

## 4.1 Software Betriebe

Teilprojektleiter:	Dr.-Ing. Stefan Böttinger
Bearbeiter:	Dipl.-Ing. Kai Oetzel Dipl.-Ing. Ludger Autermann Dipl.-Ing. Volker Brill
Einrichtung:	AGROCOM GmbH & Co. Agrarsystem KG

### 4.1.1 Ursprüngliche Aufgabenstellung

Für die Umsetzung von Precision Agriculture auf der Betriebsebene sollen Softwarelösungen realisiert werden die hinsichtlich der Akzeptanz und der Durchgängigkeit eine neue Qualitätsstufe erreichen. Für die Akzeptanz ist insbesondere die Benutzerführung und das Vorhandensein von Schnittstellen zu weiteren Softwareprodukten relevant. Die Benutzerführung soll die Qualifikation der Anwender und die geringe Häufigkeit der Softwarenutzung berücksichtigen. Schnittstellen zu weiteren PC-Produkten und zu Mobilcomputern sollen einerseits vorhandene und zukünftige Standards, andererseits marktgängige und sich in Entwicklung befindliche Produkte berücksichtigen. Hierbei handelt es sich, um Besonders hervorzuheben, sich in Entwicklung befindliche Expertenmodule, wie sie im Rahmen weiterer Teilprojekte entwickelt werden. Für diese Expertenmodule ist eine allgemeingültige Schnittstelle zu erarbeiten, die prinzipiell von allen Herstellern von PC-Softwares für die Landwirtschaft integriert werden kann. Hiermit ist dann diese Lösungen nicht nur an spezifische Produkte gebunden, sondern es ist eine offene Lösung für alle potentiellen Anwender realisiert.

### 4.1.2 Zielerfüllung/ -erreichung

Im Projektzeitraum konnte die Benutzerführung, insbesondere in Software-Bereichen die von den preagro-Demobetrieben eingesetzt wurden, verbessert werden. Für den Datenaustausch mit im Gesamtprojekt verwendeten Systemen (PC-Software, Bordcomputer, Ertragskartierungssysteme etc.) wurden weitere Schnittstellen entwickelt und somit die Durchgängigkeit der Softwarelösungen erreicht. Speziell im Bereich des Datenaustauschs wurde die Benutzerführung angepasst. Mit der PC-Software AGRO-MAP Basic konnte in Verbindung mit der Ackerschlagskartei Ackerdat der Fa. Agrocom parallel zum Projekt eine teilflächenspezifische Ackerschlagskartei realisiert werden. Hiermit steht Anwendern nun die Möglichkeit zur teilschlagspezifischen Deckungsbeitragsrechnung offen.

Es wurde eine allgemeingültige Schnittstelle zwischen beliebigen PC-Softwares für precision agriculture und Expertenmodulen entwickelt. Beispielhaft wurde an diese Schnittstelle in AGRO-MAP Basic ein Expertenmodul für die Aussaat von Winterweizen angebunden und die Funktionalität nachgewiesen. Das Expertenmodul Winterweizenaussaat wurde in Absprache mit der Gesamtprojektleitung zusätzlich in diesem Teilprojekt implementiert und hierbei die u.a. mit

den Wissenschaftlern der Modulalgorithmen erarbeiteten Richtlinien für das Benutzerinterface realisiert.

Im Projektzeitraum trat vermehrt die Nachfrage nach der aktiven Gestaltung der Schnittstellenstandardisierung (LBS DIN9684, ISO 11783) auf. Hierzu sind einerseits vielfältige Absprachen mit anderen Landtechnik-Herstellern erfolgt und andererseits wurde an der Ausarbeitung der neuen ISO 11783 mitgearbeitet. Zur Erleichterung und Beschleunigung der Einführung der ISO 11783 wurden verschiedene Aktivitäten durchgeführt. Besonders hervorzuheben ist hierbei die Mitarbeit bei der Initiierung der ISO-BUS-Initiative. Aktuell gestaltet sich die Umsetzung in der Praxis wegen teilweiser zögerlichem Verhalten einzelner Firmen langsamer als gewünscht. Zudem müssen die vorhandenen Geräte bei den Landwirten und Lohnunternehmen mit „Alt“-Technik in den Umsetzungsstrategien der Firmen berücksichtigt werden.

### **4.1.3 Einleitung und Problemstellung**

Die erfolgreiche Umsetzung von precision agriculture in der Praxis ist von mehreren, 100%ig zu erfüllenden Faktoren abhängig. Neben z.B. nachweisbaren ökonomischen Vorteilen ist die Handhabbarkeit der diversen Werkzeuge für precision agriculture elementar. Bordcomputer auf Schleppern und selbstfahrenden Landmaschinen und die PC-Software im Büro zählen zu den wichtigsten Werkzeugen. Sie müssen von den aktuell in der Landwirtschaft beschäftigten Personen bedient und verstanden werden. Hierbei müssen die unterschiedlichsten Ausbildungs- und Erfahrungsniveaus sowie die auch daraus entstehenden individuellen fachlichen Tiefen der Fragestellungen berücksichtigt werden. Neben diesen, schwerpunktmäßig die Benutzeroberfläche und Benutzerführung betreffenden Aufgabenstellungen muss das Zusammenspiel der unterschiedlichsten Komponenten (Bordcomputer, Anbaugeräte, PC-Software, Expertenmodule etc.) gewährleistet sein. Schwierigkeiten in diesem Bereich führen zu Bedienproblemen und daraus folgend zu einer abnehmenden Akzeptanz in der Praxis. U.a. dieser Problembereich ist bei der Begutachtung des Zwischenberichts 2000 hervorgehoben worden.

Allgemeine und national wie international genutzte Schnittstellen, wie z.B. DIN 9684 und ISO 11783, können nicht nur über ein Entwicklungsprojekt erarbeitet und umgesetzt werden. Zusätzlich ist die aktive Mitarbeit bei internationalen Normierungsbemühungen und bei Aktivitäten zur Förderung der Umsetzung neuer Normen nötig.

Innerhalb des Gesamtprojekts galt es zu klären, wie weit die Aufgabenbereiche der Entwickler von Expertenmodulen und der Entwickler von Managementsoftware gehen. Diese Klärung wurde für dieses TP konstruktiv durchgeführt und resultiert in der Erweiterung des Engagements von der Schnittstellendefinition und –erarbeitung innerhalb der Managementsoftware hin zu der zusätzlichen Codierung eines Expertenmoduls gemäß den inhaltlichen Vorgaben der Modulentwickler und der Schnittstellenvorgaben.

#### 4.1.4 Methoden / Vorgehensweisen

Vier Werkzeuge wurden für die Bearbeitung der Aufgaben im Teilprojekt hauptsächlich eingesetzt: die Analyse des Benutzerverhaltens, die Analyse des Marktes, die aktive Gestaltung von Standards und die Zusammenarbeit und Absprachen mit den anderen Teilprojekten.

Die Analyse des Benutzerverhaltens erfolgte entweder aus direkten oder indirekten Rückmeldungen. Direkte Rückmeldungen liefern die Nutzer der Werkzeuge an die Hotline oder die Entwicklungsmitarbeiter der Firma Agrocom. Durch gezielte Befragungen bei User-Treffen und/oder Schulungen für geübte Nutzer (advanced user) wurden ebenfalls direkte Rückmeldungen eingeholt. Indirekte Rückmeldungen erfolgen über Mittelsmänner wie z.B. Vertriebsmitarbeiter und Mitarbeiter des Verbundprojektes. Diese Informationen wurden gesammelt und daraus nötige Weiter- oder Neuentwicklungen für die Benutzer-Schnittstellen und für die Schnittstellen zwischen den einzelnen precision agriculture-Werkzeugen und -Komponenten abgeleitet.

Durch die Analyse des aktuellen Angebots an Werkzeugen und Komponenten und zusätzliche Marktinformationen konnten Neu- oder Weiterentwicklungen bei den Schnittstellen unterschiedlichster Anbieter erkannt bzw. frühzeitig Änderungen vorhergesagt werden.

Für die meisten heutigen und, soweit heute vorhersehbar, die zukünftigen Anwendungen im Bereich precision agriculture sind die Schnittstellen zwischen den Werkzeugen und Komponenten genormt bzw. befinden sich im Normungsprozess. Diese Arbeiten wurden aus Projektinteressen aktiv mitbetrieben um allgemeinen Interessen unterschiedlichster Firmen an proprietären Schnittstellen durch möglichst internationale Standards entgegenzuarbeiten bzw. vorzubeugen.

Durch Absprachen und Diskussionen insbesondere mit dem TP III 1b wurden projektspezifische Schnittstellen zwischen der „Software Betrieb“ und der „Software Lohnunternehmen“ erkannt und Realisierungsmöglichkeiten erarbeitet. Die unterschiedlichen Ansprüche der verschiedenen Nutzergruppen an die Funktionalitäten der jeweiligen Programme sowie das Zusammenspiel der Programme von Dienstleister und Auftraggeber erforderte ein abgestimmtes, koordiniertes Vorgehen bei den Software-Weiterentwicklungen.

Für die Einbindung von Expertenmodulen, wie sie von verschiedenen Teilprojekten entwickelt wurden und werden, an die PC-Softwares unterschiedlichster Hersteller wurde eine allgemein verwendbare Schnittstelle erarbeitet und beispielhaft mit der Realisierung des Aussaatmoduls für Winterweizen implementiert. Bei der Gestaltung des User-Interfaces des Aussaatmoduls sind alle Erkenntnisse bezüglich der Bedienfreundlichkeit und der unterschiedlichsten Anforderungen von unerfahrenen bis erfahrenen PC-Benutzern sowie Pflanzenbauern an die Verständlichkeit der Methoden eines Expertenmoduls umgesetzt.

## 4.1.5 Ergebnisse und Diskussion

### 4.1.5.1 Schnittstellen-Weiterentwicklung PC-Software

Für den Datentransfer zwischen den Softwareprodukten zweier Hersteller für das Büro kann prinzipiell die DIN 9684.5 genutzt werden. Von den unterschiedlichen Softwareherstellern werden meistens aber proprietäre Datenformate bzw. Abwandlungen dieser Norm eingesetzt. Aktuell werden deshalb von den Software-Anbietern zum Datenaustausch zwischen den einzelnen Büro-Softwareprodukten die Datentransferdateien zu den Bordcomputern genutzt. Für diese Dateiformate liegen häufig Importfilter vor. Allerdings sind in diesen Dateien nur Informationen enthalten, die von den Bordcomputern genutzt werden können. Für den Datenaustausch zwischen den Büro-Softwareprodukten gelten aber höhere Anforderungen: die Daten dürfen auf dieser Austauschenebene nicht auf die Machbarkeit der entsprechenden Bordcomputer und Anbaugeräte reduziert werden. Deshalb wurde gemeinsam in Zusammenarbeit mit dem Teilprojekt III 1b eine Lösung für den Datenaustausch zwischen den Softwareprodukten der Teilprojekte erarbeitet und realisiert.

Beide Produkte, SSTools und AGRO-MAP Basic, nutzen ähnlich gestaltete Softwareteile zur geostatistischen Verrechnung von Daten und zur Berechnung von Contour-Karten für die Anzeige. Für hierfür erzeugte interne Zwischenformate konnte gemeinsam eine Im- und Exportschnittstelle gestaltet werden. Mit Hilfe dieser Schnittstelle können in AGRO-MAP Basic oder SSTools erzeugte Karten von Daten z.B. von Mähdreschern mit Ertragskartierung auf Datenebene in das jeweilige andere Produkt übertragen werden, Abb. 4.1-1. Das Programm SSTools bietet einen stark ausgebauten Planungsteil mit dessen Hilfe für z.B. die Applikationsplanung der Herbstsaat 1999 des Projektes realisiert wurde. Die hierdurch erzeugten Applikationskarten konnten mittels der genannten Schnittstelle in AGRO-MAP Basic importiert und von dort aus als Maschinensteuerungsdateien für das ACT auf den Praxisbetrieben exportiert werden, Abb. 4.1-2. Dieses Feature ist schwerpunktmäßig auf dem Betrieb Hess/Thumbby eingesetzt und überprüft worden.

Die Gestaltung des User-Interfaces für den Import von Schlaggrenzen und von Wertekarten wie z.B. Applikationskarten ist in Abb. 4.1-3 und Abb. 4.1-4 dargestellt. Der Sourcecode ist im Anhang gelistet.

Die Notwendigkeit zum Import von weiteren Fremdformaten bestand im Rahmen des Projektes besonders stark bzw. nimmt durch den Einsatz und die Erprobung unterschiedlichster Komponenten zu. Zu Unterstützung dieser Anforderung wurden weitere Im- und Exportschnittstellen in der PC-Software realisiert. Diese sind im Einzelnen

- AgLeader \*.YLD-Format (Import)
- RDS-Ertragsdaten \*.X01, \*.Y01 (Im-/Export)
- Importexperte für Analyseergebnisse (ASCII-Datei oder DBF-File)

Auf den DLG-Feldtagen 2001 wurde erstmals die preagro-Version von AGRO-MAP Basic auf dem Stand des Verbundprojektes und der Fa. Agrocom präsentiert. In dieser preagro-Version

sind neben dem Hinweis auf die Förderung durch das BMBF speziell die für das Verbundprojekt erstellten Schnittstellen besonders hervorgehoben.

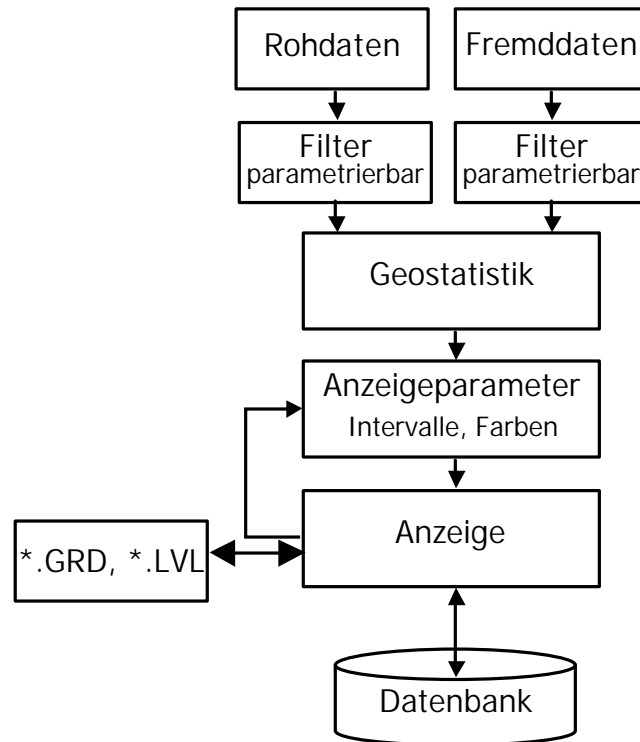


Abb. 4.1-1: Datenfluss Messwertkarte AGRO-MAP Basic mit Im- und Export-Möglichkeiten

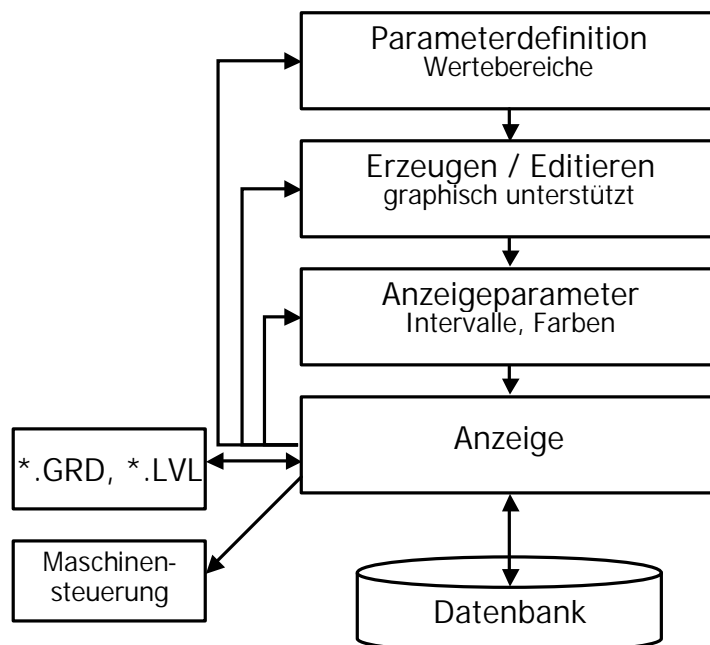
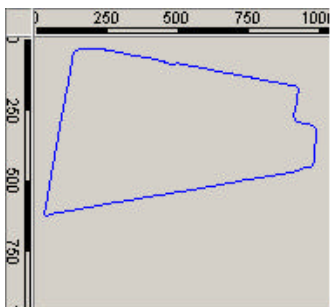
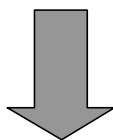
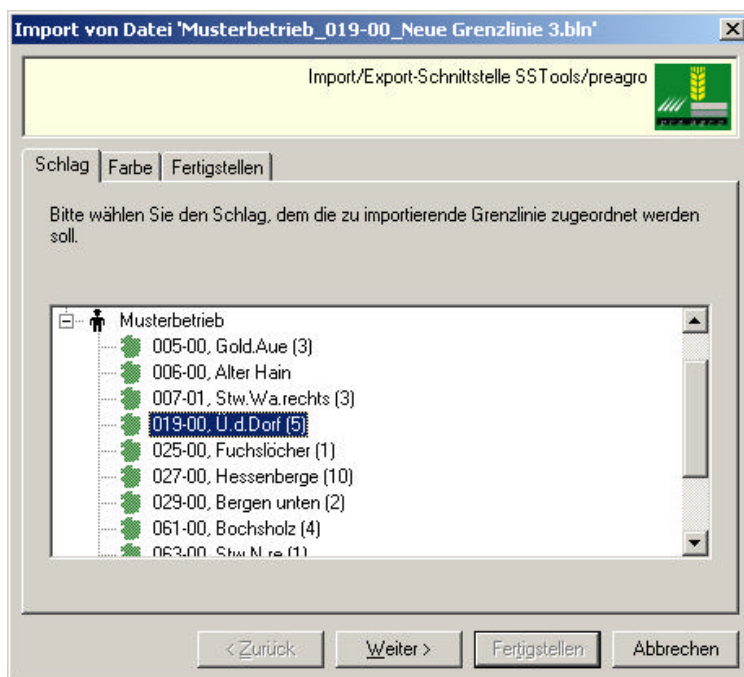


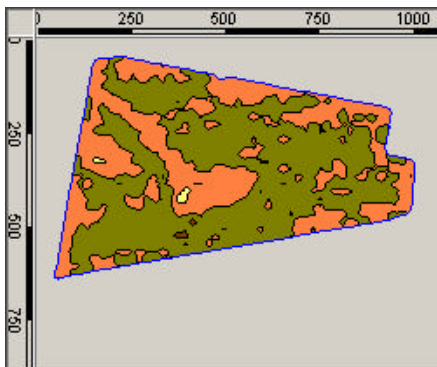
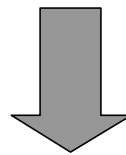
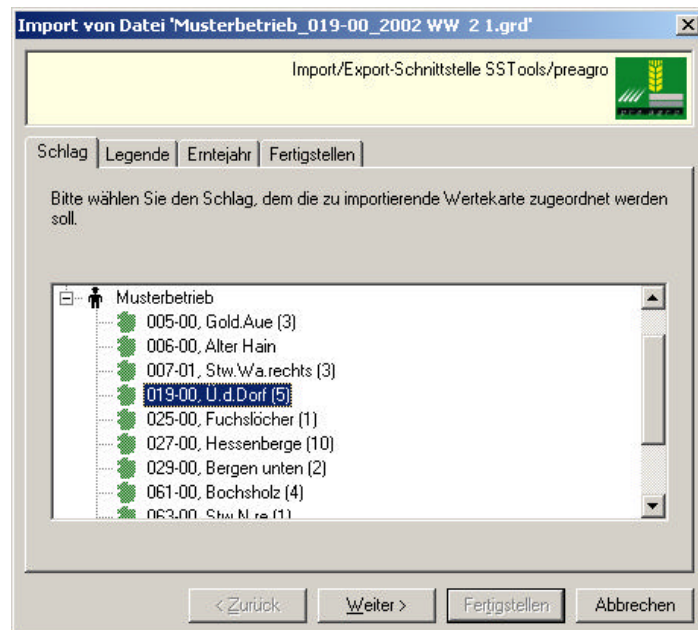
Abb. 4.1-2: Datenfluss Applikationskarte AGRO-MAP Basic mit Im-/ Export-Möglichkeiten



Einordnung in die Datenbanktabellen:

- Schläge
- Grenlinienkopfdaten
- Grenzpunkte

Abb. 4.1-3: Schematische Darstellung des Imports von Schlaggrenzen im BLN-Format



Einordnung in die Datenbanktabellen:

- Schläge
- Informationsarten (Legende)
- Ortsreferenzen
- Rasterstrukturen
- Rasterdatensätze

Abb. 4.1-4: Schematische Darstellung des Imports von Wertekarten im Surfer-Grid-Format

#### 4.1.5.2 Schnittstelle PC-Software / Expertenmodule

Es wurde ein Schnittstellenvorschlag für die Anbindung eines Expertenmoduls an eine PC-Software für ein Precision Agriculture Managementsystem entwickelt, Abb. 4.1-5. Im Rahmen intensiver Diskussionen mit den Modulentwicklern wurden zudem die Anforderungen an ein Expertenmodul, die verschiedenen Realisierungsmöglichkeiten und die Anforderungen an die Modulentwickler heraus gearbeitet.

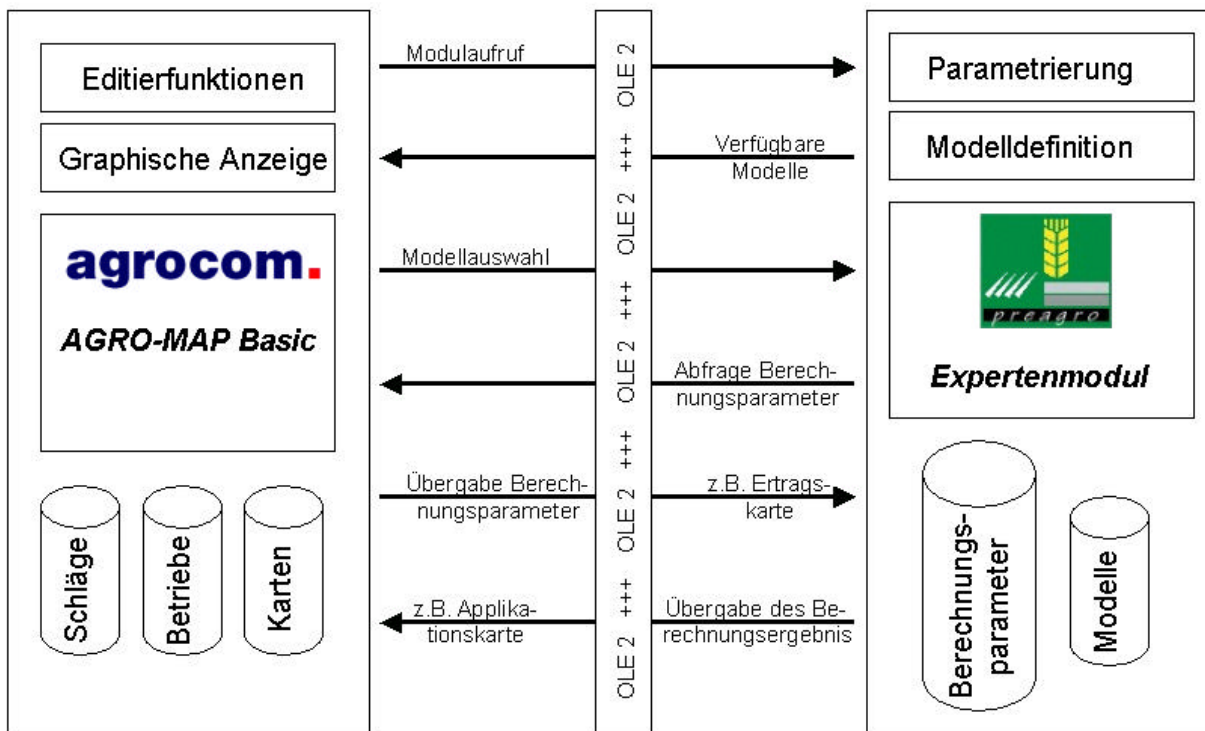


Abb. 4.1-5: Schnittstellengestaltung zur Ankopplung eines Expertenmoduls an eine PC-Software für precision agriculture

Ein Expertenmodul ist ein eigenständiger Programmrahmen, in dem ein oder mehrere Modelle zur Berechnung von beispielsweise Applikationskarten implementiert ist/sind. Dieses Modul wird an eine andere Software angekoppelt, in der Ein- und Ausgangsdaten für das Berechnungsmodell verwaltet werden. Das Expertenmodul besitzt eine eigene Datenbank zur Speicherung spezieller Berechnungsparameter, die dem „Host“-System fehlen.

