

**Software Bibliotheken
WinMDB32.dll für Windows 98/ME/2000/XP
und
libmdb.so für Linux**

**zur Kommunikation mit Geräten
mit Multi Drop Bus Protokoll**

Version 2.01

unterstützt:

**Münzprüfer
Geldwechsler
Notenprüfer**

Stand vom 18. März 2004

Inhalt

1. Systemvoraussetzungen.....	3
2. Beschreibung der Software-Bibliothek.....	3
2.1. Allgemeines	3
2.2. Funktionsbeschreibung.....	4
2.2.1. Allgemeine MDB-Funktionen	4
2.2.1.1. MDB öffnen und schließen	4
2.2.1.2. Alle Geräte am MDB zurücksetzen	4
2.2.1.3. Schaltleitungen setzen	4
2.2.2. Behandlung eines Münzprüfers.....	4
2.2.2.1. Münzprüfer öffnen und schließen.....	4
2.2.2.2. Daten vom Münzprüfer auslesen.....	4
2.2.2.3. Münzfreigabe und Sortiersteuerung setzen und abfragen	5
2.2.2.4. Münzprüfer pollen und letzten Pollstatus abfragen	5
2.2.2.5. Sonstige Funktionen	6
2.2.3. Behandlung eines Geldwechslers.....	6
2.2.3.1. Geldwechsler öffnen und schließen.....	6
2.2.3.2. Daten vom Geldwechsler auslesen.....	6
2.2.3.3. Tubenstatus holen.....	7
2.2.3.4. Münzfreigabe setzen	7
2.2.3.5. Münzen auszahlen	7
2.2.3.6. Betrag auszahlen	7
2.2.3.7. Geldwechsler pollen und letzten Pollstatus abfragen	8
2.2.3.8. Sonstige Funktionen	9
2.2.4. Behandlung eines Notenprüfers.....	9
2.2.4.1. Notenprüfer öffnen und schließen	9
2.2.4.2. Daten vom Notenprüfer auslesen.....	9
2.2.4.3. Stackerstatus auslesen	9
2.2.4.4. Notensteuerung setzen	9
2.2.4.5. Note einziehen bzw. zurückgeben	10
2.2.4.6. Notenprüfer pollen und letzten Pollstatus abfragen	10
2.2.4.7. Sonstige Funktionen	11
3. Beispiel für die Behandlung eines Münzprüfers.....	12

1. Systemvoraussetzungen

Zum Anschluß der MDB Geräte ist ein wh Münzprüfer serieller COM 200 bzw. COM 400 oder USB COM 300 Adapter erforderlich.

Als anschließbare Geräte am Multi Drop Bus (im Folgenden als MDB bezeichnet) werden die Münzprüfer der wh Münzprüfer Dietmar Trenner GmbH sowie alle MDB konformen Geldwechsler und Notenprüfer unterstützt.

2. Beschreibung der Software-Bibliothek

2.1. Allgemeines

Die Software liegt als 32-Bit Windows DLL für Windows® 98, ME, 2000 und XP bzw. als Shared Library für Linux vor. Zur Nutzung muß sie in das Systemverzeichnis der jeweiligen Windows Version bzw. in das User Lib Verzeichnis kopiert werden.

Alle Funktionen sind vom Typ `int` und geben einen Fehlercode zurück. Dieser kann entweder ein von der Protokollebene durchgereichter oder ein von der Befehlsebene generierter sein.

Funktionen, die Ergebnisse in einer Struktur zurückgeben, erwarten grundsätzlich einen Pointer auf eine *existierende* Struktur, die sie mit Daten füllen. Alle Funktionen benutzen die `stdcall` Aufruf Konventionen.

Die Integration der DLL/ Shared Library in eigene Applikationen wird unterstützt durch die ausführlich kommentierten Dateien `WinMDB32.h` für C(++) bzw. `WinMDB32_Defs.pas` und `WinMDB32_Interface.pas` für Pascal/Delphi. Hier sind auch alle Fehlercodes und Strukturdefinitionen nachzulesen.

