

MscDII Referenzhandbuch

© 2015 - 2017 Messtechnik Sachs GmbH

Diese Betriebsanleitung wurde für die Darstellung in einem Webbrowser im HTML-Format optimiert. Verwenden Sie die PDF-Version nur, wenn kein Zugriff auf die Online-Hilfe möglich ist.

1.	Einleitung	7
1.1	Impressum	8
1.2	Revisions-Historie	8
1.3	Rechtliche Hinweise	9
1.3.1	Nutzungsbedingungen für Software / elektronische Dokumentation	9
1.3.2	Qualifiziertes Personal	13
1.3.3	Haftungsausschluss	13
1.4	Vorwort	13
1.4.1	Zweck	13
1.4.2	Gültigkeitsbereich dieser Bedienungsanleitung	13
1.4.3	Bestimmungsgemäßer Gebrauch	14
1.4.4	Erforderliche Grundkenntnisse	14
1.4.5	Weitere Dokumentation	14
1.4.6	Versionsstand	14
2.	MscDll oder HighLevel-DLL?	15
3.	Die MscDll	19
3.1	Msc.cfg	20
4.	Programmierschnittstelle	25
4.1	Rückgabewerte (MSC_STATUS)	26
4.2	Allgemein	29
4.2.1	MSC_GetVersion	29
4.2.2	MSC_WriteCommand	30
4.3	Verbindung	32
4.3.1	MSC_EnumerateDevices	32
4.3.2	MSC_GetDeviceInfo	33
4.3.3	MSC_OpenDevice	34
4.3.4	MSC_InitDevice	35
4.3.5	MSC_CloseDevice	36
4.3.6	MSC_Start	37
4.3.7	MSC_Stop	39
4.3.8	MSC_GetDeviceState	40
4.4	Benachrichtigungen (Notifications)	42
4.4.1	MSC_SetNotificationMessage	43
4.4.2	MSC_SetNotificationEvent	45
4.4.3	MSC_SetNotificationCallback	47
4.5	Statische Übertragungskanäle	49
4.5.1	MSC_SetupStaticChannel	49
4.5.2	MSC_ReadStatic	51
4.5.3	MSC_RefreshChannel	53

4.6	Übertragungs-Kanal für dynamische Messwerte	54
4.6.1	MSC_SetupExtendedDynamicChannel	54
4.6.2	MSC_AttachSubChannelBuffer	56
4.6.3	MSC_DetachSubChannelBuffers	57
4.6.4	MSC_GetPosition	59
5.	Opcodes und Parameter	61
5.1	Bedeutung von Opcodes	62
5.2	Opcode-Übersicht	62
5.3	Parameter-Typ "String"	66
5.4	Opcodes: Initialisierung	67
5.4.1	opcRIV: Lese Box-Übersicht	67
5.4.2	opcRMI: Lese Box-Informationen (digitales Typenschild)	68
5.4.3	opcRSS: Lese System-String	74
5.4.4	opcWCC: Schreibe Kanal-Eigenschaften	76
5.4.5	opcRCA: Lese Kanal-Zuordnung	79
5.4.6	opcWCA: Schreibe Kanal-Zuordnung	82
5.5	Opcodes: Konfiguration & Verschiedene	85
5.5.1	opcWCL: Schreibe Kanal-Liste	85
5.5.2	opcRCL: Lese Kanal-Liste	87
5.5.3	opcACL: Aktiviere Kanal-Liste für die statische Messung	89
5.5.4	opcDT: Definiere Trigger für die dynamische Messung	91
5.5.5	opcAT: Aktiviere Trigger für die dynamische Messung	96
5.5.6	opcIT: Deaktiviere Trigger der dynamischen Messung	97
5.5.7	opcSP: Setze (Kanal-)Parameter	99
5.5.8	opcRHS: Lese Hardware-Status der Messkanäle	102
5.5.9	opcREv: Lese aktuelles Ereignis der jeweiligen Irinos-Boxen	106
5.5.10	opcSAbsT: Setze Absolut-Zeit (für den Diagnose-Speicher)	107
5.5.11	opcWEvCfg: Schreibe Ereignis-Konfiguration	110
5.5.12	opcClrEv: Lösche Ereignis	112
5.6	Opcodes: Messung	114
5.6.1	opcRS: Lese statische Messwerte	114
5.6.2	opcBIO: Lese digitale Eingänge / schreibe digitale Ausgänge	115
5.6.3	opcBIORO: Status der digitalen Ein-/Ausgänge abfragen	118
5.6.4	opcRSW: Lese Statuswort für dynamische Messungen	120
5.6.5	opcRDC: Lese Anzahl der dynamischen Messwerte	123
5.6.6	opcDDMx: Definiere dynamische Messung	125
5.6.7	opcRDMx: Lese dynamische Messwerte	127
5.7	Opcodes: Service	129
5.7.1	opcRST: System Reset	129
6.	HighLevel-Funktionen (Optional)	131
6.1	IRHL_GetVersionHL	134

6.2	IRHL_Connect	136
6.3	IRHL_Disconnect	137
6.4	IRHL_GetCommStatus	138
6.5	IRHL_WriteCommand	139
6.6	IRHL_GetCurrentHandle	140
6.7	IRHL_StaticMeasurement_Start	141
6.8	IRHL_StaticMeasurement_ReadArray	142
6.9	IRHL_StaticHardwareStatus_Start	143
6.10	IRHL_StaticHardwareStatus_ReadArray	144
6.11	IRHL_StaticBIO_Start	145
6.12	IRHL_StaticBIO_ReadInputs	146
6.13	IRHL_StaticBIO_WriteOutputs	147
6.14	IRHL_DynReset	148
6.15	IRHL_DynAddMeasChannel	149
6.16	IRHL_DynTimeStart	150
6.17	IRHL_DynPosStart	152
6.18	IRHL_DynStop	154
6.19	IRHL_DynGetValuesAvailable	155
6.20	IRHL_DynReadValueArray	156
Index		159

Einleitung

1 Einleitung

1.1 Impressum

Titel	Irinos MscDII Referenzhandbuch
Hersteller	Messtechnik Sachs GmbH Siechenfeldstraße 30/1 D-73614 Schorndorf Tel. 07181 / 99960-0 post@messtechnik-sachs.de
Gültig für	Messmodule Irinos IR
Copyright-Hinweis	© 2015 - 2017 Messtechnik Sachs GmbH
Hinweis auf Markenzeichen und Warenzeichen	Alle in diesem Handbuch genannten Bezeichnungen von Erzeugnissen sind Warenzeichen der jeweiligen Firmen.
Material-Nr.	785-1002
Änderungshinweis	Technische Änderungen vorbehalten.
Stand der Drucklegung	04.10.2017

1.2 Revisions-Historie

Version	Datum	Änderungen
A	2016-02-17	Erste Version
B	2016-	Korrektur der Beschreibung des Datenformats für den Opcode opcREv ¹⁰⁶ .

	11-02	
C	20-16-11-17	Neuer Opcode opcBIORO ^{□118} (verfügbar ab Firmware-Version 1.4.x.x).
D	20-17-05-31	Dokumentation für HighLevel-Funktionen via HighLevel-DLL ^{□116} hinzugefügt.
E	20-17-07-17	<ul style="list-style-type: none"> • Dokumentation des Hardware-Status-Bytes^{□102} für Temperatur-Eingänge. • opcSP^{□99}: Neue Möglichkeit zum Zurücksetzen des Referenzmarken-Flags bei Inkrementalgebern dokumentiert (verfügbar ab IR-INC Firmware-Version 1.5.0.24). • opcWCC^{□76}: Unterstützung der 4x-Auswertung für TTL/RS422-Inkrementalgeber (verfügbar ab IR-INC Firmware-Version 1.5.0.24).
F	20-17-10-04	Neuer Opcode opcRDC ^{□123} (verfügbar ab Firmware-Version 1.8.x.x).

1.3 Rechtliche Hinweise

1.3.1 Nutzungsbedingungen für Software / elektronische Dokumentation

Schutzrechte und Nutzungsumfang

Messtechnik Sachs stellt entweder auf portablen Datenträgern (z. B. Disketten, CD-ROMs, DVDs, ...), in schriftlicher (drucktechnischer) oder elektronischer Form Bedienungsanleitungen, Handbücher, Dokumentationen, sowie Softwareprogramme, alles und insgesamt im Folgenden als "LIZENZGEGENSTAND" bezeichnet, entgeltlich und/oder unentgeltlich zur Verfügung. Der LIZENZGEGENSTAND unterliegt u.a. urheberrechtlichen Schutzbestimmungen. Messtechnik Sachs oder Dritte haben Schutzrechte an diesem LIZENZGEGENSTAND. Soweit Dritten ganz oder teilweise Rechte an diesem LIZENZGEGENSTAND zustehen, hat Messtechnik Sachs entsprechende Nutzungsrechte. Messtechnik Sachs gestattet dem Verwender die

Nutzung des LIZENZGEGENSTANDES unter den folgenden Voraussetzungen:

1.1) Nutzungsumfang elektronische Dokumentation

- a) Mit dem Erhalt/Erwerb oder der Überlassung eines LIZENZGEGENSTANDES erhalten Sie als Verwender in Bezug auf den jeweiligen LIZENZGEGENSTAND ein einfaches, nicht übertragbares Nutzungsrecht, das den Verwender berechtigt, diesen für eigene, ausschließlich betriebsinterne Zwecke, auf beliebig vielen Maschinen innerhalb seines Betriebsgeländes (Einsatzort), zu nutzen. Dieses Nutzungsrecht umfasst ausschließlich das Recht, den LIZENZGEGENSTAND auf den am Einsatzort eingesetzten Zentraleinheiten (Maschinen) zu speichern.
- b) Bedienungsanleitungen und/oder Dokumentationen, ungeachtet in welcher Form zur Verfügung gestellt, darf der Verwender an dessen Einsatzort außerdem in beliebiger Zahl über einen Drucker ausdrucken, sofern dieser Ausdruck vollständig mit diesen Nutzungsbedingungen und sonstigen Benutzerhinweisen ausgedruckt bzw. verwahrt wird.
- c) Mit Ausnahme des Messtechnik Sachs Logos ist der Verwender berechtigt, Bilder und Texte der Bedienungsanleitungen/Dokumentationen zur Erstellung eigener Maschinen- und Anlagendokumentation zu verwenden. Die Verwendung des Messtechnik Sachs Logos bedarf der schriftlichen Genehmigung von Messtechnik Sachs. Für die Übereinstimmung genutzter Bilder und Texte mit der Maschine/Anlage bzw. dem Produkt ist der Verwender selbst verantwortlich.
- d) Weitergehende Nutzungen sind in folgendem Rahmen zulässig:

Das Vervielfältigen ausschließlich zur Verwendung im Rahmen einer Maschinen- und Anlagendokumentation aus elektronischen Dokumenten sämtlicher dokumentierter Zulieferbestandteile. Die Demonstration gegenüber Dritten ausschließlich unter Sicherstellung, dass kein Datenmaterial ganz oder teilweise in anderen Netzwerken oder anderen Datenträgern verbleibt oder dort reproduziert werden kann.

Die Weitergabe von Ausdrucken an Dritte außerhalb der Regelung in Ziffer 3 sowie jede Bearbeitung oder andersartige Verwendung sind nicht zulässig.

1.2) Nutzungsumfang Softwareprodukte

An Software von Messtechnik Sachs jeglicher Art und der dazugehörigen Dokumentation erhält der Kunde ein nicht ausschließliches, nicht übertragbares und zeitlich nicht begrenztes Nutzungsrecht auf einem bestimmten bzw. im Einzelfall festzulegenden Hardware-Produkt. Messtechnik Sachs bleibt Inhaberin des Urheberrechts sowie aller anderen

gewerblichen Schutzrechte. Das Recht Vervielfältigungen anzufertigen, ist nur zum Zwecke der Datensicherung gegeben. Copyright-Vermerke dürfen nicht entfernt werden.

2. Copyright Vermerk

Jeder LIZENZGEGENSTAND enthält einen Copyright Vermerk. Bei jeglicher Vervielfältigung die nach diesen Bestimmungen erlaubt ist, muss der entsprechende Copyright Vermerk des betreffenden Originals übernommen werden:

Bsp.: © 2015-2016, Messtechnik Sachs GmbH,

D-73614 Schorndorf

3. Übertragung der Nutzungsbefugnis

Der Verwender kann seine Nutzungsbefugnis nach diesen Bestimmungen bzgl. des jeweiligen LIZENZGEGENSTANDES in dem Umfang und mit den Beschränkungen der Bedingungen gemäß Ziffer 1 und 2 insgesamt auf einen Dritten übertragen. Auf diese Nutzungsbedingungen ist der Dritte ausdrücklich hinzuweisen.

II. Export LIZENZGEGENSTAND

Der Verwender muss beim Export des LIZENZGEGENSTANDES oder Teilen davon die Ausfuhrbestimmungen des ausführenden Landes und des Landes des Erwerbs beachten.

III. Gewährleistung

1. Produkte von Messtechnik Sachs werden hard- und softwaretechnisch weiterentwickelt. Liegt der LIZENZGEGENSTAND, gleich in welcher Form, einem Produkt nicht unmittelbar bei, d.h. wird nicht auf einem, dem Produkt beiliegenden portablen Datenträger mit dem betreffenden Produkt als Liefereinheit ausgeliefert, gewährleistet Messtechnik Sachs nicht, dass eine elektronische Dokumentation mit jedem Hard- und Software-Stand des Produkts übereinstimmt.

2. Die in einer elektronischen Dokumentation enthaltenen Informationen können von Messtechnik Sachs ohne Vorankündigungen geändert werden und stellen keine Verpflichtung seitens Messtechnik Sachs dar.

3. Messtechnik Sachs gewährleistet, dass das von ihr erstellte Softwareprogramm mit der Anwendungsbeschreibung und Programmspezifikation übereinstimmt, jedoch nicht, dass die in

der Software enthaltenen Funktionen vollständig unterbrechungs- u. fehlerfrei laufen oder dass die in der Software enthaltenen Funktionen in allen vom Erwerber gewählten Kombinationen und vorgesehenen Einsatzbedingungen ausführbar sind, bzw. den Erfordernissen entsprechen.

IV. Haftung/Haftungsbeschränkungen

1. Messtechnik Sachs stellt LIZENZGEGENSTÄNDE zur Verfügung, um den Verwender einerseits in die Lage zu versetzen Messtechnik Sachs Produkte die zum ordnungsgemäßen Betrieb einer Software bedürfen, diese vertragsgemäß einzusetzen, oder ihn bei der Erstellung seiner Maschinen- und Anlagendokumentation zu unterstützen. Für die elektronische Dokumentation, die in Form von portablen Datenträgern nicht unmittelbar einem Produkt beiliegt, d.h. nicht mit einem Produkt als Liefereinheit ausgeliefert wurde, gewährleistet

Messtechnik Sachs garantiert jedoch nicht, dass die separat vorgehaltene/gelieferte elektronische Dokumentation mit dem vom Verwender tatsächlich genutzten Produkt übereinstimmt.

Letzteres gilt insbesondere bei auszugsweisem Gebrauch für eigene Dokumentationen des Verwenders. Die Gewährleistung und Haftung für separat vorgehaltene / gelieferte portable Datenträger, d.h. mit Ausnahme der im Internet/Intranet vorgehaltenen elektronischen Dokumentation, beschränkt sich ausschließlich auf eine ordnungsgemäße Duplikation der Software, wobei Messtechnik Sachs gewährleistet, dass jeweils der neueste Stand der Dokumentation Inhalt des betreffenden, portablen Datenträgers ist. In Bezug auf die im Internet/Intranet vorgehaltene elektronische Dokumentation wird nicht gewährleistet, dass diese denselben Versionsstand aufweist wie die zuletzt drucktechnisch veröffentlichte Ausgabe.

2. Messtechnik Sachs haftet ferner nicht für mangelnden wirtschaftlichen Erfolg oder für Schäden oder Ansprüche Dritter wegen der Nutzung/Verwendung der vom Verwender eingesetzten LIZENZGEGENSTÄNDE, mit Ausnahme von Ansprüchen aus der Verletzung von Schutzrechten Dritter, welche die Nutzung der LIZENZGEGENSTÄNDE betreffen.

3. Die Haftungsbeschränkungen nach Absatz 1. und 2. gelten nicht, soweit in Fällen von Vorsatz oder grober Fahrlässigkeit oder Fehlen zugesicherter Eigenschaften eine zwingende Haftung besteht. In einem solchen Fall ist die Haftung von Messtechnik Sachs auf den Schaden begrenzt, der für Messtechnik Sachs nach der Kenntnis der konkreten Umstände erkennbar war.

V. Sicherheitsrichtlinien/Dokumentation

Gewährleistungs- und Haftungsanspruch nach Maßgabe der vorstehenden Regelungen (Ziff. III. u. IV) sind nur gegeben, wenn der Anwender die Sicherheitsrichtlinien einer Dokumentation im Zusammenhang mit der Nutzung der Maschine und deren Sicherheitsrichtlinien oder die Nutzungsbedingungen von Software beachtet hat. Für die Kompatibilität nicht mit einem Produkt als Liefereinheit ausgelieferter elektronischer Dokumentation mit dem vom Anwender tatsächlich genutzten Produkt ist der Anwender selbst verantwortlich.

1.3.2 Qualifiziertes Personal

Das zu dieser Dokumentation zugehörige Produkt/System darf nur von für die jeweilige Aufgabenstellung qualifiziertem Personal gehandhabt werden unter Beachtung der für die jeweilige Aufgabenstellung zugehörigen Dokumentation, insbesondere der darin enthaltenen Sicherheits- und Warnhinweise. Qualifiziertes Personal ist auf Grund seiner Ausbildung und Erfahrung befähigt, im Umgang mit diesen Produkten/Systemen Risiken zu erkennen und mögliche Gefährdungen zu vermeiden.

1.3.3 Haftungsausschluss

Der Inhalt dieser Dokumentation wurde von uns sorgfältig auf Übereinstimmung mit der beschriebenen Hard- und Software geprüft. Abweichungen können dennoch nicht ausgeschlossen werden, so dass wir für die vollständige Übereinstimmung keine Gewähr übernehmen. Die Angaben in dieser Druckschrift werden regelmäßig geprüft. Etwaige Korrekturen sind in den nachfolgenden Auflagen enthalten.

1.4 Vorwort

1.4.1 Zweck

Dieses Referenz-Handbuch dient als Nachschlagewerk für die Funktionen der MscDII zur Verwendung mit dem Irinos-System. Zur Zielgruppe gehören Software-Entwickler, die die MscDII in eine PC-Software (z.B. Messrechner-Software) einbinden und/oder Inbetriebnahme-Personal mit Software-Fachkenntnis, die das Irinos-System parametrieren.

1.4.2 Gültigkeitsbereich dieser Bedienungsanleitung

Diese Bedienungsanleitung gilt für das industrielle Mess- und Steuerungssystem Irinos (IR-xxx) in Verbindung mit der MscDII.

1.4.3 Bestimmungsgemäßer Gebrauch

Irinos ist ein flexibles High-Speed - Messsystem für die industrielle Fertigungsmesstechnik. Es ist für den Dauerbetrieb (24/7) geeignet.

Das Messsystem ist ausdrücklich nicht geeignet für den Einsatz in medizinischen oder explosionsgefährdeten Bereichen, Flugzeugen, für die Raumfahrt sowie für den Heim- und Bürobereich. Nicht aufgeführte Bereiche, die diesen vom Sinn her ähnlich sind, gehören ebenfalls dazu.

In sicherheitskritischen Bereichen ist die Betriebssicherheit durch externe Vorrichtungen zu gewährleisten (z.B. externer Not-Aus-Kreis).

1.4.4 Erforderliche Grundkenntnisse

Für die Einbindung und Anwendung der MscDll sind solide Kenntnisse in der PC-basierten Software-Entwicklung unter Windows erforderlich.

1.4.5 Weitere Dokumentation

Beachten Sie das Begleitblatt, das mit jeder Irinos-Box mitgeliefert wird. Dies gilt insbesondere für die darin enthaltenen Sicherheitshinweise. Die technischen Daten sind dem jeweils zugehörigen Datenblatt zu entnehmen.

Beachten Sie zudem das Irinos Benutzerhandbuch.

1.4.6 Versionsstand

Diese Bedienungsanleitung bezieht sich auf den Firmware Versionsstand V1.0.

MscDll oder HighLevel-DLL?

2 MscDll oder HighLevel-DLL?

Seit Mai 2017 ist als Ergänzung zur [MscDLL](#)¹²⁰ eine sogenannte [HighLevel-DLL](#)¹³² verfügbar.

Wozu die HighLevel-DLL / HighLevel-Funktionen?

In bestimmte Software-Pakete kann die MscDLL nicht reibungslos integriert werden, so zum Beispiel in LabView oder Excel. Die Programmierschnittstelle der HighLevel-DLL ist hierfür optimiert.

Kann / muss ich generell die HighLevel-DLL verwenden?

Können ja, müssen nein.

Die HighLevel-DLL bietet ein im Vergleich zur MscDLL vereinfachtes Programmier-Interface. Wenn Sie wollen, dann können Sie dieses nutzen. Zusätzlich sind über die HighLevel-DLL auch alle Funktionen der MscDll zugänglich.

Die meisten gängigen Entwicklungsumgebungen erlauben eine direkte Verwendung der MscDll (siehe hierzu auch die verfügbaren Beispiel-Applikationen) mit vollem Funktions-Umfang. Wenn die MscDll bereits eingesetzt wird, so ist eine Umstellung auf die HighLevel-DLL nicht notwendig.

Hat sich durch die Einführung der HighLevel-DLL etwas an der MscDll geändert?

Nein. Die MscDll bleibt unverändert. Für bestehende Implementierungen ändert sich daher nichts.

Was sind die Gemeinsamkeiten bzw. wesentlichen Unterschiede?

	MscDLL	HighLevel-DLL
Statische Messung	Ja	Ja
Bit I/O	Ja	Ja

	MscDLL	HighLevel-DLL
Hardware-Status auslesen	Ja	Ja
Dynamische Messung	Ja Voller Funktionsumfang. Parametrierung und Durchführung über mehrere Befehle. Unterscheidung zwischen "Trigger" und "dynamischer Messung".	Ja Funktionsumfang ausreichend für alle häufig verwendeten Applikationen. Parametrierung und Durchführung über wenige einfache Befehle. Keine Unterscheidung zwischen "Trigger" und "dynamischer Messung".
Speicherverwaltung	Speicher muss von der Applikation bereitgestellt werden.	Speicher wird innerhalb der DLL verwaltet.
Mehrgeräte-Fähigkeit	Mehrere Geräte nutzbar über "Device-Handles".	Ein Gerät nutzbar (z.B. ein Irinos-System).
Unterstützung für LabView / Excel	Nein	Ja

Tipp

Wir beraten Sie gerne bei der Auswahl der richtigen DLL / Programmierschnittstelle.

Rechtliche Hinweise / Legal Notes:

LabView is a trademark of National Instruments. Neither Messtechnik Sachs GmbH, nor any software programs or other

goods or services offered by Messtechnik Sachs GmbH, are affiliated with, endorsed by, or sponsored by National Instruments.

Excel is a trademark of Microsoft Corporation. Neither Messtechnik Sachs GmbH, nor any software programs or other goods or services offered by Messtechnik Sachs GmbH, are affiliated with, endorsed by, or sponsored by Microsoft Corporation.

Die MscDII

3 Die MscDll

Die MscDll ist das Bindeglied zwischen der Applikationssoftware und dem Irinos-System. Über sie erfolgt sowohl das Auslesen von Messwerten und Statusinformationen, als auch die System-Parametrierung.

Die MscDll setzt direkt auf den Windows API-Funktionen für die IP-basierte Kommunikation sowie für das Thread-Management und Timing auf.

Innerhalb der DLL läuft ein eigener Thread, der die Kommunikation steuert. Die DLL-Funktionen übergeben diesem Thread Daten bzw. umgekehrt.

Die Kommunikation zum Irinos-System erfolgt über UDP/IP. Die DLL wiederholt ein Datenpaket bei einem Verlust automatisch. Für die Nutzung des Irinos-Systems wird eine Ethernet-Direktverbindung zum Messrechner empfohlen. Komplexe Netzwerkstrukturen, wie beispielsweise umfangreiches Routing, Tunneling, VPN, etc. werden aus Timing-Gründen nicht unterstützt.

