page sizes of 32 or 64 bytes (also referred to as block size). The processor uses the type to correctly handle the operations. Factory setting: 32 bytes.

Code tags with 32 byte block size	BIS C-1_ _-02/_
	BIS C-1_ _-03/_
	BIS C-1_ _-04/_
	BIS C-1_ _-05/_

Code tags with 64 byte block size	BIS C-1_ _-10/_
	BIS C-1_ _-11/_
	BIS C-1_ _-30/_

Menu settings BIS C-6_0, Input Field

The digital control input on the BIS C-6_0 can be specified for the desired function.

– **Reset** (Factory setting)
  If Reset is selected, a High signal on this input resets the BIS C-600 processor. All pending commands are deleted.

– **Head Select**
  If Head Select is selected, this input is used to activate the desired read head.

  Input Low: Head 1 selected.
  Input High: Head 2 selected.

  -> This function always has priority. For example the function "Both read/write heads active" selected by the 'HT' command is deactivated.

– **Data bit to code tag**
  When a new code tag is recognized, a freely defined bit is written to the code tag directly or inverted to a specified address. After a successful write the defined output is set until the code tag leaved the acvtive zone of the read/write head.

  -> The 'Dynamic Mode' parameter is automatically reset.

– **No function**
  The input is not processed.

Menu Allocate Input/Outputs (BIS C-600 only)

Allocate to input/outputs

The outputs can have various functions allocated to them. The output functions are always processed when reading. The condition for this is that the respective address was read during the preceding read request.

The menu shows the factory setting.

This setup is mandatory when using the Client-Server application or with BIS C-620, i.e. nothing should be allocated to the outputs!

ALLOCATE INPUT/OUTPUTS

(•) Outputs not used.
( ) Output half-byte of the data contents of an address.
( ) Compare contents of mjltiple addresses with a reference value.
( ) Compare contents of one address with various reference values.
( ) Compare contents of multiple addresses with the contents of another address.
( ) Output data bits from variable addresses.

( ) Program input as data bit to code tag.
( ) Read and send code tag data without direct command.

< OK > < Shortform> < ESC > < Print. . . . > < F1 = Help >

"Outputs not used" deactivates processing of the outputs.

OUTPUTS NOT USED

< Data to BIS> < Store > < ESC = Exit > <F1 = Help >

"Data to BIS" sends the data to the processor. "Save" stores the data in the configuration file of your computer.

Menu Allocate Input/Outputs (BIS C-600 only) (continued)

Output Halfbyte of the Data Contents of an Address:

IN/OUTPUT CONFIGURATION

( ) Output not used.
(•) Output halfbyte of data contents of an address.
( ) Compare contents of multiple addresses with a fixed value.
( ) Compare contents of an address with various fixed values.
( ) Compare contents of multiple addresses with a content of one address.
( ) Output data bits of variable addresses.

( ) Program input as data bit to code tag.
( ) Read and transmit code tag data without command.

< OK > < Shortform > < ESC = Exit > < Print . . . . > < F1 = Help >

OUTPUT HALFBYTE OF DATA CONTENTS OF AN ADDRESS

Code Tag adress [9999]

( ) Output High Nibble ?
(•) Output Low Nibble ?

< Data to BIS > < Store > < ESC = Exit > < F1 = Help >

The 8 bits data contents of an address is output in nibbles of 4 bits each (Bit 0 on Output 1, Bit 1 on Output 2, etc.). Depending on the setup this is either the upper (High Nibble) or lower 4 bits (Low Nibble). The address is given in decimal.

Menu Allocate Input/Outputs (BIS C-600 only) (continued)

Compare Contents of multiple Addresses with a fixed Value:
The data contents of up to 4 addresses are compared with a fixed decimal value. To each address can be assigned which of the Outputs 1 to 4 is set or cancelled by a positive result of the comparison an whether in case of a negative result of the comparison the output shall not be changed or shall be set in contrary to the definition with the positive result (inverted response).
All addresses found within the read command will be processed.

Compare Contents of multiple addresses with fixed value

Fixed value: [000]

Address	Output	Positive compar.	Negative compar.
[9999]	[1]	(•) Set ( ) Clear	( ) No change (•) Invert
[9999]	[2]	(•) Set ( ) Clear	( ) No change (•) Invert
[9999]	[3]	(•) Set ( ) Clear	( ) No change (•) Invert
[9999]	[4]	(•) Set ( ) Clear	( ) No change (•) Invert

< Data to BIS > < Store > < ESC = Exit > < F1 = Help >

If the parameter "Process outputs with CT-Present" is included in the initialization, then this function also will be carried out on recognition of a new code tag (one or more of the addresses given should be on the first page of the code tag).

Menu Allocate Input/Outputs (BIS C-600 only) (continued)

Compare an Address with various fixed Values:
The data contents of an address is compared with 4 fixed decimal values. For each fixed value is indicated which of the Outputs 1 to 4 is set or cancelled by a positive result of the comparison and whether in case of a negative result of the comparison the output shall not be changed or shall be set in contrary to the definition with the positive result (inverted response).

Compare contents of an address with various fixed Values

Address: [9999]

Fixed value	Output	Positive compar.	Negative compar.
[000]	[1]	(•) Set ( ) Clear	( ) No change (•) Invert
[000]	[2]	(•) Set ( ) Clear	( ) No change (•) Invert
[000]	[3]	(•) Set ( ) Clear	( ) No change (•) Invert
[000]	[4]	(•) Set ( ) Clear	( ) No change (•) Invert

< Data to BIS > < Store > < ESC = Exit > < F1 = Help >

If the parameter "Process outputs with CT-Present" is included in the initialization, then this function also will be carried out on recognition of a new code tag (one or more of the addresses given should be on the first page of the codetag).

Menu Allocate Input/Outputs (BIS C-600 only) (continued)

Compare Contents of multiple Addresses with the Content of another Address:
The data contents of up to 4 addresses are compared with the data contents of another address. For each address is indicated which of the Outputs 1 to 4 is set or cancelled by a positive result and whether in case of a negative result of the comparison the output shall not be changed or shall be set in contrary to the definition with the positive result.
All addresses found within the read command will be processed provided the address to be compared is within the range.

Compare contents of multiple addresses with content of one address

Address: [9999]

Address	Output	Positive compar.	Negative compar.
[9999]	[1]	(•) Set ( ) Clear	( ) No change (•) Invert
[9999]	[2]	(•) Set ( ) Clear	( ) No change (•) Invert
[9999]	[3]	(•) Set ( ) Clear	( ) No change (•) Invert
[9999]	[4]	(•) Set ( ) Clear	( ) No change (•) Invert

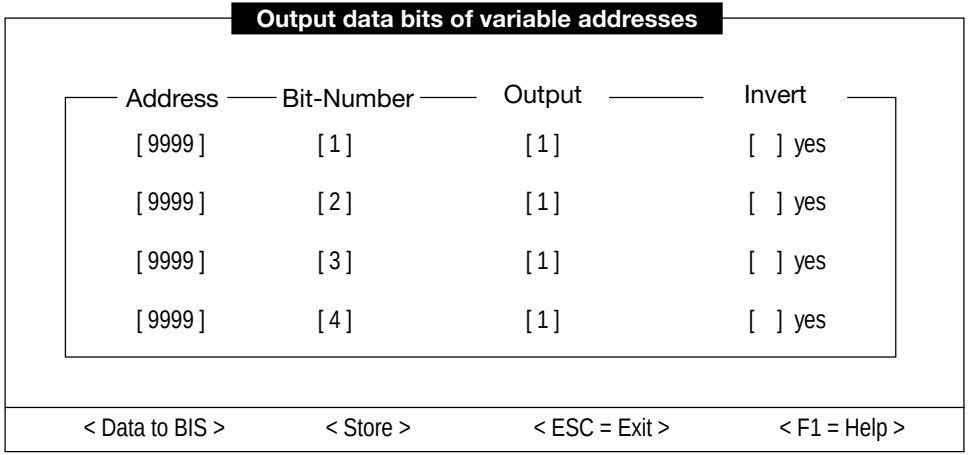
< Data to BIS > < Store > < ESC = Exit > < F1 = Help >

If the parameter "Process outputs with CT-Present" is included in the initialization, then this function also will be carried out on recognition of a new code tag (one or more of the addresses given should be on the first page of the codetag).

Menu Allocate Input/Outputs (BIS C-600 only) (continued)

Output Data Bits of variable Addresses:

1 data bit of an address or 1 bit each from up to 4 addresses can be output on one of the 4 outputs and then inverted or not inverted.



If the parameter "Process outputs with CT-Present" is included in the initialization, then this function also will be carried out on recognition of a new code tag (one or more of the addresses given should be on the first page of the codetag).

Menu Allocate Input/Outputs
(continued)

Programming a Data Bit on the Code Tag depending on the Input:

On recognition of a new code tag the state of the digital input will be written as a direct or inverted bit on the codetag.

The address range is 0...31! Bit nummer of the address is 1...8.

The outputs to be used for the ready and release signals should also be given. If release output is given as "0" then the release function will not be used. The procedure is described below.

Program input as data bit to code tag

address	[00]	[ ] Input inverse ?
bit number	[1]	
ready output	[1]	
release output	[0]	

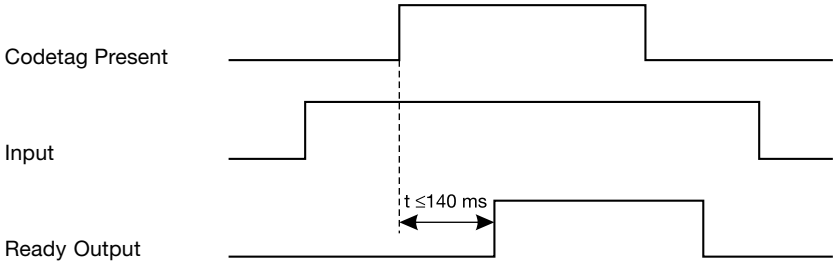
< Data to BIS > < Store > < ESC = Exit > < F1 = Help >

Procedure without Release Signal

On recognition of a new code tag the defined bit of the given address will be written, direct or inverted. After a successful write operation, the given ready output is set until the codetag leaves the active read/write range. The input must hold its state until the ready output is set.

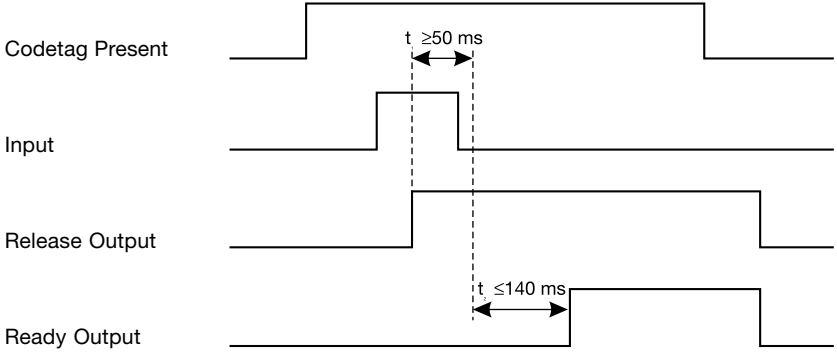
The input state that is to be written as information on the code tag must be present already before the presence of the new code tag is recognized.

Menu Allocate Input/Outputs
(continued)



In this example the bit is set to 1.

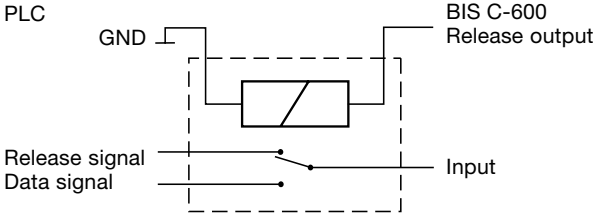
Procedure with Release Signal
After the recognition of a new code tag, the input state is continuously sampled until it is set (release given). The processor sets the release output and waits 50 ms. The input is then sampled and its state is taken over as bit-value. This value is written on the codetag directly or inverted depending on the selected configuration. After the write operation, the given ready output is set until the code tag leaves the active read/write range. The release output then reverts to low.



In this example the bit is reset to 0.

Menu Allocate Input/Outputs (continued)

The release output can be used to operate a relay in order to switch the input between the release and data signals.



As an input signal to a PLC this output can indicate that it is time to switch the data signal to the input of the BIS C-6-0. This signal is required at all places where there is a possibility that the code tag can come into the read/write range before the input data signal is present at the processor.

This function must be additionally released for use during the initializing phase.

Menu Allocate Input/Outputs
(continued)

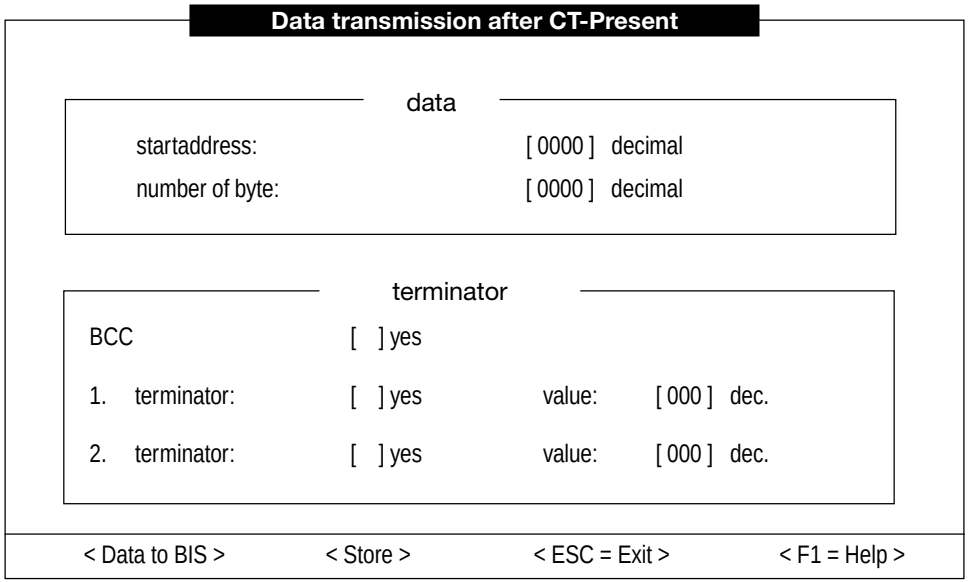
Read and transmit tag data without command:

The given number of bytes (Number of bytes from Start address) will be read from a newly recognized code tag.