Ein Expertenmodul kann auf verschiedene Art realisiert werden:

Das Rechenmodell und das Expertenmodul können separat realisiert sein. Hierbei können die Modelle von Nicht-IT-Fachleuten implementiert werden. Beliebig viele Modelle sind im Modul realisierbar und sie können einfach geändert werden. Durch eine einheitliche Oberfläche des Moduls für alle Modelle ergibt sich ein hoher Wiedererkennungseffekt. Die Trennung von Rechenmodell und Modul ermöglicht einen nur geringen Aufwand für die Qualitätssicherung. Der zusätzliche Entwicklungsaufwand für die separate Realisierung von Modell und Modul lohnt

sich erst bei mehreren Modellen und es müssen unter Umständen Kompromisse bei der Komplexität von Modellen und eventuell bei der Benutzeroberfläche gemacht werden.

Wird das Rechenmodell in das Expertenmodul integriert realisiert ist eine schnellere Umsetzung möglich. Komplexere Modelle können besser realisiert werden weil das Modul auf das Modell und auf dessen nötige Benutzeroberfläche optimal zugeschnitten werden kann. Allerdings wird für jede Modell ein eigenes Modul nötig und dadurch steigt die gesamte Entwicklungszeit bei mehreren Modellen an. Zudem ist ein höherer Aufwand für die Qualitätssicherung nötig da für jedes Modell ein eigenes Programm erstellt, gewartet und geprüft werden muss.

Für die technische Realisierung der Modulankoppelung bestehen unterschiedliche Möglichkeiten die nicht von allen gängigen bzw. in preagro verwendeten „Host“-Systemen unterstützt werden. Die ArcView-basierenden Programme SSToolbox (siehe TP III 1b) und AGRO-MAP Professional sowie das heutige, in diesem TP genutzte Programm AGRO-MAP Basic unterstützen die Anbindung von DLL und EXE. AGRO-MAP Basic und das neue Software-Produkt AGRO-NET, ein auf der agritechnica 2001 vorgestelltes Produkt der Fa. AGROCOM GmbH unterstützen zusätzlich noch die COM-Schnittstelle.

Als Standard für die Datenschnittstelle wurde für das Gesamtprojekt preagro das Shape-File-Format der Fa. ESRI verabredet. Als zusätzliches Format findet innerhalb des Projekt das sogenannte Surfer-Grid-Format Verwendung, weil es auf einer einfacheren Ebene den Datenaustausch von einfachen Rasterdaten ermöglicht. Für ein Expertenmodul sollten zumindest diese Datenschnittstellen berücksichtigt werden. Auf Basis dieser Überlegungen wurde die allgemeingültige Schnittstelle für die Anbindung eines Expertenmoduls erarbeitet.

In Zusammenarbeit mit dem TP III-2 wurde das Expertenmodul für die Winterweizen-Aussaat realisiert und an die beschriebene Schnittstelle angebunden. Die Dokumentation dieses Moduls und des API (Application Programming Interface) für die Einbindung in die PC-Software befindet sich im Anhang. Der Ablauf des Moduls ist im Folgenden anhand des Userinterfaces, bestehend aus drei Karteikarten, dargestellt.

In der ersten Karteikarte „Basisdaten“ werden das Ertragspotential bestimmt und Basisdaten für die weitere Berechnung erfasst, Abb. 4.1-6. Zur Bestimmung des Ertragspotentials werden zwei Varianten unterstützt. In der Variante A wird dies auf Basis der Reichsbodenschätzung und des Jahresniederschlags durchgeführt. Hierzu sind folgende Schritte nötig:

- Auswahl einer Shapedatei mit Daten der Reichsbodenschätzung, Abb. 4.1-7, in Feld 2 des Dialoges und Auswahl der Datenspalte mit den eigentlichen RBS Werten.
- Eingabe des Niederschlages in Feld 3.

Variante B unterstützt die direkte Übernahme des Ertragspotentials aus einer anderen Quelle Z.B. der mittleren Ertragserwartungskarte aus AGRO-MAP Basic. Die Auswahl erfolgt in Feld 1 der Basiskarte.

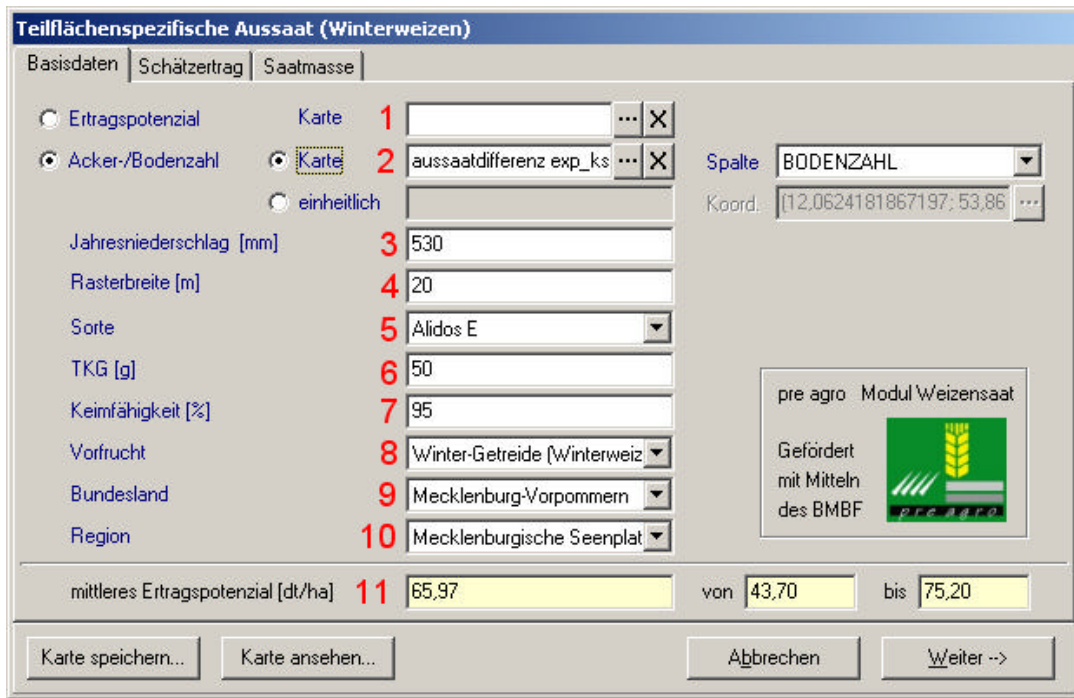


Abb. 4.1-6: Erfassung der Basisdaten zur Bestimmung des Ertragspotentials

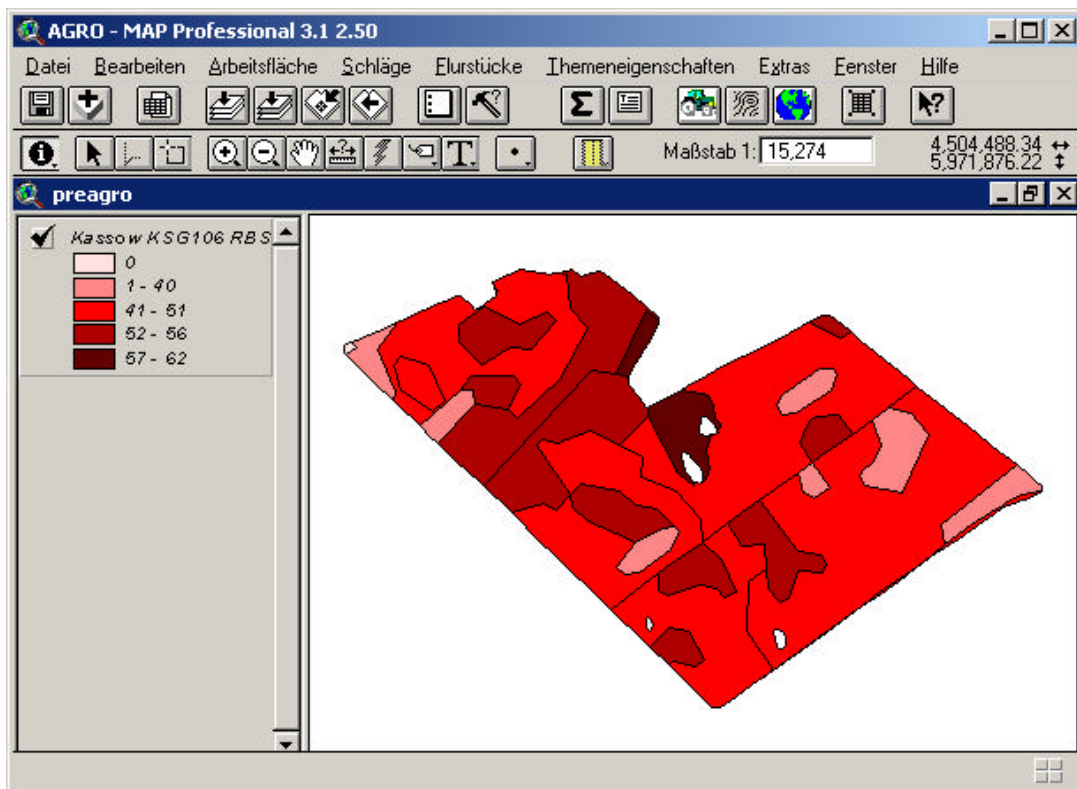


Abb. 4.1.7: Karte mit Werten der Reichsbodenschätzung

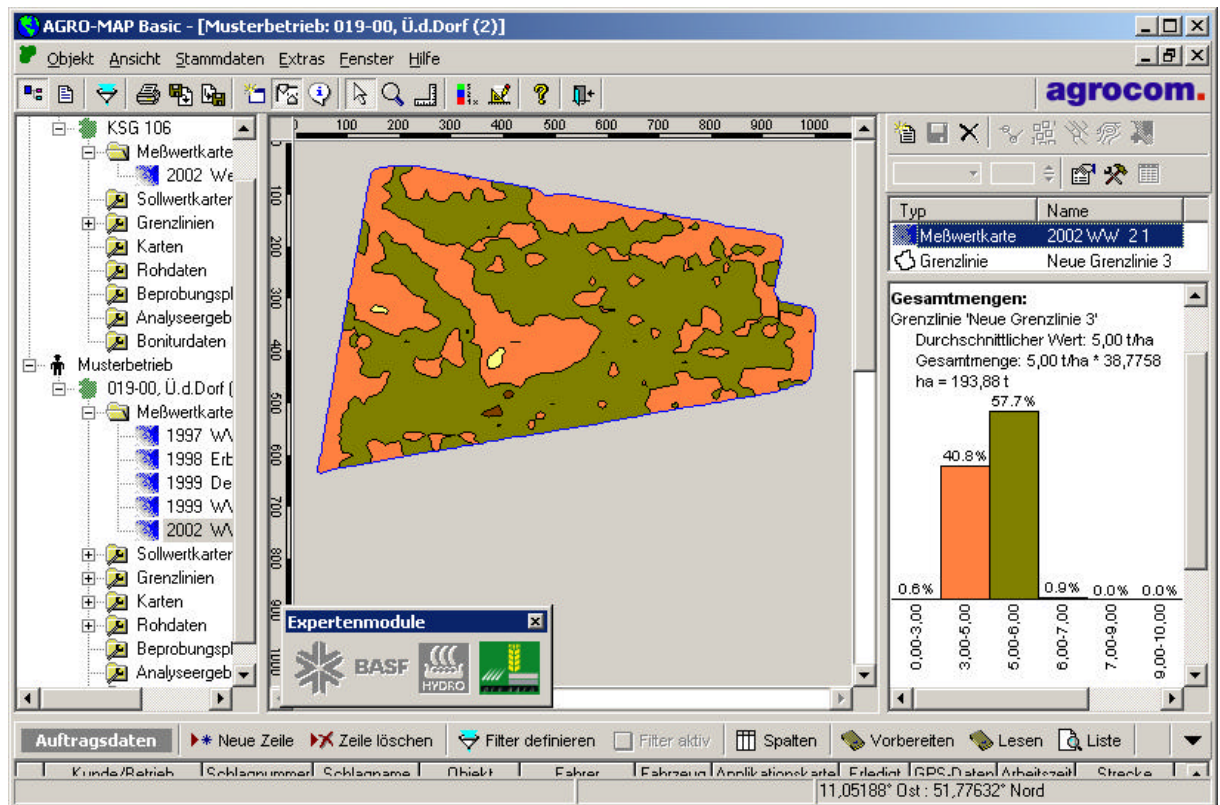


Abb. 4.1-8: Mittlere Ertragserswartungskarte in AGRO-MAP Basic

Für die weitere Berechnung müssen zusätzliche Basisdaten erfasst werden:

- Die gewünschte Rastergröße der resultierenden Saatmassenkarte wird in Feld 4 eingegeben.
- Die Sorte des zu säenden Winterweizens wird in Feld 5 ausgewählt. Die Sorteneigenschaft Tausendkorngewicht wird in Feld 6, die Keimfähigkeit in Feld 7 eingegeben. Die Sorte kann sich mindernd oder steigernd auf den Schätzertrag im nächsten Berechnungsschritt auswirken.
- Die Vorfruchtgruppe wird in Feld 8 ausgewählt. Die Vorfrucht kann sich ebenfalls ertragsmindernd oder – steigernd auf den Schätzertrag im nächsten Berechnungsschritt auswirken.
- Zur Bestimmung des idealen Aussaattermins wird das Bundesland und Region, in welcher der Schlag liegt in Feld 9 und 10 ausgewählt.

Die Höhe und Verteilung des Ertragspotentials wird in Feld 11 ausgegeben und lässt sich als Karte anzeigen oder zwischenspeichern.

In der zweiten Karteikarte „Schätzertrag“ werden Daten zur Korrektur des Schätzertrages erfasst, Abb. 4.1-9:

- Über die Angabe des tatsächlichen Saattermins ausgehend vom idealen Saattermin in Feld 12 wird der Schätzertrag gegenüber dem Ertragspotential verändert. Ein nicht idealer Saatzeitpunkt wirkt sich ertragsmindernd aus.

- Über die Auswahl in Feld 13 kann der Betriebsleiter eine zusätzliche manuelle Korrektur der Ertragserwartung vornehmen.
- Wenn eine Karte mit Angaben zum Reliefeinfluss vorliegt, kann diese optional in Feld 14 ausgewählt werden. Das Relief kann sich positiv und negativ auf das Ertragsziel auswirken.

Durch die Angaben der Basisdaten aus dem vorhergehenden Karteireiter und den Anpassungen auf diesem Karteireiter wird der Schätzertrag bestimmt. Dessen Wertebereich wird in den Feldern 15 angezeigt und kann als Karte angezeigt oder gespeichert werden.

Abb. 4.1-9: Karteikarte mit Daten zur Anpassung des Schätzertrages

Mit der letzten Karteikarte „Saatmasse“, Abb. 4.1-10, werden die Daten zur Bestimmung der Saatmasse erfasst und diese berechnet:

- Aus dem Sortentyp und der Bodenqualität (Karteikarte „Basisdaten“) wird der Einzelährenertrag errechnet.
- Aus dem Einzelährenertrag und dem Schätzertrag wird die Bestandesdichte errechnet.
- Die Pflanzenverluste über Winter werden über die Vorfrucht (Karteikarte „Basisdaten“), dem realen Saattermin (Feld 20) und die Qualität des Saatackers (Feld 21) bestimmt.
- Die Verringerung der Feldaufgangsrate wird aus dem Bodensubstrat (Feld 16), Bodenfeuchte bei Saat (Feld 17), Bodensaatgutkontakt (Feld 18), Saattiefe (Feld 19), Befall durch bodenbürtige Pilze (abgeleitet aus Bodensubstrat) und Zeitspanne Aussaat-Aufgang (abgeleitet aus realem Saattermin) bestimmt.

**Teilflächenspezifische Aussaat (Winterweizen)**

Basisdaten | Schätzertrag | Saatmasse

Bodensubstrat  Karte **16** aussaatdifferenz exp\_ks ... X Spalte BODENART  
 einheitlich sL (sandiger Lehm)

Bodenfeuchte bei Saat **17** gut


Boden-Saatgut-Kontakt **18** gut

Saattiefe [cm] **19** 3 - 5 cm (normal)

Idealer Saattermin 15.09.2002 bis 25.09.2002

Realer Saattermin **20** 1 Woche zu spät

Saatbettqualität **21** gut

pre agro Modul Weizensaat  
 Gefördert mit Mitteln des BMBF 

Saatmasse [kg/ha] **22** 154,94 von 120,00 bis 180,00

Karte speichern... Karte ansehen... <- Zurück Beenden

Abb. 4.1-10: Karteikarte mit Daten zur Bestimmung der Saatmasse

Aus allen Parametern wird die Saatmasse für jedes Flächenelement des Schlages berechnet. Die sich ergebende Saatmassenkarte wird in einem letzten Schritt geglättet und damit von sehr kleinräumigen Inhomogenitäten bereinigt.

Der Wertebereich der Saatmassenkarte wird in den Feldern 22 angegeben und lässt sich als Karte anzeigen oder speichern. Die Anzeige der Karte kann direkt in der Host-Software (z.B. AGRO-MAP Basic, Abb. 4.1-11) vorgenommen werden. Von dort ist auch der direkte Transfer zum Bordcomputer auf dem Traktor möglich.

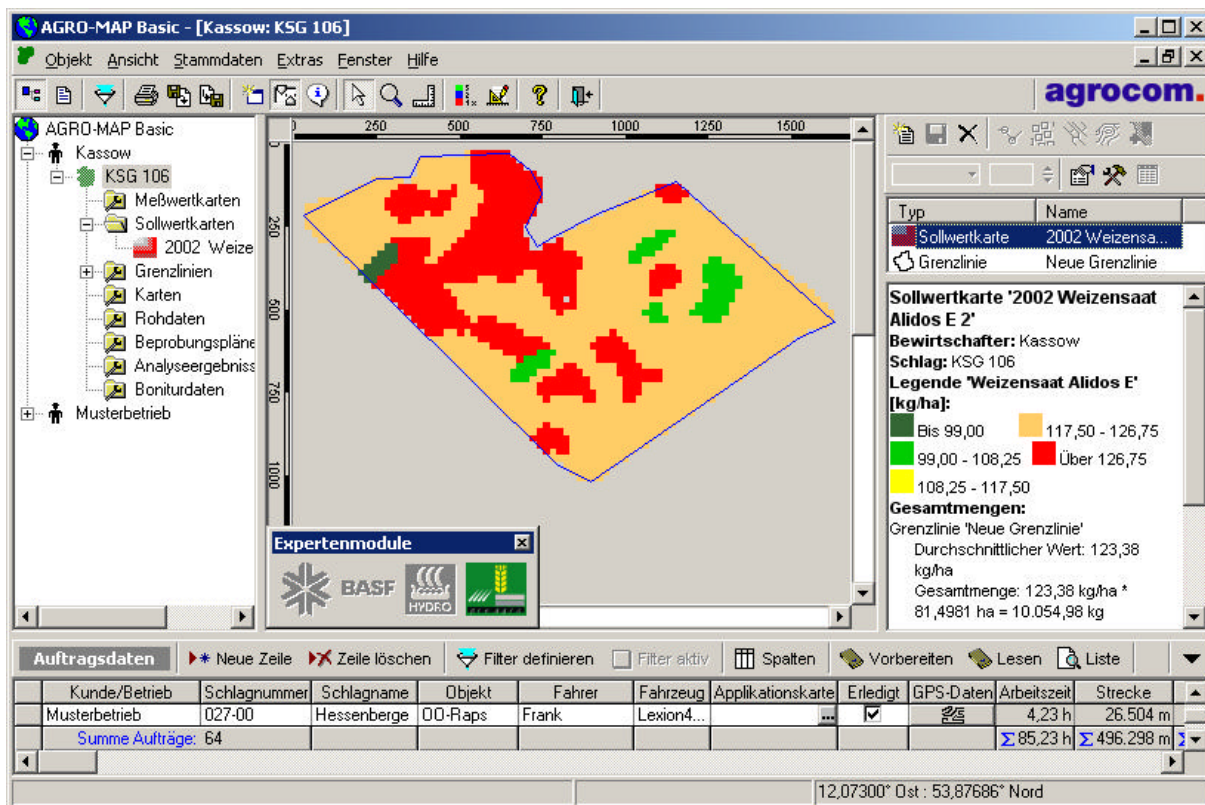


Abb. 4.1-11: Anzeige der Saatmassenkarte in AGRO-MAP Basic

#### 4.1.5.3 Schnittstellen-Standardisierung Bordcomputer / PC-Software

Teil 5 der LBS-Norm DIN 9684 beschreibt die Kommunikation zwischen Bordcomputer und Büro-Software. Diese Norm basiert auf der ADIS-Syntax (Agricultural Data Interchange Syntax). Nur sehr wenige Hersteller von Bordcomputern und von Büro-Software verwenden zum Datenaustausch ADIS. Die Analyse der verwendeten LBS-5-Formate zeigte, dass von den einzelnen Herstellern jeweils feste Untermengen von LBS-5 verwendet werden und höchstwahrscheinlich von keinem das volle LBS-5 verstanden werden kann. In allen Fällen treten teils kleinere, teils größere Abweichungen zu LBS-5 auf. Diese Abweichungen beziehen sich hauptsächlich auf einzelne Datenformate. Z.B. schreibt LBS-5 die Angabe der GPS-Position in Dezimalgrad vor, in machen Datenfiles finden sich aber Positionen in Grad/Minuten/Sekunden, in anderen in Grad/Dezimalminuten. Das reine ADIS lässt diese Freiheit und unterstützt sie durch die Verwendung von Definitionszeilen für die reinen Datenzeilen in dem Datensatz. Da die Software-Entwicklung sehr aufwendig wird wenn alle diese Varianten berücksichtigt werden müssen, wurden in LBS-5 Beschränkungen dieser Möglichkeiten eingeplant.

Unter Berücksichtigung des hohen Aufwands für eine allgemeine LBS-5 Schnittstelle inklusive Anpassung an spezielle Varianten, der bisher vorhandenen speziell angepassten Schnittstellen zum Import von Daten von Bordcomputern anderer Hersteller und unter Berücksichtigung einer alternativen Schnittstelle zum Austausch der Daten zwischen den Softwareprodukten von TP III 1a und 1b (siehe 4.5.1.4) wird bis auf weiteres innerhalb des Teilprojektes auf eine Anpassent-

wicklung für ein LBS-5 Schnittstelle verzichtet. Inwieweit eine Einigung der verschiedenen Hersteller auf LBS-5 bzw. auf eine Untermenge von LBS-5 machbar sein könnte wird derzeit geprüft. Zudem ist die kurz vor dem Abschluss stehende Normung ISO 11783, p.11 als internationale Variante der DIN 9684 mit deren Änderungen für das Datenformat zu berücksichtigen.

