| VERSIONSÄNDERUNG | | |
|---|---|---|
| NR. | AUSGABE | BESTÄTIGT |
| 1 | V 1.8 | 2008-05-26 |

# Software-Beschreibung

# MTCS-C2-DLL
# MTCS-ME1-DLL

## Beschreibung der Bibliothek (MTCSApi.dll)
## API Programmierinterface DLL

Revision 2.43

Inhaltsverzeichnis



**MAZeT GmbH** Vertrieb
Göschwitzer Straße 32
07745 Jena
Tel.: +49 3641 2809-0
Fax: +49 3641 2809-12
E-Mail: sales@MAZeT.de
Url: http://www.MAZeT.de

# 1 Einleitung

Das Dokument beschreibt die MTCSApi.dll als Schnittstelle zwischen der Firmware auf dem Evaluation Kit MTCS-ME1 (Mod EVA) oder der Firmware auf dem Board MTCS-C2 (Colorimeter2)und der Bedienoberfläche(Host). Voraussetzung für die Lauffähigkeit der Software ist eine USB 2.0 Schnittstelle. Die Software wurde in der Programmiersprache „C“ für WindowsXP© geschrieben und unter LAB Windows© und Microsoft C 6.0 getestet. Zu beachten sind die Version der Firmware des Mod EVA (>V2.22) bzw. des Colorimeter2 (V0.13) und die Versionsnummer der DLL (> V2.43). Die Kommandos für das Colorimeter2 funktionieren erst ab DLL-Version 2.40. Zum besseren Verständnis der Funktionen der Hardware wird auf das Datenblatt MTCS-ME1 bzw. MTCS-C2 verwiesen. Grundprinzip ist, dass der Host seine Anforderung als Kommando an das Mod EVA sendet. Nach der Bearbeitung des Kommandos sendet das Mod EVA seine Antwort an den Host. Die Initiative geht also immer vom Host aus und es wird immer auf die Antwort gewartet, die nach der Ausführung des Kommandos geschickt wird. Die MTSCApi.dll bedient die USB-Schnittstelle oder die serielle RS232 Schnittstelle (sofern diese bestückt ist). Im Auslieferungszustand steht auf Adresse 0x3fe im EEPROM der Wert 0xff, und das Board wird über USB bedient. Wird diese Adresse mit 0 belegt, dann wird die serielle Schnittstelle im Modus 57600,8,n,1 benutzt (sofern diese bestückt ist). Die dll sollte sich in der Directory des ausführbaren Programms befinden. Die für die Programmentwicklung notwendige lib in der Directory des entsprechenden Projektes.

*#define USB_DLL_API __declspec(dllexport) __stdcall*

# 2 MTCS – FUNKTIONEN

## 2.1 MTCS – Funktionen für MTCS-ME1 und MTCS-C2

### 2.1.1 MTCSInitSystem

**Initialisierung des Systems**

*Definition*
int USB_DLL_API MTCSInitSystem(char cTyp, int iVendorID, int iProductID);

*Parameter*

| | | |
|---|---|---|
| char cTyp | Typ der Verbindung des Boards mit der Bedienoberfläche | |
| | cTyp = 0 | USB<br>an Adresse 0x3fe im EEPROM muss 0xff stehen |
| | cTyp = 1 | Serielle Schnittstelle<br>an Adresse 0x3fe im EEPROM muss 0 stehen |
| int iVendorID | = 0x400 | Vendor- und Produkt-ID für Mod EVA |
| int iProductID | = 0xc35d | |
| int iVendorID | = 0x152a | Vendor- und Produkt-ID für Colorimeter2 |
| int iProductID | = 0x8220 | |

*Rückgabewert*

| | |
|---|---|
| ≥ 0 | fehlerfrei, Anzahl der gefunden Geräte |
| = -1 | interner Initialisierungsfehler, reset Mod EVA |
| = -2 | Devicenotifikation fehlerhaft |
| = -4 | dieses Mod EVA wurde bereits konfiguriert |
| = -5 | Gerät nicht gefunden |

### 2.1.2 MTCSCloseDevice

**Schließt das Gerät.**

*Definition*
int USB_DLL_API MTCSCloseDevice( void);

*Rückgabewert*

| | |
|---|---|
| = 0 | fehlerfrei |
| != 0 | Gerät kann nicht geschlossen werden. |

*Anmerkung*
Es ist keine Kommunikation über USB mehr möglich.

