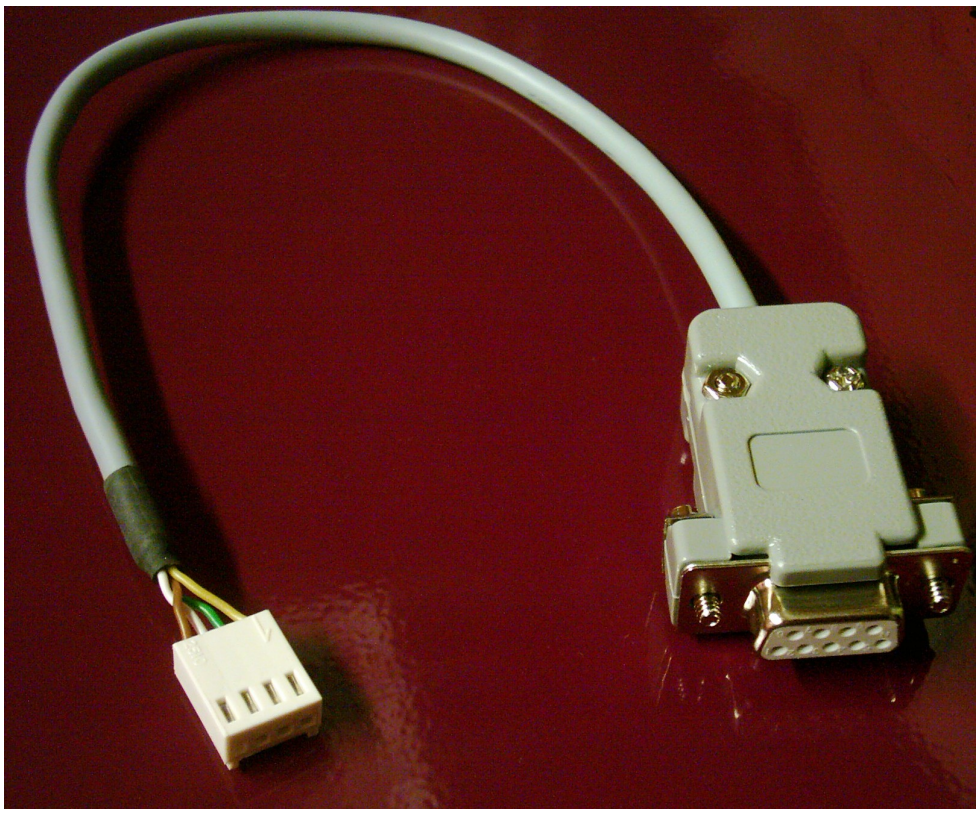
Bedienungsanleitung / Manual

für

iL-Debug_I

Interface für den Debugger iL_Debug



Ing.Büro Stefan Lehmann
Fürstenbergstraße 8a
D-77756 Hausach

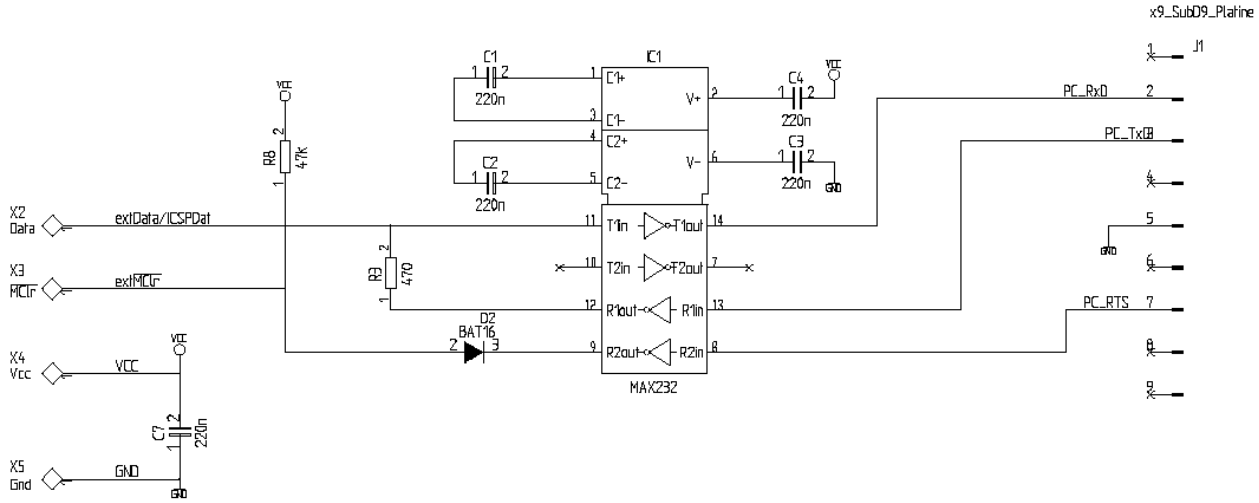
Tel. (07831) 452
Fax (07831) 96428
E-Mail SL@iL-online.de
Internet www.iL-online.de

Allgemeines

Mit der Debug-Funktion des BASIC-Compilers iL-BAS16 wurde die Programmentwicklung wesentlich vereinfacht und beschleunigt. Da der Debugger im Zielsystem läuft, die Bedienung jedoch über die serielle Schnittstelle von einem PC aus erfolgt, wird dieser auch als Remote-Debugger (Fernbedienung) bezeichnet.

