Software libraries WinMDB32.dll for Windows 98/ME/2000/XP and libmdb.so for Linux

for communication with devices with Multi Drop Bus protocol

Version 2.01

supports:

electronic coin selectors
electronic change givers
bill validators

As of 18th March 2003

Table of contents

1. System requirements

A serial adapter COM200 or an USB adapter COM300 from *w*h Münzprüfer is required for connection of the MDB devices.
All coin selectors of wh Münzprüfer Berlin GmbH that may be connected to the Multi Drop Bus (subsequently called MDB) and all conforming bill validators and change givers with MDB are supported.

2. Description of the software library

2.1. General notes

The software is a 32-bit windows DLL which runs with Windows® 98, ME, 2000 and XP or a shared library for Linux. Before utilisation the software must be copied to the systems' directory of the corresponding Windows version resp. to the User Lib library.
All functions belong to the type `int` and return an error code. This error code can either be forwarded by the protocol layer or generated by the command layer.
Functions which return their results in the form of a structure always expect a pointer on an *existing* structure which they fill with data.
The integration of DLL/shared library into personal applications is supported by the following files annotated in detail: `mdb.h` for C(++) or `XMDB32_Defines.pas` and `XMDB_Interface.pas` for Pascal/Delphi. All error codes and structural definitions can also be found in these files.
All functions are only returned after all actions have been carried out by the peripheral or after a discontinuation due to an error. Depending on the amount of data, the communication alone takes up between 5 and 50ms. To this, the internal processing times of the devices must also be added. The following kinds of procedures are particularly time-consuming:

- The reset of all devices connected to the MDB: the time of execution alone is approx. 200ms. An even longer time span may elapse until the different devices are ready for operation.
- The initiation of devices takes from 100ms to some seconds in change givers and bank note readers. This is due to mechanical procedures and properties.
- The query and setting of information about devices takes in excess 100ms because of the access to EEPROMs.
- The returning of coins out of a change giver takes up several seconds depending on the amount of coins.
- The return of a bank note out of a bill validator normally takes up to 1second.

These periods of time are only approximate values and may vary considerably, even within a type of device, depending on the manufacturer.

All MDB devices described here give a country identification which indicates for what currency they are set. Where several devices are connected, the library does not examine whether their country identification is identical. If necessary, this must be examined by means of the specific application.

2.2. Description of the function

2.2.1.1. Open and close MDB

`_stdcall int OpenMDB( int com_nr );` *(Windows)*
The COM-Port stated is opened and initialized for MDB. The successful execution of this function is a prerequisite for all further actions.
If the USB version COM300 is in use, the COM-port number can be indicated as “0”. Despite of the installed virtual port, the USB adapter will always be found.

`_stdcall int OpenMDB_str( int tty_name );` *(Linux)*
The tty device stated is opened and initialized for MDB. The successful execution of this function is a prerequisite for all further actions.

`_stdcall int CloseMDB();`
The COM-Port/tty device which has been opened for the MDB is closed and released again.

2.2.1.2. Reset all devices connected with the MDB

`_stdcall int ResetMDBDevices();`
The hardware of all connected devices is reset. This may be executed directly after the initialisation of the MDB in order to set all devices into a defined condition.

2.2.1.3. Setting of switching lines

`_stdcall int SetAuxPort ( char stat, char msk );`
By using this function, four additional switching lines may be set at the card provided that the hardware is implemented.
`Stat` indicates the condition requested and `msk` indicates whether the corresponding bit is to be affected.

2.2.2. The handling of a coin selector

2.2.2.1. Open and close coin selector

`_stdcall int OpenEMP();`
A coin selector is opened and identified in the previously initialised MDB.

`_stdcall int CloseEMP();`
An open coin selector is closed.