Alle Funktionen kehren grundsätzlich erst nach vollständiger Ausführung der Aktionen durch Peripherie bzw. bei Abbruch durch einen Fehler zurück. Die reine Kommunikation mit einem Gerät beansprucht je nach Datenmenge zwischen 5 und 50ms. Hierzu kommen noch die internen Verarbeitungszeiten der Geräte. Besonders zeitintensiv sind folgende Arten von Aufrufen:

- Reset aller Geräte am MDB: ca. 200ms reine Ausführungszeit der Funktion, bis zur Bereitschaft der einzelnen Geräte kann eine deutlich längere Zeit verstreichen.
- Initialisieren von Geräten: bei Wechslern und Notenprüfern einige 100ms bis zu einigen Sekunden, bedingt durch mechanische Positioniervorgänge.
- Abfragen und Setzen von Geräteinformationen: einige 100ms durch Zugriff auf serielle EEPROMs.
- Auszahlen von Münzen aus einem Wechsler: je nach Münzanzahl mehrere Sekunden.
- Rückgabe einer Note aus einem Notenprüfer: typisch 1s.

Die genannten Zeiten sind nur allgemeine Richtwerte und können auch innerhalb einer Geräteklasse je nach Hersteller erheblich variieren.

Alle hier beschriebenen MDB-Geräte geben u.a. eine Länderkennung aus, die angibt, für welche Währung sie eingerichtet sind. Die Bibliothek prüft nicht, ob bei mehreren angeschlossenen Geräten die Länderkennung identisch ist, dies muß im Bedarfsfall durch die übergeordnete Applikation erfolgen.

2.2. Funktionsbeschreibung

2.2.1. Allgemeine MDB-Funktionen

2.2.1.1. MDB öffnen und schließen

```
_stdcall int OpenMDB( int com_nr ); (Windows)
```

Es wird versucht, den angegebenen COM-Port zu öffnen und für MDB zu initialisieren. Die erfolgreiche Ausführung dieser Funktion ist Voraussetzung für alle weiteren Aktionen.

Wird mit der USB Version COM300 gearbeitet, kann als COM-Port Nummer auch die „0“ angegeben werden. Dann wird unabhängig vom installierten virtuellen Port immer der USB Adapter gefunden.

```
_stdcall int OpenMDB_str( int com_nr ); (Linux)
```

Es wird versucht, das angegebene tty Device zu öffnen und für MDB zu initialisieren. Die erfolgreiche Ausführung dieser Funktion ist Voraussetzung für alle weiteren Aktionen.

```
_stdcall int CloseMDB();
```

Der für den MDB geöffnete COM-Port wird geschlossen und wieder freigegeben.

2.2.1.2. Alle Geräte am MDB zurücksetzen

```
_stdcall int ResetMDBDevices();
```

An allen angeschlossenen Geräten wird ein Hardware Reset ausgelöst. Dies kann unmittelbar nach der Initialisierung des MDB durchgeführt werden, um alle Geräte in einen definierten Zustand zu versetzen.

2.2.1.3. Schaltungen setzen

```
_stdcall int SetAuxPort ( char stat, char msk );
```

Mit dieser Funktion können, soweit hardwareseitig implementiert, vier zusätzliche Schaltungen an der Karte gesetzt werden.

Dabei gibt *stat* den gewünschten Zustand an und *msk*, ob das jeweilige Bit beeinflusst werden soll.

2.2.2. Behandlung eines Münzprüfers

2.2.2.1. Münzprüfer öffnen und schließen

```
_stdcall int OpenEMP();
```

Ein Münzprüfer am zuvor initialisierten MDB wird geöffnet und identifiziert.

```
_stdcall int CloseEMP();
```

Ein geöffneter Münzprüfer wird geschlossen.