After they are read, the data will be automatically sent to the interface.

Additionally, as termination, a BBC and/or 1 or 2 freely definable terminating characters can be sent.

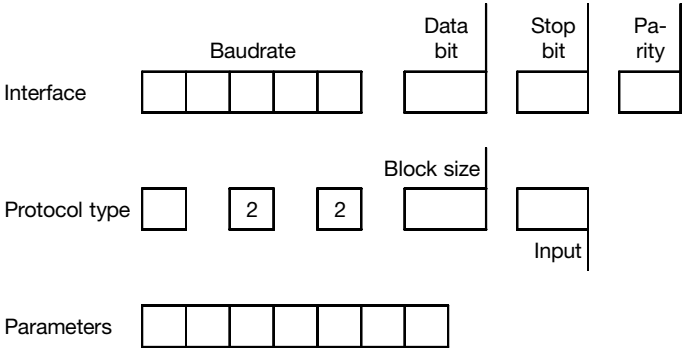
Not allowed when using the Client-Server application!



Customer Configuration

Initialization

Please note the settings in the label fields on the inside of the processor cover so that in case of repair of the processor the settings can be reset in the factory. Note the settings also in the following fields so that you can set, e.g. other units, to an identical configuration.



On the following page you will find an example which shows how you can print-out after initializing. Enter the settings in the appropriate fields so that you have them handy and can reproduce the settings at any time. You can then enter the data in short form into the mask (see also page 11).

SHORTFORM OF INITIALIZATION BIS C-600

[19200] [8] [1] [N]
[4] [2] [2] [2] [1]
[01101100]

< B = <– > < ESC = Exit > < F1 = Help >

Initialization
(continued)

Example of a print-out after initialization which you can print with the software BISC600A.EXE.

Interface settings

Baudrate	:	9600 baud
Data bit	:	8
Stop bit	:	1
Parity	:	Even

Protocol

( ) BCC
( ) CR as terminator
( ) CR
(*) LF CR as terminator

Parameter

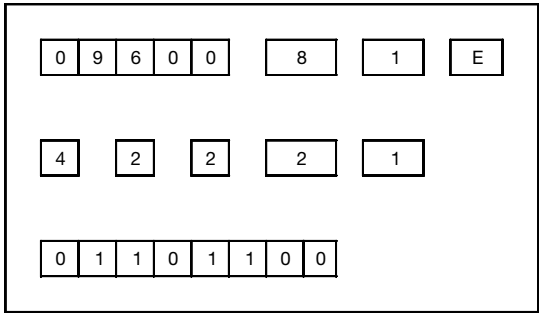
[ ] Output Codetag Present on RTS line.
[X] Direct data transmission
[X] Dynamic mode
[ ] Codetag Present signal on output 1
[X] Process outputs with Codetag Present
[X] Fast code tag recognition
[ ] BIS C-1../02B [X] = yes
[ ] CRC-16 data checking

Block size

( ) 32 Byte page size
(*) 64 Byte page size

Input

(*) Input = RESET
( ) Input = Head selection
( ) Input = Data bit on code tag
( ) Input = Not used

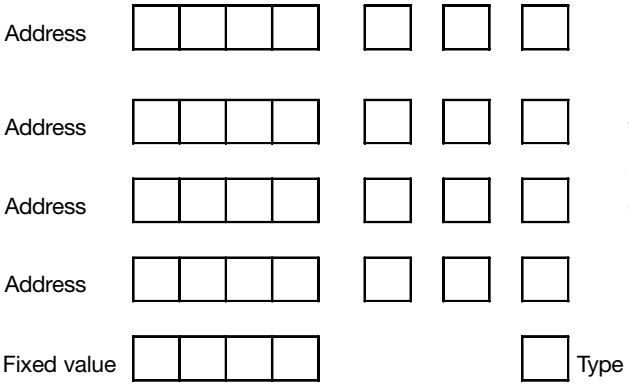


The entries in the field are either in clear text (as with Interface settings) or the number of the line marked. In the case of 'Parameters' the marked line is indicated by a 1.

Customer Configuration

Input/Output Configuration

Please note the settings in the label fields on the inside of the processor cover so that in case of repair of the processor the settings can be reset in the factory. Note the settings also in the following fields so that you can set, e.g other units, to an identical configuration.



On the following page you will find an example which shows how you can print-out after initializing. Enter the settings in the appropriate fields so that you have them handy and can reproduce the settings at any time. You can then enter the data in short form into the mask (see below).

SHORTFORM I/O CONFIGURATION BIS C-600

[0000] [0] [0] [0]
[0000] [0] [0] [0]
[0000] [0] [0] [0]
[0000] [0] [0] [0]
[0000] [1]

< B = <– > < ESC = Exit > < F1 = Help >

In case the initialization settings are available (e.g. on the cover after exchanging the processor), the settings can be directly entered into the mask "Shortform I/O configuration BIS C-600".

Input/Output Configuration (continued)

Example of a print-out after initialization which you can print with the software BISC600A.EXE.

Input/output configuration

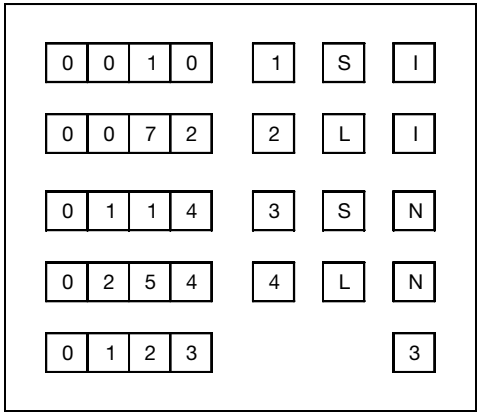
( ) Output not used.
( ) Output halfbyte of data contents of an address.
(•) Compare contents of multiple addresses with a fixed value.
( ) Compare contents of an address with various fixed values.
( ) Compare contents of multiple addresses with content of one address.
( ) Output data bits of variable addresses.

( ) Program input as data bit to code tag.
( ) Read and transmit tag data without command.

Definition

Fixed value: 123

Address:	0010	
Output:	1	
Positive comparison:	(•) Set	( ) No change
Negative comparison:	( ) Clear	(•) Invert
Address:	0072	
Output:	2	
Positive comparison:	( ) Set	( ) No change
Negative comparison:	(•) Clear	(•) Invert
Address:	0114	
Output:	3	
Positive comparison:	(•) Set	(•) No change
Negative comparison:	( ) Clear	( ) Invert
Address:	0254	
Output:	4	
Positive comparison:	( ) Set	(•) No change
Negative comparison:	(•) Clear	( ) Invert



Application
Server Software 022

Function Overview

The server performs the following functions:

– Read data from the code tag
– Write data to the code tag
– Reset identification system
– Transmit initialization information
– Break off a read/write operation
– Read error number (for synchronous operation)
– Read error text

Synchronous and Asynchronous Operation

The server processes one operation at a time. Reading and writing to code tags is time intensive as a rule. Therefore the software offers an interface for synchronous and asynchronous processing for different user applications.

In applications where time is not critical, synchronous operation can be selected. It can not be broken off; however, the server permits changing of tasks in the operating system (see example on next page).

In order not to block the host system in time-critical applications, it must be possible to break off an operation in process at any time. This is permitted in the asynchronous mode. Details for each of these options are provided in the following pages.

Change Task

Example of a synchronous REQUEST transaction permitting a task change in the loop.

Client		Server
DdeClientTransaction (..XTYP_REQUEST.. ..<timeout>..)	→	XTYP_REQUEST: Loop Carry out (partial) processing if (PeekMessage(...)) TranslateMessage (..) DispatchMessage (..) until process is finished RESULT→ global data object return (result)
	←	

Data Formats

The following conversion is used for data exchange on the DDE interface in CF_TEXT format:

Character	decimal	hexadecimal	"Intel-Hex"
<CR>	13	0D	'0' 'D'
A	65	41	'4' '1'

Installation

The server software together with all DLL files and the INI file must be copied into a common directory.

Server Initialization File

The initialization file ocntains along with the server-specific paramters the following sections for communication with the client over the DDE interface:

[Services]
; Identification system processor
Service1 = ToolIdentSystem

[Topics]
; Read/write head number (identification system)
Topic1=Unit1

[Misc]
; Miscellaneous
; maximum code tag capacity in bytes
MAX_CC_CAPACITY=<n>

[Ports]
; user COM-Port
Port=COM<n>

Enter the corresponding value for <n>. Use the INISET.EXE program, which is found on the Install diskette. Copy INISET.EXE into the directory which contains BIS022SV.INI or BALLU.INI, and run it.

Service

Service is used to talk to the identification system's processor. The services offered by the server are stored in the [Services] section of the server initialization file.

Topic

Topic is used to specify the number of the identification system's read/write head. The topics are stored in the [Topics] section of the server initialization file.

Misc

Misc is used for specifying the maximum code tag capacity (in bytes). This value can be found in the data sheet for the corresponding code tag.

Port

Port is used to specify the COM interface for the processor.

Status

In asynchronous mode the client queries the processing status **Status** over a "hot link". The processing status can assume the following values:

"000" Processing was completed with no errors.
For "Read Code Tag", the net data will follow

"-01" Request in process.

"-02" Processing was broken off.

"nnn" Identification system error number.
Error numbers are greater than "000".
The corresponding error text can be shown using the "Read Error Text" command.
It will be output in whatever language has been selected.

Transactions (Overview)

The server provides the following functions for synchronous and asynchronous processing:

	Synchronous Mode		Asynchronous Mode		
	Transaction	Item / Execute-String	Transaction	Item / Execute-String	
Read from Code Tag	REQUEST	"Data[<offset>, <count>]"	EXECUTE	"CMD=Read	OFFSET=<offset> COUNT=<count>"
			ADVISE	"Status"	
Write to Code Tag	POKE	"Data[<offset>, <count]"	EXECUTE	"CMD=Write	OFFSET=<offset> COUNT=<count> DATA=<data>"
			ADVISE	"Status"	
Reset Identification System	EXECUTE	"CMD=Reset"	EXECUTE ADVISE	"CMD=Reset "Status"	ASYNC"
Transmit Initialization Information	EXECUTE	"CMD=Init LANGUAGE=<path> [EOT=<e>]"	EXECUTE	"CMD=Init	LANGUAGE=<path> [EOT=<e>] ASYNC"
			ADVISE	"Status"	
Abort Read/Write Process	EXECUTE	"CMD=Abort"			
Read Error Number	REQUEST	"GetError"			
Read Error Text	REQUEST	"GetErrortext"			

Read Data from Code Tag

Synchronous		Asynchronous	
Transaction:	Request-Item:	Transaction:	Command String:
XTYP_REQUEST	"Data[<offset>, <count>]"	XTYP_EXECUTE	"CMD=Read OFFSET=<offset> COUNT=<count>"
		ADVISE	"Status"

The read begins at the address defined with <offset>.
The number of bytes to read are defined with <count>.
Brackets [ ] are mandatory. <offset> and <count> are entered as 4 digit numbers.
Example: Start adress = 0. No. of bytes = 25.
Synchronous: "Data[0000, 0025]"
Asynchronous: "CMD=Read OFFSET=0000 COUNT=0025"
The data are sent to the client in "Intel-Hex" format.

For initialization with data terminator:
If a data terminator was specified as initialization information, the maximum number of bytes read is the number specified by <count>. If during a read the terminator is recognized sooner, the read is ended at this point (incl. terminator).

The number of bytes delivered is derived from the length of the data object transmitted by the server.

The read cannot be abort by the client. The operating system does permit task changing.	The read can be aborted by the client.

Read Data from Code Tag, Reaction of the Server

Synchronous

The following events are processed in the callback routine:

XTYP_REQUEST

- Read data from code tag.
- Process error status of the identification system.

If data transmission was concluded without error:

- Convert data from binary into "Intel-Hex" format.
- Store data in global data object.
- Return value = act on global data object.

With identification system error:

- Return value = DDE_FNOTPROCESSED
  Client receives return value and gets error number with request "Read Error Number". Value larger than "000" indicated identification system error. Get error text with "Read Error Text".

Asynchronous

The following events are processed in the callback routine:

XTYP_EXECUTE (called up by client)
- Acknow: DDE_FACK, DDE_FBUSY or DDE_FNOTPROCESSED *.

XTYP_ADVSTART (called up by client)
- "Status" = "-01", acknowledge with TRUE.

XTYP_REQUEST (called up by client)
- Item is "Status", start timer.

XTYP_TIMER (if timer runs out)
- Callup processing function (PostMessage).

XTYP_ADVREQ (called up by server)
- Store "Status" in global data object.
- Reply value = act on global data object.