Zur Prüfung der Einführung einer zu realisierenden Untermenge der LBS-5 Schnittstelle konnte über die LAV im VDMA koordiniert Besprechungen organisiert werden, an der Hersteller von Agrarsoftware, Hersteller von Landmaschinen mit Elektronik und Wissenschaftlern teilnahmen. Einmalig waren auch Anwender vertreten um deren Bedürfnisse direkt in die Arbeit zu integrieren. Bei den Besprechungen wurden die Hürden für die Anwendung von der aktuellen Norm DIN 9684 Teil 5 gemeinsam herausgearbeitet. Zur Erleichterung der Umsetzung der Norm wurde ein Minimalkatalog erarbeitet. Von großer Bedeutung ist, dass dieser Minimalkatalog in die zukünftige ISO 11783 part 11 integriert werden soll. Die Erfahrungen bei der Realisierung und Umsetzung des Landwirtschaftlichen Bussystems DIN 9684 werden somit direkt in der internationalen Normungsarbeit verwendet.

#### **4.1.5.4 Schnittstellen-Harmonisierung Bordcomputer / Anbaugerät**

Das Landwirtschaftliche Bussystem (LBS) ist in Deutschland als DIN 9684 1-5 genormt. Während in DIN 9684-1 die Signalsteckdose als Punkt-zu-Punktverbindung standardisiert wurde, basieren die Teile 2-4 auf einem seriellen Datenbus, dem CAN-Bus. Heute sind nur vereinzelt Schlepper ab Werk mit LBS-Ausrüstung erhältlich während sich die Signalsteckdose stärker verbreitet hat. Von mehreren Anbietern sind Nachrüstätze für eine LBS-Grundausrüstung von Schleppern erhältlich. Diese Nachrüstätze können auf einer vorhandenen Signalsteckdose aufbauen oder die entsprechende Sensorik muss auf dem Schlepper ebenfalls nachgerüstet werden. Dies bedeutet, dass heute nahezu jeder auf einem landwirtschaftlichen Betrieb eingesetzte Schlepper LBS-tauglich aufgerüstet werden kann.

Im Bereich der Anbaugeräte ist die Situation komplexer. Hier werden ebenfalls nur wenige Anbaugeräte ab Werk mit LBS-Ausrüstung angeboten. Positiv zu vermerken ist, dass einige Hersteller zumindest für ihre High-End-Maschinen diese Ausrüstung anbieten bzw. in Arbeit haben. Allerdings kommen hierbei unterschiedliche Untermengen der LBS-Norm zur Anwendung. Viele Hersteller nutzen den Teil 3 der Norm, nur wenige setzen konsequent auch den Teil 4, die sogenannte virtuelle Benutzerstation um. Zudem erlaubt LBS auch sogenannte Partnersysteme für Funktionen die in der Norm nicht vereinbart wurden. Werden derartige Partnersystemfunktionen von einem Anbaugerät genutzt wird eine spezielle Softwareanpassung für den Bordcomputer nötig. Nur wenn Maschinen und Bordcomputer gemäß Teil 4 arbeiten ist gewährleistet, dass die Komponenten ohne vorherige Softwareinstallation bzw. Softwareanpassung auf dem Bordcomputer zusammen arbeiten.

Neben Anbaugeräten mit LBS-Ausstattung sind sehr viele Geräte im Markt und werden auch noch angeboten, die zur Gerätesteuerung eigene bzw. speziell an sie angepasste Bordcomputer

einsetzen. Damit diese Geräte in Precision Agriculture-Systeme eingebunden werden können sind diese Bordcomputer häufig mit einer seriellen Computerschnittstelle (RS 232) ausgestattet bzw. nachrüstbar. Über diese Schnittstelle kann ein weiterer Bordcomputer, der über georeferenzierte Auftragsdaten und ein D-GPS-System verfügt, dem ersten System Sollwerte z.B. für eine Applikation übertragen, die sofort umgesetzt werden. Manche Bordcomputer liefern über diese Schnittstelle auch die tatsächlich applizierte Menge zur Kartierung der sogenannten As-Applied-Daten zurück. Für das Protokoll auf der RS 232 gibt es keine Norm und es hat sich auch noch kein einheitliches Protokoll durchgesetzt.

Aus den genannten Gründen ist die Anforderungen an Precision-Agriculture-taugliche Bordcomputer, alle existierenden Schnittstellen der unterschiedlichsten Elektroniken der Anbaugeräte bedienen zu können, äußerst komplex. Im Rahmen des Verbundprojektes werden zudem immer wieder innovative Anbaugeräte eingesetzt um deren Eignung für den Einsatz in Precision Agriculture zu überprüfen. Dies bringt mit sich, dass auf den Praxisbetrieben des Projektes häufiger neue Schnittstellenausführungen innerhalb der drei angeführten Klassen (LBS 3, 4, RS 232) von den eingesetzten Bordcomputern angesteuert werden können müssen. Für das auf einigen Praxisbetrieben eingesetzte ACT der Fa. Agrocom ist deshalb in Absprache mit den Technikbetreuern dafür zu sorgen, dass rechtzeitig die benötigte Schnittstelle für das Anbaugerät der Wahl für die Feldarbeiten zur Verfügung stehen. Dies stellt hohe Anforderungen an die Flexibilität des Teilprojektes und erfordert kurzfristige Änderungen der Prioritäten innerhalb der Entwicklung.

Zur zukünftigen Vermeidung derartiger Schwierigkeiten ist seitens des Teilprojektes eine Vereinbarung mit Fieldstar, einem weiteren wichtigen Anbieter von Bordcomputern und der Fa. Fendt (beide Agco-Gruppe), die ihre Schlepper häufig ab Werk mit LBS tauglichen Bordcomputern ausrüstet, getroffen worden. Ziel war die Vereinheitlichung und Harmonisierung der verwendeten LBS-Protokolle zwischen Anbaugeräten und Bordrechnern. Interpretationsspielräume und individuelle Ergänzungen führten zu verschiedenen Dialekten nach DIN 9684 Teil 3. Durch die Harmonisierungsbemühungen ist es gelungen die wichtigsten Anwender dieses Normteils an einen Tisch zu bekommen und eine gemeinsame Interpretation der Norm zu erarbeiten. Unter dem Marketingnamen „LBS+“ ist diese Harmonisierung im Markt bekannt gemacht worden. Sie vereinigt die Einfachheit des Teil 3 und Vorteile von Teil 4 (Maskenupload über CAN-Bus). Dieser harmonisierte Standard hat eine gewisse Verbreitung gefunden und stellt eine praktikable Lösung bis zur erfolgreichen Einführung der Nachfolgenorm der DIN 9684, der aktuell in abschließender Bearbeitung befindlichen ISO 11783. Seitens des Teilprojektes wird ebenfalls an der zwischen den unterschiedlichsten Unternehmen abgestimmten Einführung der zukünftigen Norm mitgearbeitet.

#### **4.1.5.5 Schnittstellen-Standardisierung Bordcomputer / Anbaugerät**

Der Stand der Normungsaktivitäten für die ISO 11783 “Tractors and machinery for agriculture and forestry – Serial control and communications data network” ist in Skrandies, 2002 zusammen gefasst. Auf der agritechnica 2001 wurden von mehreren Firmen Prototypen für Bordcom-

puter und Jobrechner für Anbaugeräte und Schlepper nach der ISO 11783 präsentiert. Basis dieser Präsentation sind die Bemühungen, die über die LAV der VDMA koordiniert wurden, einheitliche Schrittmaße bei der Umsetzung und Einführung dieser Norm für alle Firmen gemeinsam zu verabreden. Hierfür wurde u.a. ein Implementation Level 1 definiert. In ihm sind die Minimal-Umfänge für Botschaften in einem ISO 11783-Netzwerk definiert. Hierdurch wird erreicht, dass für die Firmen der zu realisierende Umfang im Beginn ihrer jeweiligen Normumsetzung minimiert wird und trotzdem die Kompatibilität zu den Komponenten anderer Hersteller erreicht wird. Durch diese Aktivität konnte eine Vielzahl von Firmen zu der Teilnahme an der ISO-Präsentation auf der Messe überzeugt werden.

Durch sogenannte Plugfeste, Treffen auf Entwicklerebene bei denen die einzelnen Komponenten verschiedener Hersteller zusammen gesteckt und die Funktionsfähigkeit überprüft werden kann, ist der Informationsaustausch über Firmengrenzen und die gemeinsame Entwicklungsanstrengungen aller Beteiligten ermöglicht worden. Diese Form der Zusammenarbeit findet europaweit statt und ist in dieser Art direkt von den Kollegen aus den nordamerikanischen Firmen übernommen worden. Global agierende Unternehmen haben zudem die einfache Möglichkeit, ihre Komponenten in Europa und in Nordamerika zu testen.

Auch wenn erste serienmäßig marktverfügbare Bordcomputer, Anbaugeräte und Schlepper erst in 1-2 Jahren realisiert sein werden, so konnte durch diese Aktivität zweierlei erreicht werden: Zum Einen wurde eine beispielhafte Zusammenarbeit zwischen den Entwicklern unterschiedlichster Hersteller erreicht. Diese Zusammenarbeit wird die zukünftige Einführung durch eine Minimierung von Fehlern in den Komponenten und ihrem Zusammenspiel erleichtern und beschleunigen. Zum Anderen konnte hierdurch bei den zukünftigen Anwendern eine Sicherheit erzeugt werden, durch die Diskussionen über Kompatibilitäten, Normen etc. deutlich verringert wurden.

#### **4.1.6 Kooperation mit anderen Teilprojekten**

Die Konzeption des Gesamtprojektes preagro fußt auf der Zusammenarbeit zwischen den einzelnen Teilprojekten. Aus dem TP III 1a sind insbesondere folgende Zusammenarbeiten hervorzuheben:

- Zusammenarbeit mit dem TP Technikbetreuung zur Behebung von Problemen auf den Demobetrieben und zur Abstimmung technischer Weiterentwicklungen. Die Einführung und Umsetzung zum Teil auch neuer Techniken für Precision Agriculture bedarf einer guten Betreuung vor Ort und mit Unterstützung weiterer Fachleute. Dies wurde das TP Technikbetreuung gewährleistet. Die Spezialisten z.B. für die Software Betrieb oder des Bordcomputers mit seinen Schnittstellen zu zum Teil neuen Anbaugeräten konnten hier vielfältige Unterstützung durch Beratung, Anpassentwicklungen oder auch Neuentwicklungen leisten.
- Mit dem TP III 1b „Software Dienstleistungen“ waren hauptsächlich die Schnittstellen für den Datenaustausch zwischen den Software-Programmen abzustimmen. Hierdurch wurde

die Umsetzung der von den pflanzenbaulichen Wissenschaftlern erarbeiteten Maßnahmen erleichtert. Nur durch diese automatisierte Datenübernahmen war gewährleistet, dass die Umsetzung auf der Vielzahl von Betrieben über den gesamten Projektzeitraum leistbar war.

- Mit dem TP III 2 ist für die Erarbeitung der Benutzerschnittstelle für das von diesem TP entworfene Aussaatmodul Winterweizen zusammen gearbeitet worden. Diese interdisziplinäre Zusammenarbeit war besonders interessant und fruchtbar, da beide TPs auf unterschiedlichem Erfahrungshorizont basierend verschiedene Vorstellungen für die Benutzerschnittstelle entwickelt haben. Die Diskussionen und Abstimmungen führten zu einer Lösung die beide Aspekte, einfache Benutzerführung und Wahrung der Nachvollziehbarkeit und der Wissenschaftlichkeit sicher stellt. Des Weiteren ist mit dem TP III 2 stark zur Verifizierung und Validierung der Implementation des Winterweizen-Aussaatmoduls zusammen gearbeitet worden.
- Zusammen mit dem TP Ökonomie fand für die Entwicklung von teilflächenspezifischen DB-Berechnungen ein intensiver Austausch statt. Hintergrund bildete hierbei das bereits in der Praxis verfügbare System der Teilprojekteinrichtung und die prinzipielle Weiterentwicklung der darin abgebildeten Abläufe hin zu dem von diesem TP zu leistenden Nachweis der ökonomischen Effekte von Precision Agriculture.

#### **4.1.7 Ausblick**

Durch die Weiterentwicklung von Schnittstellen bei PC-Software, Bordcomputer, Anbaugeräten wurde die Umsetzung der vom Gesamtprojekt angelegten Versuche und der erarbeiteten Algorithmen erleichtert bzw. auch ermöglicht. Dies verdeutlicht die weiter nötige Anstrengung der Umsetzung von standardisierten Schnittstellen bei allen in der Landwirtschaft aktiven Firmen.

Die erstmalige Realisierung eines Expertenmoduls mit einer allgemeingültigen Schnittstelle ermöglicht nun den Einsatz dieses Moduls an Softwareprodukten anderer Hersteller. Erste Anfragen hierzu liegen vor. Für die Weitergabe der erstellten Software wird die gesamte Dokumentation, der Sourcecode und die Compilats auf CD für Interessierte zur Verfügung gestellt werden.

Nach der Übergabe der Modulsoftware und der dazugehörigen Dokumentation an weitere Interessierte Softwarefirmen bzw. Dienstleister ergibt sich wieder die noch nicht gelöste Frage, wie eine Pflege und Weiterentwicklung eines derartigen, im Rahmen eines öffentlichen Förderprojekts entwickelten Moduls erfolgen kann. Drei Aspekte sind hierbei zu klären um eine erfolgreiche Praxisübernahme zu erreichen:

- die Pflege und Weiterentwicklung der durch Wissenschaftler erstellten Algorithmen muss von Pflanzenbauspezialisten erfolgen
- die Pflege und Weiterentwicklung der Software durch ein Softwarehaus
- sowie die Vermarktung dieses Produktes mit der dazugehörigen Erlösverteilung

Für zukünftige Konzeptionen weiterer Expertenmodule ist die leichtere Wartbarkeit und Updatefähigkeit über z.B. Internet zu berücksichtigen. Die Integration von Expertenmodule in einen zentralen Pflanzenbau-Beratung-Server und der Aufbau eines Beratungsportals könnte eine mögliche Lösung sein. Hinsichtlich Pflege und Weiterentwicklung der Algorithmen, Erlösverteilung bleiben die Fragen aber bestehen. Hinzu kommt die zu klärende Frage wer der Betreiber eines derartiger Portales sein kann.

Im Rahmen des Gesamtprojektes und insbesondere bei der Umsetzung auf den Betrieben zeigte sich wieder deutlich die zentrale Bedeutung von einer Betriebssoftware in den Händen des Landwirtes. Weitere Vereinfachungen in der Benutzerführung, die Realisierung von durchgängigen Lösungen über Herstellergrenzen hinweg, Integration von individuellen Modulen und die Anbindung an Beraterportale stellen zukünftig die Herausforderungen dar.

## **4.1.8 Literatur**

### **4.1.8.1 Verwendete Literatur**

Autermann et al (2000): LBS+ zur Harmonisierung der DIN 9684. Unveröffentlichtes Firmendokument.

Skrandies, D. (2002): Die ISO 11783 zur standardisierten Datenverbindung in der Landwirtschaft. VDI / MEG-Tagung Landtechnik, Halle/Saale, 10./11.10.2002.

ISO 11783, 1-11 (2002): Tractors and machinery for agriculture and forestry – Serial control and communications data network.

### **4.1.8.2 Veröffentlichungen**

Böttinger, Stefan (1999): Informations- und Kommunikationstechnik in der Landwirtschaft. In: Einsatz interaktiver Medien im Unternehmen, Frank Deges (Hrsg.), Stuttgart: Schäffer-Poeschel, 1999.

Böttinger, Stefan (1999): GeoAttribut - Processing of georeferenced machine data in the municiple area. 14. ESRI European Conference, München, 15./17.11.1999, CD Conference Program.

Böttinger, Stefan (2000): Informations- und Regelsysteme am Mährescher, Stand der Technik und Entwicklungstendenzen. Landtechnik 55 (2000), SH, S. 96– 98.

Böttinger, Stefan (2000): Datenverbund, Integration von Schlagkartei, satellitengestützter Software und GIS am Beispiel von AGRO-NET. Landtechnik 55 (2000), Nr. 3, S. 220– 221.

Böttinger, Stefan (2000): Ortung und Navigation im landwirtschaftlichen Flottenmanagement. In: DGON-Symposium Ortung u. Navigation 2000 – GALILEO, 17./19.10.2000, Freising/Weihenstephan, Tagungsband, Bonn: 2000.

Böttinger, S., Schwaiberger, R. (2000): Management der Informationsverarbeitung in precision agriculture. Software. In: Schwaiberger, R., Sommer, C., Werner A. [Hrsg.]: preagro Projekt Zwischenbericht 2000; Sonderveröffentlichung 32 des KTBL: 165-176.

Böttinger, Stefan (2001): Stand der Normung der ISO 11783 und der Umsetzung durch die Industrie. In: Referate der 22. GIL-Jahrestagung in Rostock 2001, Berichte der Gesellschaft für Informatik in der Land-, Forst- und Ernährungswirtschaft, Bd. 14, Hans Kögl et al (Hrsg.), 2001.

- Böttinger, Stefan, K. Oetzel, R. Schwaiberger u. P. Leithold (2002): Software zum Management von Precision Agriculture. In: Precision Agriculture – Herausforderung an integrative Forschung, Entwicklung und Anwendung in der Praxis, Armin Werner u. Andreas Jarfe (Hrsg.), KTBL-Sonderveröffentlichung 038, Darmstadt 2002.
- Böttinger, Stefan (2002): Software Betrieb. In: Precision Agriculture – Herausforderung an integrative Forschung, Entwicklung und Anwendung in der Praxis, Armin Werner u. Andreas Jarfe (Hrsg.), KTBL-Sonderveröffentlichung 038, Darmstadt 2002.
- Böttinger, Stefan (2002): Precision Farming in Europe – Experiences, Benefits and Future Trends. In: Ist Precision Farming ökonomisch? Tagungsband, 01.03.2002, Mitsuru Hachia (Hrsg.), Obihiro, Japan: 2002.

### 4.1.8.3 Vorträge

- Böttinger, Stefan (1999): „Vernetzte Agrartechnik - der Weg nach vorn“ - Eintritt des Informationszeitalters in die Landtechnik. Landtechnische Unternehmertage 1999, 15./16.1.99 Bad Wiessee.
- Böttinger, Stefan (1999): Informationstechnik zur Optimierung der Aussenwirtschaft. Kolloquium Agrartechnik, Universität Hohenheim, Stuttgart, 25.10.1999.
- Böttinger, Stefan (1999): GeoAttribut - Processing of georeferenced machine data in the municiple area. 14. ESRI European Conference, München, 15./17.11.1999.
- Böttinger, Stefan (2000): Precision Farming und notwendige technische Einrichtungen. Technische Informationstagung „Innovatives Farm-Management auf Basis neuer Technologien“, 22.03.2000, Gödöllő, Ungarn.
- Böttinger, Stefan (2000): Erfassung und Einsatz von Geodaten in Betriebsführung und Flottenmanagement. Fachtagung: Anwendung von Geodaten in der Landwirtschaft, 3./4.04.2000, Münster.
- Böttinger, Stefan u. Ludger Autermann (2000): Die Geräteschnittstelle des LBS (DIN 9684): Erfahrungen aus Industrie und Praxis, aktuelle Weiterentwicklungen. VDI/MEG-Tagung Landtechnik 2000, 10./11.10.2000, Münster.
- Böttinger, Stefan (2000): Ortung und Navigation im landwirtschaftlichen Flottenmanagement. DGON-Symposium Ortung u. Navigation 2000– GALILEO, 17./19.10.2000, Freising/Weihenstephan.
- Böttinger, Stefan (2000): Leistungssteigerung in der Aussenwirtschaft durch Informationstechnik – Precision Farming und Flottenmanagement für Landwirte und Lohnunternehmer. Landmaschinen-Seminar der FH-Köln, 14.11.2000, Köln.
- Böttinger, Stefan (2001): GPS-Einsatz in der Landwirtschaft. VDV-Smyposium, 06.02.2001, Bielefeld.
- Böttinger, Stefan (2001): Stand der Normung der ISO 11783 und der Umsetzung durch die Industrie. 22. GIL-Jahrestagung, 19./21.09-2001, Rostock.
- Böttinger, Stefan (2001): Groß, schnell, schlau: Landmaschinen nutzen IT. VDE-Forum Informationstechnik 2001: Informationstechnik für das Fahrzeug, 07.11.2001, Dortmund.
- Böttinger, Stefan (2001): Groß, schnell, schlau: Internationale Standardisierung der Elektronik in Landmaschinen. VDMA-Tagung der LAV-KD-Leiter, 12.03.2002, Saugau.
- Böttinger, Stefan (2002): Precision Farming in Europe – Experiences, Benefits and Future Trends. Precision Farming Tagung, 01.03.2002, Obihiro, Japan.
- Schwaiberger, Reinhold, P. Leithold, S. Böttinger u. K. Oetzel (2002): Software zum Managementsystem für Precision Agriculture. Precision Agriculture Tage, 13./15.03.2002, Bonn.
- Böttinger, Stefan u. Kai Oetzel (2002): Software zum Managementsystem für Precision Agriculture – Spezielle Anforderungen für den betrieblichen Einsatz. Precision Agriculture Tage, 13./15.03.2002, Bonn.