2.2.2.2. Daten vom Münzprüfer auslesen

```
_stdcall int GetEMPInfo ( struct EMPInfo* eip );
```

Aus dem Münzprüfer werden alle notwendigen Informationen zur Münzverarbeitung ausgelesen. Die Struktur, auf die `eip` zeigt, wird mit Informationen über den Münzprüfer gefüllt. Wichtig hiervon ist vor allem die Zuordnung des Geldwertes zu den Münznummern, die benötigt wird, wenn bestimmte Münzen gesperrt werden sollen. Bei Münzprüfern mit Sortierung sind u.U. noch die Informationen zur Default-Sortiereinstellung von Interesse.

Die Länderkennung `country` enthält die internationale Telefonvorwahl des Landes, für das der Münzprüfer eingerichtet ist, z.B. 0x0044 für Großbritannien oder einen ISO 4217 Währungscode, z.B. 0x1978 für den Euro.

Die Angabe der Dezimalstellen der Währung (`decimals`) kann für optimale Formatierung von Betragsanzeigen nützlich sein.

2.2.2.3. Münzfreigabe und Sortiersteuerung setzen und abfragen

```
_stdcall int GetEMPCurrent ( struct EMPCurrent* ecp );
```

Die aktuellen Einstellungen zur Münzfreigabe und zur Sortiersteuerung werden aus dem Münzprüfer ausgelesen und in die Struktur, auf die `ecp` zeigt, eingetragen. Die Bedeutung der Daten ist identisch mit `SetEMPCurrent()` (s.u.).

```
_stdcall int SetEMPCurrent ( struct EMPCurrent* ecp );
```

Die Struktur, auf die `ecp` zeigt, enthält die Angaben zur Münzfreigabe und zur Sortiersteuerung:

<code>cashbox</code>	gibt die Nummer des Kassenschachtes an
<code>unlock[x]</code>	"1" gibt die Münze Nr. x frei, "0" sperrt sie
<code>incbox[x]</code>	"1" leitet die Münze Nr. x unabhängig vom eingestellten Sortierschacht in den Kassenschacht
<code>tube[x]</code>	gibt den Sortierschacht für die Münze Nr. x an

Bei Münzprüfern ohne Sortierung sind `cashbox` und `incbox` ohne Bedeutung. Im Zweifel sollten alle Werte für `tube`, wie auch der Wert für `cashbox` zuerst mittels `SetEMPCurrent` gelesen und unverändert gelassen werden.

Nach dem Einschalten sind alle Münzen zunächst gesperrt, die Umleitung in die Hauptkasse für alle Münzen deaktiviert und die Sortierschachtzuordnung auf die Default-Werte gesetzt.

Soll nur für bestimmte Münzen die Einstellung geändert werden, empfiehlt es sich, zuerst mit `GetEMPCurrent()` (s.u.) die aktuellen Einstellungen auszulesen, in der Struktur die gewünschten Werte zu ändern und anschließend diese Struktur an `SetEMPCurrent()` zu übergeben.

2.2.2.4. Münzprüfer pollen und letzten Pollstatus abfragen

```
_stdcall int PolleMP();
```

Soll der Münzprüfer Münzen annehmen, muß diese Funktion zyklisch ca. alle 25 bis 200ms aufgerufen werden. Durch diese Funktion werden vorhandene Informationen zur letzten Münzverarbeitung ausgelesen.

Der Rückgabewert informiert allgemein über das Ergebnis des Pollens, er ist identisch mit dem Inhalt von `status` in `EMPPollStatus`. Der Status ist ein Bitvektor, für den die folgenden Masken definiert sind:

<code>EMPP_COIN</code>	Der Münzprüfer hat eine Münze akzeptiert.
<code>EMPP_RESET</code>	Der Münzprüfer hat zuletzt eine Reset (Hard- oder Software) ausgeführt.

