PHYTEC

COMBI-Modul 515

Software-Manual

Version 1.1

Ausgabe Dezember 1999

Ein Produkt eines Unternehmens der PHYTEC Technologie Holding AG

Im Buch verwendete Bezeichnungen für Erzeugnisse, die zugleich ein eingetragenes Warenzeichen darstellen, wurden nicht besonders gekennzeichnet. Das Fehlen der © Markierung ist demzufolge nicht gleichbedeutend mit der Tatsache, daß die Bezeichnung als freier Warenname gilt. Ebensowenig kann anhand der verwendeten Bezeichnung auf eventuell vorliegende Patente oder einen Gebrauchsmusterschutz geschlossen werden.

Die Informationen in diesem Handbuch wurden sorgfältig überprüft und können als zutreffend angenommen werden. Dennoch sei ausdrücklich darauf verwiesen, daß die Firma PHYTEC Meßtechnik GmbH weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgeschäden übernimmt, die auf den Gebrauch oder den Inhalt dieses Handbuches zurückzuführen sind. Die in diesem Handbuch enthaltenen Angaben können ohne vorherige Ankündigung geändert werden. Die Firma PHYTEC Meßtechnik GmbH geht damit keinerlei Verpflichtungen ein.

Ferner sei ausdrücklich darauf verwiesen, daß PHYTEC Meßtechnik GmbH weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgeschäden übernimmt, die auf falschen Gebrauch oder falschen Einsatz der Hard- bzw. Software zurückzuführen sind. Ebenso können ohne vorherige Ankündigung Layout oder Design der Hardware geändert werden. PHYTEC Meßtechnik GmbH geht damit keinerlei Verpflichtungen ein.

Informieren Sie sich:

	EUROPA	NORD AMERIKA
Adresse:	PHYTEC Technologie Holding AG Robert-Koch-Str. 39 D-55129 Mainz GERMANY	PHYTEC America LLC 255 Ericksen Avenue NE Bainbridge Island, WA 98110 USA
Angebots Hotline:	+49 (800) 0749832 order@phytec.de	+1 (800) 278-9913 info@phytec.com
Technische Hotline:	+49 (6131) 9221-31 support@phytec.de	+1 (800) 278-9913 support@phytec.com
Fax:	+49 (6131) 9221-33	+1 (206) 780-9135
Web Seite:	http://www.phytec.de	http://www.phytec.com

2. Auflage Dezember 1999

1 Treiberbibliotheken für das COMBI-Modul 515

Das COMBI-Modul 515 stellt dem Anwender eine Vielzahl von digitalen und analogen Ein-/Ausgabekanälen sowie einen Zähler zur Verfügung, mit denen die Verarbeitung verschiedener Signale ermöglicht wird. Ein Großteil der Funktionalität wird durch die Ausnutzung integrierter Baugruppen des C515C erreicht. Die direkte Programmierung des COMBI-Modul 515 setzt daher eine genaue Kenntnis über den internen Aufbau des Prozessors sowie der peripheren Ein-/Ausgabebaugruppen voraus.

Die Treiberbibliothek stellt dem Anwender Funktionen zur Verfügung, mit denen komfortabel auf sämtliche Baugruppen des COMBI-Modul 515 zugriffen werden kann. Sie ermöglicht eine schnelle Realisierung von Projekten ohne Einarbeitung in die Programmierung der On-chip-Komponenten des C515C. Für den Zugriff auf die verschiedenen Ein- und Ausgabekanäle werden die auf den Klemmleisten aufgedruckten symbolischen Bezeichnungen verwendet, ohne daß eine Kenntnis über die Zuordnung zwischen Anschlußklemme und Portpin des Microcontrollers notwendig ist. Weiterhin berücksichtigen die Treiberfunktionen die z.T. vorhandene Negation von Pegeln durch die peripheren Ein- und Ausgabebaugruppen des COMBI-Modul 515. Außerdem wird eine korrekte Initialisierung der Komponenten des C515C bei Kanälen mit Alternativfunktionen sichergestellt.

Ergänzt wird die Treiberbibliothek für das COMBI-Modul 515 durch eine Timerfunktion, die eine Systemzeit mit einer Auflösung von 1 ms bereitstellt. Diese Timerfunktion respektive Initialisierungsfunktion ist in einer separaten Bibliothek enthalten, die optional zur Anwendung gelinkt werden kann.

2 Anwendung der Treiberbibliotheken

2.1 Funktionsumfang der Treiberbibliotheken

Die Treiberbibliothek PCMDRV51.LIB stellt dem Anwender Funktionen folgender Kategorien zur Verfügung:

- Lesen/Setzen eines einzelnen digitalen Ein-/Ausgangs
- Lesen/Setzen einer Gruppe von digitalen Ein-/Ausgängen
- Lesen/Setzen eines analogen Ein-/Ausgangs
- Lesen/Setzen eines Zählerkanals
- Setzen der Statusanzeigen und Abfrage der Schalter auf der CPU-Platine

Die Bibliothek PCMTMR51.LIB ergänzt den Umfang der Treiber um einen Systemzeitgeber mit einer Auflösung von ca.1 ms, so daß eine definierte Zeitbasis für Meß- und Steueraufgaben zur Verfügung steht.

Eine detaillierte Beschreibung der Funktionen folgt in den *Kapitel 4* und *Kapitel 5.*

2.2 Konstanten für Port - und Kanalnummern

Für den Zugriff auf die verschiedenen Ein- und Ausgänge des COMBI-Modul 515 enthalten die Files PCMDRV51.INC und PCMDRV51.H symbolische Konstanten für die Port - bzw. Kanalnummer. Diese entsprechen den auf den Anschlußleisten angegebenen Bezeichnungen, so daß der Zugriff auf alle Ein- und Ausgänge mit Hilfe symbolische Konstanten erfolgen kann, ohne spezielle Kenntnisse über die Zuordnung der Ein- und Ausgänge zu den Portpins des C515C. Insgesamt können folgende Symbole zur Programmierung genutzt werden *(siehe auch Kapitel 3)*:

IN0	... IN18	digitale Eingänge
OUT0	... OUT17	digitale Transistor- und Relaisausgänge
AIN0	... AIN3	analoge Eingänge
AOUT0	... AOUT1	analoge Ausgänge

Die symbolischen Konstanten für die Primärfunktionen entsprechen unmittelbar den Bezeichnungen auf den Anschlußleisten.