## 4.1.9 Anhang

### 4.1.9.1 VB-Sourcecode zum Im- bzw. Export von File mit den Formaten Surfer Bln, Grd und Lvl

```

'*****
***
'* ieBlnRead
'*****
Public Function ieBlnRead(sFName As String, abrdRet() As BORDER, lRetCount As Long) As Boolean
    On Error GoTo ErrHandler
    'VBLM OFF
    Dim i As Long, l As Long
    Dim s As String, asX() As String

    Dim lFile As Long, lLine As Long

    Dim cPolygon As Long
    Dim sType As String
    Dim iFirst As Long, iLast As Long
    Dim lLength As Long
    Dim lSize As Long

    ' Rückgabeveriable plätten
    Erase abrdRet()
    lRetCount = 0

    ' Datei öffnen
    lFile = FreeFile
    Open sFName For Input As #lFile

    Do Until EOF(lFile)
        ' Startzeile des Linienzugs
        ' length,flag "Pname 1"
        Line Input #lFile, s
        lLine = lLine + 1
        ' l = m_lSurferParse(s, asX())
        l = fs.ParseSurfer(s, asX())
        If l < 2 Then GoTo ExcLine ' min. 2 Token
        If Not gbIsNumeric(asX(0)) Then GoTo ExcLine ' Length ist numerich
        lLength = Val(asX(0))
        If lLength <= 0 Then GoTo ExcLine ' Muss min. 1 sein!
        If Not gbIsNumeric(asX(1)) Then GoTo ExcLine ' Flag ist numerisch
        l = Val(asX(1))
        If l <> 0 And l <> 1 Then GoTo ExcLine ' 0 - Aussenlinie, 1 - Innenlinie
        cPolygon = cPolygon + 1
        sType = IIf(l = 0, "A", "I") & cPolygon

        ' Array entsprechend vergrößern
        lSize = lSize + lLength
        Redim Preserve abrdRet(0 To lSize - 1)

        iFirst = lRetCount ' Index d. ersten Punkts merken!
        For i = 1 To lLength
            ' x1, y1
            Line Input #lFile, s
            lLine = lLine + 1
            ' l = m_lSurferParse(s, asX())
            l = fs.ParseSurfer(s, asX())
            If Not gbIsNumeric(asX(0)) Or Not gbIsNumeric(asX(1)) Then GoTo ExcLine

            iLast = lRetCount
            With abrdRet(iLast)
                .dblEast = Val(asX(0))
                .dblNorth = Val(asX(1))
                .sType = sType
            End With

            lRetCount = lRetCount + 1
        Next ' i
    
```

```

' Wenn es nicht ein Einzelpunkt ist und der erste mit dem letzten
' Punkt identisch ist, dann nehm ich den letzten wieder heraus!
If lLength > 1 Then
    If abrdRet(iFirst).dblEast = abrdRet(iLast).dblEast _
        And abrdRet(iFirst).dblNorth = abrdRet(iLast).dblNorth Then
        lRetCount = lRetCount - 1
    End If
End If

Loop ' Do Until EOF(lFile)...

' Das war's
'*** Return
ieBlnRead = True

UndTschuess:
On Error Resume Next
Close #lFile

Exit Function

ExclLine:
s = "An error occurred parsing file '" & _
    gsFTitle(sFName) & "' line " & lLine & "."
Err.Raise 7368, , s

ErrorHandler:
If gnErrorHandler(m_sMOD, "ieBlnRead", Err, Erl) = vbRetry Then Resume
Resume UndTschuess
End Function

'*****
'***
'* ieGrdRead
'*****
'***
'* Dahingehend angepasst, dass Zeilenumbrüche im Datenblock ignoriert werden.
'* Surfer selbst macht's so, und zu dem wollen wir ja kompatibel sein.
Public Function ieGrdRead(sFName As String, iegRet As ieGRDSTRUCT, adblRet() As Double) As Boolean
On Error GoTo ErrorHandler
'VBLM OFF
Dim i As Long, i_MAX As Long
Dim l As Long, x As Long, y As Long
Dim s As String
Dim adbl() As Double
Dim lFile As Long, lLine As Long, lValue As Long
Dim lXSize As Long, lYSize As Long

' Rückgabewariablen plätten
Dim ieg_NULL As ieGRDSTRUCT
iegRet = ieg_NULL
Erase adblRet
lXSize = 0: lYSize = 0

' Datei öffnen
lFile = FreeFile
Open sFName For Input As #lFile

' Kennung - muss da sein
' DSAA
Line Input #lFile, s
lLine = lLine + 1
s = Trim$(s)
If StrComp(s, "DSAA", vbTextCompare) <> 0 Then
    s = "" & gsFTitle(sFName) & "" & Chr$(32) & _
        "is not a valid DSAA-file!"
    Err.Raise 7367, , s
End If

' Spalten Zeilen - muss da sein
' 26 61
Line Input #lFile, s
lLine = lLine + 1
' l = m_lSurferParseDbl(s, adbl())
l = fs.ParseSurferDbl(s, adbl())
If l < 2 Then GoTo ExclLine

```

```

lXSize = adbl(0)
lYSize = adbl(1)
ReDim adblRet(0 To lXSize - 1, 0 To lYSize - 1)

With iegRet

    ' xmin xmax - muss da sein
    ' 9.491466917708102e+00 9.495152695258898e+00
    Line Input #lFile, s
    lLine = lLine + 1
    ' l = m_lSurferParseDbl(s, adbl())
    l = fs.ParseSurferDbl(s, adbl())
    If l < 2 Then GoTo ExcLine
    .dblLeftBottomEastWgs = adbl(0)
    .dblRightTopEastWgs = adbl(1)

    ' ymin ymax - muss da sein
    ' 5.255011163004859e+01 5.255550353263168e+01
    Line Input #lFile, s
    lLine = lLine + 1
    ' l = m_lSurferParseDbl(s, adbl())
    l = fs.ParseSurferDbl(s, adbl())
    If l < 2 Then GoTo ExcLine
    .dblLeftBottomNorthWgs = adbl(0)
    .dblRightTopNorthWgs = adbl(1)

End With

' zmin zmax - muss da sein, ist mir aber egal
' 5.502847212832270e+00 1.149688859093950e+01
Line Input #lFile, s
lLine = lLine + 1

' Daten - so viele wie ich Zeilen habe! (stehen auf dem Kopf)
' Magic ist immer gdblMAGIC
iegRet.dblMagic = gdblMAGIC
lValue = 0
Do While Not EOF(lFile)
    Line Input #lFile, s
    lLine = lLine + 1
    ' l = m_lSurferParseDbl(s, adbl())
    l = fs.ParseSurferDbl(s, adbl())
    ' If l <> lXSize Then GoTo ExcLine
    ' Eine Datenzeile muss nicht mehr lXSize Werte enthalten!

    ' Im alten Code (gbGrdRead in basGeoX) gibt es noch folgende
    ' Behandlung:
    ' If adbl(i) = 1.70141E+38 Then
    '     adblRet(i, l) = gdblMAGIC
    ' End If
    ' Der Sinn ist mir allerdings verborgen! Weder behandelt der Surfer
    ' diesen Wert als <nicht definiert> noch scheint Kemira Loris
    ' diesen Wert zu schreiben.
    ' MS: Andere Programm verwenden diesen Wert für 'nicht
    '     definiert'. Mache es jetzt ebenso, auch wenn's der
    '     Surfer nicht nutzt.

    'x i_MAX = l - 1
    'x l = lLine - 6      ' 0-basierte Datenzeile
    'x l = lYSize - 1 - 1 ' steht auf dem Kopf
    'x For i = 0 To i_MAX
    'x     adblRet(i, l) = adbl(i)
    'x Next ' i
    'x
    'x If l = 0 Then Exit Do ' Abbruchkriterium - was dahinter kommt ist mir
wurscht
    i_MAX = l - 1
    For i = 0 To i_MAX
        x = lValue Mod lXSize ' Spalte anhand der lfd. Wertenummer
        y = lValue \ lXSize ' Zeile anhand der " "
        y = lYSize - y - 1 ' Auf dem Kopf stehend!
        If adbl(i) > 1E+35 Then
            adblRet(x, y) = iegRet.dblMagic
        Else
            adblRet(x, y) = adbl(i)
        End If
        lValue = lValue + 1
    Next i

```

```

    Next ' i
Loop

'x If l <> 0 Then
'x   ' Ich bin über EOF ausgestiegen, habe aber noch nicht alle Datenzeilen
'x   ' füllen können, d.h. ein Teil meines Arrays ist nicht definiert!
'x   s = "Not enough data rows in '" & gsFTitle(sFName) & "!"
'x   Err.Raise 7369, , s
'x End If
If lValue <> lXSize * lYSize Then
'   ' Nicht genug Werte definiert!
s = "Not enough data in '" & gsFTitle(sFName) & "!"
Err.Raise 7369, , s
End If

ieGrdRead = True

UndTschuess:
On Error Resume Next
Close #lFile

Exit Function

ExcLine:
s = "An error occurred parsing file '" & _
    gsFTitle(sFName) & "' line " & lLine & "."
Err.Raise 7368, , s

ErrorHandler:
If gnErrorHandler(m_sMOD, "ieGrdRead", Err, Erl) = vbRetry Then Resume
Resume UndTschuess
End Function

'*****
'***
'* ieGrdWrite
'*****
'***
'* Parameter ieGWIB hinzugefügt.
Public Function ieGrdWrite(sFName As String, ieg As ieGRDSTRUCT, adbl() As Double, _
    ByVal ieGWIB As ieGRDWRITEINVALIDBEHAVIOR) As Boolean
On Error GoTo ErrorHandler
'VBLM OFF
Dim x As Long, x_MIN As Long, x_MAX As Long
Dim y As Long, y_MIN As Long, y_MAX As Long
Dim s As String
Dim dbl As Double, dbl_MIN As Double, dbl_MAX As Double
Dim lFile As Long
Dim sInvalid As String

lFile = FreeFile
Open sFName For Output As #lFile

' *** id
Print #lFile, "DSAA" 'VBLM SKIP

' *** nx ny
x_MIN = LBound(adbl, 1): x_MAX = UBound(adbl, 1)
y_MIN = LBound(adbl, 2): y_MAX = UBound(adbl, 2)
'# MS Print #lFile, gsFormat(X_Max - X_Min + 1); Chr$(32); _
    gsFormat(Y_Max - Y_Min + 1)
Print #lFile, CStr(x_MAX - x_MIN + 1); Chr$(32); CStr(y_MAX - y_MIN + 1)

With ieg
' *** xlo xhi
'# MS Print #lFile, gsFormat(.dblLeftBottomEastWgs); Chr$(32); _
    gsFormat(.dblRightTopEastWgs)
Print #lFile, LTrim$(Str$(.dblLeftBottomEastWgs)); Chr$(32); _
    LTrim$(Str$(.dblRightTopEastWgs))
' *** ylo yhi
'# MS Print #lFile, gsFormat(.dblLeftBottomNorthWgs); Chr$(32); _
    gsFormat(.dblRightTopNorthWgs)
Print #lFile, LTrim$(Str$(.dblLeftBottomNorthWgs)); Chr$(32); _
    LTrim$(Str$(.dblRightTopNorthWgs))
End With

' *** zlo zhi

```

```

    '~ Etwas beschleunigt!
    Debug.Assert ieg.eInvalidBehavior = iegMagic '~
    dbaStatistics adbl(0, 0), (x_MAX - x_MIN + 1) * (y_MAX - y_MIN + 1), _
        ieg.dblMagic, ByVal 0&, dbl_MIN, dbl_MAX
    '~   dbl_MIN = DOUBLE_MAX
    '~   dbl_MAX = DOUBLE_MIN
    '~   For X = X_Min To X_Max
    '~       For y = Y_Min To Y_Max
    '~           dbl = adbl(X, y)
    '~           If Not m_bInvalid(ieg, dbl) Then
    '~               If dbl < dbl_MIN Then dbl_MIN = dbl
    '~               If dbl > dbl_MAX Then dbl_MAX = dbl
    '~           End If
    '~       Next ' y
    '~   Next ' x
    dbl_MIN = dbl_MIN * ieg.dblMultiply
    dbl_MAX = dbl_MAX * ieg.dblMultiply
    '# MS Print #lFile, gsFormat(dbl_MIN); Chr$(32); gsFormat(dbl_MAX)
    Print #lFile, LTrim$(Str$(dbl_MIN)); Chr$(32); LTrim$(Str$(dbl_MAX))

    ' *** grid row 1, 2, 3,....
    If ieGWIB = ieGWIB17E38 Then
        sInvalid = "1.70141E+38"
    Else
        sInvalid = "0.00000"
    End If
    For y = y_MAX To y_MIN Step -1
        s = ""
        For x = x_MIN To x_MAX
            dbl = adbl(x, y)
            '~ If m_bInvalid(ieg, dbl) Then
            If dbl = ieg.dblMagic Then
                s = s & sInvalid & Chr$(32)
            Else
                dbl = dbl * ieg.dblMultiply
                '# MS s = s & gsFormat(dbl, "0.00000") & Chr$(32)
                s = s & fs.FormatDbl(dbl, 1, 5) & Chr$(32)
            End If
        Next ' x
        Print #lFile, s
    Next ' y

    Close #lFile
    '* Ende des Grd-File-Schreibens

    Exit Function

ErrorHandler:
    If gnErrorHandler(m_sMOD, "ieGrdWrite", Err, Erl) = vbRetry Then Resume
End Function

'*****
'***
'* ieLvlRead
'*****
'***
'* MS Leerzeilen führten bisher zu einer Exception. Mitten in der Legenden-
'* definition ist das ja noch statthaft, aber eine Leerzeile, die sich am
'* Ende eingeschlichen hat, sollte etwas gnädiger behandelt werden. Ich
'* ignoriere jetzt Leerzeilen.
Public Function ieLvlRead(sFName As String, aielRet() As ieLVLSTRUCT, lRetCount As
Long) As Boolean
    On Error GoTo ErrorHandler
    'VBLM OFF
    Dim bCommentOrEmpty As Boolean ' MS
    Dim l As Long
    Dim s As String, s2 As String, asX() As String
    Dim lFile As Long, lLine As Long, lSize As Long

    ' Rückgabeveriable plätten
    Erase aielRet: lSize = 0
    lRetCount = 0

    ' Datei öffnen
    lFile = FreeFile
    Open sFName For Input As #lFile

```

```

' Kennung - muss da sein
' LVL2
Line Input #lFile, s
lLine = lLine + 1
s = Trim$(s)
If StrComp(s, "LVL2", vbTextCompare) <> 0 Then
    s = "" & gsFTitle(sFName) & "" & Chr$(32) & _
        "is not a valid LVL2-file!"
    Err.Raise 7367, , s
End If

' Jetzt kommt entweder eine Kommentarzeile (beliebig oft) oder
' eine Wertezeile (beliebig oft)
' Level Flags LColor LStyle LWidth FFGColor FBGColor FPattern FMode
' 0.00001 0 "Black" "Solid" 0 "R255 G255 B153" "White" "Solid" 2
' 2.25001 0 "Black" "Solid" 0 "R255 G255 B102" "White" "Solid" 2
Do While Not EOF(lFile)

    Line Input #lFile, s
    lLine = lLine + 1

    s2 = Trim$(s)
    bCommentOrEmpty = (Len(s2) = 0)
    If Not bCommentOrEmpty Then
        bCommentOrEmpty = (Left$(s2, 1) = "'")
    End If
    If Not bCommentOrEmpty Then
        l = m_lSurferParse(s, asX())
        l = fs.ParseSurfer(s, asX())

        If l <= 0 Then GoTo ExcLine ' Ich erlaube keine Leerzeilen
        MS If Left$(asX(0), 1) = "" Then
            MS ' Kommentarzeile - gar nix machen
            MS Else

            ' Wertezeile - ich will min. 6 Spalten haben!
            If l < 6 Then GoSub ExcLine
            ' Spalte 1 muss numerisch sein
            If Not gbIsNumeric(asX(0)) Then GoSub ExcLine
            ' Aus Spalte 6 muss sich ein RGB-Wert bilden lassen
            If Not m_bSurferRGB(asX(5), l) Then GoSub ExcLine
            ' Binge - Werteklasse eintragen!
            GoSub ChkArray
            With aielRet(lRetCount)
                .dblFromIn = Val(asX(0))
                .lColor = l
            End With
            lRetCount = lRetCount + 1
        End If
    Loop ' Do While Not EOF(lFile)...

    ' Das war's!
    *** Return
ieLvlRead = True

UndTschuess:
On Error Resume Next
Close #lFile

Exit Function

ChkArray:
If lRetCount >= lSize Then
    lSize = lSize + 5
    ReDim Preserve aielRet(0 To lSize - 1)
End If
Return

ExcLine:
s = "An error occurred parsing file '" & _
    gsFTitle(sFName) & "' line " & lLine & "."
Err.Raise 7368, , s

ErrHandler:
If gnErrorHandler(m_sMOD, "ieLvlRead", Err, Erl) = vbRetry Then Resume
Resume UndTschuess
End Function

```

```

'*****
***
'* ieLvlWrite
'*****
***
Public Function ieLvlWrite(sFName As String, aiel() As ieLVLSTRUCT, ByVal lCount As
Long) As Boolean
    On Error GoTo ErrHandler
    'VBLM OFF
    Dim i As Long, i_MAX As Long
    Dim s As String

    Dim lR As Long, lG As Long, lB As Long
    Dim lFile As Long

    ' Die Datei öffnen
    lFile = FreeFile
    Open sFName For Output As lFile

    ' Das ist die Kennung des Surfer Level Formats
    Print #lFile, "LVL2"

    ' Das sind die Spaltenüberschriften
    ' Level: Bereichsgrenzeuntergrenze (inklusive)
    ' Flags: immer 0
    ' LColor: vermutlich Line Color - immer "Black"
    ' LStyle: vermutlich Line Style - immer "Solid"
    ' LWidth: vermutlich Line Width - immer 0 -> 1 Pixel
    ' FFGColor: vermtl. Fill Foreground Color - RGB-Wert "R141 G0 B114"
    ' FBGColor: vermtl. Fill Background Color - immer "White"
    ' FPattern: vermtl. Fill Pattern - immer "Solid"
    ' FMode: vermutlich Fill Mode - immer 2
    s = "Level Flags LColor LStyle LWidth FFGColor FBGColor FPattern FMode"
    Print #lFile, s

    i_MAX = lCount - 1
    For i = 0 To i_MAX
        With aiel(i)
            ' Level
            '# MS s = gsFormat(.dblFromIn) & Chr$(32)
            s = LTrim$(Str$(.dblFromIn)) & Chr$(32)
            ' Flags, LColor, LStyle und LWidth
            s = s & "0 " "Black" " "Solid" " 0" & Chr$(32)
            ' FFGColor
            RGBInv .lColor, lR, lG, lB
            s = s & "" "R" & lR & " G" & lG & " B" & lB & "" & Chr$(32)
            ' FBGColor, FPattern und FMode
            s = s & "" "White" " "Solid" " 2"

            Print #lFile, s           ' Raus damit
        End With
    Next ' i

    Close #lFile

    Exit Function

ErrHandler:
    If gnErrorHandler(m_sMOD, "ieLvlWrite", Err, Erl) = vbRetry Then Resume
End Function

Private Function m_bSurferRGB(sRGB As String, lRetColor As Long) As Boolean
    On Error GoTo ErrHandler
    Dim nR As Integer, nG As Integer, nB As Integer
    Dim i As Long
    Dim s As String, asX() As String

    '* Rückgabeveriable kicken
    lRetColor = vbBlack

    ' So sieht das Teil aus:
    ' R255 G255 B102
    nR = -1: nG = -1: nB = -1
    ' If m_lSurferParse(sRGB, asX()) = 3 Then
    If fs.ParseSurfer(sRGB, asX()) = 3 Then
        For i = 0 To 2

```

```
s = Mid$(asX(i), 2)
Select Case UCase$(Left$(asX(i), 1))
    Case "R": nR = Val(s) 'VBLM SKIP
    Case "G": nG = Val(s) 'VBLM SKIP
    Case "B": nB = Val(s) 'VBLM SKIP
    Case Else
        Exit For
End Select
Next ' i

If nR >= 0 And nG >= 0 And nB >= 0 Then
    *** Return
    m_bSurferRGB = True
    lRetColor = RGB(nR, nG, nB)
End If
End If

Exit Function

ErrorHandler:
If gnErrorHandler(m_sMOD, "m_bSurferRGB", Err, Erl) = vbRetry Then Resume
End Fu
```

**4.1.9.2 Dokumentation des API des Weizenaussaatmoduls**

```
(*
* Dateiname      : PASaatWWSchnittstelle.txt
* Version        : 0.9
* Copyright      : pro_Plant GmbH
* Autor          : A. Rotterdam (Ro),
*                H. Schencking (HMS)
* Beschreibung   : Schnittstellenbeschreibung für das Winterweizen-
*                Aussaatmodul implementiert in der DLL PASaatWW.dll
* Historie       :
* 08.02.2002 (HMS): Veröffentlichung der Version 0.9
*)
```

(\*\*\*\*\*)

## Global Vorgaben und Typen

-----

- Alle Prozeduren und Funktionen sind von der Aufrufkonvention "stdcall".
- Der Type TAussaatHandle ist vom Type Integer (4 Byte).
- Boolean ist ein 1 Byte Wert.
- PChar ist eine Null-terminierter String.
- Double ist ein 8 Byte Fließkommewert.
- Integer ist stets 4 Byte breit und stets vorzeichenbehaftet.
- var Parameter müssen 'by reference', die anderen 'by value' übergeben werden.

## Funktionsgruppen

-----

- a) Aussaat-Modul initialisieren, starten und beenden:
  - AussaatInit
  - AussaatRun
  - AussaatExit
- b) Setter-Methoden: Dienen dazu, jede Eigenschaft des Aussaat-Moduls programmtechnisch zu steuern. Diese Methoden sind nach dem 'AussaatInit' aufrufbar, weil sie als Parameter den AussaatHandle benötigen, der von der Funktion AussaatInit geliefert wird. Diese Methoden werden sinnvollerweise vor dem 'AussaatRun' aufgerufen.
- c) Getter-Methoden: Dienen dazu, jede Eigenschaft des Aussaat-Moduls auszulesen. Diese Methoden sind grundsätzlich nach dem 'AussaatInit', und vor dem 'AussaatExit' aufzurufen. Sinnvoll ist der Aufruf nach dem 'AussaatRun'.

## Beispiel

-----

Aus einem Delphi-Programm heraus, sieht der Aufruf des Moduls folgendermaßen aus:

```
procedure AussaatModulAufruf;
var
  newHandle : TAussaatHandle;
  ok        : Boolean;
  niederschlag: Integer;
begin
  newHandle := AussaatInit('pro_Plant AussaatTest');

  if newHandle <> nil then begin
    // Hier könnten Setter-Methoden aufgerufen werden
    // Beispiel : der Niederschlag
    ok := SetNiederschlag(newHandle, 500);
```

```

        ok := AussaatRun(newHandle);

        // Hier könnten Getter-Methoden aufgerufen werden
        // Beispiel : der Niederschlag
        ok := GetNiederschlag(newHandle, niederschlag);

        if (niederschlag <> 500) then
            ShowMessage('Sie haben den Niederschlag geändert.');
```

if ok then  
 AussaatExit(newHandle);

end;  
end;  
\*\*\*\*\*)

(\*  
\* Funktionen  
\*)

{{ -----  
Funktion : AussaatExit()  
Beendet das AussaatModul.

Parameter :  
Als Übergabeparameter wird der AussaatHandle übergeben,  
der von AussaatInit zurückgeliefert wird.

Rückgabe :  
Der Rückgabewert ist True, wenn die Beendigung erfolgreich war.  
}

function AussaatExit(AussaatHandle: TAussaatHandle): Boolean;

{{ -----  
Funktion : AussaatInit()  
Initialisieren des Aussaatmoduls.

Parameter :  
Beschreibender freier Text für das Aussaatmodul.  
Der Text erscheint im Titel des Dialogs.

Rückgabe :  
Das AussaatHandle wird als TAussaatHandle zurückgegeben.  
Der Rückgabewert ist = 0, falls die Initialisierung nicht  
erfolgreich stattfinden konnte.  
TAussaatHandle entspricht dem Typ Integer (4 Byte).  
}

function AussaatInit(aDescription: PChar): TAussaatHandle;

{{ -----  
Funktion : AussaatRun()  
Führt das Aussaatmodul aus, und erzeugt den Eingabedialog.

Parameter :  
Das Aussaathandle, das von AussaatInit zurückgegeben wurde.

Rückgabe :  
True wenn der Dialog ordnungsgemäß erzeugt wurde.  
False bei Fehler.  
}

function AussaatRun(AussaatHandle: TAussaatHandle): Boolean;

{{ -----  
Funktion : GetAckerzahl()  
Gibt die einheitliche Ackerzahl zurück.

Parameter :  
Das AussaatHandle zu dem Aussaatmodul.  
Variable vom Typ Double, der nach Beendigung die Ackerzahl enthält.