Dieses Referenz-Handbuch dient als Nachschlagewerk für die Verwendung des Irinos-Systems in Verbindung mit der MscDll. Für einige Programmiersprachen stehen Anwendungsbeispiele zur Verfügung. Verwenden Sie diese als Vorlage für die praktische Nutzung.

Alle Informationen zur Nutzung des Irinos-Systems finden Sie im Irinos-Benutzerhandbuch.

3.1 Msc.cfg

Die Konfiguration der MscDll erfolgt über die Konfigurations-Datei Msc.cfg. Der Inhalt der Datei Msc.cfg sieht üblicherweise folgendermaßen aus:

```
[System]
FTDI=OFF
XPort=ON

[XPort]
Address1=192.168.3.99:10002
EnumRetry=2
EnumTimeout=400
SendBufSize=1500
RcvBufSize=65536
```

Eine Beschreibung der Parameter zeigt folgende Tabelle:

Parameter	Beschreibung
FTDI	Nur zur Kompatibilität mit älteren Systemen. Dieser Parameter muss immer den Wert OFF haben.
XPORT	Muss immer den Wert ON haben, damit die IP-basierte Kommunikation funktioniert.
Adress1	<p>Muss die IP-Adresse des Irinos-Systems gefolgt von der Port-Nummer haben. Die Port-Nummer ist immer 10002.</p> <p>(Zur Kompatibilität mit früheren Systemen wird auch die Port-Nummer 10001 unterstützt. Dann ist die UDP-Paketgröße jedoch auf 800 Bytes begrenzt.</p> <p>Die Verwendung des Ports 10002 setzt eine UDP-Puffergröße von 1500 Bytes bzw. 65536 Bytes voraus. Siehe dazu die Parameter SendBufSize und RcvBufSize.)</p>
EnumRetry	Anzahl an Wiederholversuchen bei der Suche nach Irinos-Systemen beim Verbindungsaufbau.
EnumTimeout	Timeout der Suche nach Irinos-Systemen beim Verbindungsaufbau.
SendBufSize	Größe des UDP-Sendepuffers. Dieser sollte immer 1500 Bytes groß sein.
RcvBufSize	Größe des UDP-Empfangspuffers. Dieser sollte immer mindestens 65536 Bytes groß sein.

Wenn eine Irinos-Box mit aktiviertem DHCP-Server an einen PC angeschlossen wird, dann kann diese Datei unverändert übernommen werden. Ansonsten muss durch den Anwender lediglich die IP-Adresse über den Parameter „Adress1“ an die tatsächliche IP-Adresse des Irinos-Systems angepasst werden.

Die Konfigurations-Datei Msc.cfg kann auch über das Irinos-Tool automatisch erzeugt werden. Näheres dazu finden Sie in der Dokumentation des Irinos-Tools.

Programmierschnittstelle

4 Programmierschnittstelle

Dieser Abschnitt beschreibt die Funktionsaufrufe der MscDll.

4.1 Rückgabewerte (MSC_STATUS)

Die Funktionsaufrufe der MscDll haben stets den Rückgabewert MSC_STATUS. Dieser gibt Aufschluss darüber, ob ein Funktionsaufruf erfolgreich ausgeführt wurde, oder was gegebenenfalls die Ursache für die Nicht-Ausführung war. Der Rückgabewert ist wie folgt definiert:

Rückgabewert	Hex-Wert	Beschreibung
MSC_STATUS_SUCCESS	0x00000000	no error detected
MSC_STATUS_FAILED	0xF0000001	general error
MSC_STATUS_INVALID_HANDLE	0xF0000002	call an invalid handle
MSC_STATUS_INVALID_PARAMS	0xF0000003	invalid parameter
MSC_STATUS_NO_RESOURCES	0xF0000004	no resources for creation
MSC_STATUS_NO_DEVICES	0xF0000005	no device found
MSC_STATUS_NOT_INITIALIZED	0xF0000006	member not initialized
MSC_STATUS_ALREADY_INITIALIZED	0xF0000007	interface is already initialized, repeated call
MSC_STATUS_INVALID_OBJECT_TYPE	0xF0000008	handle identifies an invalid object type,

Rückgabewert	Hex-Wert	Beschreibung
		handle is invalid for this function call
MSC_STATUS_INVALID_CHANNEL_TYPE	0xF0000009	channel does not have the correct type for this function
MSC_STATUS_FUNCTION_NOT_ALLOWED	0xF0000100	function not allowed at this time
MSC_STATUS_NO_DATA_AVAILABLE	0xF0000200	no data available
MSC_STATUS_NO_MORE_DATA	0xF0000400	no more data
MSC_STATUS_BUFFER_TOO_SHORT	0xF0000401	buffer too short

Für die [HighLevel-Funktionen](#)¹³² sind folgende zusätzlichen Fehlercodes definiert:

Rückgabewert	Hex-Wert	Beschreibung
MSC_STATUS_INVALID_RESPONSE	0xF8000000	Received an invalid / unexpected response from the Irinos system. Get in touch with support.
MSC_STATUS_STRINGGEN_FAILED	0xF8000001	Internal failure generating a string. Get in touch with support.
MSC_STATUS_INVALID_CHANNELNO	0xF8000100	Invalid measurement channel number. Make sure that physical measurement channel exists.
MSC_STATUS_INVALID_CHANNELLIST	0xF8000101	Channel list for dynamic measurement is not allowed. Can only occur, if Irinos system setup has been changed without restart of the measurement software.
MSC_STATUS_INVALID_NIOBYTES	0xF8000200	Invalid number of bytes for Bit I/O.
MSC_STATUS_DYNAMIC_NO_INVALID	0xF8000500	Invalid dynamic measurement number.
MSC_STATUS_DYNAMIC_NO_CHANNELS	0xF8000501	No measurement channels have been added to dynamic measurement. Use IRHL_DynReset ^[148] and IRHL_DynAddMeasurementChannel ^[149] before starting a new dynamic measurement.

4.2 Allgemein

4.2.1 MSC_GetVersion

Diese Funktion gibt die aktuelle Version der DLL zurück.

Definition

```
void
MSC_GetVersion(unsigned long* ApiVersion, unsigned long*
DllVersion);
```

Parameter

ApiVersion

Gibt die API Version der DLL zurück. Die oberen 16 Bit enthalten die Hauptversion und die unteren 16 Bit die Nebenversion.

DllVersion

Gibt die Version der DLL zurück. Die oberen 16 Bit enthalten die Hauptversion und die unteren 16 Bit die Nebenversion.

Kommentare

Wenn Veränderungen an der Programmierschnittstelle vorgenommen werden, die kompatibel mit älteren Versionen sind, wird die niedrigere Versionsnummer erhöht.

Bei Änderungen, die eine Inkompatibilität mit älteren Versionen verursachen, wird die höhere Versionsnummer erhöht.

Anwendungen sollten den Rückgabewert dieser Funktion in Abhängigkeit einer in der Applikation hinterlegten Konstante überprüfen, um sicherzustellen, dass die DLL die erwartete API Version unterstützt. Die Nummer der Hauptversion muss der erwarteten Nummer exakt entsprechen. Die Nummer der Nebenversion muss größer oder gleich groß sein wie die erwartete Nummer.

Der Aufruf dieser Funktion ist auch bei inaktiver Kommunikation möglich.

4.2.2 MSC_WriteCommand

Diese Funktion schreibt Daten zum Irinos-System und wartet auf die Antwort.

Definition

```
MSC_STATUS
MSC_WriteCommand(
    MSC_HANDLE Handle,
    unsigned char OpCode,
    unsigned long SndBufferSize,
    void* SndBuffer,
    unsigned long RcvBufferSize,
    void* RcvBuffer,
    unsigned long* BytesReceived,
    unsigned long Timeout
);
```

Parameter

Handle

Handle zum Gerät, das bei einem früheren Aufruf der Funktion [MSC_OpenDevice](#)³⁴ zurückgegeben wurde.

OpCode

Der zu den Daten zugehörige [Opcode](#)⁶².

SndBufferSize

Die Größe der zu sendenden Daten in Bytes.

SndBuffer

Der Puffer mit den zu sendenden Daten. Dieser muss mindestens die Größe „SndBufferSize“ haben.

RcvBufferSize

Die Größe des Empfangspuffers in Bytes. Dieser muss ausreichend Platz für die Antwort-Daten bereithalten.

RcvBuffer

Der Puffer für die Antwort-Daten. Diese muss mindestens die Größe „RcvBufferSize“ haben.

BytesReceived

Die Größe der empfangenen Daten in Bytes.

Timeout

Der Timeout für den Datenaustausch in ms, z.B. 500.

Rückgabewert

Die Funktion gibt im Erfolgsfall MSC_STATUS_SUCCESS zurück, bei Fehlern wird der [Fehlercode](#)²⁶ zurückgegeben.

Kommentare

Diese Funktion kann genutzt werden, um einmalig ein Telegramm zu senden und um Antwort vom Gerät zu erhalten. Der Datentransfer muss gestartet sein. Die MSC DLL wartet, bis die Daten zum Irinos-System gesendet wurden und die Antwort empfangen wurde oder ein Timeout aufgetreten ist.

Siehe auch

Wird für folgende [Opcodes](#)⁶² verwendet: [opcRIV](#)⁶⁷, [opcRMI](#)⁶⁸, [opcWCC](#)⁷⁶, [opcRCA](#)⁷⁹, [opcWCA](#)⁸², [opcWCL](#)⁸⁵, [opcRCL](#)⁸⁷, [opcACL](#)⁸⁹, [opcDT](#)⁹¹, [opcAT](#)⁹⁶, [opcIT](#)⁹⁷, [opcSP](#)⁹⁹, [opcREv](#)¹⁰⁶, [opcSAbst](#)¹⁰⁷, [opcWEvCfg](#)¹¹⁰, [opcRSW](#)¹²⁰, [opcRST](#)¹²⁹

Kann auch anstelle von [MSC_SetupStaticChannel](#)⁴⁹ & [MSC_ReadStatic](#)⁵¹ für folgende Opcodes verwendet werden: [opcRS](#)¹¹⁴, [opcRHS](#)¹⁰², [opcBIO](#)¹¹⁵

4.3 Verbindung

4.3.1 MSC_EnumerateDevices

Diese Funktion zählt alle konfigurierten Geräte (d.h. Irinos-Systeme) auf und gibt die Anzahl der verfügbaren Geräte zurück.

Definition

```
MSC_STATUS
MSC_EnumerateDevices(
    unsigned long* DeviceNumber
);
```

Parameter

DeviceNumber

Gibt die Zahl der verfügbaren Geräte zurück.

Rückgabewert

Die Funktion gibt im Erfolgsfall MSC_STATUS_SUCCESS zurück, bei Fehlern wird der [Fehlercode](#)²⁶ zurückgegeben.

Kommentare

Die MscDll kann mit mehreren Irinos-Systemen gleichzeitig kommunizieren. In den allermeisten Anwendungen wird jedoch genau 1 Irinos-System verwendet. Der Parameter DeviceNumber hat dann den Wert 1.

Das Verhalten dieser Funktion hängt von den Einstellungen in der Konfigurationsdatei [Msc.cfg](#)²⁰ ab. Falls neue Geräte hinzugefügt werden, muss die Funktion sie erneut öffnen.

Die Funktion sucht nicht nach Irinos-Boxen, sondern nach Irinos-Systemen (d.h. nach Irinos-Boxen mit Ethernet-Schnittstelle).

Siehe auch

[MSC_GetDeviceInfo](#)³³

[MSC_OpenDevice](#)³⁴

4.3.2 MSC_GetDeviceInfo

Diese Funktion gibt Informationen über ein Gerät (d.h. Irinos-System) zurück.

Definition

```
MSC_STATUS
MSC_GetDeviceInfo(
    unsigned long Index,
    unsigned long* BusType,
    char UniqueID[MSC_MAX_UNIQUEID_SIZE]
);
```

Parameter

Index

Nummer des Gerätes. Das erste Gerät hat die Nummer 0. Die Anzahl der Geräte kann mit [MSC_EnumerateDevices](#)³² ermittelt werden.

BusType

Gibt den Bustyp zurück, mit dem das Gerät verbunden ist. Einziger unterstützter Bustyp ist MSC_TYPE_SOCKET_XPORT (Wert 0x00000001).

UniqueID

Gibt einen nullterminierten ASCII-String zurück mit der IP-Netzwerkadresse.
 MSC_MAX_UNIQUEID_SIZE hat den Wert 40, d.h. die ID kann maximal 40 Bytes lang sein.

Rückgabewert

Die Funktion gibt im Erfolgsfall MSC_STATUS_SUCCESS zurück, bei Fehlern wird der [Fehlercode](#)¹²⁶ zurückgegeben.

Kommentare

Zur Kompatibilität mit älteren Systemen unterstützt die DLL auch den Bustyp MSC_TYPE_USB_FTDI (Wert 0x00000002). Dieser Bustyp hat jedoch bei Neu-Systemen keine Relevanz mehr.

Siehe auch

[MSC_EnumerateDevices](#)¹³²

4.3.3 MSC_OpenDevice

Diese Funktion öffnet die Verbindung zu einem Gerät (d.h. zu einem Irinos-System).

Definition

```
MSC_STATUS
MSC_OpenDevice(
    unsigned long Index,
    MSC_HANDLE* Handle
);
```

Parameter

Index

Nummer des Gerätes. Das erste Gerät hat die Nummer 0. Die Anzahl der Geräte kann mit [MSC_EnumerateDevices](#)^[32] ermittelt werden.

Handle

Gibt ein Handle zum Gerät zurück. Das Handle wird zur Kommunikation mit dem Gerät von verschiedenen DLL-Funktionen benötigt.

Rückgabewert

Die Funktion gibt im Erfolgsfall MSC_STATUS_SUCCESS zurück, bei Fehlern wird der [Fehlercode](#)^[26] zurückgegeben.

Es können mehrere Handles zu demselben Gerät erzeugt werden.

Siehe auch

[MSC_EnumerateDevices](#)^[32]

4.3.4 MSC_InitDevice

Diese Funktion initialisiert die internen Geräte-Daten.

Definition

```
MSC_STATUS
MSC_InitDevice(
    MSC_HANDLE Handle
);
```

Parameter

Handle

Handle zu dem Gerät (Irinos-System), das bei einem früheren Aufruf der Funktion [MSC_OpenDevice](#)^[34] zurückgegeben wurde.

Rückgabewert

Die Funktion gibt im Erfolgsfall MSC_STATUS_SUCCESS zurück, bei Fehlern wird der [Fehlercode](#)²⁶ zurückgegeben.

Kommentare

Alle Kanalinformationen, Benachrichtigungen und Fehlerzähler werden gelöscht.

Siehe auch

[MSC_OpenDevice](#)³⁴

4.3.5 MSC_CloseDevice

Diese Funktion schließt den Handle eines Geräts.

Definition

```
MSC_STATUS
MSC_CloseDevice(
    MSC_HANDLE Handle
);
```

Parameter

Handle

Handle zu dem Gerät (Irinos-System), das bei einem früheren Aufruf der Funktion [MSC_OpenDevice](#)³⁴ zurückgegeben wurde.

Rückgabewert

Die Funktion gibt im Erfolgsfall MSC_STATUS_SUCCESS zurück, bei Fehlern wird der [Fehlercode](#)¹²⁶ zurückgegeben.

Kommentare

Falls das letzte Handle eines Geräts geschlossen wird, gehen alle internen Information über den Zustand des Geräts verloren.

Siehe auch

[MSC_OpenDevice](#)³⁴

4.3.6 MSC_Start

Diese Funktion startet den Datentransfer.

Definition

```
MSC_STATUS
MSC_Start(
    MSC_HANDLE Handle,
    unsigned long SndPeriod,
    unsigned long DisconnectTimeout,
    unsigned long RetryCount,
    unsigned long ResponseTimeout
);
```

Parameter

Handle

Handle zu dem Gerät (Irinos-System), das bei einem früheren Aufruf der Funktion [MSC_OpenDevice](#)³⁴ zurückgegeben wurde.

SndPeriod

[ms] Abstand zwischen zwei UDP-Telegrammen, die an das Gerät gesendet werden. Falls die Dauer geringer ist, als die Zeit, die benötigt wird, um eine Antwort vom Gerät zu erhalten, wird das nächste Telegramm sobald wie möglich gesendet.

DisconnectTimeout

[ms] Falls in diesem Zeitintervall keine Antwort vom Gerät (Irinos-System) empfangen wurde, wird die [Geräte-Benachrichtigung](#)⁴² gesetzt.

RetryCount

Anzahl an Telegrammwiederholungen, bevor ein Fehler festgestellt wird.

ResponseTimeout

[ms] Zeitintervall, nach dem ein Telegramm bei ausbleibender Antwort wiederholt wird.

Rückgabewert

Die Funktion gibt im Erfolgsfall MSC_STATUS_SUCCESS zurück, bei Fehlern wird der [Fehlercode](#)²⁶ zurückgegeben.

Kommentare

Der Datentransfer muss über diese Funktion gestartet werden, bevor irgendeine Art der Kommunikation stattfinden kann.

Beispielwerte

Folgende Beispielwerte sind für statische Messungen sowie für die meisten dynamischen Messungen sehr gut geeignet:

SndPeriod = 1
 DisconnectTimeout = 500
 RetryCount = 10
 ResponseTimeout = 75

Um eine möglichst kurze Sende-Periode zu erreichen, muss unter Windows ein Multimedia-Timer mit 1ms gestartet werden. Dieser setzt die Windows-Tick-Time vom Standardwert (15,6ms) auf ≤ 1 ms herab. Es wird empfohlen diesen Multimedia-Timer in die Applikation zu integrieren.

Alle Beispielwerte beziehen sich auf eine direkte Ethernet-Verbindung zwischen PC und Irinos-System.

Siehe auch

[MSC_OpenDevice](#)³⁴

[MSC_Stop](#)³⁹

4.3.7 MSC_Stop

Diese Funktion stoppt den Datentransfer.

Definition

```
MSC_STATUS
MSC_Stop(
    MSC_HANDLE Handle
);
```

Parameter

Handle

Handle zu dem Gerät (Irinos-System), das bei einem früheren Aufruf der Funktion [MSC_OpenDevice](#)³⁴ zurückgegeben wurde.

Rückgabewert

Die Funktion gibt im Erfolgsfall `MSC_STATUS_SUCCESS` zurück, bei Fehlern wird der [Fehlercode](#)^{□26} zurückgegeben.

Siehe auch

[MSC_OpenDevice](#)^{□34}

[MSC_Start](#)^{□37}

4.3.8 MSC_GetDeviceState

Diese Funktion gibt den Kommunikationsstatus zum Irinos-System zurück. Sie dient reinen Diagnose-Zwecken. Für den Normalbetrieb ist Sie nicht erforderlich. Ihre Implementierung ist jedoch empfohlen, um im Fehlerfall erweiterte Diagnose-Möglichkeiten zu haben.

Definition

```
MSC_STATUS
MSC_GetDeviceState(
    MSC_HANDLE Handle,
    unsigned long* LastMsgReceived,
    unsigned long* SndErrorCounter,
    unsigned long* RcvErrorCounter,
    unsigned long* CmdDiscarded,
    unsigned long* CmdDiscardedArray,
    unsigned long Flags
);
```

Parameter

Handle

Handle zum Gerät, das bei einem früheren Aufruf der Funktion [MSC_OpenDevice](#)^{□34} zurückgegeben wurde.

LastMsgReceived

Gibt das Zeitintervall zurück zwischen jetzt und dem Zeitpunkt des Empfangs der letzten gültigen Daten-Pakets.

SndErrorCounter

Gibt den Fehlerzähler der physikalisch gesendeten Übertragungsfehler zurück. Die Zähler werden auf null gesetzt, entweder durch die Funktion [MSC_Start](#)^{□37} oder falls der Flags value MSC_RESET_ERROR_COUNTERS gesetzt ist.

RcvErrorCounter

Gibt den Fehlerzähler der physikalisch empfangenen Übertragungsfehler. Die Zähler werden auf null gesetzt, entweder durch die Funktion [MSC_Start](#)^{□37} oder falls der Flags value MSC_RESET_ERROR_COUNTERS gesetzt ist.

CmdDiscarded

Gibt einen Zähler zurück, der jedes Mal ansteigt, wenn ein [Opcode](#)^{□62} empfangen wird, der nicht erwartet wurde. Dieser Zähler ist Übersicht des CmdDiscardedArray. Die Zähler werden auf null gesetzt, entweder durch die Funktion [MSC_Start](#)^{□37} oder falls der Flags value MSC_RESET_DISCARDED_COUNTERS gesetzt ist.

CmdDiscardedArray

Pointer zu einem nutzergestellten Array von 256 unsigned long values. Jeder Wert enthält den Fehlerzähler für einen Opcode spezifiziert durch einen Index. Die Zähler werden auf null gesetzt, entweder durch die Funktion [MSC_Start](#)^{□37} oder falls der Flags value MSC_RESET_DISCARDED_COUNTERS gesetzt ist.

Flags

Enthält bitweise Informationen für die MSC DLL. Die Werte können auch kombiniert auftreten.

MSC_RESET_ERROR_COUNTERS
Dieses Bit löscht den Sende- und Empfangs Fehlerzähler.

MSC_RESET_DISCARDED_COUNTERS
Dieses Bit löscht den Zähler für ungültige Telegramme.

Rückgabewert

Die Funktion gibt im Erfolgsfall `MSC_STATUS_SUCCESS` zurück, bei Fehlern wird der [Fehlercode](#)^{□26} zurückgegeben.

Siehe auch

[MSC_OpenDevice](#)^{□34}

[MSC_Start](#)^{□37}

4.4 Benachrichtigungen (Notifications)

Über Benachrichtigungen kann die Applikation über folgende Ereignisse informiert werden:

- a) Verbindungsabbruch („DisconnectTimeout“, siehe [MSC_Start](#)^{□37})
- b) Neu eingetroffene Daten bei einem statischen Kanal (siehe [MSC_SetupStaticChannel](#)^{□49})
- c) Ein voller Empfangspuffer bei einer dynamischen Messung.

Die Verwendung von Benachrichtigungen ist empfohlen, jedoch nicht notwendig.

Bei der Verwendung von Benachrichtigungen ist darauf zu achten, **dass die Benachrichtigung selbst und das Auslesen, Interpretieren und Darstellen von Daten in separaten Threads ablaufen sollten**. Ansonsten kann es zu Kommunikations-Verzögerungen oder gar Kommunikations-Unterbrechungen kommen.

In der Praxis hat sich folgende Vorgehensweise als Vorteilhaft herausgestellt:

- Beim Auftreten einer Benachrichtigung wird ein Flag gesetzt.
- In einem Thread wird dieses Flag zyklisch geprüft. Ist es gesetzt, werden die neuen Daten ausgelesen und ausgewertet. Das Flag wird zurückgesetzt.

Die zyklische Prüfung kann beispielsweise ein 30ms Timer-Event der GUI sein.

Wir empfehlen die Verwendung von Messages ([MSC_SetNotificationMessage](#)¹⁴³).

4.4.1 MSC_SetNotificationMessage

Diese Funktion aktiviert das Senden von Windows Messages für Benachrichtigungen.

Details zum Empfang von Windows Messages entnehmen Sie der Dokumentation Ihrer Entwicklungsumgebung.

Definition

```
MSC_STATUS
MSC_SetNotificationMessage(
    MSC_HANDLE Handle,
    int OpCode,
    HWND hWnd,
    ULONG MsgCode,
    ULONG wParam,
    ULONG lParam
);
```

Parameter

Handle

Handle zu dem Gerät (Irinos-System), das bei einem früheren Aufruf der Funktion [MSC_OpenDevice](#)³⁴ zurückgegeben wurde.