Die folgenden Beispiele sollen die Anwendung der Konstanten bei der Programmierung verdeutlichen:

```
Port8 = PCMGetInPort (IN8);       // Port8 abfragen
PCMSetOutPort (OUT0, 1);          // H-Impuls am Port0
PCMSetOutPort (OUT0, 0);          // ausgeben
PCMSetCounter (CIN15, 0);         // Zähler für Port15
Cnt = PCMGetCounter (CIN15);      // bearbeiten
```

Alle weiteren Konstantenvereinbarungen (ON/OFF, RUN/STOP, HIGHRES/LOWRES, ...) werden bei der Beschreibung der jeweiligen Funktionen aufgeführt, für die diese Konstanten verwendet werden können.

2.3 Auswahl von Primär- oder Alternativfunktion

Die Zählerfunktionen für den Eingang IN15 wird durch die Initialisierungsfunktion aktiviert, so daß jede fallende Flanke an dem Eingang IN15 bzw. IN18 zum Incrementieren des zugehörigen Zählers T1 führt. Mit Hilfe der Funktion **PCMGetInPort** wird der aktuelle Eingangszustand abgefragt, die Funktion **PCMGetCounter** ermittelt den bisher erreichten Zählerstand. Beide Eingabefunktionen arbeiten parallel, daß kein Umschalten des Arbeitsmodus erfolgt. Die Festlegung der Betriebsart als Zähler erfolgt einmalig beim Aufruf der Funktion **PCMInitialize**, eine andere Verwendung des Timer/Counter-Kanals ist daher durch anwenderspezifische Software möglich.

2.4 Direkter Zugriff auf Komponenten des C515C

Die vom COMBI-Modul 515 verwendeten On-chip-Komponenten des C515C gestatten teilweise die Nutzung von Betriebsarten, die den Umfang der von der Treiberbibliothek zur Verfügung gestellten Funktionalität übersteigen. Die Verwendung dieser speziellen Betriebsmodi steht dem Anwender in eigenen Routinen jedoch grundsätzlich frei. Zu beachten ist hierbei, daß die Funktion **PCMInitialize** alle von den Treiberfunktionen benötigten Ressourcen initialisiert (*eine Übersicht über die betroffenen Ressourcen ist im Anhang A zu finden*). Die Umprogrammierung von On-chip-Komponenten durch eigene Routinen sollte daher stets erst nach dem Aufruf der Funktion **PCMInitialize** erfolgen.

2.5 Unterstützung von Speichermodellen

Die Bibliotheken mit den Treiberfunktionen für das COMBI-Modul 515 und dem Systemzeitgeber sind unabhängig vom verwendeten Speichermodell (Small, Medium, Large). Die Übergabe von Parametern bzw. Returnwerten erfolgt stets in Registern des Prozessors. Es werden numerische Datentypen und Pointer verwendet.

3 Ein-/Ausgänge des COMBI-Modul 515

Die folgende Übersicht verdeutlicht die Zuordnung von symbolischen Konstanten zu den Ein- und Ausgängen des COMBI-Modul 515. Es werden jeweils die Bezeichnung für die Primärfunktion und - sofern vorhanden - für die Alternativfunktion angegeben. Weiterhin sind die für den Zugriff auf die jeweilige Gruppe von Ein-/Ausgängen anwendbaren Funktionen aufgeführt. Eine detaillierte Beschreibung der Funktionen und ihrer Parameter folgt in den *Kapiteln 4* und *Kapitel 5, „Timerfunktionen“*.

Primär-funktion	**Alternativ-funktion**	**Portpin C515C**	**Bedeutung**	**Anwendbare Funktionen**
IN0		P5.0	Eingang 24 V	PCMGetInGroup1
IN1		P5.1	Eingang 24 V	PCMGetInPort
IN2		P5.2	Eingang 24 V	
IN3		P5.3	Eingang 24 V	
IN4		P5.4	Eingang 24 V	
IN5		P5.5	Eingang 24 V	
IN6		P5.6	Eingang 24 V	
IN7		P5.7	Eingang 24 V	
IN8		P6.4	Eingang 24 V	PCMGetInGroup2
IN9		P6.5	Eingang 24 V	PCMGetInPort
IN10		P6.6	Eingang 24 V	
IN11		P6.7	Eingang 24 V	
IN12	INT0[1]	P3.2	Eingang 24 V interrupt-fähig	
IN13	INT1[1]	P3.3	Eingang 24 V interrupt-fähig	
IN14	T0[2]	P3.4	Eingang 24 V	
IN15	T1 / Counter	P3.5	Eingang 24 V Timer/Counter	PCMGetCounter PCMSetCounter
IN16		P1.5	Eingang 24 V	PCMGetInGroup3
IN17		P1.6	Eingang 24 V	PCMGetInPort
IN18		P1.7	Eingang 24 V	
OUT0		LD1.0[3]	Relaisausgang	PCMSetOutGroup1
OUT1		LD1.1	Relaisausgang	PCMSetOutPort
OUT2		LD1.2	Relaisausgang	
OUT3		LD1.3	Relaisausgang	
OUT4		LD1.4	Relaisausgang	
OUT5		LD1.5	Relaisausgang	
OUT6		LD1.6	Relaisausgang	
OUT7		LD1.7	Relaisausgang	
OUT8		LD2.0	Transistorausgang 24 V	PCMSetOutGroup2,
OUT9		LD2.1	Transistorausgang 24 V	PCMSetOutPort
OUT10		LD2.2	Transistorausgang 24 V	
OUT11		LD2.3	Transistorausgang 24 V	
OUT12		LD2.4	Transistorausgang 24 V	
OUT13		LD2.5	Transistorausgang 24 V	
OUT14		LD2.6	Transistorausgang 24 V	
OUT15		LD2.7	Transistorausgang 24 V	
OUT16	PWM0[1]	P1.0	Transistorausgang 24 V	PCMSetOutPort
OUT17	PWM1[1]	P1.1	Transistorausgang 24 V	
AIN0		P6.0	Eingang 0..10 V	PCMGetADCChannel
AIN1		P6.1	Eingang 0..10 V	
AIN2		P6.2	Eingang 0..10 V	
AIN3		P6.3	Eingang 0..10 V	
AOUT0		P1.2	Ausgang 0..10 V	PCMSetDACChannel,
AOUT1		P1.3	Ausgang 0..10 V	PCMSetDACResolution