EMPP_RETURN	Der Rückgabehebel am Münzprüfer wurde betätigt.
EMPP_REJECT	Eine Münze wurde abgewiesen.
EMPP_NOANSWER	Der Münzprüfer hat nicht geantwortet. Dies ist kein Fehler, da während der Münzverarbeitung nicht auf jeden Poll geantwortet wird.
EMPP_TIMEOUT	Fehler: Der Münzprüfer hat zu lange nicht auf Poll reagiert.
EMPP_UNKNOWN	Unbekannter Fehler
EMPP_MDBERR	Es ist ein MDB Fehler aufgetreten, Fehlercode in <code>mdberr</code> von <code>EMPPollStatus</code>

Ein Status von „0“ zeigt an, daß nichts zu vermelden war und keine Reaktionen erforderlich sind. Bei `EMPP_COIN` könnte z.B. eine Windows-Message verschickt werden, die anderen Programmteilen anzeigt, dass der Kredit anzupassen ist.

Die Reaktion auf `EMPP_RETURN` könnte z.B. bei einem zusätzlich angeschlossenen Notenprüfer darin bestehen, eine Note in Escrow-Position zurückzugeben. Ist nur ein Münzprüfer angeschlossen, erfordert dieser Status keine weiteren Aktionen.

Auf `EMPP_TIMEOUT` sollte reagiert werden, indem der Münzprüfer und alle eventuell noch angeschlossenen Geräte geschlossen werden, ein `ResetMDBDevices` ausgeführt, alle wieder geöffnet, ausgelesen und gesetzt werden. Tritt danach der Fehler wieder auf, ist abzubrechen und der Münzprüfer als defekt anzusehen.

Die übrigen Status erfordern im normalen Betrieb keine Reaktionen durch die Software bzw. deuten auf grundsätzliche Fehler in Hard- oder Software hin.

```
_stdcall int GetEMPLastPollStat( struct EMPPollStatus* epp );
```

Liefert detaillierte Informationen über das Ergebnis des letzten Polls. Ansonsten gilt im Wesentlichen das unter `PollEMP()` zu den Fehlerzuständen Gesagte.

2.2.2.5. Sonstige Funktionen

```
_stdcall int ResetEMP();
```

Im Münzprüfer wird ein Software-Reset ausgelöst.

2.2.3. Behandlung eines Geldwechslers

2.2.3.1. Geldwechsler öffnen und schließen

```
_stdcall int OpenCHG();
```

Ein Geldwechsler am zuvor initialisierten MDB wird geöffnet und identifiziert.

```
_stdcall int CloseCHG();
```

Ein geöffneter Geldwechsler wird geschlossen.

2.2.3.2. Daten vom Geldwechsler auslesen

```
_stdcall int GetCHGInfo ( struct CHGInfo* cip );
```

Aus dem Geldwechsler werden alle notwendigen Informationen zur Münzverarbeitung ausgelesen. Die Struktur, auf die `cip` zeigt, wird mit Informationen über den Geldwechsler gefüllt.

Wichtig hiervon ist vor allem die Zuordnung des Geldwertes zu den Münznummern (`coin_value`), die benötigt wird, wenn bestimmte Münzen gesperrt bzw. freigegeben werden sollen. Wichtig sind weiterhin die Informationen, welche Münze in Tuben geleitet werden können (`tube_coin`) und welches die kleinste auszahlabare Münze ist (`least_coin`). Diese Informationen, zusammen mit den Angaben zum Tubeninhalte werden benötigt, um zu entscheiden, ob ein bestimmter Betrag ausgezahlt werden soll.

Die Länderkennung `country` enthält die internationale Telefonvorwahl des Landes, für das der Geldwechsler eingerichtet ist, z.B. 0x0044 für Großbritannien oder einen ISO 4217 Währungscode, z.B. 0x1978 für den Euro.

Die Angabe der Dezimalstellen der Währung (`decimals`) kann für optimale Formatierung von Betragsanzeigen nützlich sein.

2.2.3.3. Tubenstatus holen

```
_stdcall int GetCHGTubeStat ( struct CHGTubeStat* ctp );
```

Die Füllstände und Zustände der Tuben werden gelesen und in die Struktur übertragen. Im Einzelnen sind das die Münzanzahl je Münzart (`coinctr`), die Information, ob die zugehörige Tube ihren maximalen Füllstand erreicht hat (`tube_full`) und ob die Tube evtl. gestört ist (`tube_err`). Diese Informationen werden benötigt, wenn bei Auszahlvorgang die Applikation die Münzzusammensetzung selbst bestimmen soll.

