PHYTEC

Module Driver
for
COMBI-Modul 515
Version 1.1

Software-Manual

Preliminary Edition December 1999

A product of a PHYTEC Technology Holding company

In this manual are descriptions for copyrighted products which are not explicitly indicated as such. The absence of the trademark (©) symbol does not infer that a product is not protected. Additionally, registered patents and trademarks are similarly not expressly indicated in this manual

The information in this document has been carefully checked and is believed to be entirely reliable. However, PHYTEC Elektronik GmbH assumes no responsibility for any inaccuracies. PHYTEC Elektronik GmbH neither gives any guarantee nor accepts any liability whatsoever for consequential damages resulting from the use of this manual or its associated product. PHYTEC Elektronik GmbH reserves the right to alter the information contained herein without prior notification and accepts no responsibility for any damages which might result.

Additionally, PHYTEC Elektronik GmbH offers no guarantee nor accepts any liability for damages arising from the improper usage or improper installation of the hardware or software. PHYTEC Elektronik GmbH further reserves the right to alter the layout and/or design of the hardware without prior notification and accepts no liability for doing so.

	EUROPE	NORTH AMERICA
Address:	PHYTEC Technologie Holding AG Robert-Koch-Str. 39 D-55129 Mainz GERMANY	PHYTEC America LLC 255 Ericksen Avenue NE Bainbridge Island, WA 98110 USA
Ordering Information:	+49 (800) 0749832 order@phytec.de	+1 (800) 278-9913 order@phytec.com
Technical Support:	+49 (6131) 9221-31 support@phytec.de	+1 (800) 278-9913 support@phytec.com
Fax:	+49 (6131) 9221-33	+1 (206) 780-9135
Web Site:	http://www.phytec.de	http://www.phytec.com

1st Edition: Decmber 1999

1 Driver library for the COMBI Modul-515

The COMBI Modul-515 offers a large amount of digital and analog input/output channels and counters to the user. With these components processing of different types of industry-standard signals can be done easily. Most of the module's functionality is achieved by using the integrated features of the infineon's C515 microcontroller. Programming the COMBI Modul-515 assumes detailled knowledge of the controller's internal structure and the peripheral input/output units as well.

The driver library places functions to the customers disposal, that allow comfortable and easy acces to all the units on-board of the COMBI Modul-515. They make rapid realization of complex projects possible, without needing detailed knowledge of programming the on-chip components of the C515 controller. The symbolic terms printed on the modules connector row's are used for functions that access the various input and output channels. Detailed knowledge of internal dependencies between output connectors and port pin on the microcontroller is not necessary. The driver functions also pay attention to partly done negations of signal levels caused by the peripheral input/output units of the COMBI Modul-515. Furthermore proper initialization of the C515's components for channels with alternativ functions is guaranteed.

The driver library for the COMBI Modul-515 is completed by one timer function, that offers a system time with a resolution of 1 ms. These timer functions respectively the initializing function is included in a separate library which can be linked to the application software.

2 Use of the driver library

2.1 Driver library function contents

The driver library PCMDRV51.LIB places functions of different catagories at user's disposal as listed below:

- Read/write a single digital input/output
- Read/write a group of digital inputs/outputs
- Read/write an analog input/output
- Read/write a counter channel
- Set the status displays and reads the state of the on-board switches

2.2 Constants for port and channel numbers

The files PCMDRV51.INC and PCMDRV51.H contain symbolic constants for port and channel numbers to enable access to various inputs/outputs on the COMBI Modul-515. These constants refer to terms to be found on the connector rows. The access to all inputs/outputs can be done using this symbolic constants without special knowledge of the internal links between port pins on the C515 controller and the dedicated input/output. For inputs/outputs with alternativ functions there are additional constants defined. Alltogether the follwoing symbols can be used for programming purposes (refer also to chapter 3):

Primary functions:

IN0	... IN18	digital inputs
OUT0	... OUT17	digital transistor and relay output
AIN0	... AIN3	analog inputs
AOUT0	... AOUT1	analog outputs

The symbolic constants for the primary functions correspond directly to the labels on the connector rows.