Rückgabe :  
True bei erfolgreicher Werterückgabe  
False bei Fehler  
}

```

function GetAckerzahl(      handle: TAussaatHandle;
                           var Ackerzahl: Double): Boolean;

{{ -----
Funktion : GetAckerzahlKarte()
  Gibt den Namen der Datei zurück, die die Karte mit den Ackerzahlen
  enthält.
  Der Name enthält den vollständigen Pfad und die Endung (.shp).

Parameter :
  Das AussaatHandle zu dem Aussaatmodul.

  Variable vom Typ PChar, der nach Beendigung den Dateinamen enthält

Rückgabe :
  True bei erfolgreicher Werterückgabe
  False bei Fehler
}
function GetAckerzahlKarte(handle: TAussaatHandle;
                           var Dateiname: PChar): Boolean;

{{ -----
Funktion : GetBenutzeAckerzahl()
  Ermittelt, ob Ertragspotenziale über Ackerzahlen brechent werden
  oder über eine Ertragspotenzialkarte vorliegen.

Parameter :
  Das AussaatHandle zu dem Aussaatmodul.

Rückgabe :
  True : Ackerzahlen werden verwendet
  False : Ertragspotenzialkarte wird verwendet
}
function GetBenutzeAckerzahl(handle: TAussaatHandle): Boolean;

{{ -----
Funktion : GetBenutzeAckerzahlEinheitlich()
  Ermittelt, ob die einheitliche Ackerzahl verwendet wird.
  (Wenn nicht, wird die Ackerzahlkarte verwendet)
  (= not GetBenutzeAckerzahlKarte())

Parameter :
  Das AussaatHandle zu dem Aussaatmodul.

Rückgabe :
  True : Einheitliche Ackerzahl wird verwendet
  False : Ackerzahlkarte wird verwendet
}
function GetBenutzeAckerzahlEinheitlich(handle: TAussaatHandle): Boolean;

{{ -----
Funktion : GetBenutzeAckerzahlKarte()
  Ermittelt ob im Aussaatmodul eine Ackerzahlkarte benutzt wird.
  (Wenn nicht, wird der einheitliche Ackerzahlwert benutzt.)
  (= not GetBenutzeAckerzahlEinheitlich())

Parameter :
  Das AussaatHandle zu dem Aussaatmodul.

Rückgabe :
  True : Ackerzahlkarte wird verwendet
  False : Ackerzahlkarte wird nicht verwendet
}
function GetBenutzeAckerzahlKarte(handle: TAussaatHandle): Boolean;

{{ -----
Funktion : GetBenutzeBodenSubstratEinheitlich()
  Ermittelt, ob ein einheitliches Bodensubstrat benutzt wird.
  (Wenn nicht, wird eine Bodensubstratkarte benutzt.)
  (= not GetBenutzeBodenSubstratKarte())

```

Parameter :

Das AussaatHandle zu dem Aussaatmodul.

Rückgabe :

True : Das einheitliche Bodensubstrat wird benutzt

False : Das einheitliche Bodensubstrat wird nicht benutzt

```
}
function GetBenutzeBodenSubstratEinheitlich(handle: TAussaatHandle): Boolean;
```

```
{{ -----
```

Funktion : GetBenutzeBodenSubstratKarte()

Ermittelt, ob eine Bodensubstratkarte benutzt wird.

(Wenn nicht, wird der einheitliche Bodensubstratwert benutzt.)

( = not GetBenutzeBodenSubstratEinheitlich()

Parameter :

Das AussaatHandle zu dem Aussaatmodul.

Rückgabe :

True : Die Bodensubstratkarte wird benutzt

False : Die Bodensubstratkarte wird nicht benutzt

```
}
function GetBenutzeBodenSubstratKarte(handle: TAussaatHandle): Boolean;
```

```
{{ -----
```

Funktion : GetBenutzeErtragsPotenzialKarte()

Ermittelt, ob eine Ertragspotenzialkarte benutzt wird.

(Wenn nicht, werden Ackerzahldaten benutzt.)

( = not GetBenutzeAckerzahl()

Parameter :

Das AussaatHandle zu dem Aussaatmodul.

Rückgabe :

True : Die Ertragspotenzialkarte wird benutzt

False : Das Ertragspotenzial wird über Ackerzahlen berechnet.

```
}
function GetBenutzeErtragsPotenzialKarte(handle: TAussaatHandle): Boolean;
```

```
{{ -----
```

Funktion : GetBodenfeuchteBeiSaat()

Ermittelt den Status der Bodenfeuchte bei Saat.

Parameter :

Das AussaatHandle zu dem Aussaatmodul.

Integer: BodenfeuchteBeiSaat

Die Bodenfeuchte kann folgende Werte annehmen

0 : Gut

1 : Mittel

2 : Schlecht

Rückgabe :

True bei erfolgreicher Werterückgabe

False bei Fehler

```
}
function GetBodenfeuchteBeiSaat(handle: TAussaatHandle;
var BodenfeuchteBeiSaat: Integer): Boolean;
```

```
{{ -----
```

Funktion : GetBodenSaatgutKontakt()

Ermittelt den Status des Kontaktes zwischen Boden und Saatgut.

Parameter :

Das AussaatHandle zu dem Aussaatmodul.

Integer: BodenSaatgutKontakt

Der Kontakt kann folgende Werte annehmen

0 : Gut

1 : Gestört

2 : Ungenügend

```

Rückgabe :
    True bei erfolgreicher Werterückgabe
    False bei Fehler
}
function GetBodenSaatgutKontakt(handle: TAussaatHandle;
                                var BodenSaatgutKontakt: Integer): Boolean;

{{ -----
Funktion : GetBodenSubstrat()
    Ermittelt die Bezeichnung des Bodensubstrates.

Parameter :
    Das AussaatHandle zu dem Aussaatmodul.

    Variable vom Typ PChar die den Namen des Substrates beinhalten wird

Rückgabe :
    True bei erfolgreicher Werterückgabe
    False bei Fehler
}
function GetBodenSubstrat( handle: TAussaatHandle;
                            var Bodensubstrat: PChar): Boolean;

{{ -----
Funktion : GetBodenSubstratKarte()
    Ermittelt den Dateinamen der Bodensubstratkarte.
    Dieser enthält den vollständigen Pfad und die Endung '.shp'.

Parameter :
    Das AussaatHandle zu dem Aussaatmodul.

    Eine Variable vom Typ PChar,
    die den Dateinamen der Karte beinhalten wird.

Rückgabe :
    True bei erfolgreicher Werterückgabe
    False bei Fehler
}
function GetBodenSubstratKarte( handle: TAussaatHandle;
                                var Dateiname: PChar): Boolean;

{{ -----
Funktion : GetBundesland()
    Ermittelt das gewählte Bundesland

Parameter :
    Das AussaatHandle zu dem Aussaatmodul.

    Eine Variable vom Typ PChar, die den Namen des
    Bundeslandes beinhalten wird.

Rückgabe :
    True bei erfolgreicher Werterückgabe
    False bei Fehler
}
function GetBundesland( handle: TAussaatHandle;
                        var Bundesland: PChar): Boolean;

{{ -----
Funktion : GetErtragsPotenzialKarte()
    Ermittelt den Dateinamen der ErtragsPotenzialkarte.
    Dieser enthält den vollständigen Pfad und die Endung '.shp'.

Parameter :
    Das AussaatHandle zu dem Aussaatmodul.

    Eine Variable vom Typ PChar,
    die den Dateinamen der Karte beinhalten wird.

Rückgabe :
    True bei erfolgreicher Werterückgabe
    False bei Fehler

```

```

}
function GetErtragsPotenzialKarte(handle: TAussaatHandle;
                                var dateiname: PChar): Boolean;

```

```

{{ -----
Funktion : GetKeimfaehigkeit()
    Ermittelt die Keimfähigkeit für dieses Aussaatmodul

```

```

Parameter :
    Das AussaatHandle zu dem Aussaatmodul.

    Eine Variable vom Typ TPercentage (Double), die den Wert der
    Keimfähigkeit beinhalten wird.

```

```

Rückgabe :
    True bei erfolgreicher Werterückgabe
    False bei Fehler

```

```

}
function GetKeimfaehigkeit( handle: TAussaatHandle;
                            var Keimfaehigkeit: TPercentage): Boolean;

```

```

{{ -----
Funktion : GetKoordinaten()
    Ermittelt die Koordinaten des umgebenden Rechteckes des zu
    berechnenden Gebietes. Diese Angaben sind nur sinnvoll,
    wenn man mit einheitlicher Ackerzahl arbeitet, und keine anderen
    Karteninformationen aus einer Ackerzahl- oder
    Ertragspotenzialkarte zur Verfügung hat.

```

```

Parameter :
    Das AussaatHandle zu dem Aussaatmodul.

    Vier Variablen vom Typ Double, die jeweils die untere linke
    und obere rechte Ecke der Rechteckes darstellen.
    (xmin, xmax, ymin, ymax).
    Die Werte werden im WGS84 Format zurückgegeben.

```

```

Rückgabe :
    True bei erfolgreicher Werterückgabe
    False bei Fehler

```

```

}
function GetKoordinaten(handle: TAussaatHandle;
                        var Xmin: Double;
                        var Xmax: Double;
                        var Ymin: Double;
                        var Ymax: Double): Boolean;

```

```

{{ -----
Funktion : GetKorrekturErtragserwartung()
    Ermittelt den Korrekturwert für die Ertragserwartung.

```

```

Parameter :
    Das AussaatHandle zu dem Aussaatmodul.
    Eine Variable vom Typ Integer,
    die den KorrekturWert beinhalten wird.
    Der Korrekturwert hat folgende feste Werte :
    -15, -10, -5, 0, 5, 10, 15

```

```

Rückgabe :
    True bei erfolgreicher Werterückgabe
    False bei Fehler

```

```

}
function GetKorrekturErtragserwartung(
    handle: TAussaatHandle;
    var KorrekturErtragserwartung: Integer): Boolean;

```

```

{{ -----
Funktion : GetNiederschlag()
    Ermittelt die Niederschlagsmenge für dieses Aussaatmodul.

```

```

Parameter :
    Das AussaatHandle zu dem Aussaatmodul.

```

Variable vom Typ Integer, die den Niederschlag beinhalten wird.  
Der Niederschlag kann im Bereich von 450 mm bis 850 mm liegen

Rückgabe :

True bei erfolgreicher Werterückgabe  
False bei Fehler

```
function GetNiederschlag(handle: TAussaatHandle;
                        var Niederschlag: Integer): Boolean;
```

```
{{ -----
Funktion : GetRasterbreite()
```

Ermittelt die Rasterbreite der Ergebniskarten für diese Aussaatmodul.

Parameter :

Das AussaatHandle zu dem Aussaatmodul.

Variable vom Typ Integer, die die Rasterbreite in Metern  
beinhalten wird.

Rückgabe :

True bei erfolgreicher Werterückgabe  
False bei Fehler

```
function GetRasterbreite(handle: TAussaatHandle;
                        var Rasterbreite: Integer): Boolean;
```

```
{{ -----
Funktion : GetRealerSaatTermin()
```

Ermittelt den realen Saattermin für dieses Aussaatmodul

Parameter :

Das AussaatHandle zu dem Aussaatmodul.

Variable vom Typ Integer, die den realen Saattermin beinhalten wird.  
Der reale Saattermin kann folgende Werte annehmen.

0 : zu früh  
1 : ideal  
2 : 1 Woche zu spät  
3 : 2 Wochen zu spät  
4 : 3 Wochen zu spät  
5 : 4 Wochen zu spät

Rückgabe :

True bei erfolgreicher Werterückgabe  
False bei Fehler

```
function GetRealerSaatTermin(handle: TAussaatHandle;
                            var RealerSaatTermin: Integer): Boolean;
```

```
{{ -----
Funktion : GetRegion()
```

Ermittelt die gewählte Region

Parameter :

Das AussaatHandle zu dem Aussaatmodul.

Variable vom Typ PChar, die den Namen der Region beinhalten wird.

Rückgabe :

True bei erfolgreicher Werterückgabe  
False bei Fehler

```
function GetRegion(handle: TAussaatHandle; var Region: PChar): Boolean;
```

```
{{ -----
Funktion : GetReliefEinflussKarte()
```

Ermittelt den Dateinamen der ReliefEinflusskarte.  
Dieser enthält den vollständigen Pfad und die Endung '.shp'.

Parameter :

Das AussaatHandle zu dem Aussaatmodul.

Variable vom Typ PChar, die den Namen der Karte beinhalten wird.

Rückgabe :

True bei erfolgreicher Werterückgabe  
False bei Fehler

}

function GetReliefEinflussKarte(handle: TAussaatHandle;

var Dateiname: PChar): Boolean;

{{ -----

Funktion : GetSaatBeetQualitaet()

Ermittelt den Status der SaatBettQualitaet.

Parameter :

Das AussaatHandle zu dem Aussaatmodul.

Integer: SaatBettQualitaet

Die Qualität kann folgende Werte annehmen

0 : Gut

1 : Mittel

2 : Schlecht

Rückgabe :

True bei erfolgreicher Werterückgabe  
False bei Fehler

}

function GetSaatBettQualitaet(handle: TAussaatHandle;

var SaatBettQualitaet: Integer): Boolean;

{{ -----

Funktion : GetSaatTiefe()

Ermittelt den Status der Saattiefe

Parameter :

Das AussaatHandle zu dem Aussaatmodul.

Integer: Saattiefe

Die Saattiefe kann folgende Werte annehmen

0 : '< 3 cm (zu flach);

1 : '3 - 5 cm (normal);

2 : '> 5 cm (zu tief);

Rückgabe :

True bei erfolgreicher Werterückgabe  
False bei Fehler

}

function GetSaatTiefe(handle: TAussaatHandle;

var SaatTiefe: Integer): Boolean;

{{ -----

Funktion : GetSorte()

Ermittelt die gewählte Sorte

Parameter :

Das AussaatHandle zu dem Aussaatmodul.

Variable vom Typ PChar, die den Namen der Sorte beinhalten wird.

Rückgabe :

True bei erfolgreicher Werterückgabe  
False bei Fehler

}

function GetSorte(handle: TAussaatHandle; var Sorte: PChar): Boolean;

{{ -----

Funktion : GetSpalteAckerzahl()

Ermittelt den Namen der Datenbankspalte für die  
Ackerzahl in der Ackerzahlkarte

Parameter :

Das AussaatHandle zu dem Aussaatmodul.

Variable vom Typ PChar die den Namen der Spalte beinhalten wird.

Rückgabe :

True bei erfolgreicher Werterückgabe  
False bei Fehler

```
}
function GetSpalteAckerzahl(handle: TAussaatHandle;
                               var Spalte: PChar): Boolean;
```

```
{{ -----
Funktion : GetSpalteBodensubstratKarte()
    Ermittelt den Namen der Datenbankspalte für das Bodensubstrat
    in der Bodensubstratkarte
```

Parameter :

Das AussaatHandle zu dem Aussaatmodul.

Variable vom Typ PChar, die den Namen der Spalte beinhalten wird.

Rückgabe :

True bei erfolgreicher Werterückgabe  
False bei Fehler

```
}
function GetSpalteBodensubstratKarte(handle: TAussaatHandle;
                                       var Spalte: PChar): Boolean;
```

```
{{ -----
Funktion : GetSpalteReliefEinflussKarte()
    Ermittelt den Namen der Datenbankspalte für den
    ReliefEinfluss in der ReliefEinflusskarte
```

Parameter :

Das AussaatHandle zu dem Aussaatmodul.

Variable vom Typ PChar, die den Namen der Spalte beinhalten wird.

Rückgabe :

True bei erfolgreicher Werterückgabe  
False bei Fehler

```
}
function GetSpalteReliefEinflussKarte(
    handle: TAussaatHandle;
    var SpalteReliefEinflussKarte: PChar): Boolean;
```

```
{{ -----
Funktion : GetTKG()
    Ermittelt das gewählte TKG
```

Parameter :

Das AussaatHandle zu dem Aussaatmodul.

Variable vom Typ Double, die den Wert des TKG beinhalten wird.

Rückgabe :

True bei erfolgreicher Werterückgabe  
False bei Fehler

```
}
function GetTKG(handle: TAussaatHandle; var TKG: Double): Boolean;
```

```
{{ -----
Funktion : GetVorfrucht()

    Ermittelt den Namen der gewählten Vorfrucht
```

Parameter :

Das AussaatHandle zu dem Aussaatmodul.

Variable vom Typ PChar, die den Namen der Vorfrucht beinhalten wird.

Rückgabe :

```

    True bei erfolgreicher Werterückgabe
    False bei Fehler
}
function GetVorfrucht(handle:TAussaatHandle;
                    var Vorfrucht: PChar): Boolean;

{{ -----
Funktion : SetAckerzahl()
    Setzt die einheitliche Ackerzahl

Parameter :
    Das AussaatHandle zu dem Aussaatmodul.
    Der Wertebereich für die einheitliche Ackerzahl liegt zwischen
    25 und 85.

Rückgabe :
    True bei erfolgreicher Werteübergabe
    False, falls die angegebene Ackerzahl nicht im angegebenen Werte-
    bereich liegt.
}
function SetAckerzahl(handle: TAussaatHandle; Ackerzahl: Double): Boolean;

{{ -----
Funktion : SetAckerzahlKarte()
    Setzt den Dateinamen der Ackerzahlkarte

Parameter :
    Das AussaatHandle zu dem Aussaatmodul.
    Wert von Typ PChar,
    der den Dateinamen inclusive Endung '.shp' enthält.

Rückgabe :
    True bei erfolgreicher Werteübergabe
    False, falls die angegebene Karte nicht existiert, oder nicht
    gelesen werden kann.
}
function SetAckerzahlKarte(handle: TAussaatHandle;
                        Dateiname: PChar): Boolean;

{{ -----
Funktion : SetBenutzeAckerzahl()
    Wählt aus, dass Ackerzahldaten zur Berechnung des Ertragspotenzials
    herangezogen wird, und nicht eine Ertragspotenzialkarte verwendet
    werden soll.
    (vgl. SetBenutzeErtragsPotenzialKarte());

Parameter :
    Das AussaatHandle zu dem Aussaatmodul.
}
procedure SetBenutzeAckerzahl(handle: TAussaatHandle);

{{ -----
Funktion : SetBenutzeAckerzahlEinheitlich()
    Wählt aus, ob eine einheitliche Ackerzahl verwendet werden soll.
    (vgl. SetBenutzeAckerzahlKarte());

Parameter :
    Das AussaatHandle zu dem Aussaatmodul.
}
procedure SetBenutzeAckerzahlEinheitlich(handle: TAussaatHandle);

{{ -----
Funktion : SetBenutzeAckerzahlKarte()
    Wählt aus, dass eine Ackerzahlkarte werden soll.
    (vgl. SetBenutzeAckerzahlKarte());

Parameter :
    Das AussaatHandle zu dem Aussaatmodul.
}
procedure SetBenutzeAckerzahlKarte(handle: TAussaatHandle);

```

```

{{ -----
Funktion : SetBenutzeBodenSubstratEinheitlich()
    Wählt aus, dass ein einheitliches Bodensubstrat verwendet werden soll.
    (vgl. SetBenutzeBodenSubstratKarte());

Parameter :
    Das AussaatHandle zu dem Aussaatmodul.
}
procedure SetBenutzeBodenSubstratEinheitlich(handle: TAussaatHandle);

{{ -----
Funktion : SetBenutzeBodenSubstratKarte()
    Wählt aus, dass eine Bodensubstratkarte verwendet werden soll.
    (vgl. SetBenutzeBodenSubstratEinheitlich())
Parameter :
    Das AussaatHandle zu dem Aussaatmodul.
}
procedure SetBenutzeBodenSubstratKarte(handle: TAussaatHandle);

{{ -----
Funktion : SetBenutzeErtragsPotenzialKarte()
    Wählt aus, dass eine ErtragsPotenzialkarte verwendet werden soll.
    (vgl. BenutzeAckerzahl())

Parameter :
    Das AussaatHandle zu dem Aussaatmodul.
}
procedure SetBenutzeErtragsPotenzialKarte(handle: TAussaatHandle);

{{ -----
Funktion : SetBodenfeuchteBeiSaat()
    Setzt die Bodenfeuchte bei Saat.

Parameter :
    Das AussaatHandle zu dem Aussaatmodul.

    Integer: BodenfeuchteBeiSaat
        Die Bodenfeuchte kann folgende Werte annehmen
        0 : Gut
        1 : Mittel
        2 : Schlecht

Rückgabe :
    True bei erfolgreicher Werteübergabe
    False, falls ein anderer Wert als 0, 1 oder 2 angegeben wird.
}
function SetBodenfeuchteBeiSaat(
                                handle: TAussaatHandle;
                                BodenfeuchteBeiSaat: Integer): Boolean;

{{ -----
Funktion : SetBodenSaatgutKontakt()
    Setzt die Boden-Saatgut-Kontakt

Parameter :
    Das AussaatHandle zu dem Aussaatmodul.

    Integer: BodenSaatgutKontakt
        Der Kontakt kann folgende Werte annehmen
        0 : Gut
        1 : Gestört
        2 : Ungenügend

Rückgabe :
    True bei erfolgreicher Werteübergabe
    False, falls ein anderer Wert als 0, 1 oder 2 angegeben wird.
}
function SetBodenSaatgutKontakt(
                                handle: TAussaatHandle;
                                BodenSaatgutKontakt: Integer): Boolean;

```

```

{{ -----
Funktion : SetBodenSubstrat()
    Setzt den Wert für das einheitliche Bodensubstrat

Parameter :
    Das AussaatHandle zu dem Aussaatmodul.

    Wert des Typs PChar der den Namen des Bodensubstrates beinhaltet
    Folgende Werte sind möglich :
        'IS','SL','sL','S','SI','L','LT','T','Mo';

Rückgabe :
    True bei erfolgreicher Werteübergabe
    False, falls ein anderer Wert außerhalb der oben angegebenen Liste
    übergeben wurde.
}
function SetBodenSubstrat(handle: TAussaatHandle;
                        Bodensubstrat: PChar): Boolean;

```

```

{{ -----
Funktion : SetBodensubstratKarte()
    Setzt den Dateinamen der Bodensubstratkarte.

Parameter :
    Das AussaatHandle zu dem Aussaatmodul.

    Wert des Typs PChar mit dem Dateinamen der Karte inklusive der
    Endung '.shp'.

Rückgabe :
    True bei erfolgreicher Werteübergabe
    False bei Fehler
}
function SetBodensubstratKarte(handle: TAussaatHandle;
                            Dateiname: PChar): Boolean;

```

```

{{ -----
Funktion : SetBundesland()
    Setzt das Bundesland

Parameter :
    Das AussaatHandle zu dem Aussaatmodul.

    Wert des Typs PChar mit dem Namen des Bundeslandes
    Erlaubt sind alle in der Tabelle Regionen.dbf angegebenen
    Bundesländer.

Rückgabe :
    True bei erfolgreicher Werteübergabe
    False, falls das Bundesland nicht in der Tabelle vorhanden ist.
}
function SetBundesland(handle: TAussaatHandle; Bundesland: PChar): Boolean;

```

```

{{ -----
Funktion : SetErtragsPotenzialKarte()
    Setzt den Dateinamen der ErtragsPotenzialkarte

Parameter :
    Das AussaatHandle zu dem Aussaatmodul.

    Wert des Typs PChar mit dem Dateinamen der Karte inklusive der
    Endung '.mwk' and '.grd'.

Rückgabe :
    True bei erfolgreicher Werteübergabe
    False, falls die Datei nicht vorhanden ist oder nicht gelesen
    werden kann.
}
function SetErtragsPotenzialKarte(handle: TAussaatHandle;
                                dateiname: PChar): Boolean;

```

```

{{ -----
Funktion : SetKeimfaehigkeit()
    Setzt die Keimfähigkeit

Parameter :
    Das AussaatHandle zu dem Aussaatmodul.

    Wert des Types Double mit dem Prozentwert der Keinfähigkeit
        Wertebereich (0 bis 100)

Rückgabe :
    True bei erfolgreicher Werteübergabe
    False, falls der Wert außerhalb des Bereichs 1 bis 100 ist.
}
function SetKeimfaehigkeit(handle: TAussaatHandle;
                           Keimfaehigkeit: TPercentage): Boolean;

```

```

{{ -----
Funktion : SetKoordinaten()
    Setzt die Koordinaten des umgebenden Rechtecks
    Die Koordinaten können entweder in Gauss-Krüger oder in WGS84
        eingegeben werden.
    Aber ein kompletter Satz muss einheitlich angegeben werden

Parameter :
    Das AussaatHandle zu dem Aussaatmodul.

    4 Werte vom Typ Double mit Xmin, Xmax, Ymin, Ymax

Rückgabe :
    True bei erfolgreicher Werteübergabe
    False, falls min-Werte > max-Werte.
}
function SetKoordinaten(handle: TAussaatHandle;
                       Xmin: Double;
                       Xmax: Double;
                       Ymin: Double;
                       Ymax: Double): Boolean;

```

```

{{ -----
Funktion : SetKorrekturErwartungswert()
    Setzt den Korrekturwert für die Ertragserwartung.

Parameter :
    Das AussaatHandle zu dem Aussaatmodul.

    Wert von Typ Integer für die Korrektur, der folgende Werte annehmen
        darf: -15, -10, -5, 0, 5, 10, 15

Rückgabe :
    True bei erfolgreicher Werteübergabe
    False, falls der Wert nicht einem der oben angegebenen entspricht.
}
function SetKorrekturErtragserwartung(
    handle: TAussaatHandle;
    KorrekturErtragserwartung: Integer): Boolean;

```

```

{{ -----
Funktion : SetNiederschlag()
    Setzt den Niederschlag

Parameter :
    Das AussaatHandle zu dem Aussaatmodul.

    Wert vom Typ Integer, um den Niederschlag anzugeben.
    Der Wertebereich liegt zwischen 450 mm und 850 mm pro Jahr

Rückgabe :
    True bei erfolgreicher Werteübergabe
    False, falls der Niederschlagswert außerhalb des gültigen Bereiches
    ist.
}
function SetNiederschlag(handle: TAussaatHandle;

```

Niederschlag: Integer): Boolean;

{{ -----

Funktion : SetRasterbreite()  
Setzt die Rasterbreite

Parameter :

Das AussaatHandle zu dem Aussaatmodul.