2.2.3.4. Münzfreigabe setzen

```
_stdcall int SetCHGCurrent ( struct CHGCurrent* ccp );
```

Das Array, auf die `ccp` zeigt, enthält die Münzfreigabe: Ein Wert `!=0` gibt die entsprechende Münze frei. Nach dem Einschalten bzw. einem Hardware-Reset sind alle Münzen zunächst gesperrt, ein Software-Reset verändert die Münzfreigabe jedoch nicht!

2.2.3.5. Münzen auszahlen

```
_stdcall int DispenseCHGCoins ( CHGDispense* dcp );
```

Es wird von den jeweiligen Münzarten die im Array angegebene Anzahl ausgegeben, sofern es möglich ist. Nach dem Rücksprung enthält das Array die tatsächlich ausgegebene Anzahl je Münze.

Vor Aufruf dieser Funktion sollte festgestellt werden, welche Münze überhaupt ausgegeben werden können (`tube_coin` in `CHGInfo`) und mittels `GetCHGTubeStat` der Status der entsprechenden Tube (Füllstand und ggf. Störung) abgefragt werden. Im Normalfall sollte dies jedoch nicht notwendig sein, da Auszahlungen wesentlich komfortabler mit der Funktion `DispenseCHGValue` (s.u.) getätigt werden können.

2.2.3.6. Betrag auszahlen

```
_stdcall int DispenseCHGValue ( double* dvp, int force );;
```

Es wird versucht den gewünschten Betrag auszuzahlen. Die Zusammenstellung der Münzen erfolgt durch diese Funktion.

Der Parameter `force` beeinflusst das Verhalten bei nicht ausreichendem Münzbestand in den Tuben: Ist er auf 0 gesetzt, wird in diesem Falle nichts ausgezahlt; ist er auf 1 gesetzt, wird ausgezahlt soweit es die Tubeninhalte zulassen. In jedem Fall wird in `*dvp` der tatsächlich ausgezahlte Betrag zurückgegeben.

2.2.3.7. Geldwechsler pollen und letzten Pollstatus abfragen

```
_stdcall int PollCHG();
```

Soll der Geldwechsler Münzen annehmen, muß diese Funktion zyklisch ca. alle 25 bis 200ms aufgerufen werden. Durch diese Funktion werden vorhandene Informationen zur letzten Münzverarbeitung ausgelesen.

Der Rückgabewert informiert allgemein über das Ergebnis des Pollens, er ist identisch mit dem Inhalt von `status` in `CHGPollStatus`. Der Status ist ein Bitvektor, für den u.a. die folgenden Masken definiert sind:

<code>CHGP_DEPOSIT</code>	Der Geldwechsler hat eine Münze akzeptiert, die Tubenfüllstände haben sich geändert.
<code>CHGP_DISPENSE</code>	Eine Münze wurde manuell durch die Bedientasten am Geldwechsler ausgegeben, die Tubenfüllstände haben sich geändert.
<code>CHGP_RESET</code>	Der Geldwechsler hat zuletzt eine Reset (Hard- oder Software) ausgeführt.
<code>CHGP_RETURN</code>	Der Rückgabehebel am Geldwechsler wurde betätigt.
<code>CHGP_REJECT</code>	Eine Münze wurde abgewiesen.
<code>CHGP_BUSY</code>	Der Geldwechsler ist intern beschäftigt.
<code>CHGP_POTBUSY</code>	Der Geldwechsler ist mit einem Auszahlvorgang beschäftigt
<code>CHGP_TIMEOUT</code>	Fehler: Der Geldwechsler hat zu lange nicht auf Poll reagiert.
<code>CHGP_UNKNOWN</code>	Unbekannter Fehler
<code>CHGP_MDBERR</code>	Es ist ein MDB Fehler aufgetreten, Fehlercode in <code>mdberr</code> von <code>BLVPollStatus</code>

Ein Status von „0“ zeigt an, daß nichts zu vermelden war und keine Reaktionen erforderlich sind. Bei `CHGP_DEPOSIT` könnte z.B. eine Windows-Message verschickt werden, die anderen Programmteilen anzeigt, dass der Kredit anzupassen ist.