1 Wird nicht durch Funktionen der Treiberbibliothek verwendet
2 Wird von der PCMTMR51.LIB als Systemzeitgeber verwendet
3 LDX.Y-> Latch X Datenleistung Y

4 Treiberfunktionen für das COMBI-Modul 515

Die von der Treiber - Bibliothek für das COMBI-Modul 515 (PCMDRV51.LIB) zur Verfügung gestellten Funktionen untergliedern sich in folgende Kategorien:

Initialisierungsfunktion:

PCMInitialize
PCMGetHardwareID

Funktionen zum Abfragen digitaler Eingänge:

PCMGetInGroup1
PCMGetInGroup2
PCMGetInGroup3
PCMGetInPort

Funktionen zum Setzen digitaler Ausgänge:

PCMSetOutGroup1
PCMSetOutGroup2
PCMSetOutPort

Funktion zum Abfragen analoger Eingänge:

PCMGetADCChannel

Funktionen zum Setzen analoger Ausgänge:

PCMSetDACChannel
PCMSetDACResolution

Funktionen zum Setzten und Abfragen des Zählers

PCMGetCounter
PCMSetCounter

Funktionen zum Zugriff auf Bedienelemente:

PCMSetRunLED
PCMSetSysErrLED
PCMSetCANErrLED
PCMSetCardLED
PCMSetBLowLED
PCMSetUserLED
PCMGetSwitch
PCMGetHexNumber
PCMGetDIPSwitch

4.1 Initialisierungsfunktion

Funktion:	**PCMInitialize**
Syntax:	WORD PCMInitialize (BYTE bIOAddrSel)
Input:	bIOAddrSel (R7) = Adresse des I/O Bereiches (UPPER_IO, LOWER_IO)
Output:	WORD (R7, R6) = Versionsnummer des Treibers
Verwendung:	Initialisierung des COMBI-Modul 515 (Festlegung des I/O Bereiches, Rücksetzen der Ausgänge, Initialisierung ADC, DAC und Zähler, Abschalten der Status - LED, Sperren der Interrupts).

Achtung!
Diese Funktion initialisiert alle von den Treiberfunktionen benötigten Ressourcen (*eine Übersicht über die betroffenen Ressourcen ist im Anhang A zu finden*). Die Umprogrammierung von On-chip-Komponenten durch eigene Routinen sollte daher stets erst nach dem Aufruf dieser Funktion erfolgen.

Nach dem Aufruf der Initialisierungsfunktion befindet sich das COMBI-Modul 515 in folgendem Grundzustand:

- digitale Ausgänge inaktiv (Relais abgefallen, Transistoren gesperrt)
- Interrupt für Alternativ-Funktionen der Eingänge gesperrt
- Zähler freigegeben, Betriebsart Vorwärtszähler, zählen bei fallender Flanke
- Anzeige - LED 's inaktiv

Funktion: PCMGetHardwareID

Syntax: BYTE PCMGetHardwareID (void)
Input: ---
Output: BYTE (R7) = Hardwarekennung
Verwendung: Diese Funktion fragt die Hardware-ID der CPU-Platine ab.
Bemerkung: Die Kennung dient zur Bestimmung der Platinenrevision und kann nicht verändert werden.

Beispiel:

```
main
{
BYTE Number;
// ...
Number = PCMGetHardwareID();
// ...
}
```

4.2 Funktionen zum Zugriff auf digitale Eingänge

Funktion: PCMGetInGroup1

Syntax: BYTE PCMGetInGroup1 (void)
Input: ---
Output: BYTE (R7) = Werte der Eingänge IN0..IN7 (Bit0 = IN0, ... , Bit7 = IN7)
Verwendung: Abfrage der Eingabeports IN0 bis IN7.
Bemerkung: keine
siehe auch: **PCMGetInGroup2, PCMGetInGroup3, PCMGetInPort**

Beispiel:

```
main
{
BYTE Input;
// ...
Input = PCMGetInGroup1 ();
// ...
}
```

Funktion: **PCMGetInGroup2**

Syntax: BYTE PCMGetInGroup2 (void)
Input: ---
Output: BYTE (R7) = Werte der Eingänge IN8..IN15 (Bit0 = IN8, ... , Bit7 = IN15)
Verwendung: Abfrage der Eingabeports IN8 bis IN15.
Bemerkung: keine
siehe auch: **PCMGetInGroup1, PCMGetInGroup3, PCMGetInPort, PCMGetCounter, PCMSetCounter**

