C8051F0xx Mikrocontroller und BASCOM-8051

Dr. Claus Kühnel

Obwohl immer wieder neue Mikrocontroller Konzepte mit speziellen Merkmalen am Markt erscheinen, werten die technologischen Fortschritte auch und gerade deshalb etablierte Konzepte auf. So erhält einer der ältesten Mikrocontroller am Markt einer immer wieder neues Gesicht und muss sich nicht hinter den neuen Konzepten verstecken.

Für den Entwickler selbst ist diese Tatsache nicht ohne Vorteil, denn es besteht ein großer Fundus an Applikationen, Bibliotheken, ausgereiften Tools und gesammelten Erfahrungen.

In diesem Beitrag werden die Mikrocontroller C8051F0xx von CYGNAL Integrated Products, Inc. [www.cygnal.com] und deren Programmierung mit BASCOM-8051 betrachtet.

1 Mikrocontroller C8051F0xx

Die Mikrocontroller C8051F0xx besitzen neben einem schnellen 8051-Kern eine sehr leistungsfähige analoge und digitale Peripherie auf dem Chip.

Für den Aufbau eines Messsystems bedarf es deshalb zu Anpassungs- und Filterzwecken nur noch weniger zusätzlicher Komponenten. Abbildung 1 zeigt die Komponenten der drei genannten Funktionsgruppen.

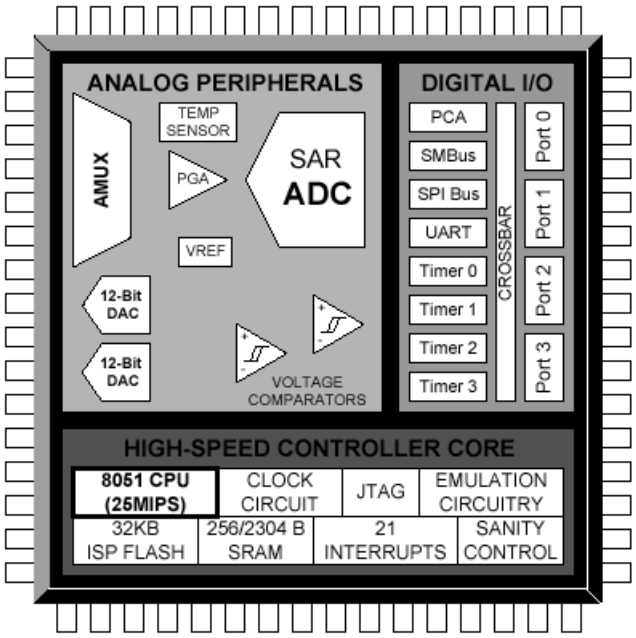


Abbildung 1 C8051F0xx Blockdiagramm

Stellvertretend für die Mikrocontroller dieser Familie soll hier der C8051F000 betrachtet werden, der nur beim RAM von einigen anderen Bausteinen überboten wird.

Die analoge Funktionseinheit weist einen neunkanaligen 12-Bit AD-Umsetzer mit einem Datendurchsatz vom 100000 Umsetzungen/sec (100 ksps) auf. Ein Multiplexer kann die acht analogen Eingänge sowohl massebezogen (single-ended) als auch differentiell an den AD-Umsetzer führen. Ein programmierbarer Verstärker dient der Anpassung der zu verarbeitenden Pegel an den Aussteuerbereich des AD-Umsetzers. Eine interne Referenzspannung mit einem Temperaturkoeffizienten von 15 ppm/°C lässt in vielen Fällen den Verzicht auf eine externe Referenzspannung zu. Ein integrierter Temperatursensor kann zur Korrektur eines möglichen Temperaturgangs der Messeinrichtung herangezogen werden.

Zur Ausgabe von analogen Spannungswerten stehen zwei 12-Bit DA-Umsetzer mit einer Einschwingzeit von 10 µs zur Verfügung.

Zwei Komparatoren mit je 16 programmierbaren Hysteresewerten können zur Auslösung von Interrupts oder einem Reset verwendet werden.

Die digitale Peripherie ist gleichermaßen umfangreich. Zum Datenaustausch stehen I^2C- und SPI-Bus zur Verfügung. Ein Hardware-UART unterstützt die serielle Kommunikation gemäss RS-232.

Vier Timer und ein programmierbares Counter/Timer Array mit fünf Capture/Compare Modulen ermöglichen komplexe Timer- bzw. Counterapplikationen.

Die Verbindung der umfangreiche Peripherie mit den I/O Pins übernimmt ein Kreuzschienenverteiler (Crossbar Switch).

Abbildung 2 zeigt das Blockschaltbild des C8051F000 etwas detaillierter. Die Funktion des Crossbar Switches ist deutlich erkennbar.

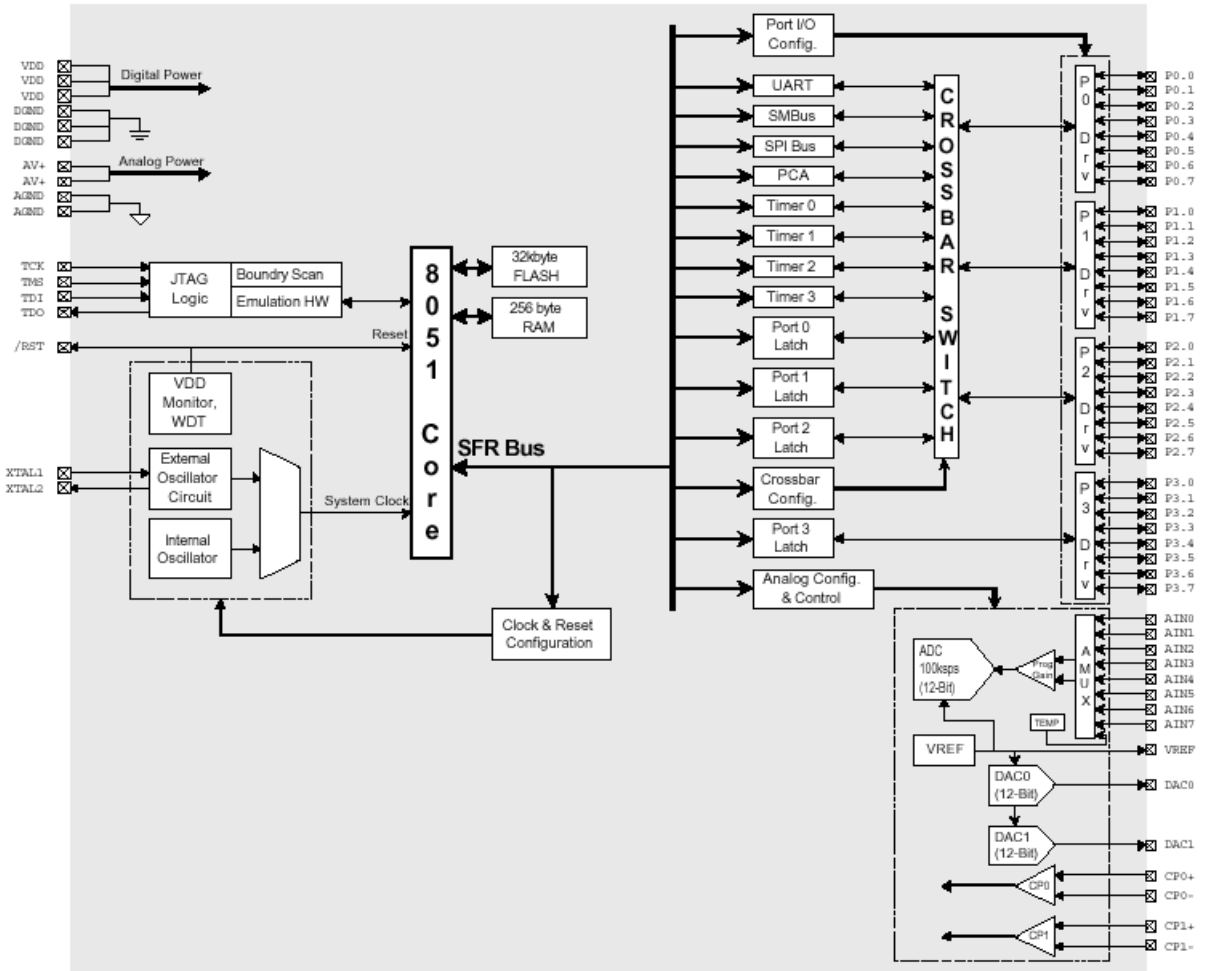


Abbildung 2 C8051F000 Blockschaltbild

Die Ausstattungsmerkmale aller Mitglieder der Mikrocontrollerfamilie C8051Fxxx sind der folgenden Tabelle zu entnehmen.

Cygnal Integrated Products - Product Selection Guide

Part Number	MIPS (peak)	Flash Memory (bytes)	RAM (bytes)	Ext Memory I/F	SMBus / I2C	SPI	UART	Timers (16-bit)	Program Counter Array	Internal Oscillator +/-%	Digital Port I/O Pins	ADC Resolution (bits)	ADC Speed (ksps)	ADC Inputs	Voltage Reference	Temperature Sensor	DAC Resolution (bits)	Voltage Comparators	Package	Price $(1,000 pc.)	Availability
C8051F000	20	32k	256	-	✔	✔	1	4	✔	20	32	12	100	8	✔	✔	12	2	TQ64	$13.00	Now
C8051F001	20	32k	256	-	✔	✔	1	4	✔	20	16	12	100	8	✔	✔	12	2	TQ48	$12.48	Now
C8051F002	20	32k	256	-	✔	✔	1	4	✔	20	8	12	100	4	✔	✔	12	1	LQ32	$12.13	Now
C8051F005	25	32k	2304	-	✔	✔	1	4	✔	20	32	12	100	8	✔	✔	12	2	TQ64	$14.91	Now
C8051F006	25	32k	2304	-	✔	✔	1	4	✔	20	16	12	100	8	✔	✔	12	2	TQ48	$14.39	Now
C8051F007	25	32k	2304	-	✔	✔	1	4	✔	20	8	12	100	4	✔	✔	12	1	LQ32	$14.21	Now
C8051F010	20	32k	256	-	✔	✔	1	4	✔	20	32	10	100	8	✔	✔	12	2	TQ64	$11.27	Now
C8051F011	20	32k	256	-	✔	✔	1	4	✔	20	16	10	100	8	✔	✔	12	2	TQ48	$10.75	Now
C8051F012	20	32k	256	-	✔	✔	1	4	✔	20	8	10	100	4	✔	✔	12	1	LQ32	$10.23	Now
C8051F015	25	32k	2304	-	✔	✔	1	4	✔	20	32	10	100	8	✔	✔	12	2	TQ64	$13.17	Now
C8051F016	25	32k	2304	-	✔	✔	1	4	✔	20	16	10	100	8	✔	✔	12	2	TQ48	$12.65	Now
C8051F017	25	32k	2304	-	✔	✔	1	4	✔	20	8	10	100	4	✔	✔	12	1	LQ32	$12.13	Now
C8051F020	25	64k	4352	✔	✔	✔	2	5	✔	20	64	12	100	8	✔	✔	12	2	TQ100	$17.23	Now
C8051F021	25	64k	4352	✔	✔	✔	2	5	✔	20	32	12	100	8	✔	✔	12	2	TQ64	$16.33	Now
C8051F022	25	64k	4352	✔	✔	✔	2	5	✔	20	64	10	100	8	✔	✔	12	2	TQ100	$15.44	Now
C8051F023	25	64k	4352	✔	✔	✔	2	5	✔	20	32	10	100	8	✔	✔	12	2	TQ64	$14.59	Now
C8051F206	25	8k	1280	-	-	✔	1	3	-	20	32	12	100	32	-	-	-	2	TQ48	$8.98	Now
C8051F220	25	8k	256	-	-	✔	1	3	-	20	32	8	100	32	-	-	-	2	TQ48	$5.46	Now
C8051F221	25	8k	256	-	-	✔	1	3	-	20	22	8	100	22	-	-	-	2	LQ32	$5.18	Now
C8051F226	25	8k	1280	-	-	✔	1	3	-	20	32	8	100	32	-	-	-	2	TQ48	$6.50	Now
C8051F230	25	8k	256	-	-	✔	1	3	-	20	32	-	-	-	-	-	-	2	TQ48	$4.59	Now
C8051F231	25	8k	256	-	-	✔	1	3	-	20	22	-	-	-	-	-	-	2	LQ32	$4.42	Now
C8051F236	25	8k	1280	-	-	✔	1	3	-	20	32	-	-	-	-	-	-	2	TQ48	$5.63	Now
C8051F300	25	8k	256	-	✔	-	1	3	✔	2	8	8	500	8	-	✔	-	1	MLP11	$5.01	Now
C8051F301	25	8k	256	-	✔	-	1	3	✔	2	8	-	-	-	-	-	-	1	MLP11	$4.70	Oct 01
C8051F302	25	8k	256	-	✔	-	1	3	✔	20	8	8	500	8	-	✔	-	1	MLP11	$4.14	Oct 01
C8051F303	25	8k	256	-	✔	-	1	3	✔	20	8	-	-	-	-	-	-	1	MLP11	$3.83	Oct 01