Die Reaktion auf `CHGP_RETURN` könnte z.B. bei einem zusätzlich angeschlossenen Notenprüfer darin bestehen, eine Note in Escrow-Position zurückzugeben und bei Bedarf den Restkredit in Münzen auszuzahlen.

Auf `CHGP_TIMEOUT` sollte reagiert werden, indem der Geldwechsler und alle eventuell noch angeschlossenen Geräte geschlossen werden, ein `ResetMDBDevices` ausgeführt, alle wieder geöffnet, ausgelesen und gesetzt werden. Dabei ist zu berücksichtigen, daß der Geldwechsler auch bei Bedienung über Tasten teilweise die Kommunikation einstellt, dies jedoch keine Fehlfunktion darstellt.

Die übrigen Status erfordern im normalen Betrieb keine Reaktionen durch die Software bzw. deuten auf grundsätzliche Fehler in Hard- oder Software hin.

```
_stdcall int GetCHGLastPollStat( stuct CHGPollStatus* cpp );
```

Liefert detaillierte Informationen über das Ergebnis des letzten Polls. Ansonsten gilt im Wesentlichen das unter `PollCHG()` zu den Fehlerzuständen Gesagte.

2.2.3.8. Sonstige Funktionen

```
_stdcall int ResetCHG();
```

Im Geldwechsler wird ein Software-Reset ausgelöst.

2.2.4. Behandlung eines Notenprüfers

2.2.4.1. Notenprüfer öffnen und schließen

```
_stdcall int OpenBLV();
```

Ein Notenprüfer am zuvor initialisierten MDB wird geöffnet und identifiziert.

```
_stdcall int CloseBLV();
```

Ein geöffneter Notenprüfer wird geschlossen.

2.2.4.2. Daten vom Notenprüfer auslesen

```
_stdcall int GetBLVInfo ( struct BLVInfo* bip );
```

Aus dem Notenprüfer werden alle notwendigen Informationen zur Banknotenverarbeitung ausgelesen. Die Struktur, auf die `bip` zeigt, wird mit Informationen über den Notenprüfer gefüllt.

Wichtig hiervon ist vor allem die Zuordnung des Geldwertes zu den Notenummern (`bill_value`), die benötigt wird, wenn bestimmte Noten gesperrt bzw. freigegeben werden sollen. Wichtig ist weiterhin die Information, ob der Notenprüfer Escrow beherrscht (`escrowcap`), d.h. ob er eine Note in einer Position halten kann, von der sie sowohl endgültig eingezogen als auch wieder ausgegeben werden kann.

Die Länderkennung `country` enthält die internationale Telefonvorwahl des Landes, für das der Notenprüfer eingerichtet ist, z.B. 0x0044 für Großbritannien oder einen ISO 4217 Währungscode, z.B. 0x1978 für den Euro.

Die Angabe der Dezimalstellen der Währung (`decimals`) kann für optimale Formatierung von Betragsanzeigen nützlich sein.

2.2.4.3. Stackerstatus auslesen

```
_stdcall int GetBLVStacker ( struct BLVStacker* bsp );
```

Der Inhalt des Notenspeichers wird ausgelesen (`bills`) und gemeldet, ob der Speicher evtl. voll ist (`full`). Bei vollem Speicher könnte z.B. eine Aufforderung zum Leeren ausgegeben werden.

2.2.4.4. Notensteuerung setzen

```
_stdcall int SetBLVCurrent ( struct BLVCurrent* bcp );
```

Die Struktur, auf die `bcp` zeigt, enthält die Notensteuerung. Die jeweilige Note kann freigegeben (`unlock`) und für die Note Escrow erlaubt werden (`escrenab`), zur Funktion von Escrow s. 2.2.4.2. und 2.2.4.5.

