Hochschule Magdeburg • Stendal

Institut für Elektrotechnik

Universidad de Valladolid

Education and Culture
Lifelong Learning Programme
ERASMUS

RESUMEN DEL PROYECTO FINAL DE CARRERA

"ENTWICKLUNG EINES DIGITALEN REGLERS FÜR SYNCHRONGENERATOREN"

MANUEL SILVA GONZÁLEZ

Índice:

Introducción:

Actualmente el uso de los motores está ampliamente extendido, el control de velocidad de un generador es una tarea muy importante ya que para la mayoría de los procesos en espacios industriales es necesario el control de un motor para que pueda adaptarse a las necesidades del espacio en el que trabaja.

En este proyecto se ha buscado la forma de crear una serie de normas digitales para controlar generadores síncronos por medio de un microcontrolador, lo que se busca por medio de ecuaciones matemáticas es hacer un variador de frecuencia digital para controlar estos generadores. Como el proyecto se basa en el control de generadores síncronos, dependiendo de la frecuencia, el generador aumentará la velocidad o la disminuirá. Para lograr el control de estos generadores es necesario crear una serie de ecuaciones matemáticas que puedan leer y corregir la frecuencia. A diferencia de generadores de corriente continua, los generadores síncronos tiene un tamaño más reducido y no existe ningún tipo de rozamiento mecánico, por lo que el motor es mucho más duradero y fiable.

Al hacer el estudio en un laboratorio, se ha prescindido de este motor y se ha cambiado por un equivalente matemático. El microcontrolador lee una señal de entrada, que corresponde a una referencia de velocidad solicitada, esta señal se procesa y se inyecta al motor. En la salida del generador tenemos un sensor de lectura para poder hacer, en caso de que sea necesario, una corrección matemática, la cual se hace por medio de una realimentación. Con esta realimentación se rectifican las funciones en caso de que el generador necesite más velocidad para igualarse a la especificación requerida, así como que el operario pueda ver la velocidad real del geerador en el momento en el que el esté trabajando.

Objetivo y explicación del proyecto:

El objetivo del proyecto es el desarrollo de un regulador digital para el control de un generador síncrono por medio del microcontrolador 80C167 de la marca Infineon, para ello el regulador tiene que leer los datos del voltaje y de la corriente, con ello se puede determinar el valor efectivo del tamaño, así como de la frecuencia del desplazamiento de fase mediante la implementación de algoritmos de control y reguladores de frecuencia y corriente. El proyecto se hace siguiendo los siguientes pasos:

1. Estudio y análisis del Hardware y el software
2. Programación de la comunicación entre el microcontrolador y el PC
3. Programación de las lecturas de corriente y tensión
4. Programación de una función que reconozca Frecuencia y fase
5. Puesta en servicio de las salidas analógicas
6. programación de un regulador PI
7. Elaborar un programa para poder ver en la pantalla los procesos dinámicos del regulador.

Explicación del programa y del microcontrolador:

Figura 1: *Logo del programa TASKIN*

Para la realización de este proyecto se ha necesitado el entorno de desarrollo Tasking TM de la empresa Altium, como se explica más detalladamente en el proyecto, se crea un entorno de trabajo y se programa en el. Para realizar la programación del microcontrolador se ha dividido el entorno en los siguientes programas:

1. MAIN.C:

   El archivo de ejecución MAIN.C es el archivo principal, el archivo que por defecto se ejecuta de manera indefinida por el microcontrolador. Desde este archivo se accede a una serie de funciones, que para hacer la lectura del programa más sencilla, se han programado en otros documentos, de esta forma, se estructura de una forma más sencilla la lectura del programa.

2. SYSDEF.C:

   En esta parte del programa se han programado las especificaciones del Hardware, es decir, las entradas que se usan, el tiempo de conversión analógico-digital, la frecuencia de reloj, etc.

3. SYSDEF.H:

   Esta es la parte de la programación que enlaza los diferentes archivos del SYSDEF.C con el archivo MAIN.C, al hacer una llamada desde MAIN.C a SYSDEF.C tiene que haber una declaración de cómo es la función situada en SYSDEF.C, el tamaño, la forma y el nombre. Estos datos son los que se encuentran en este archivo.

4. CONTROL.C:

   Este programa define las especificaciones de las funciones matemáticas y de control

5. CONTROL.H:

   Al igual que SYSDEF.H con SYSDEF.C, CONTROL.H enlaza los archivos de CONTROL.C con el archivo MAIN.C

6. ADC_INTERRUPT.C:

   En este programa se encuentra una función matemática muy simple, a esta función no se accede por medio del MAIN.C sino que es la función correspondiente a la interrupción analógico-digital. Si se analiza el programa, se puede observar que el nombre de esta función es Interrupts(0x28) void ADC_interrupt(void), el 0x28 corresponde con la interrupción analógico-digital, que se ha programado cada cierto tiempo en SYSDEF.C.

7. ADC_INTERRUPT.H:

   Al igual que las anteriores, corresponde con la declaración de ADC_INTERRUPT.C

8. TRANSMISION_INTERRUPT.C:

   Al igual que ADC_INTERRUPT.C es una interrupción correspondiente a la transmisión hacia el CAN del microcontrolador, esto se utiliza para transmitir del microcontrolador al PC una serie de datos para que puedan ser vistos en pantalla.

9. TRANSMISON_INTERRUPT.H:

   Al igual que los anteriores, corresponde con la declaración de TRANSMISION_INTERRUPT.C

Además de las declaraciones de C, el programa TASKING cuenta con una serie de ellas a mayores que hacen más fácil la programación del microcontrolador. En este proyecto, las declaraciones más usadas han sido: interrupts, que corresponde a una función de interrupción, put_bit, correspondientes a la puesta a 1 o a 0 de los bits. Aparte de estas dos declaraciones, también hay declaraciones específicas del microcontrolador C167 como por ejemplo T3CON y las declaraciones de los bits de T3CON: T3I, T3M, T3R etc. En la figura número 2 se puede ver con más claridad a que se refiere estas siglas.

Junto con el programa TASKING, el microcontrolador es controlado por el COMBI-MODUL C167. En la figura número 3 se puede observar cómo está conectado el microcontrolador a los diferentes dispositivos del módulo, esto se complementa con la figura número 4, que aclara como es conectado el módulo de forma real. Este módulo es a su vez conectado al ordenador mediante un cable VGA. Gracias a los diferentes conectores del módulo, se puede usar salidas normales, y salidas con relé, y en ambas se puede introducir un pulso PWM, esto es muy útil, ya que con las salidas normales se pueden hacer operaciones con otros dispositivos de control y con las salidas de relé, controlar el motor. También existe un total de hasta 23 entradas, de las cuales pueden usar como convertidor analógico digital, salida-entrada temporal interna del microcontrolador o como entrada digital. El sistema es alimentado por un voltaje de 24 voltios y, a pesar de que en este proyecto no se usa, el sistema tiene un sensor de temperatura, entradas digitales de detección de paso bajo a paso alto y de paso alto a paso bajo.

El microcontrolador es un C167 de la marca Infineon, en la bibliografía utilizada se ver el datasheet del mismo. Las cualidades de este microcontrolador de 16 bits son que puede transferir hasta 16bits en cada instrucción, así como la capacidad de su memoria ROM de 128 KByte, su memoria RAM de 2 KByte , sus 8 canales de control periférico y sus 16 niveles de interrupciones.

T3CON
Timer 3 Control Register **SFR ($FF42_H/A1_H$)** **Reset value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	T3 OTL	T3 OE	T3 UDE	T3 UD	T3R	T3M			T3I		
-	-	-	-	-	rwh	rw	rw	rw	rw	rw			rw		

Bit	Function
T3I	**Timer 3 Input Selection** Depends on the operating mode, see respective sections.
T3M	**Timer 3 Mode Control** (Basic Operating Mode) 000: Timer Mode 001: Counter Mode 010: Gated Timer with Gate active low 011: Gated Timer with Gate active high 100: *Reserved.* Do not use this combination. 101: *Reserved.* Do not use this combination. 110: Incremental Interface Mode 111: *Reserved.* Do not use this combination.
T3R	**Timer 3 Run Bit** 0: Timer/Counter 3 stops 1: Timer/Counter 3 runs
T3UD	**Timer 3 Up/Down Control** [1)]
T3UDE	**Timer 3 External Up/Down Enable** [1)]
T3OE	**Alternate Output Function Enable** 0: Alternate Output Function Disabled 1: Alternate Output Function Enabled
T3OTL	**Timer 3 Output Toggle Latch** Toggles on each overflow/underflow of T3. Can be set or reset by software.

Figura 2: *Explicación de las siglas usadas en las declaraciones del programa TASKING*

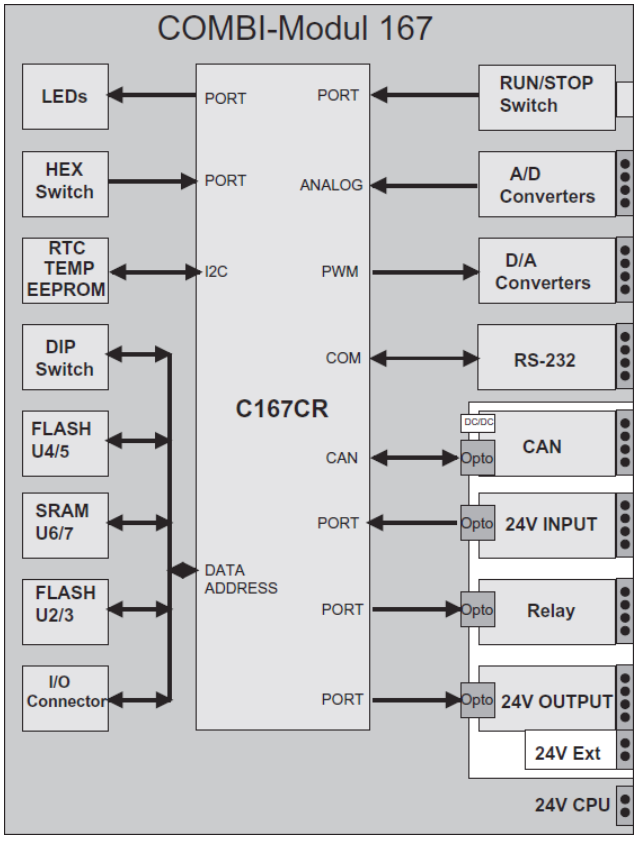


Figura 3: *Modulo-conexión C167*

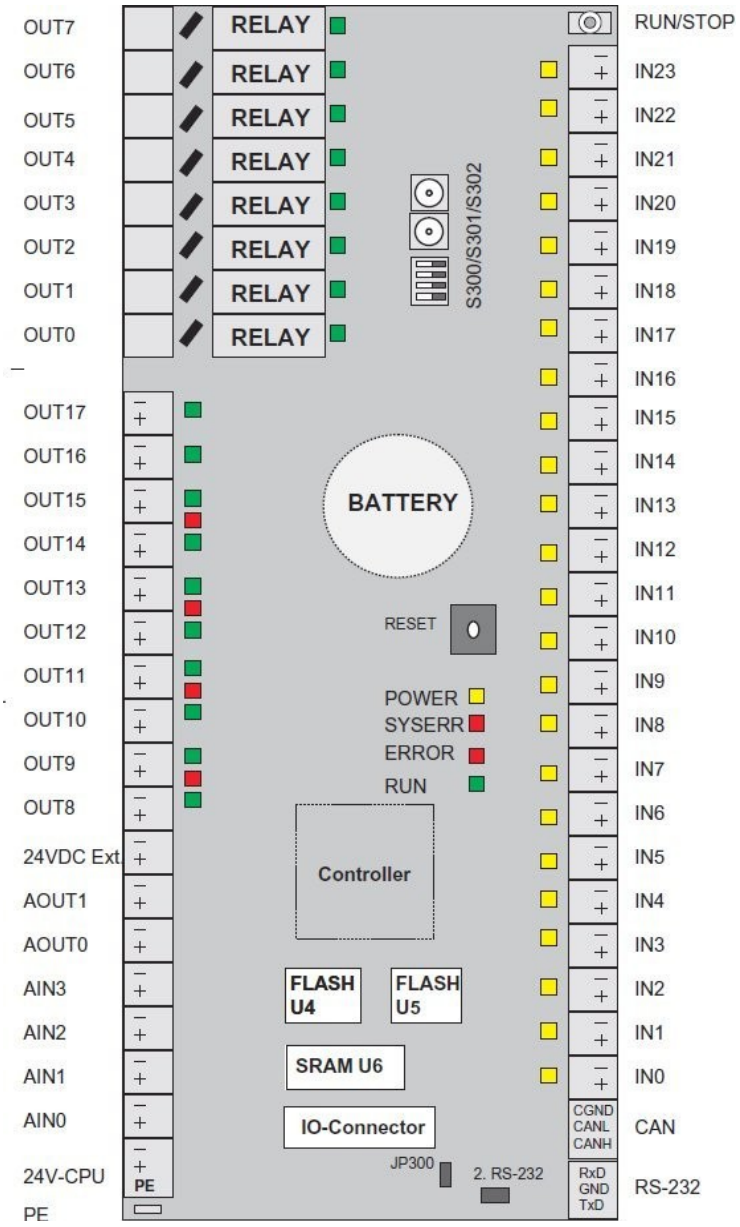


Figura 4: *Modulo Real C167*

Programación:

Como se ha explicado anteriormente, este proyecto se ha hecho mediante el programa TASKING y la programación en C, el programa se encuentra en la documentación adjunta, así como la explicación de este, paso a paso, en la documentación original del proyecto. En este apartado se explica que paso se han seguido para hacer el programa, las funciones más importantes y su funcionamiento, los diagramas y diagramas de bloques que han servido como base así como las funciones matemáticas implementadas en el microcontrolador.

En la figura número 5 se observa el esquema principal del proyecto, en primer lugar, se tiene una señal de referencia que se hace pasar por un filtro, el sentido de los filtros es que al pasar por ellos, las pequeñas variaciones de la señal no afectan al funcionamiento del microcontrolador, en caso de no tener este filtro, cada pequeño cambio en la señal se entendería como un cambio de estado. Este apartado se explicará un poco más adelante. Tras el paso por el filtro, la señal pasa a un bloque de control, la señal de referencia será comparada con la señal de realimentación del motor, a esta señal de diferencia entre las dos señales se le aplicará una serie de funciones que corresponden al controlador PI, y por último, la salida irá a los relés que harán que se produzca el funcionamiento del motor.

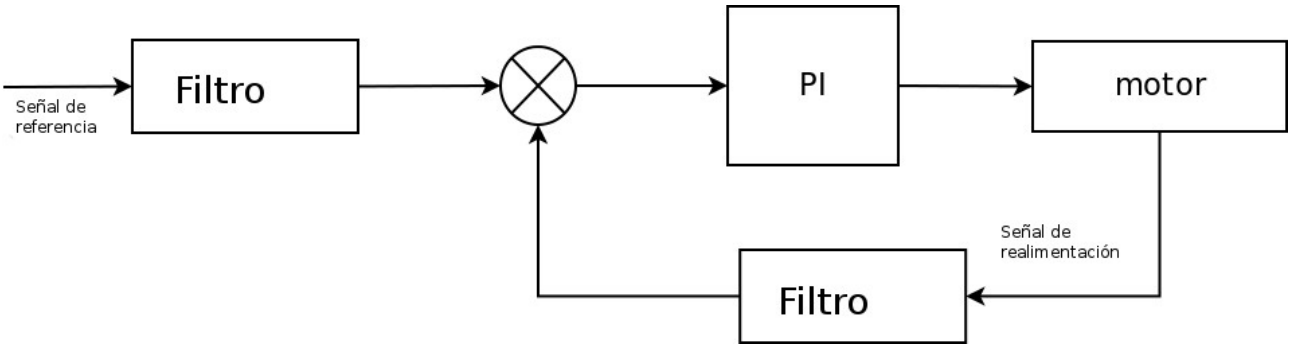


Figura 5: *Esquema general del proyecto*

En la Figura número 6 se observa de una forma más precisa el funcionamiento matemático que se ha programado en el microcontrolador, se ha seguido ese esquema para realizar el comportamiento general, y a partir de ahí, centrarse en los diferentes programas y funciones.

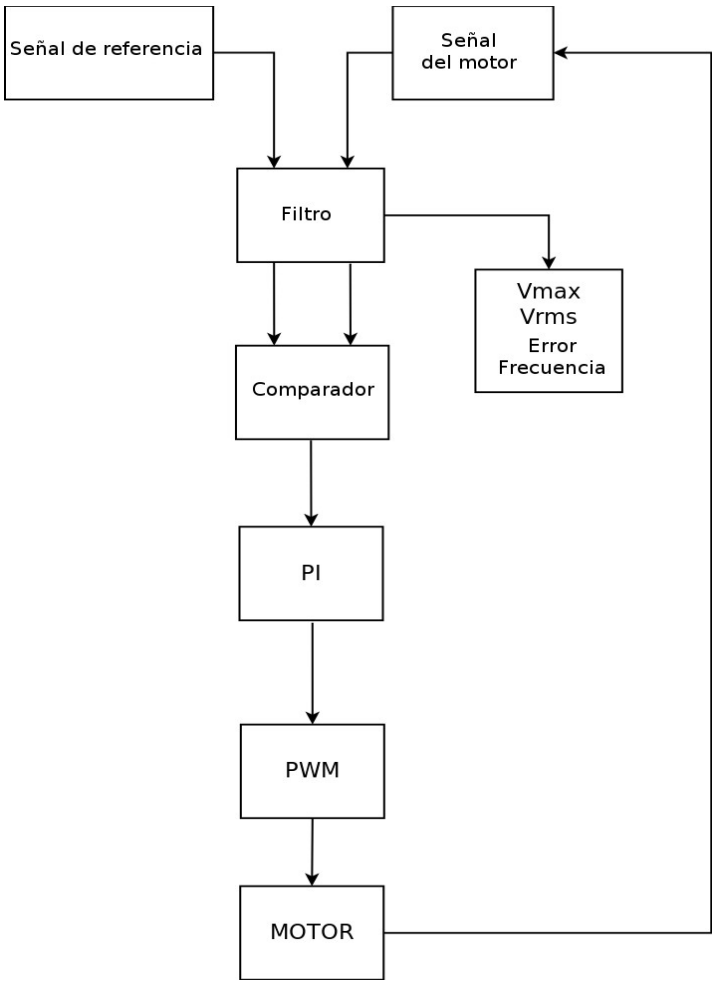


Figura 6: *Esquema específico del proyecto*

Como se observa en la figura 6 se parte de una señal de referencia, gracias a las entradas analógico-digital del microcontrolador, se puede procesar los datos de varias señales a la vez, en este caso, solamente se necesitan para este proyecto dos señales, la primera, corresponde a la señal de referencia, esta señal ha sido pensada para que el control de la velocidad del motor sea controlado por medio de la misma. A la señal de referencia hay que sumar la señal del motor, esta señal es medida mediante sensores, y se introduce al filtro mediante el convertidor analógico digital.

Cuando la señal ya ha sido filtrada, se envía, por medio de una interrupción, al PC el valor de la tensión máxima y eficaz de cada señal, el error que hay entre las señales, que corresponde al desplazamiento, y la frecuencia de ambas señales. Por otra parte, las señales van a un comparador, este comparador decide si hay que variar la frecuencia o el valor máximo de la señal, principalmente el propósito de este comparador es corregir la frecuencia de la señal y el error producido entre ambas señales, después de que el comparador actúe, entra en funcionamiento el controlador proporcional integral, cuya

función es hacer los cambios necesarios paulatinamente, de forma que se llegue al control deseado eficazmente y sin grandes sobresaltos, ya que estos sobresaltos sólo general gastos de energía e inexactitud. Por último, se traduce la función generada con el PI a una señal PWM, la cual activa los relés para activar el motor.

Con los diagramas explicados, se explica a continuación las funciones más importantes del programa:

1. *fillTxd_Buffer_DynamischWerte* y *fillTxd_Buffer_Statisch:*

   Estas dos funciones modifican un vector declarado en la función main. Este vector está diseñado para mostrar en pantalla los valores que se desean, explicados anteriormente, *fillTxd_Buffer_Statitisch* genera en las posiciones del vector siempre los mismos datos en hexadecimal, el ordenador los traduce a ASCII y por lo tanto muestra en pantalla palabras que un usuario puede entender. El *fillTxd_Buffer_DynamischWerte* genera los datos que proporciona los filtros, de esta forma el vector quedará por completo escrito en hexadecimal, al ser traducido a ASCII el vector mostrará en tiempo real el comportamiento del filtro de la siguiente forma, donde 9999 equivale al número en ese momento:

   run.. >
   mVolt = 9999
   Vrms = 9999
   Messfehlr= 9999
   Frequenz= 9999

2. *ADC_StopStart:*

   Esta función desactiva momentáneamente el convertidor analógico-digital para r*esetear el índice de la matriz del convertidor analógico digital, esto se hace para tener una lectura fiable del convertidor ADC y que no se sobrescriban datos.*

3. *ADC_SaveADCData:*

Esta función transfiere los datos almacenados en la matriz de la interrupción a una matriz idéntica. En el mismo bucle se hace una operación de bits a la función, que busca los 4 primeros bits de la cadena, en estos bits está la codificación del canal, como usamos dos señales, solamente buscamos una de ellas, si no es la que buscamos, por lógica, tiene que ser la otra, esta es la razón del uso del "else". El microcontrolador se especifica para leer dos señales, por eso se puede asegurar que no va a haber ningún error de lectura. Tras haber decidido el canal de cada una, se clasifican en el vector correspondiente y mediante una operación de bits se dejan los valores del convertidor analógico digital.

4. *Filter:*

Esta función genera los dos pasos centrales del programa, filtra las señales y las compara. El filtrado de la señal se ha programado teniendo en cuenta la función donde A(t) es la $A(t)=K_p K_o AdcChXBuf(t)temp+K_o A(t-1)temp1$ salida, Kp la constante, AdcChXBuf(t) los valores AdcCh1Buf y AdcCh0Buf en cierta iteración, temp ytemp1 constante de tiempo, estos valores son lo que mejor comportamiento han tenido en el laboratorio. El tiempo (t) y (t-1) que aparece en la ecuación no es más que la iteración actual para (t) y la iteración anterior para (t-1). La constante K0 equivale a la ecuación $K_0=\frac{1}{temp0+temp1}$

Tras esta operación, se comparan dos cosas, el valor máximo de las señales y la frecuencia de las mismas. El convertidor analógico digital, para un voltaje de 0 voltios, tiene un valor de 0 y para un voltaje de 24 voltios, que es el máximo que soporta el microcontrolador, tiene un valor de 1024, en función de esto, es fácil afirmar que la resolución del convertidor es 23mV, ya que 24V/1024=0,023V. Cuando se tienen todos estos datos se busca la frecuencia de cada señal para compararla, de esta manera se puede saber la diferencia entre ambas señales, y el desplazamiento, en caso de que las frecuencias fuesen iguales. Para saber el valor de esas frecuencias se busca el número de iteraciones desde que la onda aumenta hasta

que la onda vuelve a disminuir, aquí es donde se ve la importancia del filtro, si no existe ningún tipo de filtro, al haber un pequeño cambio en la onda, no se podría saber el valor exacto de la frecuencia, sin embargo gracias al filtro se soluciona ese problema. La figura 7 muestra el comportamiento de la onda, la parte azul es en la que se toman las lecturas, al terminar, se multiplica por dos y se obtienen el valor de la frecuencia.

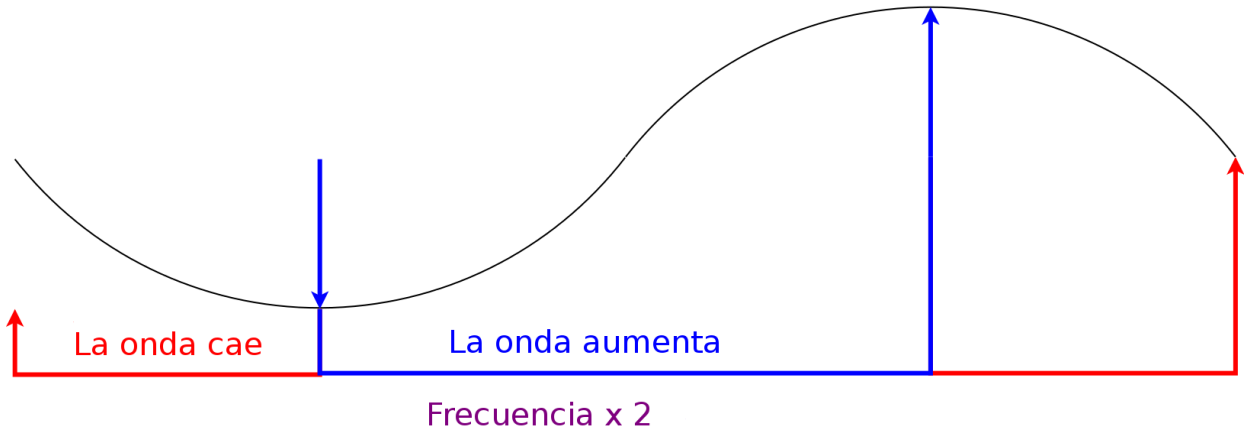


Figura 7: *Comportamiento de la onda*

5. *PIKontroler:*

Teniendo ya todos los datos del filtro se aplica el control proporcional integral, que corresponde con la siguiente función:

Donde Kc, T y Ti son tiempos definidos para optimizar la función.

$$PI(t)=P(t)+I(t)=K_c\Theta(t)(1+\frac{T}{2T_i})-K_c\Theta(t-1)(1-\frac{T}{2T_i})+PI(t-1)$$

6. *PWMpulse:*

Es la salida del pulso PWM, se establece los valores mediante funciones if, estos valores se han buscado para optimizar la onda lo máximo posible. Si los valores de salida del PI son negativos, la salida será más lenta que si los valores son positivos, esto se puede ver en las figuras 8 y 9.