Processing functions:
- Read data from code tag.
- Update processing status "Status":
  - Processing not finished: "Status" = "-01", PostMessage (processing function)
  - Abort by client: "Status" = "-02", DdePostAdvise
  - Processing finished w/out error: convert data from binary to "Intel-Hex" and copy to "Status", "Status" = "000<..data..>", DdePostAdvise
  - Processing finished w/error: "Status" = "<errornr wz-ident-system>", DdePostAdvise. Get error text with "Read Error Text".

* Client sends DDE request with item "lasterror", server replys with "-03".

Write Data to Code Tag

Synchronous		**Asynchronous**	
Transaction:	Request-Item:	Transaction:	Command String:
XTYP_POKE	"Data[<offset>, <count>]"	XTYP_EXECUTE	"CMD=Write OFFSET=<offset> COUNT=<count> DATA=<data>"
		ADVISE	"Status"

The write begins at the address specified in <offset>.
The number of bytes written is defined by <count>.
Brackets [ ] are mandatory. <offset> and <count> are entered as 4 digit numbers.
Example: Start adress = 0. No. of bytes = 25.
Synchronous: "Data[0000, 0025]"
Asynchronous: "CMD=Read OFFSET=0000 COUNT=0025"
The data are transmitted to the client in "Intel-Hex" format.

The write cannot be aborted by the client. The operating system does permit task changing.	The write can be aborted by the client.

Write Data to the Code Tag, Reaction of the Server

Synchronous

Callback routine processes the following events:

XTYP_POKE

– Read data from global data object.
– Convert data from "Intel-HEx" to binary.
– Write data to code tag
– Process error status of the identification system.

If data transmission was concluded without error:

– Return value = DDE_FACK

With identification system error:

– Return value = DDE_FNOTPROCESSED
Client receives return value DDE_FNOTPROCESSED and gets error number with request "Read Error Number". Value larger than "000" indicated identification system error. Get error text with "Read Error Text".

Asynchronous

Callback routine processes the following events:

XTYP_EXECUTE (called up by client)
– Acknow: DDE_FACK, DDE_FBUSY or DDE_FNOTPROCESSED *.

XTYP_ADVSTART (called up by client)
– "Status" = "-01", acknowledge with TRUE.

XTYP_REQUEST (called up by client)
– Item is "Status", start timer.

XTYP_TIMER (if timer runs out)
– Callup processing function (PostMessage).

XTYP_ADVREQ (called up by server)
– Store "Status" in global data object.
– Reply value = act on global data object.

Processing functions:
– Convert data from "Intel-Hex" to binary.
– Write data to code tag.
– Update processing status "Status":
  – Processing not finished: "Status" = "-01", PostMessage (processing function)
  – Abort by client: "Status" = "-02", DdePostAdvise
  – Processing finished w/out error: "Status" = "0000", DdePostAdvise
  – Processing finished w/error: "Status" = "<errornr wz-ident-system>", DdePostAdvise. Get error text with "Read Error Text".

* Client sends DDE request with item "lasterror", server replys with "-03".

Reset Identification System

Synchronous		Asynchronous	
Transaction:	Command String:	Transaction:	Command String:
XTYP_EXECUTE	"CMD=Reset"	XTYP_EXECUTE ADVISE	"CMD=Reset ASYNC" "Status"

The identification system is reset to the base state.

The action cannot be aborted by the client. The operating system does permit task changing.	The action can be aborted by the client.

Reset Identification System, Reaction of the Server

Synchronous

The following events are processed in the callback routine:
XTYP_EXECUTE

– Send reset telegram to the identification system.
– Check error status of the identification system.

If data transmission was error-free:

– Return value = DDE_FACK

If an error occurred:

– Reply value = DDE_FNOTPROCESSED
Client receives return value DDE_FNOTPROCESSED and gets error number with "Read Error Number".
Get error text with "Read Error Text".

Asynchronous

Callback routine proceses the following events:

XTYP_EXECUTE (initiated by client)
– Acknowledge: DDE_FACK, DDE_FBUSY or DDE_FNOTPROCESSED *.

XTYP_ADVSTART (initiated by client)
– "Status" = "-01", acknowledge with TRUE.

XTYP_REQUEST (initiated by client)
– Item is "Status", start timer.

WM_TIMER (if timer runs out)
– callup processing function (PostMessage).

XTYP_ADVREQ (called up by server)
– Store "status" in global data object.
– Return value = act on global data object.

Processing function:
– Send reset telegram to identification system.
– Update processing status "Status":
  – Process not finished: "Status" = "-01", PostMessage (processing function)
  – Abort by client: "Status" = "-02", DdePostAdvise
  – Process finished without error: "Status" = "000", DdePostAdvise
  – Process finished with error: "Status" = "<errornr>", DdePostAdvise.
  Get error text with "Read Error Text"

* Client sends DDE request with item "lasterror", server replys with "-03"

Transmit Initialization Information

Synchronous		Asynchronous	
Transaction:	Command String:	Transaction:	Command String:
XTYP_EXECUTE	"CMD=Init LANGUAGE=<path> [EOT=0x<h>]"	XTYP_EXECUTE ADVISE	"CMD=Init LANGUAGE=<path> [EOT=0x<h>] ASYNC" "Status"

<path> Complete path and file name of the language DLL for server error messages.

<h> Terminator for reading data from code tag as Hex string (e.g. EOT=0x2F2F defines the final character "//")

The entry [EOT=0x<h>] is optional and can be left out completely if no EOT is desired.

Prior to data exchange with the identification system, the client sends the server the path and name of the language DLL and optionally a data terminator for reading as initialization information. The language DLL is loded by the server. If a data terminator is indicated, the maximum number of bytes read is the number specified by <count>. If a terminator is recognized during the read, the read is ended at this point (incl. terminator).

The action cannot be aborted by the client. The operating system does permit task changing.	The action can be aborted by the client.

Transmit Initialization Information, Server Reaction

Synchronous

The following events are processed in the callback routine:

XTYP_EXECUTE

– Process initialization information, load language DLL corresponding to <path>.
– Process error status of the identification system.

If no error:

– Return value = DDE_FACK

If error:

– Return value = DDE_FNOTPROCESSED
Client receives return value DDE_FNOTPROCESSED and gets error number with "Read Error Number".
Get error text with "Read Error Text".

Asynchronous

Callback routine proceses the following events:

XTYP_EXECUTE (called up by client)
– Acknowledge: DDE_FACK, DDE_FBUSY or DDE_FNOTPROCESSED *.

XTYP_ADVSTART (called up by client)
– "Status" = "-01", acknowledge with TRUE.

XTYP_REQUEST (called up by client)
– Item is "Status", start timer.

WM_TIMER (when timer runs out)
– Call up processing function (PostMessage).

XTYP_ADVREQ (called up by server)
– Store "Status" in global data object.
– Reply value = act on global data object.

Processing function:
– Process initialization information.
– Update processing status "Status":
  – Processing not finished: "Status" = "-01", PostMessage (processing function)
  – Aborted byclient: "Status" = "-02", DdePostAdvise
  – Processing finished without error: "Status" = "000", DdePostAdvise
  – Processing finished with error: "Status" = "<errornr>", DdePostAdvise.
  Get error text with "Read Error Text".

* Client sends DDE request with item "lasterror", server replys with "-03"

Abort Read/Write Operation

Valid for asynchronous mode only

Transaction:	Command String:
XTYP_EXECUTE	"CMD=Abort"

For aborting a read/write operation at the request of the user.

Reaction from the Server

Event XTYP_EXECUTE is processed in the callback routine:

– Set global abort designator (quieried in the processing function).
– Return value = DDE_FACK

Read Error Number

Has meaning only in synchronous mode
(in asynchronous mode the error number is sent in the processing status)

Transaction:	Request-Item:
XTYP_REQUEST	"GetError"

After a read or write error, read the identification system error number.

Reaction from the Server

Event XTYP_EXECUTE is processed in the callback routine:

– Global error number (provided in the processing function).

Read Error Text

Transaction:	Request-Item:
XTYP_REQUEST	"GetErrortext"

For reading the error text of the identification system after an operation with error. The error texts are stored in the language DLL's in several languages. The current DLL is loaded with the initialization information. The error text then appears in the requested language. The maximum length of the error text is 60 characters.

Reaction of the Server

Event XTYP_EXECUTE is processed in the callback routine:

– Return value = error text of the last occuring error number from the language DLL in the requested language.

Available language DLL's:

German	(Ballu_gr.dll)
English	(Ballu_uk.dll)
French	(Ballu_fr.dll)
Italian	(Ballu_it.dll)
Spanish	(Ballu_sp.dll)
Portuguese	(Ballu_po.dll)

Error Numbers and Error Texts

Error-No.	Error Texts
000	"No error."
001	"No codetag present."
002	"Read error."
003	"Read broken off, codetag was removed too soon."
004	"Write error."
005	"Write broken off, codetag was removed too soon."
006	"Interface error from ID-system recognized."
007	"Telegram format error."
008	"Checksum error between Server and ID-system."
009	"Cable break between processor and read/write head."
010	"Language-DLL (error messages) could not load."
011	"Cannot open COM port."
012	"Command string unknown."
013	"Wrong number of bytes."
014	"Non-BCD-character in received write data."
015	"Cannot close COM port."
016	"Error on initialisation of COM port."
017	"Error on receiving data from COM port."
018	"Error on sending data to COM port."
019	"Startadr. + number of bytes > MAX_CC_CAPACITY in BIS022SV.INI."

Application
Standard Protocol 007

The previous sections have described the basic telegram sequence and configuration. Following is information on how to correctly construct telegrams.

There are specific telegrams for the individual operations in the BIS C identification system. They always begin with the command which corresponds to the telegram type:

Telegram types with associated command (ASCII)

- **'L'** Read code tag with head and block size selection
- **'P'** Write to code tag with head and block size selection
- **'C'** Write a constant value to the code tag with head and block size selection
- **'R'** Read code tag
- **'W'** Write to code tag
- **'H'** Select read/write head and block size with the variants
  - '?' Find next code tag (one-time)
  - '!' or Find next code tag (continuous)
- **'B'** Process outputs
- **'Q'** Restart processor (Quit)
- **'S'** Check status message

Please note:
– Continuous querying on the interface is not permitted!
– The minimum wait time between two commands is 300 ms!

Explanation of selected telegram contents

Start address and no. of bytes	The start address (A3, A2, A1, A0) and the number of bytes to send (L3, L2, L1, L0) are transmitted as ASCII characters. For the start address a range of 0000 to 8191 and for the number of bytes 0001 to 8192 can be used. A3 ... L0 stand for 1 ASCII character each. **Please note:** Start address + number of bytes may not exced the code tag capacity.
Head number and block size	For the 'L' (read with head select and page size) and 'P' (write with head select and page size) commands, first the number of the read/write head K ('1' or '2') and then the block size B ('0', '1') of the code tag are sent. B = '0' corresponds to 64 bytes, B = '1' corresponds to 32 bytes.
Acknowledgement	The <ACK> '0' is sent by the identification system if the serially transmitted characters were correctly recognized and a code tag is within the active zone of a read/write head. FOr the 'R' command, <ACK> '0' is only given if the data are ready for sending. <NAK> + 'Error No.' is sent as an acknowledgement if an error is detected or if there is no code tag within the active zone of a read/write head.
Start	<STX> starts data transmission.
Bytes sent	The data are transmitted code-transparent (not converted).

BCC Block Check

The BCC block check is formed as an EXOR of the serially transmitted binary characters of the telegram block. Example: Read 128 bytes starting at address 13.
The command line without BCC is: 'L 0013 0128 20'. The BCC is formed:

'L	=	0100 1100	EXOR
0	=	0011 0000	EXOR
0	=	0011 0000	EXOR
1	=	0011 0001	EXOR
3	=	0011 0011	EXOR
0	=	0011 0000	EXOR
1	=	0011 0001	EXOR
2	=	0011 0010	EXOR
8	=	0011 1000	EXOR
2	=	0011 0010	EXOR
0'	=	0011 0000	EXOR
Block check result: BCC	=	0100 0111	= 'G'

Variants for finish with BCC, Terminator

If necessary the finish using block check BCC can be replaced with a special ASCII character. This is:

– Carriage Return 'CR'

For hosts which always require a terminator character, this must always be included in the telegrams. Available are:

– Carriage Return 'CR' or
– Line Feed with Carriage Return 'LF CR'.

The various protocol variations are represented on the following page (see also page 13/14).

Description of Various Protocol Variants

Reference is now made to the command string 'L 0013 0128 20 G' with 'G' as BCC (see preceding page). This command string is here shown in its possible variants; also shown are the various forms of acknowledgement with and without terminator:

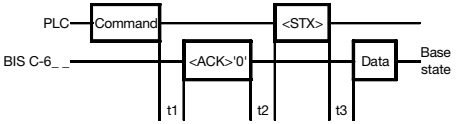
Command line from host system to BIS	Acknowledge from BIS for correct reception	Acknowledge from BIS for incorrect reception
with BCC, but no terminator 'L 0013 0128 20 G'	No terminator <ACK> '0'	No terminator <NAK> '1'
with 'CR' instead of BCC, no terminator 'L 0013 0128 20 CR'	No terminator <ACK> '0'	No terminator <NAK> '1'
no BCC, with terminator 'CR' 'L 0013 0128 20 CR'	with terminator 'CR' <ACK> '0 CR'	with terminator 'CR' <NAK> '1 CR'
no BCC, with terminator 'LF CR' 'L 0013 0128 20 LF CR'	with terminator 'LF CR' <ACK> '0 LF CR'	with terminator 'LF CR' <NAK> '1 LF CR'

For <NAK> with error number a '1' was used here (no code tag present) as an error example.