2.2.2.2. Reading out of data from coin selector

`_stdcall int GetEMPInfo ( struct EMPInfo* eip );`
All information relevant for coin processing is read out from the coin selector. The structure of which `eip` points to contains information about the coin selector. The assignment of the money value to the coin numbers is particularly important because it is needed when certain coins are to be blocked. For coin selectors with sorting shafts the information about the default setting of the sorting shafts might also be of importance.
The country identification `country` contains the international dialling code of the country for which the coin selector has been installed, e.g. 0x0044 for Great Britain or an ISO 4217 currency code e.g. 0x1978 for the Euro.
The indication of the decimal place of the currency (`decimals`) may be useful for optimum formatting of the amounts' display.

2.2.2.3. Setting and inquiring of coin release and sorting control

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

The current settings for the coin release and for the sorting control are read out of the coin selector and are entered into the structure to which `ecp` points. The meaning of the data is identical to `SetEMPCurrent()` (see below).

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

The structure to which `ecp` points contains the information for the coin release and for the sorting control:

`cashbox`	states the number of the cash box shaft
`unlock[x]`	"1" releases the coin no. x , "0" blocks the coin
`incbox[x]`	"1" leads the coin no. x to the cash box shaft irrespective of the pre-set sorting shaft
`tube[x]`	states the sorting shaft for coin no. x

The `cashbox` and the `incbox` are of no relevance for coin selectors without sorting shafts. If in doubt, all values for `tube` and the value for `cashbox` should first be read by means of `SetEMPCurrent` and should not be changed.
After the machine has been switched on, it has the following initial setting: all coins are blocked, the diversion into the main cash box for all coins is not active and the sorting shaft assignment is set to default values.
Should the need arise to change the setting for particular coins, it is advisable to read out the current setting by means of `GetEMPCurrent()`(see below) and to change the required values in the structure before passing on this structure to `SetEMPCurrent()`.

2.2.2.4. Polling of coin selectors and inquiring the last poll status

```
_stdcall int PollEMP();
```

If the coin selector shall accept coins, this function has to be called up cyclically, approximately every 25 to 200 ms. By means of this function existing information is read out for the last coin processing. The return value generally states the result of the polling and is identical with the contents of `status` in `EMPPollStatus`. The status is a bit vector, for which the following masks are defined:

`EMPP_COIN`	The coin selector has accepted a coin; the credit has been increased.
`EMPP_RESET`	The last action performed by the coin selector was a reset (hard- or software).
`EMPP_RETURN`	The reject lever at the coin selector has been operated.
`EMPP_REJECT`	A coin has been rejected.
`EMPP_NOANSWER`	The coin selector has not answered. The fact that the coin selector does not answer to every poll during the coin processing does not represent an error.
`EMPP_TIMEOUT`	Error: The coin selector did not react to the poll for a too long period of time.
`EMPP_UNKNOWN`	Unknown error
`EMPP_MDBERR`	An MDB error has occurred; Error code in `mdberr` from `EMPPollStatus.`

The status "0“ indicates that there was nothing to be reported and that no reactions are necessary. By means of `EMPP_COIN` a windows message could be sent which informs other parts of the programme that the credit has been adapted.
The reaction to `EMPP_RETURN` could lead to e.g. returning a bill in escrow position when an additional bill validator is connected. When only one coin selector is connected, this status does not require any further actions.
You should react to `EMPP_TIMEOUT` as follows: close the coin selector and all other devices which are possibly still connected; carry out a `ResetMDBDevices`, re-open all devices, read them out and set them. Should this error occur again, break off the procedure: the coin selector is defect.
The other status requires no reaction from the software during normal operation resp. they indicate basic errors in the soft- or hardware.

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

Delivers detailed information about the result of the last poll. What has been said about the error status under `PollEMP()` largely applies to all other cases.

2.2.2.5. Further functions

`_stdcall int ResetEMP();`
A software reset is triggered off in the coin selector.

2.2.3. Handling of a change giver

2.2.3.1. Opening and closing of change giver

`_stdcall int OpenCHG();`
A change giver connected with the previously initialised MDB is opened and identified.