### 2.1.3 MTCSDllGetVersion

**Die DLL-Version wird gelesen.**

*Definition*
void USB_DLL_API MTCSDllGetVersion(char* cBuf);

*Parameter*

| | |
|---|---|
| char* cBuf | Pointer auf den Puffer, der die DLL–Version enthält,<br>Puffergröße für die Antwort vom Mod EVA : 5 Byte |

### 2.1.4 MTCSReadVersion

**Die aktuelle Versionsnummer der Firmware wird geholt**.

*Definition*
int USB_DLL_API MTCSReadVersion (char* cBuf);

*Parameter*

| | |
|---|---|
| char* cBuf | Pointer auf den Puffer, der die Firmware–Version enthält,<br>z.B. 0x32, 0x2e, 0x34, 0x30 für Versionsnummer 2.40<br>Puffergröße für die Antwort vom Mod EVA : 5 Byte, das letzte Byte ist 0. |

*Rückgabewert*

| | |
|---|---|
| = 0 | fehlerfrei |
| = 1 | Parameterfehler 1 |
| = 2 | Firmware erkannte illegales Kommando |
| = 7 | Übertragungsfehler |

*Anmerkung*
Mit dem Fehlercode wird der Status der Firmware-Initialisierung nach dem Anschluss über USB zurückgegeben.
Dieses Kommando kann von der Anwendung auf der Host-Seite dazu verwendet werden, die Kommunikation über USB zu testen.

### 2.1.5 MTCSReadMemory

**iAnz Integerwerte des Inhaltes des EEPROM wird byteweise fortlaufend ab Adresse 0 gelesen.**

*Definition*
int USB_DLL_API MTCSReadMemory (unsigned char* cBuf, int iAnz);

*Parameter*

| | |
|---|---|
| unsigned char* cBuf | Pointer auf den Puffer, der den Inhalt des EEPROM enthält<br>Puffergröße für die Antwort vom Mod EVA : iAnz*2 Byte |
| int iAnz | Anzahl der Integerwerte, die gelesen werden sollen.<br>(Param1 im Beispiel) |

*Rückgabewert*

| | |
|---|---|
| = 0 | fehlerfrei |
| = 1 | Parameterfehler |
| = 2 | Firmware erkannte illegales Kommando |
| = 6 | Anzahl der Integerwerte überschreitet Gesamtanzahl für EEProm |
| = 7 | Übertragungsfehler |

*Anmerkung*
Auf dem EEProm befinden sich z.B. alle Daten der Korrekturwerte für die Sensorkalibrierung. Genauere Angaben stehen unter 4. EEProm.

### 2.1.6 MTCSWriteMemory

**iAnz Integerwerte des Inhaltes des EEPROM wird immer byteweise fortlaufend ab Adresse 0 geschrieben.**

*Definition*
int USB_DLL_API MTCSWriteMemory (unsigned char* cBuf, int iAnz);

*Parameter*

| | |
|---|---|
| unsigned char* cBuf | Pointer auf den Puffer, der den Inhalt für den EEPROM enthält<br>(Param2 im Beispiel) |
| int iAnz | Anzahl der Integerwerte, die geschrieben werden sollen.<br>(Param1 im Beispiel) |

*Rückgabewert*

| | |
|---|---|
| = 0 | fehlerfrei |
| = 2 | Firmware erkannte illegales Kommando |
| = 4..x | Parameterfehler Parameter 1, oder.. Parameter x |
| = 6 | Anzahl der Integerwerte überschreitet Gesamtanzahl für EEProm |
| = 7 | Übertragungsfehler |

### 2.1.7 MTCSSetShift

Diese Funktion verschiebt auf Bitebene die ADC Werte um den Wert iShift.

*Definition*
int USB_DLL_API MTCSSetShift (int iIndex, unsigned int iShift);

| | | |
|---|---|---|
| int iIndex | = 0 | reserviert für Devicenumber |
| unsigned int iShift | = 0….6 | |

*Rückgabewert*

| | |
|---|---|
| = 0 | fehlerfrei |
| !=0 | Verbindungsfehler |

*Anmerkung*
Es ergibt sich für den ADC Wert mit shift folgende Formel ADC*$2^x$. Dabei ist jedoch zu beachten, dass die Bitverschiebung das Rauschen ebenfalls verstärkt.

### 2.1.8 MTCSSetUpdateMode

**Versetzt die Firmware in den Update-Modus.**

*Definition*
int USB_DLL_API MTCSSetUpdateMode(void);

*Rückgabewert*

| | |
|---|---|
| = 2 | Firmware erkannte illegales Kommando |
| = 4..x | Parameterfehler Parameter 1, oder.. Parameter x |

Andere Werte sind nicht möglich, da die USB-Schnittstelle nicht mehr bedient wird.