Beispiel:

```
main
{
BYTE Input;
// ...
Input = PCMGetInGroup2 ();
// ...
}
```

Funktion: **PCMGetInGroup3**

Syntax: BYTE PCMGetInGroup3 (void)
Input: ---
Output: YTE (R7) = Werte der Eingänge IN16..IN18 (Bit0 = IN16, ... , Bit2 = IN18)
Verwendung: Afrage der Eingabeports IN16 bis IN18.
Bemerkung: keine
siehe auch: **PCMGetInGroup1, CMGetInGroup2, PCMGetInPort, PCMGetCounter, PCMSetCounter**

Beispiel:

```
main
{
BYTE Input;
// ...
Input = PCMGetInGroup3 ();
// ...
}
```


Funktion:	**PCMGetInPort**
Syntax:	BYTE PCMGetInPort (BYTE Port)
Input:	Port (R7) = Nummer des einzulesenden Ports (IN0..IN18)
Output:	BYTE (R7) = aktueller Wert eines Eingabeports
Verwendung:	Abfrage eines einzelnen Eingabeports IN0 bis IN18.
Bemerkung:	Die Konstanten IN0..IN18 sind in den Files PCMDRV51.INC bzw. PCMDRV51.H definiert. Ungültige Portnummern werden mit einer Fehlermeldung gekennzeichnet (*siehe Abschnitt 7*).
siehe auch:	**PCMGetInGroup1, PCMGetInGroup2, PCMGetInGroup3**

Beispiel:

```
main
{
bit  Port12;
// ...
Port12 = PCMGetInPort (IN12);
// ...
}
```

4.3 Funktionen zum Zugriff auf digitale Ausgänge

Funktion:	**PCMSetOutGroup1**
Syntax:	void PCMSetOutGroup1 (BYTE RelaisValue)
Input:	RelaisValue (R7) = Ausgabewert für Relais - Gruppe (Bit0 = OUT0, ... , Bit7 = OUT7)
Output:	---
Verwendung:	Setzen der Relaisausgänge OUT0 bis OUT7.
Bemerkung:	keine
siehe auch:	**PCMSetOutGroup2, PCMSetOutPort**

Beispiel:

```
main
{
// Relais für OUT0 und OUT1 einschalten
PCMSetOutGroup1 (0x03);
// ...
}
```

Funktion:	**PCMSetOutGroup2**
Syntax:	void PCMSetOutGroup2 (BYTE TransistorValue)
Input:	TransistorValue (R7) = Ausgabewert für Transistor - Gruppe (Bit0 = OUT 8, ... , Bit7 = OUT15)
Output:	---
Verwendung:	Setzen der Transistorausgänge OUT8 bis OUT15.
Bemerkung:	keine
siehe auch:	**PCMSetOutGroup1, PCMSetOutPort**

Beispiel:

```
main
{
// Transistoren der Ausgänge OUT8 und OUT10
// einschalten
PCMSetOutGroup2 (0x05);
// ...
}
```


Funktion:	**PCMSetOutPort**
Syntax:	BYTE PCMSetOutPort (BYTE Port, BYTE PortValue)
Input:	Port (R7) = Nummer des Ports (OUT0..OUT17) PortValue (R5) = Zustand für Ausgabe
Output:	BYTE (R7) = Returnwert
Verwendung:	Setzen eines einzelnen Ausgabeports OUT0 bis OUT17.
Bemerkung:	Die Konstanten OUT0..OUT17 und die Returncodes sind in den Files ***PCMDRV51.INC*** bzw. ***PCMDRV51.H*** definiert. Ungültige Portnummern werden mit einer Fehlermeldung gekennzeichnet (*siehe Abschnitt 7*).
siehe auch:	**PCMSetOutGroup1, PCMSetOutGroup2**

Beispiel:

```
main
{
BYTE ErrorCode ;
// ...
// Relaisausgang OUT4 einschalten
ErrorCode = PCMSetOutPort (OUT4, 1);
if (ErrorCode != PCM_SUCCESSFUL)
printf ("Fehler Nr. : %04x aufgetreten",ErrorCode);
// ...
}
```

4.4 Funktionen für analoge Ein- und Ausgänge

Darstellung der Analogwerte

Das folgende Schema zeigt die Darstellung von Analogwerten durch die Modultreiber. Diese bilden alle Analogwerte unabhängig von der realen Auflösung des AD- bzw. AD-Wandlers in einem einheitlichen Format als vorzeichenbehafteten 15 -bit Zahlenwert ab.

Allgemeine Darstellung von Analogwerten im Prozeßabbild:

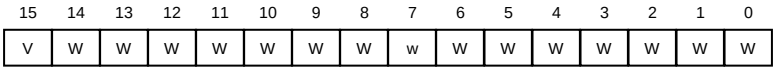
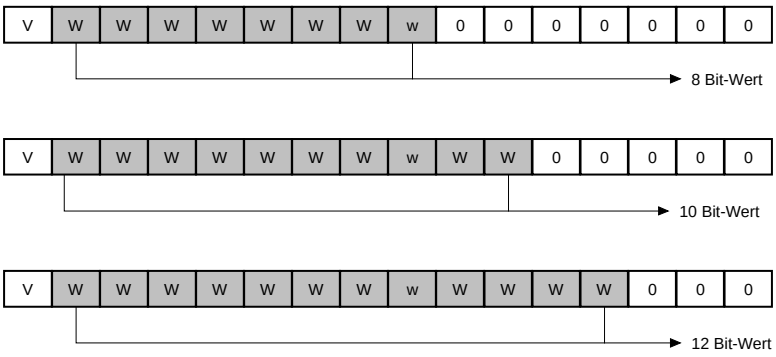


Abbildung der Analogwerte auf 8, 10 und 12 Bit-Wandler:



V = Vorzeichen
W = AD- bzw. DA-Wert

Durch das dargestellte, linksbündige Format wird der zu einer Analoggröße (Spannung, Strom) gehörende Zahlenwert unabhängig von der Auflösung des benutzten Wandlers. Mit größerer Wandlerauflösung erhöht sich jedoch die Zahl der möglichen Zwischenschritte, so daß die Analoggröße genauer abgebildet werden kann. Das gewählte Format zur Darstellung der Analogwerte begünstigt die Softwareentwicklung unabhängig von der jeweils eingesetzten Hardwareplattform und ermöglicht so eine einfache Portierung der Programme auf andere PHYTEC-Steuerungssysteme.