Die Funktionen sind u.a. Einzelschritte, Setzen von Haltepunkten (Breakpoints) und das Anzeigen und Verändern von Variableninhalten. Die Generierung des zusätzlichen Debug-Codes kann sehr einfach gesteuert werden, so dass man bestimmte Programmbereiche ausschließen kann. In diesem Fall läuft das Programm an dieser Stelle in Echtzeit ab. Das ist z.B. für serielle Übertragungen wichtig. Der Debug-Vorgang läuft ausschließlich im BASIC-Programm ab. Ein Debuggen in Maschinensprache ist nicht vorgesehen.

Dieses Debug-Interface ist eine vereinfachte Version aus dem Buch „PIC-Microcontroller-Programmierung“. Es sind jedoch alle wichtigen Funktionen aufgebaut. Es fehlen lediglich die LED und der Reset-Taster.



Die Zuordnung der Leitungsfarben zum Anschlussstift:

Pin 1	gelb	Vcc
Pin 2	grün	Daten
Pin 3	weiß	extMCLR
Pin 4	braun	GND

Anschluss des Interfaces

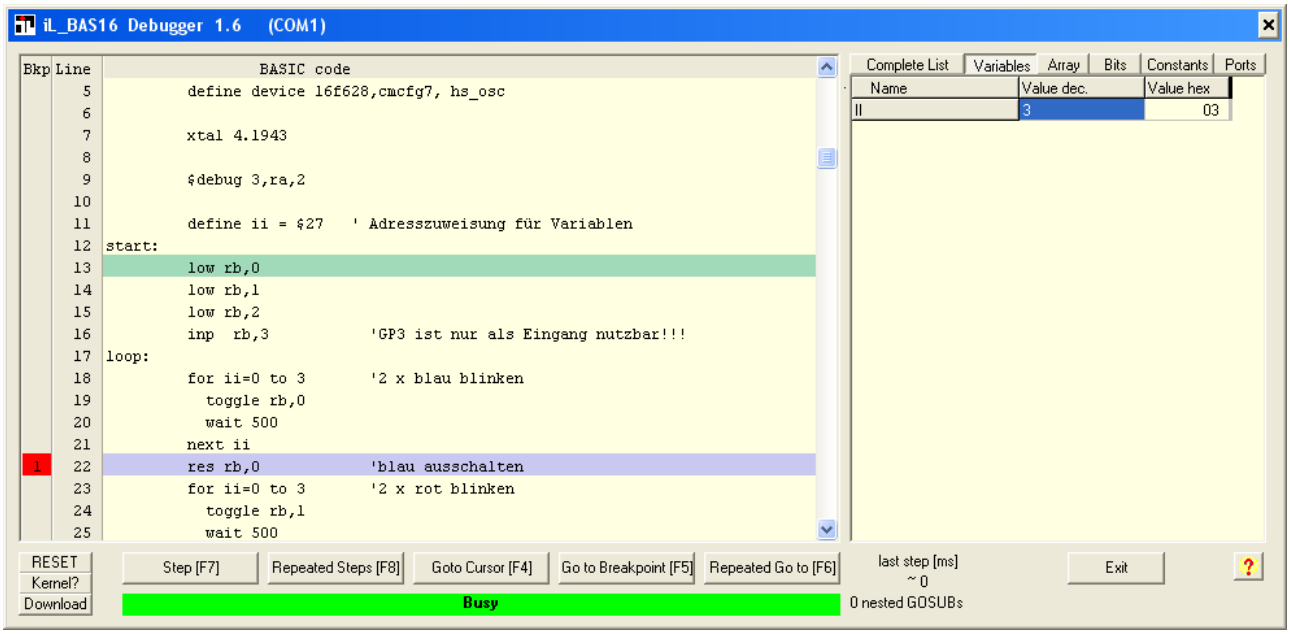
Der Debugger benötigt einen IO-Pin, Masseanschluss und ggf. den Reset. In Anwendungen, bei denen der MCLR-Pin als IO-Pin arbeitet, darf Pin 3 des Interfaces nicht angeschlossen werden. Somit entfällt auch die Möglichkeit, die externe Hardware über den Schaltknopf RESET in der Debug-Software zurückzusetzen.

Bitte beachten Sie, dass es über das Debug-Interface eine galvanische Kopplung zwischen dem PIC und der Zielhardware auf der einen und dem PC auf der anderen Seite gibt.