`_stdcall int CloseCHG();`
An open change giver is closed.

2.2.3.2. Read out data from the change giver

`_stdcall int GetCHGInfo ( struct CHGInfo* cip );`
All necessary information concerning the coin processing is read out from the change giver. The structure to which `cip` points is filled with information about the change giver.
The assignment of the money value to the coin numbers `(coin_value)`is particularly important because it is needed when certain coins are to be blocked or released. What coins may be routed into tubes `(tube_coin)` and the smallest dispensable coin `(least_coin)`are further important pieces of information. This information is required together with details about the contents of the tubes in order to decide whether a particular amount is to be dispensed.
The country identification `country` contains the international dialling code of the country for which the change giver has been installed, e.g. 0x0044 for Great Britain or an ISO 4217 currency code e.g. 0x1978 for the Euro.
The indication of the decimal place of the currency `(decimals)` may be useful for the optimal formatting of the display of amounts.

2.2.3.3. Obtain tube status

`_stdcall int SetCHGTubeStat ( struct CHGTubeStat* ctp );`
The filling levels and the conditions of the tubes are read and transferred into the structure. In detail this comprises the following information: the number of coins per coin type `(coinctr)`, whether the corresponding tube has reached its maximal filling level `(tube_full)` and whether the tube is possibly blocked `(tube_err)`. This information is required when the application is to decide by itself on the composition of coins during the dispensing operation.

2.2.3.4. Set coin release

`_stdcall int SetCHGCurrent ( struct CHGCurrent* ccp );`
The display to which `ccp` points, contains the coin release. A value !=0 releases the corresponding coin. After the switching on or after the hardware reset all coins are initially blocked. A software reset, however, does not change the coin release!

2.2.3.5. Dispense coins

`_stdcall int DispenseCHGCoins ( CHGDispense* dcp );`
The number of the type of coin specified in the display is dispensed, provided that this is possible. After the return, the array shows the actual number per dispensed coin.
Before this function is called up, it is advisable to determine what coins may be dispensed `(tube_coin` in `CHGInfo)` and to inquire the status (filling level and possibly error) of the corresponding tube by means of `GetCHGTubeStat`. However, this is not usually necessary since the dispense may be performed in a much more convenient way by means of the function `DispenseCHGValue` (see below).

2.2.3.6. Dispensing a value

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

A desired value shall be dispensed. The combination of coins is effected by the function. The parameter "force" influences the performance if the number of coins in the tubes is not sufficient: Is it set to "0", in this case nothing will be dispensed, is it set to 1, pay-out is made as far as the tubes have capacity. In any case the actual amount is disbursed in "*dvp".

2.2.3.7. Poll change giver and inquire the last polling status

`_stdcall int PollCHG();`
If the coin selector is to accept coins this function has to be called up cyclically approximately every 25 to 200 ms. By means of this function existing information is read out for the last coin processing. The return value generally states the result of the polling and is identical with the contents of `status` in `CHGPollStatus`. The status is a bit vector, for which the following masks are defined:

`CHGP_DEPOSIT`	The coin selector has accepted a coin; the filling levels of the tubes have changed.
`CHGP_DISPENSE`	A coin has been dispensed manually by means of the control keys; the filling levels of the tubes have changed.
`CHGP_RESET`	The last action performed by the coin selector was a reset (hard- or software).
`CHGP_RETURN`	The reject lever at the coin selector has been operated
`CHGP_REJECT`	A coin has been rejected.
`CHGP_BUSY`	The change giver is busy internally
`CHGP_POTBUSY`	The change giver is busy doing the dispense
`CHGP_TIMEOUT`	Error: too long a time span has elapsed since the coin selector reacted to the poll.
`CHGP_UNKNOWN`	Unknown error
`CHGP_MDBERR`	An MDB error has occurred; Error code in `mdberr` from `BLVPoll-Status`