Opcode

- Der Opcode des statischen Kanals, für den eine Message beim Eintreffen neuer Daten gesendet werden soll ([opcRS](#)¹¹⁴, [opcRHS](#)¹⁰² oder [opcBIO](#)¹¹⁵).
- -1 für eine Benachrichtigung bei einem Verbindungstimeout (siehe [MSC_Start](#)³⁷).
- Der Opcode der dynamischen Messung, für die eine Message bei einem vollen Empfangspuffer gesendet werden soll ([opcRDM1](#)¹²⁷ oder [opcRDM2](#)¹²⁷).

hWnd

Ein Windows-Handle, das die Message empfängt.

MsgCode

Von der Applikation festzulegende Message-Nummer.

wParam

Der wParam der Windows-Message.

lParam

Der lParam der Windows-Message.

Rückgabewert

Die Funktion gibt im Erfolgsfall `MSC_STATUS_SUCCESS` zurück, bei Fehlern wird der [Fehlercode](#)²⁶ zurückgegeben.

Kommentare

Die Benachrichtigung wird nur für Geräte sowie für statische und dynamische Kanäle verwendet. Bei einem Gerät stellt die Benachrichtigung einen Fehler fest.

Bei einem statischen Kanal wird die Benachrichtigung jedes Mal gesendet, wenn Daten empfangen werden; einmal in jeder Sendeperiode.

Bei einem dynamischen Kanal wird die Benachrichtigung nur gesendet, wenn der Empfangspuffer vollständig gefüllt ist.

Die Benachrichtigung kann auch aufgehoben werden, wenn die Funktion mit `hWnd = NULL` aufgerufen wird.

Anmerkung zum MsgCode

Die Message-Nummer wird von der Applikation beliebig festgelegt. Unter Visual C++ kann dies beispielsweise wie folgt aussehen:

```
#define WM_MESSAGE_MSC_READSTATIC    (WM_USER + 0)
#define WM_MESSAGE_MSC_BITIO        (WM_USER + 1)
#define WM_MESSAGE_MSC_HW_STATUS    (WM_USER + 2)
```

Siehe auch

[MSC_OpenDevice](#)³⁴

[MSC_Start](#)³⁷

4.4.2 MSC_SetNotificationEvent

Diese Funktion registriert einen Event zur Benachrichtigung.

Details zur Anwendung von Events entnehmen Sie der Dokumentation Ihrer Entwicklungsumgebung.

Definition

```
MSC_STATUS
MSC_SetNotificationEvent (
    MSC_HANDLE Handle,
    int OpCode,
    HANDLE Event
);
```

Parameter

Handle

Handle zu dem Gerät (Irinos-System), das bei einem früheren Aufruf der Funktion [MSC_OpenDevice](#)³⁴ zurückgegeben wurde.

Opcode

- Der Opcode des statischen Kanals, für den eine Message beim Eintreffen neuer Daten gesendet werden soll ([opcRS](#)¹¹⁴, [opcRHS](#)¹⁰² oder [opcBIO](#)¹¹⁵).
- -1 für eine Benachrichtigung bei einem Verbindungstimeout (siehe [MSC_Start](#)³⁷).
- Der Opcode der dynamischen Messung, für die eine Message bei einem vollen Empfangspuffer gesendet werden soll ([opcRDM1](#)¹²⁷ oder [opcRDM2](#)¹²⁷).

Event

Das initialisierte Event-Handle.

Rückgabewert

Die Funktion gibt im Erfolgsfall `MSC_STATUS_SUCCESS` zurück, bei Fehlern wird der [Fehlercode](#)²⁶ zurückgegeben.

Kommentare

Die Benachrichtigung wird nur für Geräte und für statische sowie dynamische Kanäle genutzt.

Bei einem Gerät stellt die Benachrichtigung einen Fehler fest.

Bei einem statischen Kanal wird die Benachrichtigung jedes Mal gesendet, wenn Daten empfangen werden; einmal in jeder Sendeperiode.

Bei einem dynamischen Kanal wird die Benachrichtigung nur gesendet, wenn der Empfangspuffer komplett voll ist.

Die Benachrichtigung kann auch nicht registriert sein, wenn die Funktion mit `Event=NULL` aufgerufen wird.

Die DLL setzt diesen Event niemals zurück.

Siehe auch

[MSC_OpenDevice](#)^{□34}

[MSC_Start](#)^{□37}

4.4.3 MSC_SetNotificationCallback

Diese Funktion registriert eine Callback-Funktion zur Benachrichtigung.

Definition

```
MSC_STATUS
MSC_SetNotificationCallback(
    MSC_HANDLE Handle,
    int OpCode,
    MSC_NOTIFICATION_CALLBACK* CallbackFunction,
    void* NotificationContext
);
```

Parameter

Handle

Handle zu dem Gerät (Irinos-System), das bei einem früheren Aufruf der Funktion [MSC_OpenDevice](#)^{□34} zurückgegeben wurde.

Opcode

- Der Opcode des statischen Kanals, für den eine Message beim Eintreffen neuer Daten gesendet werden soll ([opcRS](#)^{□114}, [opcRHS](#)^{□102} oder [opcBIO](#)^{□115}).
- -1 für eine Benachrichtigung bei einem Verbindungstimeout (siehe [MSC_Start](#)^{□37}).
- Der Opcode der dynamischen Messung, für die eine Message bei einem vollen Empfangspuffer gesendet werden soll ([opcRDM1](#)^{□127} oder [opcRDM2](#)^{□127}).

CallbackFunction

Die Adresse der Callback-Funktion. Eingabe von NULL, um die Registrierung aufzuheben.

NotificationContext

Ein Zeiger auf kontextabhängige Daten, der unverändert an die [Callback Funktion](#)^{□48} weitergegeben wird.

Rückgabewert

Die Funktion gibt im Erfolgsfall MSC_STATUS_SUCCESS zurück, bei Fehlern wird der [Fehlercode](#)^{□26} zurückgegeben.

Kommentare

Die Benachrichtigung wird nur für Geräte sowie für statische und dynamische Kanäle verwendet. Bei einem Gerät stellt die Benachrichtigung einen Fehler fest.

Bei einem statischen Kanal wird die Benachrichtigung jedes Mal gesendet, wenn Daten empfangen werden; einmal in jeder Sendeperiode.

Bei einem dynamischen Kanal wird die Benachrichtigung nur gesendet, wenn der Empfangspuffer vollständig gefüllt ist.

Die Benachrichtigung kann auch aufgehoben werden, wenn die Funktion mit Callback=NULL aufgerufen wird.

Die Callback-Funktion wird aus dem Thread der Kommunikation heraus aufgerufen. Führen Sie nur wenig Code in der Callback-Funktion aus, da sonst die Kommunikation ausgebremst oder unterbrochen werden kann.

Siehe auch

[MSC_OpenDevice](#)^{□34}

[MSC_Start](#)^{□37}

Callback-Funktion

Diese Funktion ist ein Prototyp für eine callback-Benachrichtigung.

Definition

```
void
MSC_NOTIFICATION_CALLBACK(
    void* NotificationContext
);
```

Parameter

NotificationContext

Dieser Parameter ist der gleiche, der an die Funktion [MSC_SetNotificationCallback](#)⁴⁷ gegeben wurde. Die Anwendung kann über diesen Pointer Kontextinformation speichern.

Kommentare

Diese Funktion wird in einem anderen Thread Kontext aufgerufen. Die Anwendung muss für die Synchronisation sorgen.

4.5 Statische Übertragungskanäle

4.5.1 MSC_SetupStaticChannel

Diese Funktion initialisiert einen statischen Übertragungs-Kanal. Bei einem statischen Übertragungs-Kanal werden Daten automatisch und kontinuierlich zwischen der DLL und dem Irinos-System ausgetauscht.

Definition

```

MSC_STATUS
MSC_SetupStaticChannel(
    MSC_HANDLE Handle,
    unsigned char OpCode,
    unsigned long SndBufferSize,
    void* SndBuffer,
    unsigned long RcvBufferSize
);
    
```

Parameter

Handle

Handle zu dem Gerät (Irinos-System), das bei einem früheren Aufruf der Funktion [MSC_OpenDevice](#)¹³⁴ zurückgegeben wurde.

Opcode

Der Opcode des statischen Kanals:
[opcRS](#)¹¹⁴ für das Auslesen statischer Messwerte.
[opcRHS](#)¹⁰² für das Auslesen des Hardware-Status.
[opcBIO](#)¹¹⁵ für den Austausch von Bit I/Os.

SndBufferSize

Die Größe der zu sendenden Daten in Bytes.

SndBuffer

Puffer mit den zu sendenden Daten. Dieser muss mindestens die Größe SndBufferSize haben.

RcvBufferSize

Die maximale Größe der Empfangsdaten.

Rückgabewert

Die Funktion gibt im Erfolgsfall MSC_STATUS_SUCCESS zurück, bei Fehlern wird der [Fehlercode](#)²⁶ zurückgegeben.

Kommentare

Die DLL vergibt einen Puffer für die Empfangsdaten. Diese werden überschrieben, wenn neue Daten empfangen werden. Sie können mit dem Befehl [MSC_ReadStatic](#)⁵¹ gelesen werden.

Der Opcode ist entweder der [opcRS](#)¹¹⁴, [opcRHS](#)¹⁰² oder [opcBIO](#)¹¹⁵.

Der Sendepuffer wird für [opcRS](#)¹¹⁴ nicht genutzt. Er muss aber immer definiert werden und eine Größe von zumindest einem Byte haben.

Sobald die Daten im Sendepuffer geändert wurden, muss die DLL über [MSC_RefreshChannel](#)⁵³ darüber informiert werden.

Siehe auch

[MSC_OpenDevice](#)³⁴

[MSC_Start](#)³⁷

[MSC_ReadStatic](#)⁵¹

[MSC_RefreshChannel](#)⁵³

4.5.2 MSC_ReadStatic

Diese Funktion liest die neuesten Daten von einem statischen Übertragungs-Kanal.

Definition

```
MSC_STATUS
MSC_ReadStatic(
    MSC_HANDLE Handle,
    unsigned char OpCode,
    unsigned long BufferSize,
    void* Buffer,
    unsigned long* DataCount
);
```

Parameter

Handle

Handle zu dem Gerät (Irinos-System), das bei einem früheren Aufruf der Funktion [MSC_OpenDevice](#)¹³⁴ zurückgegeben wurde.

Opcode

Der Opcode des statischen Kanals:
[opcRS](#)¹¹⁴ für das Auslesen statischer Messwerte.
[opcRHS](#)¹⁰² für das Auslesen des Hardware-Status.
[opcBIO](#)¹¹⁵ für das Auslesen der digitalen Eingänge.

BufferSize

Die Größe des Puffers, in welchen die empfangenen Daten kopiert werden sollen (Einheit: Bytes).

Buffer

Puffer, in welchen die empfangenen Daten kopiert werden. Dieser muss mindestens die Größe BufferSize haben.

DataCount

Die Anzahl der Bytes, die in letzter Zeit empfangen wurden. 0, falls keine neuen Daten empfangen wurden.

Rückgabewert

Die Funktion gibt im Erfolgsfall MSC_STATUS_SUCCESS zurück, bei Fehlern wird der [Fehlercode](#)²⁶ zurückgegeben.

Kommentare

Ein statischer Messkanal muss mit der Funktion [MSC_SetupStaticChannel](#)⁴⁹ initialisiert werden. Als Opcode wird dann [MSC_ReadStatic](#)⁵¹ der Opcode des statischen Kanals übergeben, um die empfangenen Werte zu lesen.

Siehe auch

[MSC_OpenDevice](#)³⁴

[MSC_SetupStaticChannel](#)⁴⁹

4.5.3 MSC_RefreshChannel

Diese Funktion aktualisiert den Sendepuffer des dazugehörigen Opcodes.

Definition

```
MSC_STATUS
MSC_RefreshChannel (
    MSC_HANDLE Handle,
    unsigned char OpCode
);
```

Parameter

Handle

Handle zu dem Gerät (Irinos-System), das bei einem früheren Aufruf der Funktion [MSC_OpenDevice](#)³⁴ zurückgegeben wurde.

Opcode

Der Opcode, dessen Sendepuffer geändert wurde ([opcBIO](#)¹¹⁵).

Rückgabewert

Die Funktion gibt im Erfolgsfall MSC_STATUS_SUCCESS zurück, bei Fehlern wird der [Fehlercode](#)²⁶ zurückgegeben.

Kommentare

Diese Funktion wird benutzt, um Outputdaten innerhalb der DLL zu aktualisieren. Einzige Anwendung hierfür sind derzeit die digitalen Ausgänge ([opcBIO](#)¹¹⁵).

[Der Opcode [opcRHS](#)¹⁰² muss nicht aktualisiert werden, da die Ausgangsdaten immer gleich sind.]

Diese Funktion löst kein Telegramm auf der Netzwerkschnittstelle aus. Sie ist lediglich eine Information für die DLL.

Siehe auch

[MSC_OpenDevice](#)³⁴

[MSC_SetupStaticChannel](#)⁴⁹

4.6 Übertragungs-Kanal für dynamische Messwerte

4.6.1 MSC_SetupExtendedDynamicChannel

Diese Funktion initialisiert einen Übertragungs-Kanal für dynamische Messwerte.

Definition

```
MSC_STATUS
MSC_SetupExtendedDynamicChannel (
    MSC_HANDLE Handle,
    unsigned char OpCode,
    unsigned char NumberOfSubChannels,
    unsigned long SndBufferSize,
    void* SndBuffer
);
```

Parameter

Handle

Handle zu dem Gerät (Irinos-System), das bei einem früheren Aufruf der Funktion [MSC_OpenDevice](#)³⁴ zurückgegeben wurde.

Opcode

Der Opcode der dynamischen Messung ([opcRDM1](#)¹²⁷ oder [opcRDM2](#)¹²⁷).

NumberOfSubChannels

Die Anzahl der Messkanäle, die in einem Telegramm übermittelt werden.

SndBufferSize

Die Größe des Sendepuffers muss 1 sein.

SndBuffer

Der Zeiger zum Sendepuffer. Bytepuffer mit mindestens einem Byte Länge. Der Inhalt des Sendepuffers ist nicht relevant. Der Sendepuffer dient lediglich dazu, eine feste Datenstruktur herzustellen.

Rückgabewert

Die Funktion gibt im Erfolgsfall MSC_STATUS_SUCCESS zurück, bei Fehlern wird der [Fehlercode](#)²⁶ zurückgegeben.

Kommentare

Ein dynamischer Übertragungs-Kanal speichert die Messwerte in FIFO – ähnlichen Puffern. Die Puffer müssen von der Anwendung mit [MSC_AttachSubChannelBuffer](#)⁵⁶ bereitgestellt werden. Die Messwert-Übertragung läuft, solange Puffer an alle Subkanäle angehängt sind und diese Puffer nicht voll sind.

Alle Messwerte sind 32 Bit signed long. Dies gilt auch für Messkanäle, deren Messbereich 16 Bit oder 8 Bit ist.

Siehe auch

[MSC_OpenDevice](#)³⁴

[MSC_AttachSubChannelBuffer](#)^{□56}

[MSC_GetPosition](#)^{□59}

4.6.2 MSC_AttachSubChannelBuffer

Diese Funktion fügt einen Messwert-Puffer zu einem dynamischen Übertragungs-Kanal hinzu. Je Messkanal, der in der dynamischen Messung verwendet wird, muss ein Puffer hinzugefügt werden.

Definition

```
MSC_STATUS
MSC_AttachSubChannelBuffer(
    MSC_HANDLE Handle,
    unsigned char OpCode,
    unsigned char SubChannel,
    unsigned long BufferSize,
    void* Buffer
);
```

Parameter

Handle

Handle zu dem Gerät (Irinos-System), das bei einem früheren Aufruf der Funktion [MSC_OpenDevice](#)^{□34} zurückgegeben wurde.

Opcode

Der Opcode der dynamischen Messung ([opcRDM1](#)^{□127} oder [opcRDM2](#)^{□127}).

SubChannel

Nummer des Messkanals. Die Nummer des ersten Kanals ist 0. Sie muss kleiner sein als NumberOfSubChannels in der Funktion [MSC_SetupExtendedDynamicChannel](#)^{□49}.

BufferSize

Die Größe des Messwert-Puffers. Jeder Messwert benötigt 4 Bytes. Zum Beispiel wird bei 1000 Messwerten pro Messkanal ein Puffer mit der Größe 4000 Bytes benötigt.

Buffer

Zeiger auf den Messwertpuffer. Dieser muss mindestens die Größe BufferSize haben.

Rückgabewert

Die Funktion gibt im Erfolgsfall MSC_STATUS_SUCCESS zurück, bei Fehlern wird der [Fehlercode](#)²⁶ zurückgegeben.

Kommentare

Der Speicher für den Puffer wird vom Aufrufer bereitgestellt (d.h. von der Applikation) und muss permanent sein, bis der Puffer mit der Funktion [MSC_DetachSubChannelBuffers](#)⁵⁷ abgetrennt wird. Die Funktion muss wiederholt aufgerufen werden, einmal für jeden Mess-Kanal.

Der Opcode ist derselbe, der mit dem Befehl [MSC_SetupExtendedDynamicChannel](#)⁴⁹ gesendet wurde.

Siehe auch

[MSC_OpenDevice](#)³⁴

[MSC_SetupExtendedDynamicChannel](#)⁴⁹

[MSC_DetachSubChannelBuffers](#)⁵⁷

[MSC_GetPosition](#)⁵⁹

4.6.3 MSC_DetachSubChannelBuffers

Diese Funktion trennt alle Messwert-Puffer von einem dynamischen Übertragungs-Kanal.

Definition

```

MSC_STATUS
MSC_DetachSubChannelBuffers (
    MSC_HANDLE Handle,
    unsigned char OpCode
);
    
```

Parameter

Handle

Handle zu dem Gerät (Irinos-System), das bei einem früheren Aufruf der Funktion [MSC_OpenDevice](#)³⁴ zurückgegeben wurde.

Opcode

Der Opcode der dynamischen Messung ([opcRDM1](#)¹²⁷ oder [opcRDM2](#)¹²⁷).

Rückgabewert

Die Funktion gibt im Erfolgsfall MSC_STATUS_SUCCESS zurück, bei Fehlern wird der [Fehlercode](#)²⁶ zurückgegeben.

Kommentare

Die Funktion trennt alle an die Subkanäle eines erweiterten dynamischen Kanals angehängten Puffer durch den Aufruf [MSC_AttachSubChannelBuffer](#)⁵⁶. Falls die Puffer abgetrennt werden, stoppt die dynamische Messung innerhalb der DLL.

Siehe auch

[MSC_OpenDevice](#)³⁴

[MSC_SetupExtendedDynamicChannel](#)⁴⁹

[MSC_AttachSubChannelBuffer](#)⁵⁶

[MSC_GetPosition](#)^{□59}

4.6.4 MSC_GetPosition

Diese Funktion gibt die Position innerhalb der dynamischen Messung wieder. Damit lässt sich bestimmen, wieviel Messwerte einer dynamischen Messung bereits aufgezeichnet und übertragen wurden.

Definition

```
MSC_STATUS
MSC_GetPosition(
    MSC_HANDLE Handle,
    unsigned char OpCode,
    unsigned long* Position
);
```

Parameter

Handle

Handle zu dem Gerät (Irinos-System), das bei einem früheren Aufruf der Funktion [MSC_OpenDevice](#)^{□34} zurückgegeben wurde.

Opcode

Der Opcode der dynamischen Messung ([opcRDM1](#)^{□127} oder [opcRDM2](#)^{□127}).

Position

Gibt die Position innerhalb der dynamischen Messung in Bytes zurück.

Rückgabewert

Die Funktion gibt im Erfolgsfall MSC_STATUS_SUCCESS zurück, bei Fehlern wird der [Fehlercode](#)^{□26} zurückgegeben.

Kommentare

Messwerte werden immer als 32 Bit – Wert abgelegt. Um die Anzahl der Messwerte zu erhalten, muss daher der Wert „Position“ durch 4 geteilt werden.

Beispiel: Beinhaltet ein Puffer 400 Bytes, so entspricht dies 100 Messwerten.

Um Herauszufinden, ob bereits alle dynamischen Messwert-Samples zum PC übertragen wurden, kann die Anzahl der Messwert-Samples mit dem Opcode [opcRDC](#)¹²³ ausgelesen werden.

Siehe auch

[MSC_OpenDevice](#)³⁴

[MSC_SetupExtendedDynamicChannel](#)⁴⁹

[MSC_AttachSubChannelBuffer](#)⁵⁶

Opcodes und Parameter

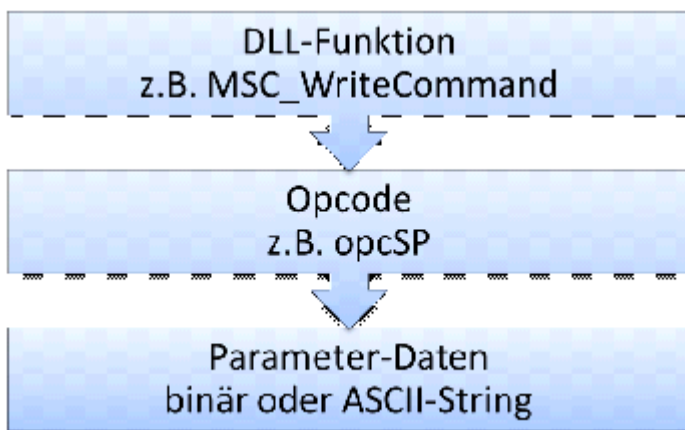
5 Opcodes und Parameter

5.1 Bedeutung von Opcodes

Die Applikation kommuniziert mit den Irinos-Boxen über die MscDll. Die DLL kann über Funktionsaufrufe angesprochen werden. Die meisten dieser DLL-Funktionen benötigen Zusatzdaten.

Zur Unterscheidung der Zusatzdaten werden Opcodes übergeben. Jeder Opcode ist an eine bestimmte Funktion gekoppelt. An jeden Opcode sind wiederum Parameter-Daten gekoppelt, die zum bzw. vom Irinos-System übertragen werden.

Bei den Parameter-Daten kann es sich um binäre Daten oder um einen ASCII-String handeln.



5.2 Opcode-Übersicht

Opcodes "Initialisierung"

Opcode	Hex-Wert	Funktion	Parameter-Typ
opcRIV ⁶⁷	0x01	Lese Box-Übersicht	String ⁶⁶
opcRMI ⁶⁸	0x03	Lese Box-Informationen (digitales Typenschild)	String ⁶⁶

Opcode	Hex-Wert	Funktion	Parameter-Typ
opcRSS ^{□74}	0x05	Lese System-String	String ^{□66}
opcWCC ^{□76}	0x09	Schreibe Kanal-Eigenschaften	String ^{□66}
opcRCA ^{□79}	0x10	Lese Kanal-Zuordnung	String ^{□66}
opcWCA ^{□82}	0x11	Schreibe Kanal-Zuordnung	String ^{□66}

Opcodes "Konfiguration & Verschiedene"

Opcode	Hex-Wert	Funktion	Parameter-Typ
opcWCL ^{□85}	0x22	Schreibe Kanal-Liste	String ^{□66}
opcRCL ^{□87}	0x23	Lese Kanal-Liste	String ^{□66}
opcACL ^{□89}	0x24 (0x26)	Aktiviere Kanal-Liste für die statische Messung	String ^{□66}
opcDT ^{□91}	0x30	Definiere Trigger für die dynamische Messung	String ^{□66}

Opcode	Hex-Wert	Funktion	Parameter-Typ
opcAT ^{□96}	0x31	Aktiviere Trigger für die dynamische Messung	String ^{□66}
opcIT ^{□97}	0x32	Deaktiviere Trigger für die dynamische Messung	String ^{□66}
opcSP ^{□99}	0x35	Setze (Messkanal-) Parameter	String ^{□66}
opcRHS ^{□102}	0x38	Lese Hardware-Status der Messkanäle	Binär
opcREv ^{□106}	0x39	Lese aktuelles Ereignis der jeweiligen Irinos-Boxen	Binär
opcSAbsT ^{□107}	0x3A	Setze Absolut-Zeit (für den Diagnose-Speicher)	String ^{□66}
opcWEvCfg ^{□110}	0x3D	Schreibe Ereignis-Konfiguration	String ^{□66}
opcClrEv ^{□112}	0x3E	Lösche Ereignis	String ^{□66}

Opcodes "Messung"

Opcode	Hex-Wert	Funktion	Parameter-Typ
opcRS ¹¹⁴	0x40	Lese statische Messwerte	Binär
opcBIO ¹¹⁵	0x42	Lese digitale Eingänge / Schreibe digitale Ausgänge	Binär
opcBIORO ¹¹⁸ (Verfügbar ab Softwarestand 1.4.x.x)	0x43	Status der digitalen Ein-/Ausgänge abfragen	Binär
opsRSW ¹²⁰	0x44	Lese Statuswort der dynamischen Messung	Binär
opcRDC ¹²³ (Verfügbar ab Softwarestand 1.8.x.x)	0x45	Lese Anzahl der dynamischen Messwerte	Binär
opcDDM1 ¹²⁵	0x50	Definiere dynamische Messung 1	String ⁶⁶
opcDDM2 ¹²⁵	0x51	Definiere dynamische Messung 2	String ⁶⁶
opcRDM1 ¹²⁷	0x60	Lese Messwerte der dynamischen Messung 1	Binär

Opcode	Hex-Wert	Funktion	Parameter-Typ
opcRDM2 ¹²⁷	0x61	Lese Messwerte der dynamischen Messung 2	Binär

Opcodes "Service"

Opcode	Hex-Wert	Funktion	Parameter-Typ
opcRST ¹²⁹	0x7E	Löse System- Reset aus	String ⁶⁶

5.3 Parameter-Typ "String"

Für Opcodes mit dem Parameter-Typ „String“ gelten folgende Regeln:

- a) Es dürfen nur ASCII-Zeichen im Bereich von 0x20 .. 0x7F gesendet werden. Die erweiterten ANSI-Zeichen oder „Wide Strings (16 Bit)“ werden nicht unterstützt.
- b) Jeder String wird von zwei '#' Zeichen eingerahmt. Diese Zeichen dienen zur Synchronisation und enthalten keine Informationen.
- c) Sofort nach dem ersten '#' – Zeichen folgt der Befehlscode. Dies ist ein variabler Teilstring, der dem vordefinierten Befehl exakt entsprechen muss. Groß- und Kleinschreibung ist wichtig.
- d) Parameter innerhalb des Strings werden durch Semikolon ; getrennt.