Nach PowerUp-Reset kann mit Hilfe der "Flip-Software(ATMEL)" über die USB-Schnittstelle die Firmware auf einen neuen Stand gebracht werden. Ein erneutes PowerUp-Reset hebt den Setup-Modus wieder auf

## 2.2 MTCS – Funktionen für MTCS-ME1 TOP, FRONT und DARK TIA

### 2.2.1 MTCSGetADCxx

Diese Kommandos werden für die Messung reflektierender Proben oder Quellen mit konstanter Helligkeit eingesetzt.
Die Helligkeit der LED-Beleuchtung (nur bei MTCS-ME1 FRONT und TOP) muss mit MTCSSetEPoti eingestellt werden.
„iCounts“ legt die Anzahl der Messwerte fest, über die eine Mittelwertbildung erfolgt.

### 2.2.2 MTCSGetADCBL

**Die fremdlichtkompensierten Messung**

*Definition*
int USB_DLL_API MTCSGetADCBL (unsigned short * usBuf, int iCounts);

*Parameter*

| | |
|---|---|
| int iCounts | Anzahl der Messwerte für die Mittelwertbildung (Param1 im Beispiel) |
| unsigned short * usBuf | Pointer auf den Puffer, der die ADC - Werte enthält<br>Puffergröße für die Antwort vom Mod EVA : 8 Byte |

*Rückgabewert*

| | |
|---|---|
| = 0 | fehlerfrei |
| = 2 | Firmware erkannte illegales Kommando |
| = 4..x | Parameterfehler Parameter1, oder… Parameterx |
| = 7 | Übertragungsfehler |

*Anmerkung*
Fremdlichtkompensiert bedeutet, es wird im beleuchteten und unbeleuchteten Zustand gemessen und die Differenz gebildet.
ADC-Mittelwerte UTD (AD-Wert der Spannung der Trenndiode), URT (AD-Wert Rot), UGR (AD-Wert Grün), UBL (AD-Wert Blau) über iCounts Messungen werden direkt ausgelesen.

### 2.2.3 MTCSGetADCAVR

**Die Messung mit -Mittelwertbildung**

*Definition*
int USB_DLL_API MTCSGetADCAVR (unsigned short * usBuf, int iCounts);

*Parameter*

| | |
|---|---|
| int iCounts | Anzahl der Messwerte für die Mittelwertbildung (Param1 im Beispiel) |
| unsigned short * usBuf | Pointer auf den Puffer, der die ADC - Werte enthält<br>Puffergröße für die Antwort vom Mod EVA : 8 Byte |

*Rückgabewert*

| | |
|---|---|
| = 0 | fehlerfrei |
| = 2 | Firmware erkannte illegales Kommando |
| = 4..x | Parameterfehler Parameter 1, oder.. Parameter x |
| = 7 | Übertragungsfehler |

*Anmerkung*
UTD (AD-Wert der Spannung der Trenndiode), URT (AD-Wert Rot), UGR (AD-Wert Grün), UBL (AD-Wert Blau) über iCounts Messungen werden in dieser Reihenfolge direkt ausgelesen.

### 2.2.4 MTCSGetADCBuf

**Auslesen des Messwertpuffers**

*Definition*
int USB_DLL_API MTCSGetADCBuf (unsigned short * usBuf);

*Parameter*
unsigned short * usBuf Pointer auf den Puffer, der die ADC - Werte enthält
Puffergröße für die Antwort vom Mod EVA : 8 Byte
*Rückgabewert*

| | |
|---|---|
| = 0 | fehlerfrei |
| = 2 | Firmware erkannte illegales Kommando |
| = 4..x | Parameterfehler Parameter 1, oder.. Parameter x |
| = 7 | Übertragungsfehler |

*Anmerkung*
Die ADC-Werte UTD, URT, UGR, UBL werden aus dem Puffer ausgelesen und die rote LED zurückgesetzt, wird verwendet nach Messung durch Tastendruck

### 2.2.5 MTCSGetADC

**Die einfache Messung**

*Definition*
int USB_DLL_API MTCSGetADC (unsigned short * usBuf);
*Parameter*
unsigned short * usBuf Pointer auf den Puffer, der die ADC - Werte enthält
Puffergröße für die Antwort vom Mod EVA : 8 Byte
*Rückgabewert*

| | |
|---|---|
| = 0 | fehlerfrei |
| = 2 | Firmware erkannte illegales Kommando |
| = 4..x | Parameterfehler Parameter 1, oder.. Parameter x |
| = 7 | Übertragungsfehler |

*Anmerkung*
Die ADC-Werte UTD (AD-Wert der Spannung der Trenndiode), URT (AD-Wert Rot), UGR (AD-Wert Grün), UBL (AD-Wert Blau) werden direkt ausgelesen.