The status "0" indicates that there was nothing to be reported and that no reactions are necessary. E.g. under `CHGP_DEPOSIT` a windows message could be sent to inform the other parts of the programme that the credit has be adapted.
The reaction to `CHGP_RETURN` could consist in e.g. returning a bill in Escrow-Position when an additional bill validator is connected and, if necessary, in dispensing the remaining credit in coins.
You should react to `CHGP_TIMEOUT` as follows: close the coin selector and all other devices which are possibly still connected; carry out a `ResetMDBDevices`, re-open all devices, read them out and set them. Bear in mind that there is no error function reported should the change giver close communication during operation via the keys.
The other status requires no reaction from the software during normal operation or suggest basic mistakes in the soft- or hardware.

`_stdcall int GetCHGLastPollStat( stuct CHGPollStatus* cpp );`
Delivers detailed information about the result of the last poll. What has been said about the error status under `PollCHG()` largely applies to all other cases.

2.2.3.8. Further functions

`_stdcall int ResetCHG();`
A software reset is triggered off in the change giver.

2.2.4. Handling of a bill validator

2.2.4.1. Open and close bill validator

`_stdcall int OpenBLV();`
A bill validator is opened and identified with the previously initialised MDB.

`_stdcall int CloseBLV();`
An open bill validator is closed.

2.2.4.2. Read out data from bill validator

`_stdcall int GetBLVInfo ( struct BLVInfo* bip );`
All information necessary for the processing of bills is read out. The structure, to which `bip` points, contains information about the bill validator.
The assignment of the money value to the bill numbers `(bill_value)`is particularly important because it is needed when certain bills are to be blocked or released. Another important information is whether the bill validator has Escrow `(escrowcap),` i.e. whether the bill validator is able to hold a bill in a position from which it may not only be stacked but also be returned again.
The country identification `country` contains the international dialling code of the country for which the bill validator has been installed, e.g. 0x0044 for Great Britain or an ISO 4217 currency code e.g. 0x1978 for the Euro.
The indication of the decimal place of the currency (`decimals)` may be useful for the optimal formatting of the display of amounts.

2.2.4.3. Read out stacker status

`_stdcall int GetBLVSTacker ( struct BLVStacker* bsp );`
The contents of the stacker are read out `(bills)` and it is reported whether the stacker is possibly full `(full).` Should the stacker be full, a demand to empty it could be emitted.

2.2.4.4. Setting of bill control

`_stdcall int SetBLVCurrent ( struct BLVCurrent* bcp );`
The structure to which `bcp` points contains the bill control. The corresponding bill may be released `(unlock)` and enabled for the escrow `(escrenab).` See points 2.2.4.2. and 2.2.4.5 concerning the function of Escrow.

When the escrow is allowed for a bill, the credit shall be already increased when an accepted bill is in thie stacker position. The credit is decreased again when the bill is returned and the credit remains unchanged when the bill is stacked. When the escrow is blocked, an accepted bill is instantly stacked and the credit is increased.
Without the escrow, a return of money for an accepted bill can only be effected in the form of coins via a connected change giver, such as in the case of a vending operation being discontinued.

2.2.4.5. Withdraw and return bills

`_stdcall int ReturnBLVBill;`
When a bill is in the escrow position, it is returned and the credit is decreased by the corresponding amount when the operation has been finished successfully.
When a vending operation is discontinued and the money is returned in a system with a bill validator and change giver you could proceed in the following manner:

- call up `ReturnBLVBill` – a bill which is possibly in the escrow position is returned and the credit is decreased by the corresponding amount.
- call up `DispenseCHGValue` in order to return remaining credit

`_stdcall int StackBLVBill;`
When a bill is in the escrow position, it is withdrawn. Since the credit has already been increased previously it remains unchanged.

Both functions may be called up "on spec". The handling of the bill validator and of the credit is carried out correctly in both cases according to the recorded internal setting and the position of the corresponding bill. A return value `BLV_ESCRERR` or `BLV_STACKERR` can be ignored. These values indicate that no bill is in escrow position and that the action could not be carried out for this reason.