Im folgenden Beispiel wird eine Spannung von 2.5V an ein analoges Eingabemodul mit einem Meßbereich von 0-10V angelegt. Abhängig von der Auflösung des verwendeten AD-Wandlers ergeben sich dabei folgende, reale Ausgabewerte:

8-bit:	40H	z.B. PHYTEC CANopen-/SPS-Chip
10-bit:	100H	z.B. PHYTEC COMBI-Module
12-bit:	400 H	z.B. PHYTEC Analogmodul
15-bit:	2000 H	

Durch die Abbildung der Analoggrößen auf einen 15 -bit Zahlenwert ergibt sich eine Auflösung von maximal 2^{15} = 32768 Schritten. Damit besitzen unipolare Module einen Wertebereich von 0...+32768.

Funktion: **PCMGetADCChannel**

Syntax: int PCMGetADCChannel (BYTE Channel)
Input: Channel (R7) = Nummer des Eingabekanals (AIN0..AIN3)
Output: int (R7, R6) = Umsetzwert des AD-Wandlers (15 -bit, linksbündig mit Vorzeichen, siehe oben)
Verwendung: Lesen eines analogen Eingangs AIN0 bis AIN3.

Achtung!
Die Funktion liefert den Umsetzwert des AD-Wandlers (linksbündig mit Vorzeichen, siehe oben). Die zugehörige Spannung [V] kann durch Multiplikation mit der Konstanten RES_AV010U ermittelt werden. Bei Angabe eines ungültigen Wertes für die Kanalnummer liefert die Funktion einen negativen Wert (-1). Die Konstanten AIN0..AIN3 sowie RES_AV010U sind in den Files ***PCMDRV51.INC*** bzw. ***PCMDRV51.H*** definiert.

siehe auch: **PCMSetDACChannel, PCMSetDACResolution**

Beispiel:

```
main
{
int   ADCin0;
float Uin0;
// ...
// Umsetzwert des AD-Wandlers einlesen
ADCin0 = PCMGetADCChannel (AIN0);
// Eingangsspannung in Volt aus Lesewert des
// AD-Wandlers berechnen
Uin0 = (float)ADCin0* RES_AV010U;
printf("AIN0 = %fV   ", Uin0);
// ...
}
```


Funktion: **PCMSetDACChannel**

Syntax: BYTE PCMSetDACChannel (BYTE Channel, int ADCOut)

Input: Channel (R7) = Nummer des Ausgabekanals (AOUT0, AOUT1)
ADCOut (R5, R4) = Umsetzwert des DA-Wandlers (15 -bit, linksbündig mit Vorzeichen, siehe oben)

Output: BYTE (R7) = Returncode

Verwendung: Schreiben des analogen Ausgangs AOUT0 oder AOUT1.

Achtung!
Die Funktion erwartet als zweiten Parameter den Umsetzwert für den DA-Wandler (linksbündig mit Vorzeichen, siehe oben). Dieser Wert kann aus der auszugebenden Spannung [V] durch Division mit der Konstanten RES_AV010U (siehe Beispiel) bestimmt werden. Die Konstanten AOUT0, AOUT1 sowie RES_AV010U sind in den Files ***PCMDRV51.INC*** bzw. ***PCMDRV51.H*** definiert.

siehe auch: **PCMGetADCChannel, PCMSetDACResolution**

Beispiel:

```
main
{
float Uout0;
int   DACout0;
PCMInitialize ();
// ...
// Ausgabespannung auf 2.75V festlegen
Uout = 2.75;
// Ausgabespannung (in Volt) in Ausgabewert für
// DA-Wandler umrechnen
DACout0 = (int) (Uout0 / RES_AV010U);
PCMSetDACChannel  (AOUT0, DACout0);
// ...
}
```

Funktion: **PCMSetDACResolution**

Syntax: BYTE PCMSetDACResolution (BYTE Resolution)
Input: Resolution (R7) = zu setzende Auflösung (LOWRES=8-bit, HIGHRES=10-bit)
Output: BYTE (R7) = Returncode
Verwendung: Setzen der Auflösung des analogen Ausgangs AOUT0 oder AOUT1 auf 8 oder 10-bit.

Achtung!
Die analoge Ausgangsspannung wird durch einen PWM-Kanal mit nachgeschaltetem aktiven Tiefpaß erzeugt. Bei einer Verringerung der Auflösung auf 8 -bit wird die Welligkeit der Ausgabespannung reduziert. Nach der Initialisierung arbeitet der DAC mit einer Auflösung von 10 -bit. Da beide analogen Ausgänge auf dem Timer2 basieren, können die Auflösungsfaktoren auch nur für beide Kanäle gemeinsam verändert werden. Die Konstanten AOUT0, AOUT1 sowie LOWRES und HIGHRES sind in den Files ***PCMDRV51.INC*** bzw. ***PCMDRV51.H*** definiert.

siehe auch: **PCMSetADCChannel, PCMGetDACChannel**

Beispiel:

```
main
{
int DACout0;
// u.a. DAC-Auflösung auf 10 -bit setzen
// (Standard-Auflösung)
PCMInitialize ();
// Auflösung von 10 -bit auf 8 -bit umschalten
PCMSetDACResolution (LOWRES);
// ...
}
```


