

Das Telegramm / Funktionen der MC

TRANSFER

Dieser grundlegende Befehl wird verwendet, wenn ein Modul einen Datenstring zu einem anderen Modul übertragen will. Bei dem Empfängermodul wird nur <data> empfangen.

TRANSFER | <modul ename> | <data>

<data> wird an Modul gesendet. <data> darf weitere Trennzeichen („|“) enthalten. Um ein Modul zu erreichen, das nicht an derselben MC angeschlossen ist (MultiMCMMode), muss beim <modul ename> auch der Name der MC, abgetrennt mit einem „-“, angegeben werden, z.B.:

TRANSFER | mcname->modul ename | datendaten

Antwort der MC: **ACK | TRANSFER | OK, ACK | TRANSFER | NOTOK** oder **ACK | TRANSFER | DELAYED**.

Letzteres wird gesendet, wenn es sich bei dem Verbindungsart des Moduls um eine lose Verbindung handelt und diese gerade unterbrochen ist, oder wenn sich das Modul auf einer externen MC befindet (im MultiMCMMode) und momentan keine Verbindung vorhanden ist.

BROADCAST

Sollen die Daten an mehrere Module gesendet werden, kann man den Broadcast-Befehl verwenden. Die Empfänger lassen sich einschränken, indem man den Typ der Empfängermodule angibt. Übergibt man beispielsweise **HARDWARE**, so empfangen nur diese Module die Daten, die sich beim Login mit eben jenem Typ angemeldet haben. Gibt man stattdessen **ALL** an, so werden die Daten an alle Module gesendet.

BROADCAST | <modul etype> | <data>

<data> wird an alle übrigen Module vom Typ <modul etype> derselben MC gesendet.

Wird **ALL** als <modul etype> angegeben, so werden alle Module benachrichtigt.

Antwort der MC: **ACK | BROADCAST | OK**.

WIDEBROADCAST

Dieser Befehl ist vergleichbar mit dem vorigen Befehl mit dem Unterschied, dass nicht nur benachbarte sondern auch entfernte Module benachrichtigt werden. Hier kann ebenfalls ein einschränkender Modul-Typ angegeben werden.

WIDEBROADCAST | <modul etype> | <data>

<data> wird an alle übrigen Module vom Typ <modul etype>, auch an die anderer MCs, gesendet (MultiMCMMode).

Wird **ALL** als <modul etype> angegeben, so werden alle Module benachrichtigt.

Antwort der MC: **ACK | WIDEBROADCAST | OK**.

ALERT

Module sind zwar normale Programme, allerdings ist es vom Programmierer nicht immer erwünscht, eine Benutzeroberfläche zu programmieren. Schließlich läuft das ganze System selbstständig und benötigt eigentlich keine Eingriffe eines Benutzers. Trotzdem kann es manchmal hilfreich sein, Meldungen auszugeben, wenn beispielsweise Fehler auftreten. Mit dem Alert-Befehl kann ein Modul so eine Meldung senden. Die MC leitet diese dann an den Benutzer weiter.

ALERT | <label> | <description>

Mit Alert werden Daten an die MC gesendet. Hiermit kann momentan entweder ein Hinweis oder ein Error gemeldet werden:

ALERT | ERROR | <description>

ALERT | HI NT | <description>

Die MC leitet die Meldung an das eventuell angeschlossene GUI-Modul weiter:

MSG | <label> | <description>

So können Module, die selber keine Benutzeroberfläche besitzen, den Benutzer erreichen.

Antwort der MC: **ACK|ALERT|OK** oder **ACK|ALERT|NOTOK**.

GETALLMODULES, EXISTSMODULE

Ein Modul kann mit diesem Befehl abfragen, welche Module an der angegebenen MC eingeloggt sind. Sinnvoll ist dies, wenn ein Modul mit einem anderen Modul zusammenarbeiten soll und deshalb überprüfen will, ob dieses auch vorhanden ist. Dazu kann auch der nächste Befehl verwendet werden.

GETALLMODULES|<mcname>

Fragt Liste der angeschlossenen Module der angegebenen MC ab.

Antwort der MC: **LOADEDMODULE|<mcname>|<modul ename>|<modul etype>|<connecti ontype>**

(Die Antwort wird für alle vorhandenen Module jeweils einmal gesendet.) Wurden alle Module übertragen, so wird das Ende der Übermittlung mit einem **ACK|GETALLMODULES|OK** gekennzeichnet.