The respective positions for the additional terminator are shown in the tables in italics.

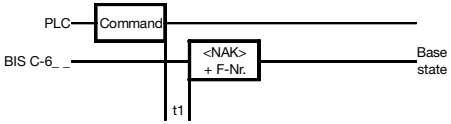
Dialog mode without Head Select

Read: a) If no error:



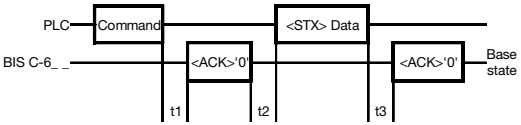
t1 depending on no. of bytes to read (see page 68/69)
t2 ≥ 0 (is not monitored by the processor)
t3 = max. 50 ms

b) With error:



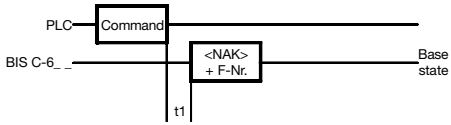
t1 depending on no. of bytes to read (see page 68/69) and error type (recommended monitor time: 15 s)

Write: a) If no error:



t1 = max. 50 ms
t2 ≥ 0 (is not monitored by the processor)
t3 depending on no. of bytes to write (see page 68/69)

b) With error in command:

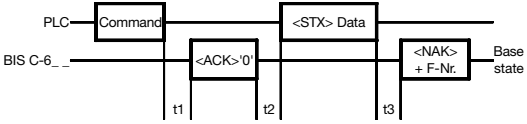


t1 = max. 50 ms
t2 ≥ 0 (is not monitored by the processor)
t3 depending on no. of bytes to write (see page 68/69) and error type (recommended monitor time: 30 s for code tags with 32 byte block size, 60 s for code tags with 64 byte block size)

The examples are valid only if:

- The processor is in the base state.
- A code tag is present in front of a read/write head.

c) With error in writing:

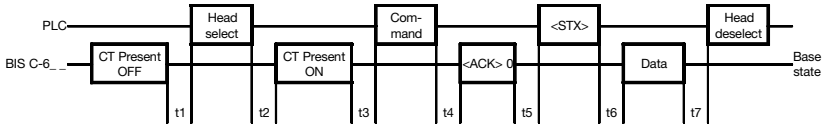


t1 = max. 50 ms

Dialog mode with Head Select

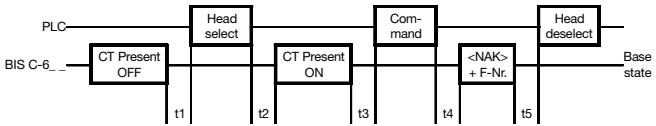
Read:

a) If no error:



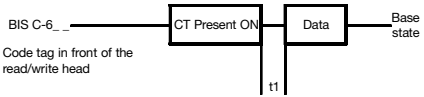
t1, t3, t7 ≥ 0
t2 = max. 500 ms
t4 depending on no. of bytes to read (see page 68/69)
t5 ≥ 0 (is not monitored by the processor)
t6 = max. 50 ms

b) With error:



t1, t3, t5 ≥ 0
t2 = max. 500 ms
t4 depending on no. of bytes to read (see page 68/69) and error type (recommended monitor time: 15 s)

Direct read mode



t1 depending on no. of bytes to read (see page 68/69)

The examples are valid only if:

- The processor is in the base state.
- A code tag is present in front of a read/write head.

Read from code tag with head select and block size
Write to code tag with head select and block size

Task	Data Flow	Com-mand	Start address of first byte to be sent	Number of bytes to be sent	Head No.	Block size	End 2)	Acknow-ledge 3)	*Termi-nator 4)*	Start trans-mission	*Termi-nator 4)*	Data (from start address to start address + no. of bytes)	End 2)	Acknow-ledge 3)	*Termi-nator 4)*
Read	from host system to BIS	**'L'**	A3 A2 A1 A0 '0 0 0 0' to '8 1 9 1'	L3 L2 L1 L0 '0 0 0 1' to '8 1 9 2'	K '1' or '2'	B '0' or '1'	BCC or see 2)			<STX>	*'CR' or 'LF CR'*				
	from BIS to host system							<ACK>'0' or <NAK> + Error-No.	*'CR' or 'LF CR'*			D1 D2 D3 ... Dn	BCC or see 2)		
			1)								1)				
Write	from host system to BIS	**'P'**	A3 A2 A1 A0 '0 0 0 0' to '8 1 9 1'	L3 L3 L1 L0 '0 0 0 1' to '8 1 9 2'	K '1' or '2'	B '0' or '1'	BCC or see 2)			<STX>		D1 D2 D3 ... Dn	BCC or see 2)		
	from BIS to host system							<ACK>'0' or <NAK> + Error-No.	*'CR' or 'LF CR'*					<ACK>'0' or <NAK> + Error-No.	*'CR' or 'LF CR'*
			1)									1)			

1) The commands Status and/or Quit are not permitted at this point.
2) Instead of block check BCC, depending on protocol variant either Carriage Return 'CR' or Line Feed with Carriage Return may be used.
3) <ACK> '0' is returned as acknowledgement if there is no error, or <NAK> + 'Error No.' if an error occurs.
4) For protocol variants which always require a terminator, either 'CR' or 'LF CR' must be inserted here as a terminator.

Data within angle brackets are control characters.
Values inside apostrophes represent the respective character(s) in ASCII code.

Telegram example for page 52:
Read from code tag with head select and block size
with block check (BCC)

-> Head 1 is selected. Read 10 bytes starting at address 50 of the code tag at read/write Head 2. The code tag at Head 4 has a block size of 64 bytes.

The host sends 'L 0 0 5 0 0 0 1 0 2 0 J' BCC ($4A_{HEX}$)

Address of first byte to read
Number of bytes to read
Read/write Head No. 2
Block size 0 = 64 bytes

The BIS processor acknowledges with <ACK> '0'
The host system gives the start command <STX>
The BIS processor provides the data from the code tag '1 2 3 4 5 6 7 8 9 A F' BCC (70_{HEX})
After the telegram sequence, Head 2 remains selected, with 64 byte block size.

Telegram example for page 52:
Write to code tag with read/write head select and block size
with block check (BCC)

-> Head 1 is selected. Write 5 bytes starting at address 500 of the code tag at read/write Head 2. The code tag at Head 2 has a block size of 64 bytes.

The host sends 'P 0 5 0 0 0 0 0 5 2 0 R' BCC (52_{HEX})

Address of first byte to write
Number of bytes to write
Read/write Head No. 2
Block size 0 = 64 Byte

The BIS processor acknowledges with <ACK> '0'
The host system gives the start command and data <STX> '1 2 3 4 5 3' BCC (33_{HEX})
The processor acknowledges with <ACK> '0'
After the telegram sequence, Head 2 remains selected, with 64 byte block size.

Data within angle brackets are control characters.
Values inside apostrophes represent the respective character(s) in ASCII code.

Writing a constant value in the code tag with read/write select and block size

This command can be used to erase a code tag data. One saves the time for the transmission of the write byte.

Task	Data Flow	Com-mand	Start address of first byte to be sent	Number of bytes to be sent	Head No.	Block size	End 2)	Acknow-ledge 3)	*Termi-nator 4)*	Start trans-mission	*Termi-nator 4)*	Data (from start address to start address + no. of bytes)	End 2)	Acknow-ledge 3)	*Termi-nator 4)*
Write	from host system to BIS	**'C'**	A3 A2 A1 A0 '0 0 0 0' to '8 1 9 1'	L3 L3 L1 L0 '0 0 0 1' to '8 1 9 2'	K '1' or '2'	B '0' or '1'	BCC or see 2)			<STX>		D	BCC or see 2)		
	from BIS to host system							<ACK>'0' or <NAK> + Error-No.	*'CR' or 'LF CR'*					<ACK>'0' or <NAK> + Error-No.	*'CR' or 'LF CR'*
			1)									1)			

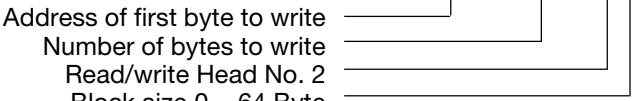
1) The commands Status and/or Quit are not permitted at this point.
2) Instead of block check BCC, depending on protocol variant either Carriage Return 'CR' or LIne Feed with Carriage Return may be used.
3) <ACK> '0' is returned as acknowledgement if there is no error, or <NAK> + 'Error No.' if an error occurs.
4) For protocol variants which always require a terminator, either 'CR' or 'LF CR' must be inserted here as a terminator.

Data within angle brackets are control characters.
Values inside apostrophes represent the respective character(s) in ASCII code.

Telegram example for page 54:
Write to code tag with read/write head select and block size *with block check (BCC)*

-> Head 1 is selected. Write 500 bytes starting at address 20 of the code tag at read/write Head 2 mit dem ASCII Datenwert 0 (30_{HEX}). The code tag at Head 2 has a block size of 64 bytes.

The host sends 'C 0020 0500 20F' BCC (46_{HEX})



The BIS processor acknowledges with <ACK> '0'
The host system gives the start command and data <STX> '02' BCC (32_{HEX})
The processor acknowledges with <ACK> '0'
After the telegram sequence, Head 2 remains selected, with 64 byte block size.

Data within angle brackets are control characters.
Values inside apostrophes represent the respective character(s) in ASCII code.

Read from Code Tag, Write to Code Tag

Task	Data Flow	Com-mand	Start address of first byte to send	Number of bytes to send	End 2)	Acknow-ledge 3)	*Termi-nator 4)*	Start trans-mission	*Termi-nator 4)*	Data (from start address to start address + no. of bytes)	End 2)	Acknow-0 ledge 3)	*Termi-nator 4)*
Read	from host system to BIS	**'R'**	A3 A2 A1 A0 '0 0 0 0' to '8 1 9 1'	L3 L3 L1 L0 '0 0 0 1' to '8 1 9 2'	BCC or see 2)			<STX>	*'CR' or 'LF CR'*				
	from BIS to host system					<ACK>'0' or <NAK> + Error-No.	*'CR' or 'LF CR'*			D1 D2 D3 ... Dn	BCC or see 2)		
			1)										
Write	from host system to BIS	**'W'**	A3 A2 A1 A0 '0 0 0 0' to '8 1 9 1'	L3 L3 L1 L0 '0 0 0 1' to '8 1 9 2'	BCC or see 2)			<STX>		D1 D2 D3 ... Dn	BCC or see 2)		
	from BIS to host system					<ACK>'0' or <NAK> + Error-No.	*'CR' or 'LF CR'*					<ACK>'0' or <NAK> + Error-No.	*'CR' or 'LF CR'*
			1)							1)			

1) The commands Status and/or Quit are not permitted at this point.
2) Instead of block check BCC, depending on protocol variant either Carriage Return 'CR' or LIne Feed with Carriage Return may be used.
3) <ACK> '0' is returned as acknowledgement if there is no error, or <NAK> + 'Error No.' if an error occurs.
4) For protocol variants which always require a terminator, either 'CR' or 'LF CR' must be inserted here as a terminator.

Data within angle brackets are control characters.
Values inside apostrophes represent the respective character(s) in ASCII code.

Telegram example from page 56:
Read from Code Tag
with block check (BCC)

Read from Code Tag: -> Read 10 bytes starting at address 50.

The host sends	'R 0 0 5 0 0 0 1 0 V' BCC (56HEX)
Address of first byte to read	
Number of bytes to read	
The BIS processor acknowledges with	<ACK> '0'
The host gives the start command	<STX>
The BIS processor provides the data from the code tag	'1 2 3 4 5 6 7 8 9 0 SOH' BCC (01HEX)

Telegram example from page 56:
Write to Code Tag
with block check (BCC)

Write to Code Tag: -> Write 5 bytes starting at address 500.

The host system sends	'W 0 5 0 0 0 0 0 5 W' BCC (57HEX)
The BIS processor acknowledges with	<ACK> '0'
The host sends the data	<STX> '1 2 3 4 5 3' BCC (33HEX)
The BIS processor acknowledges with	<ACK> '0'

The 'R' and 'W' commands represent a subtype of the 'L' and 'P' commands.

Data within angle brackets are control characters.
Values inside apostrophes represent the respective character(s) in ASCII code.

Selecting a Read/Write Head

The 'H1' command selects Read/Write Head 1, 'H2' Read/Write Head 2, and 'HT' (Head Twin) both Read/Write Heads.

If both heads are selected, please note:

1. Only one code tag is allowed to be in the active zone of a read/write head at a time.
2. The read or write time increases by ca. 40 ms – regardless of the data amount to be read or written. (This does not apply to the code tag recognition).
3. The positive acknowledgement for a read or write action is no longer <ACK> '0' but rather <ACK> '1' or <ACK> '2', depending on at which read/write head there is a code tag to be read from or written to.

Task	Data Flow	Com-mand	Head number	End 2)	Acknowledge 3)	*Terminator 4)*
Select Read/Write Head	from host system to BIS	**'H'**	'1', '2' or 'T'	BCC or see 2)		
	from BIS to host system				<ACK>'1' or <ACK>'2' resp. <NAK> + Error-No.	*'CR' or 'LF CR'*
			1)			

1) The commands Status and/or Quit are not permitted at this point.
2) Instead of block check BCC, depending on protocol variant either Carriage Return 'CR' or LIne Feed with Carriage Return may be used.
3) <ACK> '0' is returned as acknowledgement if there is no error, or <NAK> + 'Error No.' if an error occurs.
4) For protocol variants which always require a terminator, either 'CR' or 'LF CR' must be inserted here as a terminator.