The examples listed below show the use of these constants for programming purposes:

```
Port8 = PCMGetInPort (IN8);          // read request to port8

PCMSetOutPort (OUT0, 1);             // output a high level
PCMSetOutPort (OUT0, 0);             // to port0

PCMSetCounter (CIN15, 0);            // process
Cnt = PCMGetCounter (CIN15);         // counter for port20
```

All further constants definitions (ON/OFF, RUN/STOP, HIGHRES/LOWRES, ...) are described at the dedicated functions these constants can be used.

2.3 Selection of primary or alternativ function

The counter functions for input IN15 are automaticly activated by the initializing function. So every rising edge of signals connected to inputs IN15 resp. IN18 leads to incrementing of the dedicated counter T1. The function **PCMGetInPort** queries the current state of the input, the function **PCMGetCounter** calculates the current counter level. Both input functions work in parallel, switching between work modes is not caused. The definition of the operation mode as a counter is done at once at time of calling the function **PCMInitialize.** Thats why, another use of the timer/counter channels is possible.

2.4 Direct acces to the C515 components

The C515 controller's on-chip components, used by the COMBI Modul-515, allow partly utilization of certain operating modes. Use of these modes may exceed the extent the driver library's functionality. It is principle possible to use these certain modes with support of own routines by the software programmer. It has to be noted, that the function **PCMInitialize** initializes all required ressources used by the driver functions. A summary of all ressources affected by this can be found in attachment A. The reprogramming of on-chip components by own software routines should be done always after calling the function **PCMInitialize.**

2.5 Support of memory models

The library with the driver functions for the COMBI Modul-515 and the system timer are independent from memory model used (small, medium, large). The hand over of parameters respectively return values is done always using the processor registers. Exclusive numeric data types are used and no pointers. Because of the explizit prototype declaration as "FAR" the assembler and C-compiler use the CALLS instruction for a function call, independent from the adjusted memory model. The CALLS instruction allows a function call to any segment within the controller‘s address space.

3 Inputs and Outputs on the COMBI Modul-515

The summary below shows the connection between symbolic constants and inputs/outputs on the COMBI Modul-515. The term for the primary function and the alternate function if available are indicated each. Furthermore the usable functions for access to the respectiv group of inputs/outputs are listed. A more detailed description of these functions and their parameters is done in chapters 4 and 5.

Primary function	Alternate function	Portpin C515	Description	Available functions
IN0		P5.0	Input 24V	PCMGetInGroup1,
IN1		P5.1	Input 24V	PCMGetInPort,
IN2		P5.2	Input 24V	
IN3		P5.3	Input 24V	
IN4		P5.4	Input 24V	
IN5		P5.5	Input 24V	
IN6		P5.6	Input 24V	
IN7		P5.7	Input 24V	
IN8		P6.4	Input 24V	PCMGetInGroup2,
IN9		P6.5	Input 24V	PCMGetInPort
IN10		P6.6	Input 24V	
IN11		P6.7	Input 24V	
IN12	INT0 (Note1)	P3.2	Input 24V, INT-capable	
IN13	INT1 (Note1)	P3.3	Input 24V, INT-capable	
IN14	T0 (Note2)	P3.4	Input 24V, Timer	
IN15	T1` / Counter	P3.5	Input 24V, Timer/Counter	PCMGetCounter PCMSetCounter
IN16		P1.5	Input 24V	PCMGetInGroup3,
IN17		P1.6	Input 24V	PCMGetInPort
IN18		P1.7	Input 24V	
OUT0	(Note3)	LD1.0	Relay output	PCMSetOutGroup1,
OUT1		LD1.1	Relay output	PCMSetOutPort
OUT2		LD1.2	Relay output	
OUT3		LD1.3	Relay output	
OUT4		LD1.4	Relay output	
OUT5		LD1.5	Relay output	
OUT6		LD1.6	Relay output	
OUT7		LD1.7	Relay output	
OUT8		LD2.0	Transistor output 24V	PCMSetOutGroup2,
OUT9		LD2.1	Transistor output 24V	PCMSetOutPort
OUT10		LD2.2	Transistor output 24V	
OUT11		LD2.3	Transistor output 24V	
OUT12		LD2.4	Transistor output 24V	
OUT13		LD2.5	Transistor output 24V	
OUT14		LD2.6	Transistor output 24V	
OUT15		LD2.7	Transistor output 24V	
OUT16	PWM0 (Note1)	P1.0	Transistor output 24V	PCMSetOutPort,
OUT17	PWM1 (Note1)	P1.1	Transistor output 24V	
AIN0		P6.0	Input 0..10V	PCMGetADCChannel
AIN1		P6.1	Input 0..10V	
AIN2		P6.2	Input 0..10V	
AIN3		P6.3	Input 0..10V	
AOUT0		P1.2	Output 0..10V	PCMSetDACChannel,
AOUT1		P1.3	Output 0..10V	PCMSetDACResolution