Wert vom Typ Double, um die Rasterbreite in Metern anzugeben.

Rückgabe :

True bei erfolgreicher Werteübergabe  
False, falls Rasterbreite kleiner gleich Null ist.

}  
function SetRasterbreite(handle: TAussaatHandle;

Rasterbreite: Integer): Boolean;

{{ -----

Funktion : SetRealerSaatTermin()  
Setzt den realen Saattermin

Parameter :

Das AussaatHandle zu dem Aussaatmodul.

Wert vom Typ Integer, der den realen Saattermin beinhaltet.

Der reale Saattermin kann folgende Werte annehmen.

0 : zu früh

1 : ideal

2 : 1 Woche zu spät

3 : 2 Wochen zu spät

4 : 3 Wochen zu spät

5 : 4 Wochen zu spät

Rückgabe :

True bei erfolgreicher Werteübergabe  
False, falls keiner der oben angegebenen Werte übergeben wurde.

}  
function SetRealerSaatTermin(handle: TAussaatHandle;

RealerSaatTermin: Integer): Boolean;

{{ -----

Funktion : SetRegion()  
Setzt die Region

Parameter :

Das AussaatHandle zu dem Aussaatmodul.

Wert vom Typ PChar der den Namen der Region enthält.

Zugelassen sind nur die Werte aus der Tabelle 'Regionen.dbf' aus der Spalte 'REGION'.

Rückgabe :

True bei erfolgreicher Werteübergabe  
False, falls eine unbekannte Region übergeben wurde.

}  
function SetRegion(handle: TAussaatHandle; Region: PChar): Boolean;

{{ -----

Funktion : SetReliefEinflussKarte()  
Setzt den Dateinamen der ReliefEinflusskarte

Parameter :

Das AussaatHandle zu dem Aussaatmodul.

Wert des Types PChar, mit dem Dateinamen der Karte inklusive der Endung '.shp'.

Rückgabe :

True bei erfolgreicher Werteübergabe  
False, falls die Datei nicht existiert, oder nicht gelesen werden kann.

```

}
function SetReliefEinflussKarte(handle: TAussaatHandle;
                               Dateiname: PChar): Boolean;

```

```

{{ -----
Funktion : SetSaatBettQualitaet()
        Setzt die Saatbeetqualität

```

```

Parameter :
        Das AussaatHandle zu dem Aussaatmodul.

        Integer: SaatBettQualitaet
        Die Qualität kann folgende Werte annehmen
        0 : Gut
        1 : Mittel
        2 : Schlecht

```

```

Rückgabe :
        True bei erfolgreicher Werteübergabe
        False, falls der Wert nicht 0, 1 oder 2 ist.
}

```

```

function SetSaatBettQualitaet(handle: TAussaatHandle;
                               SaatBettQualitaet: Integer): Boolean;

```

```

{{ -----
Funktion : SetSaatTiefe()
        Setzt die Saattiefe

```

```

Parameter :
        Das AussaatHandle zu dem Aussaatmodul.

        Integer: Saattiefe
        Die Saattiefe kann folgende Werte annehmen
        0 : '< 3 cm (zu flach)';
        1 : '3 - 5 cm (normal)';
        2 : '> 5 cm (zu tief)';

```

```

Rückgabe :
        True bei erfolgreicher Werteübergabe
        False, falls der Wert nicht 0, 1 oder 2 ist.
}

```

```

function SetSaatTiefe(handle:TAussaatHandle;
                      Saattiefe: Integer): Boolean;

```

```

{{ -----
Funktion : SetSorte()
        Setzt die Sorte

```

```

Parameter :
        Das AussaatHandle zu dem Aussaatmodul.

        Wert vom Typ PChar, der den Namen der Sorte beinhaltet.
        Zugelassen sind alle in der Tabelle 'Sorten.dbf'
        aufgeführten Sorten.

```

```

Rückgabe :
        True bei erfolgreicher Werteübergabe
        False, falls eine unbekannt Sorte übergeben wurde.
}

```

```

function SetSorte(handle: TAussaatHandle; Sorte: PChar): Boolean;

```

```

{{ -----
Funktion : SetSpalteAckerzahl()
        Setzt den Spaltennamen der Datenbankspalte für die Ackerzahl in der
        Ackerzahlkarte.

```

```

Parameter :
        Das AussaatHandle zu dem Aussaatmodul.

        Wert von Typ PChar, der den Namen der Datenbankspalte
        für die Ackerzahl enthält.

```

```

Rückgabe :
    True bei erfolgreicher Werteübergabe
    False, falls der Spaltenname nicht in der Tabelle vorkommt.
}
function SetSpalteAckerzahl(handle: TAussaatHandle;
                               Spalte: PChar): Boolean;

```

```

{{ -----
Funktion : SetSpalteBodenSubstratKarte()
    Setzt den Spaltennamen der Datenbankspalte für das Bodensubstrat
    in der Bodensubstratkarte.

```

```

Parameter :
    Das AussaatHandle zu dem Aussaatmodul.

    Wert von Typ PChar, der den Namen der Datenbankspalte
    für das Bodensubstrat enthält

```

```

Rückgabe :
    True bei erfolgreicher Werteübergabe
    False, falls der Spaltenname nicht in der Tabelle vorkommt.
}
function SetSpalteBodenSubstratKarte(handle: TAussaatHandle;

```

Spalte: PChar): Boolean;

```

{{ -----
Funktion : SetSpalteReliefEinflussKarte()
    Setzt den Spaltennamen der Datenbankspalte für den Reliefeinfluss
    in der Reliefeinflusskarte.

```

```

Parameter :
    Das AussaatHandle zu dem Aussaatmodul.

    Wert von Typ PChar, der den Namen der Datenbankspalte
    für den Reliefeinfluss enthält

```

```

Rückgabe :
    True bei erfolgreicher Werteübergabe
    False, falls der Spaltenname nicht in der Tabelle vorkommt.
}
function SetSpalteReliefEinflussKarte(
    handle: TAussaatHandle;
    SpalteReliefEinflussKarte: PChar): Boolean;

```

```

{{ -----
Funktion : SetTKG()
    Setzt das TKG

```

```

Parameter :
    Das AussaatHandle zu dem Aussaatmodul.

    Wert vom Typ Double, der das Tausendkorngewicht enthält

```

```

Rückgabe :
    True bei erfolgreicher Werteübergabe
    False bei Fehlern
}
function SetTKG(handle: TAussaatHandle; TKG: Double): Boolean;

```

```

{{ -----
Funktion : SetVisualizer()
    Setzt die Funktion um den externen Kartenviewer zu starten.

```

```

Parameter :
    Das AussaatHandle zu dem Aussaatmodul.

    Zeiger auf eine externe Funktion die zur Kartenbetrachtung dient.
    Die Visualisierungsfunktion erwartet einen String mit dem Dateinamen
    und liefert ein Boolean zurück:
    TVisualizerFunction = function(filename: PChar): Boolean;

```

```

Rückgabe :

```

```
    True bei erfolgreicher Werteübergabe
    False bei Fehler
}
function SetVisualizer(handle:TAussaathandle;
                      visualizerFunction: TVisualizerFunction): Boolean;
```

{{ -----  
Funktion : SetVorfrucht()  
Setzt die Vorfrucht

Parameter :  
Das AussaatHandle zu dem Aussaatmodul.

Wert vom Typ PChar, der die Bezeichnung der Vorfrucht enthält.  
Gültig sind nur Werte aus der Tabelle 'Kulturen.dbf'.

Rückgabe :  
True bei erfolgreicher Werteübergabe  
False, falls eine unbekannte Vorfrucht übergeben wird.

```
}
function SetVorfrucht(handle: TAussaathandle; Vorfrucht: PChar): Boolean;
```

## 4.1.9.3 Exemplarische Einbindung des Weizenaussaatmoduls in AGRO-MAP Basic

## 4.1.9.3.1 Kontroll-Klasse für den Wizard

Option **Explicit**

DefLng A-Z

```

*****
'*****
'
' * Name           :   clsWCPASaatWW
' * Creator        :   Martin Sperlich
' * Copyright      :   (c) agrocom.
' *
' * Description    :   Kontrollklasse für den Wizard zur Erstellung von Saatmassen-
' *                  :   karten nach preagro
' *
' * Dependences   :   XLib32.bas
' *
' * History
' * -----
' * Date          User          Description
' * ****          ****          ****
' *
'*****
*****

Private Const m_sMOD = "clsWCPASaatWW"

Implements IWizardControl

Private m_lField2StartWith As Long

Private m_lFieldId_BAK As Long      ' Muss den Wechsel eines Felds mitbekommen!

'*****
'***
' * Class
'*****
'***
Private Sub Class_Initialize()
    On Error GoTo ErrHandler
    otRegister Me

    m_lField2StartWith = -1

    Exit Sub

ErrHandler:
    If gnErrorHandler(m_sMOD, "Class_Initialize", Err, Erl) = vbRetry Then Resume
End Sub

Private Sub Class_Terminate()
    On Error GoTo ErrHandler

    otUnregister Me
    Exit Sub

ErrHandler:
    If gnErrorHandler(m_sMOD, "Class_Terminate", Err, Erl) = vbRetry Then Resume
End Sub

'*****
'***
' * Publics
'*****
'***
Friend Property Let Field2StartWith(ByVal lFieldId As Long)
    m_lField2StartWith = lFieldId
End Property

'*****
'***
' * IWizardControl
'*****
'***
Private Sub IWizardControl_GetOptions(ewoRetOptions As WizardOptions)

```

```

    On Error GoTo ErrHandler

    ewoRetOptions = ewoRetOptions Or woTabs Or woFinishIsNext

    Exit Sub

ErrHandler:
    If gnErrorHandler(m_sMOD, "IWizardControl_GetOptions", Err, Erl) = vbRetry Then Re-
sume
End Sub

Private Sub IWizardControl_GetCaption(sRetCaption As String)
    On Error GoTo ErrHandler

    ' sRetCaption = Replace$(" %COMPANYNAME% Aussaatplanung für Winterweizen", _
    ' "%COMPANYNAME%", gsPREAGRO)
    sRetCaption = "Teilflächenspezifische Aussaat (Winterweizen)"

    Exit Sub

ErrHandler:
    If gnErrorHandler(m_sMOD, "IWizardControl_GetCaption", Err, Erl) = vbRetry Then Re-
sume
End Sub

Private Sub IWizardControl_GetPageExtend(lRetWidth As Long, lRetHeight As Long)
    On Error GoTo ErrHandler

    lRetWidth = 6000
    lRetHeight = 4200

    Exit Sub

ErrHandler:
    If gnErrorHandler(m_sMOD, "IWizardControl_GetPageExtend", Err, Erl) = vbRetry Then
Resume
End Sub

Private Sub IWizardControl_GetDlgPlacement(lRetLeft As Long, lRetTop As Long, lRet-
Width As Long, lRetHeight As Long)
End Sub

Private Sub IWizardControl_OnDlgUnload(ByVal nLeft As Long, ByVal nTop As Long, ByVal
nWidth As Long, ByVal nHeight As Long)
End Sub

Public Sub IWizardControl_RequestNextPage(sCurrentPageKey As String, cvc As clsVari-
antCollection, _
    sRetPageKey As String, sRetCtlName As String, sRetCaption As String, _
    bRetIsLastPage As Boolean, lRetHelpContextId As Long)
    On Error GoTo ErrHandler

    Select Case sCurrentPageKey
        Case "" ' Die erste Seite
            If m_lField2StartWith < 0 Then
                GoSub DefSelectField
            Else ' Schlag ist bereits gesetzt - muss
                ' aber noch auf den Var-Array übertragen werden!
                m_SelectFieldPresetVariables cvc
                GoSub DefSelectMaps
            End If

        Case "[SELECTFIELD]" ' Die Folgeseite zu SelectField...
            GoSub DefSelectMaps

        Case Else
            Debug.Assert False
    End Select

    Exit Sub

DefSelectField:
    sRetPageKey = "[SELECTFIELD]"
    'x sRetCtlName = "wzpSelectField" 'VBLM SKIP
    sRetCtlName = wzpNames.wzpSelectField
    sRetCaption = "Schlag"
    ' lRetHelpContextId = GetHelpContextID( 0

```

```

If Not cvc.Defined("lFieldId") Then
    m_SelectFieldResetVariables cvc
End If
Return

DefSelectMaps:
    sRetPageKey = "[SELECTMAPS]"
    'x sRetCtlName = "wzpSelectMaps" 'VBLM SKIP
    sRetCtlName = wzpNames.wzpSelectMaps
    sRetCaption = "Meßwertkarte"
    ' lRetHelpContextId = GetHelpContextID( 0
If Not cvc.Defined("lMapCount") Then
        m_SelectMapsResetVariables cvc
End If
    bRetIsLastPage = True
Return

ErrorHandler:
If gnErrorHandler(m_sMOD, "RequestNextPage", Err, Erl) = vbRetry Then Resume
End Sub

Private Sub IWizardControl_BeforeActivate(sPageKey As String, ByVal lHitCount As Long,
cvc As clsVariantCollection, ctrl As Control)
    On Error GoTo ErrorHandler
    Dim ctrlSF As wzpSelectField
    Dim ctrlSM As wzpSelectMaps

    Select Case sPageKey
        Case "[SELECTFIELD]"
            Set ctrlSF = ctrl
            With ctrlSF
                ' Variable
                .VariableEx(0) = "lFieldId" 'VBLM SKIP
                .VariableEx(1) = "sXFieldName" 'VBLM SKIP
                ' Beschreibungstext
                .Description = "Bitte wählen Sie den Schlag aus, zu dem Sie eine Saatmas-
senkarte erstellen möchten."
            End With

            Case "[SELECTMAPS]"
                Set ctrlSM = ctrl
                With ctrlSM
                    .Description = "Wählen Sie bitte hier eine Ertragskarte aus, wenn das Er-
tragspotential für die Saatmassenkarte auf Basis einer Ertragskarte berechnet werden
soll."
                    .CheckBoxes = True
                    .MultiSelect = False
                    .SelectionOptional = True ' Man muss nix selektieren!
                End With

            Case "[FINISH]"
                m_FinishBeforeActivate cvc, ctrl.GetIWizardPage()

            Case Else
                Debug.Assert False
    End Select

Exit Sub

ErrorHandler:
If gnErrorHandler(m_sMOD, "IWizardControl_BeforeActivate", Err, Erl) = vbRetry Then
Resume
End Sub

Private Function IWizardControl_AfterValidateNStore(sPageKey As String, cvcVariables
As clsVariantCollection, ctrl As Control) As Boolean
    On Error GoTo ErrorHandler
    Dim bReturn As Boolean

    If sPageKey = "[SELECTFIELD]" Then ' Schlagauswahl
        bReturn = m_bSelectFieldAfterValidateNStore(cvcVariables)

    ElseIf sPageKey = "[HYDROSELECTMAP]" Then ' Auswahl Meßwertkarte
        bReturn = m_bSelectMapsAfterValidateNStore(cvcVariables)

    Else
        bReturn = True

```

```

    End If

    IWizardControl_AfterValidateNStore = bReturn

    Exit Function

ErrorHandler:
    If gnErrorHandler(m_sMOD, "IWizardControl_AfterValidateNStore", Err, Erl) = vbRetry
    Then Resume
    End Function

Private Function IWizardControl_Finish(cvc As clsVariantCollection, ctrl As Control)
As Boolean
    On Error GoTo ErrorHandler
    Dim nFieldId As Long
    Dim nMapId As Long

    nFieldId = cvc.Value("lFieldId")
    If cvc.Value("lMapCount") <= 0 Then
        nMapId = -1
    Else
        nMapId = cvc.Value("lMapId0")
    End If

    '*** Return
    IWizardControl_Finish = PASaatWWStoreFieldNMap(nFieldId, nMapId)

    Exit Function

ErrorHandler:
    If gnErrorHandler(m_sMOD, "IWizardControl_Finish", Err, Erl) = vbRetry Then Resume
    End Function

Private Function IWizardControl_Cancel(cvcVariables As clsVariantCollection, ctrl As
Control) As Boolean
    On Error GoTo ErrorHandler

    ' Aufräumen...

    '*** Return
    IWizardControl_Cancel = True

    Exit Function

ErrorHandler:
    If gnErrorHandler(m_sMOD, "IWizardControl_Cancel", Err, Erl) = vbRetry Then Resume
    End Function

Private Function IWizardControl_PageNotify(cvcVariables As clsVariantCollection, ctrl
As Control, sMsgKey As String, vWParam As Variant, vLParam As Variant) As Long
    On Error GoTo ErrorHandler

    ' Select Case sMsgKey
    '     * Notifies von wzyEMapSelectBorder
    '     Case "[BORDER_CHECK]"
    '         m_CheckStage
    '         m_LoadBorder cvcVariables, vWParam
    '
    '     Case "[BORDER_UNCHECK]"
    '         m_UnloadBorder cvcVariables, vWParam
    '
    '     * Notifies von wszpMapSelectMaps
    '     Case "[MAP_CHECK]"
    '         m_LoadMap cvcVariables, vWParam
    '
    '     Case "[MAP_UNCHECK]"
    '         m_UnloadMap cvcVariables, vWParam
    '
    ' End Select

    Exit Function

ErrorHandler:
    If gnErrorHandler(m_sMOD, "IWizardControl_PageNotify", Err, Erl) = vbRetry Then Re-
sume
    End Function

```

```

'*****
***
'* ResetVariables
'*****
***
Private Sub m_SelectFieldPresetVariables(cvc As clsVariantCollection)
    On Error GoTo ErrorHandler
    Dim s As String
    ' Wird aufgerufen, wenn bereits mit einem Schlag gestartet wird und
    ' die Seite Schlagauswahl gar nicht angezeigt wird. Trotzdem muss
    ' ich natürlich die Variablen füllen..

    With cvc
        .AssignEx "lFieldId", vbLong, m_lField2StartWith

        gbDbNamesFromFieldId m_lField2StartWith, s
        .AssignEx "sXFieldName", vbString, s
    End With

    Exit Sub

ErrorHandler:
    If gnErrorHandler(m_sMOD, "m_SelectFieldPresetVariables", Err, Erl) = vbRetry Then
Resume
End Sub

Private Sub m_SelectFieldResetVariables(cvc As clsVariantCollection)
    On Error GoTo ErrorHandler

    With cvc
        .AssignEx "lFieldId", vbLong, -1&
        .AssignEx "sXFieldName", vbString, ""
    End With

    Exit Sub

ErrorHandler:
    If gnErrorHandler(m_sMOD, "m_SelectFieldResetVariables", Err, Erl) = vbRetry Then
Resume
End Sub

Private Sub m_SelectMapsResetVariables(cvc As clsVariantCollection)
    On Error GoTo ErrorHandler

    With cvc
        .AssignEx "lMapCount", vbLong, 0&
    End With

    Exit Sub

ErrorHandler:
    If gnErrorHandler(m_sMOD, "m_SelectMapsResetVariables", Err, Erl) = vbRetry Then
Resume
End Sub

'*****
***
'* BeforeActivate
'*****
***
Private Sub m_FinishBeforeActivate(cvc As clsVariantCollection, ctrlF As wzpFinish)
    On Error GoTo ErrorHandler

    Dim s As String
    Dim sIndent As String
    Dim sRtf As String

    With ctrlF
        s = "Drücken Sie '%FINISH%' um den Vorgang abzuschließen. Drücken Sie
'%PREVIOUS%' um Ihre Einstellungen zu kontrollieren bzw. sie zu ändern."
        .Description = Replace(Replace(s, _
            "%FINISH%", cvc.Value("_sCmdFinish_")), _
            "%PREVIOUS%", cvc.Value("_sCmdPrevious_")) 'VBLM SKIP
        .Text = cvc.Dump("", "sDataRow*")
        sIndent = Space$(6)
        sRtf = gsRtfDocHeader(locCharset) & gsRtfStd() & "\tx2200 "
    End With

```

```

        ' Datei
        s = "Schlag" & ":"
        sRtf = sRtf & "\b " & s & "\b-\tab " & _
            cvc.Value("sXFieldName") & gsRtfPar()
        ' Meßwertkarte
        s = "Meßwertkarte" & ":"
        sRtf = sRtf & "\b " & s & "\b-\tab " & _
            cvc.Value("sXMapName0") & gsRtfPar()

        sRtf = sRtf & gsRtfDocFooter()
        .TextRTF = sRtf
    End With

    Exit Sub

ErrorHandler:
    If gnErrorHandler(m_sMOD, "m_FinishBeforeActivate", Err, Erl) = vbRetry Then Resume
End Sub

'*****
'***
' * AfterValidateNStore
'*****
Private Function m_bSelectFieldAfterValidateNStore(cvc As clsVariantCollection) As Boolean
    On Error GoTo ErrorHandler

    ' Wenn das Feld gewechselt wurde, muss ich einiges auf den Folge-
    ' seiten zurücksetzen.
    If cvc.Value("lFieldId") <> m_lFieldId_BAK Then
        ' Ertragskarte weg
        m_SelectMapsResetVariables cvc

        m_lFieldId_BAK = cvc.Value("lFieldId")
    End If

    ' * Keine Nachbehandlung
    m_bSelectFieldAfterValidateNStore = True

    Exit Function

ErrorHandler:
    If gnErrorHandler(m_sMOD, "m_bSelectFieldAfterValidateNStore", Err, Erl) = vbRetry
    Then Resume
End Function

Private Function m_bSelectMapsAfterValidateNStore(cvc As clsVariantCollection) As Boolean
    On Error GoTo ErrorHandler

    '*** Return
    m_bSelectMapsAfterValidateNStore = True

    Exit Function

ErrorHandler:
    If gnErrorHandler(m_sMOD, "m_bSelectMapsAfterValidateNStore", Err, Erl) = vbRetry
    Then Resume
End Function

```

#### 4.1.9.3.2 Code des preagro-DLL-Interfaces

```
Option Explicit
```

```
DefLng A-Z
```

```

'*****
' *
' * Name : clsPASaatWW
' * Creator : Martin Sperlich
' * Copyright : (c) agrocom.
' *
' * Description:
' *
' * Dependences: XLib2.bas
' *
' * History
' * -----