Telegram example:
Selecting a Read/ Write Head
with block check (BCC)

-> Switch to Head 1.

The host sends	'H 1	y'	BCC (79_{HEX})
The BIS processor acknowledges with	<ACK> '0'		

Data within angle brackets are control characters.
Values inside apostrophes represent the respective character(s) in ASCII code.

Find Next Code Tag (one time)

The following telegram is used to find the next code tag. The next following read/write head is selected and checked to see if a code tag is in front of this read/write head. If yes, the first 4 bytes of the code tag are read. The telegram reply then contains the corresponding number of the read/write head and the four bytes read. If no tag is found, the original read/write head is reselected and checked. If no code tag is found here, then the telegram reply is: 'H ? 0000 w'.

'H ?' recognizes any code tag, regardless of the preset block size, assuming that read/write head and code tag are compatible.

Task	Data Flow	Com-mand	Des.	End 2)	Acknow-ledge	*Terminat-or 3)*	Reply	Head number	Data from code tag	End 2)
Find next code tag (one time)	from host system to BIS	**'H'**	'?'	BCC or see 2)						
	from BIS to host system				<ACK>'0'	*'CR' or 'LF CR'*	'H'	'1', '2' or '?'	D1 D2 D3 D4	BCC or see 2)
			1)							

1) The commands Status and/or Quit are not permitted at this point.
2) Instead of block check BCC, depending on protocol variant either Carriage Return 'CR' or LIne Feed with Carriage Return may be used.
3) For protocol variants which always require a terminator, either 'CR' or 'LF CR' must be inserted here as a terminator.

Telegram example:
Find Next Code Tag (one time)
with block check (BCC)

-> Head 1 is selected. Only read/write head 2 has a code tag in front of it, whose first four bytes are 9876.

The host sends	'H ?	w' BCC (77_{HEX})
The BIS processor acknowledges with	<ACK> '0'	
and sends the data	'H 2 9 8 7 6	z' BCC ($7A_{HEX}$)

Data within angle brackets are control characters.
Values inside apostrophes represent the respective character(s) in ASCII code.

Find Next Code Tag (continuous)

The following telegram is used to find the next code tag. The next following read/write head is selected and checked to see if a code tag is in front of this read/write head. If yes, the first four bytes of the code tag are read. The telegram reply then contains the corresponding number of the read/write head and the four bytes read. If no tag is found, the original read/write head is reselected and checked. This procedure is repeated until a code tag is found.

H !' recognizes any code tag, regardless of the preset block size, assuming that read/write head and code tag are compatible.

Task	Data Flow	Command	Designator	End 2)	Acknowledge	*Terminator 3)*	Reply	Head number	Data from code tag	End 2)
Find next code tag (contin.)	from host system to BIS	**'H'**	'!'	BCC or see 2)						
	from BIS to host system				<ACK>'0'	*'CR' or 'LF CR'*	'H'	'1' or '2'	D1 D2 D3 D4	BCC or see 2)
			1)							

1) The commands Status and/or Quit are not permitted at this point.
2) Instead of block check BCC, depending on protocol variant either Carriage Return 'CR' or LIne Feed with Carriage Return may be used.
3) For protocol variants which always require a terminator, either 'CR' or 'LF CR' must be inserted here as a terminator.

Telegram example:
Find Next Code Tag (continuous)
with block check (BCC)

-> Read/write head 2 has a code tag in front of it whose first four bytes are 9876.

The host sends 'H ! i' BCC (69HEX)

The BIS processor acknowledges with <ACK> '0'
and sends the data 'H 2 9 8 7 6 z' BCC (7AHEX)

Data within angle brackets are control characters.
Values inside apostrophes represent the respective character(s) in ASCII code.

Processing the Outputs

A telegram can be sent to set or cancel the four outputs.

Task	Data Flow	Com-mand	Designator	Termi-nator 2)	Acknow-ledge	Termi-nator 3)
Process outputs (set or cancel)	from host system to BIS	**'B'**	'00' bis 'A1' (see below)	BCC or see 2)		
	from BIS to host system				<ACK>'0'	'CR' or 'LF CR'
			1)			

Designator meaning:	Output No.	0	1	2	3	all outputs
	Cancel output	00	10	20	30	A0
	Set output	01	11	21	31	A1

1) The commands Status and/or Quit are not permitted at this point.
2) Instead of block check BCC, depending on protocol variant either Carriage Return 'CR' or LIne Feed with Carriage Return may be used.
3) For protocol variants which always require a terminator, either 'CR' or 'LF CR' must be inserted here as a terminator.

Telegram example:
Processing the Outputs
with block check (BCC)

The host sends	'B 21	A'	BCC (41$_{HEX}$)
The BIS processor acknowledges with	<ACK> '0'		

After the telegram is completed, output 2 is set.

Data within angle brackets are control characters.
Values inside apostrophes represent the respective character(s) in ASCII code.

Show Output Condition

This telegram is used to check the condition (status) of all four outputs.

Task	Data Flow	Com-mand	Desig-nator	End 2)	Acknow-ledge	Condition of the 4 outputs	Termi-nator 3)	End 2)
Show output condition	from host system to BIS	**'B'**	'AO'	BCC or see 2)				
	from BIS to host system				<ACK>'0'	'XXXX' '0' =cancelled, '1' = set	'CR' or 'LF CR'	BCC or see 2)
			1)					

Output status is shown in sequential order 0 1 2 3

1) The commands Status and/or Quit are not permitted at this point.
2) Instead of block check BCC, depending on protocol variant either Carriage Return 'CR' or LIne Feed with Carriage Return may be used.
3) For protocol variants which always require a terminator, either 'CR' or 'LF CR' must be inserted here as a terminator.

Telegram example:
Show Output Condition
with block check (BCC)

-> Outputs 0 and 1 are set, outputs 2 and 3 are cancelled.

The host sends	'B AO	L' BCC ($4C_{HEX}$)
The BIS processor acknowledges with	<ACK> '0'	
and sends the data	'1100	NUL' BCC (00_{HEX})

Data within angle brackets are control characters.
Values inside apostrophes represent the respective character(s) in ASCII code.

Restart the Processor (Quit)

Sending the Restart command causes a telegram in process to be aborted and puts the processor in the ground state. After this telegram is acknowledged, a minimum of 1600 ms pause should be allowed before starting a new telegram.

Important! The Quit command is not permitted while the processor is waiting for a terminator (BCC, 'CR' or 'LF CR'). In this situation, the Quit would be incorrectly interpreted as a terminator or datum.

Task	Data Flow	Command	Terminator 2)	Acknowledge	Terminator 2)
Restart (Quit)	from host system to BIS	**'Q'**	BCC or see 2)		
	from BIS to host system			'Q'	BCC or see 2)
			1)		

1) The commands Status and/or Quit are not permitted at this point.
2) Instead of block check BCC, depending on protocol variant either Carriage Return 'CR' or LIne Feed with Carriage Return may be used.

Telegram example with block check (BCC):

Put the BIS system into the ground state.

The host sends 'Q Q' BCC (51HEX)

The BIS processor acknowledges with 'Q Q' BCC (51HEX)

Data within angle brackets are control characters.
Values inside apostrophes represent the respective character(s) in ASCII code.

Querying the status message

The status telegram checks to see what kind of telegram is in process.

Important! The Status command is not permitted while the processor is waiting for a termination character (BCC, 'CR' or 'LF CR'). In this situation Status would be misinterpreted as a termination or data character.

Important: A status check during a red or write operation on a code tag (Codetag Operating LED on) increases the read or write time. Especially in dynamic mode this can result in insufficient time for a full read or write while the tag is in the active zone of the read/write head. Continuous status checking disturbs the processing of the code tag, and the tag may not be recognized!

The characters between the apostrophes represent the respective character(s) in ASCII code. '_' = Space = ASCII 20HEX.

Task	Data Flow	Command	Terminator 2)	Status message	Terminator 2)
Check Status Message	From host system to BIS	**'S'**	BCC or see 2)		
	From BIS to host system			'S' '_', 'R', 'W', 'L', 'P' or 'H'	BCC or see 2)
			1)		

1) The Status and/or Quit commands are not permitted at this point.
2) Instead of BCC block check, depending on the protocol variant either Carriage Return 'CR' or Line Feed with Carriage Return 'LF CR' can be used.

Status messages and their meaning:

'S L'	=	Read code tag with head select and block size
'S P'	=	Write to code tag with head select and block size
'S R'	=	Read from code tag
'S W'	=	Write to code tag
'S H'	=	Select head and block size
'S _'	=	No telegram in process

Telegram examples for page 64:
Query status message
with (BCC) blockcheck

-> To check the BIS status just after a **read telegram** has been sent.

Host sends	'S S'	BCC (53_{HEX})
BIS acknowledges with	'S L US'	BCC ($1F_{HEX}$)

-> To check the BIS status just after a **write telegram** has been sent.

Host sends	'S S'	BCC (53_{HEX})
BIS processor acknowledges with	'S P ETX'	BCC (03_{HEX})

-> To check the BIS status just after a **Select read/write head telegram** has been sent.

Host sends	'S S'	BCC (53_{HEX})
BIS processor acknowledges with	'S H ESC'	BCC ($1B_{HEX}$)

-> To check the BIS status when **no telegram** has just been sent.

Host sends	'S S'	BCC (53_{HEX})
BIS processor acknowledges with	'S _'	BCC (20_{HEX})

Error Numbers

The BIS C-600 always outputs an error number. The meaning of these error numbers is indicated in the following table.

No.	Error Description	Effect
1	No code tag present	Telegram aborted, processor goes into ground state.
2	Read error	Read telegram aborted, processor goes into ground state.
3	Read aborted, since the code tag was removed	Processor goes into ground state.
4	Write error	Write telegram aborted, processor goes into ground state. CAUTION: Some new data may still have been written to the code tag!
5	Writing aborted,since the code tag was removed	Processor goes into ground state. CAUTION: Some new data may still have been written to the code tag!
6	Interface error	Processor goes into ground state. (parity or stop bit error)
7	Telegram format error	Processor goes into ground state. Possible format errors: - Command is not 'L'/'P'/'C'/'R'/'W'/'H'/'B'/'Q'/'S'. - Start address or number of bytes exceed permissible range

Error Numbers
(continued)

8	BCC error, the transmitted BCC is wrong	Telegram is aborted, processor goes into ground state.
9	Cable break, Codetag Present LED flashes	Telegram is aborted, processor goes into ground state. Cable break from read/write head or cable not connected. If both read/write heads were selected using 'HT', one head may not be connected. If both read/write heads are selected, the cable break message only comes if there is no code tag in front of the connected, functional head.
A	New command not possible, since a read command is already in process	After error message the read command is stopped, internally, but not acknowledged. Processor goes into ground state.
B	New command not possible, since a write command is already in process.	After error message the write command is stopped, internally, but not acknowledged. Processor goes into ground state. CAUTION: If errors occur after new attempts to write to the code tag, no further error messages will be given.
C	New command not possible, since a head select is already in process.	After the error message, no positive acknowledge is given, even thought the head select was successful. Processor goes into ground state.

Read/Write Times

Read Times in Static Mode (Configuration: without dynamic mode)

For double read and compare:

Code tag with 32 byte blocks	
No. of bytes	Read time [ms]
from 0 to 31	110
for each additional 32 bytes add	120
from 0 to 255	= 950

Code tag with 64 byte blocks	
No. of bytes	Read time [ms]
from 0 to 63	220
for each additional 64 bytes add	230
from 0 to 2047	= 7350

Write Times in Static Mode (Configuration: without dynamic mode)

Including readback and compare:

Code tag with 32 byte blocks	
No. of bytes	Write time [ms]
from 0 to 31	110 + n * 10
for 32 bytes or more	y * 120 + n * 10

Code tag with 64 byte blocks	
No. of bytes	Write time [ms]
from 0 to 63	220 + n * 10
for 64 bytes or more	y * 230 + n * 10

n = number of contiguous bytes to write
y = number of blocks to be written

Example:
Read 17 bytes starting at address 187. Code tag with 32 byte blocks.
Blocks 5 and 6 have to be accessed, because start address 187 is in block 5 and end address 203 is in block 6.
t = 2 * 120 + 17 * 10 = 410

The indicated times apply after the code tag has been recognized. Otherwise an additional 45 ms must be added to allow for the energy field to be established until the code tag is recognized.

Read/Write Times

Read Times in Dynamic Mode (Configuration: with dynamic mode)

Read Times for 1 Block with double read and compare:

Code tag with 32 byte blocks	
No. of bytes	Read time [ms]
from 0 to 3	14
for each additional byte add	3.5
from 0 to 31	112

Code tag with 64 byte blocks	
No. of bytes	Read time [ms]
from 0 to 3	14
for each additional byte add	3.5
from 0 to 63	224

m = highest address to be read

Formula: $t = (m + 1) * 3.5$ ms

Example: Read 11 bytes starting at address 9. Hence the highest address to be read is 19. This results in 70 ms.

Write Times in Dynamic Mode (Configuration: with dynamic mode)

Including readback and compare:

Code tag with 32 byte blocks	
No. of bytes	Write time [ms]
from 0 to 31	14 + n * 10
for each additional byte add	3.5

Code tag with 64 byte blocks	
No. of bytes	Write time [ms]
from 0 to 63	14 + n * 10
for each additional byte add	3.5

n = Number of contiguous bytes to be written

The indicated times apply after the code tag has been recognized. Otherwise an additional 45 ms must be added to allow for the energy field to be established until the code tag is recognized.

LED Display

LED Display:
System Ready
Codetag Present
Codetag Operating

The BIS C-600 Processor uses three LED's on the front panel to indicate the most important operating conditions.