EXI STSMODULE|<modul ename>

Gibt zurück, ob das angegebene Modul existiert.

Antwort der MC: **EXI STSMODULE|<modul ename>|YES** im positiven Fall, sonst:
EXI STSMODULE|<modul ename>|NO.

WATCHCOM

Mit diesem Befehl kann ein Modul die Kommunikation eines anderen Moduls mit der MC überwachen. Bei der Fehlersuche kann dies hilfreich sein, für andere Zwecke sollte Watchcom jedoch nicht verwendet werden, da er viel Netzwerk-Traffic erzeugt.

WATCHCOM|<observedmodul ename>

Die MC richtet bei diesem Befehl einen Listener ein, der das Module bei allen Kommunikationen zwischen derselben MC und einem bestimmten Modul benachrichtigt.

Meldung der MC: **RESULTCOM|<observedmodul ename>|<di recti on>|<data>**

Werte für <di recti on>:

- >> [MC sendet an Modul]
- << [Modul sendet an MC]

Es können durchaus auch auf noch nicht existierende Module Listener gesetzt werden, um Probleme zu vermeiden, die durch Erstellen eines Listeners vor der **READY|**-Benachrichtigung auftreten können, wenn das zu beobachtende Modul zu diesem Zeitpunkt noch nicht angemeldet ist.

Antwort der MC: **ACK|WATCHCOM|OK**.

DELCOMWATCHER|<observedmodul ename>

Löscht den Ereignis-Listener für ein beobachtetes Modul. Es werden dann keine Benachrichtigungen mehr an das überwachende Modul gesendet.

Antwort der MC: **ACK|DELCOMWATCHER|OK**.

ADD

erstellt eine neue Variable. Neben dem Namen wird angegeben, ob es sich um eine lokale oder öffentliche Variable handelt. Lokale Variablen können nur von dem Modul geschrieben werden, das sie auch erstellt hat. Öffentliche können hingegen von allen Modulen geschrieben werden. Im MultiMCMMode werden nur Variablen des letzten Typs synchronisiert.

ADD|<varname>|<varaccess>

Variable wird auf MC erstellt.

Werte für <varaccess>:

- 1 [system variable]
- 2 [local]
- 3 [public]

Im MultiMCMMode werden nur Variablen vom Typ 3 beachtet.

Antwort der MC: ACK|ADD|OK oder ACK|ADD|NOTOK.

ADDSTRUCTURE

Nicht immer lässt sich alles in normalen Variablen speichern. Mit diesem Befehl kann eine ganze Variablenstruktur angelegt werden.

ADDSTRUCTURE | <varname> | <varaccess> | <definition>
Variable mit angegebener Struktur wird erstellt (Strukturvariable).

Die Namen der Untervariablen werden mit [und] eingeschlossen. Um mehrere Variablen auf gleicher Ebene zu erstellen, werden sie mit Kommata abgetrennt. Ohne Komma wird die rechte Variable der linken untergeordnet. Die eckigen Klammern werden auch zum Gruppieren verwendet:

Beispiele für <definition>; <varname> ist Var:

- [A], [B] Var.A und Var.B
- [A][B] Var.A.B
- [[A], [B]][[C], [D]] Var.A.C, Var.A.D, Var.B.C und Var.B.D
- [A], [B(5)] Var.A, Var.B[0], Var.B[1], Var.B[2], Var.B[3], Var.B[4]

Antwort der MC: **ACK** | **ADDSTRUCTURE** | **OK** oder **ACK** | **ADDSTRUCTURE** | **NOTOK**.

SET

SET | <varname> | <varvalue>

Wert einer Variable wird auf MC geschrieben.

Antwort der MC: **ACK** | **SET** | **OK** oder **ACK** | **SET** | **NOTOK**.

GET

GET | <varname> | <varaccess>

Fragt den Wert einer Variablen ab.

Antwort der MC: **RESULT** | <varname> | <varvalue> | <varaccess> | <varcreator>
Der Wert <varcreator> gibt den Namen des Moduls an, das die Variable erstellt hat.

Bei einem Fehler wird **ACK** | **GET** | **NOTOK** zurückgegeben.

EXISTS

EXISTS | <varname>

Gibt zurück, ob die angegebene Variable existiert.

Antwort der MC: **EXISTS** | **YES**, **EXISTS** | **NO** oder **ACK** | **EXISTS** | **NOTOK**.