Table 1: Summary of inputs and outputs on the COMBI Modul-515

Note1: Not used by functions of the driver library
Note2: Used by the PCMTMR51.LIB as system timer
Note3: LD X.Y -> Latch X, data line Y

4 Driver functions for the COMBI Modul 515

The functions offered by the driver library for the COMBI Modul-515 (PCMDRV51.LIB) are structured in categories listed below:

Initializing functions

PCMInitialize
PCMGetHardwareID

Functions for query digital inputs:

PCMGetInGroup1
PCMGetInGroup2
PCMGetInGroup3
PCMGetInPort

Functions for setting digital outputs:

PCMSetOutGroup1
PCMSetOutGroup2
PCMSetOutPort

Functions for query analog inputs:

PCMGetADCChannel

Functions for setting analog outputs:

PCMSetDACResolution
PCMSetDACChannel

Functions for setting and query counters:

PCMGetCounter
PCMSetCounter

Functions to access the control and display units

PCMSetRunLED
PCMSetSysErrLED
PCMSetCANErrLED
PCMSetCardLED
PCMSetBLowLED
PCMSetUserLED
PCMGetSwitch
PCMGetHexNumber
PCMGetDIPSwitch

4.1 Initializing function

Function: PCMInitialize

Syntax: WORD PCMInitialize (BYTE bIOAddrSel)

Input: bIOAddrSel (R7) = I/O address range (UPPER_IO, LOWER_IO)

Output: WORD (R7, R6) = number of driver version

Usage: Initialize the COMBI Modul-515 (defines the I/O range, resets the outputs, initialize ADC, DAC and counters, turns off the status-LEDs, suspends the interrupts).

Comment: This function initializes all ressources required from the driver functions (an overview of all affected ressources can be found in Attachment A). Any reprogramming of on-chip components by user software routines always should be done after calling this function first.

After calling this initializing function the COMBI Modul-515 is in the following pre-operational state:

- Digital outputs are inactiv (relays sloped down, transistors cut off)
- Interrupt disabled for inputs alternative functions
- counter released, mode of operation is upward counter, counts at rising signal edge
- Display-LEDs inactiv

Function: PCMGetHardwareID

Syntax: BYTE PCMGetHardwareID (void)

Input: ---

Output: BYTE (R7) = hardware identification

Usage: This function detects the current hardware ID of the COMBImodul-515.

Comment: This hardware identification is used to determine the board‘s revision number and can not be changed.

Example:

```
main
{

BYTE  Number;

// ...

Number = PCMGetHardwareID();

// ...

}
```

4.2 Functions to access digital inputs

Function: PCMGetInGroup1

Syntax: BYTE PCMGetInGroup1 (void)

Input: ---

Output: BYTE (R7) = value on inputs IN0..IN7 (bit0 = IN0, ... , bit7 = IN7)

Usage: Query the input ports IN0 to IN7.

Comment: none

Refer also to: PCMGetInGroup2, PCMGetInGroup3, PCMGetInPort

Example:

```
main
{

BYTE  Input;

// ...

Input = PCMGetInGroup1 ();

// ...

}
```


Function: PCMGetInGroup2

Syntax: BYTE PCMGetInGroup2 (void)

Input: ---

Output: BYTE (R7) = value on inputs IN8..IN15 (Bit0 = IN8, ... , Bit7 = IN15)

Usage: Query the input ports IN8 to IN15.