4.5 Funktion für Zähler

Funktion:	**PCMGetCounter**
Syntax:	WORD PCMGetCounter (void)
Input :	---
Output:	WORD (R7,R6) = Zählerstand
Verwendung:	Lesen des Zählerkanals CIN15 bzw. CIN18
Bemerkung:	Der Eingang für den Zähler ist eine Alternativfunktion des Einganges IN 15. Die Funktion **PCMSetCounter** ermöglicht ein Voreinstellen des Zählers mit einem beliebigen Wert.
Siehe auch:	**PCMSetCounter**

Beispiel: siehe Beispiel für **PCMSetCounter**

Funktion:	**PCMSetCounter**
Syntax:	void PCMSetCounter (WORD ChannelValue)
Input :	ChannelValue (R7,R6) = zu setzender Zählerwert (Voreinstellwert)
Output:	---
Verwendung:	Setzen des Zählerkanals CIN15
Bemerkung:	Der Eingang für den Zähler ist eine Alternativfunktion des Einganges IN15. Die Funktion **PCMSetCounter** ermöglicht ein Voreinstellen des Zählers mit einem beliebigen Wert.
Siehe auch:	**PCMGetCounter**

Beispiel:

```
main
{
WORD Counter;
/ Zähler für Eingang IN18 auf Zählwert
// 0x100 voreinstellen
PCMSetCounter (0x100);
// ...
// erreichenten Zählerstand auslesen
Counter = PCMGetCounter ();
// ...
}
```

4.6 Funktionen zum Zugriff auf Bedienelemente

Funktionen:	**PCMSetRunLED**, **PCMSetSysErrLED, PCMSetCANErrLED, PCMSetCardLED, PCMSetBLowLED, PCMSetUserLED**
Syntax:	void PCMSetRunLED (BYTE State) void PCMSetSysErrLED (BYTE State) void PCMSetCANErrLED (BYTE State) void PCMSetCardLED (BYTE State) void PCMSetBLowLED (BYTE State) void PCMSetUserLED (BYTE State, BYTE LEDNummer)
Input:	BYTE (R7) = Zustand der LED (ON/OFF) nur PCMSetUserLED: BYTE (R5) = Nummer der UserLED (USER1, USER2, USER3)
Output:	---
Verwendung:	Ein- bzw. Ausschalten der Run-LED, SystemError-LED, CANError-LED, CardLED, BLow-LED oder der User-LED's.

Achtung!
Alle Konstanten (ON, OFF, USER1, USER2, USER3) sind in den Files ***PCMDRV51.INC*** bzw. ***PCMDRV51.H*** definiert.

Beispiel:

```
main
{
// u.a. LED's abschalten
PCMInitialize ();
PCMSetRunLED (ON);
// ...
if (error)
   {
   PCMSetSysErrLED (ON);
   PCMSetRunLED (OFF);
   PCMSetUserLED (ON, USER1);
}
}
```


Funktion: **PCMGetSwitch**

Syntax:	BYTE PCMGetSwitch (void)
Input:	---
Output:	BYTE (R7) = Schalterstellung des Run/Stop - Schalters SWITCH_STOP -> Schalterstellung STOP SWITCH_RUN -> Schalterstellung RUN SWITCH_MRES -> Schalterstellung MRES
Verwendung:	Abfrage des Run/Stop - Schalters.
Bemerkung:	Die Konstanten SWITCH_STOP, SWITCH_RUN und SWITCH_MRES sind in den Files ***PCMDRV51.INC*** bzw. ***PCMDRV51.H*** definiert.

Beispiel:

```
main
{
// u.a. LED’s abschalten
PCMInitialize ();
while (PCMGetSwitch() != SWITCH_RUN);
PCMSetRunLED (ON);
do
  {
   // Zyklus-Schleife
   }
while (PCMGetSwitch() == SWITCH_RUN);
PCMSetRunLED (OFF);
}
```

Funktion:	**PCMGetHexNumber**
Syntax:	BYTE PCMGetHexNumber (void)
Input:	---
Output:	BYTE (R7) = Wert des Hex - Codierschalters
Verwendung:	Abfrage der an den Hex-Codierschaltern eingestellten Nummer.
Bemerkung:	Der Hex - Codierschalter ist für CPU - Adresse reserviert, diese Information ist jedoch nur in Systemen mit mehreren CPU 's von Bedeutung (z.B. CANOpenSlave).

Beispiel:

```
main
{
BYTE CPUAddr;
// ...
// CPU-Adresse abfragen
CPUAddr = PCMGetHexNumber ();
printf ("eingestellte CPU-Adresse: %02BX\n",
        CPUAddr);
// ...
}
```

Funktion: PCMGetDIPSwitch

Syntax: BYTE PCMGetDIPSwitch (void)
Input: ---
Output: BYTE (R7) = Wert des DIP - Schalters
Verwendung: Abfrage des am DIP-Schalter eingestellten Wertes.

Achtung!
Der DIP-Schalter ist für die Einstellung der CAN - Baudrate reserviert. Bei der freien Programmierung durch den Anwender, kann die Funktion vom Anwender bestimmt werden.

Beispiel:

```
main
{
BYTE DIPSw;
// ...
// DIP-Schalter abfragen
DIPSw = PCMGetDIPSwitch ();
printf ("DIP-Schalter: %02BX\n", DIPSw);
// ...
}
```

5 Timerfunktionen

Die Treiberbibliothek mit den bisher beschriebenen Funktionen wird durch eine zweite Bibliothek ergänzt, mit deren Hilfe der Timer0 des C515C als Systemzeitgeber verwendet werden kann.

Die Auslagerung der Timerfunktionen in eine separate Library verhindert die permanente Verwendung des Timer0 und ermöglicht somit die Verwendung der vom Systemzeitgeber verwendeten Ressource in anwenderspezifischen Applikationen. Um den Systemzeitgeber einzubinden, ist die Bibliothek PCMTMR51.LIB im Linkfile anzugeben.