2.2.4.6. Poll bill validator and inquire the last poll status

`_stdcall int PollBLV();`
If the bill validator is to accept bills, this function has to be called up cyclically approximately every 25 to 200 ms. By means of this function existing information are read out for the last bill processing. The return value generally states the result of the polling and is identical with the contents of `status` in `CHGPollStatus`. The status is a bit vektor, for which the following masks are defined:

`BLVP_BILL`	The bill validator has accepted a bill. The position of the bill depends on whether the validator has Escrow and whether Escrow has been enabled for the bill.
`BLVP_JAMMED`	A bill got stuck in the reader
`BLVP_RESET`	The last action performed by the bank note reader was a reset (hard- or software).
`BLVP_REJECTED`	A bill has been rejected.
`BLVP_BUSY`	The bank note reader is busy internally.
`BLVP_TIMEOUT`	Error: too long a time span has elapsed since the bill validator reacted to the poll.
`BLVP_UNKNOWN`	Unknown error
`BLVP_MDBERR`	An MDB error has occurred; Error code in `mdberr` from `BLVPollStatus`

The status "0“ indicates that there was nothing to be reported and that no reactions are necessary. E.g. under `BLVP_BILL` a windows message could be sent off to inform the other parts of programme that the credit has to be increased.

You should react to `BLVP _TIMEOUT` as follows: close the coin selector and all other devices which are possibly still connected; carry out a `ResetMDBDevices`, re-open all devices, read them out and set them. Should the error occurs again, break off the procedure: the bill validator is defect.
The other status require no reaction from the software during normal operation or they suggest basic mistakes in the soft- or hardware.

`int GetBLVLastPollStat( stuct BLVPollStatus* bpp );`
Delivers detailed information about the result of the last poll.
What has been said about the error status under `PollCHG()`largely applies to all other cases.

2.2.4.7. Further functions

`_stdcall int ResetBLV();`
A software reset is triggered off in the bill validator.

3. Example for the handling of a coin selector

In the following. an example is given in order to explain the handling of a coin selector. It is assumed that the default values are kept for the sorting control.

1. Initialize MDB by means of `OpenMDB()/OpenMDB_str().` Reset all devices connected to the MDB by means of `ResetMDBDevices().`

2. Open the coin selector by means of `OpenEMP().`

3. Get the information to the coin selector by means of `GetEMPInfo().`

4. Determine the coin values out of the structure `EMPInfo`. Example: the contents of `coin_value` could look like this:

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

   This is only an example and the definitive configuration may vary and must be determined in all cases. It may occur that coin values are stated several times. This must be taken into consideration as far as the coin release is concerned.
   In special types of application the multiple entry is useful for processing a type of coin with sets of parameters that are set with a varying scope for the acceptance of coins. Furthermore, parameters that are set with a broader scope may be blocked. Normally, *all* entries of a coin are enabled.

5. Get the current coin control by means of `GetEMPCurrent().`

6. Setting of the coin release by means of `SetEMPCurrent().` Example: Where only coins of 2 EUR, 1 EUR and 0,50 EUR are to be accepted in the above-mentioned coin assignment, `unlock` must be configured in the structure `EMPCurrent` as follows:

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

   The fields `incbox`, `tube` and `cashbox` are not changed. This makes sure that the default values for the sorting control are kept.

7. In case coins are to be accepted, a cyclic polling must be carried out, e.g. via a windows timer:

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

8. When no more coins are to be accepted, stop the polling and block the coin selector by means of `SetEMPCurrent()`. For this purpose set the whole field unlock to "0".
   The credit should be erased by means of `GetEMPCredit( null, 1 )` during the last inquiry.

10. Repeat the procedure beginning with point 3 for a new acceptance of coins.

11. After the application has been finished, close the coin selector again by means of `CloseEMP()` and release the `COM-Ports` by means of `CloseMDB().`