```

```

'* Date      User  Description
'* ****      ****  *****
'*
'*****

Private Const m_sMOD = "clsPASaatWW"

Private Declare Function AussaatInit Lib "PASaatWW.dll" (ByVal pszTitel As String) As Long
Private Declare Function AussaatRun Lib "PASaatWW.dll" (ByVal handle As Long, ByVal pszDateinameErtragspotenzial As String, ByVal pszDateinameSchaetzertrag As String, ByVal pszDateinameSaatmasse As String) As Byte
Private Declare Function AussaatExit Lib "PASaatWW.dll" (ByVal handle As Long) As Byte
Private Declare Function GetAckerzahl Lib "PASaatWW.dll" (ByVal handle As Long, pdblAckerzahl As Double) As Byte
Private Declare Function GetAckerzahlKarte Lib "PASaatWW.dll" (ByVal handle As Long, ByVal pszDateiname As String) As Byte
Private Declare Function GetBenutzeAckerzahl Lib "PASaatWW.dll" (ByVal handle As Long) As Byte
Private Declare Function GetBenutzeAckerzahlEinheitlich Lib "PASaatWW.dll" (ByVal handle As Long) As Byte
Private Declare Function GetBenutzeAckerzahlKarte Lib "PASaatWW.dll" (ByVal handle As Long) As Byte
Private Declare Function GetBenutzeBodenSubstratEinheitlich Lib "PASaatWW.dll" (ByVal handle As Long) As Byte
Private Declare Function GetBenutzeBodenSubstratKarte Lib "PASaatWW.dll" (ByVal handle As Long) As Byte
Private Declare Function GetBenutzeErtragsPotenzialKarte Lib "PASaatWW.dll" (ByVal handle As Long) As Byte
Private Declare Function GetBodenfeuchteBeiSaat Lib "PASaatWW.dll" (ByVal handle As Long, pnBodenfeuchteBeiSaat As Long) As Byte
Private Declare Function GetBodenSaatgutKontakt Lib "PASaatWW.dll" (ByVal handle As Long, pnBodenSaatgutKontakt As Long) As Byte
Private Declare Function GetBodenSubstrat Lib "PASaatWW.dll" (ByVal handle As Long, ByVal pszBodensubstrat As String) As Byte
Private Declare Function GetBodenSubstratKarte Lib "PASaatWW.dll" (ByVal handle As Long, ByVal pszDateiname As String) As Byte
Private Declare Function GetBundesland Lib "PASaatWW.dll" (ByVal handle As Long, ByVal pszBundesland As String) As Byte
Private Declare Function GetErtragsPotenzialKarte Lib "PASaatWW.dll" (ByVal handle As Long, ByVal pszDateiname As String) As Byte
Private Declare Function GetKeimfaehigkeit Lib "PASaatWW.dll" (ByVal handle As Long, pdblKeimfaehigkeit As Double) As Byte
Private Declare Function GetKoordinatenDll Lib "PASaatWW.dll" Alias "GetKoordinaten" (ByVal handle As Long, pdblXmin As Double, pdblXmax As Double, pdblYmin As Double, pdblYmax As Double) As Byte
Private Declare Function GetKorrekturErtragserwartung Lib "PASaatWW.dll" (ByVal handle As Long, pnKorrekturErtragserwartung As Long) As Byte
Private Declare Function GetNiederschlag Lib "PASaatWW.dll" (ByVal handle As Long, pnNiederschlag As Long) As Byte
Private Declare Function GetRasterbreite Lib "PASaatWW.dll" (ByVal handle As Long, pnRasterbreite As Long) As Byte
Private Declare Function GetRealerSaatTermin Lib "PASaatWW.dll" (ByVal handle As Long, pnRealerSaatTermin As Long) As Byte
Private Declare Function GetRegion Lib "PASaatWW.dll" (ByVal handle As Long, ByVal pszRegion As String) As Byte
Private Declare Function GetReliefEinflussKarte Lib "PASaatWW.dll" (ByVal handle As Long, ByVal pszDateiname As String) As Byte
Private Declare Function GetSaatBettQualitaet Lib "PASaatWW.dll" (ByVal handle As Long, pnSaatBettQualitaet As Long) As Byte
Private Declare Function GetSaatTiefe Lib "PASaatWW.dll" (ByVal handle As Long, pnSaatTiefe As Long) As Byte
Private Declare Function GetSorte Lib "PASaatWW.dll" (ByVal handle As Long, ByVal pszSorte As String) As Byte
Private Declare Function GetSpalteAckerzahl Lib "PASaatWW.dll" (ByVal handle As Long, ByVal pszSpalte As String) As Byte
Private Declare Function GetSpalteBodensubstratKarte Lib "PASaatWW.dll" (ByVal handle As Long, ByVal pszSpalte As String) As Byte
Private Declare Function GetSpalteReliefEinflussKarte Lib "PASaatWW.dll" (ByVal handle As Long, ByVal pszSpalteReliefEinflussKarte As String) As Byte
Private Declare Function GetTKG Lib "PASaatWW.dll" (ByVal handle As Long, pdblTKG As Double) As Byte
Private Declare Function GetVorfrucht Lib "PASaatWW.dll" (ByVal handle As Long, ByVal pszVorfrucht As String) As Byte
Private Declare Function SetAckerzahl Lib "PASaatWW.dll" (ByVal handle As Long, ByVal dblAckerzahl As Double) As Byte

```

```

Private Declare Function SetAckerzahlKarte Lib "PASaatWW.dll" (ByVal handle As Long,
ByVal pszDateiname As String) As Byte
Private Declare Function SetBodenfeuchteBeiSaat Lib "PASaatWW.dll" (ByVal handle As
Long, ByVal nBodenfeuchteBeiSaat As Long) As Byte
Private Declare Function SetBodenSaatgutKontakt Lib "PASaatWW.dll" (ByVal handle As
Long, ByVal nBodenSaatgutKontakt As Long) As Byte
Private Declare Function SetBodenSubstrat Lib "PASaatWW.dll" (ByVal handle As Long,
ByVal pszBodenSubstrat As String) As Byte
Private Declare Function SetBodenSubstratKarte Lib "PASaatWW.dll" (ByVal handle As
Long, ByVal pszDateiname As String) As Byte
Private Declare Function SetBundesland Lib "PASaatWW.dll" (ByVal handle As Long, ByVal
pszBundesland As String) As Byte
Private Declare Function SetErtragsPotenzialKarte Lib "PASaatWW.dll" (ByVal handle As
Long, ByVal pszDateiname As String) As Byte
Private Declare Function SetKeimfaehigkeit Lib "PASaatWW.dll" (ByVal handle As Long,
ByVal dblKeimfaehigkeit As Double) As Byte
Private Declare Function SetKoordinatenDll Lib "PASaatWW.dll" Alias "SetKoordinaten"
(ByVal handle As Long, ByVal xmin As Double, ByVal xmax As Double, ByVal ymin As Dou-
ble, ByVal ymax As Double) As Byte
Private Declare Function SetKorrekturErtragserwartung Lib "PASaatWW.dll" (ByVal handle
As Long, ByVal nKorrekturErtragserwartung As Long) As Byte
Private Declare Function SetNiederschlag Lib "PASaatWW.dll" (ByVal handle As Long,
ByVal nNiederschlag As Long) As Byte
Private Declare Function SetRasterbreite Lib "PASaatWW.dll" (ByVal handle As Long,
ByVal nRasterbreite As Long) As Byte
Private Declare Function SetRealerSaatTermin Lib "PASaatWW.dll" (ByVal handle As Long,
ByVal nRealerSaatTermin As Long) As Byte
Private Declare Function SetRegion Lib "PASaatWW.dll" (ByVal handle As Long, ByVal
pszRegion As String) As Byte
Private Declare Function SetReliefEinflussKarte Lib "PASaatWW.dll" (ByVal handle As
Long, ByVal pszDateiname As String) As Byte
Private Declare Function SetSaatBettQualitaet Lib "PASaatWW.dll" (ByVal handle As
Long, ByVal nSaatBettQualitaet As Long) As Byte
Private Declare Function SetSaatTiefe Lib "PASaatWW.dll" (ByVal handle As Long, ByVal
nSaatTiefe As Long) As Byte
Private Declare Function SetSorte Lib "PASaatWW.dll" (ByVal handle As Long, ByVal
pszSorte As String) As Byte
Private Declare Function SetSpalteAckerzahl Lib "PASaatWW.dll" (ByVal handle As Long,
ByVal pszSpalte As String) As Byte
Private Declare Function SetSpalteBodenSubstratKarte Lib "PASaatWW.dll" (ByVal handle
As Long, ByVal pszSpalte As String) As Byte
Private Declare Function SetSpalteReliefEinflussKarte Lib "PASaatWW.dll" (ByVal handle
As Long, ByVal pszSpalteReliefEinflussKarte As String) As Byte
Private Declare Function SetTKG Lib "PASaatWW.dll" (ByVal handle As Long, ByVal dblTKG
As Double) As Byte
Private Declare Function SetVisualizer Lib "PASaatWW.dll" (ByVal handle As Long, ByVal
pfnVisualizerFunction As Long) As Byte
Private Declare Function SetVorfrucht Lib "PASaatWW.dll" (ByVal handle As Long, ByVal
pszVorfrucht As String) As Byte
Private Declare Sub SetBenutzeAckerzahl Lib "PASaatWW.dll" (ByVal handle As Long)
Private Declare Sub SetBenutzeAckerzahlEinheitlich Lib "PASaatWW.dll" (ByVal handle As
Long)
Private Declare Sub SetBenutzeAckerzahlKarte Lib "PASaatWW.dll" (ByVal handle As Long)
Private Declare Sub SetBenutzeBodenSubstratEinheitlich Lib "PASaatWW.dll" (ByVal han-
dle As Long)
Private Declare Sub SetBenutzeBodenSubstratKarte Lib "PASaatWW.dll" (ByVal handle As
Long)
Private Declare Sub SetBenutzeErtragsPotenzialKarte Lib "PASaatWW.dll" (ByVal handle
As Long)

' Private Declare Function GetModuleFileName Lib "kernel32" Alias "GetModuleFileNameA"
(ByVal hModule As Long, ByVal lpFileName As String, ByVal nSize As Long) As Long

Private m_handle As Long
Private m_hmod As Long

#Const STRINGGET = True

'*****
' * Class
'*****
Private Sub Class_Initialize()
    On Error GoTo ErrHandler
    ' otRegister Me

Exit Sub

```

```

ErrorHandler:
  If gnErrorHandler(m_sMOD, "Class_Initialize", Err, Erl) = vbRetry Then Resume
End Sub

Private Sub Class_Terminate()
  On Error GoTo ErrorHandler

  Done

  ' otUnregister Me
Exit Sub

ErrorHandler:
  If gnErrorHandler(m_sMOD, "Class_Terminate", Err, Erl) = vbRetry Then Resume
End Sub

'*****
' * Friends
'*****
Friend Function Init(Optional ByVal sSubCaption As String, Optional ByVal sLibraryPath
As String) As Boolean
  On Error GoTo ErrorHandler
  ' Dim n As Long
  Dim s As String

  If Len(sLibraryPath) > 0 Then
    s = gsFSlash(sLibraryPath) & "PASaatWW.dll" 'VBLM SKIP
    If gbFExists(s) Then
      m_hmod = LoadLibrary(s)
      If m_hmod = 0 Then
        Err.Raise 7367, , "LoadLibrary of '" & s & "' failed" & Chr$(32) & _
          "(" & Err.LastDllError & ")"
      Else
        ' s = Space$(256)
        ' n = GetModuleFileName(hmod, s, 256)
        ' If n > 0 Then
        ' s = Left$(s, n)
        ' MsgBox s, vbInformation
        ' End If
      End If
    End If
  End If

  m_handle = AussaatInit(sSubCaption)

  If m_handle <> 0 Then
    '*** Return
    Init = True
  End If

Exit Function

ErrorHandler:
  If gnErrorHandler(m_sMOD, "Init", Err, Erl) = vbRetry Then Resume
End Function

Friend Sub Done()
  On Error GoTo ErrorHandler

  If m_handle <> 0 Then
    AussaatExit m_handle
    m_handle = 0
  End If

  If m_hmod <> 0 Then
    FreeLibrary m_hmod
    m_hmod = 0
  End If

Exit Sub

ErrorHandler:
  If gnErrorHandler(m_sMOD, "Done", Err, Erl) = vbRetry Then Resume
End Sub

Friend Function Run(sFMwkErtragsPotential As String, sFMwkSchaetzErtrag As String,
sFMwkSaatMasse As String) As Boolean

```

```

If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
On Error GoTo ErrHandler
Dim n As Long

n = AussaatRun(m_handle, sFMwkErtragsPotential, sFMwkSchaetzErtrag, sFMwkSaatMasse)
'*** Return
Run = (n <> 0)

Exit Function

ErrHandler:
If gnErrHandler(m_sMOD, "Run", Err, Erl) = vbRetry Then Resume
End Function

Friend Property Get Ackerzahl() As Double
If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
On Error GoTo ErrHandler
Dim dbl As Double

If GetAckerzahl(m_handle, dbl) <> 0 Then
'*** Return
Ackerzahl = dbl
Else
On Error GoTo 0
Err.Raise 7367, , "GetAckerzahl failed."
End If

Exit Property

ErrHandler:
If gnErrHandler(m_sMOD, "Ackerzahl", Err, Erl) = vbRetry Then Resume
End Property

Friend Property Let Ackerzahl(ByVal dblAckerzahl As Double)
If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
On Error GoTo ErrHandler

If SetAckerzahl(m_handle, dblAckerzahl) = 0 Then
On Error GoTo 0
Err.Raise 7367, , "SetAckerzahl failed."
End If

Exit Property

ErrHandler:
If gnErrHandler(m_sMOD, "Ackerzahl", Err, Erl) = vbRetry Then Resume
End Property

#If STRINGGET Then
Friend Property Get AckerzahlKarte() As String
If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
On Error GoTo ErrHandler
Dim n As Long
Dim s As String

s = Space$(512) ' Ist 2 x MAX_PATH (256)
If GetAckerzahlKarte(m_handle, s) <> 0 Then
n = InStr(1, s, Chr$(0))
If n > 0 Then s = Left$(s, n - 1)
'*** Return
AckerzahlKarte = s
Else
On Error GoTo 0
Err.Raise 7367, , "GetAckerzahlKarte failed."
End If

Exit Property

ErrHandler:
If gnErrHandler(m_sMOD, "AckerzahlKarte", Err, Erl) = vbRetry Then Resume
End Property
#End If

Friend Property Let AckerzahlKarte(ByVal sDateiNamen As String)
If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
On Error GoTo ErrHandler

```

```

If SetAckerzahlKarte(m_handle, sDateiNamen) = 0 Then
  On Error GoTo 0
  Err.Raise 7367, , "SetAckerzahlKarte failed."
End If

Exit Property

ErrorHandler:
  If gnErrorHandler(m_sMOD, "AckerzahlKarte", Err, Erl) = vbRetry Then Resume
End Property

Friend Property Get BenutzeAckerzahl() As Boolean
  If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
  *** Return
  BenutzeAckerzahl = (GetBenutzeAckerzahl(m_handle) <> 0)
End Property

Friend Property Let BenutzeAckerzahl(ByVal bBenutzeAckerzahl As Boolean)
  ' Man kann BenutzeAckerzahl nur auf True setzen. Auf False bekommt
  ' man es indem man BenutzeBodenSubstratEinheitlich oder Benutze-
  ' BodenSubstratKarte setzt!
  If Not bBenutzeAckerzahl Then Err.Raise 7367, , "You can only assign 'True' to 'Be-
  nutzeAckerzahl'."
  If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
  SetBenutzeAckerzahl m_handle
End Property

Friend Property Get BenutzeAckerzahlEinheitlich() As Boolean
  If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
  *** Return
  BenutzeAckerzahlEinheitlich = (GetBenutzeAckerzahlEinheitlich(m_handle) <> 0)
End Property

Friend Property Let BenutzeAckerzahlEinheitlich(ByVal bBenutzeAckerzahlEinheitlich As
Boolean)
  If Not bBenutzeAckerzahlEinheitlich Then Err.Raise 7367, , "You can only assign
'True' to 'BenutzeAckerzahlEinheitlich'."
  If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
  SetBenutzeAckerzahlEinheitlich m_handle
End Property

Friend Property Get BenutzeAckerzahlKarte() As Boolean
  If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
  *** Return
  BenutzeAckerzahlKarte = (GetBenutzeAckerzahlKarte(m_handle) <> 0)
End Property

Friend Property Let BenutzeAckerzahlKarte(ByVal bBenutzeAckerzahlKarte As Boolean)
  If Not bBenutzeAckerzahlKarte Then Err.Raise 7367, , "You can only assign 'True' to
'BenutzeAckerzahlKarte'."
  If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
  SetBenutzeAckerzahlKarte m_handle
End Property

Friend Property Get BenutzeBodenSubstratEinheitlich() As Boolean
  If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
  *** Return
  BenutzeBodenSubstratEinheitlich = (GetBenutzeBodenSubstratEinheitlich(m_handle) <>
0)
End Property

Friend Property Let BenutzeBodenSubstratEinheitlich(ByVal bBenutzeBodenSubstratEin-
heitlich As Boolean)
  If Not bBenutzeBodenSubstratEinheitlich Then Err.Raise 7367, , "You can only assign
'True' to 'BenutzeBodenSubstratEinheitlich'."
  If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
  SetBenutzeBodenSubstratEinheitlich m_handle
End Property

Friend Property Get BenutzeBodenSubstratKarte() As Boolean
  If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
  *** Return
  BenutzeBodenSubstratKarte = (GetBenutzeBodenSubstratKarte(m_handle) <> 0)
End Property

Friend Property Let BenutzeBodenSubstratKarte(ByVal bBenutzeBodenSubstratKarte As Boo-
lean)

```

```

    If Not bBenutzeBodenSubstratKarte Then Err.Raise 7367, , "You can only assign
'True' to 'BenutzeBodenSubstratKarte'."
    If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
    SetBenutzeBodenSubstratKarte m_handle
End Property

Friend Property Get BenutzeErtragsPotenzialKarte() As Boolean
    If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
    '*** Return
    BenutzeErtragsPotenzialKarte = (GetBenutzeErtragsPotenzialKarte(m_handle) <> 0)
End Property

Friend Property Let BenutzeErtragsPotenzialKarte(ByVal bBenutzeErtragsPotenzialKarte
As Boolean)
    If Not bBenutzeErtragsPotenzialKarte Then Err.Raise 7367, , "You can only assign
'True' to 'BenutzeErtragsPotenzialKarte'."
    If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
    SetBenutzeErtragsPotenzialKarte m_handle
End Property

Friend Property Get BodenfeuchteBeiSaat() As Long
    If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
    On Error GoTo ErrHandler
    Dim n As Long

    If GetBodenfeuchteBeiSaat(m_handle, n) <> 0 Then
        '*** Return
        BodenfeuchteBeiSaat = n
    Else
        On Error GoTo 0
        Err.Raise 7367, , "GetBodenfeuchteBeiSaat failed."
    End If

    Exit Property

ErrHandler:
    If gnErrHandler(m_sMOD, "BodenfeuchteBeiSaat", Err, Erl) = vbRetry Then Resume
End Property

Friend Property Let BodenfeuchteBeiSaat(ByVal nBodenfeuchteBeiSaat As Long)
    If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
    On Error GoTo ErrHandler

    If SetBodenfeuchteBeiSaat(m_handle, nBodenfeuchteBeiSaat) = 0 Then
        On Error GoTo 0
        Err.Raise 7367, , "SetBodenfeuchteBeiSaat failed."
    End If

    Exit Property

ErrHandler:
    If gnErrHandler(m_sMOD, "BodenfeuchteBeiSaat", Err, Erl) = vbRetry Then Resume
End Property

Friend Property Get BodenSaatgutKontakt() As Long
    If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
    On Error GoTo ErrHandler
    Dim n As Long

    If GetBodenSaatgutKontakt(m_handle, n) <> 0 Then
        '*** Return
        BodenSaatgutKontakt = n
    Else
        On Error GoTo 0
        Err.Raise 7367, , "BodenSaatgutKontakt failed."
    End If

    Exit Property

ErrHandler:
    If gnErrHandler(m_sMOD, "BodenSaatgutKontakt", Err, Erl) = vbRetry Then Resume
End Property

Friend Property Let BodenSaatgutKontakt(ByVal nBodenSaatgutKontakt As Long)
    If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
    On Error GoTo ErrHandler

```

```

If SetBodenSaatgutKontakt(m_handle, nBodenSaatgutKontakt) = 0 Then
  On Error GoTo 0
  Err.Raise 7367, , "SetBodenSaatgutKontakt failed."
End If

Exit Property

ErrorHandler:
  If gnErrorHandler(m_sMOD, "BodenSaatgutKontakt", Err, Erl) = vbRetry Then Resume
End Property

#If STRINGGET Then
Friend Property Get BodenSubstrat() As String
  If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
  On Error GoTo ErrorHandler
  Dim n As Long
  Dim s As String

  s = Space$(256)
  If GetBodenSubstrat(m_handle, s) <> 0 Then
    n = InStr(1, s, Chr$(0))
    If n > 0 Then s = Left$(s, n - 1)
    '*** Return
    BodenSubstrat = s
  Else
    On Error GoTo 0
    Err.Raise 7367, , "GetBodenSubstrat failed."
  End If

  Exit Property

ErrorHandler:
  If gnErrorHandler(m_sMOD, "BodenSubstrat", Err, Erl) = vbRetry Then Resume
End Property
#End If

Friend Property Let BodenSubstrat(ByVal sBodenSubstrat As String)
  If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
  On Error GoTo ErrorHandler

  If SetBodenSubstrat(m_handle, sBodenSubstrat) = 0 Then
    On Error GoTo 0
    Err.Raise 7367, , "SetBodenSubstrat failed."
  End If

  Exit Property

ErrorHandler:
  If gnErrorHandler(m_sMOD, "BodenSubstrat", Err, Erl) = vbRetry Then Resume
End Property

#If STRINGGET Then
Friend Property Get BodenSubstratKarte() As String
  If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
  On Error GoTo ErrorHandler
  Dim n As Long
  Dim s As String

  s = Space$(512)
  n = GetBodenSubstratKarte(m_handle, s)
  '+ If n <> 0 Then
    n = InStr(1, s, Chr$(0))
    If n > 0 Then s = Left$(s, n - 1)
    '*** Return
    BodenSubstratKarte = s
  '+ Else
  '+ On Error GoTo 0
  '+ Err.Raise 7367, , "GetBodenSubstratKarte failed."
  '+ End If

  Exit Property

ErrorHandler:
  If gnErrorHandler(m_sMOD, "BodenSubstratKarte", Err, Erl) = vbRetry Then Resume
End Property
#End If