### 2.2.6 MTCSSetSwitch

**Der Schalter für Transimpedanzverstärkung der Trenndiode wird gesetzt.**

*Definition*
int USB_DLL_API MTCSSetSwitch(int iCounts);

*Parameter*

Int iCounts — Wert, auf den der Schalter gestellt werden soll,
0(aus) oder 4(an)
(Param1 im Beispiel)

*Rückgabewert*

| | |
|---|---|
| = 0 | fehlerfrei |
| = 2 | Firmware erkannte illegales Kommando |
| = 4..x | Parameterfehler Parameter 1, oder.. Parameter x |
| = 7 | Übertragungsfehler |

### 2.2.7 MTCSSetEPoti

**Die Helligkeit der LED-Beleuchtung (nur bei MTCS-ME1 FRONT und TOP) wird eingestellt.**

*Definition*
int USB_DLL_API MTCSSetEPoti (int iCounts);

*Parameter*
Int iCounts — Wert, auf den das EPoti gestellt werden soll
(Param1 im Beispiel)

*Rückgabewert*

| | |
|---|---|
| = 0 | fehlerfrei |
| = 2 | Firmware erkannte illegales Kommando |
| = 4..x | Parameterfehler Parameter 1, oder.. Parameter x |
| = 7 | Übertragungsfehler |

*Anmerkung*
Das E-Poti auf übergebenen Wert (0x00...0xFF) eingestellt.

## 2.3 MTCS – Funktionen für MTCS-ME1 DARK CCC

### 2.3.1 MTCSStartRGB, MTCSStopRGB, MTCSGetRGB

Diese Kommandos werden zur Messung von Lichtquellen eingesetzt, z.B. bei LED-Beleuchtungen oder auch bei Monitoren.
Der Parameter iTime wird die Integrationszeit eingetragen. Sie ist zwischen 5000µs und 25000µs variierbar. Das entspricht der Synchronisation auf Frequenzen von Quellen (z.B. CRT-Monitoren) 200Hz bis 40Hz.
Eine Messung erfolgt über ein vielfaches der angegebenen Integrationszeit. Der Messwert („ADC / Anzahl der Iterationen“) repräsentiert den integralen Hub des Signalpegels geteilt durch die Anzahl der Integrationsintervalle („Anzahl der Iterationen“).
Mit dem Parameter „iZyklen“ wird das Abbruchkriterium, die maximale Anzahl der aufgenommenen Integrationsintervalle für einen Messzyklus, festgelegt.

### 2.3.2 MTCSStartRGB

**Die Intervallzeit und die maximale Anzahl der Integrationszyklen wird festgelegt. Die RGB-Integration wird gestartet.**

*Definition*
int USB_DLL_API MTCSStartRGB (unsigned short* usBuf, int iTime, int iZyklen);

*Parameter*

| | |
|---|---|
| unsigned short* usBuf | Pointer auf den Puffer, der die Ergebnismeldung enthält<br>Puffergröße für die Antwort vom Mod EVA : 2 Byte |
| int iTime | Intervallzeit<br>(Param1 im Beispiel) |
| int iZyklen | max. Integrationszyklen<br>(Param2 im Beispiel) |

*Rückgabewert*

| | |
|---|---|
| = 0 | fehlerfrei |
| = 2 | Firmware erkannte illegales Kommando |
| = 4..x | Parameterfehler Parameter 1, oder.. Parameter x |
| = 7 | Übertragungsfehler |

### 2.3.3 MTCSStopRGB

**Das Kommando stoppt eine Messung und liest die aktuellen RGB-Werte aus und die Anzahl der Intervalle, die durchlaufen wurden**

*Definition*
int USB_DLL_API MTCSStopRGB (unsigned short* usBuf);

*Parameter*

| | |
|---|---|
| unsigned short* usBuf | Pointer auf den Puffer, der die Ergebnismeldung enthält (Rot, Grün, Blau und Anzahl der Zyklen für Rot, Anzahl der Zyklen für Grün, Anzahl der Zyklen für Blau.<br>Puffergröße für die Antwort vom Mod EVA : 12 Byte |

*Rückgabewert*

| | |
|---|---|
| = 0 | fehlerfrei |
| = 2 | Firmware erkannte illegales Kommando |
| = 4..x | Parameterfehler Parameter 1, oder.. Parameter x |
| = 7 | Übertragungsfehler |

.