Comment: none

Refer also to: PCMGetInGroup1, PCMGetInGroup3, PCMGetInPort, PCMGetCounter, PCMSetCounter

Example:

```
main
{

BYTE  Input;

// ...

Input = PCMGetInGroup2 ();

// ...

}
```

Function: PCMGetInGroup3

Syntax: BYTE PCMGetInGroup3 (void)

Input: ---

Output: BYTE (R7) = value on inputs IN16...IN18 (Bit0 = IN16, ... , Bit2 = IN18)

Usage: Query the input ports IN16 to IN18.

Comment: none

Refer also to: PCMGetInGroup1, PCMGetInGroup2, PCMGetInPort, PCMGetCounter, PCMSetCounter

Example:

```
main
{

BYTE  Input;

// ...

Input = PCMGetInGroup3 ();

// ...

}
```


Function: PCMGetInPort

Syntax: BYTE PCMGetInPort (BYTE Port)

Input: Port (R7) = number of the port to be read (IN0..IN18)

Output: bit (R4.0) = current value of the input port

Usage: Query the several input ports IN0 to IN18.

Comment: Invalid port numbers will be ignored without any error message. The constants IN0..IN18 are defined in the files PCMDRV51.INC resp. PCMDRV51.H.

Refer also to: PCMGetInGroup1, PCMGetInGroup2, PCMGetInGroup2

Example:

```
main
{

bit   Port12;

// ...

Port12 = PCMGetInPort (IN12);

// ...

}
```

4.3 Functions to access digital outputs

Function: PCMSetOutGroup1

Syntax: void PCMSetOutGroup1 (BYTE RelaisValue)

Input: RelaisValue (R7) = output value to relay group (Bit0 = OUT0, ... , Bit7 = OUT7)

Output: ---

Usage: Set the relay outputs OUT0 to OUT7.

Comment: none

Refer also to: PCMSetOutGroup2, PCMSetOutPort

Example:

```
main
{

// Turn on Relays for OUT0 and OUT1
PCMSetOutGroup1 (0x03);

// ...

}
```


Function: PCMSetOutGroup2

Syntax: void PCMSetOutGroup2 (BYTE TransistorValue)

Input: TransistorValue (R7) = output value for transistor group (Bit0 = OUT 8, ... , Bit7 = OUT15)

Output: ---

Usage: set transistor outputs OUT8 to OUT15.

Comment: none

Refer also to: PCMSetOutGroup1, PCMSetOutPort

Example:

```
main
{

// Turn on transistors for outputs OUT8 and OUT10

PCMSetOutGroup2 (0x05);

// ...

}
```

Function: PCMSetOutPort

Syntax: BYTE PCMSetOutPort (BYTE Port, BYTE PortValue)

Input: Port (R7) = number of the port (OUT0..OUT17)
PortValue (R5) = output bit value

Output: BYTE (R7) = Return value

Usage: Set a single output port OUT0 to OUT17.

Comment: The constants OUT0..OUT17 and the return codes are defined in the files PCMDRV51.INC resp. PCMDRV51.H. Invalid port numbers will be ignored without any error message.

Refer also to: PCMSetOutGroup1, PCMSetOutGroup2

Example:

```
main
{
BYTE ErrorCode ;

// ...

// Turn on Relay output OUT4
ErrorCode = PCMSetOutPort (OUT4, 1);
if (ErrorCode != PCM_SUCCESSFUL)
   printf ("Error Nr. : %04x occured"
           ,ErrorCode);
// ...

}
```


4.4 Function for analog inputs and outputs

Format definition for analog values

Function: PCMGetADCChannel

Syntax: int PCMGetADCChannel (BYTE Channel)

Input: Channel (R7) = number of the input channel (AIN0..AIN3)

Output: int (R7, R6) = conversion result of the ADC (15 Bit), Align left with sign

Usage: Reading an analog input AIN0 to AIN3.