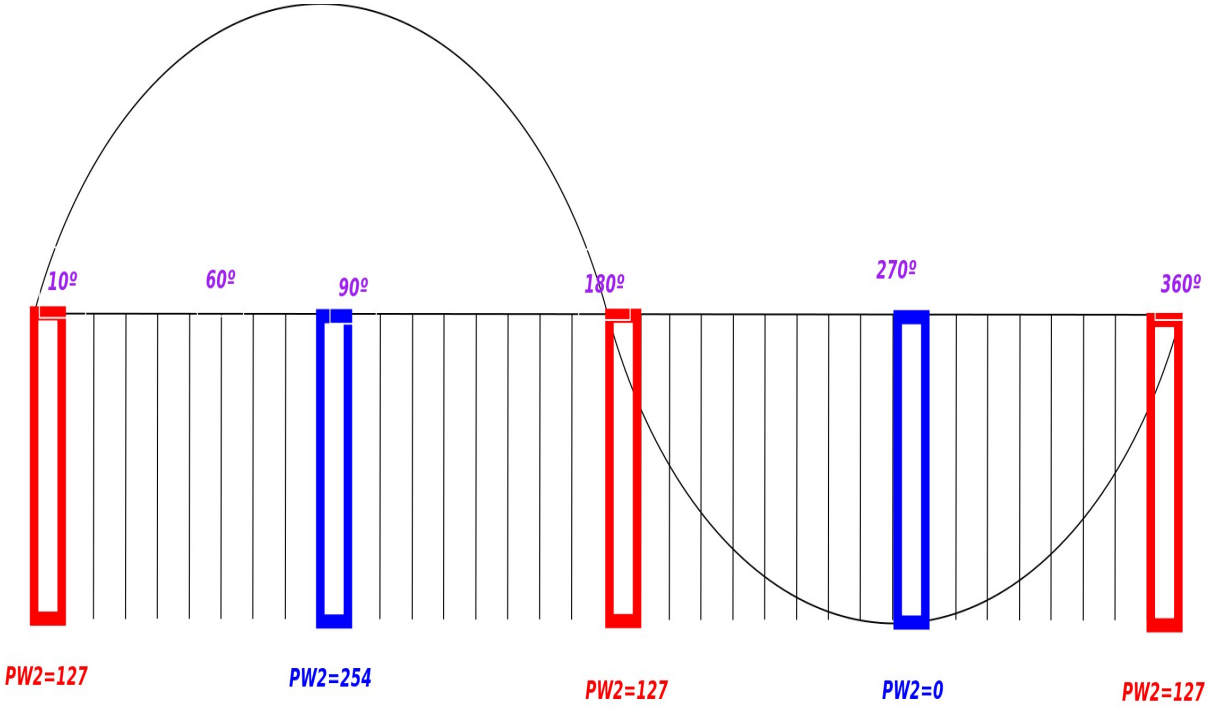


Figura 8: *Grados necesarios para generar la función con el PWM*

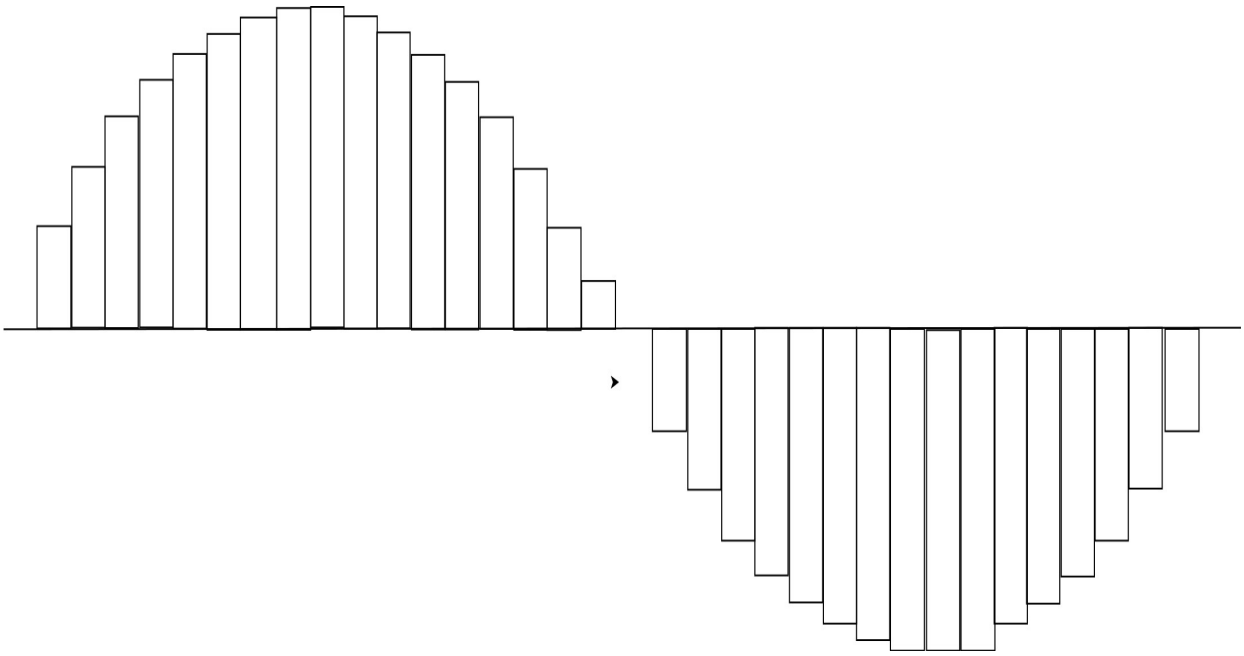


Figura 9: *Salida real del pulso PWM*

Limitaciones:

En este proyecto la limitaciones se basan en la frecuencia que puede leer el microcontrolador, si las frecuencias son muy altas, al microcontrolador le costará mucho detectar con cierta exactitud el aumento o la disminución de la onda, esto generaría un calculo incorrecto de la frecuencia y por lo tanto un error en la velocidad del motor. De igual forma, para frecuencias pequeñas el contador se desborda, esto se debe a que se desborda el contador máximo establecido de la semionda, la lectura será el máximo valor del contador.

Además de los problemas de frecuencia, como todo microcontrolador, las entradas tanto analógicas como digitales tienen que estar comprendidas entre 0 y 24 voltios en caso de el Infineon C167, a valores negativos, el microcontrolador puede estropearse, esto se debe a que los componentes del microcontrolador no están pensados para aguantar ningún tipo de corriente inversa. Para transformar la onda de corriente alterna del generador síncrono hay que escalarla a valores máximos de ±12V, para compensarla hay que sumarle una corriente continua de 12V positivos, de esta forma transformamos una corriente alterna de paso por cero en una corriente continua de valores entre 0V y 24V.

El PWM tiene que ir conectado a dispositivos lo suficientemente rápidos y que se disparen mediante tensión, a su vez, estos dispositivos tienen que poder aguantar grandes tensiones y corrientes, dependiendo siempre de la carga que se quiere controlar. Los elementos más recomendables son los dispositivos MOSFET y los dispositivos IGBTs, estos dispositivos se activan por tensión. Para frecuencias y tensiones elevadas, donde puede haber corrientes de encendido altas, es recomendable usar el dispositivo IGBT.

Conclusión del proyecto:

Gracias a las capacidades de procesos y a las velocidades de los microcontroladores se puede hacer sistemas de control mucho más eficientes. Anteriormente existían muchas dificultades para usar generadores síncronos de corriente alterna ya que la velocidad de estos motores depende de la frecuencia, como se ha demostrado en este proyecto, el manejo de microcontroladores da un paso más allá en la variación simple de frecuencia con los pulsos PWM, creando un sistema de realimentación que examina constantemente el comportamiento del motor.

Se puede afirmar que utilizar microcontroladores para el control de motores abarata los costes, mejora la calidad, hace más eficiente el sistema y genera una mayor versatilidad para usar motores de corriente alterna. Como ejemplo de esta mejora se puede decir que este nuevo tipo de control se está aplicando en electrodomésticos, maquinaria agrícola, ascensores, etc.

Institut für Elektrotechnik

Bachelor Abschlussarbeit

Entwicklung eines digitalen Reglers für Synchrongeneratoren

Manuel Silva González

Inhaltverzichnis

1 Analyse der vorhandenen Hardware und Software

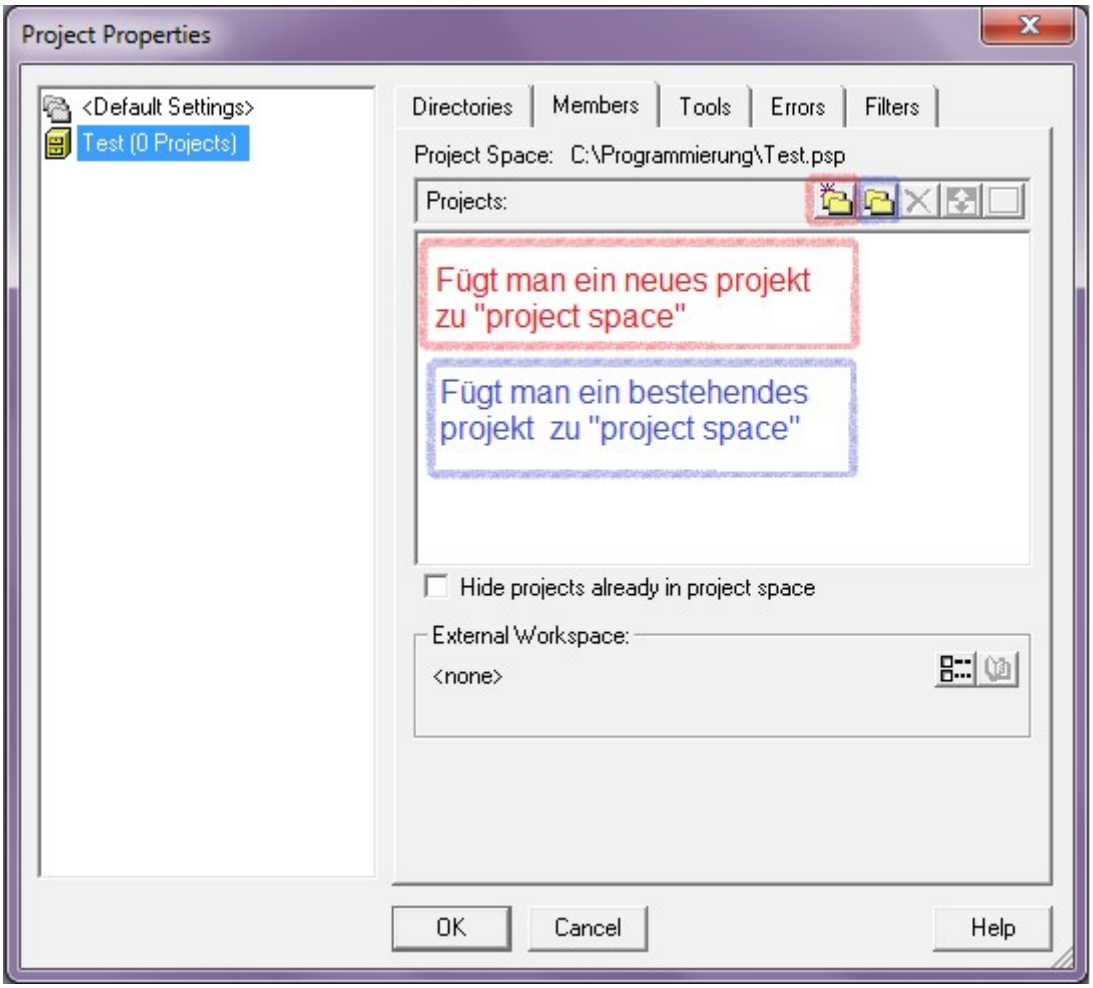
ю Tasking C166/ST10

Erklärung der Mikrorechner, wie das funktioniert und so weiter und sofort.

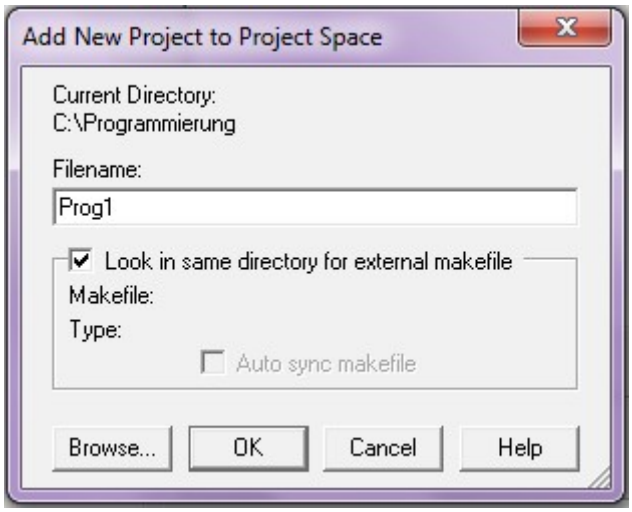
Logo von TASKING programm.

Man Erstellt ein neues Projekt mit Test als Name.

Man drückt „OK“, danach erscheint das folgende Fenster:

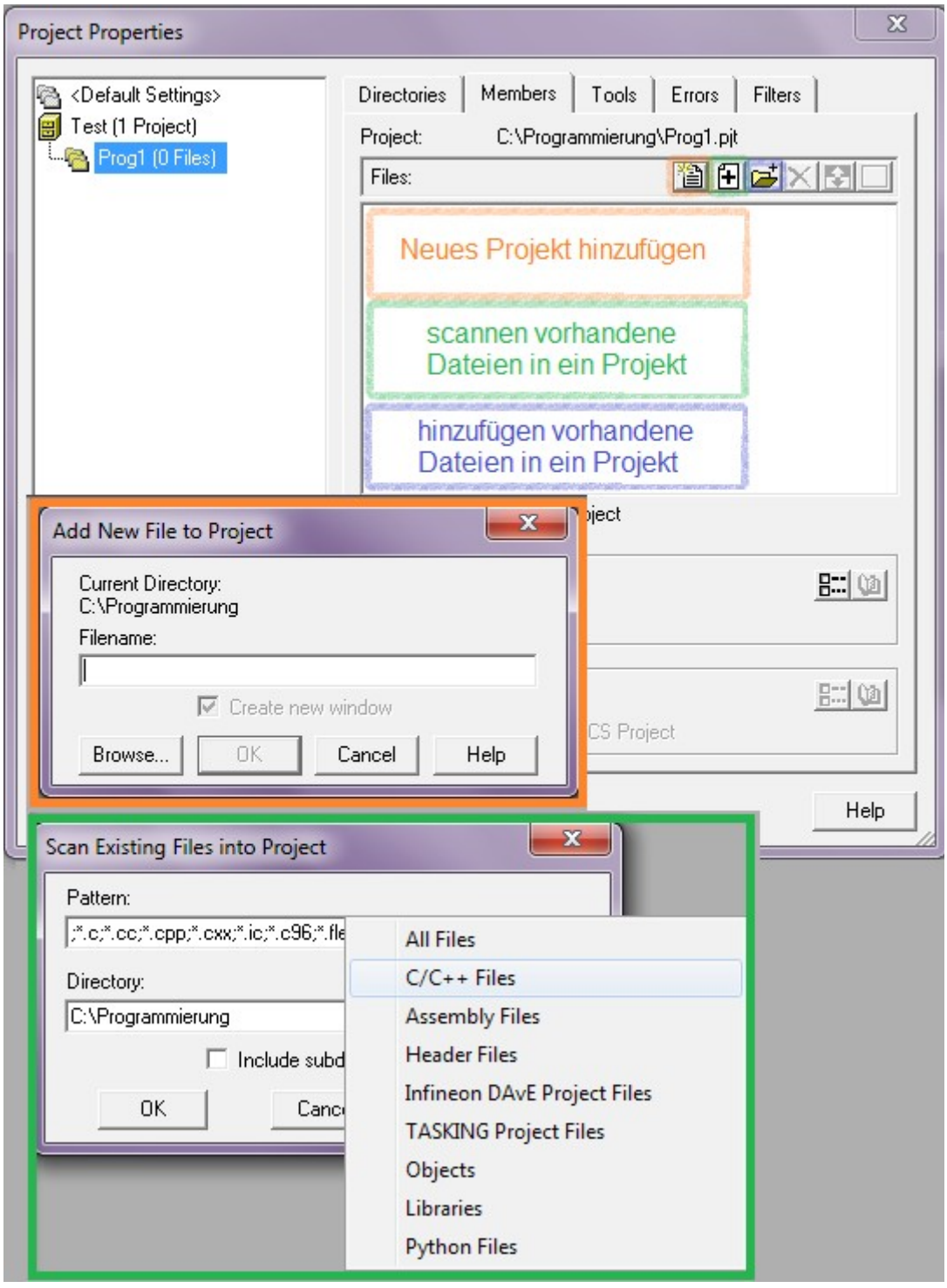


Wenn man ein neues Projekt machen will, muss man der folgende Bildschirm weitermachen, in diesem Fall wird das Projekt mit dem Name „Prog1“ gebildet.



Man schreibt ein Name oder sucht ein bestehendes Projekt und drückt „OK“.

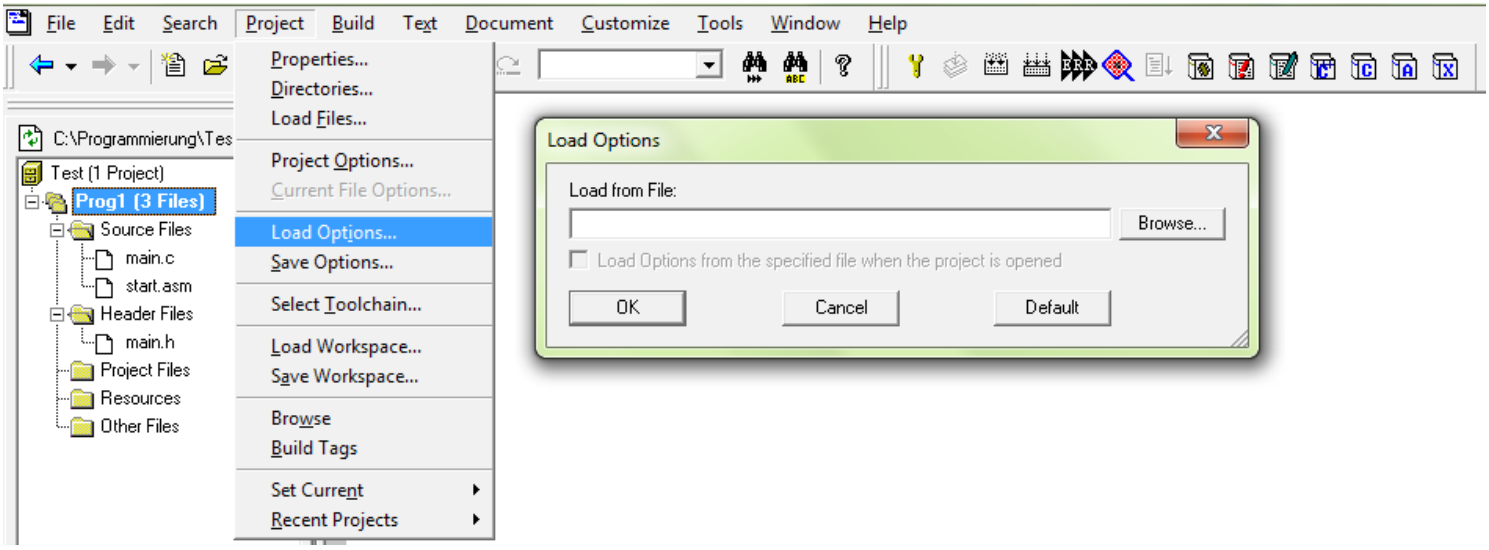
vorhandene Dateien in ein Projekt scannen

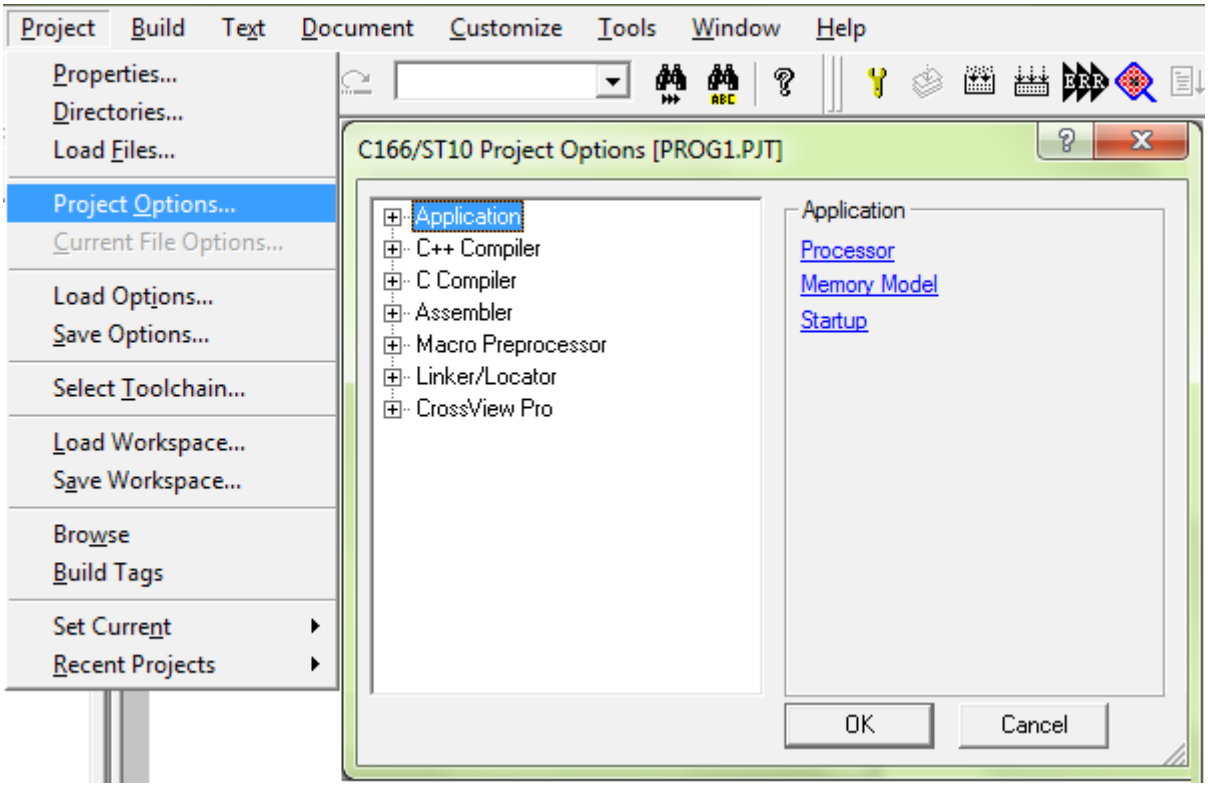


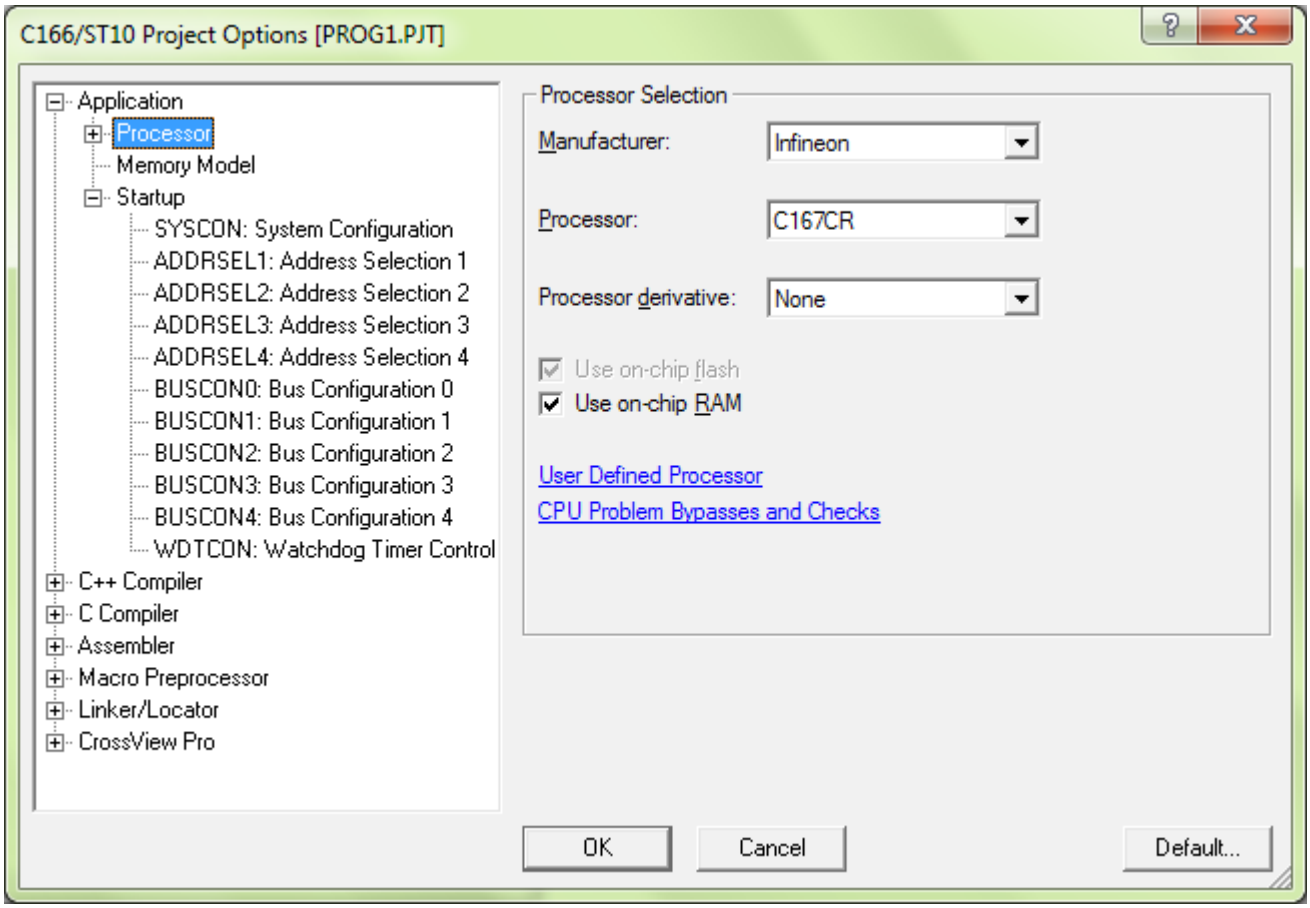
Damit hat man die Umgebung, um zu arbeiten, da kann man einige Dateien benutzen:

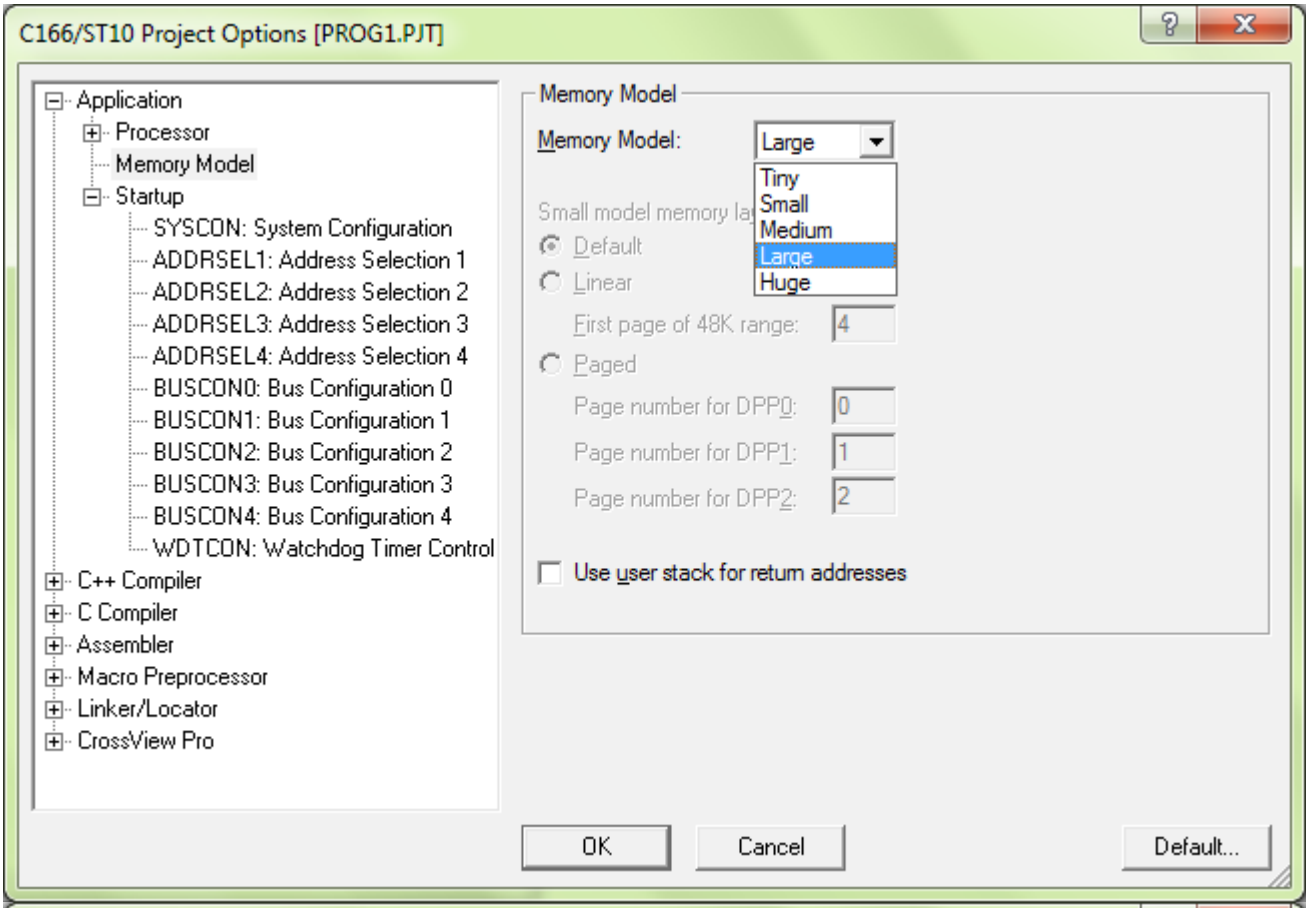
- Datei.c → Programmierdatei
- Datei.h → Header-datei
- Datei.asm → assembler-datei
- Datei.map → Auswertungsdatei

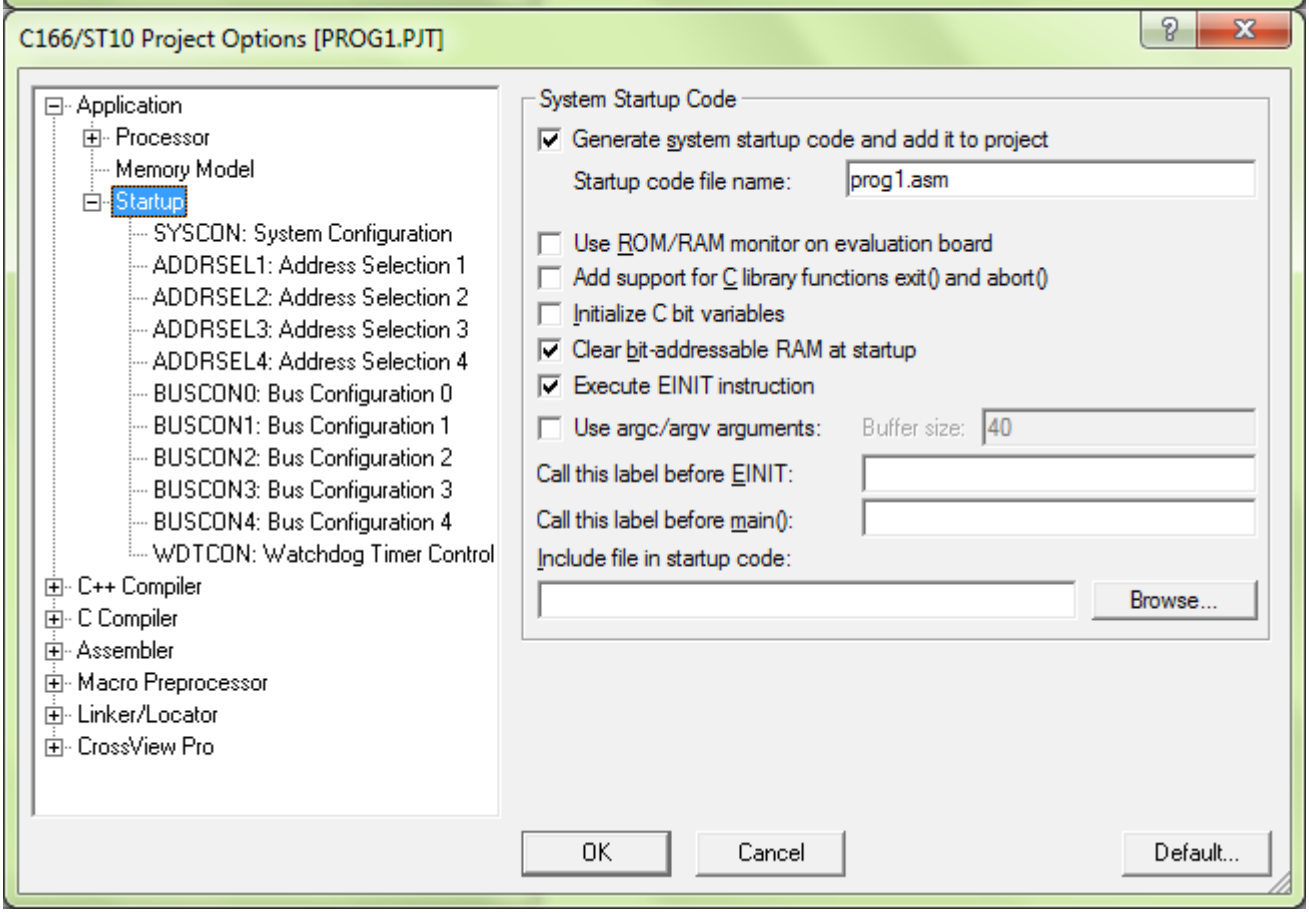
Die Programmierdateien und die Header-dateien erklären sich seit dem zweiten Absatz, im Folgenden erklärt man die assembler-dateien für die Version 8.9 und die Auswertung der Memory Maps

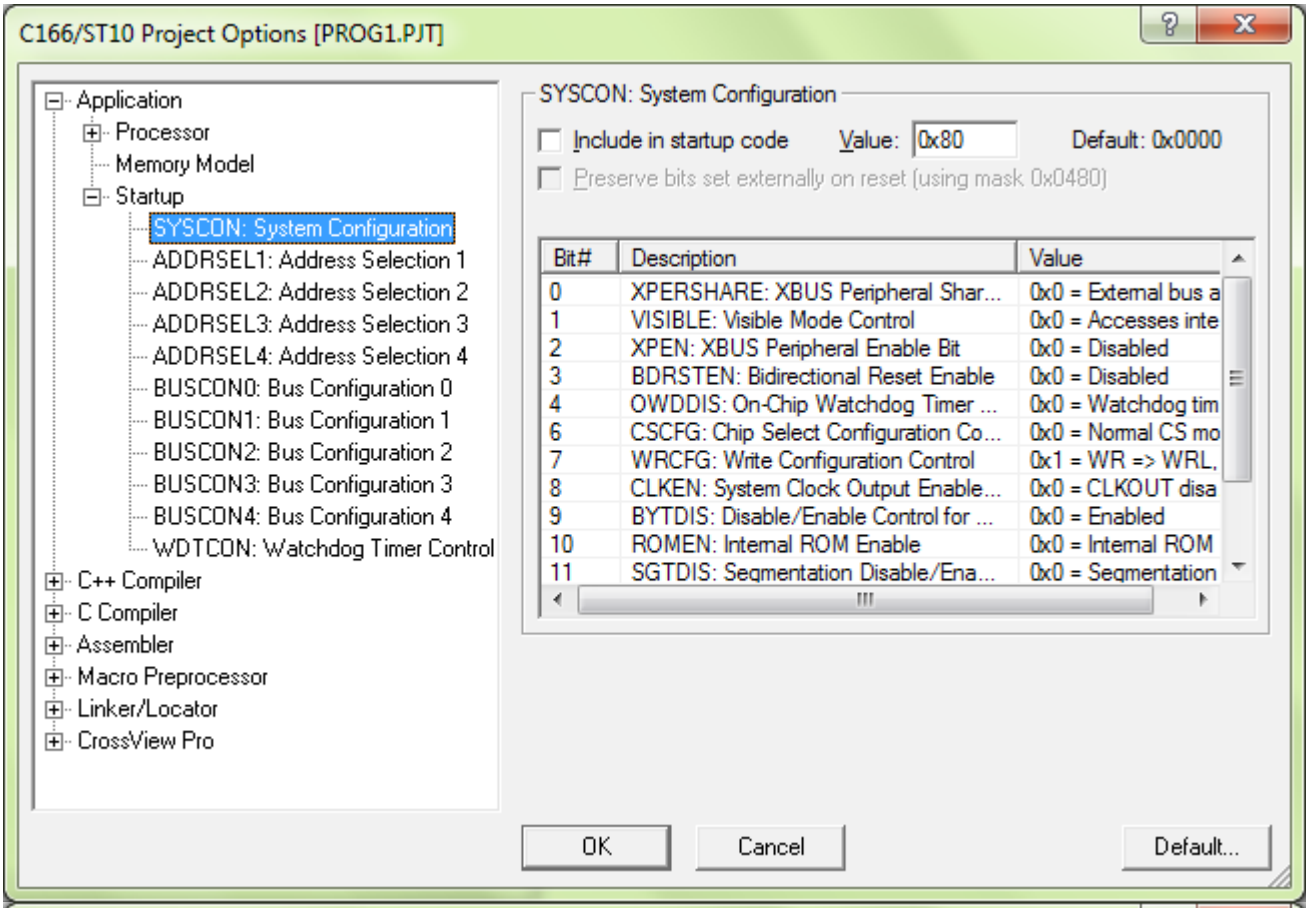


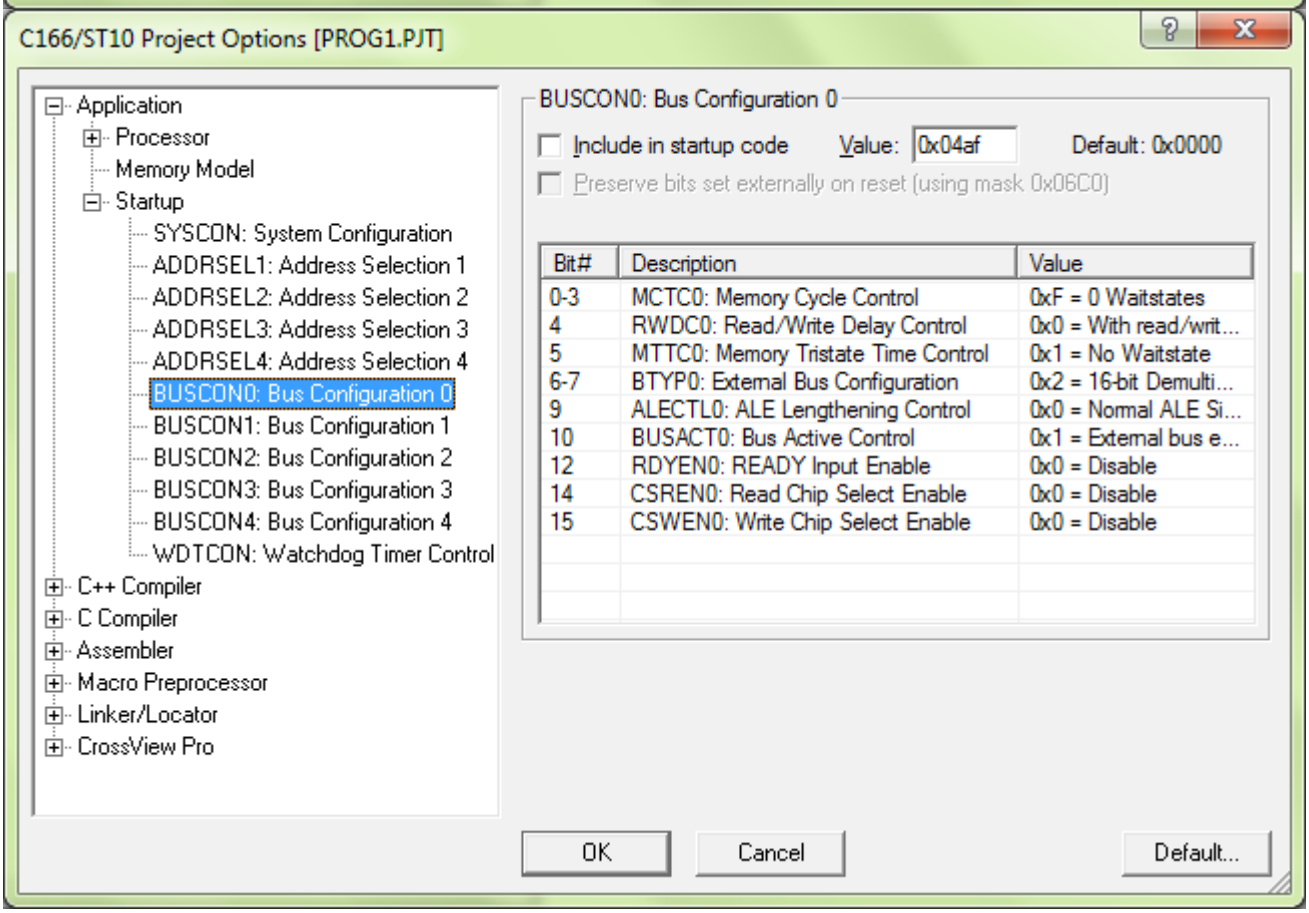

Project
Build
Text
Document
Customize
Tools
Window
Help
Properties...
Directories...
Load Files...
Project Options...
Current File Options...
Load Options...
Save Options...
Select Toolchain...
Load Workspace...
Save Workspace...
Browse
Build Tags
Set Current
Recent Projects
C166/ST10 Project Options [PROG1.PJT]
Application
C++ Compiler
C Compiler
Assembler
Macro Preprocessor
Linker/Locator
CrossView Pro
Application
Processor
Memory Model
Startup
OK
Cancel

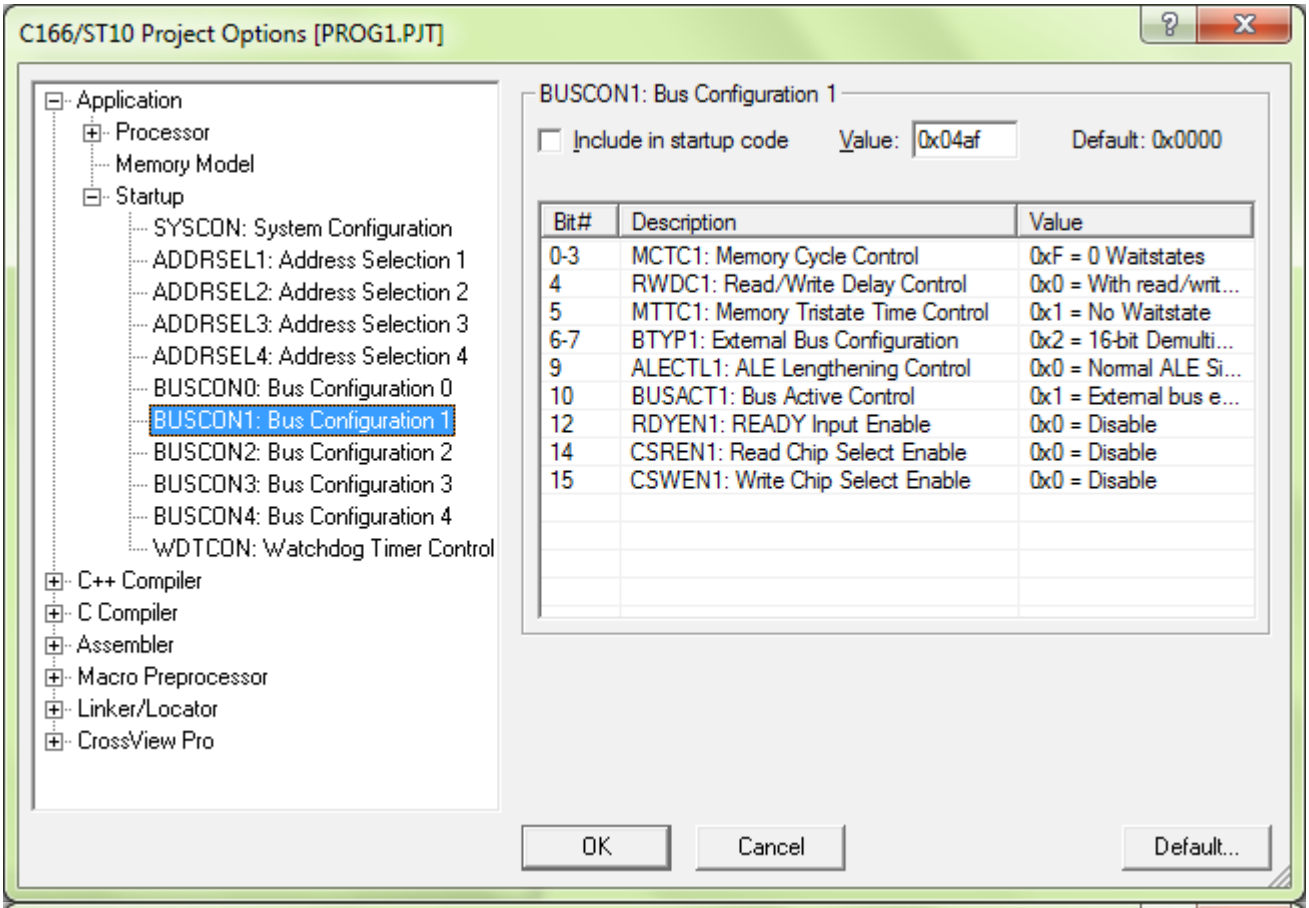

C166/ST10 Project Options [PROG1.PJT]
Application
Processor
Memory Model
Startup
SYSCON: System Configuration
ADDRSEL1: Address Selection 1
ADDRSEL2: Address Selection 2
ADDRSEL3: Address Selection 3
ADDRSEL4: Address Selection 4
BUSCON0: Bus Configuration 0
BUSCON1: Bus Configuration 1
BUSCON2: Bus Configuration 2
BUSCON3: Bus Configuration 3
BUSCON4: Bus Configuration 4
WDTCON: Watchdog Timer Control
C++ Compiler
C Compiler
Assembler
Macro Preprocessor
Linker/Locator
CrossView Pro
Processor Selection
Manufacturer:
Infineon
Processor:
C167CR
Processor derivative:
None
Use on-chip flash
Use on-chip RAM
User Defined Processor
CPU Problem Bypasses and Checks
OK
Cancel
Default...

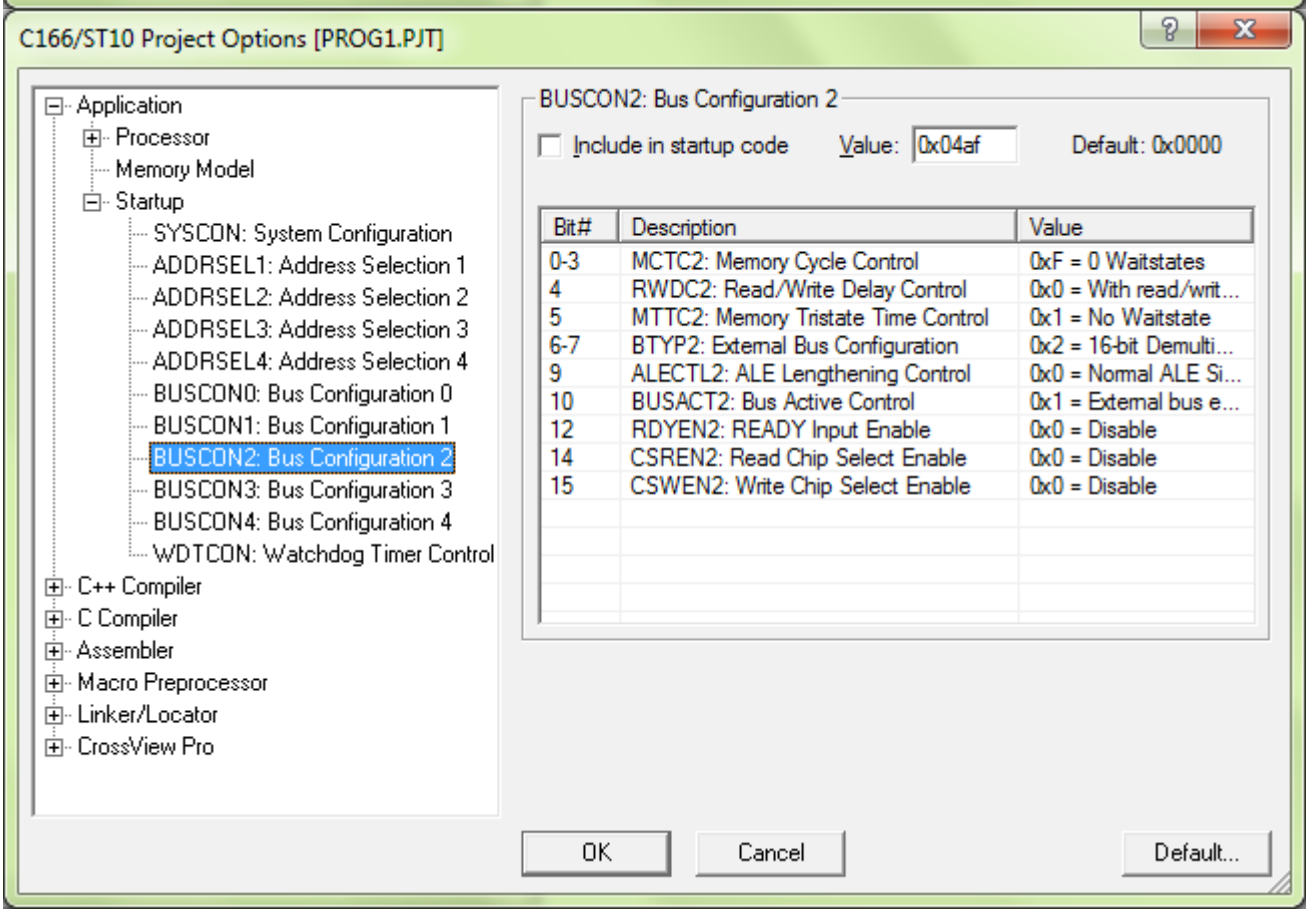

C166/ST10 Project Options [PROG1.PJT]
Application
Processor
Memory Model
Startup
SYSCON: System Configuration
ADDRSEL1: Address Selection 1
ADDRSEL2: Address Selection 2
ADDRSEL3: Address Selection 3
ADDRSEL4: Address Selection 4
BUSCON0: Bus Configuration 0
BUSCON1: Bus Configuration 1
BUSCON2: Bus Configuration 2
BUSCON3: Bus Configuration 3
BUSCON4: Bus Configuration 4
WDTCON: Watchdog Timer Control
C++ Compiler
C Compiler
Assembler
Macro Preprocessor
Linker/Locator
CrossView Pro
Memory Model
Memory Model:
Large
Tiny
Small
Medium
Large
Huge
Small model memory la
Default
Linear
First page of 48K range:
4
Paged
Page number for DPP0:
0
Page number for DPP1:
1
Page number for DPP2:
2
Use user stack for return addresses
OK
Cancel
Default...