Arbeiten mit dem Debugger

Der Debugger benötigt zusätzlichen Code innerhalb ihres BASIC-Anwendungsprogramms. Dieser Zusatzcode wird automatisch generiert, sobald Sie im BASIC-Quelltext den Compilerschalter $DEBUG aktivieren. Soll in bestimmten Teilen des Anwendungsprogramms kein Debug-Code generiert werden, lässt sich dies durch $TROFF und $TRON steuern.

Ist der Debug-Code vorhanden, läuft das Anwenderprogramm nur noch in Verbindung mit dem Debugger.



grün: diese Zeile wird als nächstes ausgeführt
blau: Breakpoint; es können bis zu 4 Breakpoints (rot) gesetzt werden

Introduction

The DEBUG function of the PIC-BASIC-Compiler iL-BAS16 speeds-up and simplifies the developing of applications dramaticly. This is a so called remote debugger because the debugger runs on the target hardware, while it is controlled via the COM port by a Windows program.

The functions of the debugger are single steps, breakpoints, show and modifying contents of variables etc. Inserting and supressing of the necessary debug code is easy to control. Transfering datas to the computer needs a lot of time. This slows down the execution speed. But sometimes it is necessary to run certain parts of the application in full speed (real time) e.g. serial communication protocols. You can get full speed if you use $TROFF and $TRON to switch off and on the debug code generation. Debugging is only available in BASIC and not in machine language.

This debug interface is a simplified version described in the book „PIC-Microcontroller-Programmierung“. But all important functions are built in except the led and reset button.

Assignment of color and connector pins:

Pin 1	yellow	Vcc
Pin 2	green	Data
Pin 3	white	extMCLR
Pin 4	brown	GND

Connecting the interface

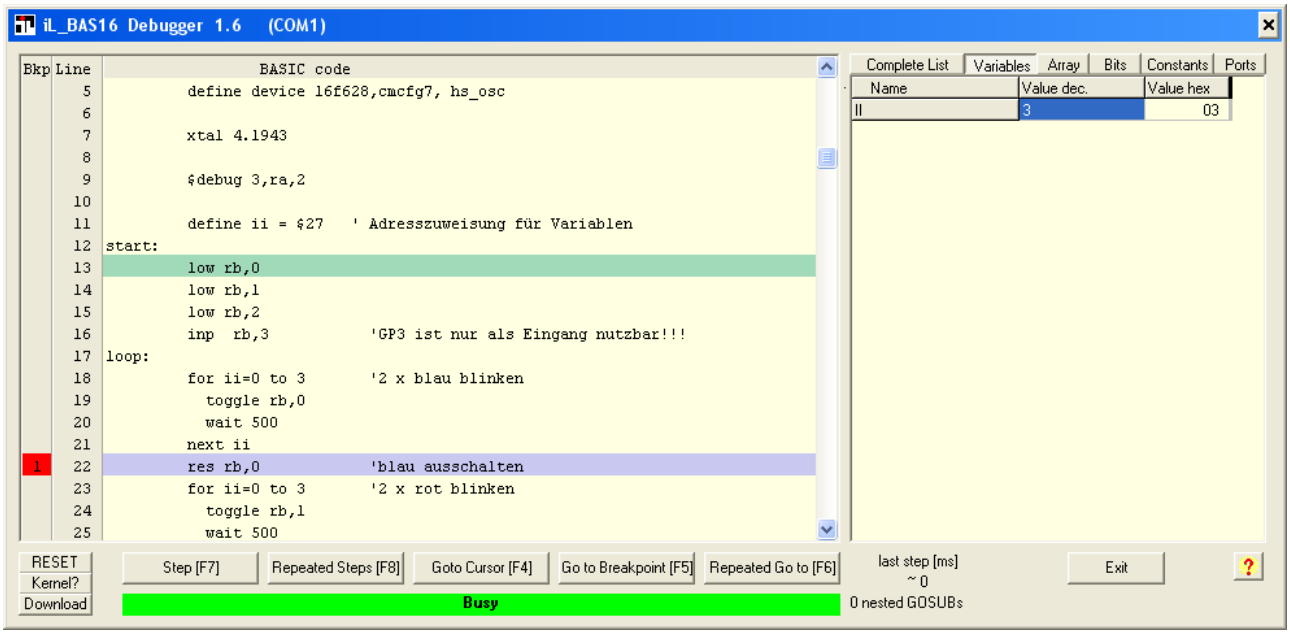
The debugger only needs one single io pin, ground and, if possible, the reset pin. Within the application where the reset pin is used as a general io pin, this connection is not allowed and pin 3 of the interface connector must be unplugged. In this case there is no possibility to reset the target via debugger.

Please note that your computer and the target hardware are connect to the same signal ground.

Working with the debugger

The debugger needs additional code for working. This code is automaticly inserted in your application program but is not visible. Use compiler switch $DEBUG to insert debug code and $TROFF and $TRON to inhibit debug code on parts of your basic program.

If $DEBUG is active, your application runs only under the control of the debugger.



green: this line is the next to be executed
blue: breakpoint; you can set up to 4 breakpoints (red)