Der 8051er Kern greift über Special Function Register (SFR) auf die eben erläuterte Peripherie zu. Die 32 KB Flash Memory sind In-System programmierbar. Das RAM umfasst 256 Bytes.

Bei einer Taktfrequenz von 20 MHz erreicht der Prozessor einen Durchsatz von 20 MIPS. Der Takt kann im Bereich von 2 bis 16 MHz intern erzeugt werden. Für eine externe Takterzeugung bietet der C8051F000 komfortable Anpassungsmöglichkeiten. Zwischen interner und externer Takterzeugung kann im laufenden Betrieb umgeschaltet werden.

Ein sicherer Betrieb des Mikrocontrollers wird durch die interne Betriebsspannungsüberwachung und einen Watchdog sichergestellt.

Versorgt wird der C8051F000 mit einer Betriebsspannung zwischen 2,7 und 3,6 V, wobei die typische Stromaufnahme bei einem Takt von 20 MHz bei nur 10 mA liegt.

Die On-Chip JTAG und Emulation Logik erlaubt In-Circuit Emulation über das 4-Pin JTAG Interface nach IEEE 1149.1. CYGNAL's Emulation System unterstützt das Auslesen und die Modifikation des Speichers und der Register, das setzen von Break- und Watchpoints, das Monitoring des Stacks sowie Single-Step Betrieb. Eine sogenannte Emulation Cartridge adaptiert das JTAG Interface und kommuniziert mit dem PC über RS-232.

Mit den letztgenannten Hardwarevoraussetzungen ergeben sich komfortable Möglichkeiten für die erforderliche Entwicklungsumgebung.

2 C8051F0XX Entwicklungskit

Von CYGNAL werden komplette Entwicklungskits angeboten, die ein problemloses Kennenlernen des Mikrocontrollers C8051F0XX sowie Programmentwicklung und Debugging ermöglichen. Abbildung 3 zeigt das komplette C8051F0XX Entwicklungskit.

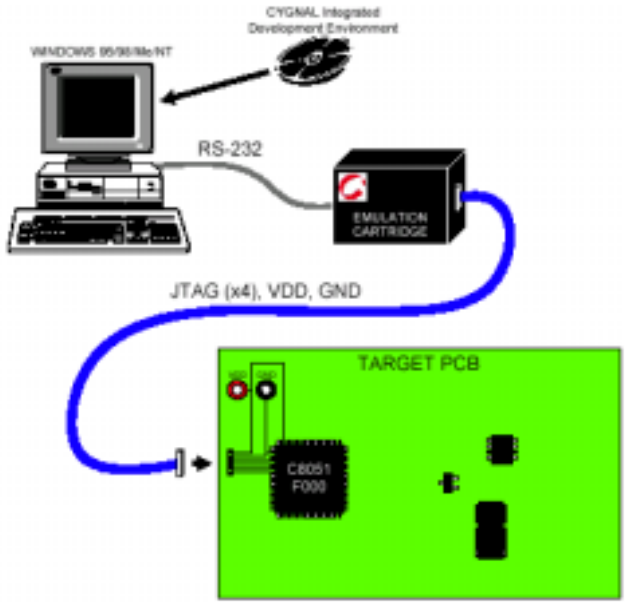


Abbildung 3 CYGNAL's C8051F0XX Entwicklungskit

Die Emulation Cartridge verbindet die Entwicklungsumgebung auf dem PC mit dem Targetboard über dessen JTAG Interface. Für Programmdownload und Debugging werden also keine weiteren Ressourcen des Mikrocontrollers blockiert. Abbildung 4 zeigt das C8051F000 Targetboard.

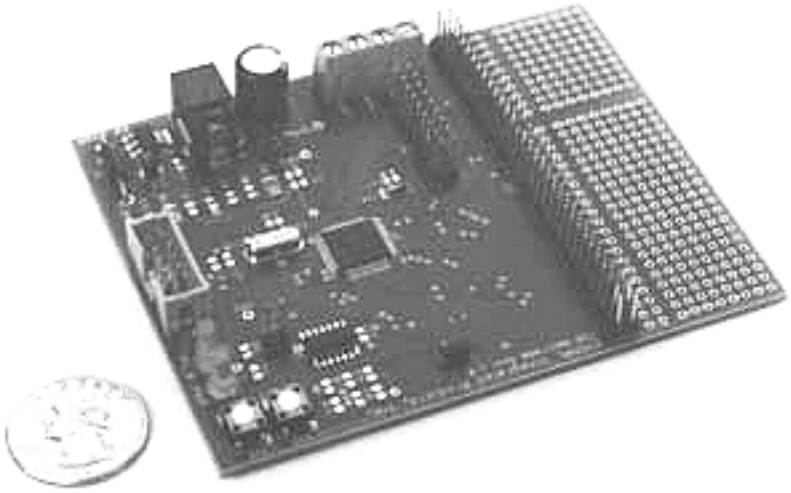


Abbildung 4 CYGNAL C8051F000 Targetboard

Die Software der CYGNAL Entwicklungsumgebung präsentiert sich gemäss Abbildung 5 und ist aber weitgehend anpassbar.

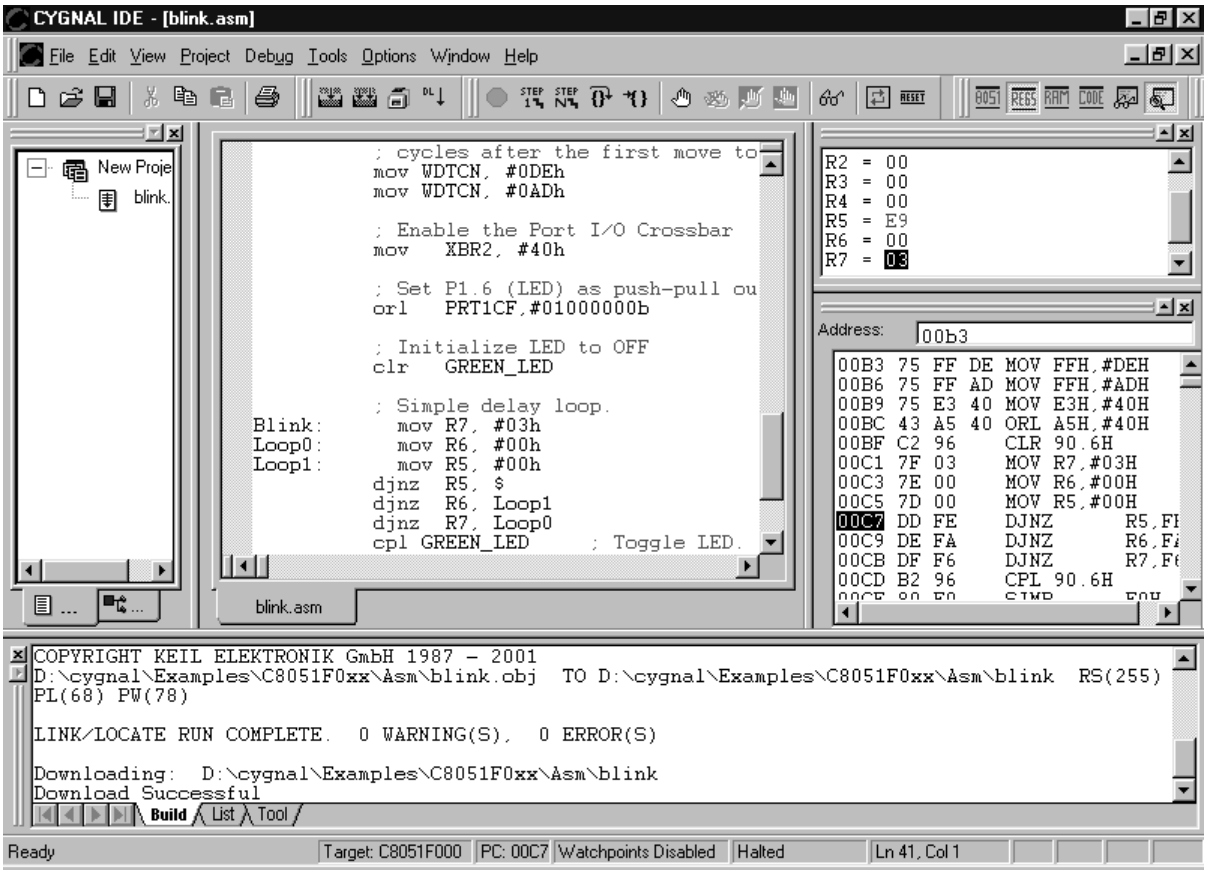


Abbildung 5 CYGNAL Entwicklungsumgebung

Es können in dieser Entwicklungsumgebung C- und Assembler-Programme bearbeitet werden. Als C-Compiler kommt der in der 8051-Welt sehr weit verbreitete Compiler von Keil zum Einsatz. Wir wollen uns hier wegen des Bezugs zu BASCOM-8051 mit dem Assembler begnügen.

In Abbildung 5 ist das Programm BLINK.ASM geladen und erfolgreich ins Targetboard geladen worden.

Über die Werkzeugleiste können die gewünschten Funktionen selektiert werden. Abbildung 6 zeigt die wichtigsten Funktionen.

Compilieren/Assemblieren, Verbindung zum Target, Download

Debugging (Single Step, Breakpoints etc.)

Anzeigeoptionen

Abbildung 6 Werkzeugleiste der Entwicklungsumgebung

Diese Funktionen sind in fast allen Entwicklungsumgebungen in vergleichbarer Form zu finden.