Condition	LED	Meaning
System Ready	on (green)	Supply voltage OK; no hardware error.
	off	Supply voltage or hardware not OK, or read/write head cable break or not connected.
Codetag Present	on (yellow)	Code tag ready to read or write. (If a read/write error occurs during a read/write operation, System Ready LED goes out, if the protocol variant "without error number" is used!)
	flashes	Read/write head cable break or not connected. If both read/write heads were selected using 'HT', one head may not be connected. If both read/write heads are selected, the cable break message only comes if there is no code tag in front of the connected, functional head.
	off	No code tag in active zone of read/write head.
Codetag Operating	on (yellow)	Command being processed.
	off	No command in process.

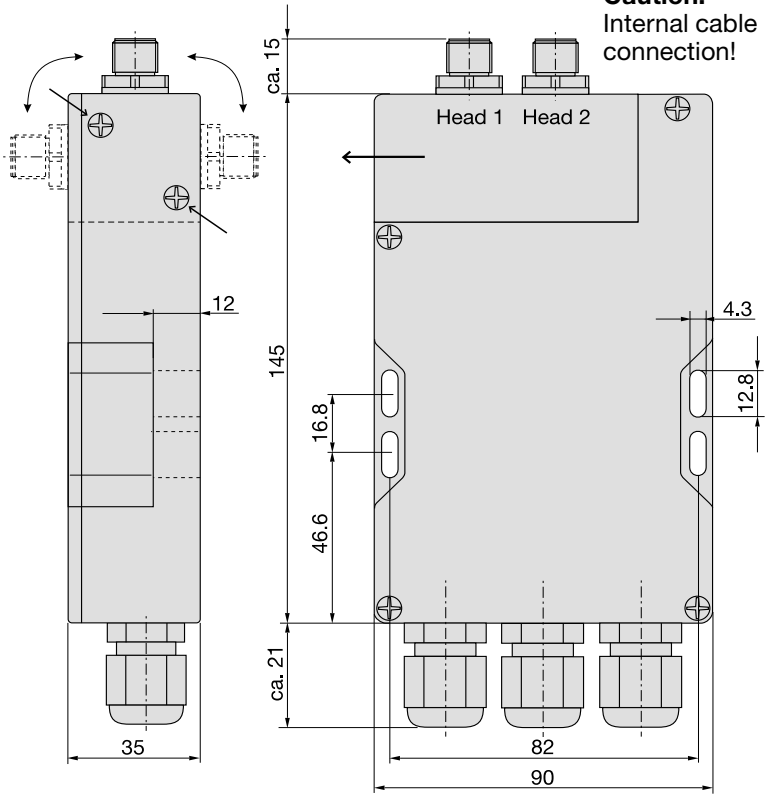
If all three LED's flash on and off in synch, the processor needs to be returned to the factory for repair.

Assembling the BIS C-600 processor and configuring the read/write head or BIS C-650 adapter

The processor is attached at the 4 side through-holes.

Depending on the processor version either a read/write head or the adapter for remote read/write head is fitted. Both the read/write head and the adapter can be rotated by the user by + or - 90° to the desired orientation (see illustration). Be sure that the device is turned off. Remove both screws (indicated by arrows in the illustration). Carefully pull the head or adapter out towards the side (direction of arrow, right illustration). **Caution: Internal cable connection!** Attach it in the desired position and tighten down with the screws.

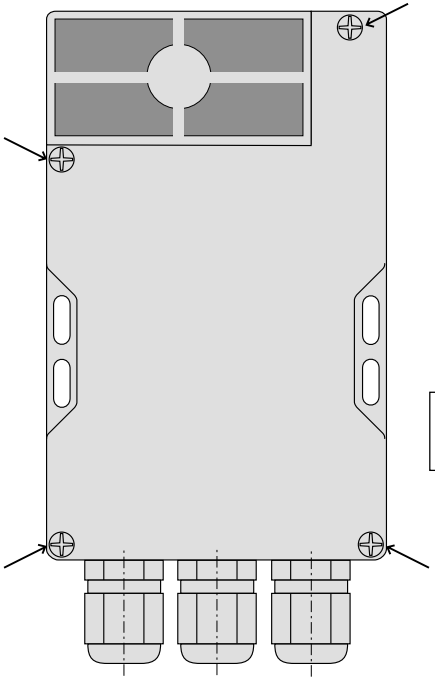
Caution:
Internal cable connection!



Opening the processor BIS C-600

The BIS C-600 processor must be opened in order to make the connections.

Ensure that the device is turned off. Remove the 4 screws on the BIS C-600 and lift off the cover. Guide the connection cable through the cable fittings. See following pages for additional details.



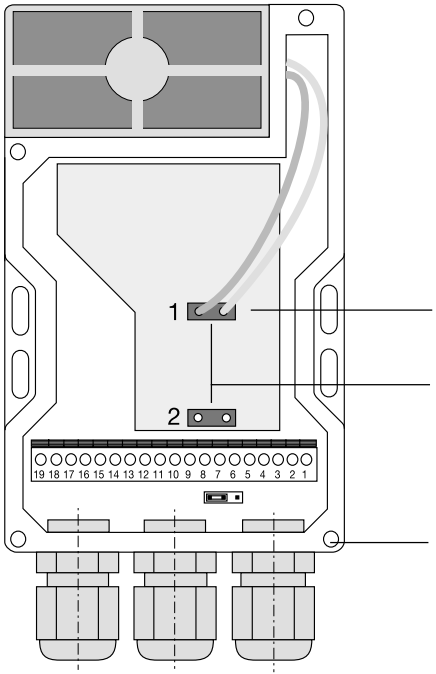
Attaching the cover (4 screws),
max. permissible torque 0.15 Nm

Write your configuration data on the sticker supplied and apply it to the interior of the housing cover.

Opening the processor

Attaching a read/ write head or BIS C-650/-670 adapter

To change out a read/write head: Turn processor off and open. Remove mounting screws on the read/write head (see page 71) and remove processor cover (see page 72). Unplug the read/write head from the circuit board and draw the connection cable out through the cable opening. To attach the new head, proceed in reverse.



To attach the adapter, proceed as described above. For the BIS C-650 both connection cables must be plugged into the circuit board.

Caution! When connecting the BIS C-650 adapter, no additional read/write head is permitted to be connected to the terminal strip.

Connecting the integrated read/write head or BIS C-670 adapter

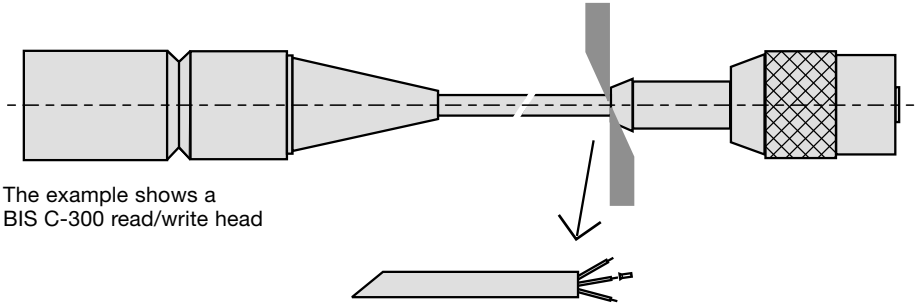
Connections for the BIS C-650 adapter
1 = Head 1
2 = Head 2

Attaching the cover (4 screws), max. permissible torque 0.15 Nm

Preparing the read/ write head for connecting to the BIS C-600 terminal strip

To attach a series BIS C-3_ _ read/write head (except BIS C-350 and -352) to the BIS C-600 terminal strip, the connector on the cable end must first be removed.

Please note that the cable must be cut right at the connector as shown below, since the cable length and the read/write head function are mutually dependent. Reliable data transmission cannot be guaranteed if the cable length is altered. The read/write head cable may not exceed 5 m in length.



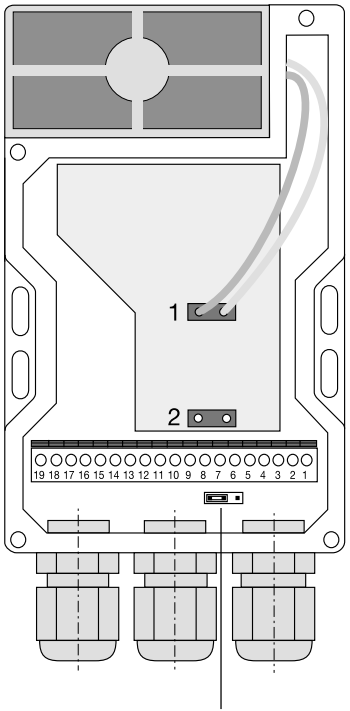
The example shows a
BIS C-300 read/write head

Wiring the 2nd read/ write head to the BIS C-600 processor

Wire color	Function	BIS C-60_ terminal block
BU	AH	15
BN	EH	14
Shield	GND	16

Strip the cable insulation to a length of approx. 5 cm. Strip the lead ends to a length of approx. 5 mm and attach crimp terminals sized for AWG 24 or 22.

Wiring diagram for BIS C-600 processors with integrated read/write head



Terminal strip

Shunt for handshake setting when using RS 232 (see Wiring diagrams on following pages)

Connection names and locations

19	18	17	16	15	14
+VS	–VS	⏚	⏚	AH	EH
POWER			HEAD #2		

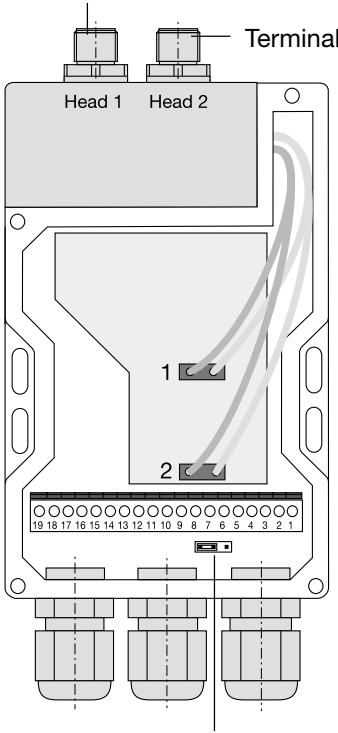
13	12	11	10	9	8	7	6
+VS	–VS	1	2	3	4	+IN	–IN
OUTPUT						INPUT	

5	4	3	2	1	
COM	RxD	CTS	TxD	RTS	
RS 232					BIS C-600...00

5	4	3	2	1	
n.c.	RxD+	RxD-	TxD+	TxD-	
	20 mA (TTY)				BIS C-600...01

Terminal assignments
n.c. = leave empty!

Wiring diagram for BIS C-600 processors with adapter BIS C-650



Connection names and locations

19	18	17	16...14
+VS	–VS	⏚	n.c.
POWER			

13	12	11	10	9	8	7	6
+VS	–VS	1	2	3	4	+IN	–IN
OUTPUT						INPUT	

5	4	3	2	1
COM	RxD	CTS	TxD	RTS
RS 232				

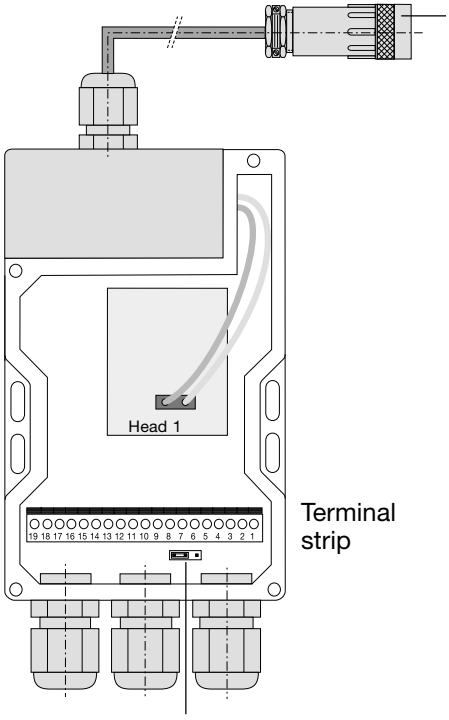
BIS C-600...00

5	4	3	2	1
n.c.	RxD+	RxD-	TxD+	TxD-
	20 mA (TTY)			

BIS C-600...01

Terminal assignments
n.c. = leave empty!

Wiring diagram for BIS C-600 processors with adapter BIS C-670



Connection names and locations

19	18	17	16...14
+VS	–VS	⏚	n.c.
POWER			

13	12	11	10	9	8	7	6
+VS	–VS	1	2	3	4	+IN	–IN
OUTPUT						INPUT	

5	4	3	2	1
COM	RxD	CTS	TxD	RTS
RS 232				

BIS C-600...00

5	4	3	2	1
n.c.	RxD+	RxD-	TxD+	TxD-
	20 mA (TTY)			

BIS C-600...01

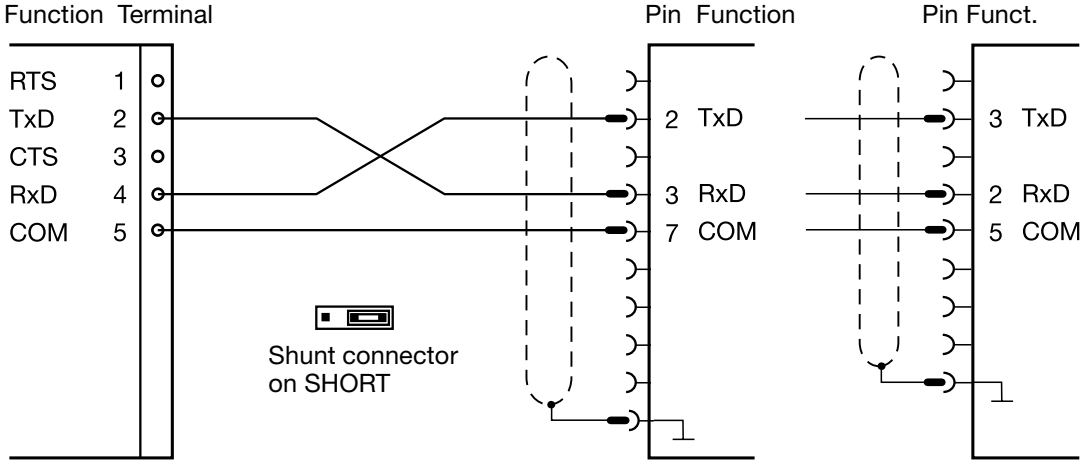
Terminal assignments
n.c. = leave empty!