DEL

Sofern ein Modul Schreibzugriff auf eine Variable hat, kann es diese auch löschen.

DEL | <varname>

Löscht die angegebene Variable. Es können nur Variablen der obersten Ebene gelöscht werden. Untervariablen, die mit **ADDSTRUCTURE** | erstellt wurden, können in dieser Struktur nicht einzeln gelöscht werden, was einen hohen Programmieraufwand erfordern würde.

Antwort der MC: **ACK** | **DEL** | **OK** oder **ACK** | **DEL** | **NOTOK**.

GETVARS

GETVARS |

Fordert MC auf, alle vorhandenen Variablen zurückzugeben.

Antwort der MC: **RESULT** | <varname> | <varvalue> | <varaccess> | <varcreator>

(Die Antwort wird für alle vorhandenen Variablen jeweils einmal gesendet.) Wurden alle Variablen übertragen, so wird das Ende der Übermittlung mit einem **ACK** | **GETVARS** | **OK** gekennzeichnet.

WATCH

Neben dem GET-Befehl hat ein Modul auch die Möglichkeit, eine Variable zu überwachen. Die MC sendet automatisch den Variablenwert an das Modul, wenn sich dieser geändert hat.

WATCH | <varname> | <mini megap>

Bei diesem Befehl richtet die MC einen Ereignis-Listener ein, der das Modul benachrichtigt, wenn die angegebene Public Variable sich ändert. In diesem Fall sendet sie an das Modul:

RESULT | <varname> | <variable> | <varaccess>.

Falls eine Variable angegeben wird, die noch weitere Untervariablen enthält (erstellt mit **ADDSTRUCTURE** | . .), dann wird der Listener für alle diese Untervariablen aktiviert.

Wenn **ALL** als <varname> übergeben wird, so wird ein Listener für alle Variablen eingerichtet. Dieser benachrichtigt auch über neu erstellte Variablen. Wurde eine Strukturvariable angelegt, so wird wie bei einer normalen Variable nur einmal **RESULT** | . . (siehe **GETVARS** |) gesendet (und nicht auch noch für jede Untervariable). Dabei wird an der Stelle von <variable> [...] gesendet.

Wurde eine beobachtete Variable gelöscht, so wird als <variable> [] gesendet.

Es können durchaus auch auf noch nicht existierende Variablen Listener gesetzt werden.

Über den Parameter <mini megap> kann angegeben werden, wie oft der Listener das Modul benachrichtigen soll. Wird als Wert beispielsweise 1000 angegeben, so bedeutet dies, dass, auch wenn sich der Variablenwert alle 10ms ändert, nur jede Sekunde eine Benachrichtigung abgeschickt wird. Alle Wertänderungen innerhalb dieser Zeitlücke werden ignoriert. Damit kann das Netz entlastet werden, wenn nicht jede Änderung von Bedeutung ist.

Antwort der MC: **ACK** | **WATCH** | **OK** oder **ACK** | **WATCH** | **NOTOK**.

DELWATCHER | <varname>

Löscht den Ereignis-Listener für eine Variable. Es werden dann keine Benachrichtigungen mehr an das Modul gesendet.

Antwort der MC: **ACK** | **DELWATCHER** | **OK**.

WRITEACCESS

WRITEACCESS | <varname>

Gibt zurück, ob das Modul, das diese Anfrage sendet, Schreibzugriff auf die Variable hat. (siehe **ADD** | . . bzw. **ADDSTRUCTURE** | . .).

Antwort der MC: **WRITEACCESS** | <varname> | **YES**, **WRITEACCESS** | <varname> | **NO** oder **ACK** | **WRITEACCESS** | **NOTOK**.

GETCONNECTEDMCS

Mit diesem Befehl kann ein Modul abfragen, mit welchen anderen MCs momentan eine Verbindung besteht.

GETCONNECTEDMCS |

Fordert MC auf, die Daten aller ihr bekannten MCs zurückzugeben.

Antwort der MC: **CONNECTED** | <mcname> | <address> | <connected>

(Die Antwort wird für alle bekannten MCs jeweils einmal gesendet.) Wurden alle Variablen übertragen, so wird das Ende der Übermittlung mit einem **ACK** | **GETCONNECTEDMCS** | **OK** gekennzeichnet.

Der Wert <address> setzt sich aus der IP-Adresse und dem Empfangsport der MC zusammen. Weiteres dazu weiter unten.

<connected> gibt an, ob eine Verbindung zu der MCs besteht oder nicht.