*Anmerkung*

Das Kommando MTCSStopRGB ist im normalen Betrieb nicht notwendig, da die Messung nach der Anzahl der Iterationen beendet wird.

### 2.3.4 MTCSGetRGB

**Das Kommando liest die aktuellen RGB-Werte und die Anzahl der Iterationen aus, die durchlaufen wurden.**

*Definition*

int USB_DLL_API MTCSGetRGB (unsigned short* usBuf);

*Parameter*

| | |
|---|---|
| unsigned short* usBuf | Pointer auf den Puffer, der die Ergebnismeldung enthält (Rot, Grün, Blau und Anzahl der Iterationen für Rot, Anzahl der Iterationen für Grün, Anzahl der Iterationen für Blau. Puffergröße für die Antwort vom Mod EVA : 12 Byte |

*Rückgabewert*

| | |
|---|---|
| = 0 | fehlerfrei |
| = 2 | Firmware erkannte illegales Kommando |
| = 4..x | Parameterfehler Parameter 1, oder.. Parameter x |
| = 7 | Übertragungsfehler |

## 2.4 MTCS – Funktionen für MTCS-C2

### 2.4.1 MTCSGetADCAVR2

**Die Verstärkung und ADC-Mittelwerte URT(AD-Wert Rot), UGR(AD-Wert Grün), UBL(AD-Wert Blau) über iCounts Messungen werden direkt ausgelesen.**

*Definition*
int USB_DLL_API MTCSGetADCAVR2 (int iIndex, unsigned short* usBuf, int iCounts);

*Parameter*

| | | |
|---|---|---|
| int iIndex | = 0 | reserviert für Devicenumber |
| int iCounts | Anzahl der Messwerte für die Mittelwertbildung (Param1 im Beispiel) | |
| unsigned short * usBuf | Pointer auf den Puffer, der den Verstärkungsfaktor und die ADC - Werte enthält<br>Puffergröße für die Antwort vom Col2 : 8 Byte | |

*Rückgabewert*

| | |
|---|---|
| = 0 | fehlerfrei |
| = 2 | Firmware erkannte illegales Kommando |
| = 3 | Messwertdifferenz zu groß |
| = 4..x | Parameterfehler Parameter 1, oder.. Parameter x |
| = 7 | Übertragungsfehler |

*Anmerkung*
Die Verstärkung kann mit MTCSSetParameter gesetzt oder mit MTCSSearchAmplification automatisch bestimmt werden.
Wenn der Verstärkungsfaktor 0 ist, wird zunächst der optimale Verstärkungsfaktor für die Messreihe gesucht. Dazu werden solange AD-Wandlungen auf den Kanälen Rot, Grün, Blau durchgeführt, bis alle Kanäle innerhalb eines definierten ADC-Arbeitsbereiches Werte liefern. Wenn der Verstärkungsfaktor im Bereich (1, 8) liegt, gibt es die Phase „Automatische Suche des optimalen Verstärkungsfaktors“ nicht.
Es werden N-Messreihen durchgeführt und der Mittelwert gebildet.
Weichen Folgemesswerte vom ersten Messwert um mehr als 10% ab so wird diese als Fehlercode 0xE003 protokolliert. (Dient als Hinweis auf größere Störquellen oder gepulster Lichtquellen für die das Messverfahren nicht geeignet ist.
Diese 3 Mittelwerte und der eingestellte Verstärkungsfaktor werden in der Antwort an den Host gesendet.

### 2.4.2 MTCSGetADCSummen

**Die Verstärkung und ADC-Mittelwerte URT(AD-Wert Rot), UGR(AD-Wert Grün), UBL(AD-Wert Blau) über iCounts Millisekunden werden aufsummiert und ausgelesen.**

*Definition*
int USB_DLL_API MTCSGetADCSummen (int iIndex, unsigned long* ulBuf, int iCounts);

*Parameter*

| | | |
|---|---|---|
| int iIndex | = 0 | reserviert für Devicenumber |
| int iCount | | Anzahl der Millisekunden, die für die Summenbildung gemessen werden soll. |

(Param1 im Beispiel)

unsigned long * ulBuf Pointer auf den Puffer, der den Verstärkungsfaktor und die ADC – Werte und die Anzahl der aufsummierten Messungen enthält
Puffergröße für die Antwort vom Col2: 20 Byte