Eine sehr wichtige Funktion der Entwicklungsumgebung ist der sogenannte Configuration Wizard, der durch das Ausfüllen eines Configuration Templates ein große Hilfe bei der Konfiguration der umfangreichen On-Chip Peripherie der C8051Fxxx Mikrocontroller darstellt. Listing 1 zeigt ein solches Configuration Template, welches nach der Konfiguration dann zum eigentlichen Programm erweitert werden kann.

```
;------------------------------------------------------------------
; CYGNAL Integrated Products
;
; Assembly Code Configuration Tool: F000 INITIALIZATION/CONFIGURATION CODE
;------------------------------------------------------------------
; This file is read only. To insert the code into your
; application, simply cut and paste or use the "Save As"
; command in the file menu to save the file in your project
; directory.
;------------------------------------------------------------------

;------------------------------------------------------------------
; GLOBAL VARIABLES AND ASSIGNMENTS
;------------------------------------------------------------------

$INCLUDE(C8051F000.INC)  ; Register definition file.

;------------------------------------------------------------------
; INTERRUPT VECTOR CODE
```

```
;-----------------------------------------------------------------

     org 00h
     LJMP Config
; Place jump to reset handler and interrupt service routines here.

Config:
     org 0B3h            ; End of Interrupt Vector space.

;-----------------------------------------------------------------
; Watchdog Timer Configuration
;
; WDTCN.[7:0]: WDT Control
;   Writing 0xA5 enables and reloads the WDT.
;   Writing 0xDE followed within 4 clocks by 0xAD disables the WDT
;   Writing 0xFF locks out disable feature.
;
; WDTCN.[2:0]: WDT timer interval bits
;   NOTE! When writing interval bits, bit 7 must be a 0.
;
;  Bit 2 | Bit 1 | Bit 0
;------------------------
;    1   |   1   |   1      Timeout interval = 1048576 x Tsysclk
;    1   |   1   |   0      Timeout interval =  262144 x Tsysclk
;    1   |   0   |   1      Timeout interval =   65636 x Tsysclk
;    1   |   0   |   0      Timeout interval =   16384 x Tsysclk
;    0   |   1   |   1      Timeout interval =    4096 x Tsysclk
;    0   |   1   |   0      Timeout interval =    1024 x Tsysclk
;    0   |   0   |   1      Timeout interval =     256 x Tsysclk
;    0   |   0   |   0      Timeout interval =      64 x Tsysclk
;------------------------

    mov WDTCN, #007h     ; Watchdog Timer Control Register
    mov WDTCN, #0DEh     ; Disable WDT
    mov WDTCN, #0ADh


;-----------------------------------------------------------------
; CROSSBAR REGISTER CONFIGURATION
;
; NOTE: The crossbar register should be configured before any
; of the digital peripherals are enabled. The pinout of the
; device is dependent on the crossbar configuration so caution
; must be exercised when modifying the contents of the XBR0,
; XBR1, and XBR2 registers. For detailed information on
; Crossbar Decoder Configuration, refer to Application Note
; AN001, "Configuring the Port I/O Crossbar Decoder".
;-----------------------------------------------------------------

; Configure the XBRn Registers

     mov XBR0, #000h     ; XBAR0: Initial Reset Value
     mov XBR1, #000h     ; XBAR1: Initial Reset Value
     mov XBR2, #000h     ; XBAR2: Initial Reset Value

; Select Pin I/0

; NOTE: Some peripheral I/O pins can function as either inputs or
; outputs, depending on the configuration of the peripheral. By default,
; the configuration utility will configure these I/O pins as push-pull
; outputs.  If the I/O direction changes to input once the peripheral is
; configured, the peripheral hardware will override the PRTnCF register
; setting and change the pin configuration to input.

                         ; Port configuration (1 = Push Pull Output)
     mov PRT0CF, #000h   ; Output configuration for P0
```

```
        mov PRT1CF, #040h   ; Output configuration for P1
        mov PRT2CF, #000h   ; Output configuration for P2
        mov PRT3CF, #000h   ; Output configuration for P3

    ; View port pinout

    ; The current Crossbar configuration results in the
    ; following port pinout assignment:

        ; Port 0
        ; P0.0 = Gp I / O
        ; P0.1 = Gp I / O
        ; P0.2 = Gp I / O
        ; P0.3 = Gp I / O
        ; P0.4 = Gp I / O
        ; P0.5 = Gp I / O
        ; P0.6 = Gp I / O
        ; P0.7 = Gp I / O

        ; Port 1
        ; P1.0 = Gp I / O
        ; P1.1 = Gp I / O
        ; P1.2 = Gp I / O
        ; P1.3 = Gp I / O
        ; P1.4 = Gp I / O
        ; P1.5 = Gp I / O
        ; P1.6 = Gp I / O          (Push-Pull Output)
        ; P1.7 = Gp I / O          (Open-Drain Output/Input)

        ; Port 2
        ; P2.0 = Gp I / O
        ; P2.1 = Gp I / O
        ; P2.2 = Gp I / O
        ; P2.3 = Gp I / O
        ; P2.4 = Gp I / O
        ; P2.5 = Gp I / O
        ; P2.6 = Gp I / O
        ; P2.7 = Gp I / O

        ; Port 3
        ; P3.0 = Gp I / O
        ; P3.1 = Gp I / O
        ; P3.2 = Gp I / O
        ; P3.3 = Gp I / O
        ; P3.4 = Gp I / O
        ; P3.5 = Gp I / O
        ; P3.6 = Gp I / O
        ; P3.7 = Gp I / O

    ;-----------------------------------------------------------------
    ; Comparators Register Configuration
    ;
    ; Bit 7  | Bit 6  | Bit 5  | Bit 4  | Bit 3 | Bit 2 | Bit 1 | Bit 0
    ;-----------------------------------------------------------------
    ;  R/W   |  R     | R/W    | R/W    | R/W   | R/W   | R/W   | R/W
    ;-----------------------------------------------------------------
    ; Enable | Output | Rising | Falling|  Positive     |  Negative
    ;        | State  | Edge   | Edge   |  Hysterisis   |  Hysterisis
    ;        | Flag   | Int.   | Int.   |  00: Disable  |  00: Disable
    ;        |        | Flag   | Flag   |  01:  5mV     |  01:  5mV
    ;        |        |        |        |  10: 10mV     |  10: 10mV
    ;        |        |        |        |  11: 20mV     |  11: 20mV
    ; ----------------------------------------------------------------

        mov CPT0CN, #000h     ; Comparator 0 Control Register
        mov CPT1CN, #000h     ; Comparator 1 Control Register
```

```
     ;Comp1 marker

;-----------------------------------------------------------------
; Oscillator Configuration
;-----------------------------------------------------------------

     mov OSCXCN, #030h   ; External Oscillator Control Register
     mov OSCICN, #004h   ; Internal Oscillator Control Register


;-----------------------------------------------------------------
; Reference Control Register Configuration
;-----------------------------------------------------------------

     mov REF0CN, #000h   ; Reference Control Register

;-----------------------------------------------------------------
; SPI Configuration
;-----------------------------------------------------------------

     mov SPI0CN, #000h   ; SPI Control Register
     mov SPI0CFG, #000h  ; SPI Configuration Register
     mov SPI0CKR, #000h  ; SPI Clock Rate Register

;-----------------------------------------------------------------
; DAC Configuration
;-----------------------------------------------------------------

     mov DAC0CN, #000h   ; DAC0 Control Register
     mov DAC0L, #000h    ; DAC0 Low Byte Register
     mov DAC0H, #000h    ; DAC0 High Byte Register

     mov DAC1CN, #000h   ; DAC1 Control Register
     mov DAC1L, #000h    ; DAC1 Low Byte Register
     mov DAC1H, #000h    ; DAC1 High Byte Register

;-----------------------------------------------------------------
; UART Configuration
;-----------------------------------------------------------------

     mov SCON, #000h     ; Serial Port Control Register
     mov PCON, #000h     ; Power Control Register


;-----------------------------------------------------------------
; SMBus Configuration
;-----------------------------------------------------------------

     mov SMB0CN, #000h   ; SMBus Control Register
     mov SMB0ADR, #000h  ; SMBus Address Register
     mov SMB0CR, #000h   ; SMBus Clock Rate Register


;-----------------------------------------------------------------
; PCA Configuration
;-----------------------------------------------------------------

     mov PCA0MD, #000h   ; PCA Mode Register
     mov PCA0CN, #000h   ; PCA Control Register
     mov PCA0L, #000h    ; PCA Counter/Timer Low Byte
     mov PCA0H, #000h    ; PCA Counter/Timer High Byte

     ;Module 0
     mov PCA0CPM0, #000h ; PCA Capture/Compare Register 0
     mov PCA0CPL0, #000h ; PCA Counter/Timer Low Byte
```

```
      mov PCA0CPH0, #000h ; PCA Counter/Timer High Byte

      ;Module 1
      mov PCA0CPM1, #000h ; PCA Capture/Compare Register 1
      mov PCA0CPL1, #000h ; PCA Counter/Timer Low Byte
      mov PCA0CPH1, #000h ; PCA Counter/Timer High Byte

      ;Module 2
      mov PCA0CPM2, #000h ; PCA Capture/Compare Register 2
      mov PCA0CPL2, #000h ; PCA Counter/Timer Low Byte
      mov PCA0CPH2, #000h ; PCA Counter/Timer High Byte

      ;Module 3
      mov PCA0CPM3, #000h ; PCA Capture/Compare Register 3
      mov PCA0CPL3, #000h ; PCA Counter/Timer Low Byte
      mov PCA0CPH3, #000h ; PCA Counter/Timer High Byte

      ;Module 4
      mov PCA0CPM4, #000h ; PCA Capture/Compare Register 4
      mov PCA0CPL4, #000h ; PCA Counter/Timer Low Byte
      mov PCA0CPH4, #000h ; PCA Counter/Timer High Byte

;-----------------------------------------------------------------
; ADC Configuration
;-----------------------------------------------------------------

      mov AMX0CF, #060h   ; AMUX Configuration Register
      mov AMX0SL, #000h   ; AMUX Channel Select Register
      mov ADC0CF, #000h   ; ADC Configuraion Register
      mov ADC0CN, #000h   ; ADC Control Register

      mov ADC0LTH, #000h  ; ADC Less-Than High Byte Register
      mov ADC0LTL, #000h  ; ADC Less-Than Low Byte Register
      mov ADC0GTH, #0FFh  ; ADC Greater-Than High Byte Register
      mov ADC0GTL, #0FFh  ; ADC Greater-Than Low Byte Register

;-----------------------------------------------------------------
; Timer Configuration
;-----------------------------------------------------------------

      mov CKCON, #000h    ; Clock Control Register
      mov TH0, #000h      ; Timer 0 High Byte
      mov TL0, #000h      ; Timer 0 Low Byte
      mov TH1, #000h      ; Timer 1 High Byte
      mov TL1, #000h      ; Timer 1 Low Byte
      mov TMOD, #000h     ; Timer Mode Register
      mov TCON, #000h     ; Timer Control Register

      mov RCAP2H, #000h   ; Timer 2 Capture Register High Byte
      mov RCAP2L, #000h   ; Timer 2 Capture Register Low Byte
      mov TH2, #000h      ; Timer 2 High Byte
      mov TL2, #000h      ; Timer 2 Low Byte
      mov T2CON, #000h    ; Timer 2 Control Register

      mov TMR3RLL, #000h  ; Timer 3 Reload Register Low Byte
      mov TMR3RLH, #000h  ; Timer 3 Reload Register High Byte
      mov TMR3H, #000h    ; Timer 3 High Byte
      mov TMR3L, #000h    ; Timer 3 Low Byte
      mov TMR3CN, #000h   ; Timer 3 Control Register

;-----------------------------------------------------------------
; Reset Source Configuration
;
; Bit 7  | Bit 6  | Bit 5  | Bit 4  | Bit 3 | Bit 2 | Bit 1 | Bit 0
;-------------------------------------------------------------------
;    R   |   R/W  |  R/W   |  R/W   |   R   |   R   |  R/W  |  R
```

```
;-------------------------------------------------------------------
;  JTAG  |Convert | Comp.0 | S/W    | WDT   | Miss. | POR   | HW
; Reset  |Start   | Reset/ | Reset  | Reset | Clock | Force | Pin
; Flag   |Reset/  | Enable | Force  | Flag  | Detect| &     | Reset
;        |Enable  | Flag   | &      |       | Flag  | Flag  | Flag
;        |Flag    |        | Flag   |       |       |       |
;-------------------------------------------------------------------
; NOTE! : Comparator 0 must be enabled before it is enabled as a
; reset source.
;
; NOTE! : External CNVSTR must be enalbed through the crossbar, and
; the crossbar enabled prior to enabling CNVSTR as a reset source
;-------------------------------------------------------------------

     mov RSTSRC, #000h   ; Reset Source Register

;-------------------------------------------------------------------
; Interrupt Configuration
;-------------------------------------------------------------------

     mov IE, #000h       ;Interrupt Enable
     mov IP, #000h       ;Interrupt Priority
     mov EIE1, #000h     ;Extended Interrupt Enable 1
     mov EIE2, #000h     ;Extended Interrupt Enable 2
     mov EIP1, #000h     ;Extended Interrupt Priority 1
     mov EIP2, #000h     ;Extended Interrupt Priority 2

; other initialization code here...

;-------------------------------------------------------------------
; MAIN PROGRAM CODE
;-------------------------------------------------------------------

Main:

     ; main code routines here...

END
```