Ist Escrow für eine Note erlaubt, sollte der Kredit bereits erhöht werden, wenn eine akzeptierte Note sich in dieser Position befindet, sollte wieder verringert werden wenn sie zurückgegeben wird und unverändert bleiben, wenn sie eingezogen wird. Ist Escrow gesperrt, wird eine akzeptierte Note sofort eingezogen und der Kredit erhöht.

Ohne Escrow kann bei Abbruch eines Verkaufsvorganges eine Geldrückgabe für eine akzeptierte Note nur in Form von Münzen über einen angeschlossenen Geldwechsler erfolgen.

2.2.4.5. Note einziehen bzw. zurückgeben

```
_stdcall int ReturnBLVBill;
```

Befindet sich eine Note in Escrow-Position, wird sie zurückgegeben, der Kredit muß ggf. entsprechend verringert werden

Wird in einem System mit Notenprüfer und Geldwechsler ein Verkaufsvorgang abgebrochen und das Geld zurückgegeben, könnte in folgender Weise verfahren werden:

- `ReturnBLVBill` aufrufen - eine eventuell in Escrow-Position befindliche Note wird zurückgegeben und der Kredit entsprechend verringert.
- `DispenseCHGValue` aufrufen, um Restkredit auszusahlen.

```
_stdcall int StackBLVBill;
```

Befindet sich eine Note in Escrow-Position, wird sie eingezogen. Da der Kredit bereits vorher erhöht wurde, bleibt er unverändert.

Beide Funktionen können gewissermaßen „auf Verdacht“ aufgerufen werden, die Behandlung des Notenprüfers und des Kredits erfolgt in jedem Falle korrekt, entsprechend der intern festgehaltenen Einstellungen und Position der fraglichen Note. Ein Rückgabewert `BLV_ESCRERR` bzw. `BLV_STACKERR` kann ignoriert werden. Diese Werte zeigen an, daß sich keine Note in der Escrow-Position befand und daher die Aktion nicht durchführbar war.

2.2.4.6. Notenprüfer pollen und letzten Pollstatus abfragen

```
_stdcall int PollBLV();
```

Soll der Notenprüfer Noten annehmen, muß diese Funktion zyklisch ca. alle 25 bis 200ms aufgerufen werden. Durch diese Funktion werden vorhandene Informationen zur letzten Notenverarbeitung ausgelesen.

Der Rückgabewert informiert allgemein über das Ergebnis des Pollens, er ist identisch mit dem Inhalt von `status` in `BLVPollStatus`. Der Status ist ein Bitvektor, für den u.a. die folgenden Masken definiert sind:

<code>BLVP_BILL</code>	Der Notenprüfer hat eine Note akzeptiert. Die Position der Noten hängt davon ab, ob der Prüfer Escrow beherrscht und ob es für die Note erlaubt wurde
<code>BLVP_JAMMED</code>	Eine Note hat sich im Prüfer verklemmt
<code>BLVP_RESET</code>	Der Notenprüfer hat zuletzt eine Reset (Hard- oder Software) ausgeführt.

BLVP_REJECTED	Eine Note wurde abgewiesen.
BLVP_BUSY	Der Notenprüfer ist intern beschäftigt.
BLVP_TIMEOUT	Fehler: Der Notenwechsler hat zu lange nicht auf Poll reagiert.
BLVP_UNKNOWN	Unbekannter Fehler
BLVP_MDBERR	Es ist ein MDB Fehler aufgetreten, Fehlercode in mdberr von BLVPollStatus

Ein Status von „0“ zeigt an, daß nichts zu vermelden war und keine Reaktionen erforderlich sind. Bei BLVP_BILL könnte z.B. eine Windows-Message verschickt werden, die anderen Programmteilen anzeigt, daß der Kredit erhöht werden muß.

Auf BLVP_TIMEOUT sollte reagiert werden, indem der Notenprüfer und alle eventuell noch angeschlossenen Geräte geschlossen werden, ein `ResetMDBDevices` ausgeführt, alle wieder geöffnet, ausgelesen und gesetzt werden. Tritt danach der Fehler wieder auf, ist abzubrechen und der Notenprüfer als defekt anzusehen.