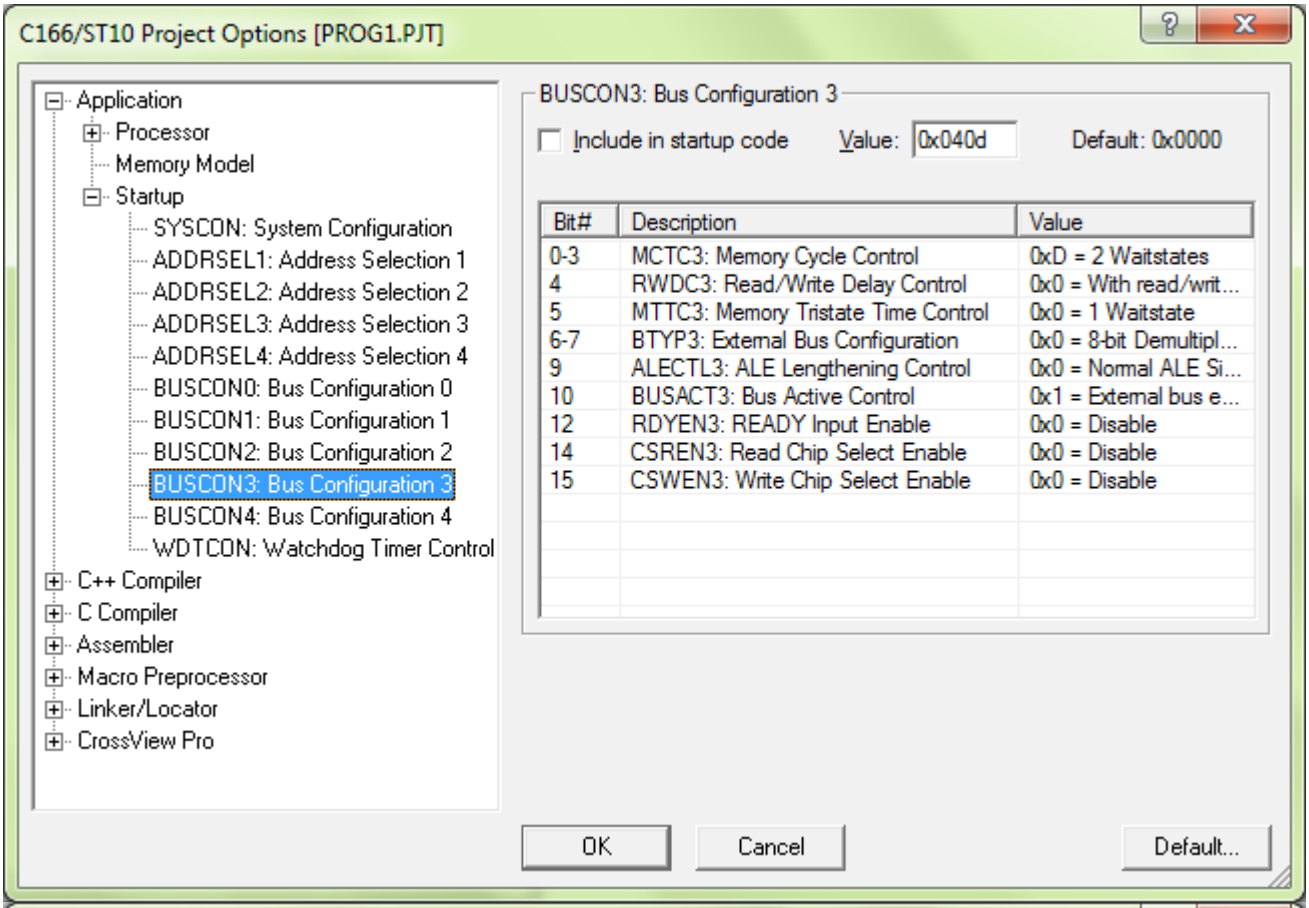

C166/ST10 Project Options [PROG1.PJT]
Application
Processor
Memory Model
Startup
SYSCON: System Configuration
ADDRSEL1: Address Selection 1
ADDRSEL2: Address Selection 2
ADDRSEL3: Address Selection 3
ADDRSEL4: Address Selection 4
BUSCON0: Bus Configuration 0
BUSCON1: Bus Configuration 1
BUSCON2: Bus Configuration 2
BUSCON3: Bus Configuration 3
BUSCON4: Bus Configuration 4
WDTCON: Watchdog Timer Control
C++ Compiler
C Compiler
Assembler
Macro Preprocessor
Linker/Locator
CrossView Pro
System Startup Code
Generate system startup code and add it to project
Startup code file name:
prog1.asm
Use ROM/RAM monitor on evaluation board
Add support for C library functions exit() and abort()
Initialize C bit variables
Clear bit-addressable RAM at startup
Execute EINIT instruction
Use argc/argv arguments:
Buffer size:
40
Call this label before EINIT:
Call this label before main():
Include file in startup code:
Browse...
OK
Cancel
Default...

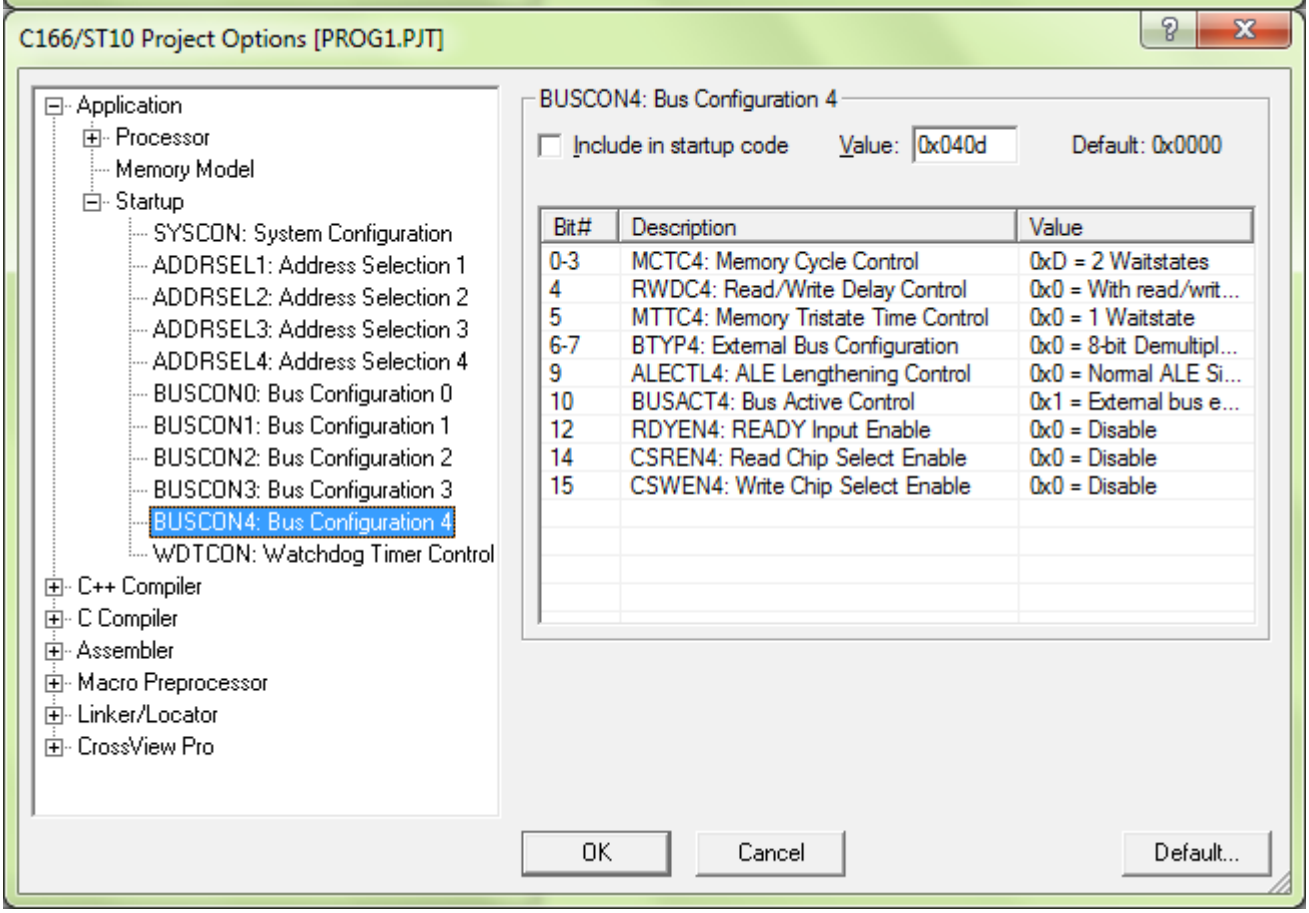

C166/ST10 Project Options [PROG1.PJT]
Application
Processor
Memory Model
Startup
SYSCON: System Configuration
ADDRSEL1: Address Selection 1
ADDRSEL2: Address Selection 2
ADDRSEL3: Address Selection 3
ADDRSEL4: Address Selection 4
BUSCON0: Bus Configuration 0
BUSCON1: Bus Configuration 1
BUSCON2: Bus Configuration 2
BUSCON3: Bus Configuration 3
BUSCON4: Bus Configuration 4
WDTCON: Watchdog Timer Control
C++ Compiler
C Compiler
Assembler
Macro Preprocessor
Linker/Locator
CrossView Pro
SYSCON: System Configuration
Include in startup code
Value: 0x80
Default: 0x0000
Preserve bits set externally on reset (using mask 0x0480)
Bit# Description Value
0 XPERSHARE: XBUS Peripheral Shar... 0x0 = External bus a
1 VISIBLE: Visible Mode Control 0x0 = Accesses inte
2 XPEN: XBUS Peripheral Enable Bit 0x0 = Disabled
3 BDRSTEN: Bidirectional Reset Enable 0x0 = Disabled
4 OWDDIS: On-Chip Watchdog Timer ... 0x0 = Watchdog tim
6 CSCFG: Chip Select Configuration Co... 0x0 = Normal CS mo
7 WRCFG: Write Configuration Control 0x1 = WR => WRL,
8 CLKEN: System Clock Output Enable... 0x0 = CLKOUT disa
9 BYTDIS: Disable/Enable Control for ... 0x0 = Enabled
10 ROMEN: Internal ROM Enable 0x0 = Internal ROM
11 SGTDIS: Segmentation Disable/Ena... 0x0 = Segmentation
OK
Cancel
Default...

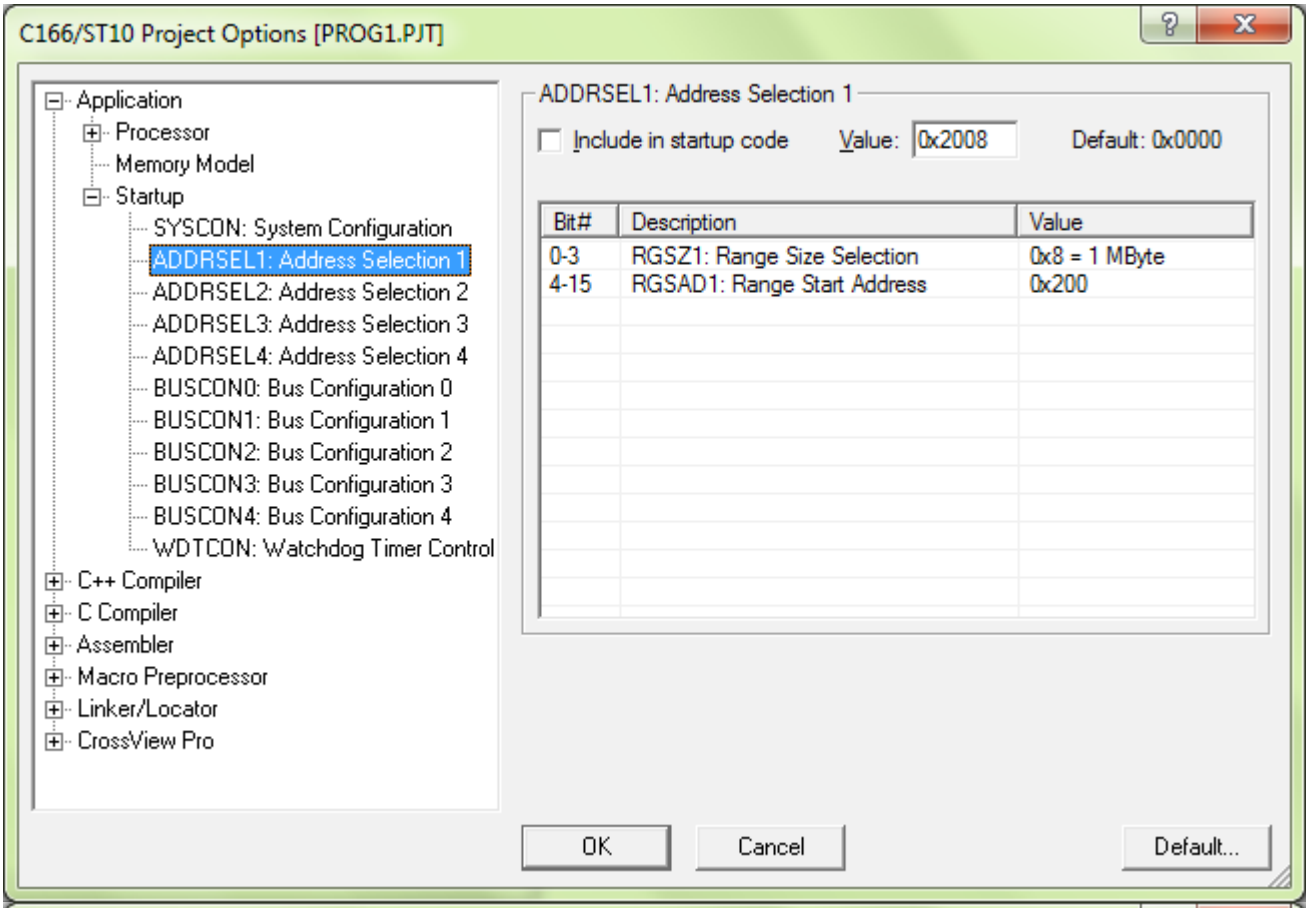

C166/ST10 Project Options [PROG1.PJT]
Application
Processor
Memory Model
Startup
SYSCON: System Configuration
ADDRSEL1: Address Selection 1
ADDRSEL2: Address Selection 2
ADDRSEL3: Address Selection 3
ADDRSEL4: Address Selection 4
BUSCON0: Bus Configuration 0
BUSCON1: Bus Configuration 1
BUSCON2: Bus Configuration 2
BUSCON3: Bus Configuration 3
BUSCON4: Bus Configuration 4
WDTCON: Watchdog Timer Control
C++ Compiler
C Compiler
Assembler
Macro Preprocessor
Linker/Locator
CrossView Pro
BUSCON0: Bus Configuration 0
Include in startup code
Value: 0x04af
Default: 0x0000
Preserve bits set externally on reset (using mask 0x06C0)
Bit# Description Value
0-3 MCTC0: Memory Cycle Control 0xF = 0 Waitstates
4 RWDC0: Read/Write Delay Control 0x0 = With read/writ...
5 MTTC0: Memory Tristate Time Control 0x1 = No Waitstate
6-7 BTYP0: External Bus Configuration 0x2 = 16-bit Demulti...
9 ALECTL0: ALE Lengthening Control 0x0 = Normal ALE Si...
10 BUSACT0: Bus Active Control 0x1 = External bus e...
12 RDYEN0: READY Input Enable 0x0 = Disable
14 CSREN0: Read Chip Select Enable 0x0 = Disable
15 CSWEN0: Write Chip Select Enable 0x0 = Disable
OK
Cancel
Default...

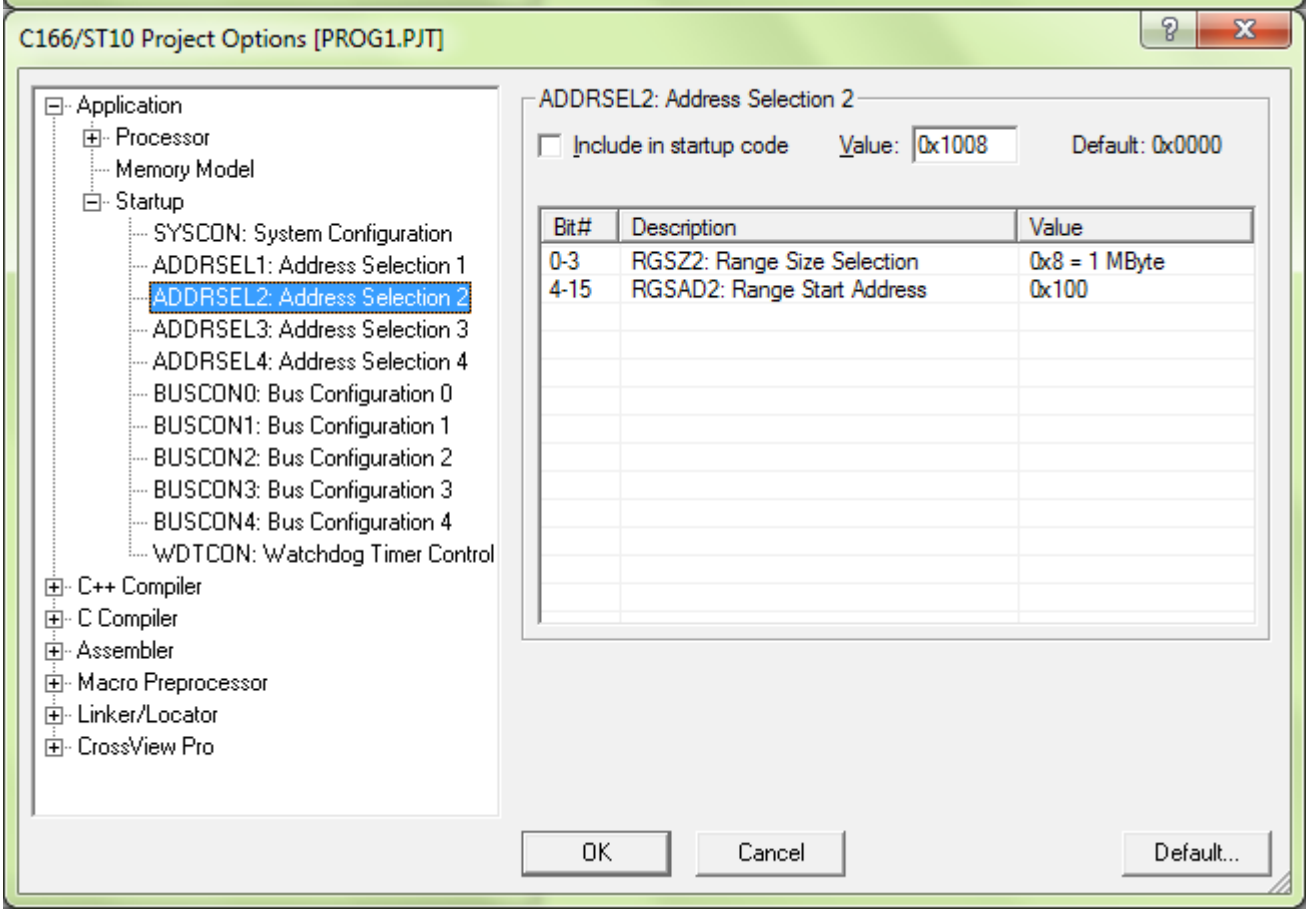

C166/ST10 Project Options [PROG1.PJT]
Application
Processor
Memory Model
Startup
SYSCON: System Configuration
ADDRSEL1: Address Selection 1
ADDRSEL2: Address Selection 2
ADDRSEL3: Address Selection 3
ADDRSEL4: Address Selection 4
BUSCON0: Bus Configuration 0
BUSCON1: Bus Configuration 1
BUSCON2: Bus Configuration 2
BUSCON3: Bus Configuration 3
BUSCON4: Bus Configuration 4
WDTCON: Watchdog Timer Control
C++ Compiler
C Compiler
Assembler
Macro Preprocessor
Linker/Locator
CrossView Pro
BUSCON1: Bus Configuration 1
Include in startup code
Value: 0x04af
Default: 0x0000
Bit# Description Value
0-3 MCTC1: Memory Cycle Control 0xF = 0 Waitstates
4 RWDC1: Read/Write Delay Control 0x0 = With read/writ...
5 MTTC1: Memory Tristate Time Control 0x1 = No Waitstate
6-7 BTYP1: External Bus Configuration 0x2 = 16-bit Demulti...
9 ALECTL1: ALE Lengthening Control 0x0 = Normal ALE Si...
10 BUSACT1: Bus Active Control 0x1 = External bus e...
12 RDYEN1: READY Input Enable 0x0 = Disable
14 CSREN1: Read Chip Select Enable 0x0 = Disable
15 CSWEN1: Write Chip Select Enable 0x0 = Disable
OK
Cancel
Default...

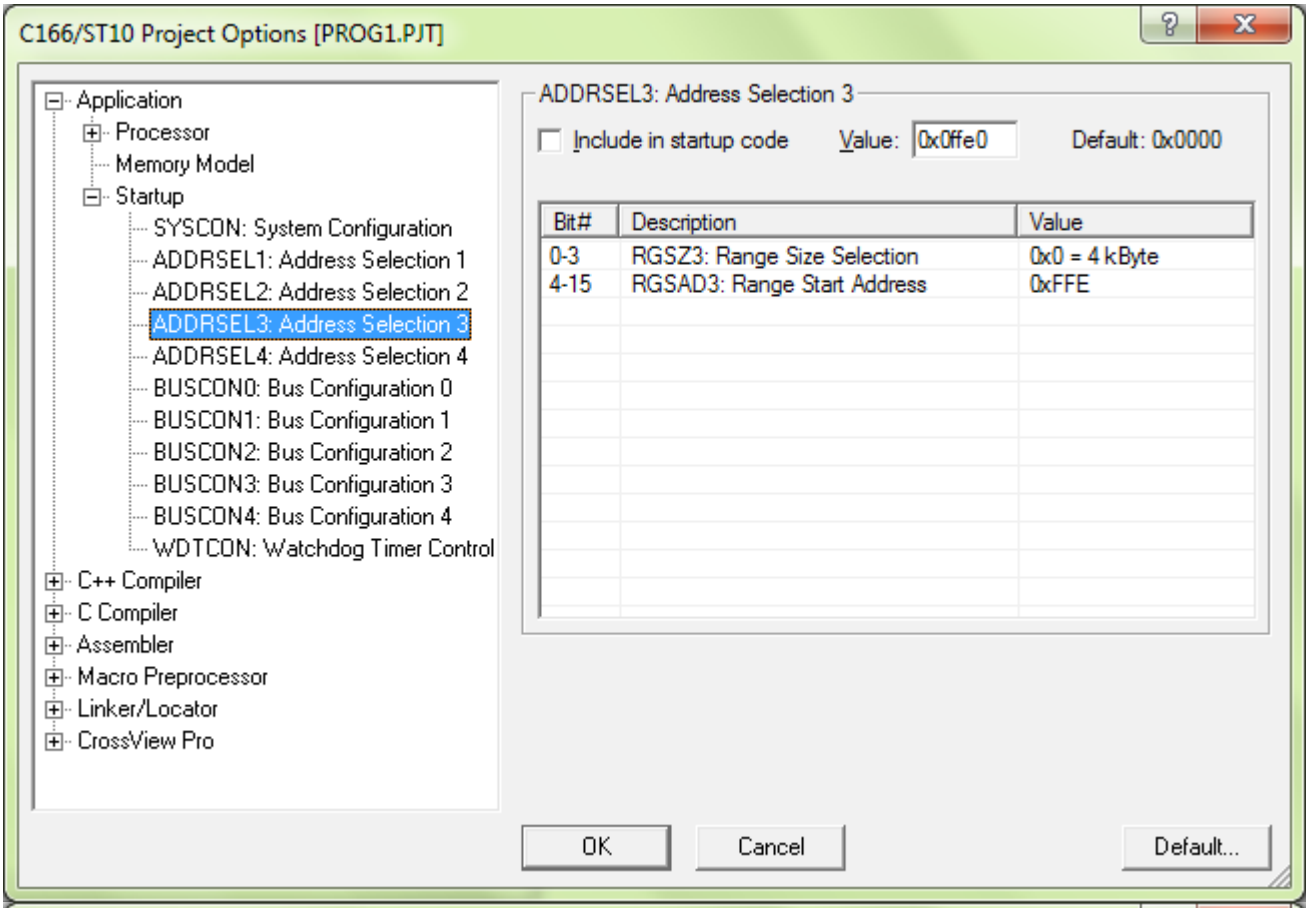

C166/ST10 Project Options [PROG1.PJT]
Application
Processor
Memory Model
Startup
SYSCON: System Configuration
ADDRSEL1: Address Selection 1
ADDRSEL2: Address Selection 2
ADDRSEL3: Address Selection 3
ADDRSEL4: Address Selection 4
BUSCON0: Bus Configuration 0
BUSCON1: Bus Configuration 1
BUSCON2: Bus Configuration 2
BUSCON3: Bus Configuration 3
BUSCON4: Bus Configuration 4
WDTCON: Watchdog Timer Control
C++ Compiler
C Compiler
Assembler
Macro Preprocessor
Linker/Locator
CrossView Pro
BUSCON2: Bus Configuration 2
Include in startup code
Value: 0x04af
Default: 0x0000
Bit# Description Value
0-3 MCTC2: Memory Cycle Control 0xF = 0 Waitstates
4 RWDC2: Read/Write Delay Control 0x0 = With read/writ...
5 MTTC2: Memory Tristate Time Control 0x1 = No Waitstate
6-7 BTYP2: External Bus Configuration 0x2 = 16-bit Demulti...
9 ALECTL2: ALE Lengthening Control 0x0 = Normal ALE Si...
10 BUSACT2: Bus Active Control 0x1 = External bus e...
12 RDYEN2: READY Input Enable 0x0 = Disable
14 CSREN2: Read Chip Select Enable 0x0 = Disable
15 CSWEN2: Write Chip Select Enable 0x0 = Disable
OK
Cancel
Default...