5.4 Opcodes: Initialisierung

5.4.1 opcRIV: Lese Box-Übersicht

Über diesen Opcode kann die Anzahl der im System vorhandenen Irinos-Boxen ermittelt werden.

Übersicht

Opcode: 0x01
Name: opcRIV
Funktion: Lese Box-Übersicht (Read inventory)
Parameter-Typ: [String](#)⁶⁶

DLL-Funktion

Wird mit [MSC_WriteCommand](#)³⁰ übertragen.

Anfragestring zum Irinos-System

Keiner, d.h. es müssen keine Daten zum Irinos-System gesendet werden.

Antwortstring vom Irinos-System

```
#{Anzahl Irinos-Boxen};{Anzahl Module}#
```

Anzahl Irinos-Boxen

Gesamtanzahl der im System vorhandenen Irinos-Boxen.

Anzahl Module

Dieser Wert ist immer identisch mit „Anzahl Irinos-Boxen“. Er ist nur aus Kompatibilitätsgründen zu älteren Systemen vorhanden.

Beispiel:

#3;3# -> Es wurden 3 Irinos-Boxen gefunden.

Dieser Opcode kann keine Fehlerantwort liefern.

Kommentar

Die Master-Box (mit Ethernet-Schnittstelle zum PC) wird ebenfalls als Box gezählt. Das obige Beispiel könnte z.B. durch folgende Boxen-Kombination entstanden sein:

- 1 Master-Box IR-MASTER-KB1-68-68-SYSP-ETHIL
- 1 Slave-Box für induktive Messtaster IR-TFV-8-TESA-M16-IL
- 1 Slave-Box für Inkrementalgeber IR-INC-4-SEL1VSS-D15F-IL

Kompatibilität zu früheren Systemen

Dieser Opcode ist uneingeschränkt kompatibel zu früheren Systemen.

5.4.2 opcRMI: Lese Box-Informationen (digitales Typenschild)

Über diesen Opcode können Informationen über eine Irinos-Box abgerufen werden.

Übersicht

Opcode:	0x03
Name:	opcRMI

Funktion: Lese Box-Informationen / digitales Typenschild
(Read module information)

Parameter-Typ: [String](#)^{D66}

DLL-Funktion

Wird mit [MSC_WriteCommand](#)^{D30} übertragen.

Anfragestring zum Irinos-System

`#{Box-Nummer};{Wert 2}#`

z.B. #0;2#

Box-Nummer

Nummer der Irinos-Box beginnend bei 0.

0 ist immer die Master-Box.

≥ 1 ist eine Slave-Box

Wenn die Box-Informationen aller Boxen abgefragt werden sollen, dann muss die Abfrage mit diesem Opcode für jede Irinos-Box einmal ausgeführt werden.

Wert 2

Hier muss immer der Wert 2 stehen. Fehlt dieser Teil des Parameterstrings, wird zur Kompatibilität mit älteren Systemen ein Antwortstring in einem veralteten Format zurückgesandt.

Antwortstring vom Irinos-System

```
#{Box-Nummer};{Geräte-Bezeichnung};{MAC-Adresse};{Serien-Nr.};
{Fertigungs-Code};{Hardware-Version};{Hardware-Revision};
{Firmware-Version};{Sample-Periode};{Gesamtanzahl Messkanäle};
{Anzahl Messkanäle 64Bit};{Anzahl Messkanäle 32Bit};{Anzahl
Messkanäle 16Bit};{Anzahl Messkanäle 8Bit};{immer 0};{immer 0};
{immer 0};{immer 0};{immer 0};{Anzahl digitale Eingänge};
{Anzahl digitale Ausgänge};{GUID};{Anwender-Bezeichnung};
{Bestell-Nummer}#
```

Beispiel:

```
#0;0;IR-TFV-8-IET-M16-ETHIL;A0-BB-3E-E0-00-03;I123456;S-W3-
28;HW V1.1;HWRev 1;SW V1.0.0.27;50;8;0;0;8;0;0;0;0;0;0;2;0;
{0C003B23-2C74-49A0-BCB1-E81C7C32C42A};LBox 0;828-5006#
```

Box-Nummer (1)

z.B.: 0

Nummer der Box, von welcher die Box-Informationen abgefragt wurden. Stimmt mit dem Parameter Box-Nummer des Anfrage-Strings überein.

Geräte-Bezeichnung (2)

z.B. IR-TFV-8-IET-M16-ETHIL

Typ-Bezeichnung der Irinos-Box, wie sie auch auf dem Typenschild steht.

MAC-Adresse (3)

z.B. A0-BB-3E-E0-00-03

MAC-Adresse der Irinos-Box.

Jede Irinos-Box hat eine eindeutige MAC-Adresse. Diese wird bei der Produktion vergeben und steht auch auf dem Typenschild. Über die MAC-Adresse kann beispielsweise identifiziert werden, ob eine Irinos-Box gegen ein baugleiches Modell getauscht wurde.

Die MAC-Adresse ist sowohl bei Master- als auch bei Slave-Boxen vorhanden. Bei Master-Boxen wird sie auch als MAC-Adresse für die Ethernet-Schnittstelle verwendet.

Serien-Nr (4)

z.B. I123456

Von der Produktion vergebene Seriennummer der Irinos-Box. Sie steht auch auf dem Typenschild.

Wenn eine Irinos-Box zur Reparatur eingesandt wird, dann kann sich im Falle eines Hardware-Tauschs die MAC-Adresse ändern, die Seriennummer jedoch nicht.

Fertigungs-Code (5)

z.B. S-W3-28

Über den Fertigungscode können durch den Hersteller weitere Fertigungs-Informationen ermittelt werden.

Hardware-Version (6)

z.B. HW V1.1

Versionsstand der Elektronik.

Hardware-Revision (7)

z.B. HWRev 1

Kompatibilitäts-Code der Hardware zur Firmware. Dadurch wird sichergestellt, dass bei einem Firmware-Update nur die zulässige Firmware-Version auf die Irinos-Box geladen wird.

Firmware-Version (8)

z.B. SW V1.0.0.27

Versionsstand der Firmware.

Der erste Teil der Firmware-Version wird bei einer umfangreichen Änderung inkrementiert.

Der zweite Teil der Firmware-Version wird inkrementiert, wenn neue Funktionen hinzugekommen sind.

Der dritte Teil der Firmware-Version wird inkrementiert, wenn ein Fehler behoben wurde.

Der vierte Teil der Firmware-Version ist ein interner Zähler.

Sample-Periode (9)

z.B. 50

Abtast-Periode für die Messwert-Aufnahme in μs .

Die Abtast-Periode ist in jeder Irinos-Box immer konstant. Sie ist unabhängig von der Abtast-Periode der statischen oder dynamischen Messung.

Die Periode einer Zeit-gesteuerten dynamischen Messung muss immer ein Vielfaches der Abtast-Periode der beteiligten Irinos-Boxen sein. Mehr dazu siehe beim Opcode [opcDT](#)⁹¹.

Gesamtanzahl Messkanäle (10)

z.B. 8

Gesamtanzahl aller Messkanäle der Irinos-Box.

Anzahl Messkanäle 64 Bit (11)

immer 0

Platzhalter für zukünftige Verwendung. 64 Bit – Messkanäle werden momentan nicht unterstützt.

Anzahl Messkanäle 32 Bit (12)

z.B. 0

Anzahl aller Messkanäle der Irinos-Box mit einem internen Wertebereich von 32 Bit. Dies sind beispielsweise Inkrementalgeber-Messkanäle.

Anzahl Messkanäle 16 Bit (13)

z.B. 8

Anzahl aller Messkanäle der Irinos-Box mit einem internen Wertebereich von 16 Bit. Dies sind beispielsweise Messeingänge für induktive Messtaster oder Analog-Eingänge $\pm 10\text{V}$.

16 Bit-Messwerte werden immer als 32 Bit-Werte zur DLL übertragen. Für die Applikation gibt es nur 32 Bit-Werte.

Anzahl Messkanäle 8 Bit (14)

z.B. 0

Anzahl aller Messkanäle der Irinos-Box mit einem internen Wertebereich von 8 Bit.

8 Bit-Messwerte werden immer als 32 Bit-Werte zur DLL übertragen. Für die Applikation gibt es nur 32 Bit-Werte.

Parameter 15 - 19

Immer 0. Platzhalter für zukünftige Verwendung.

Anzahl digitale Eingänge (20)

z.B. 2

Anzahl der digitalen Eingänge der Irinos-Box.

Zur DLL hin werden die digitalen Eingänge immer zu 8 Bits zusammengefasst. Wenn es weniger Eingänge gibt, dann wird aufgerundet.

Im Beispiel gibt es 2 digitale Eingänge. Zur DLL hin werden 8 digitale Eingänge übertragen. Die Eingänge 3-8 ändern Ihren Zustand nie.

Anzahl digitale Ausgänge (21)

z.B. 0

Anzahl der digitalen Ausgänge der Irinos-Box.

Vergleichbar zu den digitalen Eingängen wird immer auf 8 digitale Ausgänge aufgerundet.

D.h. hat eine Irinos-Box beispielsweise 4 digitale Ausgänge, so entstehen „virtuelle“ Ausgänge 5-8. Diese können zwar per DLL angesprochen werden, sie sind jedoch physikalisch nicht vorhanden.

GUID (22)

z.B. 0C003B23-2C74-49A0-BCB1-E81C7C32C42A

Platzhalter für zukünftige Verwendung.

Anwender-Bezeichnung (23)

z.B. LBox 0

Platzhalter für zukünftige Verwendung

Bestell-Nummer (24)

z.B. 828-5006

Bestellnummer der Irinos-Box.

Antwortstring vom Irinos-System im Fehlerfall

#-99# wenn der prinzipielle Aufbau des Anfragestrings fehlerhaft ist.

#-1# wenn die Box-Nummer des Anfragestrings ungültig ist.

Kompatibilität zu früheren Systemen

Zur Kompatibilität mit früheren Systemen wird der Anfragestring `#{Box-Nummer}#` unterstützt.

Der Antwort-String hat dann das Format:

```
#{Box-Typ};{Box-Nummer};{Anzahl Messkanäle};{Firmware-Version 1};{Firmware-Version 2};{Interne Nummer};{Textinfo}#
```

Die Verwendung bei Neu-Applikation ist nicht empfohlen.

5.4.3 opcRSS: Lese System-String

Über diesen Opcode kann ein String abgerufen werden, der die im System vorhandenen Irinos-Boxen sowie deren Reihenfolge widerspiegelt.

Übersicht

Opcode:	0x05
Name:	opcRSS
Funktion:	Lese System-String (Read system string)

Parameter-Typ: [String](#)¹⁶⁶

DLL-Funktion

Wird mit [MSC_WriteCommand](#)³⁰ übertragen.

Anfragestring zum Irinos-System

#1#

Antwortstring vom Irinos-System

```
{Wert 1};{Anzahl Irinos-Boxen};{Bestell-Nr. Irinos-Box 0};  
{Bestell-Nr. Irinos-Box 2};...;{Bestell-Nr. Irinos-Box n}#
```

Beispiel für ein System mit den beiden Irinos Boxen IR-INC (Box 0, Bestell-Nr. 828-5013) und IR-TFV (Box 1, Bestell-Nr. 828-5003):
#1;2;828-5013;828-5003#

Wert 1

Hier kommt immer der Wert 1.

Anzahl Irinos-Boxen

z.B. 2

Anzahl der im System vorhandenen Irinos-Boxen

Bestell-Nr. Irinos-Box n

z.B. 828-5013

Für alle angeschlossenen Irinos-Boxen werden in Reihenfolge ihrer Nummerierung die Bestellnummern aufgeführt.

Antwortstring vom Irinos-System im Fehlerfall

#-99# wenn der prinzipielle Aufbau des Anfragestrings fehlerhaft ist.

#-1# wenn der Anfragestring nicht den Wert 1 enthält.

Kommentar

Es wird empfohlen den erwarteten String in der Messrechner-Software zu hinterlegen (z.B. in deren Ini-File) und nach jedem Verbindungsaufbau mit dem vom Irinos-System gelieferten String zu vergleichen. Damit kann geprüft werden, ob alle Irinos-Boxen vorhanden sind und ob diese korrekt angeschlossen sind.

Kompatibilität zu früheren Systemen

Dieser Opcode ist bei älteren Systemen nicht verfügbar. Deshalb ist keine Rückwärtskompatibilität erforderlich.

5.4.4 opcWCC: Schreibe Kanal-Eigenschaften

Über diesen Opcode kann ein Messkanal parametrierbar werden. Die Parametrierung hängt vom jeweiligen Messkanal-Typ ab. Derzeit wird dieser Opcode nur für Inkrementalgeber-Messkanäle IR-INC verwendet.

Übersicht

Opcode: 0x09

Name: opcWCC

Funktion: Schreibe Kanal-Eigenschaften (Write channel characteristics)

Parameter-Typ: [String](#)⁶⁶

DLL-Funktion

Wird mit [MSC_WriteCommand](#)^{D30} übertragen.

Anfragestring zum Irinos-System

```
#{KanalName};{Konfigurationstyp};{Permanent speichern}#
```

z.B. #T10;TTL;1#

KanalName

Name des Messkanals.

Nach dem Einschalten werden den Messkanälen in aufsteigender Reihenfolge die Namen T1 T2 T3 T4 ... Tn vergeben.

Der Namen eines Messkanals kann aber durch die Applikation über den Opcode [opcWCA](#)^{D82} geändert werden. Wenn dies gemacht wurde, dann muss hier auch der vergebene Kanalname angegeben werden.

Konfigurationstyp

„1VSS“ um den Inkrementalgeber-Kanal als 1 Vss – Kanal zu parametrieren.

„TTL“ um den Inkrementalgeber-Kanal als TTL/RS422 – Kanal mit 1x-Auswertung zu parametrieren (1 Inkrement = 1 Digit).

„TTL4X“ um den Inkrementalgeber-Kanal als TTL/RS422 – Kanal mit 4x-Auswertung zu parametrieren (1 Inkrement = 4 Digits). *Verfügbar ab IR-INC Firmware-Version 1.5.0.24.*

Permanent speichern

„0“: Die Änderung bleibt bis zum nächsten Einschalten/Reset aktiv. Anschließend wird wieder die alte Konfiguration verwendet.

„1“: Die Änderung wird permanent gespeichert und ist auch nach einem Aus-/Einschalten bzw. Reset noch gültig.

Bitte beachten Sie: Die Anzahl der Schreibvorgänge in den permanenten Speicher ist begrenzt. Weitere Angaben dazu entnehmen Sie dem Benutzerhandbuch.

Wenn ein Messkanal bereits so parametrier ist, wie im Konfigurationstyp angegeben, dann wird kein Speichervorgang ausgelöst. D.h. es ist möglich den Konfigurationsstring mit „Permanent speichern = 1“ nach jedem Einschalten zu senden.

Antwortstring vom Irinos-System bei einem Inkrementalgeber-Messkanal (IR-INC)

#0# Erfolg

#-n# Parameter ‚n‘ des Anfragestrings falsch

#-99# Führende oder abschließende Raute fehlt, oder Gesamtgröße falsch

Antwortstring vom Irinos-System bei einem Messkanal. der diesen Opcode nicht unterstützt

#-98#

Kommentar

Ein Inkrementalgeber-Messkanal lässt sich auch über das Irinos-Tool dauerhaft als 1Vss- oder TTL/RS422-Kanal parametrieren.

Nach der Parametrierung wird die Inkrementalgeber-Position zurückgesetzt.

Wenn die Konfiguration über diesen Opcode nach jedem Verbindungsaufbau durch die Applikationssoftware geschrieben wird, dann ist es unerheblich, wie eine Irinos-Box vorparametriert ist. Dadurch kann ein Hardware-Tausch ohne definierte Vor-Parametrierung der „neuen“ Irinos-Box erfolgen.

Kompatibilität zu früheren Systemen

Dieser Opcode ist bei älteren Systemen nicht verfügbar. Deshalb ist keine Rückwärtskompatibilität erforderlich.

5.4.5 opcRCA: Lese Kanal-Zuordnung

Über diesen Opcode kann die aktuelle Zuordnung der Messkanäle zu den physikalischen Messeingängen ausgelesen werden.

Übersicht

Opcode: 0x10
Name: opcRCA
Funktion: Lese Kanal-Zuordnung (Read channel assignment)
Parameter-Typ: [String](#)⁶⁶

DLL-Funktion

Wird mit [MSC_WriteCommand](#)³⁰ übertragen.

Anfragestring zum Irinos-System

`#{Segmentindex}#`

z.B. #1#

Segmentindex

Siehe Beschreibung der Segmente

Antwortstring vom Irinos-System

Liste Siehe Beispiel

#-1# Segment-Index ungültig

#-99# Führende oder abschließende Raute fehlt, oder Gesamtgröße falsch

Beispiel-Antwort (Liste)

```
#1;1;  
T1,1,0,1,1;  
T2,2,0,1,2;  
T3,3,0,1,3;  
T4,4,0,1,4;  
T5,5,1,1,1;  
T6,6,1,1,2;  
T7,7,1,1,3;  
T8,8,1,1,4;  
T9,9,1,1,5;  
T10,10,1,1,6;  
T11,11,1,1,7;  
T12,12,1,1,8  
#
```

Der erste Block ist #{Segment};{AnzahlSegmente}#

z.B. 1;2

Die weiteren Blöcke definieren jeweils einen logischen Messkanal. Die Werte haben folgende Bedeutung:

T1,1[a] Kanalbezeichnung (ASCII-String, max. 4 Zeichen)

,0,1,

1

T1, [b] Logische Kanalnummer in aufsteigender Reihenfolge

1,0,

1,1

T1,1 [c] Nummer der Irinos-Box (Box-ID)

,

0,1,

1

T1,1 [d] Modul-ID. Immer 1. Zur Kompatibilität mit früheren

,0, Systemen.

1,1

T1,1 [e] Physikalische Kanalnummer innerhalb der Irinos-Box mit der

,0,1, unter [c] aufgeführten Box-ID. Beginnt bei jeder Irinos-Box bei

1 1.

Segmente

Bei Irinos-Systemen mit sehr vielen Messkanälen kann die Liste der Kanaluordnung nicht mit einem einzigen Datentransfer zum PC übertragen werden. Deshalb wird die Liste in Segmente aufgeteilt.

Wenn das erste Segment per Segment-Index #1# abgerufen wird, dann beginnt die Antwort zunächst mit demselben Segment-Index, gefolgt von der Anzahl der Segmente (z.B. #1;2;). Dieser zweite Wert sagt aus, in wie vielen Teilen (Segmenten) die Kanaluordnungs-Liste ausgelesen werden muss. Gibt es beispielsweise 2 Segmente, dann muss der erste Teil über den Segment-Index #1# und der zweite Teil über den Segment-Index #2# ausgelesen werden.

Die Segment-Größe ist per Definition auf 32 Messkanäle festgelegt.

Kommentar

Beim Start (Power On) des Irinos-Systems wird die Kanal-Zuordnung automatisch erzeugt. Die Zuordnung ist dynamisch, d.h. sie passt sich automatisch den jeweils gefundenen Irinos-Boxen an.

Mit dem Opcode [opcWCA](#)^{□82} kann eine feststehende Kanalreihenfolge erzwungen werden.

Die Kanal-Zuordnung entspricht automatisch der Kanal-Liste 0, die über den Opcode [opcRCL](#)^{□87} abgefragt werden kann. Sie kann auch als Kanalliste für die dynamische Messung verwendet werden. Dies wird aber nicht empfohlen. Verwenden Sie hierfür eine separate Kanal-Liste.

Kompatibilität zu früheren Systemen

Dieser Opcode ist uneingeschränkt kompatibel zu früheren Systemen.

Die Segment-Größe war bei früheren Systemen in der Regel 20 Messkanäle. Nun beträgt sie 32 Messkanäle.

5.4.6 opcWCA: Schreibe Kanal-Zuordnung

Über diesen Opcode kann die aktuelle Zuordnung der Messkanäle zu den physikalischen Messeingängen geändert werden.

Dieser Opcode ist in erster Linie zur Kompatibilität mit früheren Systemen verfügbar. Bei Neu-Applikationen wird er nicht benötigt.

Übersicht

Opcode:	0x11
Name:	opcWCA
Funktion:	Schreibe Kanal-Zuordnung (Write channel assignment)
Parameter-Typ:	String ^{□66}

DLL-Funktion

Wird mit [MSC_WriteCommand](#)^[30] übertragen.

Anfragestring zum Irinos-System

Beispiel:

```
#T1,1,0,1,1;T2,2,0,1,2;T3,3,0,1,3#
```

Zuordnungsliste (1. Teil des Beispiels):

T1, [a] Kanalbezeichnung (ASCII-String, max. 4 Zeichen)

```
1,0,  
1,1
```

T1, [b] Logische Kanalnummer in aufsteigender Reihenfolge

```
1,0,  
1,1
```

T1,1[c] Nummer der Irinos-Box (Box-ID)

```
,  
0,1,  
1
```

T1,1[d] Modul-ID. Immer 1. Zur Kompatibilität mit früheren

```
,0, Systemen.  
1,1
```

T1,1[e] Physikalische Kanalnummer innerhalb der Irinos-Box mit der

```
,0,1 unter [c] aufgeführten Box-ID. Beginnt bei jeder Irinos-Box bei  
,1 1.
```

Antwortstring vom Irinos-System

#0# Erfolg

#-1# Kanalbezeichnung (z.B. T1) ist größer 4 Zeichen

#-2# Logischer Kanal ungültig

- #-3# Nummer der Irinos-Box (Box-ID) ungültig
- #-4# Modul-ID ungültig
- #-5# Physikalische Kanalnummer der Irinos-Box ungültig
- #-6# Zu wenig Parameter
- #-7# Kein Semikolon
- #-99# Führende oder anschließende Raute fehlt oder Gesamtgröße falsch

Kommentar

Beim Start (Power On) des Irinos-Systems wird die Kanal-Zuordnung automatisch erzeugt. Die Zuordnung ist dynamisch, d.h. sie passt sich automatisch den jeweils gefundenen Irinos-Boxen an.