*Rückgabewert*

| | |
|---|---|
| = 0 | fehlerfrei |
| = 2 | Firmware erkannte illegales Kommando |
| = 3 | Messwertdifferenz zu groß |
| = 4..x | Parameterfehler Parameter 1, oder.. Parameter x |
| = 7 | Übertragungsfehler |

*Anmerkung*

Die Verstärkung kann mit MTCSSetParameter gesetzt oder mit MTCSSearchAmplification automatisch bestimmt werden. Wenn der Verstärkungsfaktor 0 ist, wird zunächst der optimale Verstärkungsfaktor für die Messreihe gesucht. Dazu werden solange AD-Wandlungen auf den Kanälen Rot, Grün, Blau durchgeführt, bis alle Kanäle innerhalb eines definierten ADC-Arbeitsbereiches Werte liefern. Wenn der Verstärkungsfaktor im Bereich (1, 8) liegt, gibt es die Phase „Automatische Suche des optimalen Verstärkungsfaktors“ nicht. Es werden über iCounts Millisekunden N-Messreihen durchgeführt und die Werte aufsummiert. Es werden der eingestellte Verstärkungsfaktor, diese 3 aufsummierten AD-Werte und die Anzahl der aufsummierten Messungen in der Antwort an den Host gesendet. Alle Werte sind unsigned long Werte.

### 2.4.3 MTCSSetParameter

**Die Parameter Verstärkungsfaktor und Toleranz werden gesetzt.**

*Definition*

int USB_DLL_API MTCSSetParameter (int iIndex, int iAmpl, int iTol);

*Parameter*

| | | |
|---|---|---|
| int iIndex | = 0 | reserviert für Devicenumber |
| int iAmpl | | Verstärkungsfaktor |
| int iTol | | Toleranz |
| | | (Param1 and Param2 im Beispiel) |

*Rückgabewert*

| | |
|---|---|
| = 0 | fehlerfrei |
| = 2 | Firmware erkannte illegales Kommando |
| = 7 | Übertragungsfehler |

*Anmerkung*

Defaultwerte sind: Verstärkungsfaktor = 8 Toleranz = 90(Prozent)
Die beiden Parameter gehen in die Phase „Automatische Suche des optimalen Verstärkungsfaktors“ ein.

### 2.4.4 MTCSSearchAmplification

**optimalen Verstärkungsfaktor suchen**

*Definition*
int USB_DLL_API MTCSSearchAmplification (int iIndex, unsigned short* usBuf, int iLimit);

*Parameter*

| | | |
|---|---|---|
| int iIndex | = 0 | reserviert für Devicenumber |
| int iLimit | | Grenzwerte(in %), 1 Byte |
| unsigned *short * usBuf* | | Pointer auf den Puffer, der den Verstärkungsfaktor und die ADC - Werte enthält<br>Puffergröße für die Antwort: 8 Byte |

*Rückgabewert*

| | |
|---|---|
| = 0 | fehlerfrei |
| = 2 | Firmware erkannte illegales Kommando |
| = 7 | Übertragungsfehler |

*Anmerkung*
Der optimale Verstärkungsfaktor für die Messreihe wird gesucht. Dazu werden solange AD-Wandlungen auf den Kanälen Rot, Grün, Blau mit kleiner werdenden Verstärkungsfaktor durchgeführt, bis der stärkste Kanal ein Signal liefert, das unterhalb des Grenzwertes (Wertebereich 1..100%) liegt.
Die Messwerte werden entsprechend dem Parameter Bitverschiebung erweitert.
Der gefundene Verstärkungsfaktor und die 3 letzten Messwerte aus dem Algorithmus werden in der Antwort an den Host gesendet.
Der gefundene Verstärkungsfaktor wird für die folgenden Messungen eingestellt.
Dieser Algorithmus kann dazu verwendet werden, die AD-Wandlungen in einen optimalen Arbeitsbereich zu bringen. Das ist notwendig, wenn durch sehr helle Lichtquellen der Farbsensor beim höchsten Verstärkungsfaktor übersteuert wird. Dadurch würde die AD-Wandlung immer maximale AD-Werte liefern.