```

```

Friend Property Let BodenSubstratKarte(ByVal sBodenSubstratKarte As String)
  If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
  On Error GoTo ErrHandler

  If SetBodenSubstratKarte(m_handle, sBodenSubstratKarte) = 0 Then
    On Error GoTo 0
    Err.Raise 7367, , "SetBodenSubstratKarte failed."
  End If

  Exit Property

ErrHandler:
  If gnErrorHandler(m_sMOD, "BodenSubstratKarte", Err, Erl) = vbRetry Then Resume
End Property

#If STRINGGET Then
Friend Property Get Bundesland() As String
  If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
  On Error GoTo ErrHandler
  Dim n As Long
  Dim s As String

  s = Space$(256)
  If GetBundesland(m_handle, s) <> 0 Then
    n = InStr(1, s, Chr$(0))
    If n > 0 Then s = Left$(s, n - 1)
    *** Return
    Bundesland = s
  Else
    On Error GoTo 0
    Err.Raise 7367, , "GetBundesland failed."
  End If

  Exit Property

ErrHandler:
  If gnErrorHandler(m_sMOD, "Bundesland", Err, Erl) = vbRetry Then Resume
End Property
#End If

Friend Property Let Bundesland(ByVal sBundesland As String)
  If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
  On Error GoTo ErrHandler

  If SetBundesland(m_handle, sBundesland) = 0 Then
    On Error GoTo 0
    Err.Raise 7367, , "SetBundesland failed."
  End If

  Exit Property

ErrHandler:
  If gnErrorHandler(m_sMOD, "Bundesland", Err, Erl) = vbRetry Then Resume
End Property

#If STRINGGET Then
Friend Property Get ErtragsPotenzialKarte() As String
  If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
  On Error GoTo ErrHandler
  Dim n As Long
  Dim s As String

  s = Space$(512)
  If GetErtragsPotenzialKarte(m_handle, s) <> 0 Then
    n = InStr(1, s, Chr$(0))
    If n > 0 Then s = Left$(s, n - 1)
    *** Return
    ErtragsPotenzialKarte = s
  Else
    On Error GoTo 0
    Err.Raise 7367, , "GetErtragsPotenzialKarte failed."
  End If

  Exit Property

ErrHandler:
  If gnErrorHandler(m_sMOD, "ErtragsPotenzialKarte", Err, Erl) = vbRetry Then Resume

```

```
End Property
#End If

Friend Property Let ErtragsPotenzialKarte(ByVal sDateiNamen As String)
    If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
    On Error GoTo ErrHandler

    If SetErtragsPotenzialKarte(m_handle, sDateiNamen) = 0 Then
        On Error GoTo 0
        Err.Raise 7367, , "SetErtragsPotenzialKarte failed."
    End If

    Exit Property

ErrorHandler:
    If gnErrorHandler(m_sMOD, "ErtragsPotenzialKarte", Err, Erl) = vbRetry Then Resume
End Property

Friend Property Get Keimfaehigkeit() As Double
    If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
    On Error GoTo ErrHandler
    Dim dbl As Double

    If GetKeimfaehigkeit(m_handle, dbl) <> 0 Then
        '*** Return
        Keimfaehigkeit = dbl
    Else
        On Error GoTo 0
        Err.Raise 7367, , "GetKeimfaehigkeit failed."
    End If

    Exit Property

ErrorHandler:
    If gnErrorHandler(m_sMOD, "Keimfaehigkeit", Err, Erl) = vbRetry Then Resume
End Property

Friend Property Let Keimfaehigkeit(ByVal dblKeimfaehigkeit As Double)
    If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
    On Error GoTo ErrHandler

    If SetKeimfaehigkeit(m_handle, dblKeimfaehigkeit) = 0 Then
        On Error GoTo 0
        Err.Raise 7367, , "SetKeimfaehigkeit failed."
    End If

    Exit Property

ErrorHandler:
    If gnErrorHandler(m_sMOD, "Keimfaehigkeit", Err, Erl) = vbRetry Then Resume
End Property

Friend Sub GetKoordinaten(dblRetXMin As Double, dblRetXMax As Double, dblRetYMin As
Double, dblRetYMax As Double)
    If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
    On Error GoTo ErrHandler

    If GetKoordinatenDll(m_handle, dblRetXMin, dblRetXMax, dblRetYMin, dblRetYMax) = 0
Then
        On Error GoTo 0
        Err.Raise 7367, , "GetKoordinatenDll failed."
    End If

    Exit Sub

ErrorHandler:
    If gnErrorHandler(m_sMOD, "GetKoordinaten", Err, Erl) = vbRetry Then Resume
End Sub

Friend Sub SetKoordinaten(ByVal dblXMin As Double, ByVal dblXMax As Double, ByVal
dblYMin As Double, ByVal dblYMax As Double)
    If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
    On Error GoTo ErrHandler

    If SetKoordinatenDll(m_handle, dblXMin, dblXMax, dblYMin, dblYMax) = 0 Then
        On Error GoTo 0
        Err.Raise 7367, , "SetKoordinatenDll failed."
    End If
```

```
End If

Exit Sub

ErrorHandler:
If gnErrorHandler(m_sMOD, "GetKoordinaten", Err, Erl) = vbRetry Then Resume
End Sub

Friend Property Get KorrekturErtragserwartung() As Long
If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
On Error GoTo ErrorHandler
Dim n As Long

If GetKorrekturErtragserwartung(m_handle, n) <> 0 Then
    *** Return
    KorrekturErtragserwartung = n
Else
    On Error GoTo 0
    Err.Raise 7367, , "GetKorrekturErtragserwartung failed."
End If

Exit Property

ErrorHandler:
If gnErrorHandler(m_sMOD, "KorrekturErtragserwartung", Err, Erl) = vbRetry Then Resume
End Property

Friend Property Let KorrekturErtragserwartung(ByVal nKorrekturErtragserwartung As Long)
If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
On Error GoTo ErrorHandler

If SetKorrekturErtragserwartung(m_handle, nKorrekturErtragserwartung) = 0 Then
    On Error GoTo 0
    Err.Raise 7367, , "SetKorrekturErtragserwartung failed."
End If

Exit Property

ErrorHandler:
If gnErrorHandler(m_sMOD, "KorrekturErtragserwartung", Err, Erl) = vbRetry Then Resume
End Property

Friend Property Get Niederschlag() As Long
If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
On Error GoTo ErrorHandler
Dim n As Long

If GetNiederschlag(m_handle, n) <> 0 Then
    *** Return
    Niederschlag = n
Else
    On Error GoTo 0
    Err.Raise 7367, , "GetNiederschlag failed."
End If

Exit Property

ErrorHandler:
If gnErrorHandler(m_sMOD, "Niederschlag", Err, Erl) = vbRetry Then Resume
End Property

Friend Property Let Niederschlag(ByVal nNiederschlag As Long)
If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
On Error GoTo ErrorHandler

If SetNiederschlag(m_handle, nNiederschlag) = 0 Then
    On Error GoTo 0
    Err.Raise 7367, , "SetNiederschlag failed."
End If

Exit Property

ErrorHandler:
If gnErrorHandler(m_sMOD, "Niederschlag", Err, Erl) = vbRetry Then Resume
```

**End Property**

```
Friend Property Get Rasterbreite() As Long
  If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
  On Error GoTo ErrHandler
  Dim n As Long

  If GetRasterbreite(m_handle, n) <> 0 Then
    *** Return
    Rasterbreite = n
  Else
    On Error GoTo 0
    Err.Raise 7367, , "GetRasterbreite failed."
  End If

Exit Property
```

```
ErrHandler:
  If gnErrHandler(m_sMOD, "Rasterbreite", Err, Erl) = vbRetry Then Resume
End Property
```

```
Friend Property Let Rasterbreite(ByVal nRasterbreite As Long)
  If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
  On Error GoTo ErrHandler

  If SetRasterbreite(m_handle, nRasterbreite) = 0 Then
    On Error GoTo 0
    Err.Raise 7367, , "SetRasterbreite failed."
  End If

Exit Property
```

```
ErrHandler:
  If gnErrHandler(m_sMOD, "Rasterbreite", Err, Erl) = vbRetry Then Resume
End Property
```

```
Friend Property Get RealerSaatTermin() As Long
  If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
  On Error GoTo ErrHandler
  Dim n As Long

  If GetRealerSaatTermin(m_handle, n) <> 0 Then
    *** Return
    RealerSaatTermin = n
  Else
    On Error GoTo 0
    Err.Raise 7367, , "GetRealerSaatTermin failed."
  End If

Exit Property
```

```
ErrHandler:
  If gnErrHandler(m_sMOD, "RealerSaatTermin", Err, Erl) = vbRetry Then Resume
End Property
```

```
Friend Property Let RealerSaatTermin(ByVal nRealerSaatTermin As Long)
  If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
  On Error GoTo ErrHandler

  If SetRealerSaatTermin(m_handle, nRealerSaatTermin) = 0 Then
    On Error GoTo 0
    Err.Raise 7367, , "SetRealerSaatTermin failed."
  End If

Exit Property
```

```
ErrHandler:
  If gnErrHandler(m_sMOD, "RealerSaatTermin", Err, Erl) = vbRetry Then Resume
End Property
```

```
#If STRINGGET Then
Friend Property Get Region() As String
  If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
  On Error GoTo ErrHandler
  Dim n As Long
  Dim s As String
```

```

s = Space$(256)
If GetRegion(m_handle, s) <> 0 Then
    n = InStr(1, s, Chr$(0))
    If n > 0 Then s = Left$(s, n - 1)
    '*** Return
    Region = s
Else
    On Error GoTo 0
    Err.Raise 7367, , "GetRegion failed."
End If

Exit Property

ErrorHandler:
If gnErrorHandler(m_sMOD, "Region", Err, Erl) = vbRetry Then Resume
End Property
#End If

Friend Property Let Region(ByVal sRegion As String)
If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
On Error GoTo ErrorHandler

If SetRegion(m_handle, sRegion) = 0 Then
    On Error GoTo 0
    Err.Raise 7367, , "SetRegion failed."
End If

Exit Property

ErrorHandler:
If gnErrorHandler(m_sMOD, "Region", Err, Erl) = vbRetry Then Resume
End Property

#If STRINGGET Then
Friend Property Get ReliefEinflussKarte() As String
If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
On Error GoTo ErrorHandler
Dim n As Long
Dim s As String

s = Space$(512)
n = GetReliefEinflussKarte(m_handle, s)
'+ If n <> 0 Then
    n = InStr(1, s, Chr$(0))
    If n > 0 Then s = Left$(s, n - 1)
    '*** Return
    ReliefEinflussKarte = s
'+ Else
'+ On Error GoTo 0
'+ Err.Raise 7367, , "GetReliefEinflussKarte failed."
'+ End If

Exit Property

ErrorHandler:
If gnErrorHandler(m_sMOD, "ReliefEinflussKarte", Err, Erl) = vbRetry Then Resume
End Property
#End If

Friend Property Let ReliefEinflussKarte(ByVal sDateiNamen As String)
If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
On Error GoTo ErrorHandler

If SetReliefEinflussKarte(m_handle, sDateiNamen) = 0 Then
    On Error GoTo 0
    Err.Raise 7367, , "SetReliefEinflussKarte failed."
End If

Exit Property

ErrorHandler:
If gnErrorHandler(m_sMOD, "ReliefEinflussKarte", Err, Erl) = vbRetry Then Resume
End Property

Friend Property Get SaatBettQualitaet() As Long
If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
On Error GoTo ErrorHandler

```

```

Dim n As Long

If GetSaatBettQualitaet(m_handle, n) <> 0 Then
    '*** Return
    SaatBettQualitaet = n
Else
    On Error GoTo 0
    Err.Raise 7367, , "GetSaatBettQualitaet failed."
End If

Exit Property

ErrorHandler:
If gnErrorHandler(m_sMOD, "SaatBettQualitaet", Err, Erl) = vbRetry Then Resume
End Property

Friend Property Let SaatBettQualitaet(ByVal nSaatBettQualitaet As Long)
If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
On Error GoTo ErrorHandler

If SetSaatBettQualitaet(m_handle, nSaatBettQualitaet) = 0 Then
    On Error GoTo 0
    Err.Raise 7367, , "SetSaatBettQualitaet failed."
End If

Exit Property

ErrorHandler:
If gnErrorHandler(m_sMOD, "SaatBettQualitaet", Err, Erl) = vbRetry Then Resume
End Property

Friend Property Get SaatTiefe() As Long
If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
On Error GoTo ErrorHandler
Dim n As Long

If GetSaatTiefe(m_handle, n) <> 0 Then
    '*** Return
    SaatTiefe = n
Else
    On Error GoTo 0
    Err.Raise 7367, , "GetSaatTiefe failed."
End If

Exit Property

ErrorHandler:
If gnErrorHandler(m_sMOD, "SaatTiefe", Err, Erl) = vbRetry Then Resume
End Property

Friend Property Let SaatTiefe(ByVal nSaatTiefe As Long)
If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
On Error GoTo ErrorHandler

If SetSaatTiefe(m_handle, nSaatTiefe) = 0 Then
    On Error GoTo 0
    Err.Raise 7367, , "SetSaatTiefe failed."
End If

Exit Property

ErrorHandler:
If gnErrorHandler(m_sMOD, "SaatTiefe", Err, Erl) = vbRetry Then Resume
End Property

#If STRINGGET Then
Friend Property Get Sorte() As String
If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
On Error GoTo ErrorHandler
Dim n As Long
Dim s As String

s = Space$(256)
n = GetSorte(m_handle, s)
' Der Rückgabewert scheint auch bei Gelingen 0 zu sein???
'+ If n <> 0 Then
    n = InStr(1, s, Chr$(0))

```

```

        If n > 0 Then s = Left$(s, n - 1)
        '*** Return
        Sorte = s
'+ Else
'+     On Error GoTo 0
'+     Err.Raise 7367, , "GetSorte failed."
'+ End If

Exit Property

ErrorHandler:
    If gnErrorHandler(m_sMOD, "Sorte", Err, Erl) = vbRetry Then Resume
End Property
#End If

Friend Property Let Sorte(ByVal sSorte As String)
    If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
    On Error GoTo ErrorHandler

    If SetSorte(m_handle, sSorte) = 0 Then
        On Error GoTo 0
        Err.Raise 7367, , "SetSorte failed."
    End If

Exit Property

ErrorHandler:
    If gnErrorHandler(m_sMOD, "Sorte", Err, Erl) = vbRetry Then Resume
End Property

#If STRINGGET Then
Friend Property Get SpalteAckerzahl() As String
    If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
    On Error GoTo ErrorHandler
    Dim n As Long
    Dim s As String

    s = Space$(256)
    If GetSpalteAckerzahl(m_handle, s) <> 0 Then
        n = InStr(1, s, Chr$(0))
        If n > 0 Then s = Left$(s, n - 1)
        '*** Return
        SpalteAckerzahl = s
    Else
        On Error GoTo 0
        Err.Raise 7367, , "GetSpalteAckerzahl failed."
    End If

Exit Property

ErrorHandler:
    If gnErrorHandler(m_sMOD, "SpalteAckerzahl", Err, Erl) = vbRetry Then Resume
End Property
#End If

Friend Property Let SpalteAckerzahl(ByVal sSpalteAckerzahl As String)
    If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
    On Error GoTo ErrorHandler

    If SetSpalteAckerzahl(m_handle, sSpalteAckerzahl) = 0 Then
        On Error GoTo 0
        Err.Raise 7367, , "SetSpalteAckerzahl failed."
    End If

Exit Property

ErrorHandler:
    If gnErrorHandler(m_sMOD, "SpalteAckerzahl", Err, Erl) = vbRetry Then Resume
End Property

#If STRINGGET Then
Friend Property Get SpalteBodensubstratKarte() As String
    If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
    On Error GoTo ErrorHandler
    Dim n As Long
    Dim s As String

```

```

s = Space$(256)
n = GetSpalteBodensubstratKarte(m_handle, s)
'+ If n <> 0 Then
  n = InStr(1, s, Chr$(0))
  If n > 0 Then s = Left$(s, n - 1)
  '*** Return
  SpalteBodensubstratKarte = s
'+ Else
'+   On Error GoTo 0
'+   Err.Raise 7367, , "GetSpalteBodensubstratKarte failed."
'+ End If

Exit Property

ErrorHandler:
  If gnErrorHandler(m_sMOD, "SpalteBodensubstratKarte", Err, Erl) = vbRetry Then Resume
End Property
#End If

Friend Property Let SpalteBodensubstratKarte(ByVal sSpalteBodensubstratKarte As
String)
  If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
  On Error GoTo ErrorHandler

  If SetSpalteBodenSubstratKarte(m_handle, sSpalteBodensubstratKarte) = 0 Then
    On Error GoTo 0
    Err.Raise 7367, , "SetSpalteBodensubstratKarte failed."
  End If

Exit Property

ErrorHandler:
  If gnErrorHandler(m_sMOD, "SpalteBodensubstratKarte", Err, Erl) = vbRetry Then Resume
End Property

#If STRINGGET Then
Friend Property Get SpalteReliefEinflussKarte() As String
  If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
  On Error GoTo ErrorHandler
  Dim n As Long
  Dim s As String

  s = Space$(256)
  n = GetSpalteReliefEinflussKarte(m_handle, s)
  '+ If n <> 0 Then
  n = InStr(1, s, Chr$(0))
  If n > 0 Then s = Left$(s, n - 1)
  '*** Return
  SpalteReliefEinflussKarte = s
'+ Else
'+   On Error GoTo 0
'+   Err.Raise 7367, , "GetSpalteReliefEinflussKarte failed."
'+ End If

Exit Property

ErrorHandler:
  If gnErrorHandler(m_sMOD, "SpalteReliefEinflussKarte", Err, Erl) = vbRetry Then Re-
sume
End Property
#End If

Friend Property Let SpalteReliefEinflussKarte(ByVal sSpalteReliefEinflussKarte As
String)
  If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
  On Error GoTo ErrorHandler

  If SetSpalteReliefEinflussKarte(m_handle, sSpalteReliefEinflussKarte) = 0 Then
    On Error GoTo 0
    Err.Raise 7367, , "SetSpalteReliefEinflussKarte failed."
  End If

Exit Property

ErrorHandler:
  If gnErrorHandler(m_sMOD, "SpalteReliefEinflussKarte", Err, Erl) = vbRetry Then Re-
sume

```

**End Property**

**Friend Property Get TKG() As Double**

```
If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
On Error GoTo ErrHandler
Dim dbl As Double
```

```
If GetTKG(m_handle, dbl) <> 0 Then
    '*** Return
    TKG = dbl
Else
    On Error GoTo 0
    Err.Raise 7367, , "GetTKG failed."
End If
```

**Exit Property**

ErrHandler:

```
If gnErrorHandler(m_sMOD, "TKG", Err, Erl) = vbRetry Then Resume
```

**End Property**

**Friend Property Let TKG(ByVal dblTKG As Double)**

```
If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
On Error GoTo ErrHandler
```

```
If SetTKG(m_handle, dblTKG) = 0 Then
    On Error GoTo 0
    Err.Raise 7367, , "SetTKG failed."
End If
```

**Exit Property**

ErrHandler:

```
If gnErrorHandler(m_sMOD, "TKG", Err, Erl) = vbRetry Then Resume
```

**End Property**

**#If STRINGGET Then**

**Friend Property Get Vorfrucht() As String**

```
If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
On Error GoTo ErrHandler
Dim n As Long
Dim s As String
```

```
s = Space$(256)
n = GetVorfrucht(m_handle, s)
'+ If n <> 0 Then
    n = InStr(1, s, Chr$(0))
    If n > 0 Then s = Left$(s, n - 1)
    '*** Return
    Vorfrucht = s
'+ Else
'+     On Error GoTo 0
'+     Err.Raise 7367, , "GetVorfrucht failed."
'+ End If
```

**Exit Property**

ErrHandler:

```
If gnErrorHandler(m_sMOD, "Vorfrucht", Err, Erl) = vbRetry Then Resume
```

**End Property**

**#End If**

**Friend Property Let Vorfrucht(ByVal sVorfrucht As String)**

```
If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
On Error GoTo ErrHandler
```

```
If SetVorfrucht(m_handle, sVorfrucht) = 0 Then
    On Error GoTo 0
    Err.Raise 7367, , "SetVorfrucht failed."
End If
```

**Exit Property**

ErrHandler:

```
If gnErrorHandler(m_sMOD, "Vorfrucht", Err, Erl) = vbRetry Then Resume
```

**End Property**

```

Friend Sub SetVisualizerCallback(ByVal pfnVisualizerFunction As Long)
  If m_handle = 0 Then Err.Raise 7367, , "Invalid handle."
  On Error GoTo ErrHandler

  If SetVisualizer(m_handle, pfnVisualizerFunction) = 0 Then
    On Error GoTo 0
    Err.Raise 7367, , "SetVisualizerDll failed."
  End If

  Exit Sub

ErrHandler:
  If gnErrorHandler(m_sMOD, "SetVisualizerCallback", Err, Erl) = vbRetry Then Resume
End Sub

Friend Function Dump() As String
  On Error GoTo ErrHandler
  'VBLM OFF
  Dim dblXMin As Double, dblXMax As Double, dblYMin As Double, dblYMax As Double
  Dim s As String

  s = s & "Ackerzahl:" & vbTab & Ackerzahl & vbCrLf
  s = s & "AckerzahlKarte:" & vbTab & "" & AckerzahlKarte & "" & vbCrLf
  s = s & "BenutzeAckerzahl:" & vbTab & BenutzeAckerzahl & vbCrLf
  s = s & "BenutzeAckerzahlEinheitlich:" & vbTab & BenutzeAckerzahlEinheitlich &
vbCrLf
  s = s & "BenutzeAckerzahlKarte:" & vbTab & BenutzeAckerzahlKarte & vbCrLf
  s = s & "BenutzeBodenSubstratEinheitlich:" & vbTab & BenutzeBodenSubstratEinheitlich & vbCrLf
  s = s & "BenutzeBodenSubstratKarte:" & vbTab & vbTab & BenutzeBodenSubstratKarte &
vbCrLf
  s = s & "BenutzeErtragsPotenzialKarte:" & vbTab & BenutzeErtragsPotenzialKarte &
vbCrLf
  s = s & "BodenfeuchteBeiSaat:" & vbTab & BodenfeuchteBeiSaat & vbCrLf
  s = s & "BodenSaatgutKontakt:" & vbTab & BodenSaatgutKontakt & vbCrLf
  s = s & "BodenSubstrat:" & vbTab & vbTab & "" & BodenSubstrat & "" & vbCrLf
  s = s & "BodenSubstratKarte:" & vbTab & "" & BodenSubstratKarte & "" & vbCrLf
  s = s & "Bundesland:" & vbTab & vbTab & "" & Bundesland & "" & vbCrLf
  s = s & "ErtragsPotenzialKarte:" & vbTab & "" & ErtragsPotenzialKarte & "" &
vbCrLf
  GetKoordinaten dblXMin, dblXMax, dblYMin, dblYMax
  s = s & "Koordinaten:" & vbTab & dblXMin & ";" & dblXMax & ";" & dblYMin & ";" &
dblYMax & vbCrLf
  s = s & "Keimfaehigkeit:" & vbTab & Keimfaehigkeit & vbCrLf
  s = s & "KorrekturErtragserwartung:" & vbTab & KorrekturErtragserwartung & vbCrLf
  s = s & "Niederschlag:" & vbTab & Niederschlag & vbCrLf
  s = s & "Rasterbreite:" & vbTab & Rasterbreite & vbCrLf
  s = s & "RealerSaatTermin:" & vbTab & RealerSaatTermin & vbCrLf
  s = s & "Region:" & vbTab & vbTab & "" & Region & "" & vbCrLf
  s = s & "ReliefEinflussKarte:" & vbTab & "" & ReliefEinflussKarte & "" & vbCrLf
  s = s & "SaatBettQualitaet:" & vbTab & SaatBettQualitaet & vbCrLf
  s = s & "SaatTiefe:" & vbTab & SaatTiefe & vbCrLf
  s = s & "Sorte:" & vbTab & vbTab & "" & Sorte & "" & vbCrLf
  s = s & "SpalteAckerzahl:" & vbTab & "" & SpalteAckerzahl & "" & vbCrLf
  s = s & "SpalteBodensubstratKarte:" & vbTab & "" & SpalteBodensubstratKarte & ""
& vbCrLf
  s = s & "SpalteReliefEinflussKarte:" & vbTab & "" & SpalteReliefEinflussKarte &
"" & vbCrLf
  s = s & "TKG:" & vbTab & vbTab & TKG & vbCrLf
  s = s & "Vorfrucht:" & vbTab & "" & Vorfrucht & ""
'   s = "Katzenklo"

  '*** Return
  Dump = s

  Exit Function

ErrHandler:
  If gnErrorHandler(m_sMOD, "Dump", Err, Erl) = vbRetry Then Resume
End Function

```