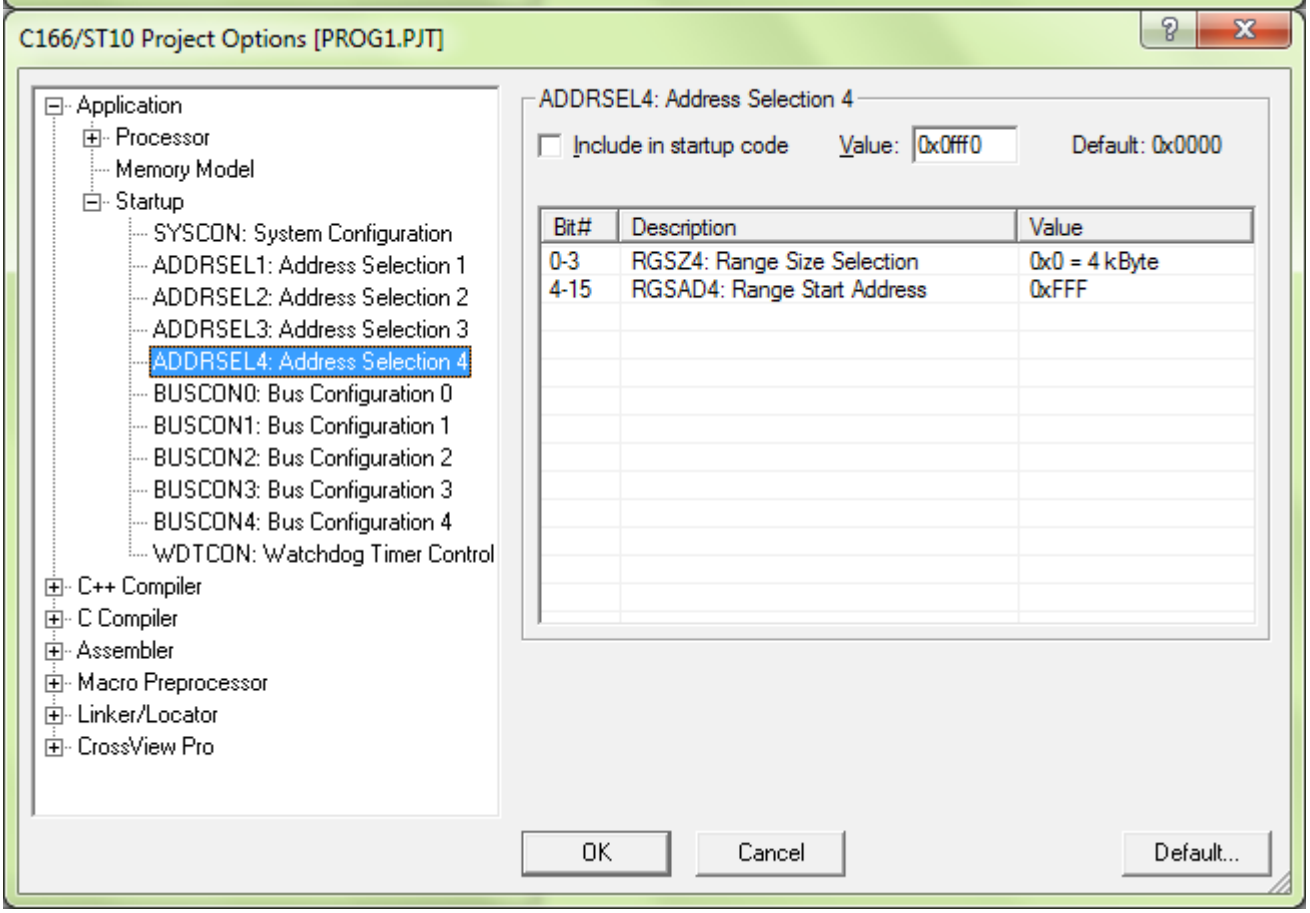

C166/ST10 Project Options [PROG1.PJT]
Application
Processor
Memory Model
Startup
SYSCON: System Configuration
ADDRSEL1: Address Selection 1
ADDRSEL2: Address Selection 2
ADDRSEL3: Address Selection 3
ADDRSEL4: Address Selection 4
BUSCON0: Bus Configuration 0
BUSCON1: Bus Configuration 1
BUSCON2: Bus Configuration 2
BUSCON3: Bus Configuration 3
BUSCON4: Bus Configuration 4
WDTCON: Watchdog Timer Control
C++ Compiler
C Compiler
Assembler
Macro Preprocessor
Linker/Locator
CrossView Pro
BUSCON3: Bus Configuration 3
Include in startup code
Value: 0x040d
Default: 0x0000
Bit# Description Value
0-3 MCTC3: Memory Cycle Control 0xD = 2 Waitstates
4 RWDC3: Read/Write Delay Control 0x0 = With read/writ...
5 MTTC3: Memory Tristate Time Control 0x0 = 1 Waitstate
6-7 BTYP3: External Bus Configuration 0x0 = 8-bit Demultipl...
9 ALECTL3: ALE Lengthening Control 0x0 = Normal ALE Si...
10 BUSACT3: Bus Active Control 0x1 = External bus e...
12 RDYEN3: READY Input Enable 0x0 = Disable
14 CSREN3: Read Chip Select Enable 0x0 = Disable
15 CSWEN3: Write Chip Select Enable 0x0 = Disable
OK
Cancel
Default...

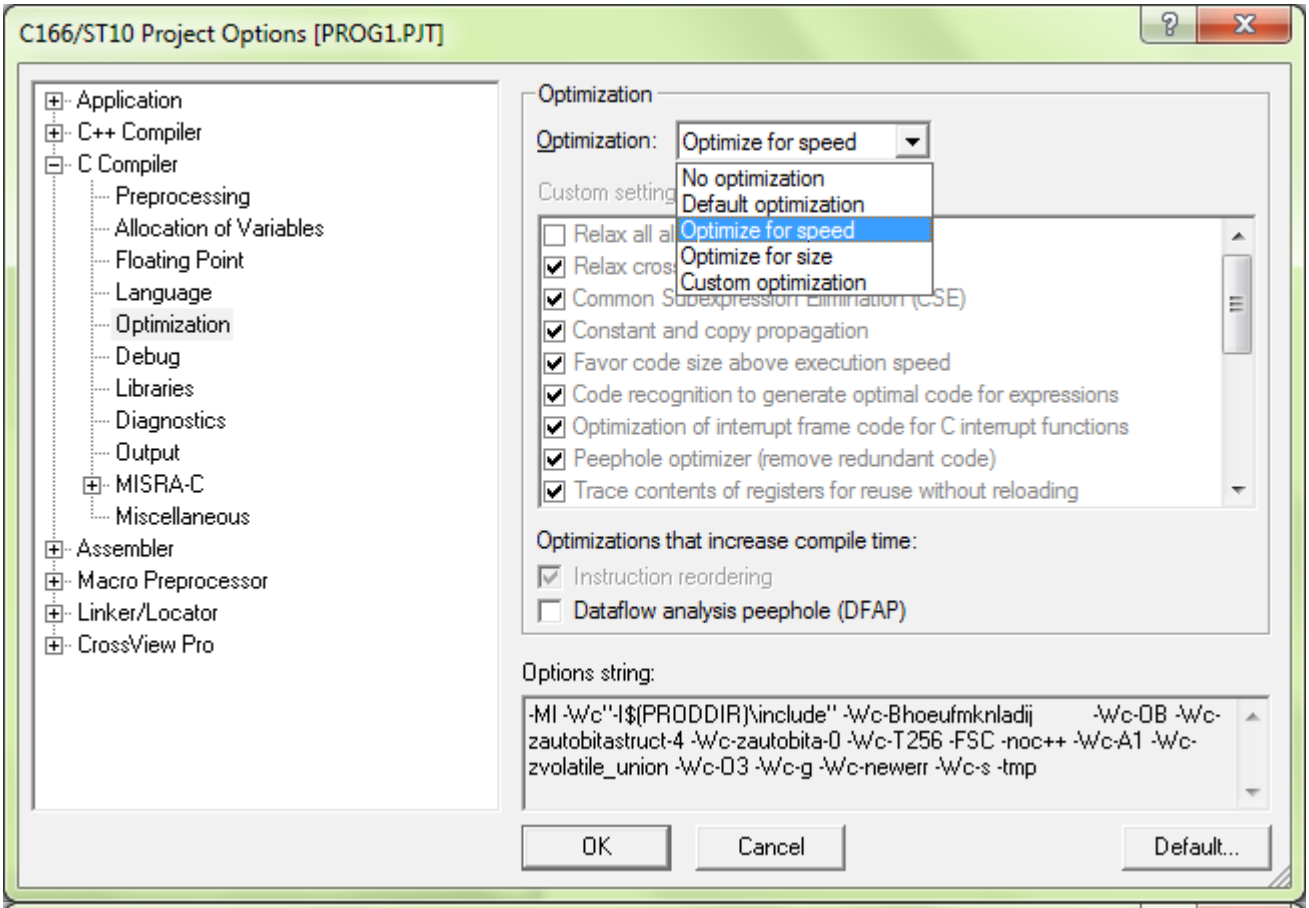

C166/ST10 Project Options [PROG1.PJT]
Application
Processor
Memory Model
Startup
SYSCON: System Configuration
ADDRSEL1: Address Selection 1
ADDRSEL2: Address Selection 2
ADDRSEL3: Address Selection 3
ADDRSEL4: Address Selection 4
BUSCON0: Bus Configuration 0
BUSCON1: Bus Configuration 1
BUSCON2: Bus Configuration 2
BUSCON3: Bus Configuration 3
BUSCON4: Bus Configuration 4
WDTCON: Watchdog Timer Control
C++ Compiler
C Compiler
Assembler
Macro Preprocessor
Linker/Locator
CrossView Pro
BUSCON4: Bus Configuration 4
Include in startup code
Value: 0x040d
Default: 0x0000
Bit# Description Value
0-3 MCTC4: Memory Cycle Control 0xD = 2 Waitstates
4 RWDC4: Read/Write Delay Control 0x0 = With read/writ...
5 MTTC4: Memory Tristate Time Control 0x0 = 1 Waitstate
6-7 BTYP4: External Bus Configuration 0x0 = 8-bit Demultipl...
9 ALECTL4: ALE Lengthening Control 0x0 = Normal ALE Si...
10 BUSACT4: Bus Active Control 0x1 = External bus e...
12 RDYEN4: READY Input Enable 0x0 = Disable
14 CSREN4: Read Chip Select Enable 0x0 = Disable
15 CSWEN4: Write Chip Select Enable 0x0 = Disable
OK
Cancel
Default...

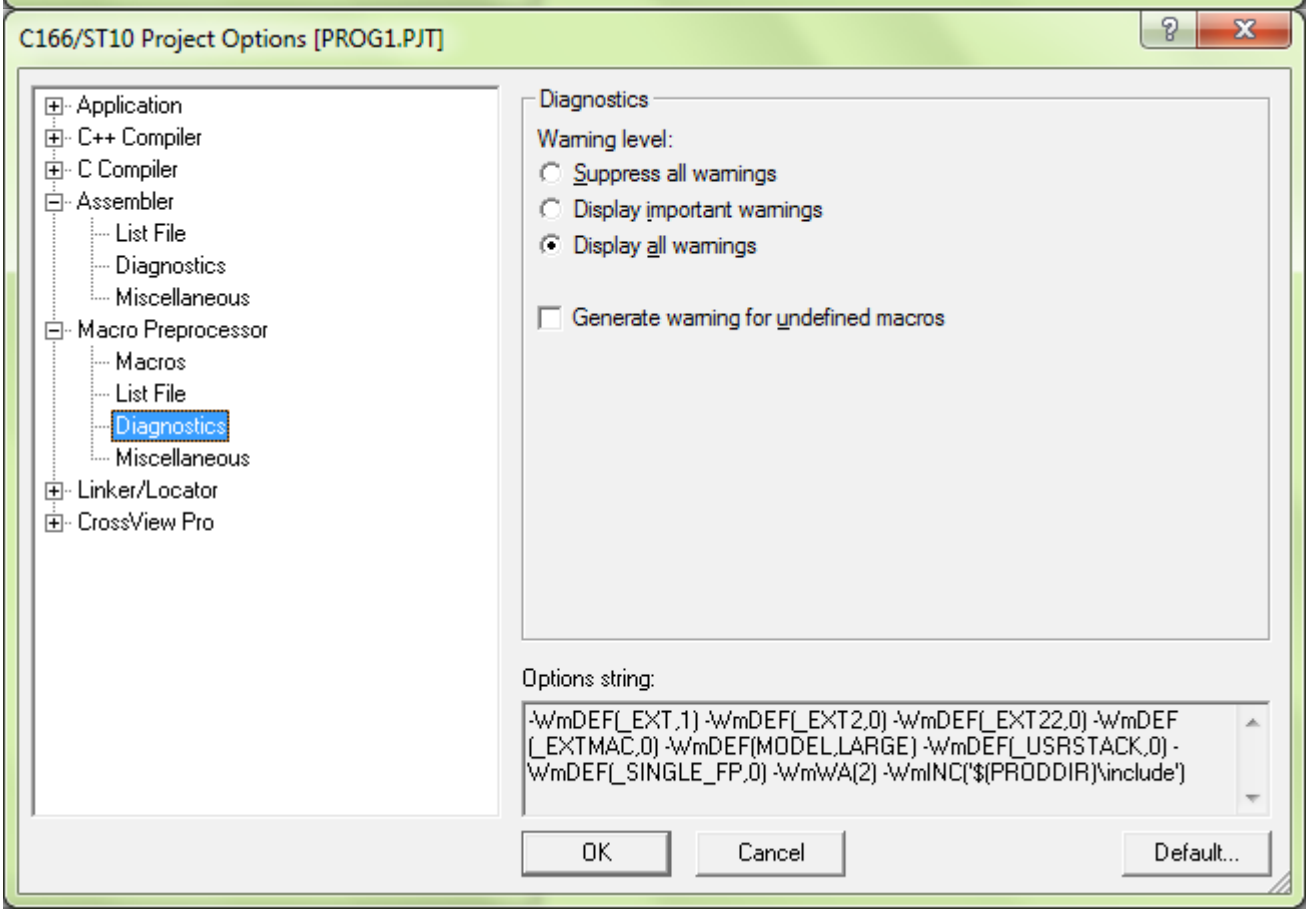

C166/ST10 Project Options [PROG1.PJT]
Application
Processor
Memory Model
Startup
SYSCON: System Configuration
ADDRSEL1: Address Selection 1
ADDRSEL2: Address Selection 2
ADDRSEL3: Address Selection 3
ADDRSEL4: Address Selection 4
BUSCON0: Bus Configuration 0
BUSCON1: Bus Configuration 1
BUSCON2: Bus Configuration 2
BUSCON3: Bus Configuration 3
BUSCON4: Bus Configuration 4
WDTCON: Watchdog Timer Control
C++ Compiler
C Compiler
Assembler
Macro Preprocessor
Linker/Locator
CrossView Pro
ADDRSEL1: Address Selection 1
Include in startup code
Value: 0x2008
Default: 0x0000
Bit# Description Value
0-3 RGSZ1: Range Size Selection 0x8 = 1 MByte
4-15 RGSAD1: Range Start Address 0x200
OK
Cancel
Default...

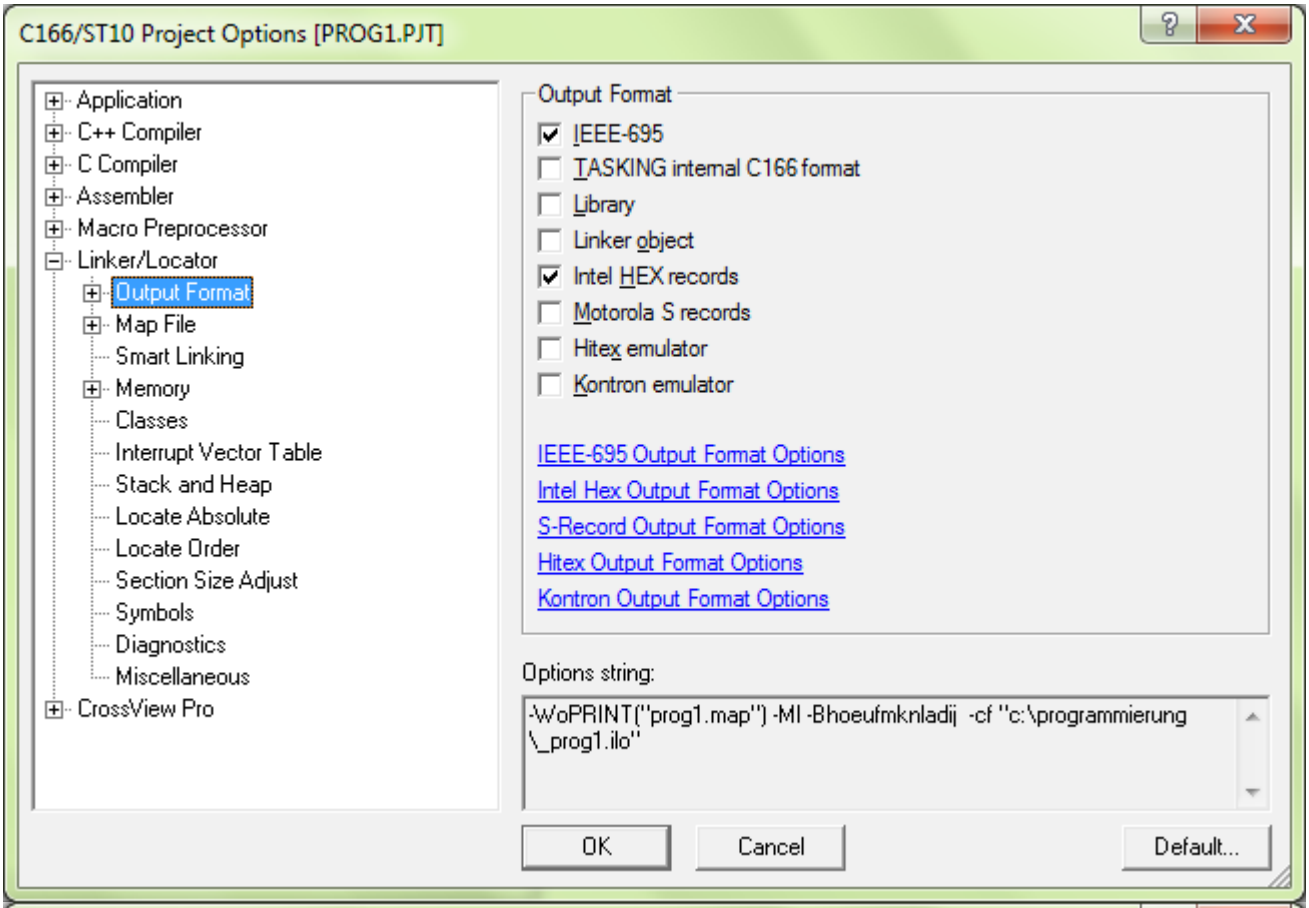

C166/ST10 Project Options [PROG1.PJT]
Application
Processor
Memory Model
Startup
SYSCON: System Configuration
ADDRSEL1: Address Selection 1
ADDRSEL2: Address Selection 2
ADDRSEL3: Address Selection 3
ADDRSEL4: Address Selection 4
BUSCON0: Bus Configuration 0
BUSCON1: Bus Configuration 1
BUSCON2: Bus Configuration 2
BUSCON3: Bus Configuration 3
BUSCON4: Bus Configuration 4
WDTCON: Watchdog Timer Control
C++ Compiler
C Compiler
Assembler
Macro Preprocessor
Linker/Locator
CrossView Pro
ADDRSEL2: Address Selection 2
Include in startup code
Value: 0x1008
Default: 0x0000
Bit# Description Value
0-3 RGSZ2: Range Size Selection 0x8 = 1 MByte
4-15 RGSAD2: Range Start Address 0x100
OK
Cancel
Default...

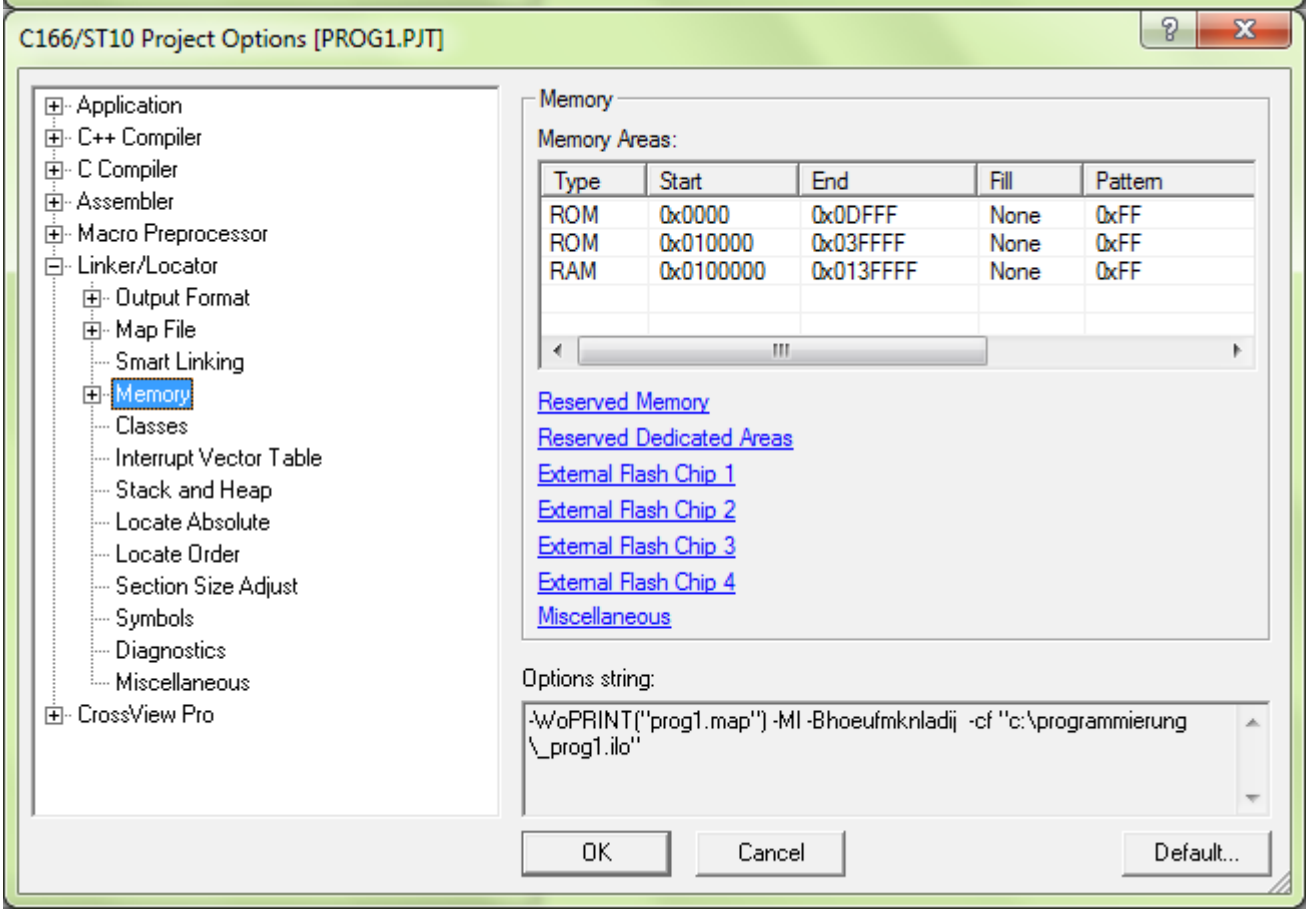

C166/ST10 Project Options [PROG1.PJT]
Application
Processor
Memory Model
Startup
SYSCON: System Configuration
ADDRSEL1: Address Selection 1
ADDRSEL2: Address Selection 2
ADDRSEL3: Address Selection 3
ADDRSEL4: Address Selection 4
BUSCON0: Bus Configuration 0
BUSCON1: Bus Configuration 1
BUSCON2: Bus Configuration 2
BUSCON3: Bus Configuration 3
BUSCON4: Bus Configuration 4
WDTCON: Watchdog Timer Control
C++ Compiler
C Compiler
Assembler
Macro Preprocessor
Linker/Locator
CrossView Pro
ADDRSEL3: Address Selection 3
Include in startup code
Value: 0x0ffe0
Default: 0x0000
Bit# Description Value
0-3 RGSZ3: Range Size Selection 0x0 = 4 kByte
4-15 RGSAD3: Range Start Address 0xFFE
OK
Cancel
Default...

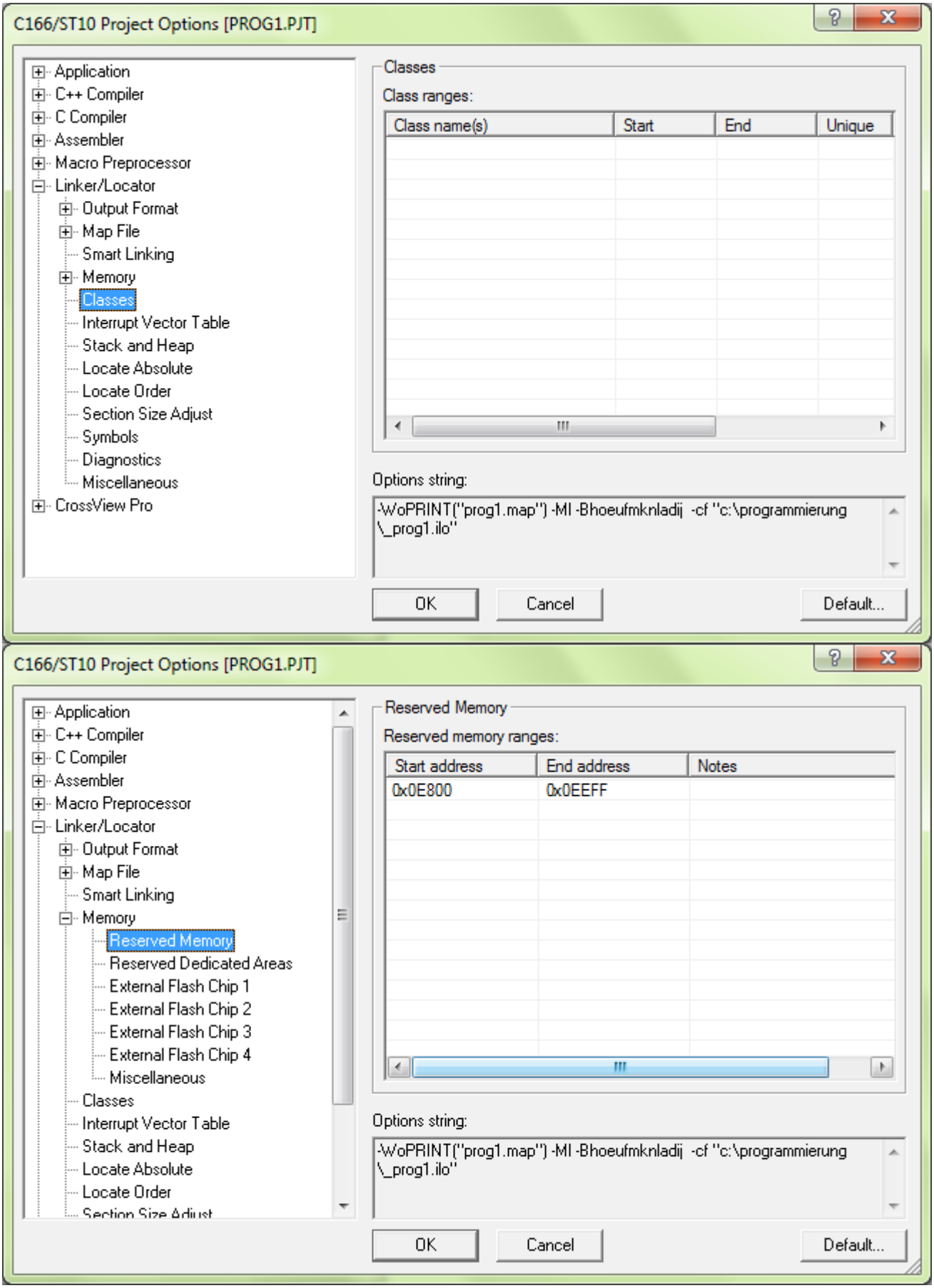

C166/ST10 Project Options [PROG1.PJT]
Application
Processor
Memory Model
Startup
SYSCON: System Configuration
ADDRSEL1: Address Selection 1
ADDRSEL2: Address Selection 2
ADDRSEL3: Address Selection 3
ADDRSEL4: Address Selection 4
BUSCON0: Bus Configuration 0
BUSCON1: Bus Configuration 1
BUSCON2: Bus Configuration 2
BUSCON3: Bus Configuration 3
BUSCON4: Bus Configuration 4
WDTCON: Watchdog Timer Control
C++ Compiler
C Compiler
Assembler
Macro Preprocessor
Linker/Locator
CrossView Pro
ADDRSEL4: Address Selection 4
Include in startup code
Value: 0x0fff0
Default: 0x0000
Bit# Description Value
0-3 RGSZ4: Range Size Selection 0x0 = 4 kByte
4-15 RGSAD4: Range Start Address 0xFFF
OK
Cancel
Default...