Comment: This function returns the directly convertion value of the ADC (Align left with sign). The corresponding voltage [V] can be calculated by multiplication with the constant RES_AV010U. Using an invalid value for the channel number this function returns a negativ value (-1). The constants AIN0..AIN3 as well as RES_AV010U are defined in the files PCMDRV51.INC resp. PCMDRV51.H.

Refer also to: PCMSetDACChannel, PCMSetDACResolution

Example:

```
main
{

int   ADCin0;
float Uin0;

// ...

// Read ADC Convertion value
ADCin0 = PCMGetADCChannel (AIN0);

// Calculate the input voltage
// of the AD-Converter in Volt

Uin0 = (float)ADCin0* RES_AV010U;
printf("AIN0 = %fV   ", Uin0);
// ...

}
```

Function: PCMSetDACChannel

Syntax: BYTE PCMSetDACChannel (BYTE Channel,
int ADCOut)

Input: Channel (R7) = number of output channel (AOUT0, AOUT1)
ADCOut (R5, R4) = Convertion value for DAC (15 Bit, Align left with sign)

Output: BYTE (R7) = Return code

Usage: Writing the analog Outputs AOUT0 or AOUT1.

Comment: This function expects the convertion value for the DAC as the second parameter (align left with sign). This value can be calculated by dividing the voltage [V] that wants to be output and the constant RES_AV010U (refer to the examplel).
The constants AOUT0, AOUT1 as well as RES_AV010Uare defined in the files PCMDRV51.INC resp. PCMDRV51.H.

Refer also to: PCMGetADCChannel, PCMSetDACResolution

Example:

```
main
{

float Uout0;
int   DACout0;

PCMInitialize ();

// ...

// set the output voltage to 2.75 V
Uout = 2.75;

// convert the output voltage (in volt) into
// the output value for the DA-Converter

DACout0 = (int) (Uout0 / RES_AV010U);

PCMSetDACChannel  (AOUT0, DACout0);

// ...

}
```

Function: PCMSetDACResolution

Syntax: BYTE PCMSetDACResolution (BYTE Channel, BYTE Resolution)

Input: Resolution (R7) = resolution to be set (LOWRES=8Bit, HIGHRES=10Bit)

Output: BYTE (R7) = Return code

Usage: Setting the resolution of the analog output AOUT0 or AOUT1 to 8 or 10 Bit.

Comment: The analog output voltag is generated by a PWM channel with a downline activ low-pass. Reducing the resolution to 8 bit causes a reduction of signal noise of the output voltage. After initialization the DAC works with a resolution of 10 Bit. As both analog output signals are based on the timer2, the resolution for both analog outputs can be changed only together.
The constants AOUT0, AOUT1 as well as LOWRES and HIGHRES are defined in the files PCMDRV51.INC resp. PCMDRV51.H.

Refer also to: PCMSetDACChannel

Example:

```
main
{

int DACout0;

// a.o. set DAC-resolution to 10 Bit
// (standard resolution)
PCMInitialize ();


// switch resolution from10 Bit to 8 Bit
PCMSetDACResolution (LOWRES);



// ...

}
```

4.5 Counter functions

Function: PCMGetCounter

Syntax: WORD PCMGetCounter (void)

Input: ---

Output: WORD (R7, R6) = counter state

Usage: Read a counter channel CIN15 resp. CIN18.

Comment: The input for the counter is an alternativ function of the input port IN15. The function **PCMSetCounter** makes a preset of the counter with a certain value possible. The constants CIN15 resp. CIN18 are defined in the files PCMDRV51.INC resp. PCMDRV51.H.

Refer also to: PCMSetCounter

Example: Refer to Example for **PCMSetCounter**

Function: PCMSetCounter

Syntax: void PCMSetCounter (WORD ChannelValue)

Input: ChannelValue (R7, R6) = counter value to be set (preadjust value)

Output: ---

Usage: Setting a counter channel CIN15

Comment: The counter input is an alternative function for the digital inputs IN15. The function **PCMSetCounter** makes a preset of the counter with a certain value possible. The constants CIN15 are defined in the files PCMDRV51.INC resp. PCMDRV51.H.