Interface functions

A serial port is provided for connecting the BIS C-600 processor to a host system (e.g., PC or PLC). Depending on the version the following configurations are available:

BIS C-600...00 RS 232 interface (V.24) or
BIS C-600...01 20-mA current loop interface (TTY).

Interface RS 232 (V.24) w/out hardware handshake



Connection to processor terminal strip

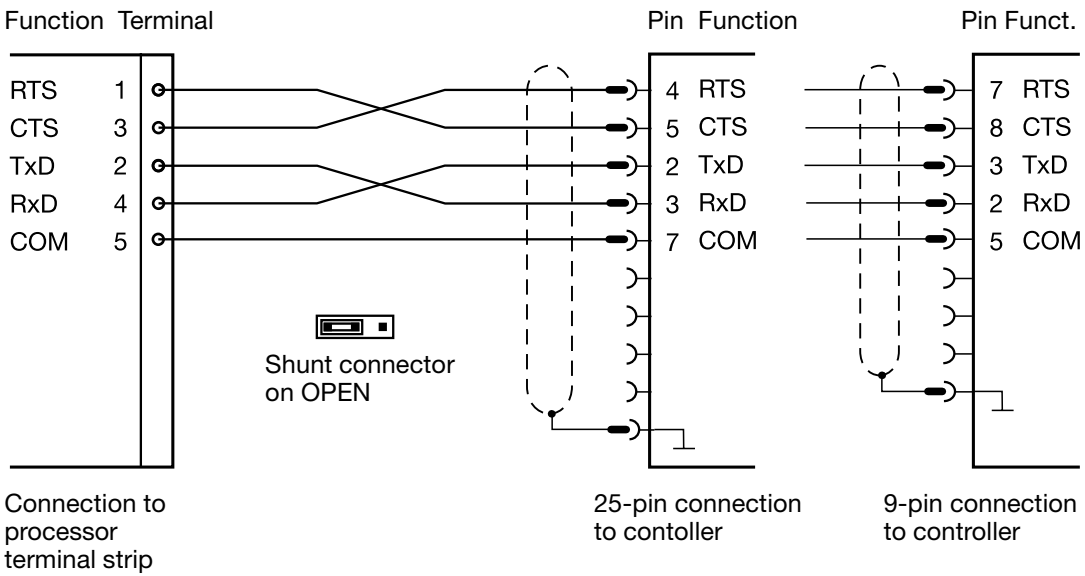
25-pin connection to processor

9-pin connection to controller

OPEN / SHORT

CTS  Factory setting = SHORT.
Since the CTS signal is not used, the shunt connector remains in the SHORT position.

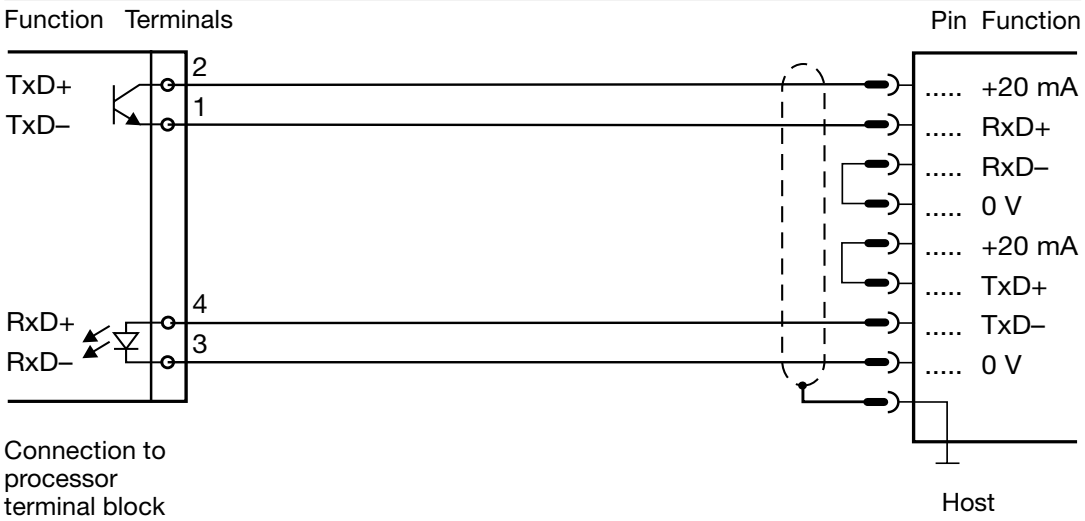
Interface RS 232 (V.24) with hardware handshake



OPEN / SHORT

CTS  Factory setting = SHORT.
Since the CTS signal is used, the shunt connector is set to the OPEN position.

20 mA Interface (TTY), Processor passive



You may write in the pin numbers for wiring to your host in the spaces above.

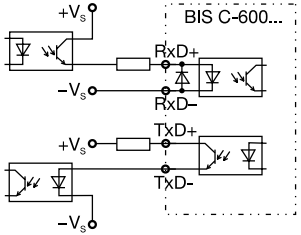
BIS C-600
Technical Data

Dimensions, Weight	**Housing**	Plastic PS
	Dimensions with BIS C-652 read/write head	approx. 169 x 90 x 35 mm
	Dimensions with BIS C-650 adapter	approx. 184 x 90 x 35 mm
	Weight	approx. 400 g
Temperature range	Ambient temperature	0 °C to +60 °C
Connections	Terminal strip	19-point
	Cable fitting	3 x clamping PG 9
	Cable diameter	4 to 8 mm
	Wire gauges	AWG 26 to AWG 17
	with crimp contacts	AWG 24 to AWG 22
Protection	Protection	IP 65 (when connected)
Electrical connections	**Supply voltage V_S, input**	DC 24 V ± 20 %
	Ripple	≤ 10 %
	Current draw	≤ 400 mA
	Read/write head	integrated, BIS C-65_ and following*);
	alternate for BIS C-650 adapter *)	2 x integrated connectors 4-pin (male) for all BIS C-3_ _ read/write heads with 4-pin connector (female), **not** BIS C-350 and BIS C-352
	alternate for BIS C-670 adapter *)	1 x 8-pin male connector for one of the read/write heads BIS C-350 and BIS C-352
	Serial interface	RS 232 (V.24) or 20-mA current loop (TTY)

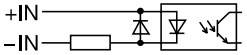
*) can be rotated by ± 90°

Electrical connections (cont.)

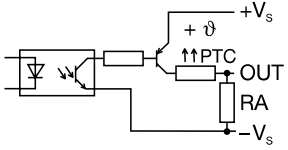
20-mA current loop (TTY)	optocoupler isolated
Receiver	Current loop, 20 mA passive
Quiescent current	20 to 50 mA
Voltage drop at 20 mA	approx. 3 V
Sender	Current loop, 20 mA passive
Idle voltage V_S	max. 50 V
Voltage drop at 20 mA	approx. 3 V
Line current	max. 50 mA



Digital input (+IN, –IN)	optocoupler isolated
Control voltage active	4 V to 40 V
Control voltage inactive	1.5 V to –40 V
Input current at 24 V	11 mA
Typical delay time	5 ms



Control outputs (01 to 04)	optocoupler isolated
Output current max. 20 mA	
Voltage drop at 20 mA	approx. 2.5 V
Output resistance R_A	10 kΩ to V_S
Output circuit	PNP (current sourcing)
Supply voltage, output V_S	DC 24 V ± 20 %
Ripple	≤ 10 %



Function displays

System Ready	LED green
Codetag Present	LED yellow
Codetag Operating	LED yellow

With the CE Mark we affirm that our products are in accordance with the requirements of the EU (European Union) Guideline
89/336/EEC (EMC Guideline)

and the EMC Law. It has been verified in our EMC Laboratory, which is accredited by the DATech for Testing of Electromagnetic Compatibility, that Balluff products meet the EMC requirements of the Harmonized Standard

EN 50081-2 (Emission), EN 50082-2 (Noise Immunity)

BIS C-600
Ordering Information

Ordering Code

BIS C-600-_ _ _-_ _ _-_ _-KL1

Balluff Identification System

Series C

Hardware type
600 = Compact

Software type
022 = Server software
007 = Balluff protocol

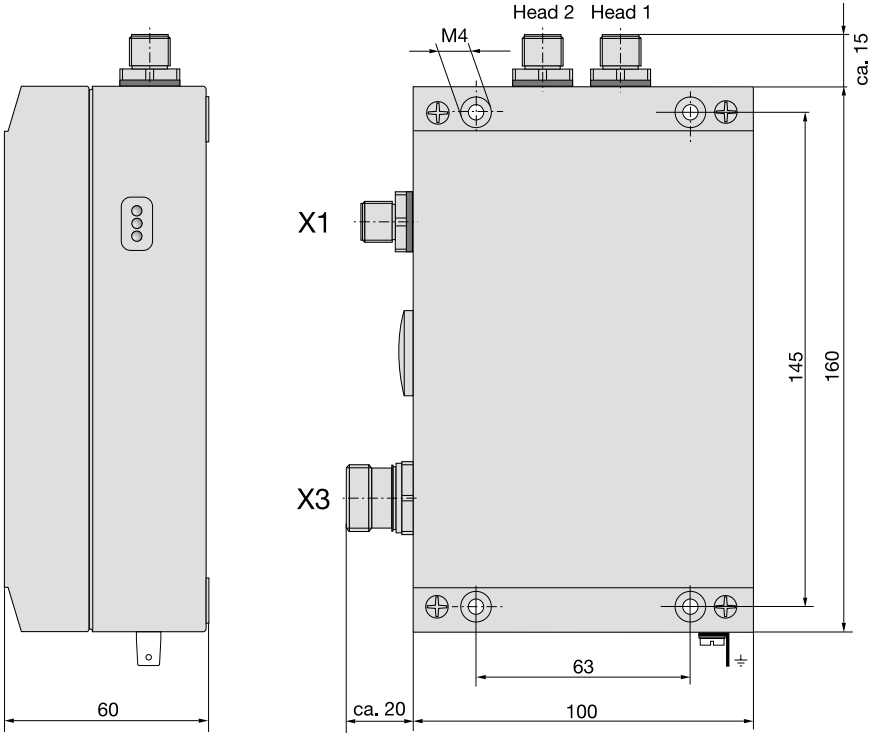
Read/Write Head
000 = no read/write head
651 = with read/write head BIS C-651 (circular antenna on top)
652 = with read/write head BIS C-652 (circular antenna on front)
653 = with read/write head BIS C-653 (bar antenna)
650 = adapter with two connections for external read/write heads BIS C-3_ _ (except BIS C-350 and -352)
670 = adapter with cable connection for an external read/write head BIS C-350 or BIS C-352

Interface
00 = RS 232 interface (V.24)
01 = 20-mA current loop interface (TTY)

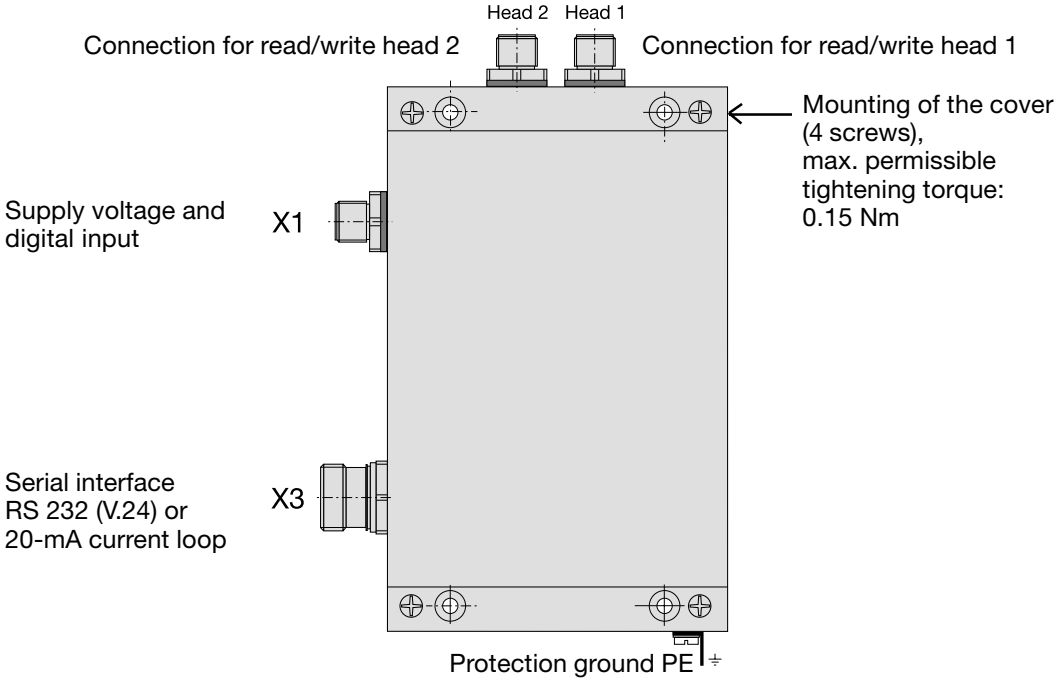
User Connections
KL1 = 3x PG 9 cord seal fittings

Installing the BIS C-620 processor

The processor is mounted using 4 size M4 screws.