Die Timer-Bibliothek PCMTMR51.LIB stellt dem Anwender folgende Funktionen zur Verfügung:

Funktion zum Starten von Timer0:

StartTimer

Funktion zum Auslesen von Timer0:

GetTickCount

5.1 Funktion zum Starten von Timer0

Funktion:	**StartTimer**
Syntax:	void StartTimer (void)
Input:	---
Output:	---
Verwendung:	Durch Aufruf von StartTimer wird der Timer0 gestartet.
Bemerkung:	Die Initialisierung von Timer0 findet beim Aufruf von **StartTimer** statt. Der verwendete Timer0 hat die fest eingestellte Auflösung von 1 ms. Diese kann nicht verändert werden.
siehe auch:	**GetTickCount**
Beispiel:	siehe Beispiel zu **GetTickCount**

5.2 Funktion zum Auslesen von Timer0

Funktion:	**GetTickCount**
Syntax:	DWORD GetTickCount (void)
Input:	---
Output:	DWORD (R7, R6, R5, R4) = Anzahl der TimerTicks
Verwendung:	Bestimmung der Zeit in ms (Anzahl der TimerTicks) seit dem Aufruf der Funktion **StartTimer** .

Bemerkung: keine

Beispiel:

```
main
{
WORD  wPCMDrvVer;
DWORD Time;
// Initialisieren der Hardware
wPCMDrvVer = PCMInitialize (UPPER_IO);
// Timer0 starten
StartTimer();
// globalen Interrupt freigeben !
EAL = 1;
// ...
// Anzahl der Ticks seit Programmstart lesen
Time = GetTickCount();
// ...
}
```

6 Nutzung von Interrupts

Die Eingänge IN12 und IN13 des COMBI-Modul 515 sind intern auf das Port 3 des C515C geführt und können Interrupts auslösen. Dabei gilt folgende Zuordnung:

IN12 -> Interrupteingang INT0
IN13 -> Interrupteingang INT1

Weiterhin kann der Eingang IN15 als Interrupteingang verwendet werden, wenn die Zählerfunktion nicht benötigt wird. IN15 ist mit dem Eingang T1 des C515 verbunden.

Nach dem Aufruf der Initialisierungsfunktion sind zunächst alle Interrupts gesperrt. Die Interruptprogrammierung erfolgt in der für den C515C üblichen Art und Weise. Es sei hier auf das Handbuch zum C515C verwiesen.

Die explizite Bereitstellung des Interrupthandlers durch den Anwender ist notwendig, da sich die Interruptvektor - Tabelle nur bei Verwendung des Monitor - Modus im RAM befindet, sonst jedoch im Flash liegt und damit zur Programmlaufzeit nicht mehr verändert werden kann. Der Interruptvektor muß daher bereits zum Zeitpunkt der Compilierung bekannt sein, damit ein Segment mit einem entsprechenden Sprungbefehl statisch generiert werden kann.

7 Errorcodes

Die von den Treiberfunktionen zurückgegebenen Errorcodes sind in den Files ***PCMDRV51.INC*** bzw. ***PCMDRV51.H*** definiert und besitzen folgende Bedeutung:

PCM_SUCCESSFUL (0 x 00):
Funktion erfolgreich ausgeführt

PCM_INVALID_CHANNEL (0 x FF):
die beim Aufruf der Funktion angegebene Kanalnummer ist ungültig

PCM_INVALID_AD_CHANNEL (0 x FFFF):
ungültige Kanalnummer für ADC

PCM_INVALID_RESOLUTION (0 x FD):
der beim Aufruf der Funktion angegebene Wert für die PWM – Auflösung ist ungültig.

Bei Definition des Symbols **PCM_ENABLE_WARNING** durch den Anwender (*siehe Kapitel 8*) wird die interne Begrenzung von Übergabeparametern durch eine Warning an das aufrufende Programm gemeldet:

PCM_SUPPRESS_OVERFLOW (0 x 0100):
Übergabeparameter intern begrenzt

8 Der Schalter PCM_ENABLE_WARNING

Im Grundzustand der Treiberbibliothek ist das Symbol **PCM_ENABLE_WARNING** nicht definiert. Damit gilt die Standarddeklaration für die Prototypen der Treiberfunktionen, so daß diese nur Errors als Returnwerte zurückgeben. Damit sind alle Fehler durch ungültige Parameter erkennbar, da diese Fehler einen Returncode kleiner 0x100 liefern. Ein solcher Fehler bedeutet, daß die Funktion nicht ausgeführt wurde.

Wir das Symbol **PCM_ENABLE_WARNING** vom Anwender definiert, erweitert sich der Returncode bestimmter Funktionen (siehe Prototypen) auf ein WORD. Dadurch werden auch die an das aufrufende Programm übergeben Warnungen erkennbar. Diese zeigen an, daß ein bestimmter Parameter intern auf seinen zulässigen Wert begrenzt wurde (z.B. Überlauf des Ausgabewertes bei analogen Modulen). Warnungen besitzen einen Returncode größer als 0 x 100 und setzen daher die Erweiterung des Rückgabewertes auf des Registers R6 voraus.

Bei nicht definiertem Symbol **PCM_ENABLE_WARNING** (Grundzustand), kann durch die Unterdrückung von Warnungen der vom Treiber zurückgelieferte Returnwert leichter ausgewertet werden.

```
int DACOut;
DACOut = (int)(9.99/RES_HIGH);
if ( PCMSetDACChannel(AOUT0, DACOut) )
{
// schwerwiegender Fehler!
}
// die Bereichsüberschreitung um ein Digit
// durch Rundungsfehler wird ignoriert, der
// Treiber hat den Ausgabewert jedoch intern
// auf die maximal zulässige Größe begrenzt
```

Zu beachten ist, daß die Begrenzung unabhängig von der Deklaration des Symbols **PCM_ENABLE_WARNING** erfolgt, für das aufrufende Programm wird sie aber nur bei definiertem Symbol erkennbar!