Über diesen Opcode kann die automatisch erzeugte Kanal-Zuordnung überschrieben werden.

Es kann mit einem Datenpaket nur ein String begrenzter Länge vom PC zum Irinos-System übertragen werden. Deshalb muss bei Systemen mit vielen Messkanälen (> 32) das Schreiben der Kanal-Zuordnung aufgeteilt werden. Wir empfehlen immer max. 32 logische Messkanäle auf einmal zu schreiben.

Beispiel: Es sollen 42 logische Messkanäle geschrieben werden. Schreiben Sie zuerst die logischen Messkanäle 1 – 32 und anschließend die Messkanäle 33 – 42.

Wichtig ist, dass die logische Kanalnummer in aufsteigender Reihenfolge geschrieben wird.

Kompatibilität zu früheren Systemen

Dieser Opcode ist uneingeschränkt kompatibel zu früheren Systemen.

5.5 Opcodes: Konfiguration & Verschiedene

5.5.1 opcWCL: Schreibe Kanal-Liste

Dieser Opcode schreibt eine Kanal-Liste zum Irinos-System.

Eine Kanal-Liste ist für eine dynamische Messung zwingend erforderlich. Bei der statischen Messung kann eine Kanal-Liste zur Beschränkung der Kanal-Zahl genutzt werden.

Übersicht

Opcode: 0x22

Name: opcWCL (früher opcCL)

Funktion: Schreibe Kanal-Liste (Write channel list)

Parameter-Typ: [String](#)⁶⁶

DLL-Funktion

Wird mit [MSC_WriteCommand](#)³⁰ übertragen.

Anfragestring zum Irinos-System

```
{Nummer Kanalliste};{Kanalbezeichnung 1};...;{Kanalbezeichnung n}#
```

Beispiel:

```
#2;T1;T2;T5;T18#
```

Nummer Kanalliste

Nummer der Kanalliste, die geschrieben werden soll. Zulässige Werte: 1..10

Kanalbezeichnung

Bezeichnung des Messkanals aus der Kanalzuordnung.

Nach dem Einschalten (Power On) werden alle Messkanäle in aufsteigender Reihenfolge benannt: T1, T2, T3, usw.

Die Kanalnamen können über die Kanalzuordnung ([opcWCA⁸²](#)) geändert werden.

Antwortstring vom Irinos-System

#0# Erfolg

#-n# Parameter Nr n ungültig

#-99# Führende oder anschließende Raute fehlt oder Gesamtgröße falsch

Kommentar

Kanallisten beinhalten alle oder einen Teil der verfügbaren Messkanäle.

Sie werden bei der dynamischen Messung benutzt um die Messkanäle zu definieren, die bei einer nachfolgenden dynamischen Messung aufgenommen werden sollen (siehe [opcDDMx¹²⁵](#)).

Wird eine Kanalliste während einer aktiven dynamischen Messung geändert, so hat dies keine Auswirkung auf diese dynamische Messung. D.h. es wird die Kanalliste verwendet, die beim Start der dynamischen Messung gültig war. Erst bei einem erneuten Start wird die geänderte Kanalliste übernommen.

Kanallisten werden bei der statischen Messung benutzt, um die Anzahl der übertragenen Messkanäle einzuschränken (siehe [opcACL⁸⁹](#)). Bei Systemen mit wenigen Messkanälen (ca. ≤ 64) macht dies keinen Sinn. Bei mehr Messkanälen kann damit die Übertragungsbandbreite zum PC zugunsten der dynamischen Messung entlastet werden.

Wird eine Kanalliste während einer aktiven statischen Messung geändert, so wird die neue Kanalliste unmittelbar angewandt.

Die Messwerte werden in der Messkanal-Reihenfolge der Kanal-Liste übertragen.

Die Kanalliste 0 entspricht der Kanal-Zuordnung (siehe [opcRCA](#)^{□79}) und beinhaltet alle verfügbaren Messkanäle. Sie kann nur über den Opcode `opcWCA` geändert werden.

Die Kanallisten 1-10 können jederzeit neu definiert und überschrieben werden.

Kompatibilität zu früheren Systemen

Dieser Opcode ist uneingeschränkt kompatibel zu früheren Systemen.

5.5.2 **opcRCL: Lese Kanal-Liste**

Über diesen Opcode kann eine Kanal-Liste vom Irinos-System ausgelesen werden.

Übersicht

Opcode: 0x23
Name: `opcRCL`
Funktion: Lese Kanal-Liste (Read channel list)
Parameter-Typ: [String](#)^{□66}

DLL-Funktion

Wird mit [MSC_WriteCommand](#)^{□30} übertragen.

Anfragestring zum Irinos-System

`#{Nummer Kanalliste}#`

Beispiel: `#3#`

Nummer Kanalliste

Nummer der Kanalliste, die ausgelesen werden soll. Zulässige Werte: 0..10

Antwortstring vom Irinos-System

Liste Siehe Beschreibung der Kanalliste

`#-n#` Parameter Nr n ungültig

`#-99#` Führende oder anschließende Raute fehlt oder Gesamtgröße falsch

Beschreibung der Kanalliste

`#{Nummer Kanalliste};{Kanalbezeichnung 1};...;{Kanalbezeichnung n}#`

Beispiel: `#2;T1;T2;T5;T18#`

Nummer Kanalliste

Nummer der Kanalliste, die ausgelesen wurde.

Kanalbezeichnung

Bezeichnung des Messkanals.

Kommentar

Nach dem Einschalten (Power On) werden alle Kanal-Listen aus der Kanal-Zuordnung automatisch erzeugt. Sie beinhalten dann alle Messkanäle.

Vor dem Start einer dynamischen Messung muss die zugehörige Kanal-Liste mit den für die dynamische Messung benötigten Messkanälen geschrieben werden (siehe [opcWCL](#)^{□85}).

Kanallisten beinhalten alle oder einen Teil der verfügbaren Messkanäle.

Die Kanalliste 0 entspricht der Kanal-Zuordnung (siehe [opcRCA](#)^{□79}) und beinhaltet alle verfügbaren Messkanäle. Sie kann nur über den Opcode [opcWCA](#)^{□82} geändert werden.

Kompatibilität zu früheren Systemen

Dieser Opcode ist bei älteren Systemen nicht verfügbar. Deshalb ist keine Rückwärtskompatibilität erforderlich.

5.5.3 opcACL: Aktiviere Kanal-Liste für die statische Messung

Über diesen Opcode kann eine Kanal-Liste für die statische Messung angewandt werden.

Übersicht

Opcode: 0x24 (0x26; siehe Kommentar)

Name: opcACL

Funktion: Aktiviere Kanal-Liste für statische Messung
 (Activate channel list)

Parameter-Typ: [String](#)^{□66}

DLL-Funktion

Wird mit [MSC_WriteCommand](#)^{□30} übertragen.

Anfragestring zum Irinos-System

`#{Nummer Kanalliste}#`

Beispiel: `#2#`

Nummer Kanalliste

Nummer der Kanalliste, die für die statische Messung aktiviert werden soll. Zulässige Werte: 0..10

Antwortstring vom Irinos-System

`#0#` Erfolg

`#-1#` Nummer der Kanalliste ungültig

`#-99#` Führende oder anschließende Raute fehlt oder Gesamtgröße falsch

Kommentar

Dieser Opcode wird nur in Anwendungen mit vielen Messkanälen benötigt, bei denen parallel eine dynamische Messung mit hoher Datenrate und/oder vielen Messkanälen verwendet wird.

Im Regelfall wird für statische Messungen die voreingestellte Kanal-Liste 0 verwendet. Die Kanal-Liste 0 enthält alle Messkanäle.

Mit diesem Opcode ist es möglich, eine eingeschränkte Kanalliste zu verwenden. Dies kann z.B. bei eine Anwendung mit sehr vielen Messtastern (ca. > 64) sinnvoll sein, von denen immer nur ein kleiner Teil gleichzeitig verwendet wird. In der Kanalliste können dann nur die aktuell benutzten Kanäle definiert sein. Somit werden nur diese in den Datentransfer zum PC aufgenommen. Die Übertragungs-Bandbreite wird damit zugunsten der Übertragung der dynamischen Messwerte entlastet.

Die zur dynamischen Messung verwendete Kanalliste wird bei der Definition der dynamischen Messung übergeben. Der Opcode `opcACL` hat auf sie keinen Einfluss.

Kompatibilität zu früheren Systemen

Dieser Opcode ist uneingeschränkt kompatibel zu früheren Systemen.

In der Vergangenheit wurde für diesen Opcode manchmal der Wert 0x24 und manchmal der Wert 0x26 verwendet. Aus Kompatibilitätsgründen können deshalb beide verwendet werden. Es wird jedoch empfohlen den Wert 0x24 zu verwenden.

5.5.4 **opcDT: Definiere Trigger für die dynamische Messung**

Über diesen Opcode kann ein Trigger für die dynamische Messung konfiguriert werden.

Übersicht

Opcode: 0x30

Name: opcDT

Funktion: Definiere Trigger für die dynamische Messung
(Define Trigger)

Parameter-Typ: [String](#)^{D66}

DLL-Funktion

Wird mit [MSC_WriteCommand](#)^{D30} übertragen.

Anfragestring zum Irinos-System

```
#{Trigger-Nummer};{Typ};{Quelle};{Skalierung};{Abstand};
{Start};{Ende}#
```

Trigger-Nummer

Nummer des Triggers. Es können zwei unabhängige Trigger definiert werden. Zulässige Werte sind 1 .. 2.

Typ

T für Zeit-Trigger

P für Positions-Trigger

Quelle

Positions-gesteuerte Messung: Bezeichnung des Trigger-Kanals (z.B. T7)

Zeit-gesteuerte Messung: Unbenutzt. Verwenden Sie das Zeichen * als Quelle.

Skalierung

Positions-gesteuerte Messung: Gleitkommawert zur Umrechnung des Messwertes der Trigger-Quelle in eine physikalische Einheit, z.B. μm oder $^\circ$. Dadurch können die folgenden Parameter Abstand, Start und Ende in physikalischen Einheiten angegeben werden.

Verwenden Sie den Skalierungs-Wert 1 um die folgenden Parameter in der Messwert-Einheit anzugeben (d.h. Inkremente oder Digit).

Der Skalierungsfaktor kann auch -1 sein, um das Vorzeichen zu drehen.

Zeit-gesteuerte Messung: Unbenutzt. Verwenden Sie immer den Wert 1.

Abstand

Positions-gesteuerte Messung: Abstand zwischen 2 Trigger-Impulsen (Gleitkommawert).

Zeit-gesteuerte Messung: Zeitlicher Abstand zwischen 2 Messwert-Aufnahmen in ms, z.B. 0.1 für $100\mu\text{s}$ oder 1 für 1ms. Der zeitliche Abstand muss ein Vielfaches der Abtastzeit der beteiligten Irinos-Boxen sein, mindestens jedoch 0.1 (= $100\mu\text{s}$).

Bei den meisten Irinos-Boxen ist die interne Abtastzeit $50\mu\text{s}$. Zulässige Werte wären also beispielsweise 0.1, 0.25, 0.85, 1, 1.5, 12, usw.

Hätte einer der beteiligten Irinos-Boxen „nur“ eine interne Abtastzeit von $100\mu\text{s}$, dann wären zulässige Werte beispielsweise 0.1, 0.2, 0.3, 0.8, 0.9, 1, 1.5, 12, usw.

Start

Positions-gesteuerte Messung: Die Messung beginnt, nachdem der angegebene Wert überschritten wurde (Gleitkommawert).

Zeit-gesteuerte Messung: Verzögerung bis zum Start der dynamischen Messung in ms (Gleitkommawert).

Ende

Positions-gesteuerte Messung: Die Messung wird beendet, wenn der angegebene Wert überschritten wurde (Gleitkommawert). Dies gilt auch dann, wenn die maximale Anzahl an dynamischen Messwerten noch nicht erreicht wurde. Alternativ kann auch das Zeichen * als Parameter verwendet werden. Die dynamische Messung läuft dann solange, bis die maximale Anzahl an aufzuzeichnenden dynamischen Messwerten erreicht wurde.

Zeit-gesteuerte Messung: Dauer der dynamischen Messung in ms ab deren Start. Es wird empfohlen ein zeitliches Ende bei der zeit-gesteuerten Messung nicht zu verwenden, indem das Zeichen * als Parameter verwendet wird. Verwenden Sie stattdessen die maximale Anzahl an dynamischen Messwerten bei der Definition der dynamischen Messung, um deren Dauer zu begrenzen.

Antwortstring vom Irinos-System

#0# Erfolg

#-n# Parameter Nr n ungültig

#-99# Führende oder anschließende Raute fehlt oder Gesamtgröße falsch

Beispiele Parameterstring für Positions-gesteuerte Messung

#1;P;T2;20.0;0.1;50.0;*#

Dieser String definiert den Trigger Nr. 1 als positionsgesteuert. Die Position kommt vom Eingang T2 (zum Beispiel der Inkrementalgeber 2).

Der binäre Eingangswert (Inkremente) muss durch 20.0 geteilt werden, um den Positionswert in mm zu erhalten.

Der Trigger sollte zehn Mal pro mm ausgelöst werden (alle 0.1mm) und der erste Trigger-Impuls sollte bei der Position 50.0mm erzeugt werden.

Es gibt keinen definierten Endpunkt. Stattdessen markiert ein * den Ende-Parameter als ungenutzt.

```
#2;P;T17;-1.0;10.0;0.0;3600.0#
```

Der String definiert den Trigger Nr. 2 als positionsgesteuert. Die Position kommt vom Eingang T17 (zum Beispiel ein inkrementeller Drehgeber mit 3600 Inkremente / Umdrehung).

Der binäre Eingangswert von T17 wird nicht geteilt. Die Messung soll aber in die negative Drehrichtung erfolgen, deshalb wird über den Skalierungsfaktor das Vorzeichen umgekehrt. D.h. Skalierungsfaktor -1.0.

Der Trigger sollte alle 10.0 Inkremente ausgelöst werden, d.h. in 1 ° - Schritten. Der erste Impuls soll bei 0 Inkrementen (= 0°) ausgelöst werden. Nach 3600 Inkrementen (= 360°) soll die zugehörige dynamische Messung beendet werden.

Beispiele für Zeit-gesteuerte Messung

```
#2;T;*;1.0;1.0;0.0;*#
```

Dieser String definiert den Trigger Nr. 2 als zeitgesteuert. Die Periodendauer ist 1.0ms. Dies entspricht 1 kSample/s.

Die dynamische Messung soll sofort gestartet werden (0.0ms).

Die dynamische Messung läuft solange, bis die in der dynamischen Messung definierte Anzahl an Messwerten aufgezeichnet wurde oder, bis die dynamische Messung manuell über [opcIT⁹⁷](#) oder [opcDDMx¹²⁵](#) beendet wird.

```
#1;T;*;1.0;0.2;500.0;*#
```

Dieser String definiert den Trigger Nr. 1 als zeitgesteuert. Die Periodendauer ist 0.2ms. Dies entspricht 5 kSample/s.

Die dynamische Messung soll mit einer Verzögerung von 500.0ms gestartet werden.

Die dynamische Messung läuft solange, bis die in der dynamischen Messung definierte Anzahl an Messwerten aufgezeichnet wurde oder, bis die dynamische Messung manuell über [opcIT⁹⁷](#) oder [opcDDMx¹²⁵](#) beendet wird.

Kommentar

Bei einem Positions-Trigger können die Werte für Skalierung, Abstand, Start und Ende jeweils negativ sein. Bei einem Zeit-Trigger können diese Werte nur positiv sein.

Es ist möglich, wenn auch häufig nicht notwendig, einen Endpunkt für den Trigger zu bestimmen. Die Anzahl der aufzunehmenden Messwerte wird über den Opcode [opcDDMx¹²⁵](#) bestimmt. Wird ein Endpunkt definiert, deaktiviert sich die dynamische Messung selbst, sobald dieser Punkt erreicht wurde. Dies kann zu Problemen führen, wenn die dynamische Messung mit einer festen Anzahl von Messwerten definiert wurde und noch nicht alle Werte aufgenommen wurden, wenn der Trigger stoppt. Die dynamische Messung wird in diesem Fall niemals die vollständige Anzahl von Messwerten aufnehmen. Man sollte entweder ein Trigger-Ende oder eine feste Anzahl von Messwerten definieren, aber nicht beides.

Die Trigger-Konfiguration wird immer mit dem Start der dynamischen Messung übernommen. Wird während einer aktiven dynamischen Messung die Trigger-Konfiguration geändert, so hat dies keine Auswirkung auf die laufende dynamische Messung. Dies gilt erst ab dem Start der nächsten dynamischen Messung.

Üblicherweise wird für jede dynamische Messung ein separater Trigger verwendet. Derselbe Trigger kann jedoch auch für beide dynamische Messungen verwendet werden.

Kompatibilität zu früheren Systemen

Dieser Opcode ist grundsätzlich kompatibel zu früheren Systemen. Da die System-interne Realisierung der dynamischen Messung grundlegend geändert wurde, kann in Einzelfällen eine anderweitige Parametrierung erforderlich sein.

Es gelten darüber hinaus folgende Einschränkungen:

- Über den Buchstaben \times bei der Trigger-Quelle konnten beide Trigger-Kanäle bisher miteinander verknüpft werden. Diese Möglichkeit besteht nicht mehr.
- Der Trigger-Typ \mathbb{E} (extern) wird nicht mehr unterstützt. Dieser war bei früheren Systemen nur mit einer Sonder-Hardware nutzbar.

5.5.5 **opcAT: Aktiviere Trigger für die dynamische Messung**

Über diesen Opcode kann ein Trigger für die dynamische Messung aktiviert werden. Ist eine dynamische Messung konfiguriert, welcher dieser Trigger zugeordnet ist, so wird auch die dynamische Messung gestartet.

Übersicht

Opcode:	0x31
Name:	opcAT
Funktion:	Aktiviere Trigger für dynamische Messung (Activate Trigger)
Parameter-Typ:	String ⁶⁶

DLL-Funktion

Wird mit [MSC_WriteCommand](#)³⁰ übertragen.

Anfragestring zum Irinos-System

#{Trigger-Nummer}#

Beispiel: #2#

Trigger-Nummer

Nummer des Triggers, der aktiviert werden soll (1 oder 2)

Antwortstring vom Irinos-System

#0# Erfolg

#-1# Falsche Trigger-Nummer

#-99# Führende oder anschließende Raute fehlt oder Gesamtgröße falsch

Kommentar

Wenn eine dynamische Messung ohne aktiven Trigger gestartet wird, werden keine Messwerte aufgezeichnet.

Für den Start einer dynamischen Messung muss entweder zuerst die dynamische Messung und dann der zugehörige Trigger aktiviert werden, oder umgekehrt.

Kompatibilität zu früheren Systemen

Dieser Opcode ist grundsätzlich kompatibel zu früheren Systemen. Beachten Sie jedoch, dass der Trigger nun gemeinsam mit der dynamischen Messung gestartet wird.

5.5.6 opclT: Deaktiviere Trigger der dynamischen Messung

Über diesen Opcode kann ein Trigger für die dynamische Messung deaktiviert werden. Ist er einer dynamischen Messung zugeordnet, die aktiv ist, dann wird auch die dynamische Messung beendet.

Übersicht

Opcode: 0x32

Name: opcIT

Funktion: Deaktiviere Trigger für dynamische Messung
(Inactivate Trigger)

Parameter-Typ: [String](#)⁶⁶

DLL-Funktion

Wird mit [MSC_WriteCommand](#)³⁰ übertragen.

Anfragestring zum Irinos-System

`#{Trigger-Nummer}#`

Beispiel: #2#

Trigger-Nummer

Nummer des Triggers, der deaktiviert werden soll (1 oder 2)

Antwortstring vom Irinos-System

#0# Erfolg

#-1# Falsche Trigger-Nummer

#-99# Führende oder anschließende Raute fehlt oder Gesamtgröße falsch

Kompatibilität zu früheren Systemen

Dieser Opcode ist grundsätzlich kompatibel zu früheren Systemen. Beachten Sie jedoch, dass mit dem Trigger nun gemeinsam alle dynamischen Messungen beendet werden, die diesen Trigger verwenden.

5.5.7 opcSP: Setze (Kanal-)Parameter

Über diesen Opcode können die Messparameter eines Messkanals parametrisiert werden. Die Parameter sind abhängig vom Messkanal-Typ. Derzeit wird dieser Opcode nur für Inkrementalgeber-Eingänge (1Vss/TTL) unterstützt.

Übersicht

Opcode: 0x35
Name: opcSP
Funktion: Setze (Kanal-) Parameter (Set Parameter)
Parameter-Typ: [String](#)^{D66}

DLL-Funktion

Wird mit [MSC_WriteCommand](#)^{D30} übertragen.

Anfragestring zum Irinos-System

#{KanalName};{Position};{Referenzmarke ein/aus}#

KanalName

Name des Messkanals.

Nach dem Einschalten werden den Messkanälen in aufsteigender Reihenfolge die Namen T1 T2 T3 T4 ... Tn vergeben.

Der Namen eines Messkanals kann aber durch die Applikation über den Opcode [opcWCA](#)⁸² geändert werden. Wenn dies gemacht wurde, dann muss hier auch der vergebene Kanalname angegeben werden.

Position

- Positionswert, den der Messkanal ab sofort haben soll. Damit kann die aktuelle Position vorgegeben werden.
- * um die Position des Messkanals unverändert zu lassen. Dies wird benötigt, wenn nur die Referenzmarke ein-/ausgeschaltet werden soll. Ab Firmware V1.5.0.24 kann damit auch das Referenzmarken-Flag zurückgesetzt werden.
- ~ um die Verstärkungs- und Offset-Regelung des Messkanals zurückzusetzen. Der Zählerwert wird auf 0 zurückgesetzt. Dies ist nur bei 1Vss-Messeingängen sinnvoll.
- \$ um dem Messeingang und den Inkrementalgeber komplett zurückzusetzen. Dazu wird der Inkrementalgeber kurzzeitig spannungsfrei geschaltet (ca. 10 ms). Die Position wird auf 0 gesetzt.

Referenzmarke ein/aus

Zulässige Parameterwerte:

- REFON um die Referenzmarkenverarbeitung einzuschalten
Hiermit wird gleichzeitig das Statusbit ‚Refmark‘ (siehe opcRHS) freigegeben.
- REFOFF um die Referenzmarkenverarbeitung auszuschalten

Antwortstring vom Irinos-System bei einem Inkrementalgeber-Messkanal (IR-INC)

#0# Erfolg

#-n# Parameter Nr n ungültig

#-99# Führende oder anschließende Raute fehlt oder Gesamtgröße falsch

Antwortstring vom Irinos-System bei einem Messkanal, der diesen Opcode nicht unterstützt

#-98#

Beispiele für Anfragestrings zu Inkrementalgeber-Messkanälen

#T5;-2000;REFOFF#

Die aktuelle Position des Messkanals T5 wird auf -2000 gesetzt. Die Referenzmarkenverarbeitung wird deaktiviert.

#T5;*;REFON#

Die Referenzmarkenverarbeitung des Messkanals T5 wird aktiviert. Die Position bleibt unverändert.

An Firmware-Version 1.5.0.24 wird auch das Referenzmarken-Flag zurückgesetzt.

#T13;~;REFOFF#

Die Verstärkungs- und Offsetregelung des Messkanals T5 wird zurückgesetzt. Die Referenzmarkenverarbeitung wird deaktiviert.

Kommentare für Inkrementalgeber-Messkanäle

Die Fehler-Flags sowie das Referenzmarken-Bit ‚Refmark‘ für den Messkanal werden zurückgesetzt (diese sind über [opcRHS](#)¹⁰² auslesbar).

Während der Ausführung dieses Opcodes mit dem Positions-Parameter \$ werden zwei der Messeingänge der betroffenen Irinos-Box bis zu 500ms deaktiviert. Es wird dann keine aktuelle Mess-Position geliefert. Dies gilt auch dann, wenn eine dynamische Messung aktiv ist.

Es werden je nach ausgewähltem Messkanal die Messeingänge 1 & 3, oder 2 & 4 deaktiviert. Dieser Opcode sollte deshalb nur ausgeführt werden, wenn die Messwerte der betroffenen Eingänge nicht benötigt werden.