Wiring for the BIS C-620 processor

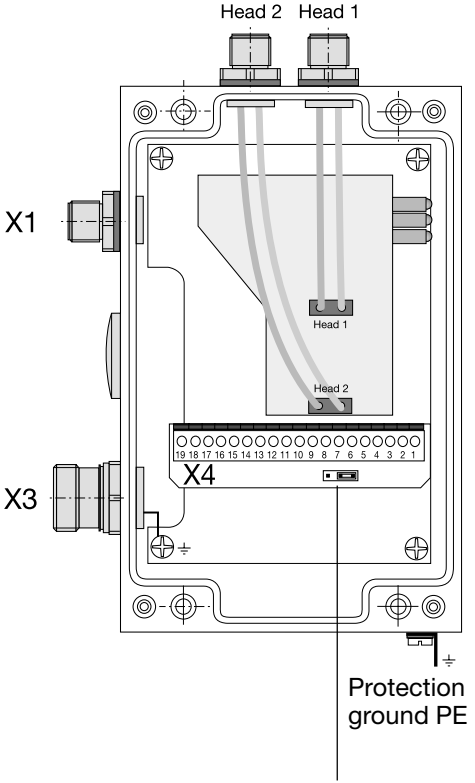


Connection locations and names

Opening the BIS C-620 processor

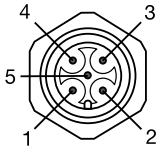
To insert the shunt connector for the handshake setting when using RS 232, the BIS C-620 processor must be opened.
Ensure that the device is turned off. Remove the 4 screws on the BIS C-620 and lift off the cover. See following pages for additional details.

Wiring for the BIS C-620 processor



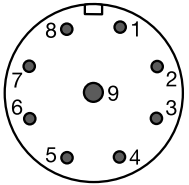
Shunt for handshake setting when using RS 232 (see Wiring diagrams on following pages)

X1, Supply voltage and digital input



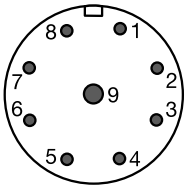
Pin	Function
1	+Vs
2	–IN
3	–Vs
4	+IN
5	n.c.

X3, RS 232
BIS C-620...00



Pin	Function
1	n.c.
2	RxD
3	TxD
4	n.c.
5	COM
6	n.c.
7	RTS
8	CTS
9	n.c.

X3, 20-mA current loop
BIS C-620...01



Pin	Function
1	TxD–
2	n.c.
3	n.c.
4	TxD+
5	n.c.
6	RxD–
7	n.c.
8	n.c.
9	RxD+

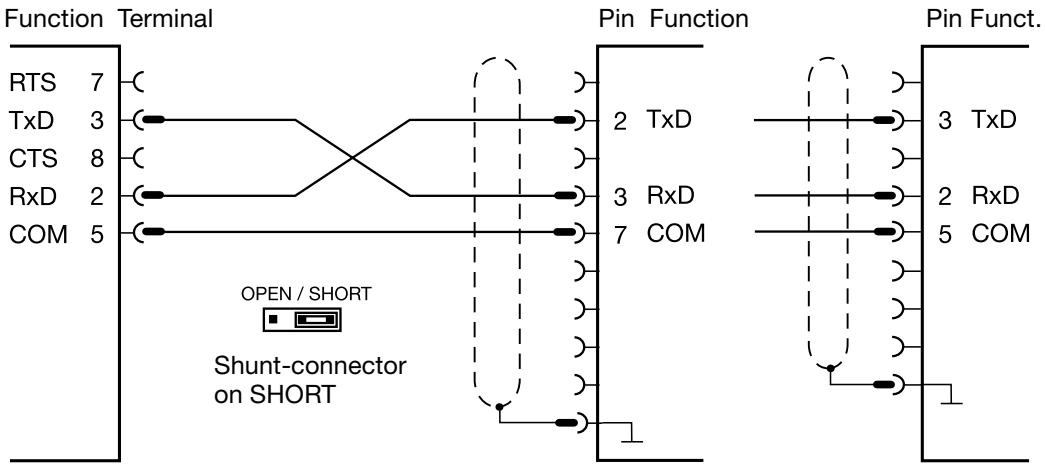
n.c. = leave empty

Function of the Interfaces

The BIS C-620 processor uses a serial interface for communicating with a host system (e.g. PC or PLC). Depending on version, the following configurations are available:

BIS C-620...00 the interface RS 232 (V.24) or
BIS C-620...01 the 20 mA current loop interface (TTY) .

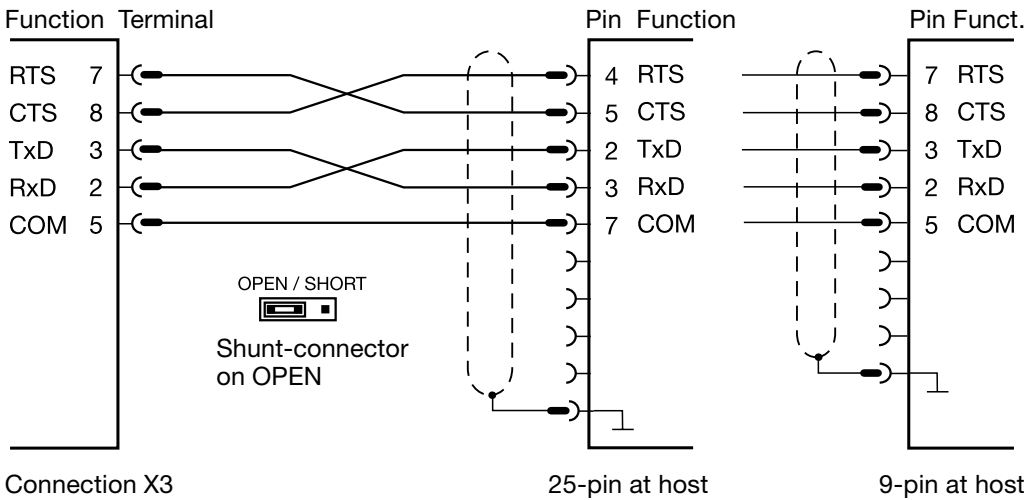
Interface RS 232 (V.24) without hardware handshake



OPEN / SHORT

CTS Factory setting = SHORT.
Since the CTS signal is not used, the shunt connector remains in the SHORT position.

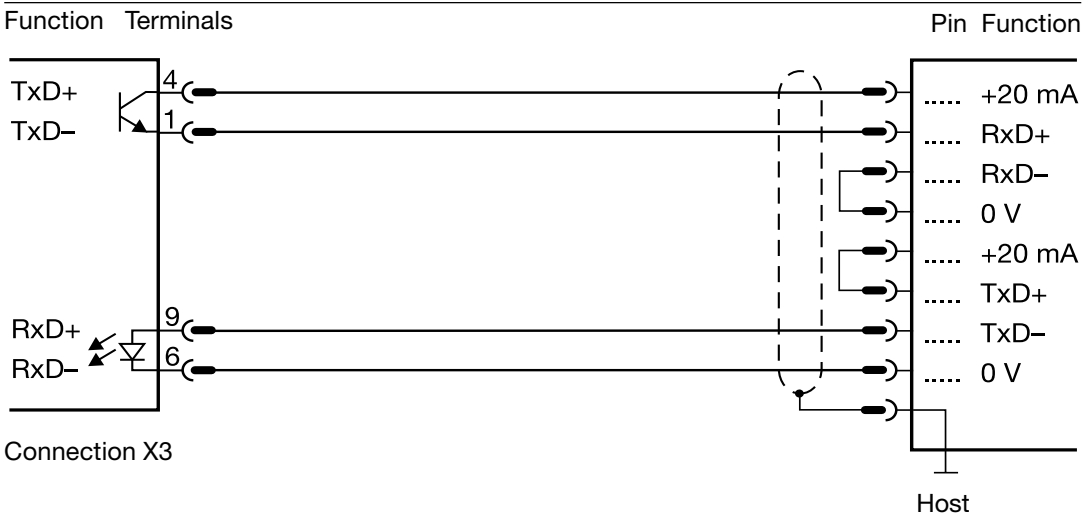
**RS 232 (V.24)
Interface with
hardware handshake**



OPEN / SHORT

CTS  Factory setting = SHORT.
Since the CTS signal is used, the shunt connector is set to the OPEN position.

20 mA Current Loop Interface (TTY), Processor passive



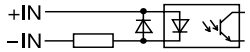
You may write in the pin numbers for wiring to your host in the spaces above.

BIS C-620
Technical Data

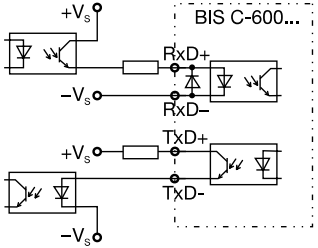
Dimensions, weight	**Housing** Dimensions Weight	Metal 175 x 120 x 60 mm 800 g
Temperature range	Ambient temperature	0 °C to +60 °C
Connections	Integrated connector X1 Einbaustecker Head 1, Head 2 Rundsteckverbinder X3	5-pole (male) 4-pole (male) 9-pole (male)
Protection	Protection	IP 65 (when connected)
Electrical connections	**Input X1, supply voltage V_S** Ripple Current draw	DC 24 V ± 20 % ≤ 10 % ≤ 400 mA
	Terminal X3, serial interface	RS 232 (V.24) or 20-mA current loop (TTY)
	Connections to read/write head Head 1, Head 2	2 x integral connectors 4-pole (male) for all BIS C-3_ _ read/write heads with 4-pole connector (female), **not** BIS C-350 and BIS C-352

Electrical Connections

Digital Input (+IN, –IN)	Optocoupler isolated
Control voltage active	4 V to 40 V
Control voltage inactive	1.5 V to –40 V
Input current at 24 V	11 mA
Typical delay time	5 ms



Serial Interface (TTY)	Optocoupler isolated
Receiver	Current loop 20 mA passive
Line current for base state	20 to 50 mA
Voltage drop at 20 mA	ca. 3 V
Sender	Current loop 20 mA passive
Idle voltage	max. 50 V
Voltage drop at 20 mA	ca. 3 V
Line current	max. 50 mA



Function Displays

System Ready	LED green
Codetag Present	LED yellow
Codetag Operating	LED yellow

With the CE Mark we affirm that our products are in accordance with the requirements of the EU (European Union) Guideline 89/336/EEC (EMC Guideline) and the EMC Law. It has been verified in our EMC Laboratory, which is accredited by the DATech for Testing of Electromagnetic Compatibility, that Balluff products meet the EMC requirements of the Harmonized Standard

EN 50081-2 (Emission), EN 50082-2 (Noise Immunity)

BIS C-620
Ordering Information

Ordering code

BIS C-620-_ _ _-050-_ _-ST2

Balluff Identification System

Series C

Hardware Type
620 = Metal housing

Software Type
022 = Server software
007 = Balluff protocol

Read/write head
050 = with two connectors for external read/write heads
BIS C-3_ _ (except BIS C-350 and -352)

Interface
00 = RS 232 (V.24)
01 = 20-mA current loop (TTY)

User Connections
ST2 = Connector variant
1 round connector for supply voltage,
1 round connector for RS 232 (V.24) / 20-mA current loop (TTY)

Accessories
(optional, not included)

Article		Ordering code
Mating connector	for X1	BKS-S 79-00
	for X3	BKS-S 84-00
Protection cap	for Head 1, Head 2	BES 12-SM-2

Appendix, ASCII Table

Decimal	Hex	Control Code	ASCII
0	00	Ctrl @	NUL
1	01	Ctrl A	SOH
2	02	Ctrl B	STX
3	03	Ctrl C	ETX
4	04	Ctrl D	EOT
5	05	Ctrl E	ENQ
6	06	Ctrl F	ACK
7	07	Ctrl G	BEL
8	08	Ctrl H	BS
9	09	Ctrl I	HT
10	0A	Ctrl J	LF
11	0B	Ctrl K	VT
12	0C	Ctrl L	FF
13	0D	Ctrl M	CR
14	0E	Ctrl N	SO
15	0F	Ctrl O	SI
16	10	Ctrl P	DLE
17	11	Ctrl Q	DC1
18	12	Ctrl R	DC2
19	13	Ctrl S	DC3
20	14	Ctrl T	DC4
21	15	Ctrl U	NAK

Decimal	Hex	Control Code	ASCII
22	16	Ctrl V	SYN
23	17	Ctrl W	ETB
24	18	Ctrl X	CAN
25	19	Ctrl Y	EM
26	1A	Ctrl Z	SUB
27	1B	Ctrl [	ESC
28	1C	Ctrl \	FS
29	1D	Ctrl]	GS
30	1E	Ctrl ^	RS
31	1F	Ctrl _	US
32	20		SP
33	21		!
34	22		"
35	23		#
36	24		$
37	25		%
38	26		&
39	27		'
40	28		(
41	29		)
42	2A		*
43	2B		+

Decimal	Hex	ASCII
44	2C	,
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?
64	40	@

Decimal	Hex	ASCII
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U

Decimal	Hex	ASCII
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j

Decimal	Hex	ASCII
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	DEL