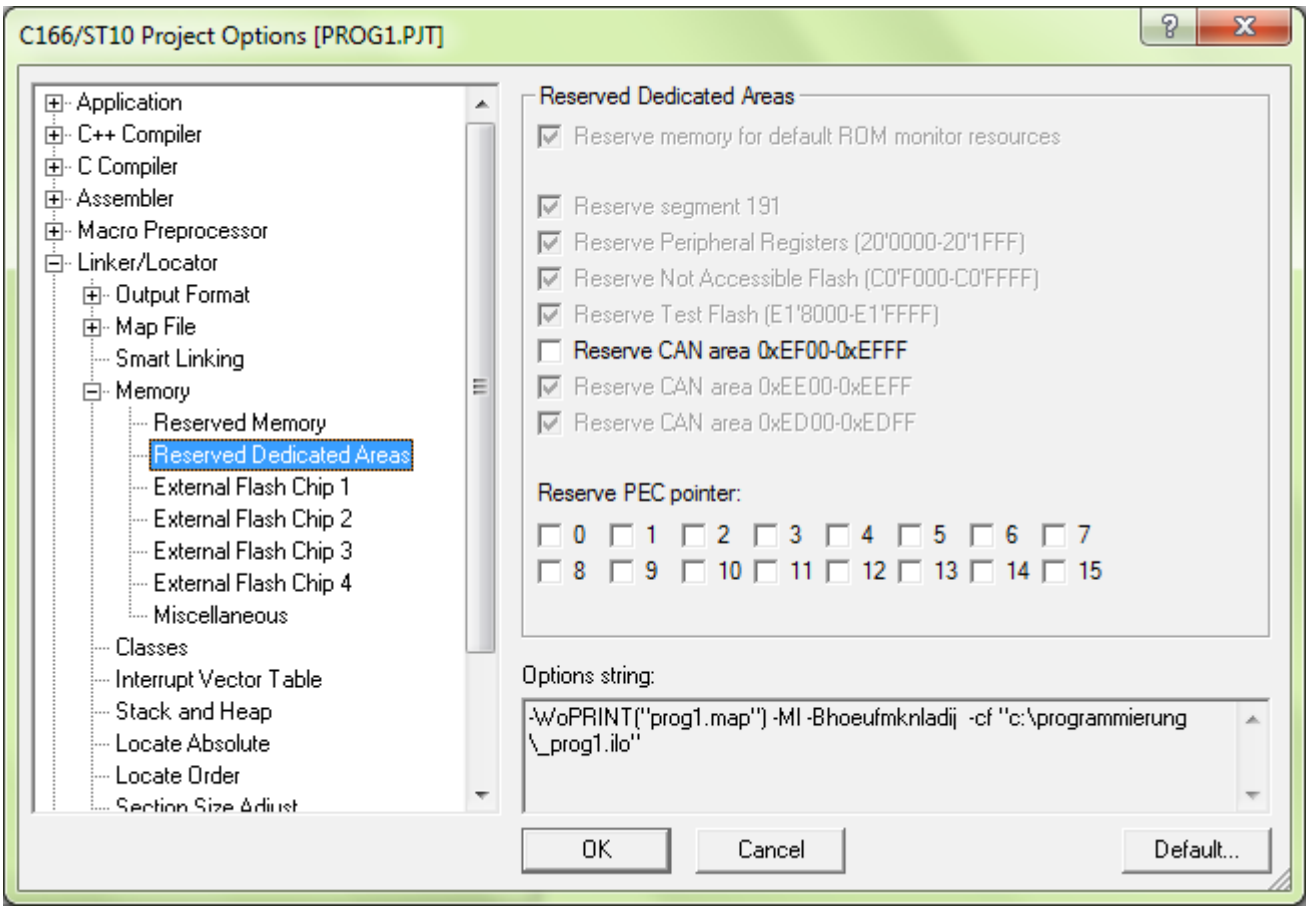

C166/ST10 Project Options [PROG1.PJT]
Application
C++ Compiler
C Compiler
Preprocessing
Allocation of Variables
Floating Point
Language
Optimization
Debug
Libraries
Diagnostics
Output
MISRA-C
Miscellaneous
Assembler
Macro Preprocessor
Linker/Locator
CrossView Pro
Optimization
Optimization:
Optimize for speed
No optimization
Default optimization
Optimize for speed
Optimize for size
Custom optimization
Custom setting
Relax all al
Relax cros
Common Subexpression Elimination (CSE)
Constant and copy propagation
Favor code size above execution speed
Code recognition to generate optimal code for expressions
Optimization of interrupt frame code for C interrupt functions
Peephole optimizer (remove redundant code)
Trace contents of registers for reuse without reloading
Optimizations that increase compile time:
Instruction reordering
Dataflow analysis peephole (DFAP)
Options string:
-MI -Wc"I$(PRODDIR)\include" -Wc-Bhoeufmknladij -Wc-0B -Wc-zautobitastruct-4 -Wc-zautobita-0 -Wc-T256 -FSC -noc++ -Wc-A1 -Wc-zvolatile_union -Wc-03 -Wc-g -Wc-newerr -Wc-s -tmp
OK
Cancel
Default...

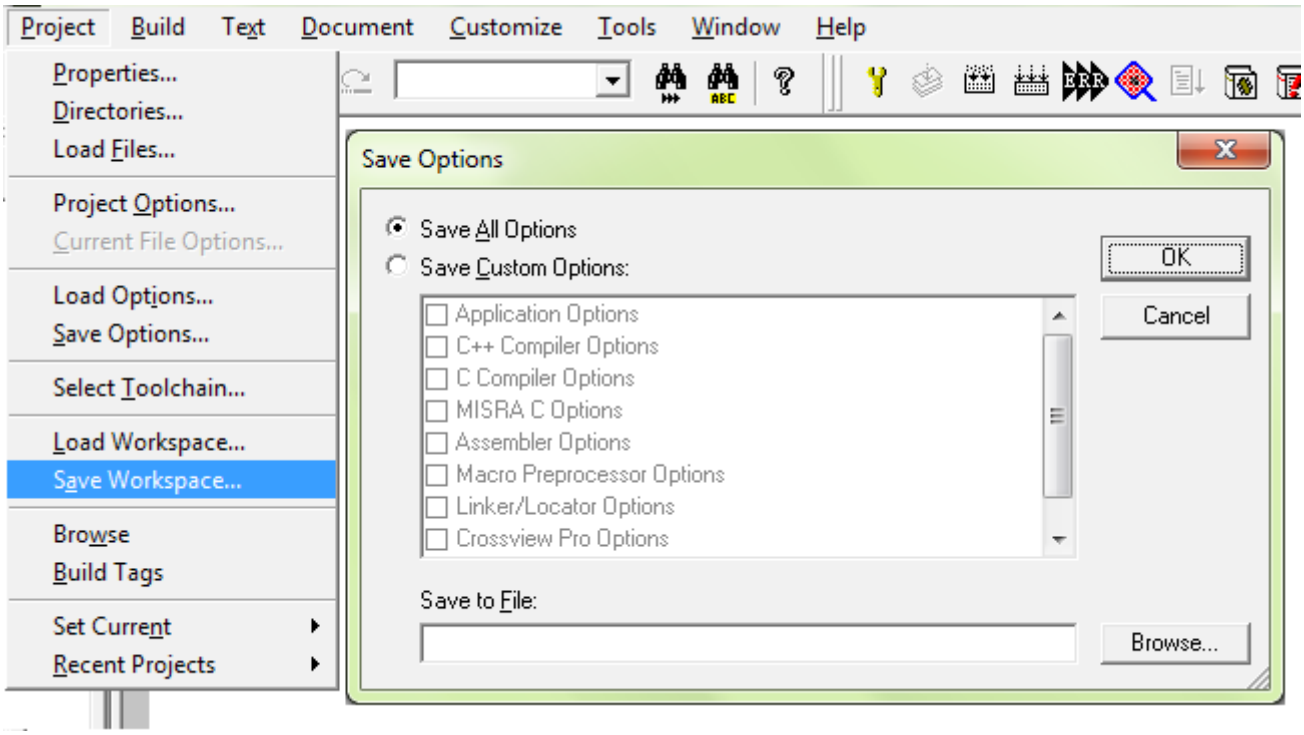

C166/ST10 Project Options [PROG1.PJT]
Application
C++ Compiler
C Compiler
Assembler
List File
Diagnostics
Miscellaneous
Macro Preprocessor
Macros
List File
Diagnostics
Miscellaneous
Linker/Locator
CrossView Pro
Diagnostics
Warning level:
Suppress all warnings
Display important warnings
Display all warnings
Generate warning for undefined macros
Options string:
-WmDEF(_EXT,1) -WmDEF(_EXT2,0) -WmDEF(_EXT22,0) -WmDEF(_EXTMAC,0) -WmDEF(MODEL,LARGE) -WmDEF(_USRSTACK,0) -WmDEF(_SINGLE_FP,0) -WmWA(2) -WmINC('$(PRODDIR)\include')
OK
Cancel
Default...

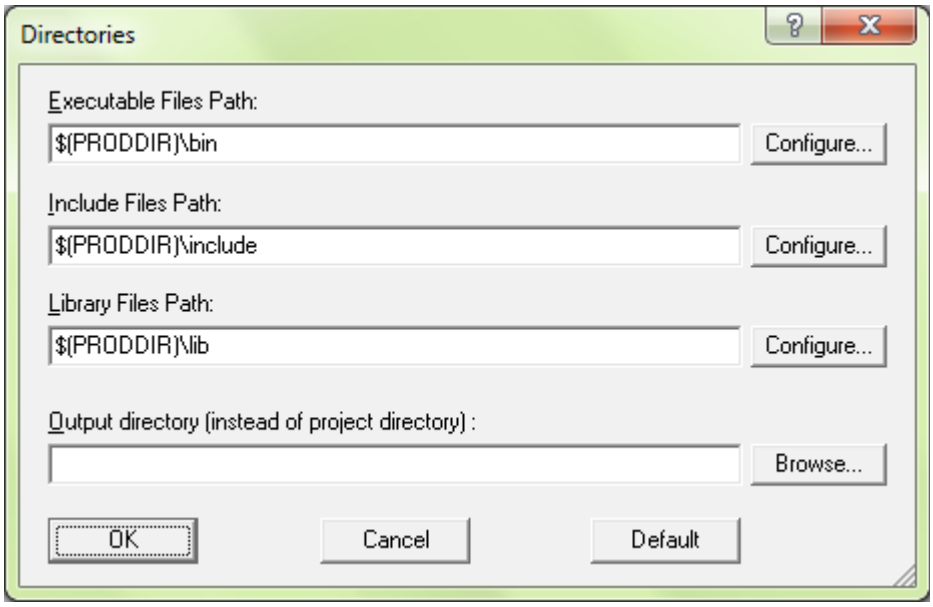

C166/ST10 Project Options [PROG1.PJT]
Application
C++ Compiler
C Compiler
Assembler
Macro Preprocessor
Linker/Locator
Output Format
Map File
Smart Linking
Memory
Classes
Interrupt Vector Table
Stack and Heap
Locate Absolute
Locate Order
Section Size Adjust
Symbols
Diagnostics
Miscellaneous
CrossView Pro
Output Format
IEEE-695
TASKING internal C166 format
Library
Linker object
Intel HEX records
Motorola S records
Hitex emulator
Kontron emulator
IEEE-695 Output Format Options
Intel Hex Output Format Options
S-Record Output Format Options
Hitex Output Format Options
Kontron Output Format Options
Options string:
-WoPRINT("prog1.map") -MI -Bhoeufmknladij -cf "c:\programmierung_prog1.ilo"
OK
Cancel
Default...


C166/ST10 Project Options [PROG1.PJT]
Application
C++ Compiler
C Compiler
Assembler
Macro Preprocessor
Linker/Locator
Output Format
Map File
Smart Linking
Memory
Classes
Interrupt Vector Table
Stack and Heap
Locate Absolute
Locate Order
Section Size Adjust
Symbols
Diagnostics
Miscellaneous
CrossView Pro
Memory
Memory Areas:
Type Start End Fill Pattern
ROM 0x0000 0x0DFFF None 0xFF
ROM 0x010000 0x03FFFF None 0xFF
RAM 0x0100000 0x013FFFF None 0xFF
Reserved Memory
Reserved Dedicated Areas
External Flash Chip 1
External Flash Chip 2
External Flash Chip 3
External Flash Chip 4
Miscellaneous
Options string:
-WoPRINT("prog1.map") -MI -Bhoeufmknladij -cf "c:\programmierung_prog1.ilo"
OK
Cancel
Default...

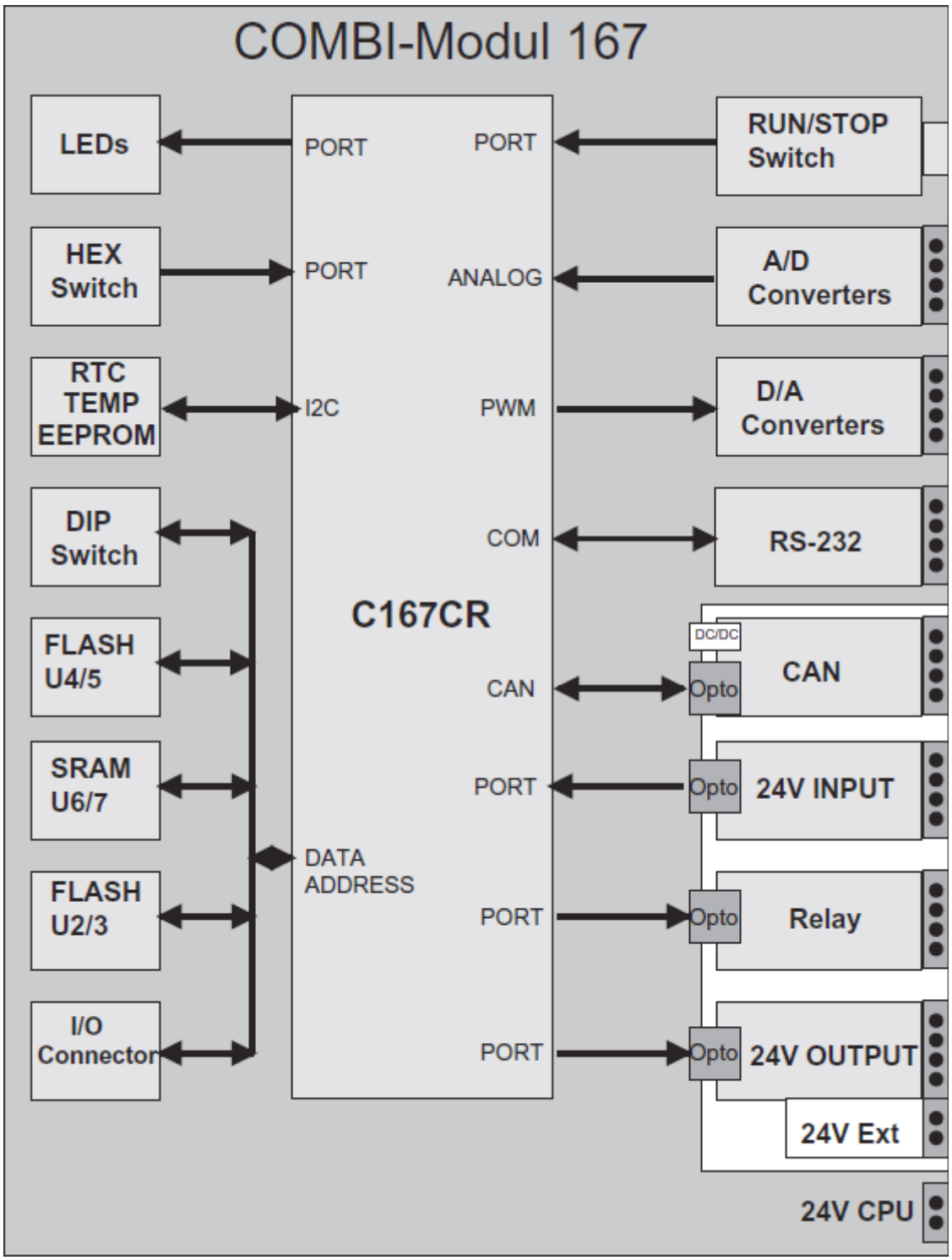

C166/ST10 Project Options [PROG1.PJT]
Application
C++ Compiler
C Compiler
Assembler
Macro Preprocessor
Linker/Locator
Output Format
Map File
Smart Linking
Memory
Classes
Interrupt Vector Table
Stack and Heap
Locate Absolute
Locate Order
Section Size Adjust
Symbols
Diagnostics
Miscellaneous
CrossView Pro
Classes
Class ranges:
Class name(s)
Start
End
Unique
Options string:
-WoPRINT("prog1.map") -MI -Bhoeufmknladij -cf "c:\programmierung_prog1.ilo"
OK
Cancel
Default...
C166/ST10 Project Options [PROG1.PJT]
Application
C++ Compiler
C Compiler
Assembler
Macro Preprocessor
Linker/Locator
Output Format
Map File
Smart Linking
Memory
Reserved Memory
Reserved Dedicated Areas
External Flash Chip 1
External Flash Chip 2
External Flash Chip 3
External Flash Chip 4
Miscellaneous
Classes
Interrupt Vector Table
Stack and Heap
Locate Absolute
Locate Order
Reserved Memory
Reserved memory ranges:
Start address
End address
Notes
0x0E800
0x0EEFF
Options string:
-WoPRINT("prog1.map") -MI -Bhoeufmknladij -cf "c:\programmierung_prog1.ilo"
OK
Cancel
Default...

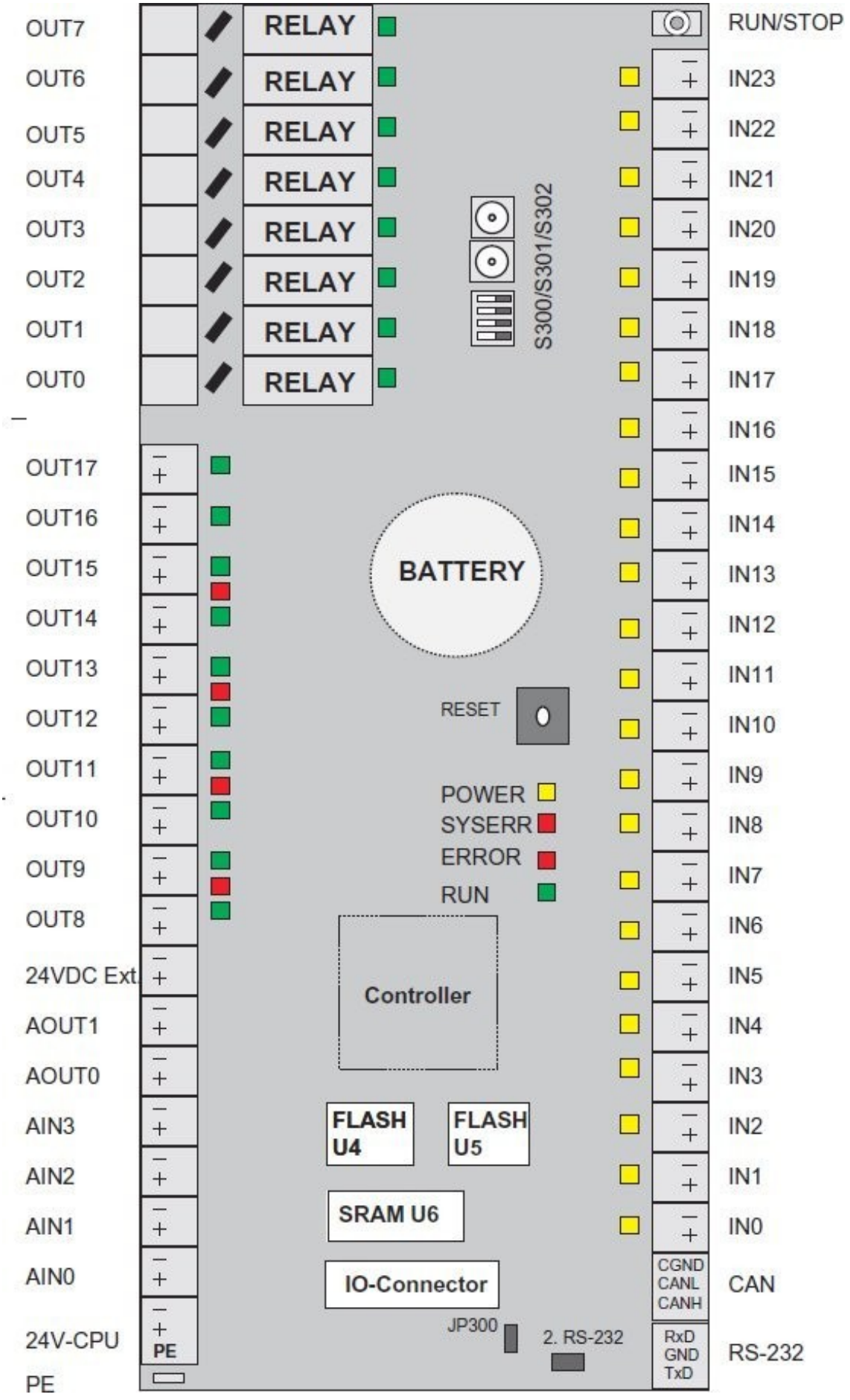

C166/ST10 Project Options [PROG1.PJT]
Application
C++ Compiler
C Compiler
Assembler
Macro Preprocessor
Linker/Locator
Output Format
Map File
Smart Linking
Memory
Reserved Memory
Reserved Dedicated Areas
External Flash Chip 1
External Flash Chip 2
External Flash Chip 3
External Flash Chip 4
Miscellaneous
Classes
Interrupt Vector Table
Stack and Heap
Locate Absolute
Locate Order
Reserved Dedicated Areas
Reserve memory for default ROM monitor resources
Reserve segment 191
Reserve Peripheral Registers (20'0000-20'1FFF)
Reserve Not Accessible Flash (C0'F000-C0'FFFF)
Reserve Test Flash (E1'8000-E1'FFFF)
Reserve CAN area 0xEF00-0xEFFF
Reserve CAN area 0xEE00-0xEEFF
Reserve CAN area 0xED00-0xEDFF
Reserve PEC pointer:
0 1 2 3 4 5 6 7
8 9 10 11 12 13 14 15
Options string:
-WoPRINT("prog1.map") -MI -Bhoeufmknladij -cf "c:\programmierung_prog1.ilo"
OK
Cancel
Default...


Project
Build
Text
Document
Customize
Tools
Window
Help
Properties...
Directories...
Load Files...
Project Options...
Current File Options...
Load Options...
Save Options...
Select Toolchain...
Load Workspace...
Save Workspace...
Browse
Build Tags
Set Current
Recent Projects
Save Options
Save All Options
Save Custom Options:
Application Options
C++ Compiler Options
C Compiler Options
MISRA C Options
Assembler Options
Macro Preprocessor Options
Linker/Locator Options
Crossview Pro Options
OK
Cancel
Save to File:
Browse...