Listing 1 Configuration Template

Wie der Configuration Wizard Unterstützung bei der Konfiguration der On-Chip Peripherie bietet, zeigen die folgenden Screenshots.

Abbildung 7 zeigt den Aufruf des Configuration Wizards

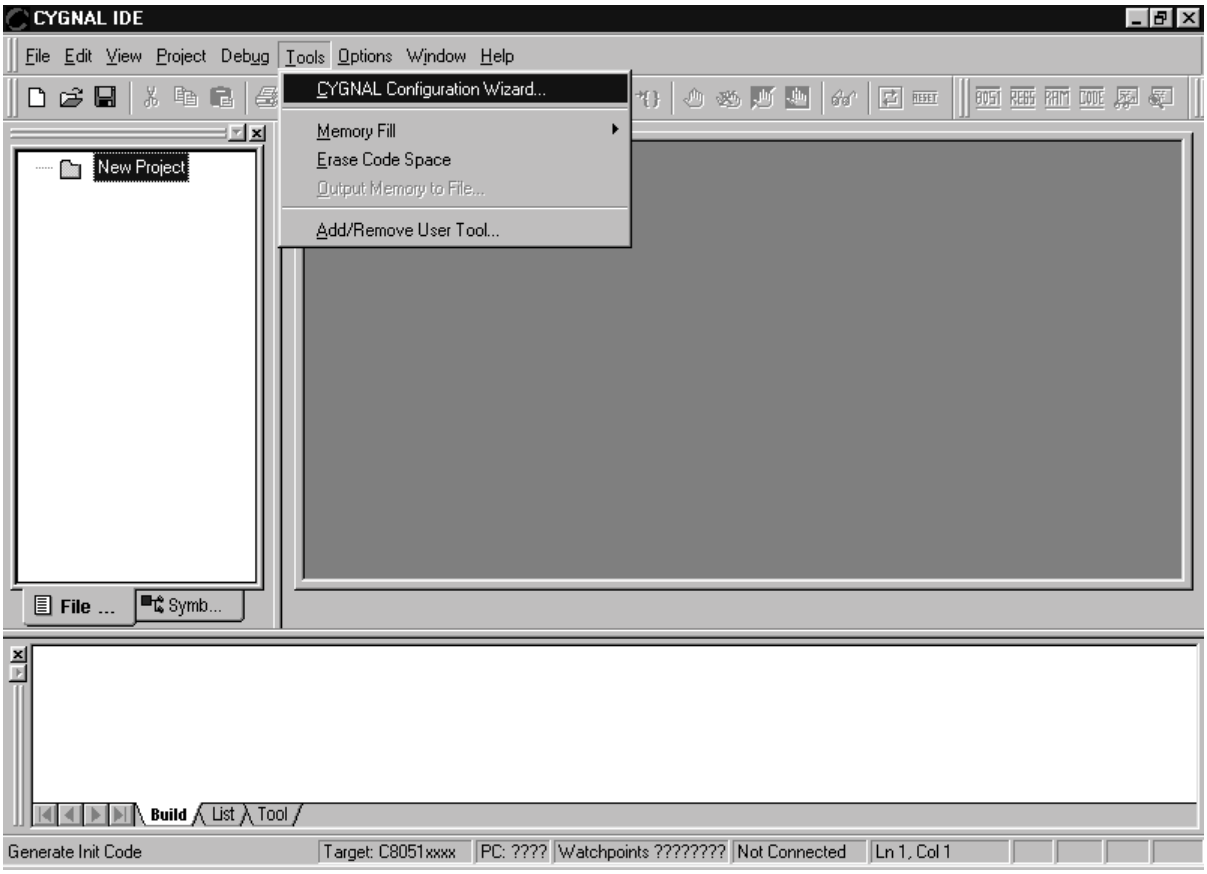


Abbildung 7 Aufruf des Configuration Wizards

Vor der eigentlichen Konfiguration sind der Mikrocontroller selbst sowie C- bzw. Assembler-Template auszuwählen (Abbildung 8).

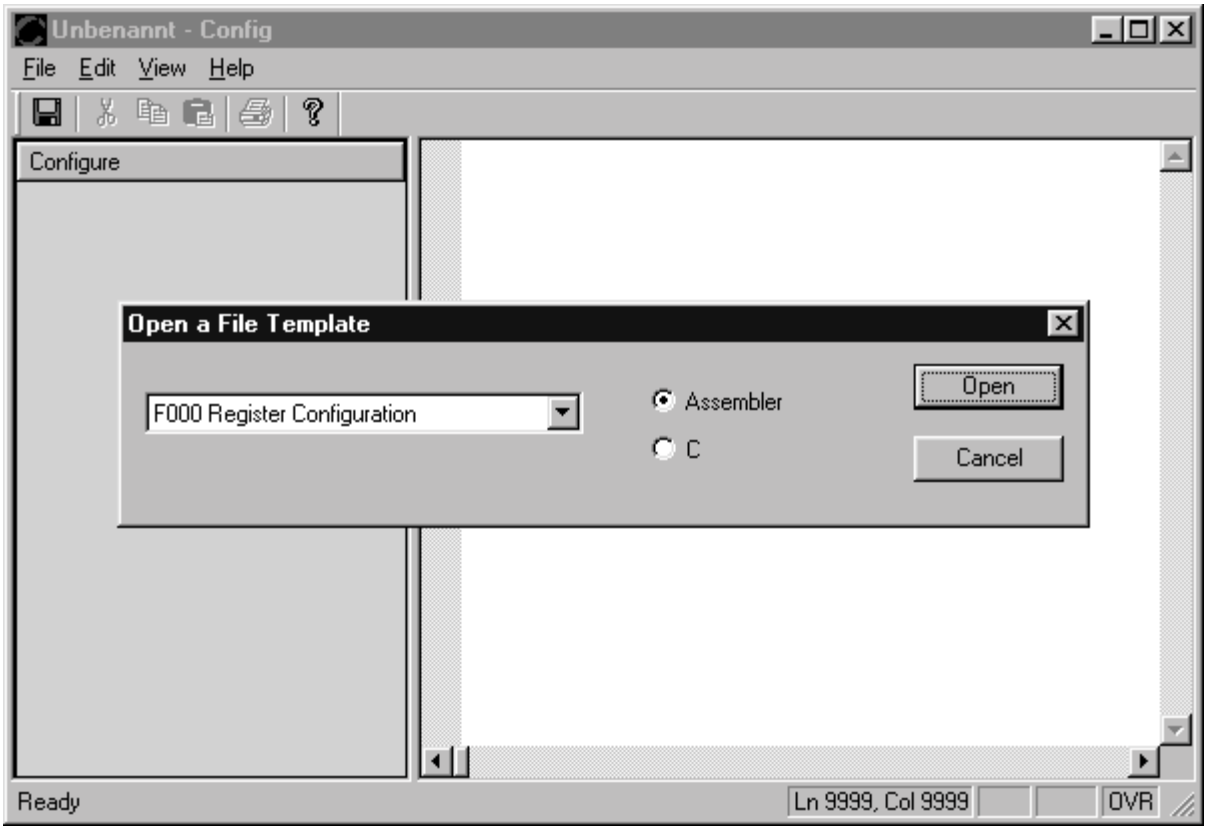


Abbildung 8 Auswahl Assembler Template für C8051F000

Hat man diese Auswahl getroffen, dann präsentiert sich die gesamte zu konfigurierende Peripherie im Fenster „Config“ (Abbildung 9).

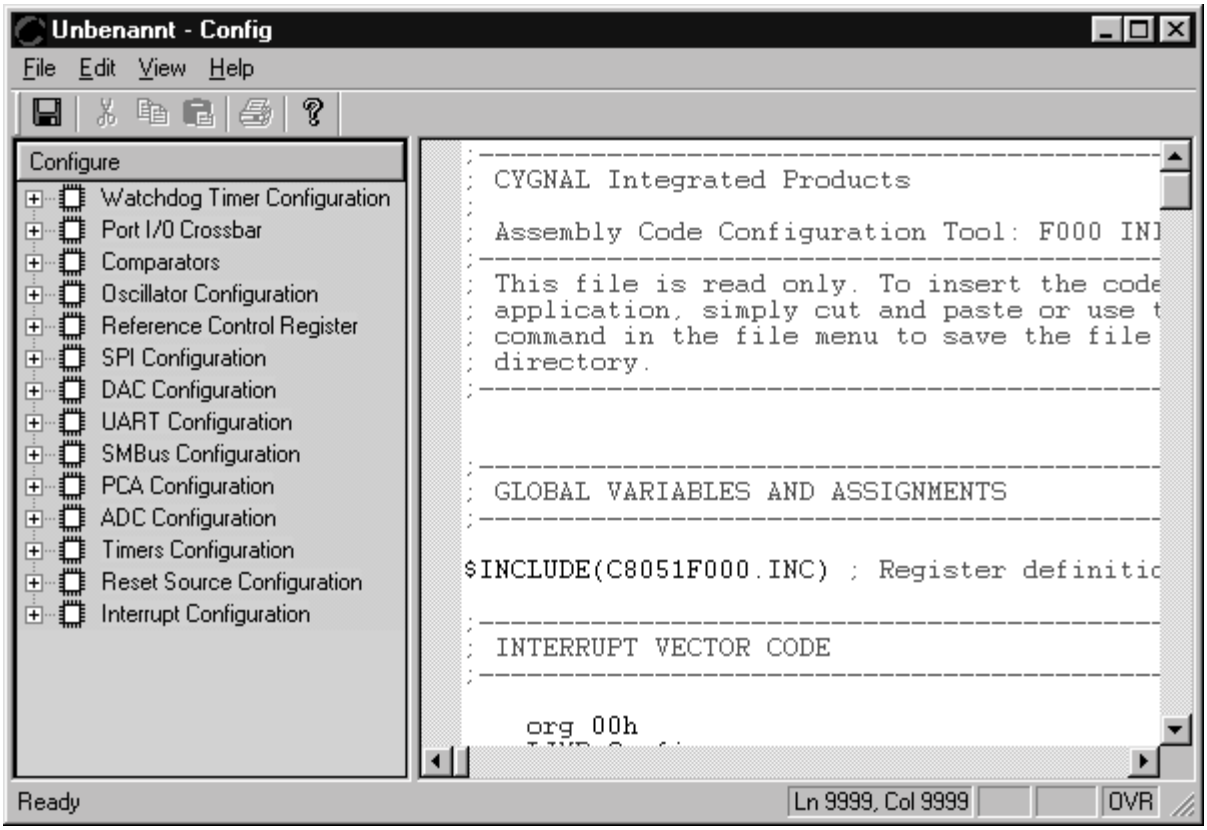


Abbildung 9 Konfigurationsliste

Die gelisteten Elemente sind der Reihe nach abzuarbeiten. Die Konfiguration von Port1 ist in Abbildung 10 beispielhaft gezeigt.