Refer also to: PCMGetCounter

Example:

```
    main
    {

    WORD Counter;

    // Preset counter for input IN18
    // to a counter value 0x100
    PCMSetCounter (0x100);

    // ...

    // read current counter value
    Counter = PCMGetCounter ();

    // ...

}
```

4.6 Functions for access to the display units

Functions: **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 LED number)

Input: BYTE (R7) = LED state (ON/OFF)
Only valid for PCMSetUserLED:
BYTE (R5) = Number of the user LED (USER1, USER2, USER3)

Output: ---

Usage: Turn-on resp. turn-off the Run-LED, SystemError-LED, CANError-LED, CardLED, Blow-LED or the user LED‘s.

Comment: All constants ON, OFF, USER1, USER2, USER3 are defined in the files PCMDRV51.INC resp. PCMDRV51.H.

Example:

```
main
{
// a.o. turn-off LED's
PCMInitialize ();
PCMSetRunLED (ON);

// ...

if (error)
   {
   PCMSetSysErrLED (ON);
   PCMSetRunLED (OFF);
   PCMSetUserLED (ON, USER1);
   }
}
```

Function: PCMGetSwitch

Syntax: BYTE PCMGetSwitch (void)

Input: ---

Output: BYTE (R7) = position of the Run/Stop switch
SWITCH_STOP -> position STOP
SWITCH_RUN -> position RUN
SWITCH_MRES -> position MRES

Usage: Query the Run/Stop switch.

Comment: The constants SWITCH_STOP, SWITCH_RUN and SWITCH_MRES are defined in the files PCMDRV51.INC resp. PCMDRV51.H.

Example:

```
main
{
// a.o. turn-off LED's
PCMInitialize ();

while (PCMGetSwitch() != SWITCH_RUN);
PCMSetRunLED (ON);

do
   {

   // cycle loop

   }
while (PCMGetSwitch() == SWITCH_RUN);
PCMSetRunLED (OFF);
}
```


Function: PCMGetHexNumber

Syntax: BYTE PCMGetHexNumber (void)

Input: ---

Output: BYTE (R7) = number adjusted on the HEX encode switch

Usage: Query the number adjusted on the HEX encode switch.

Comment: The HEX encode switch is reserved for the CPU address. This is only relevant for systems using several CPU's, such as in a network as CANopenSlave.

Example:

```
main
{

BYTE CPUAddr;

// ...

// Get CPU-addresse
CPUAddr = PCMGetHexNumber ();

printf ("selected CPU-Address: %02BX\n",
        CPUAddr);

// ...

}
```

Function: PCMGetDIPSwitch

Syntax: BYTE PCMGetDIPSwitch (void)

Input: ---

Output: BYTE (R7) = value adjusted on DIP-switches

Usage: Query the adjusted value of the DIP-switch

Comment: The DIP-switch is reserved for selection of the CAN transmission baudrate..

Example:

```
main
{

BYTE DIPSw;

// ...

// Query DIP-Switch
DIPSw = PCMGetDIPSwitch ();

printf ("DIP-Switch: %02BX\n", DIPSw);

// ...

}
```


5 Timer functions

The driver library with all the functions described so far is complemented by a second library. Using this second library the timer0 of the C515 can be used as system timer .

The exclusion of this timer function into a separate library relieves the permanent use of timer0. This enables the free use of this system timer ressource in the own application. To include the system timer in the user software, inlude the library PCMTMR51.LIB into the linkfile.

The timer library PCMTMR51.LIB provides the following functions:

Function to start timer0

StartTimer

Function to read timer0

GetTickCount

5.1 Function to start Timer0

Function: **StartTimer**

Syntax: void StartTimer (void)

Input: ---

Output: ---

Usage: Calling the function **StartTimer** starts the Timer0.

Comment: Timer0 will be initialized at the time calling the function **StartTimer**. The resolution for Timer0 is fixed to 1 ms and can not be changed.