### 2.4.5 MTCSWriteMemToAdr

**iAnz Integerwerte des Inhaltes des EEPROM wird immer byteweise fortlaufend ab Adresse iAdr geschrieben.**

*Definition*
int USB_DLL_API MTCSWriteMemToAdr(int iIndex, unsigned char* cBuf, int iAdr, int iAnz)

*Parameter*

| | | |
|---|---|---|
| int iIndex | = 0 | reserviert für Devicenumber |
| unsigned char* cBuf | | Pointer auf den Puffer, der den Inhalt für den EEPROM enthält<br>(Param1 im Beispiel) |
| int iAdr | | Adresse, ab welcher der EEProm neu geschrieben werden soll.<br>(Param2 im Beispiel) |
| int iAnz | | Anzahl der Integerwerte, die geschrieben werden sollen. |

(=1 im Beispiel)

*Rückgabewert*

| | |
|---|---|
| = 0 | fehlerfrei |
| = 2 | Firmware erkannte illegales Kommando |
| = 4..x | Parameterfehler Parameter 1, oder.. Parameter x |
| = 6 | Anzahl der Integerwerte überschreitet Gesamtanzahl für EEProm |
| = 7 | Übertragungsfehler |

*Anmerkung*
Es werden iAnz Worte als Datenstrom in den externen EEPROM ab einer bestimmten Adresse iAdr geschrieben.
iAnz = (1, 57)
Der EEPROM ist 128 Byte groß. Er kann also im Bereich (0x00, 0x6F) adressiert werden. Wenn Adresse + 2*Anzahl zu schreibender Datenworte größer ist als die obere Adresse, wird der Fehlercode 6 zurückgegeben.

### 2.4.6 MTCSReadMemFromAdr

**iAnz Integerwerte des Inhaltes des EEPROM wird immer byteweise fortlaufend ab Adresse iAdr gelesen.**

*Definition*
int USB_DLL_API MTCSReadMemFromAdr(int iIndex, unsigned char* cBuf, int iAdr, int iAnz);

*Parameter*

| | | |
|---|---|---|
| int iIndex | = 0 | reserviert für Devicenumber |
| unsigned char* cBuf | Pointer auf den Puffer, der den Inhalt des EEPROM enthält<br>Puffergröße für die Antwort vom Colorimeter2 : iAnz*2 Byte | |
| int iAdr | Adresse, ab welcher der EEProm neu geschrieben werden soll. (Param1 im Beispiel) | |
| int iAnz | Anzahl der Integerwerte, die gelesen werden sollen. (Param2 im Beispiel) | |

*Rückgabewert*

| | |
|---|---|
| = 0 | fehlerfrei |
| = 1 | Parameterfehler |
| = 2 | Firmware erkannte illegales Kommando |
| = 6 | Anzahl der Integerwerte überschreitet Gesamtanzahl für EEProm |
| = 7 | Übertragungsfehler |

*Anmerkung*
Es werden iAnz - Worte als Datenstrom vom externen EEPROM ab einer bestimmten Adresse iAdr gelesen.
Der EEPROM ist 128 Byte groß. Er kann also im Bereich (0x00, 0x6F) adressiert werden. Wenn Adresse + 2* iAnz zu lesender Datenworte größer ist als die obere Adresse, wird der Fehlercode 6 zurückgegeben.

### 2.4.7 MTCSGetSerienNummer

**Lesen der Seriennummer aus dem EEPROM des Boards**

*Definition*

void USB_DLL_API MTCSGetSerienNummer (int iIndex, char* cBuf);

*Parameter*

| | | |
|---|---|---|
| int iIndex | = 0 | reserviert für Devicenumber |
| char* cBuf | Pointer auf den Puffer, der die Seriennummer enthält, die im USB-Descriptor steht. Puffergröße für die Antwort vom Colorimeter2 : 16 Byte | |

*Rückgabewert*

| | |
|---|---|
| = 0 | fehlerfrei |
| = 2 | Firmware erkannte illegales Kommando |
| = 7 | Übertragungsfehler |

# 3 EEPROM

Der Inhalt des EEPROM ist strukturiert.
Systemparameter sind Daten, die dauerhaft gespeichert werden.
Der Zugriff auf die Daten erfolgt über die Kommandos ReadMemory, WriteMemory, WriteMemToAdr und ReadMemFromAdr. Diese Kommandos behandeln den Speicher wie einen Datenstrom.