Hinweis für 1Vss-Eingänge:

Wird bei 1Vss-Eingängen ein Fehler festgestellt (auslesbar über [opcRHS](#)¹⁰²), dann sind oder waren die Inkrementalgeber-Signale außerhalb der Spezifikation. Mehr dazu finden Sie im Benutzerhandbuch.

Es wird dann empfohlen nicht nur die Fehlerflags durch das Setzen der Position zurückzusetzen, sondern auch die Verstärkungs- und Offsetregelung. Übergeben Sie dazu das Zeichen ~ beim Parameter Position.

Das komplette Zurücksetzen des Messeingangs über \$ ist in den meisten Fällen nicht erforderlich.

Nach dem Rücksetzen benötigen die Verstärkungs- und die Offset-Regelung einige elektrische Signalperioden, um die optimalen Regelwerte zu bestimmen. Währenddessen ist die Interpolationsgenauigkeit eingeschränkt. Die Messwerte können ungenau sein (es gehen jedoch keine Inkremente verloren). Weiterhin ist die Signaltoleranz in diesem Bereich geringer.

Kompatibilität zu früheren Systemen

Dieser Opcode ist uneingeschränkt kompatibel zu früheren Systemen.

5.5.8 opcRHS: Lese Hardware-Status der Messkanäle

Über diesen Opcode kann ausgelesen werden, ob an den einzelnen Messeingängen ein Fehler erkannt wurde. Weiterhin kann bei Inkrementalgebern mit aktivierter Referenzmarkenverarbeitung ausgelesen werden, ob die Referenzmarke überfahren wurde.

Übersicht

Opcode: 0x38

Name: opcRHS

Funktion: Lese Hardware-Status (Read hardware status)

Parameter-Typ: **Binär**

DLL-Funktion

Der Hardware-Status kann kontinuierlich über [MSC_SetupStaticChannel](#)^[49] und [MSC_ReadStatic](#)^[51] übertragen werden.

Alternativ ist auch eine Übertragung mit [MSC_WriteCommand](#)^[30] möglich.

Anfragedaten zum Irinos-System

Byte 0: Binärwert 2

Antwortdaten vom Irinos-System

Byte 0: Hardware-Status Messkanal 1

Byte 1: Hardware-Status Messkanal 2

Byte n: Hardware-Status Messkanal n+1

Status-Byte für Inkrementalgeber-Messkanäle

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PwrOvld		Refmark	Vector	GComp	OCmp	AmpErr	Fast
d		k		p	p		

PwrOvld

Es wurde eine Überlast in der Spannungsversorgung des/der Inkrementalgeber festgestellt.

Refmark

Die Referenzmarke wurde überfahren.

Vector

Der aus Cosinus- und Sinussignal gebildete Signalvektor ist zu klein. (Kann nur bei Eingangskonfiguration für 1Vss – Inkrementalgeber auftreten.)

GComp

Der Amplitudenregler hat einen Grenzwert erreicht. (Kann nur bei Eingangskonfiguration für 1Vss – Inkrementalgeber auftreten.)

OComp

Der Offsetregler hat einen Grenzwert erreicht. (Kann nur bei Eingangskonfiguration für 1Vss – Inkrementalgeber auftreten.)

AmpErr

Ein oder beide AD-Wandler für die Messung des Sinus-/Cosinussignals sind übersteuert. (Kann nur bei Eingangskonfiguration für 1Vss – Inkrementalgeber auftreten.)

Fast

Die Eingangsfrequenz ist zu hoch.

Status-Byte für induktive Messeingänge Tesa oder IET

Bit 7 Bit 6 Bit 5 Bit 4 Bit 3 Bit 2 Bit 1 Bit 0

ShortCi
rc

ShortCirc

Oszillator-Kurzschluss

Status-Byte für Analog-Eingänge AIN

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
24VOvld	VRefOvld						

24VOvld

Überlast des 24V-Ausgangs am Analog-Stecker.

VRefOvld

Überlast des Referenz-Ausgangs am Analog-Stecker.

Status-Byte für Temperatur-Eingänge TEMP

Wenn der Hardware-Status den Wert 0x00 hat, dann ist der Temperaturwert in Ordnung.

In allen anderen Fällen kann der Temperaturwert ungültig sein (z.B. Kabelbruch). Auf eine Auflistung der einzelnen Fehler-Bits wird hier verzichtet, da eine Interpretation für erweiterte Diagnose-Zwecke nur durch den Hersteller sinnvoll ist.

Kommentar

Der Hardware-Status der Messeingänge sollte regelmäßig überprüft werden, um die Gültigkeit der Messwerte zu verifizieren. Weitere Erläuterungen dazu finden Sie im Benutzerhandbuch.

Kompatibilität zu früheren Systemen

Zur Kompatibilität mit früheren Systemen unterstützt dieser Opcode auch das alte Format, bei dem die Antwort aus Langworten aufgebaut ist, wobei jedes Langwort für eine Irinos-Box steht. Dazu werden keine Anfragedaten an das Irinos-System gesendet.

Hat eine Irinos-Box mehr als 4 Messeingänge, dann wird „nur“ der Hardware-Status dieser Eingänge im Langwort übertragen. Alle anderen nicht.

Darüber hinaus ergeben sich folgende Änderungen:

- Bei den Inkrementalgeber-Messkanälen gibt es das zusätzliche Bit PwrOvld.
- Bei den Messkanälen für induktive Messtaster sowie bei den Analogeingängen wurde bisher kein Hardware-Status gemeldet (d.h. das Langwort hatte immer den Wert 0). Nun wird auch hier ein Hardware-Status gemeldet.

5.5.9 **opcREv: Lese aktuelles Ereignis der jeweiligen Irinos-Boxen**

Über diesen Opcode kann das zuletzt gemeldete Ereignis der einzelnen Irinos-Boxen abgefragt werden. Bei Irinos-Boxen mit 7-Segment-Anzeige wird das Ereignis auf dieser ebenfalls angezeigt.

Übersicht

Opcode:	0x39
Name:	opcREv
Funktion:	Lese aktuelles Ereignis (Read event)
Parameter-Typ:	Binär

DLL-Funktion

Wird mit [MSC_WriteCommand](#)³⁰ übertragen.

Anfragedaten zum Irinos-System

Keine

Antwortdaten vom Irinos-System

Langwort 0: Aktuelles Ereignis der ersten Irinos-Box

Langwort 1: Aktuelles Ereignis der zweiten Irinos-Box

Langwort n: Aktuelles Ereignis der n-ten Irinos-Box

(Langwort = 32 Bit unsigned integer)

Kommentar

Die Nummern und Bedeutungen der Ereignisse entnehmen Sie dem Benutzerhandbuch. 0 bedeutet „kein Ereignis“.

Über den Opcode [opcWEvCfg](#)^{D110} können Ereignisse aktiviert / deaktiviert werden.

Kompatibilität zu früheren Systemen

Dieser Opcode ist bei älteren Systemen nicht verfügbar. Deshalb ist keine Rückwärtskompatibilität erforderlich.

5.5.10 opcSAbsT: Setze Absolut-Zeit (für den Diagnose-Speicher)

Über diesen Opcode können das aktuelle Datum sowie die aktuelle Uhrzeit des Irinos-Systems gesetzt werden. Dadurch werden diese Informationen den darauffolgend erzeugten Einträgen im Diagnose-Speicher hinzugefügt.

Übersicht

Opcode: 0x3A
Name: opcSAbsT
Funktion: Setze Absolut-Zeit (Set absolute time)
Parameter-Typ: [String](#)^{D66}

DLL-Funktion

Wird mit [MSC_WriteCommand](#)^{D30} übertragen.

Anfragestring zum Irinos-System

```
#{Wert 1};{Jahr};{Monat};{Tag};{Stunde};{Minute};{Sekunde};  
{Millisekunde}#
```

Beispiel: #1;2015;06;26;16;49;32;532#

Wert 1

Hier muss immer der Wert 1 stehen.

Jahr

Jahreszahl (immer 4-stellig)

Monat

Monat (1- oder 2-stellig)

Tag

Tag (1- oder 2-stellig)

Stunde

Stunde (1- oder 2-stellig) im 24h-Zeitformat

Minute

Minute (1- oder 2-stellig)

Sekunde

Sekunde (1- oder 2-stellig)

Millisekunde

Millisekunde (1-, 2- oder 3-stellig)

Antwortstring vom Irinos-System

#0# Erfolg

#-n# Parameter Nr n ungültig

#-99# Führende oder anschließende Raute fehlt oder Gesamtgröße falsch

Kommentar

Das Schreiben der System-Zeit ist für den Betrieb des Irinos-Systems nicht erforderlich. Es wird jedoch empfohlen, um zu einem späteren Zeitpunkt exakt nachvollziehen zu können, wann ein Diagnose-Ereignis eingetreten ist.

Beim Ausschalten bzw. Reset des Irinos-Systems geht die System-Zeit verloren. Sie muss daher nach jedem Einschalten (Power On) neu geschrieben werden.

Die interne Uhr berücksichtigt keine Schaltjahre oder Schaltsekunden. Weiterhin hat die interne Uhr keine hohe Genauigkeit. Es wird daher empfohlen die System-Zeit bei Dauerbetrieb einmal täglich zu aktualisieren. Sie kann beliebig oft überschrieben werden.

Kompatibilität zu früheren Systemen

Dieser Opcode ist bei älteren Systemen nicht verfügbar. Deshalb ist keine Rückwärtskompatibilität erforderlich.

5.5.11 opcWEvCfg: Schreibe Ereignis-Konfiguration

Über diesen Opcode kann die Konfiguration eines Ereignisses geändert werden. Beispielsweise lässt sich eine Ereignis-Meldung deaktivieren.

Für die meisten Applikationen sind die Standard-Einstellungen ausreichend, so dass dieser Opcode nicht benötigt wird.

Übersicht

Opcode:	0x3D
Name:	opcWEvCfg
Funktion:	Schreibe Ereignis-Konfiguration (Write event configuration)
Parameter-Typ:	String ^{D66}

DLL-Funktion

Wird mit [MSC_WriteCommand](#)^{D30} übertragen.

Anfragestring zum Irinos-System

```
{Nr Irinos-Box};{Wert 1};{Ereignis-Nr};{EreignisFreigegeben};  
{Max. Anzahl Diagnose-Einträge}#
```

Beispiel: #2;1;7;1;5#

Nr Irinos-Box

Nummer (Adresse) der Irinos-Box, bei welcher die Ereignis-Konfiguration geändert werden soll.

Wert 1

Hier muss immer der Wert 1 stehen.

Ereignis-Nr

Nr des Ereignisses. Die Ereignis-Nummern entnehmen Sie dem Benutzerhandbuch.

EreignisFreigegeben

- 0 bedeutet, dass das Auftreten des Ereignisses ignoriert wird.
- 1 bedeutet, dass das Ereignis beim Auftreten gemeldet wird, z.B. durch Anzeige auf der 7-Segment-Anzeige.

Max. Anzahl Diagnose-Einträge

Über diesen Wert kann die maximale Anzahl an erzeugten Diagnose-Einträgen (seit dem Einschalten) geändert werden. Durch den Wert 0 werden überhaupt keine Diagnose-Einträge erzeugt. Der Vorgabe-Wert für die meisten Ereignisse ist 10. Damit soll verhindert werden, dass der Diagnose-Speicher bei einem wiederkehrenden Ereignis mit nichts-sagenden Einträgen gefüllt wird.

Antwortstring vom Irinos-System

#0# Erfolg

#-n# Parameter Nr n ungültig

#-99# Führende oder anschließende Raute fehlt oder Gesamtgröße falsch

Kommentar

Inwieweit die Ereignisse parametrierbar sind, entnehmen Sie dem Benutzerhandbuch.

Die Ereignis-Konfiguration wird nur bei derjenigen Irinos-Box geändert, die über den Parameter „Nr Irinos-Box“ angesprochen ist. Soll die Ereignis-Konfiguration an mehreren oder allen Irinos-Boxen geändert werden, so muss dieser Opcode separat an jede Irinos-Box gesendet werden. Gleiches gilt, wenn die Konfiguration verschiedener Ereignisse geändert werden soll.

Kompatibilität zu früheren Systemen

Dieser Opcode ist bei älteren Systemen nicht verfügbar. Deshalb ist keine Rückwärtskompatibilität erforderlich.

5.5.12 opcClrEv: Lösche Ereignis

Über diesen Opcode kann ein aktives Ereignis gelöscht werden.

Übersicht

Opcode: 0x3E
Name: opcClrEv
Funktion: Lösche Ereignis (Clear event)
Parameter-Typ: [String](#)⁶⁶

DLL-Funktion

Wird mit [MSC_WriteCommand](#)³⁰ übertragen.

Anfragestring zum Irinos-System

`{Nr Irinos-Box};{Wert 1};{Ereignis-Nr}#`

Beispiel: `#0;1;7#`

Nr Irinos-Box

Nummer (Adresse) der Irinos-Box, bei welcher das Ereignis gelöscht werden soll.

Wert 1

Hier muss immer der Wert 1 stehen.

Ereignis-Nr

Nr des Ereignisses. Die Ereignis-Nummern entnehmen Sie dem Benutzerhandbuch.

Antwortstring vom Irinos-System

#0# Erfolg

#-n# Parameter Nr n ungültig

#-99# Führende oder anschließende Raute fehlt oder Gesamtgröße falsch

Kommentare

- Üblicherweise wurde das zu löschende Ereignis zuvor über den Opcode [opcREv](#)¹⁰⁶ ausgelesen.
- Wenn ein Ereignis gelöscht werden soll, das nicht aktiv ist, dann hat dieser Opcode keine Auswirkung.
- Das Löschen eines Ereignisses behebt nicht seine Ursache! Je nach Ereignis kann es nach dem Löschen gleich wieder auftreten.
- Das Ereignis wird nur bei derjenigen Irinos-Box gelöscht, die über den Parameter „Nr Irinos-Box“ angesprochen ist. Soll das Ereignis bei mehreren Irinos-Boxen gelöscht werden, so muss dieser Opcode separat an jede Irinos-Box gesendet werden.
- Bestimmte Ereignisse werden automatisch gelöscht, sobald die Ursache behoben wurde. Dies gilt beispielsweise für den Oszillator-Kurzschluss bei der Irinos-Box für induktive Wegaufnehmer. Details dazu entnehmen Sie dem Benutzerhandbuch.

Kompatibilität zu früheren Systemen

Dieser Opcode ist bei älteren Systemen nicht verfügbar. Deshalb ist keine Rückwärtskompatibilität erforderlich.

5.6 Opcodes: Messung

5.6.1 opcRS: Lese statische Messwerte

Über diesen Opcode können die aktuellen (statischen) Messwerte ausgelesen werden.

Übersicht

Opcode: 0x40

Name: opcRS

Funktion: Lese statische Messwerte (Read static)

Parameter-Typ: **Binär**

DLL-Funktion

Die statischen Messwerte können kontinuierlich über [MSC_SetupStaticChannel](#)^[49] und [MSC_ReadStatic](#)^[51] übertragen werden. Alternativ ist auch eine Übertragung mit [MSC_WriteCommand](#)^[30] möglich.

Anfragedaten zum Irinos-System

Keine

Antwortdaten vom Irinos-System

Langwort 0 Messwert von Messeingang 1

Langwort 1 Messwert von Messeingang 2

Langwort n Messwert von Messeingang n+1

Kommentar

Ein Langwort entspricht einem 32 Bit signed integer. Alle Messwerte werden als Langwort übertragen. Dies gilt auch für 8 und 16 Bit Messeingänge.

Es werden immer die Messwerte übertragen, die in der aktiven Kanal-Liste definiert sind (siehe [opcWCA](#)^{□82}, [opcWCL](#)^{□85} und [opcACL](#)^{□89}). Nach dem Einschalten sind dies alle Messkanäle.

Kompatibilität zu früheren Systemen

Dieser Opcode ist uneingeschränkt kompatibel zu früheren Systemen.

5.6.2 **opcBIO: Lese digitale Eingänge / schreibe digitale Ausgänge**

Über diesen Opcode können die digitalen Eingänge eingelesen und die digitalen Ausgänge gesetzt werden.

Übersicht

Opcode: 0x42

Name: opcBIO

Funktion: Lese digitale Eingänge / schreibe digitale Ausgänge (Bit I/O)

Parameter-Typ: **Binär**

DLL-Funktion

Die digitalen Ein-/Ausgänge können kontinuierlich über [MSC_SetupStaticChannel](#)^{□49} und [MSC_ReadStatic](#)^{□51} übertragen werden. Sobald die Binärdaten der digitalen Ausgänge geändert wurden, muss die Funktion [MSC_RefreshChannel](#)^{□53} aufgerufen werden, damit die DLL diese Daten auch übernimmt.

Alternativ ist auch eine Übertragung mit [MSC_WriteCommand](#)^{□30} möglich.

Anfragedaten zum Irinos-System

Byte 0	Digitale Ausgänge 1..8
Byte 1	Digitale Ausgänge 9..16
Byte 2	Digitale Ausgänge 17..24
usw.	

Antwortdaten vom Irinos-System

Byte 0	Ist-Zustand der digitalen Ausgänge 1..8
Byte 1	Ist-Zustand der digitalen Ausgänge 9..16
Byte 2	Ist-Zustand der digitalen Ausgänge 17..24
usw.	
Byte x+0	Zustand der digitalen Eingänge 1..8
Byte x+1	Zustand der digitalen Eingänge 9..16
Byte x+2	Zustand der digitalen Eingänge 17..24
usw.	

Kommentar

In den Antwortdaten werden immer zunächst die Ausgangs-Bytes zurückgespiegelt. Die Datenlänge der digitalen Ausgänge ist immer identisch zur Datenlänge der digitalen Eingänge. D.h.

- Werden 8 Ausgangsbytes in den Anfragedaten geschrieben, dann werden in den Antwortdaten 8 zurückgespiegelte Ausgangsbytes plus 8 Eingangsbytes zurückgegeben.
- Werden 16 Ausgangsbytes in den Anfragedaten geschrieben, dann werden in den Antwortdaten 16 zurückgespiegelte Ausgangsbytes plus 16 Eingangsbytes zurückgegeben.

Die Anzahl der Ein-/Ausgangs-Bytes bzw. -Bits ist unabhängig von der tatsächlichen Anzahl an verfügbaren Ein-/Ausgängen:

- Werden mehr Ausgangs-Bits geschrieben, als Ausgänge verfügbar sind, dann werden die nicht zugeordneten Bits ignoriert.
- Werden weniger Eingangs-Bits gelesen, als Eingänge verfügbar sind, dann bleiben die weiteren Eingänge unberücksichtigt.
- Werden mehr Eingangs-Bits gelesen, als Eingänge verfügbar sind, so werden die nicht zugeordneten Bits mit 0 gefüllt.

Bit I/O ist in gewisser Weise ähnlich zu statischen Messungen. Diese Informationen werden im Regelfall kontinuierlich zwischen PC und Irinos Box ausgetauscht. Ein wichtiger Unterschied ist, dass Bit I/O bidirektional funktioniert. Bit I/O Daten können sowohl gesendet als auch empfangen werden.

Normalerweise wird dieser Opcode zusammen mit der [MSC_SetupStaticChannel](#)^{D49} Funktion genutzt. Diese Funktion definiert einen Sende- und einen Empfangspuffer. Der Sendepuffer enthält die Ausgangsbits und der Empfangspuffer die Eingangsbits.

Die DLL enthält einen lokalen Puffer für die Ausgabedaten. Wenn die Applikation Änderungen der Ausgabedaten wünscht, muss sie dies der DLL mit der Funktion [MSC_RefreshChannel](#)^{D53} mitteilen.

Kompatibilität zu früheren Systemen

Dieser Opcode ist uneingeschränkt kompatibel zu früheren Systemen.

Bei früheren Systemen war die Länge der Ein-/Ausgangsbits auf jeweils 64 Bits (= 8 Bytes) fixiert. Diese Beschränkung gilt nicht mehr. Es ist jedoch auch kein Problem, wenn weiterhin jeweils 64 Bits übertragen werden, solange keine darüber hinausgehenden Ein-/Ausgänge genutzt werden sollen.

5.6.3 **opcBIORO: Status der digitalen Ein-/Ausgänge abfragen**

Dieser Opcode ist verfügbar ab Softwarestand 1.4.x.x.

Über diesen Opcode kann der Status der digitalen Ein- & Ausgänge eingelesen werden.

Die Abfrage erfolgt im selben Format wie über den Opcode [opcBIO](#)^{D115}, jedoch werden die gesendeten Ausgangsdaten nicht übernommen. Ein typischer Anwendungsfall ist die einmalige Abfrage der Ist-Zustände nach dem Start der Messrechner-Software.

Übersicht

Opcode:	0x43
Name:	opcBIORO
Funktion:	Status der digitalen Ein-/Ausgänge abfragen (Bit I/O)
Parameter-Typ:	Binär

DLL-Funktion

Wird mit [MSC_WriteCommand](#)^{D30} übertragen.

Anfragedaten zum Irinos-System

Byte 0	Digitale Ausgänge 1..8
Byte 1	Digitale Ausgänge 9..16
Byte 2	Digitale Ausgänge 17..24
usw.	

Um das gleiche Datenformat wie beim Opcode [opcBIO](#)^{D115} zu gewährleisten, werden auch hier die Ausgangsdaten gesendet. Sie werden aber vom Irinos-System nicht übernommen.

Antwortdaten vom Irinos-System

Byte 0	Ist-Zustand der digitalen Ausgänge 1..8
Byte 1	Ist-Zustand der digitalen Ausgänge 9..16
Byte 2	Ist-Zustand der digitalen Ausgänge 17..24
usw.	
Byte x+0	Zustand der digitalen Eingänge 1..8
Byte x+1	Zustand der digitalen Eingänge 9..16
Byte x+2	Zustand der digitalen Eingänge 17..24
usw.	

Kommentar

In den Antwortdaten werden immer zunächst die Ausgangs-Bytes zurückgespiegelt. Die Datenlänge der digitalen Ausgänge ist immer identisch zur Datenlänge der digitalen Eingänge. D.h.

- Werden 8 Ausgangsbytes in den Anfragedaten geschrieben, dann werden in den Antwortdaten 8 zurückgespiegelte Ausgangsbytes plus 8 Eingangsbytes zurückgegeben.

- Werden 16 Ausgangsbytes in den Anfragedaten geschrieben, dann werden in den Antwortdaten 16 zurückgespiegelte Ausgangsbytes plus 16 Eingangsbytes zurückgegeben.

Die Anzahl der Ein-/Ausgangs-Bytes bzw. -Bits ist unabhängig von der tatsächlichen Anzahl an verfügbaren Ein-/Ausgängen:

- Werden weniger Eingangs-Bits gelesen, als Eingänge verfügbar sind, dann bleiben die weiteren Eingänge unberücksichtigt.
- Werden mehr Eingangs-Bits gelesen, als Eingänge verfügbar sind, so werden die nicht zugeordneten Bits mit 0 gefüllt.

Kompatibilität zu früheren Systemen

Dieser Opcode ist bei älteren Systemen nicht verfügbar. Deshalb ist keine Rückwärtskompatibilität erforderlich.

5.6.4 opcRSW: Lese Statuswort für dynamische Messungen

Über diesen Opcode kann der aktuelle Zustand der dynamischen Messungen abgefragt werden.

Übersicht

Opcode:	0x44
Name:	opcRSW
Funktion:	Lese Statuswort für dynamische Messungen (Read statusword)
Parameter-Typ:	Binär

DLL-Funktion

Wird mit [MSC_WriteCommand](#)³⁰ übertragen.

Anfragedaten zum Irinos-System

Keine

Antwortdaten vom Irinos-System

Langwort (32 Bit) mit Status-Informationen. Siehe folgende Tabelle.