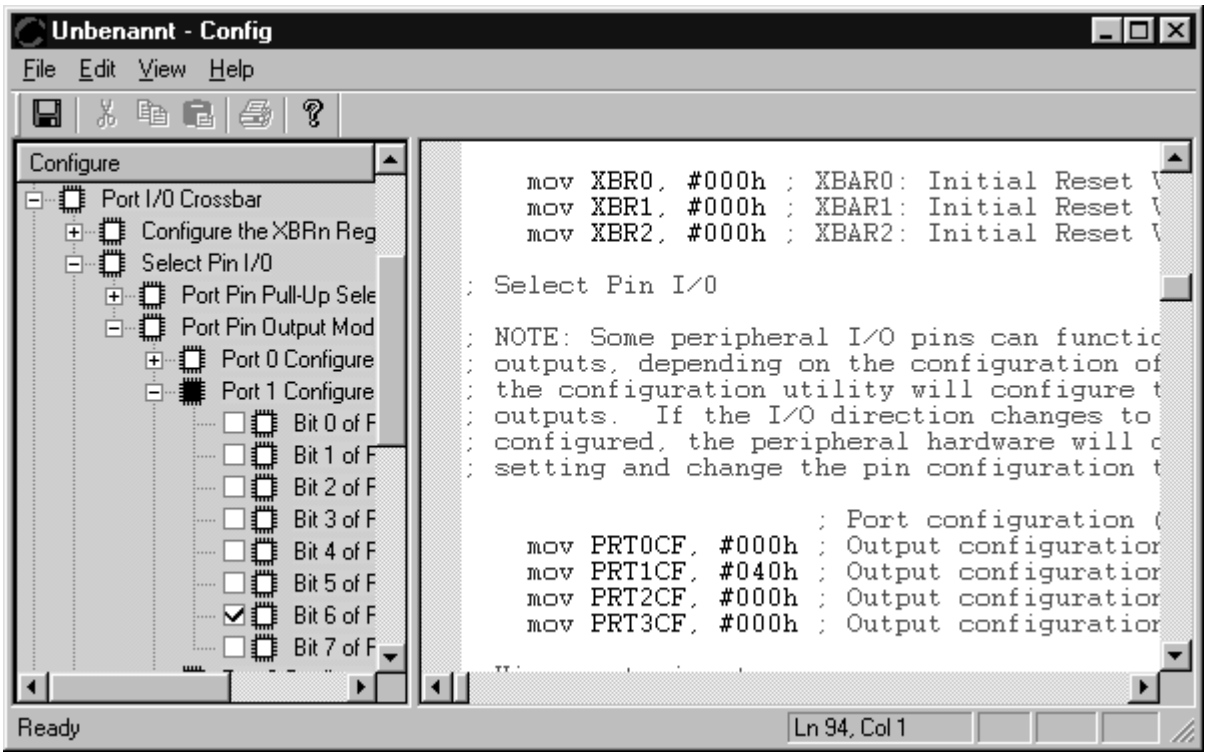


Abbildung 10 Konfiguration Port I/O

Ein kleines Programmbeispiel soll die Vorgehensweise verdeutlichen. Die beim Targetboard an P1.6 angeschlossene LED soll im Takt blinken. Der interne Oszillator wird als Systemtakt verwendet.

Listing 2 zeigt das betreffende Assemblerprogramm BLINK1.ASM mit den fett markierten Einträgen, die mit Hilfe des Configuration Wizards erzeugt wurden.

```
;-------------------------------------------------------------------------
;    Copyright (C) 2000 CYGNAL INTEGRATED PRODUCTS, INC.
;    All rights reserved.
;
;
;    FILE NAME   : BLINK1.ASM
;    TARGET MCU  : C8051F000
;    DESCRIPTION : This program illustrates how to disable the watchdog timer,
;                  configure the Crossbar, configure a port and write to a port
;                  I/O pin.
;                  The internal oscillator will be set to 16 MHz.
;
;
;-------------------------------------------------------------------------

;-------------------------------------------------------------------------
; EQUATES
;-------------------------------------------------------------------------

$include (c8051f000.inc)         ; Include regsiter definition file.
```

```
GREEN_LED  EQU   P1.6           ; Port I/O pin connected to Green LED.

;-----------------------------------------------------------------------------
; VARIABLES
;-----------------------------------------------------------------------------


;-----------------------------------------------------------------------------
; RESET and INTERRUPT VECTORS
;-----------------------------------------------------------------------------

         ; Reset Vector
         org     00h
         ljmp    Main                  ; Jump beyond interrupt vector space.

;-----------------------------------------------------------------------------
; MAIN PROGRAM CODE
;-----------------------------------------------------------------------------

         org     0B3h                  ; End of interrupt vector space.

Main:
         ; Disable the WDT. (IRQs not enabled at this point.)
         ; If interrupts were enabled, we would need to explicitly disable
         ; them so that the 2nd move to WDTCN occurs no more than four clock
         ; cycles after the first move to WDTCN.
         mov     WDTCN, #0DEh;
         mov     WDTCN, #0ADh

         ; Enable the Port I/O Crossbar
         mov     XBR2, #40h

         ; Set P1.6 (LED) as push-pull output.  All others default to open-drain.
         orl     PRT1CF,#01000000b

         ; Set internal oscillator to 16 MHz
         orl     OSCICN,#00000111b
         mov     OSCXCN,#0

         ; Initialize LED to OFF
         clr     GREEN_LED

         ; Simple delay loop.
Blink:   mov     R7, #03h
Loop0:   mov     R6, #00h
Loop1:   mov     R5, #00h
         djnz    R5, $
         djnz    R6, Loop1
         djnz    R7, Loop0
         cpl     GREEN_LED                ; Toggle LED.
         jmp     Blink

;-----------------------------------------------------------------------------
; End of file.
END
```

Listing 2 Programmbeispiel BLINK1.ASM

Die Zuordnung der On-Chip Peripherie ist durch den Crossbar Switch sehr flexibel. Bei der Pinzuordnung müssen aber gewisse Prioritäten beachtet werden. Abbildung 11 zeigt die so-genannte „Priority Decode Table“, nach der die Prioritäten bei der Pinzuordnung geregelt sind.

In der Prioritätentabelle markiert ein (●) das jeweilige I/O Pin, welchem das betreffende Signal zugeordnet werden kann.

	P0								P1								P2							
PIN I/O	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
SDA	●																							
SCL		●																						
SCK	●		●																					
MISO		●		●																				
MOSI			●		●																			
NSS				●		●																		
TX	●		●		●		●																	
RX		●		●		●		●																
CEX0	●		●		●		●		●															
CEX1		●		●		●		●		●														
CEX2			●		●		●		●		●													
CEX3				●		●		●		●		●												
CEX4					●		●		●		●		●											
ECI	●	●	●	●	●	●	●	●	●	●	●	●	●	●										
CP0	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●									
CP1	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●								
T0	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●							
/INT0	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●						
T1	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●					
/INT1	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●				
T2	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●			
T2EX	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●		
/SYSCLK	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
CNVSTR	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●

Abbildung 11 Priority Decode Table

Die Pinzuordnung erfolgt in der Prioritätentabelle von oben nach unten. Aktiviert man den I^2C-Bus dann wird dieser immer den I/O Pins P0.0 und P0.1 zugeordnet. Die Signale Tx und Rx des UART werden nur dann den I/O Pins P0.0 und P0.1 zugeordnet, wenn weder I^2C- noch SPI-Bus enabled wurden. Sind sowohl UART als auch I^2C-Bus enabled, dann liegen an P0.0 und P0.1 die Signale des I^2C-Busses und an P0.2 und P0.3 die UART-Signale Tx und Rx.

3 C8051F000 und BASCOM-8051

Im folgenden Abschnitt soll nun BASCOM-8051 zum Programmieren der betreffenden Applikation eingesetzt werden. Das bereits vorgestellt Blinkprogramm wird als erstes in BASCOM-8051 umgesetzt und damit das grundsätzliche Vorgehen beschrieben.

Programmbeispiele zum Timerinterrupt und zur AD-DA-Umsetzung zeigen weitere Möglichkeiten des komplexen Mikrocontrollers C8051F000.

3.1 Blinkende LED

Sicher muss man für eine blinkende LED nicht unbedingt eine Hochsprache zum Einsatz bringen. Die unterschiedliche Behandlung der beiden Bereiche der Programmierung eines Mikrocontrollers - Initialisierung und Anwendung – zeigt aber bereits dieses einfache Beispiel deutlich.

Für den Initialisierungsteil verwenden wir wieder das Configuration Wizard der CYGNAL Entwicklungsumgebung.

In unserem Beispiel sollen der Watchdog ausgeschaltet werden und der interne Oszillator eine Taktfrequenz von 2 MHz liefern. Pin1.6 treibt wieder die auf dem Entwicklungsboard vorhandene LED.

Aus der mit Hilfe des Configuration Wizards der CYGNAL Entwicklungsumgebung erstellten Assemblerdatei werden schließlich die zur Initialisierung erforderlichen Assemblerzeilen ausgeschnitten und als Assembler-Quelltext in den zu erstellenden BASCOM-Quelltext übernommen. Zu beachten sind die kleinen Notationsunterschiede.

Anhand der Watchdog Timer Konfiguration sollen die Unterschiede verdeutlicht werden. Die folgenden Zeilen zeigen einen Ausschnitt aus den vom Configuration Wizard erzeugten Assemblerzeilen.

```
; Watchdog Timer Configuration
mov WDTCN, #0DEh                        ; Disable WDT
mov WDTCN, #0ADh
```

Kommentare werden in BASCOM-8051 durch das Zeichen ‘ eingeleitet. Die hexadezimale Darstellung des Datenbytes im MOV-Befehl muss ebenfalls angepasst werden.