Example: Refer to Example for **GetTickCount**

5.2 Function to read Timer0

Function: **GetTickCount**

Syntax: DWORD GetTickCount (void)

Input: ---

Output: DWORD (R7, R6, R5, R4) = number of TimerTicks

Usage: Calculating the time in ms (number of TimerTicks) since calling the function **StartTimer**

Comment: none

Example:

```
main
{
WORD  wPCMDrvVer;
DWORD Time;

// Init the hardware
wPCMDrvVer = PCMInitialize (UPPER_IO);

// start Timer
StartTimer();

// enable global Interrup !
EAL = 1;

// ...
// number of TimerTicks since calling the function
Time = GetTickCount();

// ...
}
```

6 Using interrupts

The input ports IN12..IN13 on the COMBI Modul-515 are routed internal to port 3 of the C515C controller. Thats why they can release edge triggered interrupts. The following configuration is uesed:

IN12 -> Interrupt Input INT0
IN13 -> Interrupt Input INT1

Furthermore, input IN15 can also be used as an interrupt input, in case the counter function is not required. IN15 is internal connected to input T1 in the C515C controller.

After calling the initialize function all interrupts are suspend first. The programming procedure for interrupts is done according to the description of the C515C microcontroller, for further information refer to the controller user‘s manual.

It is necessary to provide the interrupt handler by the user, because the interrupt vector table is only situated in RAM at the time using the monitor mode. Otherwise it is located in the Flash and therefore not can be changed at program’s run time. Therefore the interrupt vector must be known at the moment of compiling, so that a segment with an appropriate jump instruction can be generated staticly.

7 Error codes

The error codes returned by the driver functions are defined in the file PCMDRV51.INC resp. PCMDRV51.H. The meaning of these codes is listed below:

PCM_SUCCESSFUL (0x00):
Function done successful

PCM_INVALID_CHANNEL (0xFF):
The channel number used for this function call is not valid

PCM_INVALID_AD_CHANNEL (0xFFFF):
invalid channel number for ADC

PCM_INVALID_RESOLUTION (0xFD):
The value for the PWM resolution used for this function call is not valid

If the user defines the symbol **PCM_ENABLE_WARNING**, the internal limitation of parameters to hand over is reported to the calling program by a warning message:

PCM_SUPPRESS_OVERFLOW (0x0100):
Parameter to hand over limited internal

8 The switch PCM_ENABLE_WARNING

The symbol PCM_ENABLE_WARNING is not defined in the default state. That means, the standard declarations for the prototypes of the driver functions are valid, so that they only give errors back as return value. With that all errors can be recognized by invalid parameters because these errors give an error code lower than 0x100. Such an error means the function could not be executed.

If the user defines the symbol PCM_ENABLE_WARNING, the return code of certain functions will be extented to a WORD. So warnings can be recognized by the calling function too. These warnings indicate that a certain parameter is limited internal to his permissible value, f.e. overflow of the output value on analog modules. Warnings come with an return code larger than 0x100. So they assume the extension of the return value to register R6.

If the symbol PCM_ENABLE_WARNING is not defined (default state), by oppression of warnings the return value coming from the driver can be analyzed easier.


```
int   DACOut;

DACOut = (int)(9.99/RES_HIGH);

if ( PCMSetDACChannel(AOUT0, DACOut) )
   {
      // hard error !
   }

   // die Bereichsüberschreitung um ein Digit
   // durch Rundungsfehler wird ignoriert, der
   // Treiber hat den Ausgabewert jedoch intern
   // auf die maximal zulässige Größe begrenzt
```

Note, the internal limitation occurs independend from the declaration of the symbol **PCM_ENABLE_WARNING**. In the program, calling the function, it will be only visible if the symbol is defined !