Die übrigen Status erfordern im normalen Betrieb keine Reaktionen durch die Software bzw. deuten auf grundsätzliche Fehler in Hard- oder Software hin.

```
__stdcall int GetBLVLastPollStat( stuct BLVPollStatus* bpp );
```

Liefert detaillierte Informationen über das Ergebnis des letzten Polls.

Ansonsten gilt im Wesentlichen das unter `PollBLV()` zu den Fehlerzuständen gesagte.

2.2.4.7. Sonstige Funktionen

```
__stdcall int ResetBLV();
```

Im Notenprüfer wird ein Software-Reset ausgelöst.

3. Beispiel für die Behandlung eines Münzprüfers

Im folgenden soll kurz an einem Beispiel das Vorgehen bei der Behandlung eines Münzprüfers erläutert werden, wobei davon ausgegangen wird, daß für die Sortiersteuerung die Default-Werte beibehalten werden.

1. Initialisieren des MDB mit `OpenMDB()`. Zurücksetzen aller Geräte am MDB mit `ResetMDBDevices()`.
2. Öffnen des Münzprüfers mit `OpenEMP()`.
3. Holen der Informationen zum Münzprüfer mit `GetEMPInfo()`.
4. Ermitteln der Münzwerte aus der Struktur `EMPInfo`. Beispielsweise könnte der Inhalt von `coin_value` folgendermaßen aussehen:

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Wert	0,00	2,00	1,00	0,50	0,50	0,10	0,00	2,00	1,00	0,50	0,00	0,00	0,00	0,00	0,00	0,00

Wobei dies lediglich ein Beispiel ist, die konkrete Belegung kann variieren und muß in jedem Fall ermittelt werden! Dabei können Münzwerte durchaus mehrfach angegeben sein, was bei der Münzfreigabe zu berücksichtigen ist.

Die Mehrfachbelegung von Münzen dient in speziellen Anwendungsfällen dazu, eine Münzart mit unterschiedlich weit eingestellten Parametersätzen für die Annahme zu verarbeiten und ggf. die weiter eingestellten zu sperren. Im Normalfall werden *alle Belegungen* einer Münze freigegeben.

5. Holen der aktuellen Münzsteuerung mit `GetEMPCurrent()`.
6. Einstellen der Münzfreigabe mittels `SetEMPCurrent()`. Sollen z.B. bei obiger Münzzuordnung nur 2,00€, 1,00€ und 0,50€ Münzen akzeptiert werden, so muß `unlock` in der Struktur `EMPCurrent` folgendermaßen belegt sein:

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Wert	0	1	1	1	0	0	0	1	1	0	0	0	0	0	0	0

Die Felder `inbox`, `tube` und `cashbox` werden unverändert gelassen. Damit werden für die Sortiersteuerung die Default-Werte beibehalten.

7. Sollen Münzen angenommen werden, ist ein zyklisches Pollen durchzuführen, beispielsweise über einen Windows-Timer:

```

...
SetTimer ( hWnd, 0, 200, &DoPoll );
...
TIMERPROC DoPoll() {
    int pres;
    pres = PolleMP();
    if (pres & EMPP_COIN) ...;
    if (pres & EMPP_RETURN) ...;}

```

8. Sollen keine Münzen mehr angenommen werden, ist das Pollen einzustellen und der Münzprüfer mittels `SetEMPCurrent()` zu sperren, indem das Feld `unlock` komplett auf "0" gesetzt wird.
Der Kredit sollte bei der letzten Abfrage mittels `GetEMPCredit(null, 1)` gelöscht werden.
9. Für eine erneute Münzannahme ist der Ablauf ab Pkt. 3. zu wiederholen.
10. Bei Beendigung der Applikation Münzprüfer wieder schließen mit `CloseEMP()` und Freigeben des COM-Ports mit `CloseMDB()`.