Der EEProm des Mod EVA ist maximal 1024 Byte groß. Die letzten 10 Byte sind für interne Kennungen reserviert. Es sollten nur die notwendigen Daten in den EEProm geschrieben werden, da EEProm schreiben langsam ist.

| **Nr.** | **Inhalt** | **Länge in WORD** |
|---|---|---|
| 1. | Applikationscode | 1 |
| 2. | E-Poti | 1 |
| 3. | Dark value | 3 |
| 4. | White value | 3 |
| 5. | BL (Fremdlichtkompensation 0=off, 1=on) | 1 |
| 6. | Anzahl Mittelwert | 1 |
| 7. | Integrationszeit | 1 |
| 8. | Korrekturmatrix Sensor | 9 |
| 9. | Korrekturmatrix Display | 9 |
| 10. | freier Speicher | |

*Tabelle 1: Übersicht über die Systemparameter im EEProm des Mod EVA*

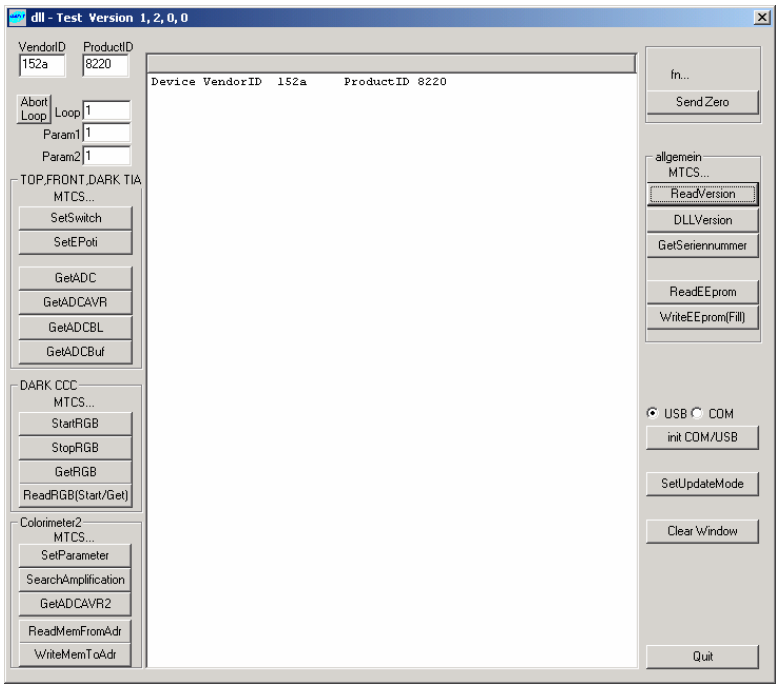
Der EEProm des Colorimeter 2 ist maximal 128 Byte groß
Die oberen 16 Byte sind für die Seriennummer reserviert.

| | **Inhalt** | **Länge in WORD** | **Bemerkungen** |
|---|---|---|---|
| 1 | Applikationscode | 1 | Der Applikationscode für das Colorimeter 2 ist 0x0011 |
| 2 | E-Poti | 1 | keine Bedeutung |
| 3 | Dark value | 3 | ist Parameter für die PC-Anwendung |
| 4 | White value | 3 | ist Parameter für die PC-Anwendung |
| 5 | Fremdlichtkompen sation BL | 1 | keine Bedeutung |
| 6 | Anzahl Mittelwerte | 1 | für GetADCAvr2 |
| 7 | Verstärkungsfaktor | 1 | für GetADC2, GetADCAvr2 |
| 8 | Korrekturmatrix Sensor | 9 | ist Parameter für die PC-Anwendung |
| 9 | Korrekturmatrix Display | 9 | ist Parameter für die PC-Anwendung |
| | | | |

*Tabelle 2: Übersicht über die Systemparameter im EEProm des Colorimeter2*

# 4 TESTOBERFLÄCHE

Das Projekt „dll_test.dsw“ (Sourcecode in Dll_test.zip) ist ein c++ - Projekt für das Microsoft Visual Studio 6.0 und demonstriert die Benutzung der einzelnen in der dll benutzten Funktionen.
Wird das include-File MTCSApi.h unverändert benutzt, muß die Präprozessor-Definition EXE im Projekt angegeben werden.




Mit MTCSInitSystem() wird das System beim Start initialisiert.

Die MTCS… - Aufrufe arbeiten mit maximal 2 Eingabeparametern. Die Bedeutung der Parameter ist der Kommandobeschreibung zu entnehmen. Loop enthält die Anzahl der Durchläufe für das aufgerufene Kommando. Mit „Abort Loop“ kann diese Schleife abgebrochen werden.
„Clear Window“ löscht das vorhandene Anzeigefenster. Alle einzugebenden Parameter werden als Dezimalzahlen interpretiert, VendorID und ProductID werden als hex-Zahlen interpretiert.

---

Für weitere Informationen wenden Sie sich bitte an:

**MAZeT GmbH**
**Vertrieb:**
Göschwitzer Straße 32
07745 Jena
Tel: +49 3641 2809-0
Fax: +49 3641 2809-12
E-Mail: sales@MAZeT.de
Url: http://www.MAZeT.de