Directories
Executable Files Path:
$(PRODDIR)\bin
Configure...
Include Files Path:
$(PRODDIR)\include
Configure...
Library Files Path:
$(PRODDIR)\lib
Configure...
Output directory (instead of project directory) :
Browse...
OK
Cancel
Default

ю Mikrorechner




OUT7
OUT6
OUT5
OUT4
OUT3
OUT2
OUT1
OUT0
OUT17
OUT16
OUT15
OUT14
OUT13
OUT12
OUT11
OUT10
OUT9
OUT8
24VDC Ext
AOUT1
AOUT0
AIN3
AIN2
AIN1
AIN0
24V-CPU
PE
PE
RELAY
RELAY
RELAY
RELAY
RELAY
RELAY
RELAY
RELAY
S300/S301/S302
BATTERY
RESET
POWER
SYSERR
ERROR
RUN
Controller
FLASH U4
FLASH U5
SRAM U6
IO-Connector
JP300
2. RS-232
RUN/STOP
IN23
IN22
IN21
IN20
IN19
IN18
IN17
IN16
IN15
IN14
IN13
IN12
IN11
IN10
IN9
IN8
IN7
IN6
IN5
IN4
IN3
IN2
IN1
IN0
CGND
CANL
CANH
CAN
RxD
GND
TxD
RS-232

- I/O Peripherie:

En esta parte tengo que explicar la comunicación desde el osciloscopio al micro, desde el pc al micro, del micho al pc, las salidas en pwm hacia los reles

- Programm:

  Hier zeigt man die Programme, die das Projekt machen

******SYSDEF.C******

In SYSDEF.C zeigt man die Initialisierung

```
#include <c166.h>
#include <reg167cr.h>

#include "glob_def.h"
#include "sysdef.h"
/*****************************************************************************
 ---------------------- function -------------------------------------------
 *****************************************************************************/
/*
* name:              RS232_Init
* description:
* input:             -
* globals:           -
* output:            -
* return:            -
*
*****************************************************************************/
void RS232_Init (void)  {

   _putbit(0,ODP3,10);   // P3.10 = TxD0 output driver in push/pull mode
   _putbit(0,ODP3,11);   // P3.11 = RxD0 not necessary
   _putbit(1,P3,10);   // P3.10 Initial value
   _putbit(1,P3,11);   // P3.10 Initial value
   _putbit(1,DP3,10);  // P3.10 is an output (TxD0)
   _putbit(0,DP3,11);  // P3.11 is an input  (RxD0)

   // seriell channel 0 Transmit Interrupt Control
   S0TIC = 0x00;
   //seriell channel 0 Error Interrupt Control
   S0EIC = 0x00;
   // seriell channel 0 Transmit Buffer Interrupt Control
   S0TBIC =  ( 0 << 7)               // S0TIR  TxD0 Interrupt request
         |  ( 1 << 6)          // S0TIE  TxD0 Interrupt enable
         |  ( SERTX0_INTR_LVL << 2)    // ILVL   TxD0 Interrupt Priority
         |  ( SERTX0_INTR_GLVL << 0);  // GLVL   TxD0 Interrupt GroupPri

```

```
36
37    //seriell channel 0 Receive Interrupt Control
38    S0RIC =  ( 0 << 7)              // S0RIR  TxD0 Interrupt request
39           |  ( 1 << 6)           // S0RIE  TxD0 Interrupt enable
40           |  ( SERRX0_INTR_LVL << 2)    // ILVL   TxD0 Interrupt Priority
41           |  ( SERRX0_INTR_GLVL << 0);  // GLVL   TxD0 Interrupt GroupPri
42
43
44
45    S0BG  = Baudrate19200;
46
47    S0CON = 0x0011;       // Recieve enable, 8-bit Data asynchron
48 }
49
50 /*****************************************************************************
51  ---------------------- function -------------------------------------------
52  *****************************************************************************/
53 /*
54 * name:              Timer3_Init
55 * description:
56 * input:             -
57 * globals:           -
58 * output:            -
59 * return:            -
60 *
61 *****************************************************************************/
62 void Timer3_Init(void)  {
63
64    T3CON = ( 0  <<10)   /* TxOTL output toggle latch                    */
65           | ( 0  <<9)  /* TxOE  output enable                          */
66           | ( 0  <<8)  /* TxUDE up/down external control enable bit    */
67           | ( 1  <<7)    /* TxUD  up/down control 0=up                   */
68           | ( 0  <<6)    /* TxR   Run enable                             */
69           | ( 0  <<3)    /* TxM   Mode control 000b = timer mode         */
70           | ( 1  <<0);   /* TxI   Input selection 001b (resolution 800 ns)  */
```

```
70          | ( 1  <<0);   /* TxI   Input selection 001b (resolution 800 ns)  */
71
72    T3 = 37500;          /* Start value = 30ms / 0,0008ms = 37500           */
73
74    T3IC =  ( 0 << 7)        /* TxIR Interrupt request                   */
75       | ( 0 << 6)           /* TxIE Interrupt enable                    */
76       | ( T3_INTR_LVL << 2)   /* ILVL Interrupt Level (Priority)         */
77       | ( T3_INTR_GLVL << 0); /* GLVL Interrupt GroupLevel               */
78
79 }
80
81 /*****************************************************************************
82  ---------------------- function -------------------------------------------
83  *****************************************************************************/
84 /*
85 * name:              Timer5_Init
86 * description:
87 * input:             -
88 * globals:           -
89 * output:            -
90 * return:            -
91 *
92 *****************************************************************************/
93 void Timer5_Init(void)
94 {
95   T5CON = (0 << 10)|(0 << 9)| (0 << 8)| (0 << 7)| (0 << 6)| (1 << 3)| (1 << 0);
96   T5=0;
97   T5IC=(0 << 7)|(0 <<6)| ( 3 << 2)| ( 0 << 0);
98 }
99
100 /*****************************************************************************
101  --------------------- function -------------------------------------------
102 *****************************************************************************/
103 /*
```

```
102 ************************************************************************/
103 /*
104 * name:           DiDoLines_Init
105 * description:
106 * input:          -
107 * globals:        -
108 * output:         -
109 * return:         -
110 *
111 ************************************************************************/
112 void DiDoLines_Init(void)  {
113
114    /* ######## INPUT LINES ################################################*/
115    // settings of IN0 .. IN7  == P2.0 .. P2.7
116    P2 = P2 & 0xff00;     // set latch = 0
117    DP2 = DP2 & 0xff00;   // set direction of input by clearing the flags
118
119
120    /* ######## OUPUT LINES#################################################*/
121    // settings of OUT0 .. OUT7   ==  P8.0 .. P8.7  Relais
122    // High activ
123    P8 = P8 & 0xff00;     // set latch = 0
124    DP8 = DP8 | 0x00ff;   // set direction of output by setting the flags
125    ODP8 = ODP8 & 0x00;   // output driver = 0 --> Gegentakt
126
127    // settings of OUT8 .. OUT11   ==  P3.0 .. P3.3  Transistor
128    // low activ
129    P3 = P3 | 0x000f;     // set latch = 1
130    DP3 = DP3 | 0x000f;   // set direction of output by setting the flags
131    ODP3 = ODP3 & 0xfff0; // output driver = 0 --> Gegentakt
132
133    // settings of OUT12 .. OUT15   ==  P7.4 .. P7.7  Transistor
134    // low activ
135    P7 = P7 | 0x00f0;     // set latch = 1
136    DP7 = DP7 | 0x00f0;   // set direction of output by setting the flags
137    ODP7 = ODP7 & 0xff0f; // output driver = 0 --> Gegentakt
```

++-+

```
136    DP7 = DP7 | 0x00f0;   // set direction of output by setting the flags
137    ODP7 = ODP7 & 0xff0f; // output driver = 0 --> Gegentakt
138
139    // settings of OUT100         ==  P4.4 = RUN-LED D5
140    _putbit(1,DP4,4);     // set direction of output
141    _putbit(0,P4,4);      // set latch = 0
142    _nop();
143
144 }
145
146 /***************************************************************************
147  ---------------------- function -----------------------------------------
148  ***************************************************************************/
149 /*
150 * name:              ADC_Init
151 * description:
152 * input:             -
153 * globals:           -
154 * output:            -
155 * return:            -
156 *
157 ***************************************************************************/
158 void ADC_Init(void)  {
159    // init control register
160    ADCON = (    0  <<15)      // ADCTC Conversion Time Control
161          | (    0  <<12)      // ADSTC Sample Time Control
162          | (    0  <<11)      // ADCRQ Injection Request flag
163          | (    0  <<10)      // ADCIN Injection enable
164          | (    0  <<9)       // ADWR  Wait for read control
165          | (    0  <<8)       // ADBSY Busy Flag
166          | (    0  <<7)       // ADST  ADC start bit = Stop of conversion
167          | (    3  <<4)       // ADM   Mode selection = Autoscan Contin. Conver
168          | (    1     ) ;     // ADCH  Channel input selection = Channel1 start
169
170    // init interrupt control
```

```
170    // init interrupt control
171    ADCIC = ( 0 << 7)        // TxIR  Interrupt request
172          | ( 1 << 6)        // TxIE  Interrupt enable
173          | ( ADC_INTR_LVL << 2) // ILVL Interrupt Level (Priority)
174          | ( ADC_INTR_GLVL);  // GLVL Interrupt GroupLevel
175
176    // init ERROR Interrupt control
177    ADEIC = 0;
178
179    // Port 5 digital Input disable
180    P5DIDIS = P5DIDIS | 0x000f;  // Digi.Input disable for P5.0 .. P5.3
181
182
183 }
184
185 /*****************************************************************************
186  -------------------- function -----------------------------------------
187 *****************************************************************************/
188 /*
189 * name:           PWM_OUT_Init
190 * description:
191 * input:         -
192 * globals:       -
193 * output:        -
194 * return:        -
195 *
196 *****************************************************************************/
197 void PWM_OUT_Init(void)  {
198
199     // settings of OUT16 und OUT17   ==  P7.0 und P7.1 Transistor
200   //P7 = P7 | 0x0003;     // set latch = 1  --> indirekte Proport.
201   P7 = P7 & 0xfffc;       // set latch = 0  --> direkte Proportionalit\E4t
202     DP7 = DP7 | 0x0003;     // set direction of output by setting the flags
203     ODP7 = ODP7 & 0xfffc;   // output driver = 0 --> Gegentakt
204
```

```
204
205   PT0 = 0;         // Z\E4hlregister vom PWM0
206   PT1 = 0;         // Z\E4hlregister vom PWM1
207
208   PP0 = 1023;        // PWM0 Periode
209   PP1 = 1023;        // PWM1 Periode
210
211   PWMCON0 = PWMCON0 & 0xcccc; // Control-Register0, nur PWM0 und PWM1
212
213   PWMCON1 = PWMCON1 | 0x0003; // Control-Register0, nur PWM0 und PWM1
214
215     PWMIC = 0;         // Interrupt Control
216
217   // Start Pulsweite
218     PW0 = 0;   // 0%
219     PW1 = 0;   // 0%
220
221 }
222
223
224 /*****************************************************************************
225  -------------------- function ---------------------------------------
226 *****************************************************************************/
227 /*
228 * name:           Analog_OUT_Init
229 * description:
230 * input:         -
231 * globals:       -
232 * output:        -
233 * return:        -
234 *
235 *****************************************************************************/
```

```
236 void Analog_OUT_Init(void)  {
237
238     // settings of AOUT0 und AOUT1   ==  P7.2 und P7.3 PWM mit PT1 Gl\E4ttung
239   P7 = P7 | 0x000c;      // set latch = 1  --> XOR  --> indirekte Proport.
240   //P7 = P7 & 0xfff3;       // set latch = 0  --> direkte Proportionalit\E4t
241     DP7 = DP7 | 0x000c;      // set direction of output by setting the flags
242     ODP7 = ODP7 & 0xfff3;    // output driver = 0 --> Gegentakt
243
244   PT2 = 0;         // Z\E4hlregister vom PWM2
245   PT3 = 0;         // Z\E4hlregister vom PWM3
246
247   PP2 = 255;         // PWM2 Periode
248   PP3 = 255;         // PWM3 Periode
249
250   PWMCON0 = PWMCON0 & 0x3333; // Control-Register0, nur PWM2 und PWM3
251
252   PWMCON1 = PWMCON1 | 0x000c; // Control-Register0, nur PWM2 und PWM3
253
254     PWMIC = 0;         // Interrupt Control
255
256   // Start Pulsweite
257     PW2 = 0;  // 0%
258     PW3 = 0;  // 0%
259
260 }
261
262
263
264
265 /*****************************************************************************
266  -------------------- function -----------------------------------------
267 *****************************************************************************/
268 /*
269 * name:           SYS_HW_Init
270 * description:    This function call all system initialize functions
271 *             will be called by MAIN function
272 *
273 *
274 * input:          -
275 * globals:        -
276 * output:         -
277 * return:         -
278 *
279 *****************************************************************************/
280
281 void SYS_HW_Init(void) {
282
283   RS232_Init();
284   Timer3_Init();
285   Timer5_Init();
286   DiDoLines_Init();
287   ADC_Init();
288   PWM_OUT_Init();
289   Analog_OUT_Init();
290
291 }
```

******CONTROL.C******

In CONTROL.C finden sie sich die Funktionen, die das Projekt erzeugen

```
1 /*****************************************************************************
2  ---------------------------------- module ----------------------------------
3  *****************************************************************************/
4 /*
5 * file name:   CONTROL.C
6 * description: include functions of control inputs / outputs
7 *
8 * function :   unsigned char ReadDI()
9 * function :   WriteDO(unsigned char Channel,unsigned char State)
.0 *
.1 * author:    schmied
.2 * version:     06.04.2011  V0.01
.3 */
.4 /*****************************************************************************
.5  ---------------------------- revision history -----------------------------
.6  *****************************************************************************/
.7 /* date:     name:  version:  description:
.8 *
.9 *****************************************************************************/
:0
:1 #include <c166.h>
:2 #include <reg167cr.h>
:3 #include <string.h>
:4 #include <stdlib.h>
:5 #include <math.h>
:6
:7 #include "control.h"
:8 #include "TxD_Function.h"
:9 #include "RxD_Function.h"
;0 #include "ADC_Interrupt.h"
;1
;2
;3
;4 /*****************************************************************************
;5  --------------------- function ------------------------------------------
;6 *****************************************************************************/
```

```
36 ************************************************************************/
37 /*
38 * name:           ReadDI
39 * description:    read digital lines
40 *                 called by MAIN
41 *
42 *                 INPUT LINES ******************************************
43 *                 IN0 .. IN7  == P2.0 .. P2.7
44 *         und weitere, siehe Manual
45 *
46 * input:          Input channel 0 .. 7
47 * globals:        -
48 * output:         -
49 * return:         State of line
50 *
51 ************************************************************************/
52 unsigned char ReadDI(unsigned int Channel)  {
53    unsigned char State=0;
54
55    switch (Channel) {
56       case 0: if(_getbit(P2,0)) State=1;
57      break;
58       case 1: if(_getbit(P2,1)) State=1;
59      break;
60       case 2: if(_getbit(P2,2)) State=1;
61      break;
62       case 3: if(_getbit(P2,3)) State=1;
63      break;
64       case 4: if(_getbit(P2,4)) State=1;
65      break;
66       case 5: if(_getbit(P2,5)) State=1;
67      break;
68       case 6: if(_getbit(P2,6)) State=1;
69      break;
70       case 7: if(_getbit(P2,7)) State=1;
```

```
70       case 7: if(_getbit(P2,7)) State=1;
71      break;
72
73       /* following channels ...
74       case Channel: if(_getbit(Px,y)) State=1;
75      break; */
76    }
77
78    return (State);
79 }
80
81
82 /************************************************************************
83  --------------------- function ------------------------------------------
84 ************************************************************************/
85 /*
86 * name:           WriteDO
87 * description:    write digital lines
88 *                 called by MAIN
89 *
90 *                 OUTPUT LINES
91 *                 High activ
92 *         OUT0 .. OUT7   ==  P8.0 .. P8.7  Relais
93 *
94 *                 Low activ
95 *         OUT8 .. OUT11  ==  P3.0 .. P3.3  Transistor
96 *         OUT12.. OUT15  ==  P7.4 .. P7.7  Transistor
97 *           OUT100         ==  P4.4 = RUN-LED D5
98 *
99 *           Die Unterscheidung ob high oder low activ
100 *               wird in der Funktion verwirklicht!
101 *           Alle Ausgänge werden EIN -geschaltet mit WriteDO(KanalNr,1);
102 *               Alle Ausgänge werden AUS -geschaltet mit WriteDO(KanalNr,0);
103 *
```

```
103 *
104 * input:          channel number, state
105 * globals:        -
106 * output:         -
107 * return:         -
108 *
109 ***************************************************************************/
110 void WriteDO(unsigned char Channel,unsigned char State)  {
111
112    switch (Channel) {
113     // OUT0 .. OUT7   ==  P8.0 .. P8.7  Relais
114     // high activ
115       case 0: if(State) _putbit(1,P8,0); else _putbit(0,P8,0);
116      break;
117       case 1: if(State) _putbit(1,P8,1); else _putbit(0,P8,1);
118      break;
119       case 2: if(State) _putbit(1,P8,2); else _putbit(0,P8,2);
120      break;
121       case 3: if(State) _putbit(1,P8,3); else _putbit(0,P8,3);
122      break;
123       case 4: if(State) _putbit(1,P8,4); else _putbit(0,P8,4);
124      break;
125       case 5: if(State) _putbit(1,P8,5); else _putbit(0,P8,5);
126      break;
127       case 6: if(State) _putbit(1,P8,6); else _putbit(0,P8,6);
128      break;
129       case 7: if(State) _putbit(1,P8,7); else _putbit(0,P8,7);
130      break;
131
132     // OUT8 .. OUT11  ==  P3.0 .. P3.3  Transistor
133     // low activ
134     case 8: if(State) _putbit(0,P3,0); else _putbit(1,P3,0);
135      break;
136     case 9: if(State) _putbit(0,P3,1); else _putbit(1,P3,1);
```

```
136     case 9: if(State) _putbit(0,P3,1); else _putbit(1,P3,1);
137      break;
138     case 10: if(State) _putbit(0,P3,2); else _putbit(1,P3,2);
139      break;
140     case 11: if(State) _putbit(0,P3,3); else _putbit(1,P3,3);
141      break;
142
143     // OUT12.. OUT15  ==  P7.4 .. P7.7  Transistor
144       // low activ
145     case 12: if(State) _putbit(0,P7,4); else _putbit(1,P7,4);
146      break;
147       case 13: if(State) _putbit(0,P7,5); else _putbit(1,P7,5);
148      break;
149       case 14: if(State) _putbit(0,P7,6); else _putbit(1,P7,6);
150      break;
151       case 15: if(State) _putbit(0,P7,7); else _putbit(1,P7,7);
152      break;
153
154     // OUT100         ==  P4.4 = RUN-LED D5
155       case 100: if(State) _putbit(1,P4,4); else _putbit(0,P4,4);
156      break;
157
158    }
159
160 }
161
162
163 /*****************************************************************************
164  -------------------- function -----------------------------------------
165 *****************************************************************************/
166 /*
167 * name:           printZAHL
168 * description:    jede Zehner-Potenz der Zahl in ein Sendebyte schreiben
169 *                 ASCII "0" = 48dec oder 0x30
170 *
```

```
169 *                   ASCII "0" = 48dec oder 0x30
170 *
171 * input:           Zahl die in den Sendepuffer geschrieben werden soll
172 * globals:         Zahl[5]
173 * output:          -
174 * return:          -
175 *
176 ****************************************************************************/
177 unsigned char Zahl[5];
178 void printZAHL (unsigned int Value) {
179
180    unsigned int dummy,modulo;
181    int i0;
182
183    dummy = Value;
184    for (i0=4;i0>=0;i0--) {
185     modulo = (dummy % 10)+48;
186       Zahl[i0] = (unsigned char)modulo;
187       dummy = dummy / 10;
188    }
189 }
190
191
192 /*****************************************************************************
193  --------------------- function ------------------------------------------
194 *****************************************************************************/
195 /*
196 * name:            fillTxd_Buffer_Statisch
197 * description:     wird in der MAIN zur Initialisierung aufgerufen
198 *          füllt den Sendepuffer mit statischem Text
199 *          die Struktur ist in der Dokumentation beschrieben
200 *
201 * input:           -
202 * globals:         -
203 * output:          -
204 * return:          -
```

```
203 * output:          -
204 * return:          -
205 *
206 ***************************************************************************/
207 void fillTxd_Buffer_Statisch(void)  {
208   // Text max. 12 Byte !!!
209   unsigned char  Text0[] = "run.. >  ";
210   unsigned char  Text1[] = "mVolt = ";
211   unsigned char  Text2[] = "Vrms = ";
212   unsigned char  Text3[] = "Messfehlr= ";
213   unsigned char  Text4[] = "Frequenz= ";
214
215
216
217   // Variable 0 - run..>  mit dem TimeStamp
218   memcpy(TxData+1,Text0,sizeof(Text0));
219   *(TxData+0) =0x0c;  //HOME
220   *(TxData+18)=0x0a;  //LF
221   *(TxData+19)=0x0d;  //CR
222
223   // Variable 1 - Rechenzeit innerhalb der 30ms MainLoop
224   memcpy(TxData+20,Text1,sizeof(Text1));
225   *(TxData+38)=0x0a;  //LF
226   *(TxData+39)=0x0d;  //CR
227
228   // Variable 2 - Maximale Rechenzeit innerhalb 30ms
229   memcpy(TxData+40,Text2,sizeof(Text2));
230   *(TxData+58)=0x0a;  //LF
231   *(TxData+59)=0x0d;  //CR
232
233     // Variable 3 - Zähler vom ReceiveBuffer
234   memcpy(TxData+60,Text3,sizeof(Text3));
235   *(TxData+78)=0x0a;  //LF
236   *(TxData+79)=0x0d;  //CR
237
```

```
238     // Variable 4 - TerminalEingabe Variable a -> ECHO
239   memcpy(TxData+80,Text4,sizeof(Text4));
240   *(TxData+98)=0x0a;  //LF
241   *(TxData+99)=0x0d;  //CR
242 
243 }
244 
245 /*****************************************************************************
246  --------------------- function ------------------------------------------
247 *****************************************************************************/
248 /*
249 * name:           fillTxd_Buffer_DynamischWerte
250 * description:    wird in der MAIN alle 2 Sekunden aufgerufen
251           Abbruch der Funktion, wenn ein TRACE aktiv ist
252 *         füllt den Sendepuffer mit aktuellen Werten
253 *         die Struktur ist in der Dokumentation beschrieben
254 * Hinweis       NEGATIVE ZAHLEN ausgeben:
255 *           printZAHL((unsigned int)(abs(dummy2)));
256 *         if (dummy2<0) *(TxData+52)=0x2d;  // "-"
257 *
258 * input:          -
259 * globals:        -
260 * output:         -
261 * return:         -
262 *
263 *****************************************************************************/
264 void fillTxd_Buffer_DynamischWerte(void)  {
265 
266   static unsigned int timeStamp=0;
267 
268   // ------------------------------------------------------
269   // Trace ist aktiv  --- >>>  Funktion beenden  !!!!
270   if (ReadZahl_a == 1) return;
271   // ------------------------------------------------------
```

```
270   if (ReadZahl_a == 1) return;
271   // -------------------------------------------------------
272
273   // -------------------------------------------------------
274   // Variable 0 - run..>  mit dem TimeStamp
275     timeStamp++;
276   printZAHL(mVolt);
277   memcpy(TxData+13,Zahl,sizeof(Zahl));
278
279   // Variable 1 - Rechenzeit innerhalb der 30ms MainLoop
280   printZAHL(Vrms);
281   memcpy(TxData+33,Zahl,sizeof(Zahl));
282
283   // Variable 2 - Maximale Rechenzeit innerhalb 30ms
284   printZAHL(Messfehlr);
285   memcpy(TxData+53,Zahl,sizeof(Zahl));
286
287     // Variable 3 - Zähler vom ReceiveBuffer
288   printZAHL(Frequenz);
289   memcpy(TxData+73,Zahl,sizeof(Zahl));
290
291     // Variable 4 - TerminalEingabe Variable a -> ECHO
292
293   CntTBuf=0;          // Zähler für SendeBytes
294   S0TBUF = TxData[0]; // senden anstoßen
295
296
297 }
298
299 /*****************************************************************************
300  -------------------- function ---------------------------------------
301 *****************************************************************************/
302 /*
303 * name:           ReadZahl
304 * description:
```

```
304 * description:
305 *
306 * input:          -
307 * globals:        -
308 * output:         -
309 * return:         -
310 *
311 ***************************************************************************/
312 unsigned int ReadZahl_a = 0;
313 unsigned int ReadZahl_b = 0;
314 unsigned int ReadZahl_c = 0;
315 unsigned int ReadZahl_d = 0;
316
317 unsigned int ReadZahl(void) {
318   unsigned int zahl=0,i;
319   unsigned int help;
320   unsigned int potenz=10000;
321
322   for(i=1;i<6;i++) {
323     help=(unsigned int)(RxData[i] - 0x30);
324     zahl=zahl+(help*potenz);
325     potenz=potenz/10;
326   }
327
328   return(zahl);
329 }
330
331 /***************************************************************************
332  ---------------------- function -----------------------------------------
333  ***************************************************************************/
334 /*
335 * name:       ADC_StopStart
336 * description:    wird alle 30ms in der Main abgearbeitet
337           stoppt den ADC
338           sichert Werte
```

```
341 * input:              -
342 * globals:            -
343 * output:             -
344 * return:             -
345 *
346 ********************************************************************************/
347 unsigned int actNumberOfADC_Conv;  // Sicherung der Anzahl von ADC Wandlungen
348
349
350 void ADC_StopStart(void) {
351   // Stopp ADC
352   ADST = 0;
353
354   // Sicherung der Anzahl von ADC Wandlungen
355   actNumberOfADC_Conv = CntAdcBuf;
356
357   // Bereich zum Schreiben wechseln
358   if (actWriteIndex==0) actWriteIndex=4000;
359   else  actWriteIndex=0;
360
361   // Bereich zum Lesen wechseln
362   if (actReadIndex==0) actReadIndex=4000;
363   else  actReadIndex=0;
364
365   // setzt den Zähler zur Bestimmung der Anzahl der Wandlungen zurück
366   CntAdcBuf = 0;
367
368   // Start ADC
369   ADST = 1;
370   Filter();
371   PIkontroler();
372 }
```

```
373
374 /*****************************************************************************
375  ---------------------- function ------------------------------------------
376  *****************************************************************************/
377 /*
378 * name:       ADC_SaveADCData
379 * description:    wird alle 30ms in der Main abgearbeitet
380           sichert Daten vom ADC - Kanal 0
381           sichert Daten vom ADC - Kanal 1
382
383 * input:              -
384 * globals:            -
385 * output:             -
386 * return:             -
387 *
388 *****************************************************************************/
389 unsigned int AdcCh0Buf[2000];  // Sicherung ADC Kanal 0 Daten
390 unsigned int AdcCh1Buf[2000];  // Sicherung ADC Kanal 1 Daten
391
392 void ADC_SaveADCData(void) {
393   unsigned int i,tempADDAT;
394   unsigned int kanal;
395   unsigned int index0=0,index1=0;
396
397   for (i=0;i<actNumberOfADC_Conv;i++) {
398     tempADDAT = *(ADC_BUFFER+i+actReadIndex);
399     kanal = (tempADDAT & 0xf000) >> 12;
400     if (kanal==0) {
401       AdcCh0Buf[index0] = tempADDAT & 0x03ff;
402       index0++;
403     }
404     else {
405       AdcCh1Buf[index1] = tempADDAT & 0x03ff;
```

```
404     else {
405       AdcCh1Buf[index1] = tempADDAT & 0x03ff;
406       index1++;
407     }  // if .. else
408
409   }  // for
410
411 }  // function
412
413 /*****************************************************************************
414  ---------------------- function -------------------------------------------
415  *****************************************************************************/
416 /*
417 * name:       ShowCalcTime
418 * description:    wird alle 30ms in der Main abgearbeitet
419           liest den Timer T3 (mainLoop)
420           berechnet die Rechenzeit in mikrosekunden
421           Zeit = (37500 - T3) * 0,8µs
422           delta_T3 = 800ns = 0,8µs
423           37500 ==> 37500 * 800ns = 30ms
424 * input:             -
425 * globals:           -
426 * output:            -
427 * return:            -
428 *
429 *****************************************************************************/
430 unsigned int CalcTime;
431 unsigned int MaxCalcTime=0;
432
433 void ShowCalcTime(void) {
434   unsigned long int help;
435   static unsigned int Cnt2Rst=0;
436
437     help = 37500 - T3;
438   help = help * 8;
439   help = help / 10;
```

```
438   help = help * 8;
439   help = help / 10;
440
441   CalcTime =(unsigned int)(help);
442
443   if (CalcTime > MaxCalcTime) MaxCalcTime=CalcTime;
444   Cnt2Rst++;
445   if (Cnt2Rst > 1000) {
446     Cnt2Rst=0;
447     MaxCalcTime=0;
448   }
449 }
450 /*****************************************************************************
451
452  ---------------------- function ------------------------------------------
453  *****************************************************************************/
454 /*
455
456 * name:       Filter
457 * description:
458 * input:              -
459 * globals:            -
460
461 * output:             -
462 * return:             -
463
464 *****************************************************************************/
465
466 int Filtfinal[2][4000],puls[2][4000],freq[2];
467  int ,j;
468  int Maxch[2]={0,0},Minch[2]={1024,1024},Maxfil[2]={0,0},Minfil[2]={1024,1024};
469 void Filter()
470 {
471   /******************
472     Datei
473   ******************/
```

```
471  /******************
472    Datei
473  ******************/
474  int u,i,ADC,j,error,temporal[2],Stapel[2][30],datenzaeler_i;
475  volatile float temp[2]={0.0001,0.0001} temp1[2]={0.002,0.002},Filt[2][2][2];
476  int kp[2]={1,1},ko[2];
477
478  /*******************************
479  * - Filter
480  *******************************/
481  for(u=0;u<2;u++)
482  {
483    if(u==0)ADC=AdcCh0Buf[0];
484    if(u==1)ADC=AdcCh1Buf[0];
485    ko[u]=1/(temp[u]+temp1[u]);
486    Filt[u][0][0]=kp[u]*ko[u]*1/temp[u]*ADC;
487    if(Filt[u][0][0]<0)Filt[u][0][1]=0;
488    if(Filt[u][0][0]>1024)Filt[u][0][1]=1024;
489    Filt[u][1][0]=kp[u]*ko[u]*1*temp[u]*Filt[u][0][0];
490    if(Filt[u][1][0]<0)Filt[u][1][1]=0;
491    if(Filt[u][1][0]>1024)Filt[u][1][1]=1024;
492    Filtfinal[u][0]=(int)(Filt[u][1][1]);
493    for(i=1; i<4000; i++)
494    {
495      if(u==0)ADC=AdcCh0Buf[i];
496      if(u==1)ADC=AdcCh1Buf[i];
497      Filt[u][0][1]=kp[u]*ko[u]*(float)(ADC)+ko[u]*Filt[u][0][0]*temp1[u];
498      if(Filt[u][0][1]<0)Filt[u][0][1]=0;
499      if(Filt[u][0][1]>1024)Filt[u][0][1]=1024;
500      Filt[u][1][1]=kp[u]*ko[u]*Filt[u][0][1]*temp[u]+ko[u]*Filt[u][1][0]*temp1[u];
501      if(Filt[u][1][1]<0)Filt[u][1][1]=0;
502      if(Filt[u][1][1]>1024)Filt[u][1][1]=1024;
503      Filtfinal[u][i]=(int)(Filt[u][1][1]);
504      Filt[u][0][0]=Filt[u][0][1];
505      Filt[u][1][0]=Filt[u][1][1];
506    }
```

```
506     }
507     //creador de pulso
508     j=0;
509     for(i=1; i<4000; i++)
510     {
511       error=Filtfinal[u][i-1]-Filtfinal[u][i];
512       if(error>0)
513       {
514         puls[u][i]=1;
515         if(temporal[u]==0)
516         {
517           //cuenta la frecuencia
518           Stapel[u][j]=i-datenzaeler_i;
519           datenzaeler_i=i;
520           j++;
521         }
522         temporal[u]=1;
523       }
524       if(error==0)
525       {
526         puls[u][i]=temporal[u];
527       }
528       if(error<0)
529       {
530         puls[u][i]=0;
531         if(temporal[u]==1)
532         {
533           Stapel[u][j]=i-datenzaeler_i;
534           datenzaeler_i=i;
535           j++;
536         }
537         temporal[u]=0;
538       }
539     }
540     freq[u]=0;
```

```
539     }
540     freq[u]=0;
541     for(i=1;i<j;i++)
542     {
543       freq[u]=freq+Stapel[u][i];
544     }
545     freq[u]=freq[u]/(j-1);
546   }
547   //máximo y mínimo de cada uno
548   for(u=0; u<2; u++)
549   {
550     Maxch[u]=0
551     Minch[u]=1024
552     Maxfil[u]=0
553     Minfil[u]=1024
554     for(i=300;i<2000;i++)
555     {
556       if(Maxch[u]<AdcCh1Buf[i])
557       {
558         Maxch[u]=AdcCh1Buf[i];
559       }
560       if(Maxfil[u]<Filtfinal[1][i])
561       {
562         Maxfil[u]=Filtfinal[1][i];
563       }
564       if(Minch[u]>AdcCh1Buf[i])
565       {
566         Minch[u]=AdcCh1Buf[i];
567       }
568       if(Minfil[u]>Filtfinal[1][i])
569       {
570         Minfil[u]=Filtfinal[1][i];
571       }
572     }
```

```
573   }
574 }
575 /*************************************************************************
576  ---------------------- function -----------------------------------------
577  *************************************************************************/
578 /*
579 * name:       PI
580 * description:
581 * input:            -
582 * globals:          -
583 * output:           -
584 * return:           -
585 *************************************************************************/
586 unsigned int Gewpi, Tsampling,Tintegral;
587 int fehler[2]={0,0},propinte[2]={0,0};
588 void PIkontroler()
589 {
590   fehler[1]=freq[1]-freq[2];
591   propinte[1]=Gewpi*fehler[1](1+Tsampling/(2*Tintegral));
592   propinte[1]=propinte[1]-Gewpi*fehler[0](1-Tsampling/(2*Tintegral))+propinte[0];
593   fehler[0]=fehler[1];
594   propinte[0]=propinte[1];
595 }
596 /*************************************************************************
597  ---------------------- function -----------------------------------------
598  *************************************************************************/
599 /*
600 * name:       PWM
601 * description:
602 * input:            -
603 * globals:          -
604 * output:           -
605 * return:           -
606
607 *************************************************************************/
608 float freqausgang,gradPWM
```

```
606
607 ********************************************************************************/
608 float freqausgang,gradPWM
609 void PWMpulse()
610 {
611   int konstant=0;
612   freqausgangn=propinte[0]*0.2*2;//tiempo de salida
613   gradPWM=freqausgangn/(0,0008);//disparo el PWM cada 10 grados
614   T5=gradPWM;
615   konstant=T5/36;
616   while(!(T5IR))
617   {
618     if(T5<=konstant*36)
619     {
620       if(T5>konstant*35)
621       {
622         PW2=127;
623       }
624     }
625     if(T5<=konstant*35)
626     {
627       if(T5>konstant*34)
628       {
629         PW2=149;
630       }
631     }
632     if(T5<=konstant*34)
633     {
634       if(T5>konstant*33)
635       {
636         PW2=170;
637       }
638     }
639     if(T5<=konstant*33)
640     {
```

```
639     if(T5<=konstant*33)
640     {
641       if(T5>konstant*35)
642       {
643         PW2=190;
644       }
645     }
646     if(T5<=konstant*36)
647     {
648       if(T5>konstant*32)
649       {
650         PW2=209;
651       }
652     }
653     if(T5<=konstant*32)
654     {
655       if(T5>konstant*31)
656       {
657         PW2=224;
658       }
659     }
660     if(T5<=konstant*31)
661     {
662       if(T5>konstant*30)
663       {
664         PW2=237;
665       }
666     }
667     if(T5<=konstant*30)
668     {
669       if(T5>konstant*29)
670       {
671         PW2=246;
672       }
673     }
674     if(T5<=konstant*29)
```

```
673     }
674     if(T5<=konstant*29)
675     {
676       if(T5>konstant*28)
677       {
678         PW2=252;
679       }
680     }
681     if(T5<=konstant*28)
682     {
683       if(T5>konstant*27)
684       {
685         PW2=254;
686       }
687     }
688     if(T5<=konstant*27)
689     {
690       if(T5>konstant*26)
691       {
692         PW2=252;
693       }
694     }
695     if(T5<=konstant*26)
696     {
697       if(T5>konstant*25)
698       {
699         PW2=246;
700       }
701     }
702     if(T5<=konstant*25)
703     {
704       if(T5>konstant*24)
705       {
706         PW2=237;
707       }
```

```
807     if(T5<=konstant*10)
808     {
809       if(T5>konstant*9)
810       {
811         PW2=0;
812       }
813     }
814     if(T5<=konstant*9)
815     {
816       if(T5>konstant*8)
817       {
818         PW2=2;
819       }
820     }
821     if(T5<=konstant*8)
822     {
823       if(T5>konstant*7)
824       {
825         PW2=8;
826       }
827     }
828     if(T5<=konstant*7)
829     {
830       if(T5>konstant*6)
831       {
832         PW2=17;
833       }
834     }
835     if(T5<=konstant*6)
836     {
837       if(T5>konstant*5)
838       {
839         PW2=30;
840       }
841     }
```

```
840       }
841     }
842     if(T5<=konstant*5)
843     {
844       if(T5>konstant*4)
845       {
846         PW2=45;
847       }
848     }
849     if(T5<=konstant*4)
850     {
851       if(T5>konstant*3)
852       {
853         PW2=63;
854       }
855     }if(T5<=konstant*3)
856     {
857       if(T5>konstant*2)
858       {
859         PW2=84;
860       }
861     }
862     if(T5<=konstant*2)
863     {
864       if(T5>konstant*1)
865       {
866         PW2=105;
867       }
868     }
869     if(T5<=konstant*1)
870     {
871       if(T5>konstant*0)
872       {
873         PW2=127;
874       }
```

```
865       {
866         PW2=105;
867       }
868     }
869     if(T5<=konstant*1)
870     {
871       if(T5>konstant*0)
872       {
873         PW2=127;
874       }
875     }
876   }//while
877 }//PWM
878
879 /*****************************************************************************
880  ---------------------- function ------------------------------------------
881  *****************************************************************************/
882 /*
883 * name:       messen
884 * description:
885 * input:              -
886 * globals:            -
887 * output:             -
888 * return:             -
889
890 *****************************************************************************/
891 unsigned int mVolt, Vrms, Messfehlr,Frequenz;
892 void messen()
893 {
894   mVolt=Maxch[0]*23;//mV
895   Vrms=mVolt*0.707;
896   Messfehlr=(Maxch[0]-Maxfil[0])*23;
897   Frequenz=freq[0]*9.7//ts+40tbc+2tcpu=(1600+800+100)ns=9700ns=9700ns
898   Frequenz=1000000/Frequenz;
899 }
900
```