Kommentar

Folgende Bits können mit Verzögerung aktiv / inaktiv werden:

- „Trigger X war aktiv und ist nun inaktiv“
- „Mindestens 1 Triggerpuls wurde auf dem Trigger Nr. X erzeugt“
- „Dynamische Messung X war aktiv und ist nun inaktiv“
- „Mindestens 1 Messwert wurde durch die dynamische Messung X aufgenommen.“

Kompatibilität zu früheren Systemen

Dieser Opcode ist kompatibel zu früheren Systemen. Es gelten folgende Anmerkungen:

- Die Bits 12 und 28 wurden bisher für interne Informationen verwendet. Dies ist nun nicht mehr der Fall. Auf die Applikation hat dies jedoch keinen Einfluss.
- Der interne Ablauf der dynamischen Messung hat sich grundlegend geändert. Folgende Bits können deshalb unter Umständen mit Verzögerung aktiv/inaktiv werden:
 - „Trigger X war aktiv und ist nun inaktiv“

- „Mindestens 1 Triggerpuls wurde auf dem Trigger Nr. X erzeugt“
- „Dynamische Messung X war aktiv und ist nun inaktiv“
- „Mindestens 1 Messwert wurde durch die dynamische Messung X aufgenommen.“

Statuswort der dynamischen Messung

Bit Nr.	Funktion
0	Trigger 1 aktiv
1	Trigger 1 war aktiv und ist nun inaktiv
2	Mindestens 1 Triggerpuls wurde auf dem Trigger Nr. 1 erzeugt
3	
4	Dynamische Messung 1 ist aktiv
5	Dynamische Messung 1 war aktiv und ist nun inaktiv
6	Mindestens 1 Messwert wurde durch die dynamische Messung 1 aufgenommen
7	Messwertabfrage der dynamischen Messung 1 durch den PC ist aktiv
8	Messwertpuffer für die dynamische Messung 1 im Irinos-System ist voll
9	
10	
11	
12	

Bit Nr.	Funktion
13	
14	
15	
16	Trigger 2 aktiv
17	Trigger 2 war aktiv und ist nun inaktiv
18	Mindestens 1 Triggerpuls wurde auf dem Trigger Nr. 2 erzeugt
19	
20	Dynamische Messung 2 ist aktiv
21	Dynamische Messung 2 war aktiv und ist nun inaktiv
22	Mindestens 1 Messwert wurde durch die dynamische Messung 2 aufgenommen
23	Messwertabfrage der dynamischen Messung 2 durch den PC ist aktiv
24	Messwertpuffer für die dynamische Messung 2 im Irinos-System ist voll
25	
26	
27	
28	
29	
30	
31	

5.6.5 opcRDC: Lese Anzahl der dynamischen Messwerte

Dieser Opcode ist verfügbar ab Softwarestand 1.8.x.x.

Über diesen Opcode kann die Anzahl der aufgezeichneten Messwert-Samples einer dynamischen Messung ausgelesen werden.

Übersicht

Opcode: 0x45
Name: opcRDC
Funktion: Lese Anzahl der dynamischen Messwert-Samples
Parameter-Typ: **Binär**

DLL-Funktion

Wird mit [MSC_WriteCommand](#)³⁰ übertragen.

Anfragedaten zum Irinos-System

Keine

Antwortdaten vom Irinos-System

Je dynamische Messung 1 Langwort (32 Bit) mit dem Zählerstand.

Kommentar

Dieser Opcode kann hilfreich sein, um nach einem vorzeitigen Stop einer dynamischen Messung herauszufinden, ob bereits alle Messwert-Samples zum PC übertragen wurden, oder ob die Übertragung noch andauert.

Während eine dynamische Messung aktiv ist, wird der Zählerstand 0 ausgegeben (d.h. zwischen Start und Stop einer dynamischen Messung).

Kompatibilität zu früheren Systemen

Dieser Opcode war auf früheren Systemen nicht verfügbar.

5.6.6 opcDDMx: Definiere dynamische Messung

Über diesen Opcode wird eine dynamische Messung konfiguriert.

Übersicht

Opcode:	0x50 für dynamische Messung 1 0x51 für dynamische Messung 2
Name:	opcDDM1 für dynamische Messung 1 opcDDM2 für dynamische Messung 2
Funktion:	Definiere dynamische Messung (Define dynamic measurement)
Parameter-Typ:	String ^{D66}

DLL-Funktion

Wird mit [MSC_WriteCommand](#)^{D30} übertragen.

Anfragestring zum Irinos-System

```
#{Trigger-Nummer};{Nummer der Kanal-Liste};{Aktiv};{Anzahl  
Messwerte}#
```

Trigger-Nummer

Nummer des für die dynamische Messung verwendeten Triggers (1 oder 2). Der Trigger muss über den Opcode [opcDT](#)^{D91} definiert werden.

Nummer der Kanal-Liste

Nummer der Kanal-Liste, die für die dynamische Messung verwendet werden soll (1..10). Die Kanal-Liste muss über den Opcode [opcWCL](#)^{D85} definiert werden.

Aktiv

1 um die dynamische Messung zu aktivieren.
0 um die dynamische Messung zu deaktivieren.

Anzahl Messwerte

Maximale Anzahl aufzunehmender Messwerte. Verwenden Sie * für ‚keine Begrenzung‘.

Antwortstring vom Irinos-System

#0# Erfolg

#-n# Parameter Nr n ungültig

#-99# Führende oder anschließende Raute fehlt oder Gesamtgröße falsch

Kommentar

Eine dynamische Messung wird gestartet, wenn eine dynamische Messung „aktiv“ ist und der zugehörige Trigger ebenfalls aktiv ist. Dazu kann entweder zuerst der Trigger und dann die dynamische Messung aktiviert werden oder umgekehrt.

Eine dynamische Messung kann auf 4 Wegen beendet werden:

- Über den Parameter „Aktiv = 0“ dieses Opcodes opcDDMx.
- Über den Parameter „Anzahl Messwerte“ dieses Opcodes opcDDMx.
- Indem der zugehörige Trigger deaktiviert wird (siehe opcIT).

- Indem dem Trigger eine Ende-Wert zugewiesen wird (siehe opcDT).

Eine Endlos-Messung ist nicht zulässig. Typischerweise liegt die Dauer einer dynamischen Messung im Bereich < 60s.

Eine dynamische Messung nimmt die Werte von allen der in der angegebenen Kanalliste definierten Mess-Eingänge auf. Dies ergibt eine Kurve für jeden Eingang. Für jede dieser Kurven muss der DLL mit der Funktion [MSC_AttachSubChannelBuffer](#)^{D56} ein Puffer übergeben werden. Es ist sehr wichtig, dass alle erforderlichen Puffer übergeben wurden und dass diese groß genug sind.

Die Kanalliste 0 enthält stets alle Messkanäle und sollte für die dynamische Messung nicht benutzt werden. Weitere Angaben zur Kanalzahl / Datenrate entnehmen Sie der Bedienungsanleitung des Irinos-Systems.

Kompatibilität zu früheren Systemen

Dieser Opcode ist grundsätzlich kompatibel zu früheren Systemen. Da die System-interne Realisierung der dynamischen Messung grundlegend geändert wurde, kann in Einzelfällen eine anderweitige Parametrierung erforderlich sein.

5.6.7 opcRDMx: Lese dynamische Messwerte

Über diesen Opcode können bereits aufgezeichnete dynamische Messwerte ausgelesen werden.

Übersicht

Opcode:	0x60 für dynamische Messung 1
	0x61 für dynamische Messung 2

Name: opcRDM1 für dynamische Messung 1
 opcRDM2 für dynamische Messung 2

Funktion: Lese dynamische Messwerte (Read dynamic measurement)

Parameter-Typ: **Binär**

DLL-Funktion

Das Auslesen der dynamischen Messwerte ist ein komplexer Vorgang. Die Interpretation der Binär-Daten übernimmt immer die DLL. Eine manuelle Interpretation ist nicht vorgesehen.

Um dynamische Messwerte auslesen zu können, muss ein dynamischer Übertragungs-Kanal verwendet werden:

- Ein dynamischer Übertragungs-Kanal wird mit [MSC_SetupExtendedDynamicChannel](#)⁵⁴ angelegt.
- Für jeden Mess-Eingang, der in der dynamischen Messung verwendet wird, muss ein Messwert-Puffer erzeugt und dem dynamischen Übertragungs-Kanal via [MSC_AttachSubChannelBuffer](#)⁵⁶ zugewiesen werden.

Die DLL schreibt die Messwerte selbsttätig in die zugewiesenen Puffer. Die Applikation kann sie direkt aus den Puffern auslesen. Alle Messwerte werden als 32 Bit – Werte (signed) im Puffer abgelegt.

Über die Funktion [MSC_GetPosition](#)⁵⁹ kann während und nach einer dynamischen Messung ermittelt werden, wie viele Messwerte bereits in die zugewiesenen Puffer geschrieben wurden.

Kompatibilität zu früheren Systemen

Dieser Opcode ist uneingeschränkt kompatibel zu früheren Systemen.

5.7 Opcodes: Service

5.7.1 opcRST: System Reset

Über diesen Opcode kann ein Reset des gesamten Irinos-Systems durchgeführt werden. Verwenden Sie diesen Opcode nur nach Rücksprache mit dem Hersteller.

Übersicht

Opcode: 0x7E
Name: opcRST
Funktion: System-Reset
Parameter-Typ: [String](#)⁶⁶

DLL-Funktion

Wird mit [MSC_WriteCommand](#)³⁰ übertragen.

Anfragestring zum Irinos-System

```
#RESET_MTS;{Verzögerung Master-Box};{Verzögerung Slave-Boxen}#
```

Verzögerung Master-Box

Zeit in ms bis der Reset an der Master-Box (d.h. der ersten Box) ausgeführt wird.

Verzögerung Slave-Boxen

Zeit in ms bis der Reset an den Slave-Boxen (d.h. auf allen weiteren Boxen) ausgeführt wird.

Antwortstring vom Irinos-System

#0# Erfolg

#-n# Parameter Nr n ungültig

#-99# Führende oder anschließende Raute fehlt oder Gesamtgröße falsch

Kommentar

Die Verzögerungs-Zeit für den Master muss größer sein, als die Verzögerungs-Zeit an den Slaves. Verwenden Sie folgende Werte:

```
#RESET_MTS;2000;500#
```

Der Reset-Befehl wird vom Master per ILink-Kommunikation an die Slaves gesendet. Wenn aus irgendeinem Grund die Kommunikation nicht mehr funktioniert, kann an den Slaves kein Reset ausgelöst werden.

Kompatibilität zu früheren Systemen

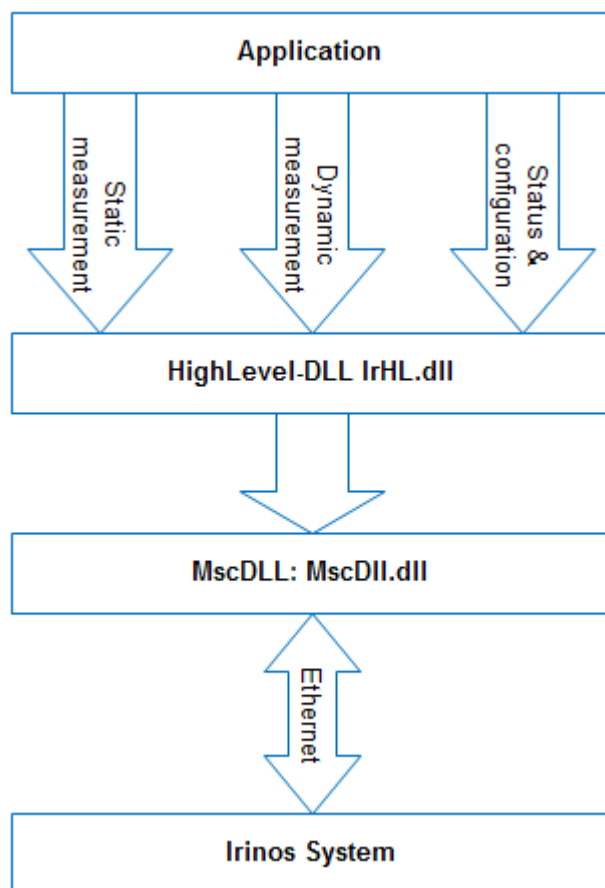
Dieser Opcode ist bei älteren Systemen nicht verfügbar. Deshalb ist keine Rückwärtskompatibilität erforderlich.

HighLevel-Funktionen (Optional)

6 HighLevel-Funktionen (Optional)

Alle HighLevel-Funktionen können mit der HighLevel-DLL (IrHL.dll) genutzt werden. Eine Übersicht über die [Unterschiede zwischen HighLevel-DLL und MscDll](#)¹⁶ zeigt das entsprechende Kapitel dieser Dokumentation.

Die HighLevel DLL setzt auf der [MscDLL](#)²⁰ auf, d.h. sie ersetzt diese nicht, sondern kapselt verschiedene Funktionen der MscDLL in übergeordneten Funktionen:



HighLevel-DLL Struktur

Allgemeine Funktionen

Funktions-Name	Beschreibung
IRHL_GetVersionHL ¹³⁴	Versions-Informationen der HighLevel-DLL abrufen.

Funktions-Name	Beschreibung
IRHL_Connect ¹³⁶	Verbindung aufbauen
IRHL_Disconnect ¹³⁷	Verbindung beenden
IRHL_GetCommStatus ¹³⁸	Verbindungs-Status abfragen
IRHL_WriteCommand ¹³⁹	Direkten Befehl ausführen
IRHL_GetCurrentHandle ¹⁴⁰	Aktuelles Verbindungshandle abfragen (nur erforderlich zur Verwendung der MscDLL-Funktionen)

Statische Messung / Bit-I/O / Hardware-Status

Funktions-Name	Beschreibung
IRHL_StaticMeasurement_Start ¹⁴¹	Kontinuierliche Abfrage der statischen Messwerte starten
IRHL_StaticMeasurement_ReadArray ¹⁴²	Aktuelle statischen Messwerte abfragen
IRHL_StaticHardwareStatus_Start ¹⁴³	Kontinuierliche Abfrage des Hardware-Status starten
IRHL_StaticHardwareStatus_ReadArray ¹⁴⁴	Aktuellen Hardware-Status abfragen
IRHL_StaticBIO_Start ¹⁴⁵	Kontinuierlichen Austausch der Bit-I/O - Daten starten (digitale Ein-/Ausgänge)
IRHL_StaticBIO_ReadInputs ¹⁴⁶	Aktuelle Eingangs-Bits lesen

Funktions-Name	Beschreibung
IRHL_StaticBIO_WriteOutputs ¹⁴⁷	Aktuelle Ausgangs-Daten schreiben

Dynamische Messung

Funktions-Name	Beschreibung
IRHL_DynReset ¹⁴⁸	Dynamische Messung zurücksetzen
IRHL_DynAddMeasChannel ¹⁴⁹	Messkanal zur dynamischen Messung hinzufügen
IRHL_DynTimeStart ¹⁵⁰	Zeit-gesteuerte dynamische Messung starten
IRHL_DynPosStart ¹⁵²	Positions-gesteuerte dynamische Messung starten
IRHL_DynStop ¹⁵⁴	Dynamische Messung beenden
IRHL_DynGetValuesAvailable ¹⁵⁵	Anzahl der bereits verfügbaren Messwerte auslesen
IRHL_DynReadValueArray ¹⁵⁶	Dynamische Messwerte auslesen

6.1 IRHL_GetVersionHL

Diese Funktion gibt die aktuelle Version der HighLevel-DLL zurück.

Definition

MSC_STATUS

```
IRHL_GetVersionHL(unsigned long* pulApi_Major, unsigned long*  
pulApi_Minor, unsigned long* pulDLL_Major, unsigned long*  
pulDLL_Minor);
```

Parameter

pulApi_Major

Gibt den Major-Teil der API Version der HighLevel-DLL zurück.

pulApi_Minor

Gibt den Minor-Teil der API Version der HighLevel-DLL zurück.

pulDLL_Major

Gibt den Major-Teil der Version der HighLevel-DLL zurück.

pulDLL_Minor

Gibt den Minor-Teil der Version der HighLevel-DLL zurück.

Rückgabewert

Die Funktion gibt immer MSC_STATUS_SUCCESS zurück.

Kommentare

Die API-Version wird nur bei Änderungen an der Programmierschnittstelle erhöht.

Die DLL-Version wird bei jeder Änderung der DLL erhöht.

Für die API-Version gilt:

Wenn Veränderungen an der Programmierschnittstelle vorgenommen werden, die kompatibel mit älteren Versionen sind, wird die niedrigere Versionsnummer erhöht.

Bei Änderungen, die eine Inkompatibilität mit älteren Versionen verursachen, wird die höhere Versionsnummer erhöht.

Anwendungen sollten den Rückgabewert dieser Funktion in Abhängigkeit einer in der Applikation hinterlegten Konstante überprüfen, um sicherzustellen, dass die DLL die erwartete API Version unterstützt. Die Nummer der Hauptversion muss der erwarteten Nummer exakt entsprechen. Die Nummer der Nebenversion muss größer oder gleich groß sein wie die erwartete Nummer.

Der Aufruf dieser Funktion ist auch bei inaktiver Kommunikation möglich.

6.2 IRHL_Connect

Diese Funktion baut eine Verbindung auf.

Definition

```
MSC_STATUS  
IRHL_Connect(void);
```

Rückgabewert

Die Funktion gibt im Erfolgsfall MSC_STATUS_SUCCESS zurück, bei Fehlern wird der [Fehlercode](#)¹²⁶ zurückgegeben.

Kommentare

Falls bereits eine Verbindung besteht, wird diese zuvor geschlossen.

Nach einem erfolgreichen Verbindungs-Aufbau wird automatische die Verbindungsüberwachung über Timeouts aktiviert (siehe hierzu auch [IRHL_GetCommStatus](#)¹³⁸).

Vor dem Programmende muss die Verbindung immer über die Funktion [IRHL_Disconnect](#)¹³⁷ beendet werden.

Intern verwendete Funktionen der MscDLL

[MSC_EnumerateDevices](#)¹³²

[MSC_GetDeviceInfo](#)^{□33}

[MSC_OpenDevice](#)^{□34}

[MSC_InitDevice](#)^{□35}

[MSC_Start](#)^{□37}

[MSC_SetNotificationCallback](#)^{□47}

6.3 IRHL_Disconnect

Diese Funktion beendet eine bestehende Verbindung.

Definition

```
MSC_STATUS  
IRHL_Disconnect(void);
```

Rückgabewert

Die Funktion gibt immer MSC_STATUS_SUCCESS zurück.

Kommentare

Rufen Sie diese Funktion immer auf, bevor Sie eine Applikation beendet. Es kann sonst zu einer Fehlermeldung der noch offenen DLL kommen.

Falls keine Verbindung besteht, dann passiert nichts.

Intern verwendete Funktionen der MscDLL

[MSC_Stop](#)^{□39}

[MSC_CloseDevice](#)^{□36}

6.4 IRHL_GetCommStatus

Diese Funktion ermöglicht die Abfrage des aktuellen Verbindungs-Status.

Definition

```
unsigned char  
IRHL_GetCommStatus(unsigned long* pulNTmos, unsigned long*  
pulNStaticMsgs, unsigned long* pulReserved1, unsigned long*  
pulReserved2);
```

Parameter

pulNTmos

Liefert die Anzahl der aufgetretenen Verbindungs-Timeouts zurück. Übergeben Sie den Wert NULL, wenn die Abfrage nicht benötigt wird.

pulNStaticMsgs

Liefert die Anzahl der erhaltenen statischen Messwert-Updates zurück. Übergeben Sie den Wert NULL, wenn die Abfrage nicht benötigt wird.

pulReserved1

Reserviert für zukünftige Verwendung. Übergeben Sie hier immer NULL.

pulReserved2

Reserviert für zukünftige Verwendung. Übergeben Sie hier immer NULL.

Rückgabewert

- 0 Kommunikation aktiv
- 1 keine Kommunikation

Kommentare

Zur Abfrage des Verbindungsstatus muss zuvor die statische Messung über den Befehl [IRHL_StaticMeasurement_Start](#)¹⁴¹ gestartet werden.

6.5 IRHL_WriteCommand

Diese Funktion kapselt die Funktion [MSC_WriteCommand](#)¹³⁰ und ermöglicht damit einen direkten Datenaustausch mittels eines [OpCodes](#)⁶².

Definition

```
MSC_STATUS  
IRHL_WriteCommand(unsigned char ucOpcode, unsigned char  
aucBuf[], unsigned long ulBufSize, unsigned long ulBytesToSend,  
unsigned long* pulBytesReceived);
```

Parameter

ucOpcode

Opcode, siehe [Opcode-Liste](#)⁶²

aucBuf

Zeiger auf den Puffer mit den zu sendenden Daten. Hier werden auch die empfangenen Daten abgelegt.

ulBufSize

Größe des Puffers aucBuf in Bytes.

ulBytesToSend

Anzahl der zu sendenden Bytes. Dieser Wert wird automatisch auf ulBufSize begrenzt.

pulBytesReceived

Anzahl der empfangenen Bytes.

Rückgabewert

Die Funktion gibt im Erfolgsfall MSC_STATUS_SUCCESS zurück, bei Fehlern wird der [Fehlercode](#)¹²⁶ zurückgegeben.

Kommentare

-

Intern verwendete Funktionen der MscDLL

[MSC_WriteCommand](#)³⁰

6.6 IRHL_GetCurrentHandle

Ermöglicht die Abfrage des aktuellen "Verbindungs-Handles". Dies ist nur dann erforderlich, wenn LowLevel-Funktionen der MscDLL verwendet werden sollen.

Definition

```
MSC_STATUS  
IRHL_GetCurrentHandle (MSC_HANDLE *pHandle);
```

Parameter

pHandle

Über diesen Parameter wird das aktuelle Handle zurückgegeben.

Rückgabewert

Die Funktion gibt im Erfolgsfall MSC_STATUS_SUCCESS zurück, wenn kein gültiges Handle verfügbar ist, wird MSC_STATUS_INVALID_HANDLE zurückgegeben.

Kommentare

Ein Handle ist nur erforderlich, wenn direkt Funktionen der MscDLL aufgerufen werden sollen (diese beginnen mit MSC_).

6.7 IRHL_StaticMeasurement_Start

Start des kontinuierlichen Datenaustauschs statischer Messwerte zwischen Gerät und DLL.

Definition

```
MSC_STATUS  
IRHL_StaticMeasurement_Start(void);
```

Rückgabewert

Die Funktion gibt im Erfolgsfall MSC_STATUS_SUCCESS zurück, bei Fehlern wird der [Fehlercode](#)¹²⁶ zurückgegeben.

Kommentare

Diese Funktion sollte von der Applikation einmalig nach dem Verbindungsaufbau aufgerufen werden. Die statischen Messwerte werden dann kontinuierlich zwischen Gerät und DLL ausgetauscht. Die Abfrage der aktuellen statischen Messwerte erfolgt über die Funktion [IRHL_StaticMeasurement_ReadArray](#)¹⁴².

Eine Erkennung des Verbindungsstatus (siehe [IRHL_GetCommStatus](#)¹³⁸) erfolgt über den Austausch statischer Messwerte, d.h. die Abfrage des Verbindungsstatus macht erst nach dem Aufruf dieser Funktion Sinn.

Intern verwendete Funktionen der MscDLL

[MSC_SetupStaticChannel](#)¹⁴⁹

[MSC_SetNotificationCallback](#)¹⁴⁷

6.8 IRHL_StaticMeasurement_ReadArray

Auslesen der aktuellen statischen Messwerte von der DLL durch die Applikation.

Definition

```
MSC_STATUS  
IRHL_StaticMeasurement_ReadArray(signed long aslValues[],  
unsigned long ulMaxChannels);
```

Parameter

aslValues

Array, in welches die statischen Messwerte geschrieben werden sollen.

ulMaxChannels

Maximale Anzahl der in das Array aslValues zu schreibenden Messwerte (entspricht der Anzahl der Messkanäle).

Rückgabewert

Die Funktion gibt im Erfolgsfall MSC_STATUS_SUCCESS zurück, bei Fehlern wird der [Fehlercode](#)²⁶ zurückgegeben.

Kommentare

Der kontinuierliche Austausch der statischen Messwerte muss zuvor mit [IRHL_StaticMeasurementStart](#)¹⁴¹ gestartet werden.

Das Array aslValues muss mindestens die Größe ulMaxChannels x 4 Bytes haben.

Es können auch mehr Messkanäle abgefragt werden, als physikalisch vorhanden sind. Für die nicht verfügbaren Messkanäle wird der aktuelle Wert im Array dann nicht verändert.