9 Softwarestruktur

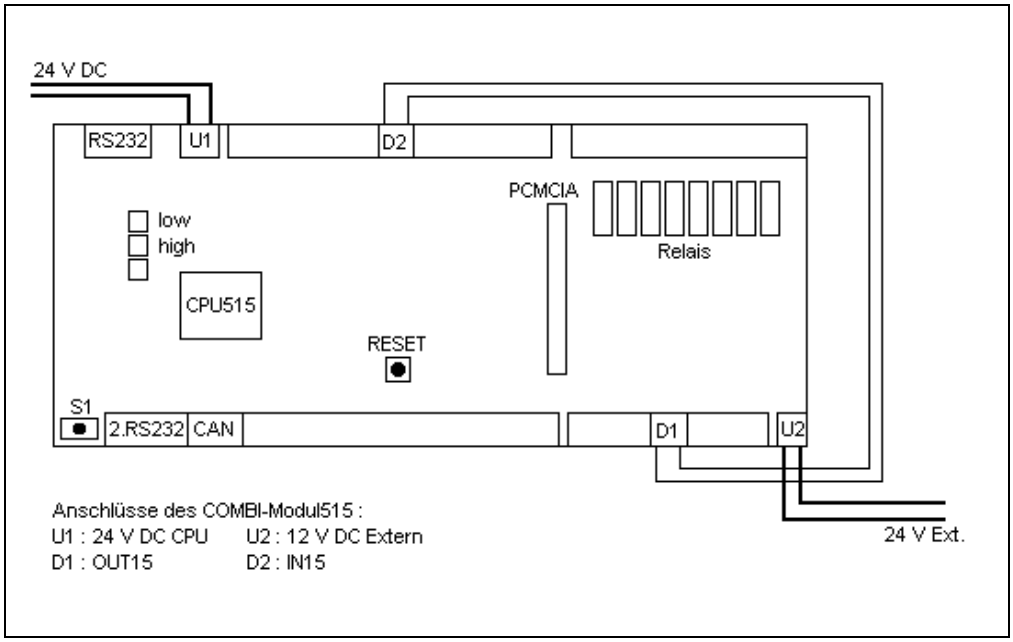
Die Treiberfunktionen für das COMBI-Modul 515 ermöglichen einen einfachen Zugriff auf die verschiedenen Baugruppen des Moduls. Die zum Standard - Lieferumfang gehörenden Dateien besitzen folgende Bedeutung:

PCMDRV51.LIB	Bibliothek mit Funktionen zum Zugriff auf Baugruppen des COMBI-Modul 515
PCMTMR51.LIB	Bibliothek mit Funktionen für Systemzeitgeber
PCMDRV51.H PCMDRV51.INC	Definitionsdatei für Bibliothek mit Treiberfunktionen (Konstanten, Returncodes, Prototypen)
PCMTMR51.H PCMTMR51.INC	Definitionsdatei für Bibliothek mit Systemzeitgeber (Prototypen)
PCMDEMO.C	Demoprogramm für Funktionen der Bibliotheken PCMDRV51.LIB und PCMTMR51.LIB
PCMDEMO.L51	Linkerfile für Demoprogramm
STARTUP.A51	Startup für Demoprogramm

Darüber hinaus besteht die Möglichkeit, bei der Fa. PHYTEC eine Quellcode-Lizenz der Bibliotheken zu erwerben.

10 Hinweise zum Demoprogramm

Das Programm **PCMDEMO.C** setzt zur Veranschaulichung des vollen Funktionsumfanges eine externe Verbindungen des Ausgang OUT15 ist mit den Eingang IN15 voraus.



Zum besseren Verständnis der Funktionalität wird das Demoprogramm im folgenden kurz erläutert:

Das Demoprogramm initialisiert die Hardware, den Timer0 und gibt den globalen Interrupt frei. Nachdem der Schalter S1 (Run/Stop-Schalter) in die Stellung 'RUN' (rechts) gebracht wurde, beginnt die Abarbeitung des "Programmzyklus". Dazu wird jeweils nach 100 ms der Wert der Ausgangsruppe2 um eine Bitposition nach rechts verschoben. Über den Ausgang OUT15 dieser Ausgangsgruppe wird der Eingang CIN15 getaktet. Der auf einen vorinitialisierten Wert eingestellte Zähler wird bei jeder fallenden Flanke inkrementiert. Desweiteren werden die HEX-Drehschalter und die DIP-Schalter abgefragt. Je nach Stellung der Schieber 1 und 2 am DIP-Schalter werden die USER-LED's gesetzt. Das Programm wird beendet, indem der Schalter S1 in die Stellung 'STOP' (Mitte) gebracht wird.
Über die RS-232 Schnittstelle und ein Terminalprogramm, können die Aktivitäten des COMBI-Modul 515 beobachtet werden.

Anhang A

Von den Treiberfunktionen für das COMBI-Modul 515 werden folgende On-chip-Komponenten des C515C verwendet:

- P1, P3, P4, P5
- T1, T2
- ADCON

Bei Verwendung der Bibliothek für den Systemzeitgeber werden zusätzlich noch folgende Ressourcen belegt:

- T0

Index

Dokument: **COMBI-Modul 515**
Dokumentnummer: **L-343d_2, Dezember 1999**

Wie würden Sie dieses Handbuch verbessern?

Haben Sie in diesem Handbuch Fehler entdeckt? Seite

Eingesandt von:

Kundennummer: _____

Name: _____

Firma: _____

Adresse: _____

_____

Einsenden an:

PHYTEC Technologie Holding AG
Postfach 100403
D-55135 Mainz, Germany
Fax : +49 (6131) 9221-33

Published by

PHYTEC

Ordering No. L-343d_2
Printed in Germany