Unter Beachtung dieser Erfordernisse werden die betreffenden Assemblerzeilen zwischen die Compilerdirektiven `$asm` und `$end asm` platziert. Die eigentliche Anwendung kann sich nun als BASCOM-Quelltext anschließen.

```
$noinit
$crystal = 2000000

$asm

' Watchdog Timer Configuration
 mov WDTCN, #h'DE                          ' Disable WDT
 mov WDTCN, #h'AD

' Configure The Xbrn Registers
 mov XBR2, #h'40                           ' XBAR2: Initial Reset Value

' Port Configuration(1 = Push Pull Output)
 mov PRT1CF, #h'40                         ' Output configuration for P1

' Oscillator Configuration
 mov OSCXCN, #h'00                         ' External Oscillator Control Register
 mov OSCICN, #h'04                         ' Internal Oscillator Control Register

$end Asm

Led Alias P1.6

Led = 0

do
   Led = not Led                           ' Toggle LED
   waitms 100                              ' Wait 100 ms
loop

End
```

Listing 3 Programmbeispiel BLINK.BAS

Das compilierte Programm kann im Simulator wie gewohnt getestet werden. Hat man das Programm BLINK.HEX mit Hilfe der CYGNAL Entwicklungsumgebung in das Flash Memory des eingesetzten Mikrocontrollers C8051F000 heruntergeladen, dann kann es in der Zielhardware gestartet werden. Abbildung 12 weist auf eine Einschränkung beim Debugging hin.

Abbildung 12 Hinweis zum Debugging

In der CYGNAL Entwicklungsumgebung kann das disassemblierte Programm Aufschluss über den erzeugten Code geben. Abbildung 13 zeigt den Bereich der Initialisierung.

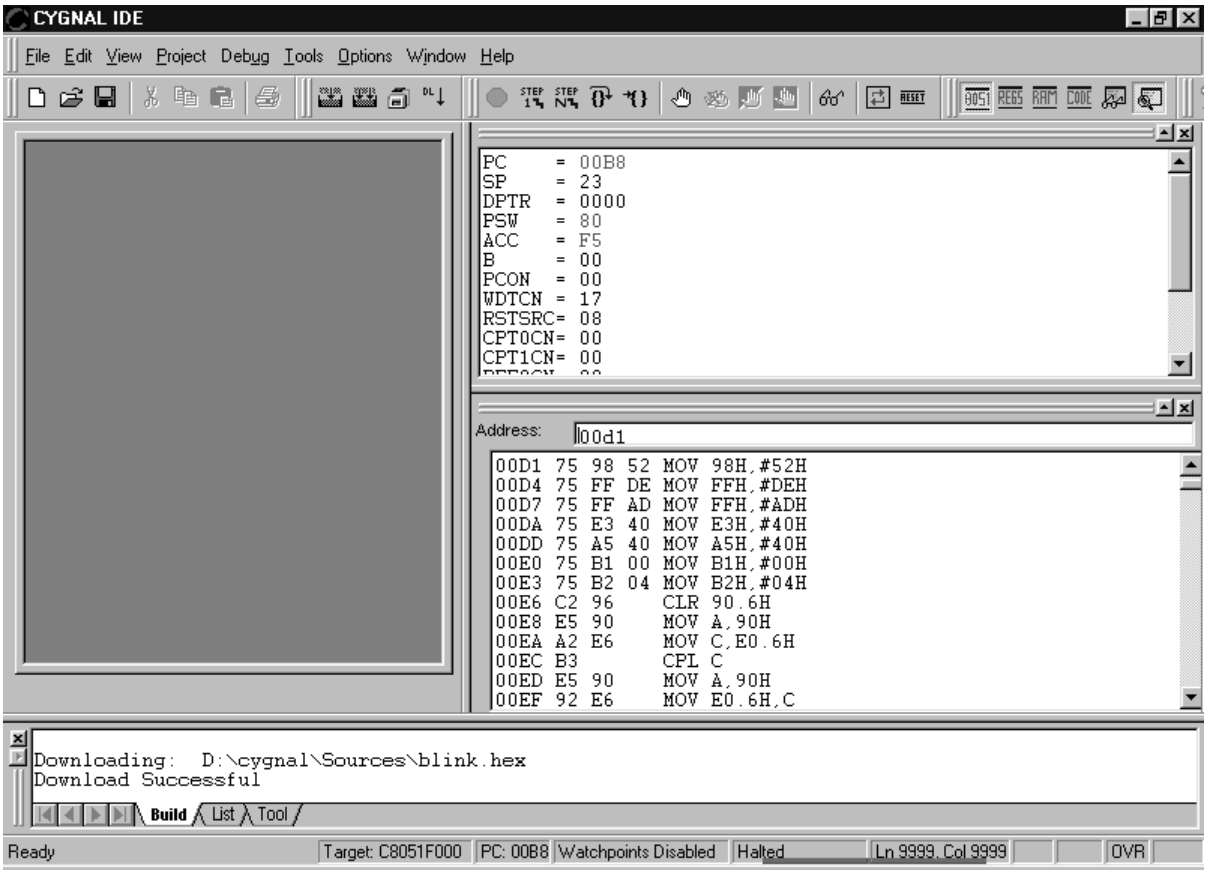


Abbildung 13 Disassembliertes Programm

Dieses einfache Programmbeispiel zeigt die grundsätzliche Vorgehensweise. Ein komplexeres Programmbeispiel, welches auch die Ressourcen des C8051F000 verwendet, soll im folgenden Abschnitt vorgestellt werden.

3.2 Timerinterrupt

Die LED soll im folgenden Programmbeispiel durch einen Timerinterrupt zum Blinken gebracht werden. Eine solche Möglichkeit ist immer dann von Nutzen, wenn man einen von der Anwendung nicht benutzten Timer als „Lebenszeichen“ verwenden will.

Für unsere Zwecke soll also Timer0 als 16-Bit Timer arbeiten. Der vom Timer Overflow angeforderte Interrupt toggelt nun die LED. Listing 4 zeigt den Quelltext des Programms TIMER0.BAS.

Für Timer0 wird mit der Instruktion `On Timer0 Timer0_isr Nosave` eine Interrupt Service Routine vorbereitet, der am Ende des Programms noch der abzuarbeitende Code zugewiesen wird.

Im Initialisierungsteil stehen nun zusätzliche Anweisungen für den Timer und den verwendeten Interrupt.

Die vorgenommen Initialisierung von Timer0 bedeutet in der Reihenfolge der Anweisungen

- Timer0 wird mit dem durch 12 geteilten Systemtakt getaktet
- Timer0 wird mit dem Wert 0 initialisiert
- Timer0 arbeitet als 16-Bit Timer, d.h. nach 65536 Takten erfolgt ein Timer Overflow
- Timer0 wird sofort nach der Initialisierung gestartet

Das Hauptprogramm besteht aus einer leeren Endlosschleife, da alle Programmaktivitäten in der Interrupt Service Routine liegen.

```
$crystal = 2000000

On Timer0 Timer0_isr Nosave

$asm

' Watchdog Timer Configuration
 mov WDTCN, #h'DE                          ' Disable WDT
 mov WDTCN, #h'AD

' Configure The Xbrn Registers
 mov XBR2, #h'40                           ' XBAR2: Initial Reset Value

' Port Configuration(1 = Push Pull Output)
 mov PRT1CF, #h'40                         ' Output configuration for P1

' Oscillator Configuration
 mov OSCXCN, #h'00                         ' External Oscillator Control Register
 mov OSCICN, #h'04                         ' Internal Oscillator Control Register

' Timer Configuration
 mov CKCON, #h'00                          ' Clock Control Register - Prescaler 12
 mov TH0, #h'00                            ' Timer 0 High Byte Initialization
 mov TL0, #h'00                            ' Timer 0 Low Byte Initialization
 mov TMOD, #h'01                           ' Timer Mode 1 - 16 Bit Mode
 mov TCON, #h'10                           ' Timer Control Register - Start Timer

' Interrupt Configuration
 mov IE, #h'82                             ' Interrupt Enable
 mov IE1, #h'00                            ' Interrupt Priority IP

$end Asm

Led Alias P1.6

Led = 0
```

```
Do
                                          ' Empty Loop as Main Program
Loop

End

Timer0_isr:
      Led = Not Led                       ' Toggle LED
Return
```

Listing 4 Programmbeispiel TIMER0.BAS

3.3 AD-DA-Umsetzung

Wie eingangs dargestellt hat der C8051F000 alle Ressourcen für ein komplettes System zur Messwertverarbeitung bereits auf dem Chip. Abbildung 14 zeigt die einzelnen Funktionseinheiten in einem Blockschema.

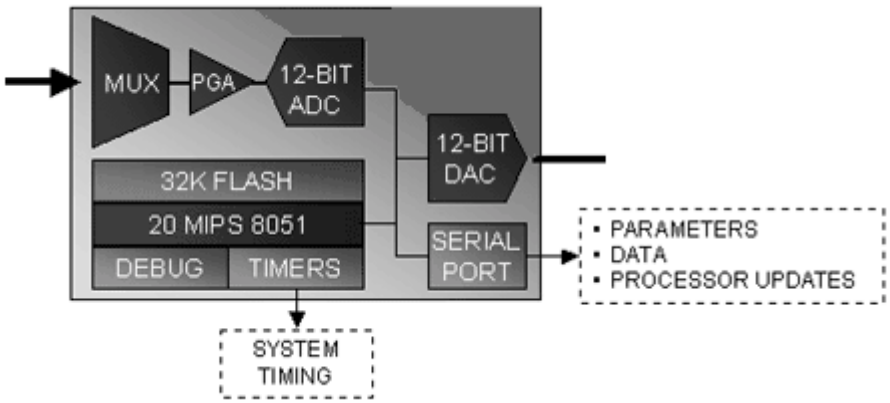


Abbildung 14 Messwertverarbeitung auf den Chip

Im folgenden Programmbeispiel wird der Ausgang eines DA-Umsetzers mit einem Eingang des AD-Umsetzers verbunden. Damit kann das Programmbeispiel gleichzeitig zum Test der Linearität der AD-Umsetzung dienen.

Die Sequenz Ansteuerung DA-Umsetzer und Auslesen des AD-Umsetzers soll wieder in einem Timerinterrupt erfolgen. Überall dort, wo exakte Abtastzeitpunkte für die spätere Signalverarbeitung erforderlich sind, wird man die AD-Umsetzung von einem Timer aus starten müssen. Außerdem soll der Timerinterrupt wieder als Lebenszeichen dienen.