9 Software Structure

The driver functions for the COMBI Modul-515 enable an easy way to acces all the various on-board units. The description on the different files on the tool disk, shipped with the COMBI Modul-515, is given below:

PCMDRV51.LIB	Library with functions to access the peripheral units on the COMBI Modul-515
PCMTMR51.LIB	Library with functions for the system timer
PCMDRV51.H PCMDRV51.INC	Definition files for library with driver functions, such as Constants, Return Codes, Prototypes
PCMTMR51.H PCMTMR51.INC	Definition files for library with system timer, such as Prototypes
PCMDEMO.C	Demo program for functions of the libraries PCMDRV51.LIB and PCMTMR51.LIB
PCMDEMO.L51	Linker file for Demo program
STARTUP.A51	Startup file for Demo program

Furthermore, a source code licence of the driver library can be purchased at PHYTEC.

10 Using the Demo program

To demonstrate the complete function capability of the program PCMDEMO.C it assumes an external connection from output OUT15 to input IN15:

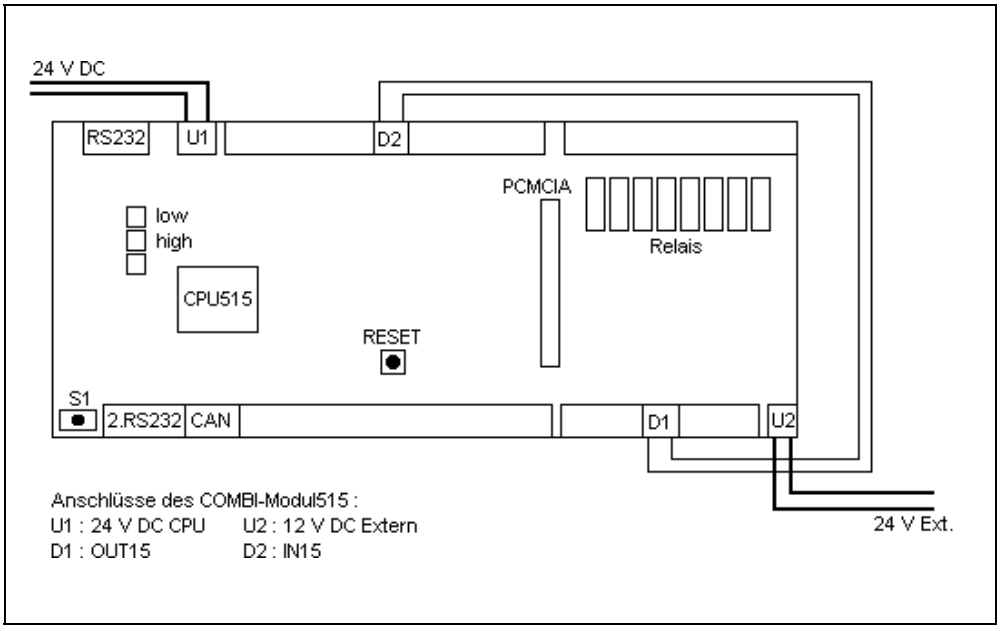

Figure 1: External connections

For a better understanding of the functionality the demo program is commented below:

The Demo program initializes the hardware, the timer0 and releases the global interrupt.
Moving the RUN/STOP switch to position “RUN“ (right) execution of the program starts.
The software shifts the value for the output group 2 one position to the right after each 100 ms.
A clock is generated by the output group 2 and output to OUT15 and available on input CIN15 using the external cable connection.
The counter is preinitialized with a certain value and will be incremented at each falling edge.

Furthermore the program reads the HEX encoding switch and the DIP switch values.
Depending on the position of slide switch 1 and 2 on the DIP switch the user LED‘s will illuminate.
To exit the program, move the the RUN/STOP switch to position “STOP“ (middle)

The output messages for all these actions can be viewed on a host PC with a terminal program. This repuires a RS-232 extension 1:1 cable connected between the serial interface on the COMBI Modul-515 and the COM port on the computer.

11 Appendix A

The driver functions for the COMBI Modul-515 use on-chip components of the C515 as listed below:

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

If using the library for the system timer additional ressources will be taken, as shown below:

- T0

Index

Document:	**Driver for COMBI Modul-515**
Document number:	**L-343e_1, December 1999**

How would you improve this manual?

Did you find any mistakes in this manual? page

Submitted by:

Customer number: _____

Name: _____

Company: _____

Address: _____

_____

Return to:

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