Intern verwendete Funktionen der MscDLL

[MSC_ReadStatic](#)⁵¹

6.9 IRHL_StaticHardwareStatus_Start

Start des kontinuierlichen Datenaustauschs des Hardware-Status zwischen Gerät und DLL.

Definition

```
MSC_STATUS  
IRHL_StaticHardwareStatus_Start(void);
```

Rückgabewert

Die Funktion gibt im Erfolgsfall MSC_STATUS_SUCCESS zurück, bei Fehlern wird der [Fehlercode](#)²⁶ zurückgegeben.

Kommentare

Wenn eine Abfrage des Hardware-Status gewünscht / notwendig ist, so muss diese Funktion einmalig nach dem Verbindungsaufbau aufgerufen werden. Der HardwareStatus wird dann kontinuierlich zwischen Gerät und DLL ausgetauscht. Die Abfrage des aktuellen Hardware-Status erfolgt über die Funktion [IRHL_StaticHardwareStatus_ReadArray](#)¹⁴⁴.

Intern verwendete Funktionen der MscDLL

[MSC_SetupStaticChannel](#)⁴⁹

6.10 IRHL_StaticHardwareStatus_ReadArray

Auslesen des aktuellen Hardware-Status der einzelnen Messkanäle von der DLL durch die Applikation.

Definition

```
MSC_STATUS  
IRHL_StaticHardwareStatus_ReadArray(unsigned char  
auchHardwareStatus[], unsigned long ulMaxChannels);
```

Parameter

auchHardwareStatus

Array, in welches die aktuellen Hardware-Stati geschrieben werden sollen.

ulMaxChannels

Maximale Anzahl der in das Array auchHardwareStatus zu schreibenden Stati (entspricht der maximalen Anzahl der Messkanäle).

Rückgabewert

Die Funktion gibt im Erfolgsfall MSC_STATUS_SUCCESS zurück, bei Fehlern wird der [Fehlercode](#)¹²⁶ zurückgegeben.

Kommentare

Der kontinuierliche Austausch der Hardware-Stati muss zuvor mit [IRHL_StaticHardwareStatus_Start](#)¹⁴³ gestartet werden.

Das Array auchHardwareStatus muss mindestens die Größe ulMaxChannels Bytes haben.

Es können auch mehr Hardware-Stati abgefragt werden, als physikalische Messkanäle vorhanden sind. Für die nicht verfügbaren Messkanäle wird der aktuelle Wert im Array dann nicht verändert.

Intern verwendete Funktionen der MscDLL

[MSC_ReadStatic](#)⁵¹

6.11 IRHL_StaticBIO_Start

Start des kontinuierlichen Datenaustauschs der Bit I/O-Daten (digitale Ein-/Ausgänge) zwischen Gerät und DLL.

Definition

```
MSC_STATUS  
IRHL_StaticBIO_Start(unsigned long ulNIOBytes);
```

Parameter

ulNIOBytes

Anzahl der auszutauschenden Ein-/Ausgangsbytes. Z.B. 16, wenn jeweils 128 digitale Ein-/Ausgänge ausgetauscht werden sollen. Der Maximalwert ist 64.

Rückgabewert

Die Funktion gibt im Erfolgsfall `MSC_STATUS_SUCCESS` zurück, bei Fehlern wird einer der [Fehlercodes](#)²⁶ zurückgegeben.

Wenn die maximale Anzahl an I/O-Bytes überschritten wurde, dann wird `MSC_STATUS_INVALID_NIOBYTES` zurückgegeben.

Kommentare

Wenn der Austausch der Bit I/O-Daten gewünscht / notwendig ist, so muss diese Funktion einmalig nach dem Verbindungsaufbau aufgerufen werden. Die Bit I/O-Daten werden dann kontinuierlich zwischen Gerät und DLL ausgetauscht. Die Abfrage der aktuellen Eingangs-Daten erfolgt über die Funktion

[IRHL_StaticBIO_ReadInputs](#)¹⁴⁶. Das aktualisieren der Ausgangsdaten erfolgt über die Funktion [IRHL_StaticBIO_WriteOutputs](#)¹⁴⁷.

Die Anzahl der auszutauschenden Ein-/Ausgangs-Bites kann auch größer sein, als die Anzahl der physikalisch verfügbaren Ein-/Ausgänge. Für die meisten Applikationen reichen 16 Ein-/Ausgangs-Bytes.

Intern verwendete Funktionen der MscDLL

[MSC_SetupStaticChannel](#)¹⁴⁹

6.12 IRHL_StaticBIO_ReadInputs

Auslesen der aktuellen Bit I/O Eingangs-Daten.

Definition

```
MSC_STATUS  
IRHL_StaticBIO_ReadInputs(unsigned char aucInputBytes[],  
unsigned long ulMaxNoOfBytes);
```

Parameter

aucInputBytes

Array, in welches die aktuellen Eingangs-Daten geschrieben werden sollen.

ulMaxNoOfBytes

Maximale Anzahl der in das Array aucInputBytes zu schreibenden Eingangs-Bytes.

Rückgabewert

Die Funktion gibt im Erfolgsfall MSC_STATUS_SUCCESS zurück, bei Fehlern wird der [Fehlercode](#)²⁶ zurückgegeben.

Kommentare

Der kontinuierliche Austausch der Bit I/O-Daten muss zuvor mit [IRHL_StaticBIO_Start](#)¹⁴⁵ gestartet werden.

Das Array aucInputBytes muss mindestens die Größe ulMaxNoOfBytes Bytes haben.

Es können auch mehr Eingangs-Daten abgefragt werden, als physikalische Eingänge vorhanden sind.

Intern verwendete Funktionen der MscDLL

[MSC_ReadStatic](#)⁵¹

6.13 IRHL_StaticBIO_WriteOutputs

Schreiben der aktuellen Bit I/O Ausgangs-Daten.

Definition

```
MSC_STATUS  
IRHL_StaticBIO_WriteOutputs(unsigned char aucOutputBytes[],  
unsigned long ulNoOfBytes);
```

Parameter

aucOutputBytes

Array, aus welchem die aktuellen Ausgangs-Daten gelesen werden sollen.

ulNoOfBytes

Anzahl der im Array aucOutputBytes enthaltenen Ausgangs-Bytes.

Rückgabewert

Die Funktion gibt im Erfolgsfall `MSC_STATUS_SUCCESS` zurück, bei Fehlern wird der [Fehlercode](#)¹²⁶ zurückgegeben.

Kommentare

Der kontinuierliche Austausch der Bit I/O-Daten muss zuvor mit [IRHL_StaticBIO_Start](#)¹⁴⁵ gestartet werden.

Das Array `aucOutputBytes` muss mindestens die Größe `ulNoOfBytes` Bytes haben.

Es können auch mehr Ausgangs-Daten geschrieben werden, als physikalische Ausgänge vorhanden sind. Die "überflüssigen" Ausgangs-Bits haben dann keine Funktion.

Intern verwendete Funktionen der MscDLL

[MSC_RefreshChannel](#)⁵³

6.14 IRHL_DynReset

Diese Funktion setzt eine dynamische Messung zurück. Sie muss immer vor dem Start einer neuen dynamischen Messung aufgerufen werden.

Definition

```
MSC_STATUS  
IRHL_DynReset(unsigned long ulDynNo);
```

Parameter

ulDynNo

Nummer der dynamischen Messung (1 oder 2)

Rückgabewert

Die Funktion gibt im Erfolgsfall `MSC_STATUS_SUCCESS` zurück, bei Fehlern wird der [Fehlercode](#)^[26] zurückgegeben.

Kommentare

Der Ablauf zum Start einer dynamischen Messung ist wie folgt:

1. Dynamische Messung zurücksetzen mit `IRHL_DynReset`.
2. Messkanäle zur dynamischen Messung hinzufügen mit [IRHL_DynAddMeasChannel](#)^[149].
3. Dynamische Messung starten mit [IRHL_DynTimeStart](#)^[150] oder [IRHL_DynPosStart](#)^[152].

Intern verwendete Funktionen der MscDLL

[MSC_DetachSubChannelBuffers](#)^[57]

[MSC_WriteCommand](#)^[30]

(Bei zurücksetzen einer dynamischen Messung wird intern in der DLL die aktuelle [Kanalzuordnung](#)^[79] geladen.)

6.15 IRHL_DynAddMeasChannel

Diese Funktion fügt einen Messkanal zur dynamischen Messung hinzu.

Definition

```
MSC_STATUS  
IRHL_DynAddMeasChannel(unsigned long ulDynNo, unsigned long  
ulChannelNo);
```

Parameter

ulDynNo

Nummer der dynamischen Messung (1 oder 2)

ulChannelNo

Nummer des Messkanals, der hinzugefügt werden soll, beginnend bei 1.

Rückgabewert

Die Funktion gibt im Erfolgsfall MSC_STATUS_SUCCESS zurück, bei Fehlern wird der [Fehlercode](#)¹²⁶ zurückgegeben.

Kommentare

Diese Funktion muss nach dem Rücksetzen einer dynamischen Messung und vor dem Start einer neuen dynamischen Messung aufgerufen werden. Siehe dazu auch die Ablaufbeschreibung im Kommentar in [IRHL_DynReset](#)¹⁴⁸.

Intern verwendete Funktionen der MscDLL

Keine.

6.16 IRHL_DynTimeStart

Diese Funktion startet eine Zeit-gesteuerte dynamische Messung.

Definition

```
MSC_STATUS  
IRHL_DynTime_Start(unsigned long ulDynNo, unsigned long  
ulTimeUs, unsigned long ulNValues);
```

Parameter

ulDynNo

Nummer der dynamischen Messung (1 oder 2)

ulTimeUs

Abtastzeit (Trigger-Abstand) in μs , z.B. 100 für 10 kSample/s oder 1000 für 1 kSample/s

ulNValues

Anzahl der aufzuzeichnenden Messwerte je Kanal, z.B. 10000

Rückgabewert

Die Funktion gibt im Erfolgsfall MSC_STATUS_SUCCESS zurück, bei Fehlern wird der [Fehlercode](#)^[26] zurückgegeben.

Beispiele

Es soll eine dynamische Messung mit der Abtastrate 1000 Samples/s gestartet werden. Die Messung soll 8 Sekunden dauern. Verwenden Sie dazu folgende Parameter:

```
ulDynNo = 1
ulTimeUs = 1000
ulNValues = 8000
```

Es soll eine dynamische Messung mit der Abtastrate 10000 Samples/s gestartet werden. Die Messung soll 4,5 Sekunden dauern. Verwenden Sie dazu folgende Parameter:

```
ulDynNo = 1
ulTimeUs = 100
ulNValues = 45000
```

Kommentare

Vor dem Start der dynamischen Messung muss diese [zurückgesetzt](#)^[148] sowie die [beteiligten Kanäle parametrier](#)^[149] werden. Siehe dazu auch die Ablaufbeschreibung im Kommentar von [IRHL_DynReset](#)^[148].

Die Abtastzeit (Trigger-Abstand) muss ein Vielfaches der Abtastzeit der beteiligten Irinos-Boxen sein, mindestens jedoch $100\mu\text{s}$. Bei den meisten Irinos-Boxen ist die interne Abtastzeit $50\mu\text{s}$. Zulässige Werte wären also beispielsweise 100, 250, 850, 1000, 1500, 12000, usw.

Jede Irinos-Box hält intern Speicher für 100.000 dynamische Messwerte je Kanal bereit.

Intern verwendete Funktionen der MscDLL

[MSC_WriteCommand](#)³⁰

[MSC_DetachSubChannelBuffers](#)⁵⁷

[MSC_SetupExtendedDynamicChannel](#)⁵⁴

[MSC_AttachSubChannelBuffer](#)⁵⁶

6.17 IRHL_DynPosStart

Diese Funktion startet eine Positions-gesteuerte dynamische Messung.

Definition

```
MSC_STATUS  
IRHL_DynPos_Start(unsigned long ulDynNo, unsigned long  
ulReserved, unsigned long ulTrgChannel, signed long slStart,  
signed long slDistance, unsigned long ulNValues);
```

Parameter

ulDynNo

Nummer der dynamischen Messung (1 oder 2)

ulReserved

Platzhalter für zukünftige Verwendung. Übergeben Sie hier immer den Wert 0.

ulTrgChannel

Nummer des Trigger-Kanals, aus dem die Positions-Information gewonnen wird (beginnend bei 1).

slStart

Messwert, ab dem die dynamische Messung gestartet werden soll.

slDistance

Positions-Abstand zwischen 2 Messpunkten.

ulNValues

Anzahl der aufzuzeichnenden Messwerte je Kanal, z.B. 10000

Rückgabewert

Die Funktion gibt im Erfolgsfall MSC_STATUS_SUCCESS zurück, bei Fehlern wird der [Fehlercode](#)¹²⁶ zurückgegeben.

Beispiele

Es wird ein Drehgeber mit 3600 Inkrementen / Umdrehung als Positions-Trigger verwendet. Der Inkrementalgeber ist an Messkanal 9 angeschlossen. Ab der 0-Position sollen in 1°-Schritten Messwerte aufgezeichnet werden. Verwenden Sie dazu die folgenden

Parameter:

```
ulDynNo = 1
ulReserved = 0
ulTrgChannel = 9
slStart = 0
slDistance = 10
ulNValues = 360
```

Es wird ein 1Vss-Glasmaßstab mit einer Auflösung von 2500 Inkremente 10mm verwendet. Der Glasmaßstab ist an Messkanal 17 angeschlossen. Durch die 200-fache 1Vss-Interpolation entspricht dies 2500*200=500000 Inkremente. Im Bereich -10mm bis -80mm sollen in 0,1mm-Schritten Messwerte aufgezeichnet werden. Verwenden Sie dazu die folgenden Parameter:

```
ulDynNo = 1
ulReserved = 0
ulTrgChannel = 17
slStart = -500000
slDistance = -5000
ulNValue = 700
```

Kommentare

Vor dem Start der dynamischen Messung muss diese [zurückgesetzt](#)^[148] sowie die [beteiligten Kanäle parametrier](#)^[149] werden. Siehe dazu auch die Ablaufbeschreibung im Kommentar von [IRHL_DynReset](#)^[148].

Jede Irinos-Box hält intern Speicher für 100.000 dynamische Messwerte je Kanal bereit.

Intern verwendete Funktionen der MscDLL

[MSC_WriteCommand](#)^[30]

[MSC_DetachSubChannelBuffers](#)^[57]

[MSC_SetupExtendedDynamicChannel](#)^[54]

[MSC_AttachSubChannelBuffer](#)^[56]

6.18 IRHL_DynStop

Diese Funktion stoppt eine dynamische Messung vorzeitig, d.h. bevor die parametrierte maximale Anzahl an Messwerten aufgezeichnet wurde.

Definition

```
MSC_STATUS  
IRHL_DynStop(unsigned long ulDynNo);
```

Parameter

ulDynNo

Nummer der dynamischen Messung (1 oder 2)

Rückgabewert

Die Funktion gibt im Erfolgsfall MSC_STATUS_SUCCESS zurück, bei Fehlern wird der [Fehlercode](#)^[26] zurückgegeben.

Kommentare

Zum Stoppen wird der Trigger der dynamischen Messung deaktiviert.

Intern verwendete Funktionen der MscDLL

[MSC_WriteCommand](#)^{□30}

6.19 IRHL_DynGetValuesAvailable

Über diese Funktion kann die Anzahl der bereits verfügbaren dynamischen Messwerte abgefragt werden.

Definition

```
MSC_STATUS  
IRHL_DynGetValuesAvailable(unsigned long ulDynNo, unsigned long  
*pulNValues);
```

Parameter

ulDynNo

Nummer der dynamischen Messung (1 oder 2)

pulNValues

Rückgabe der Anzahl der bereits verfügbaren dynamischen Messwerte (je Messkanal).

Rückgabewert

Die Funktion gibt im Erfolgsfall MSC_STATUS_SUCCESS zurück, bei Fehlern wird der [Fehlercode](#)^{□26} zurückgegeben.

Kommentare

-

Intern verwendete Funktionen der MscDLL

[MSC_GetPosition](#)⁵⁹

6.20 IRHL_DynReadValueArray

Über diese Funktion können die dynamischen Messwerte als Array ausgelesen werden.

Definition

```
MSC_STATUS  
IRHL_DynReadValueArray(unsigned long ulDynNo, signed long  
aslValues[], unsigned long ulMaxValues, unsigned long  
ulIdxFirstValue, unsigned long ulChannelNo, unsigned long  
*pulValuesCopied);
```

Parameter

ulDynNo

Nummer der dynamischen Messung (1 oder 2)

aslValues

Array, in welches die Messwerte geschrieben werden sollen.

ulMaxValues

Größe des Arrays aslValues (Anzahl der Array-Elemente, NICHT Anzahl der Bytes)

ulIdxFirstValue

Index des ersten Wertes, der kopiert werden soll (beginnend bei 0). 0 kopiert alle Werte.

ulChannelNo

Nummer des Messkanals, dessen dynamische Messwerte kopiert werden sollen.

pulValuesCopied

Anzahl der tatsächlich kopierten Messwerte.

Rückgabewert

Die Funktion gibt im Erfolgsfall MSC_STATUS_SUCCESS zurück, bei Fehlern wird der [Fehlercode](#)¹²⁶ zurückgegeben.

Kommentare

Diese Funktion muss für jeden Messkanal der dynamischen Messung separat aufgerufen werden. Wenn beispielsweise die Messkanäle 3, 4 und 9 in der dynamischen Messung verwendet werden, so muss diese Funktion 3x aufgerufen werden. Beim ersten Aufruf ist der Parameter ulChannelNo 3, beim zweiten Aufruf 4 und beim dritten Aufruf 9.

Es ist möglich die Messwerte auch blockweise auszulesen. So können während einer noch aktiven dynamischen Messung die bereits verfügbaren dynamischen Messwerte ausgelesen und sofort angezeigt werden, während weitere Messwerte noch folgen werden. Relevant hierfür sind die Parameter ulIdxFirstValue sowie pulValuesCopied. In folgendem Pseudo-Code werden die Messwerte blockweise in einer Timer-Routine ausgelesen:

```
unsigned long ulValuesCopied = 0; // Diese Variable beim Start einer dynamischen
signed long aslValues[1000];

void TimerEvent(void)
{
    unsigned long ulTemp = 0;
    IRHL_DynReadValueArray(1, aslValues, 1000, ulValuesCopied, 3, &ulTemp);
    ulValuesCopied = ulValuesCopied + ulTemp;
    // Update GUI
}
```

Intern verwendete Funktionen der MscDLL

-

- 1 -

1Vss 76, 99

- 2 -

24VOvld 102

- A -

Absolut-Zeit 107
 Aktiviere Kanal-Liste 89
 AmpErr 102
 Anzahl Irinos-Boxen 67

- B -

Benachrichtigungen 42
 Beschränkung der Kanal-Zahl 85
 Bit I/O 115
 Box-Informationen 68
 Box-Übersicht 67

- C -

Callback-Funktion 47

- D -

Datum 107
 Definiere Trigger 91
 Diagnose-Speicher 107
 digitale Ausgänge 115
 digitale Eingänge 115
 digitales Typenschild 68
 DisconnectTimeout 37
 dynamische Messung 96, 120, 125, 127
 dynamische Messwerte 127
 dynamischen Messung 85, 97

- E -

Ereignis 106, 110, 112
 Event 45

- F -

Fast 102

Fehlercode 26

Füllstand des Puffers für dynamische Messwerte 59

- G -

GComp 102
 Geräte-Bezeichnung 68

- H -

Handle 34
 Hardware-Status 102
 HighLevel-DLL 16, 132

- I -

Inkrementalgeber 99
 IRHL_Connect 136
 IRHL_Disconnect 137
 IRHL_DynAddMeasChannel 149
 IRHL_DynGetValuesAvailable 155
 IRHL_DynPosStart 152
 IRHL_DynReadValueArray 156
 IRHL_DynReset 148
 IRHL_DynStop 154
 IRHL_DynTimeStart 150
 IRHL_GetCommStatus 138
 IRHL_GetCurrentHandle 140
 IRHL_GetVersionHL 134
 IRHL_StaticBIO_ReadInputs 146
 IRHL_StaticBIO_Start 145
 IRHL_StaticBIO_WriteOutputs 147
 IRHL_StaticHardwareStatus_ReadArray 144
 IRHL_StaticHardwareStatus_Start 143
 IRHL_StaticMeasurement_ReadArray 142
 IRHL_StaticMeasurement_Start 141
 IRHL_WriteCommand 139

- K -

Kanalbezeichnung 82
 Kanal-Eigenschaften 76
 Kanal-Liste 85, 87, 89, 125
 Kanal-Zuordnung 79, 82
 Kommunikationsstatus 40

- M -

MAC-Adresse 68
 Msc.cfg 20

MSC_AttachSubChannelBuffer 56
 MSC_CloseDevice 36
 MSC_DetachSubChannelBuffers 57
 MSC_EnumerateDevices 32
 MSC_GetDeviceInfo 33
 MSC_GetDeviceState 40
 MSC_GetPosition 59
 MSC_GetVersion 29
 MSC_InitDevice 35
 MSC_MAX_UNIQUEID_SIZE 33
 MSC_OpenDevice 34
 MSC_ReadStatic 51
 MSC_RefreshChannel 53
 MSC_RESET_DISCARDED_COUNTERS 40
 MSC_RESET_ERROR_COUNTERS 40
 MSC_SetNotificationCallback 47
 MSC_SetNotificationEvent 45
 MSC_SetNotificationMessage 43
 MSC_SetupExtendedDynamicChannel 54
 MSC_SetupStaticChannel 49
 MSC_Start 37
 MSC_STATUS_ALREADY_INITIALIZED 26
 MSC_STATUS_BUFFER_TO_SHORT 26
 MSC_STATUS_FAILED 26
 MSC_STATUS_FUNCTION_NOT_ALLOWED 26
 MSC_STATUS_INVALID_CHANNEL_TYPE 26
 MSC_STATUS_INVALID_HANDLE 26
 MSC_STATUS_INVALID_OBJECT_TYPE 26
 MSC_STATUS_INVALID_PARAMS 26
 MSC_STATUS_NO_DATA_AVAILABLE 26
 MSC_STATUS_NO_DEVICES 26
 MSC_STATUS_NO_MORE_DATA 26
 MSC_STATUS_NO_RESOURCES 26
 MSC_STATUS_NOT_INITIALIZED 26
 MSC_STATUS_SUCCESS 26
 MSC_Stop 39
 MSC_TYPE_SOCKET_XPORT 33
 MSC_TYPE_USB_FTDI 33
 MSC_WriteCommand 30

- O -

OComp 102
 opcACL 89
 opcAT 96
 opcBIO 115
 opcBIORO 118
 opcClrEv 112
 opcDDMx 125
 opcDT 91
 opcIT 97

Opcodes 62
 opcRCA 79
 opcRCL 87
 opcRDMx 127
 opcREv 106
 opcRHS 102
 opcRIV 67
 opcRMI 68
 opcRS 114
 opcRST 129
 opcRSW 120
 opcSAbtT 107
 opcSP 99
 opcWCA 82
 opcWCC 76
 opcWCL 85
 opcWEvCfg 110
 Oszillator-Kurzschluss 102

- P -

Position vorgeben 99
 Positions-Trigger 91
 PwrOvld 102

- R -

Referenzmarke 99
 Refmark 99, 102
 REFOFF 99
 REFON 99
 Reset 129
 ResponseTimeout 37
 RS422 76

- S -

Sample-Periode 68
 Segmentindex 79
 Sende-Periode 37
 Serien-Nr 68
 Set Parameter 99
 ShortCirc 102
 statische Messwerte 114
 Statuswort 120
 String 66

- T -

Trigger 91, 96, 97, 120

TTL 76

- U -

Uhrzeit 107

- V -

Vector 102

Version der DLL 29

Verstärkungs- und Offset-Regelung 99

VRefOvld 102

- W -

Windows Messages 43

WM_MESSAGE_MSC_BITIO 43

WM_MESSAGE_MSC_HW_STATUS 43

WM_MESSAGE_MSC_READSTATIC 43

- Z -

Zeit-Trigger 91