Listing 5 zeigt das Programmbeispiel ADDA_T0.BAS, bei dem der Ausgang DAC0 mit dem Eingang AIN0 verbunden wurde.

```
$crystal = 2000000

On Timer0 Timer0_isr

Declare Sub Write_dac(dac As Word)
Declare Sub Read_adc()

Dim Idx As Word , Dac As Word , Adc As Word , Tmp As Word
Dim ADReady as byte

Led Alias P1.6

Led = 0
Idx = 0

$asm

' Watchdog Timer Configuration
 mov WDTCN, #h'DE                          ' Disable WDT
 mov WDTCN, #h'AD

' Configure The Xbrn Registers
 mov XBR2, #h'40                           ' XBAR2: Initial Reset Value

' Port Configuration(1 = Push Pull Output)
 mov PRT1CF, #h'40                         ' Output configuration for P1

' Oscillator Configuration
 mov OSCXCN, #h'00                         ' External Oscillator Control Register
 mov OSCICN, #h'04                         ' Internal Oscillator Control Register

' Reference Control Register Configuration
 mov REF0CN, #003h                        ' Reference Control Register

' Dac Configuration
 mov DAC0CN, #h'80h                        ' DAC0 Control Register
 mov DAC0L, #h'00h                         ' DAC0 Low Byte Register
 mov DAC0H, #h'00h                         ' DAC0 High Byte Register

' Adc Configuration
 mov AMX0CF, #h'00h                        ' AMUX Configuration Register - SE
 mov AMX0SL, #h'00h                        ' AMUX Channel Select Register - AIN0
 mov ADC0CF, #h'00h                      ' ADC Configuraion Register - SYSCLK, G=1
 mov ADC0CN, #h'C0h                        ' ADC Control Register - ADBusy

 mov ADC0LTH, #h'00h                       ' ADC Less-Than High Byte Register
 mov ADC0LTL, #h'00h                       ' ADC Less-Than Low Byte Register
 mov ADC0GTH, #h'FFh                       ' ADC Greater-Than High Byte Register
 mov ADC0GTL, #h'FFh                       ' ADC Greater-Than Low Byte Register

' Timer Configuration
 mov CKCON, #h'00                        ' Clock Control Register - Prescaler 12
 mov TH0, #h'00                            ' Timer 0 High Byte Initialization
 mov TL0, #h'00                            ' Timer 0 Low Byte Initialization
 mov TMOD, #h'01                           ' Timer Mode 1 - 16 Bit Mode
 mov TCON, #h'10                           ' Timer Control Register - Start Timer

' Interrupt Configuration
 mov IE, #h'82                             ' Interrupt Enable
 mov IE1, #h'00                            ' Interrupt Priority IP

$end Asm
```

```
Do
                                          ' Empty Loop as Main Program
Loop

End

' Timer0 Interrupt Service Routine
Timer0_isr:
   Led = Not Led                          ' Toggle LED
   Write_dac Idx                          ' Write DAC
   Delay                                  ' Wait for analog outputs
   Read_adc                               ' Read ADC
   Idx = Idx + 64                         ' Increment index
Return

' Write Digital-to-Analog Converter
Sub Write_dac(dac As Word)
   Dac0l = Low(dac)
   Dac0h = High(dac)
End Sub

' Read Analog-to-Digital Converter
Sub Read_adc()
   orl ADC0CN, #h'10                      ' Set Adbusy
   do
      ADReady = ADC0CN AND &H10           ' Wait for end of AD Conversion
   loop until ADReady = 0
   Tmp = Adc0h                            ' Read Hi Byte of ADC
   Shift Tmp , Left , 8
   Adc = Tmp + Adc0l                      ' Build Result of AD Conversion
End Sub
```

Listing 5 Programmbeispiel ADDA_T0.BAS

Das Programm ADDA_T0.BAS ist vom Aufbau her identisch mit den bisher betrachteten Programmbeispielen. Zu Begin des Programms werden die Interrupt Service Routine, sowie zwei Subroutinen zur Ansteuerung von DA- und AD-Umsetzer deklariert. Diesen Deklarationen folgen die Variablendeklarationen.

Der sich anschließende Assemblerteil ist wieder mit dem Configuration Wizard der CYGNAL Entwicklungsumgebung erzeugt worden und initialisiert die verwendeten Special Function Register.

Wichtig an dieser Stelle sind die Initialisierungen der Referenzspannung, des DA-Umsetzers sowie des AD-Umsetzers.

Als Referenzspannung wird die interne Referenzspannung von typisch 2,43 V verwendet. Ein Quantisierungsschritt beträgt damit ca. 0,59 mV.

Der DA-Umsetzer wird enabled und mit einem Ausgangswert belegt. Durch Beschreiben der Register DAC0L und DAC0H kann der gewünschte Ausgangswert eingestellt werden. Da die Daten nach dem Beschreiben des Registers DAC0H gelatched werden, ist die Reihenfolge des Beschreibens der Register DAC0L und DAC0H unbedingt einzuhalten. Abbildung 15

zeigt ein Prinzipschaltbild für die beiden 12-Bit DA-Umsetzer im C8051F000 mit den betreffenden Special Function Registern.

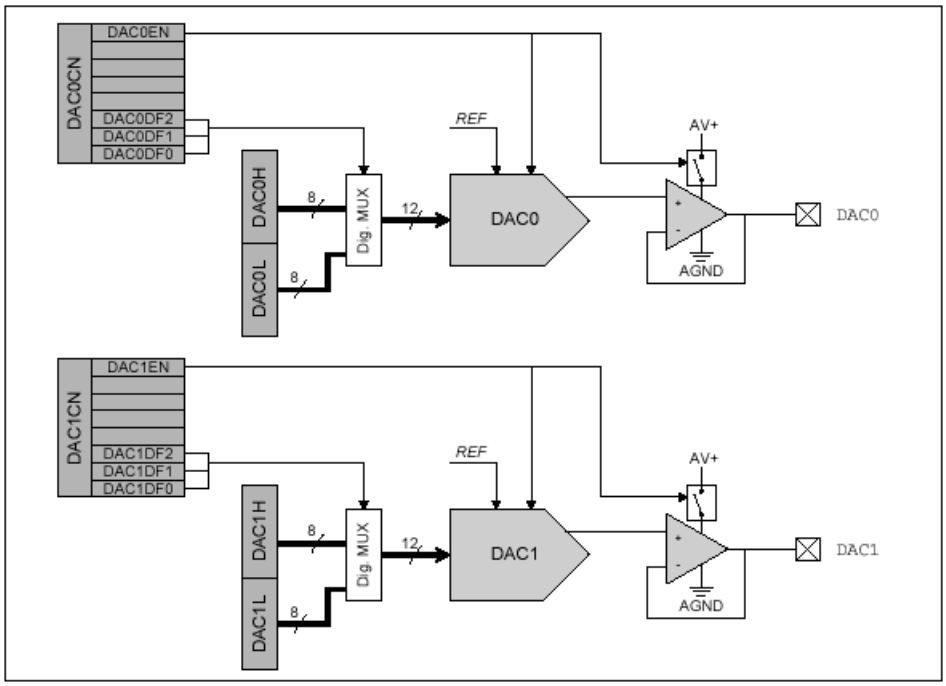


Abbildung 15 DA-Umsetzer DAC0 und DAC1

Der AD-Umsetzer bietet sehr komplexe Möglichkeiten. Abbildung 16 zeigt ein Prinzipschaltbild für den 12-Bit AD-Umsetzer im C8051F00 mit den betreffenden Special Function Registern.

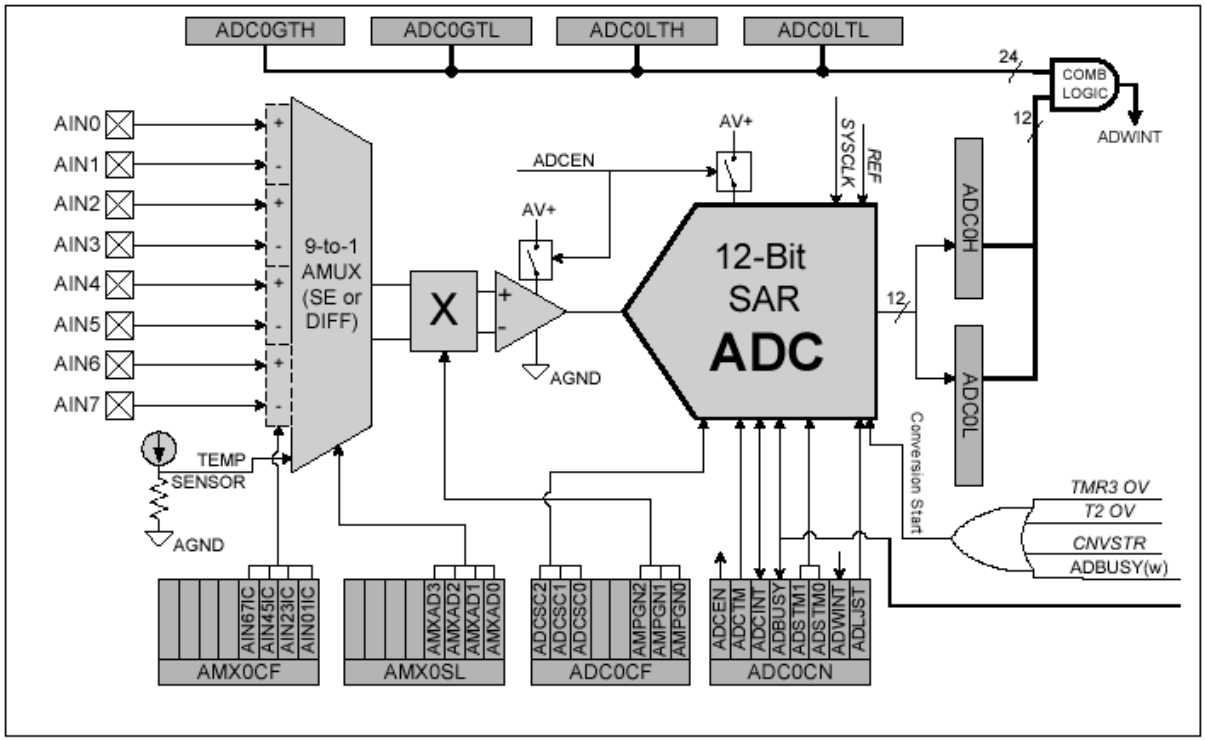


Abbildung 16 12-Bit AD-Umsetzer

Das komplette Subsystem umfasst einen 9-kanaligen Analog-Multiplexer, einen programmierbaren Verstärker und den nach dem Verfahren der sukzessiven Approximation arbeitenden 12-Bit AD-Umsetzer. Integriert sind weiterhin eine Track&Hold-Stufe sowie ein Fensterkomparator.

Durch die vorgenommen Initialisierung arbeiten alle Eingänge separat (single-ended) und der Analog-Multiplexer schaltet AIN0 an den programmierbaren Verstärker, dessen Verstärkung auf 1 gesetzt wurde. SYSCLK dient als Takt für den AD-Umsetzer. Arbeitet der C8051F000 mit höheren Taktfrequenzen, dann kann die Taktfrequenz des AD-Umsetzers auch herabgesetzt werden (SYSCLK/2 u.s.w.). Die Taktfrequenz der AD-Umsetzung sollte 2 MHz nicht überschreiten.

Für die Triggerung der AD-Umsetzung gibt es wiederum verschiedene Möglichkeiten:

1. Steuerung über das Bit ADBUSY im Register ADC0CN
2. Timer3 Overflow
3. Steigende Flanke des Signals CNVSTR (externer Trigger)
4. Timer2 Overflow

Im Programmbeispiel wurde die Triggerung über das Bit ADBUSY gewählt. Dieses Bit ist zum Start der AD-Umsetzung zu setzen. Nach Abschluss der AD-Umsetzung setzt der AD-Umsetzer dieses Bit zurück. Das Ergebnis der AD-Umsetzung steht dann zum Auslesen bereit.

In der Subroutine Read_adc() sind die Schritte

- Start der AD-Umsetzung durch Setzen von ADBUSY
- Warten auf das Ende der AD-Umsetzung durch Abfrage von ADBUSY
- Auslesen der Ergebnisbytes

wiederzufinden.

Das Hauptprogramm selbst besteht nur aus einer leeren Endlosschleife. Das Beschreiben des DA-Umsetzers und das Auslesen des AD-Umsetzers werden durch den Timerinterrupt gesteuert.

Die betreffende Interrupt Service Routine umfasst

- Toggeln der LED („Lebenszeichen“),
- Beschreiben des DA-Umsetzers (mit einem Indexwert),
- Kurze Verzögerung zum Einschwingen der Analog-Ausgänge,
- Start und Abfrage des AD-Umsetzers,
- Erhöhen des Indexwertes (für den nächsten Zyklus).

Durch diesen Zyklus wird die gesamte Kennlinie des DA-AD-Subsystems durchfahren.

Setzt man nach dem Download des Hex-Files an das Ende der Interrupt Service Routine einen Breakpoint, dann können die Ergebnisse direkt im Speicher verfolgt werden. Abbildung 17 zeigt das Setzen des Breakpoint in der CYGNAL Entwicklungsumgebung.

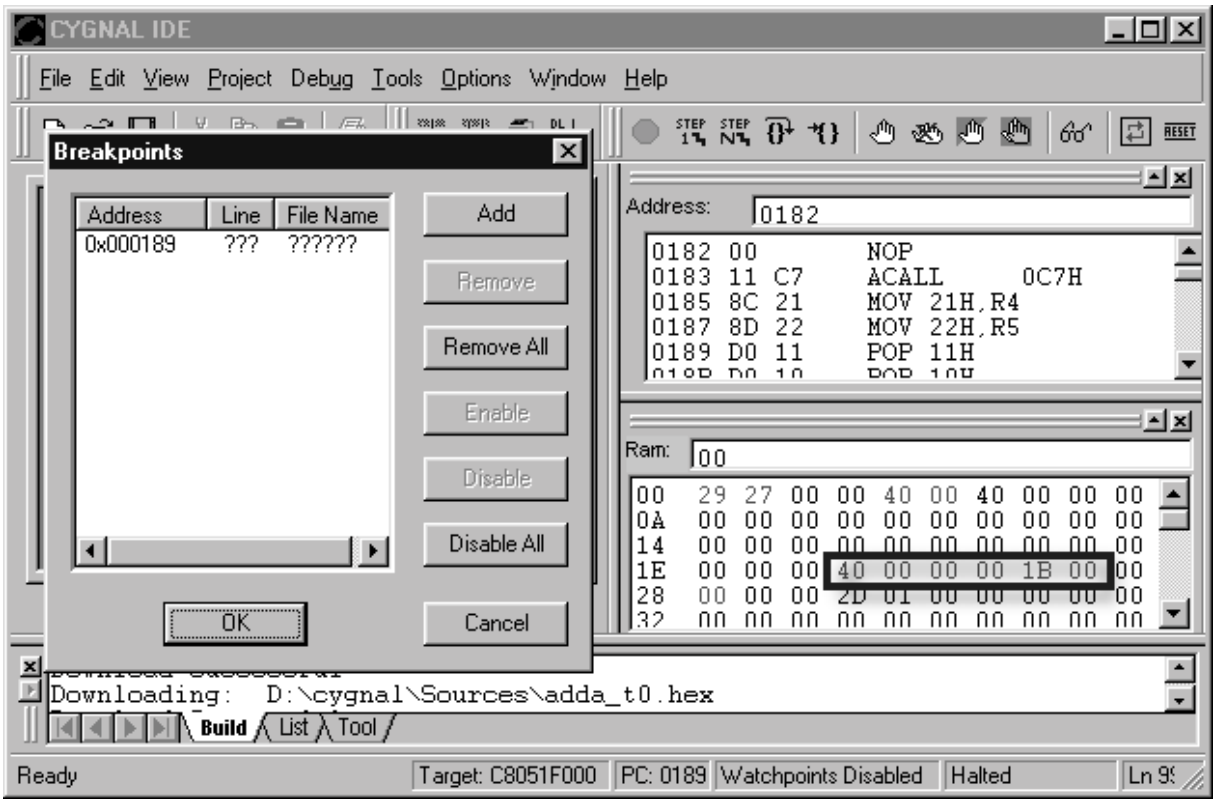


Abbildung 17 Breakpoint am Ende der Interrupt Service Routine

Da in der CYGNAL Entwicklungsumgebung nur das durch BASCOM-8051 erzeugte HEX File verwendet wird, sind nicht alle komfortablen Funktionen dieser Umgebung verfügbar.

Im Disassembler wurde eine Stelle am Ende der Interrupt Service Routine gesucht. Vor dem Pop der Register ist das beispielsweise Adresse 0189_H. Aus dem Reportfile von BASCOM-8051 erhält man die Adressen der interessierenden Variablen im RAM. Hier sind die folgenden Werte von Interesse:

Variable	*Adresse*
IDX	21_H ; 22_H
DAC	23_H ; 24_H
ADC	25_H ; 26_H

Abbildung 17 zeigt also die Werte der Variablen nach dem ersten Interrupt. Die Variable IDX ist bereit inkrementiert, der DA-Umsetzer wurde mit dem Wert 0 beschrieben, während der AD-Umsetzer aber einen Wert $1B_H$ erkannt hat.

Schrittweise kann nun die Kennlinie der DA-AD-Umsetzung aufgenommen werden. Abbildung 18 zeigt diese Kennlinie. ADU bezeichnet die Resultate der AD-Umsetzung, wäh-

rend DAU den in den DA-Umsetzer geschriebenen Ausgangswert bezeichnet. Abbildung 19 hingegen zeigt den resultierenden Fehler.

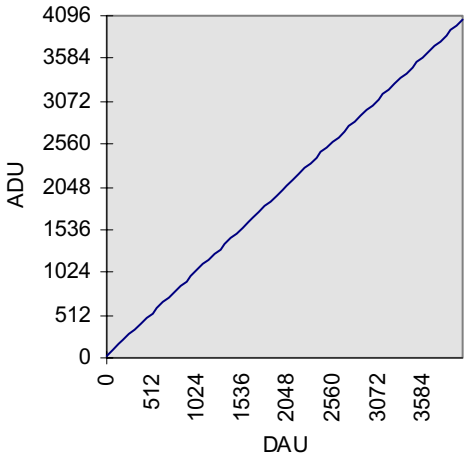


Abbildung 18 Kennlinie ADU - DAU

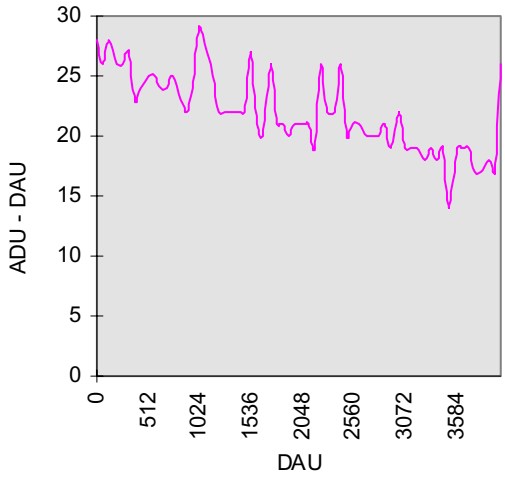
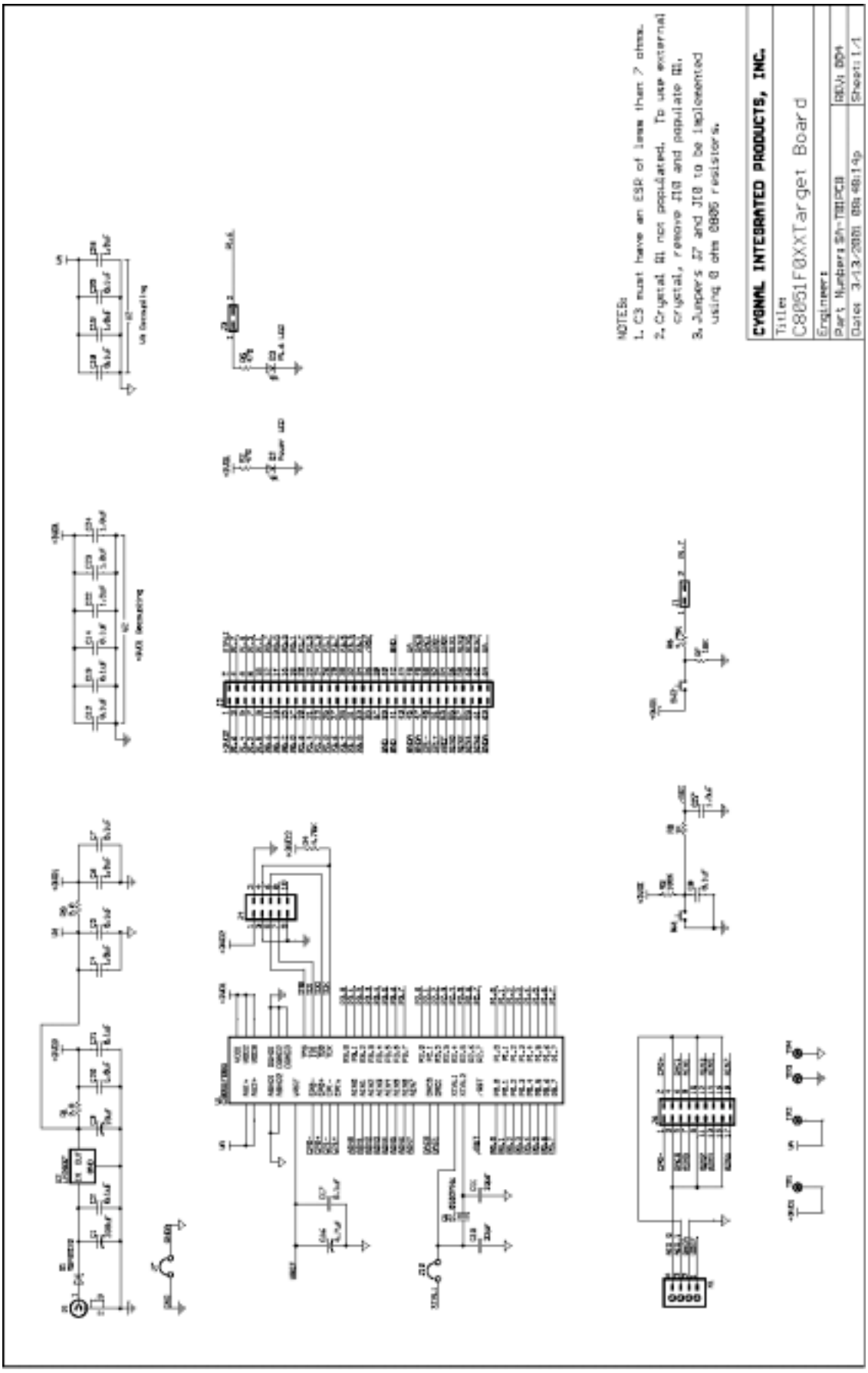


Abbildung 19 Linearitätsfehler ADU - DAU

Der mittlere Fehler liegt bei 21 Counts, d.h. bei einem Quantisierungsschritt von 0,59 mV sind das etwa 12,46 mV. Dieser Fehler spiegelt aber nicht zwangsläufig den Fehler des AD-Umsetzers wieder, sondern wird weitgehend durch den Schaltungsaufbau bestimmt. Mit anderen Worten, will man die Genauigkeit der 12-Bit AD-Umsetzung voll ausnutzen, dann werden an Layout und Schaltungsaufbau hohe Forderungen gestellt.

Zur Verdeutlichung der hier verwendeten Hardware ist abschliessend noch das Schaltbild des CYGNAL Entwicklungsboards gezeigt.

Alle weiteren Informationen können der Webseite des Herstellers CYGNAL Integrated Products, Inc. [www.cygnal.com] entnommen werden.


NOTES:
1. C3 must have an ESR of less than 7 ohms.
2. Crystal X1 not populated. To use external crystal, remove J18 and populate X1.
3. Jumpers J7 and J10 to be implemented using 0 ohm 0805 resistors.
CYGNAL INTEGRATED PRODUCTS, INC.
Title: C8051F0XXTarget Board
Engineer:
Part Number: Sh-TBPCB
REV: 0D4
Date: 3/13/2001 09:48:14p
Sheet 1/1