MAIN

MAIN ist die wichtigste Datei, In Main finde sich die Hauptfunktion

```
/*****************************************************************************
 ---------------------------------- module -----------------------------------
 *****************************************************************************
* main module:
* file name:   main.C
* description: MAIN_PROGRAM
* author:
* version:     25.04.2000
*/
/*****************************************************************************
--------------------------- revision history -------------------------
*****************************************************************************
* date:     name:  version:  description:
*
*****************************************************************************/

#include <c166.h>
#include <reg167.h>
#include <string.h>


#include "glob_def.h"
#include "sysdef.h"
#include "control.h"
#include "TxD_Function.h"
#include "RxD_Function.h"


/*****************************************************************************
---------------------- function -------------------------------------
*****************************************************************************
name:         main
description:
input:        -
globals:      -
```

```
35 input:       -
36 globals:     -
37 output:      -
38 return:      -
39
40 ********************************************************************************/
41 unsigned int Cntprueba;
42 void main (void)
43 {
44   /* constant definitions ---------------------------------------------*/
45
46
47   /* Variable definitions ---------------------------------------------*/
48   unsigned int   Counter=0;
49
50   // Fills the first n bytes of s with character c.
51   memset(TxData,0x0000, sizeof(TxData));
52   memset(RxData,0x30, sizeof(RxData));
53   fillTxd_Buffer_Statisch();
54
55   /* Hardware initializations -----------------------------------------*/
56   SYS_HW_Init();
57
58   /* Starts of HW -----------------------------------------------------*/
59   S0R = 1;    // start RS232 (baudrate generator enable)
60   T3R = 1;    // start Timer 3 of Main cycle-time
61   ADST = 1;   // start ADC
62
63   /* Global interrupt enable */
64   IEN = 1;  // enable all interrupts
65
66   // ================================================================
67   //   MAIN LOOP          -->  NEW in 30ms Steps  <--
68   // ================================================================
69   while(1)
70   {
```

```
69  while(1)
70  {
71    while(!(T3IR));   // wait of underflow for 30ms cycle
72    T3 = 37500;   // reload value for 30ms cycle with 800ns resolution
73
74    T3IR=0;     // reset T3 interr request flag
75
76    Counter++;    // increment program - cycles of 30ms steps
77
78    ADC_StopStart();  // managed ADC buffer, index, ...
79    ADC_SaveADCData();// save data from ADC Buffer into several data buffer
80    if(Counter > 33)
81    {
82      WriteDO(100,1);
83      PWMpulse();
84    }
85    if(Counter > 66)
86    {
87      WriteDO(100,0);
88      Counter = 0;
89      messen();
90      CntTBuf=0;          // Zähler für SendeBytes
91      Cntprueba++;
92      fillTxd_Buffer_DynamischWerte();
93      S0TBUF = TxData[0];// senden anstoßen
94    }  // if (Counter > 66)
95
96    ShowCalcTime();
97  }  // while(1)
98 }  // main()
```

******ADC_INTERRUPT******

ADC_INTERRUPT ist die Unterbrechung um Datei zu verwandeln (von digital zu analog)

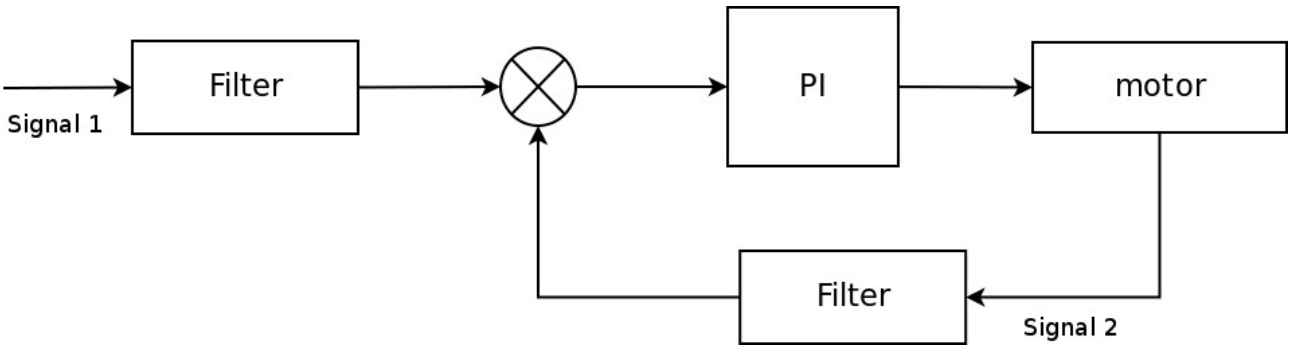
```
#include <c166.h>
#include <reg167cr.h>

/*****************************************************************************
 ---------------------- function ------------------------------------------
 *****************************************************************************/
/*
* name:               ADC_Interrupt
* description:        function will be called by ADCIR = 1
                      INTERRUPT - FUNCTION
* input:              -
* globals:            -
* output:             -
* return:             -
*
*****************************************************************************/
unsigned int ADC_BUFFER[8000];      // Puffe für 2 * 4000 ADDAT - Werte
unsigned int CntAdcBuf=0;           // Counter ADC Data-Buffer = Zeilennummer
unsigned int actWriteIndex=0;     // Schreib-Bereich im Datenfeld
unsigned int actReadIndex=4000;     // Lese - Bereich im Datenfeld


interrupt(0x28) void ADC_Interrupt(void)  {
    if(CntAdcBuf==8000)CntAdcBuf=0;
    *(ADC_BUFFER+/*filas*/CntAdcBuf+/*columnas*/actWriteIndex)=ADDAT;
    CntAdcBuf++;
}
```

******TRANSMISION_INTERRUPT******

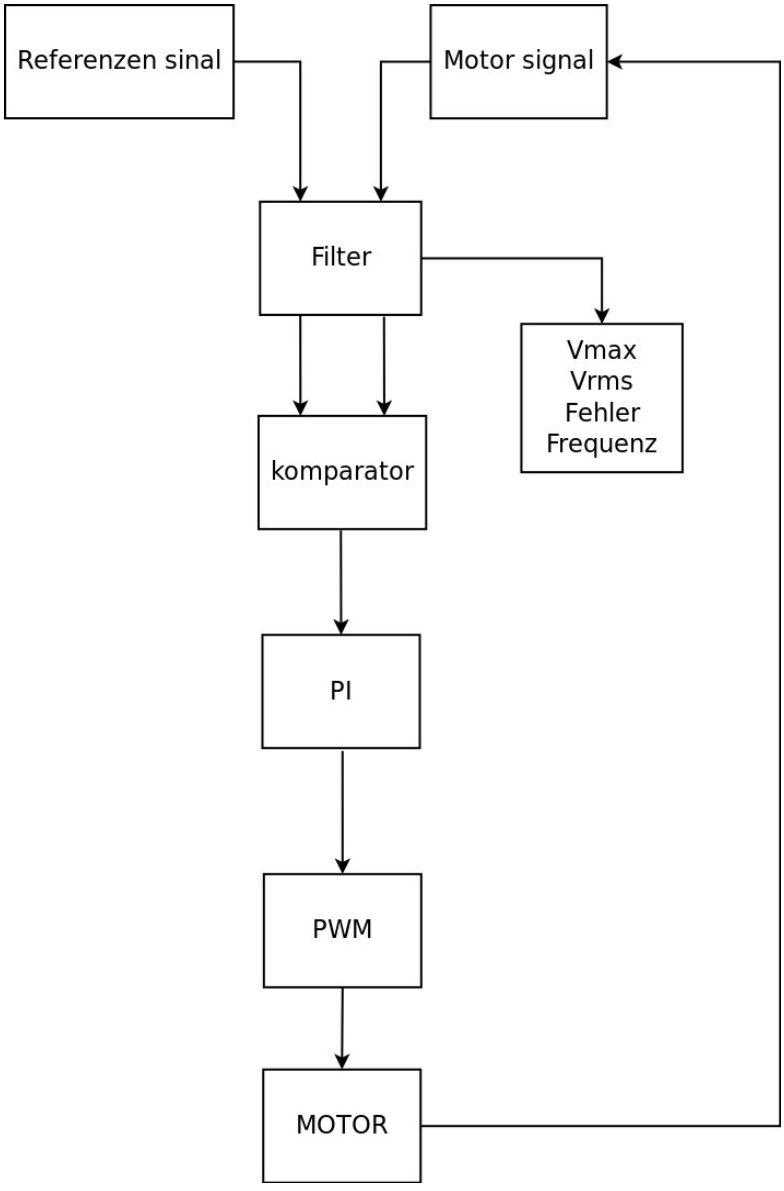
TRANSMISION_INTERRUPT schickt die Datei nach dem PC

```
#include <c166.h>
#include <reg167.h>
#include "CONTROL.H"

/*****************************************************************************
 ---------------------- function ------------------------------------------
 *****************************************************************************/
/*
* name:              TxD_Data
* description:       function will be called by S0TBIR = 1
                     INTERRUPT - FUNCTION
* input:             -
* globals:           TxData[],CntTBuf
* output:            -
* return:            -
*
*****************************************************************************/
unsigned char TxData[1024];                  // Transmit Data
unsigned int   CntTBuf=0;

interrupt(0x47) void INT_TxD_Data(void)  {

   if(CntTBuf<1024) {
   CntTBuf++;
   S0TBUF = TxData[CntTBuf];
   }

}

```

Schematische Darstellung:



Datenflussdiagramm:



2 Programmierung der Kommunikation zum PC

- S0TBUF

  SYSDEF.C:

  - **Zeile 18 bis 48:** die Eingang von Pin 3.10 muss initialisieren:

    ODP3- „output driver in push-pull“ aus

    DP3- Pin 3,10 ist eine Ausgang

    P3- Pin 3.10 initial Wert ist 0

    S0TIC- Seriell Channel 0 TransmitInterrupt Control

    S0EIC- Seriell Channel 0 Error Interrupt Control

    S0TBIC- Seriell Channel 0 Transmit Buffer Interrupts control

  CONTROL.C

  - **Zeile 207 bis 243:** hier schreibt man die Statische Variable, die immer gleich bleiben.

    Variable 0: run...

    Variable 1: mVolt (Schpannung in mili volt)

    Varialbe 2: Vrms (Wirksamwert)

    Variable 3: Messfehlr (fehler in Messung)

    Variable 4: Frequenz (Frequenzwert)
  - **Zeile264 bis 297:** hier schreibt man die Dynamischwerte, die Zahlen sein :

    Variable 1: Voltwert

    Varialbe 2: Wirksamwert

    Variable 3: fehler in Messung zwischen Maxima

    Variable 4: Frequenzwert

  MAIN.C

  - **Zeile 51:** hier legt man die Dateigröße fest.
  - **Zeile 53:** hier ruf die Statischfunktion an
  - **Zeile 93:** hier senden die Datei nach dem PC

- S0RBUF

  SYSDEF.C

- **Zeile 18 bis 48:** die Eingang von Pin 3.11 muss initialisieren:
  (Gleich als S0Tbuf)
  S0RIC: Seriell Channel 0 Transmit Buffer Interrupts control

CONTROL.C

In diese Programm, benutzt man nicht S0RBUF

MAIN.C

- **Zeile 52:** hier legt man die Dateigröße fest, wenn in die Programme benutzen muss

- BAUD

  SYSDEF.C

  - **Zeile 45:** S0BG = Baudrate9600, das ist die Geschwindigkeit wenn die Kontroller nach dem PC die Datei schickt.

- ASCII

  Die PC zeigt die Datei als die PC die Datei bekomme, aber man muss die Datei verstehen, um das zu machen, muss man in ASCII übersetzen.
  In **CONTROL.C Zeile 177 bis 189** eine Funktion übersetze die Datei nach ASCII, damit kann man die Nummer verstehen.

3 ***Programmierung: Einlesen von Spannung und Strom***

- Arbeitsweise der Analogwertverarbeitung mittels:
  - Polling:

    Prozessorzeit wird verschwendet beim unnötigen Durchtesten des Status aller peripheren Einheiten in jedem Durchlauf.
    Sie ist vom Prinzip her langsam, da der Status aller E/A-Einheiten getestet werden muß, bevor man zur Abarbeitung einer bestimmten Anfrage kommt. Das kann in einem Echtzeitsystem, in dem man eine Peripherieeinheit, in einem festgelegtem Zeitabschnitt bearbeitet werden muß, echte Schwierigkeiten bereiten. Insbesondere, wenn schnelle Peripherieeinheiten an das System angeschlossen sind, kann die Abfragetechnik einfach nicht

schnell genug sein, um noch eine rechtzeitige Bearbeitung der Anfrage zu gewährleisten.

- Interrupts:

  Programmunterbrechungen, kurz Unterbrechungen (Interrupts), sind ein asynchroner Mechanismus. Jede E/A-Einheit oder ihr Steuerbaustein ist an eine Unterbrechungsleitung angeschlossen. Diese Leitung überträgt eine Unterbrechungsanforderung (interrupt request) an den Mikroprozessor. Jedesmal, wenn eine der E/A-Einheiten bedient werden muß, erzeugt sie einen Impuls oder einen bestimmten Pegel auf dieser Leitung, um den Mikroprozessor auf sich aufmerksam zu machen.
  Ein Mikroprozessor testet am Ende jedes Befehlszyklus, ob eine Unterbrechungsanfrage vorliegt. Ist dies der Fall, wird die Unterbrechung durchgeführt. Ist keine Anfrage vorhanden, wird der nächste Befehl übernommen. Dies ist in untenstehendem Flußdiagramm dargestellt.
  Werden kritische Prozesse bearbeitet, muß sichergestellt sein, daß die Programmabarbeitung nicht durch eine Unterbrechung gestört wird. Jedesmal, wenn das Maskierungsbit eingeschaltet ist, werden die Unterbrechungsanforderungen nicht beachtet. Die „Maskierungsfähigkeit“ wird oft als „Aktivierung“ (enable) bezeichnet. Eine Unterbrechung ist aktiviert, d.h. ermöglicht, wenn sie nicht maskiert ist.

  SYSDEF.C

  - Die **Zeile 32, 38, 74, 97, 177 y 215** zeigen die Priorität dieses Interrupts

- PEC –

  PEC ist ein Peripheriegerät, das Begebenheit kontroliert, das PEC lasst die wünschende Unterbrechung zu, um mit einfachen Daten (Wort-oder Byte), die nur einen Befehlszyklus verbraucht und erfordert keine Speichern des Zustands zu beantworten, ist jeder Unterbrechung in dem Programm priorisiert, wenn Unterbrechung nennen CPU, die es Prioritätsstufe zu denen Interrupt Handlungen zu bestimmen testet, wird der aktuelle Zustand zur Wiederverwendung genutzt gespeichert.

- Filter

  CONTROL.C

  Zeile 483 bis 506: Hier macht man die Gleichung, um die Filter zu erzeugen Der Filter ist unter der folgenden Gleitung gemacht :

  $$A(t)=K_p K_o AdcChXBuf(t)temp0+K_o A(t-1)temp1$$

  A(t)=Ausgang

  K_p=konstant

  AdcChXBuf(t)=AdcCh0Buf(t) und AdcCh1Buf(t)

  temp0= konstant zeitlich 1

  temp1= konstant zeitlich 2

  $$K_0=\frac{1}{temp0+temp1}$$

- Glättung der Größen

  CONTROL.C

  - **Zeile 548 bis 573:** Hier sucht man die Manima von der Welle, Maxch[1] oder Maxch{2} abhängen von die Welle.

- Ermittlung der Effektivwerte:

  CONTROL.C

  - **Zeile 895:** Maximalspannung mal 0,707, das ist die Effektivwerte

- Ermittlung der Messfehler, der digitalen Auflösung:

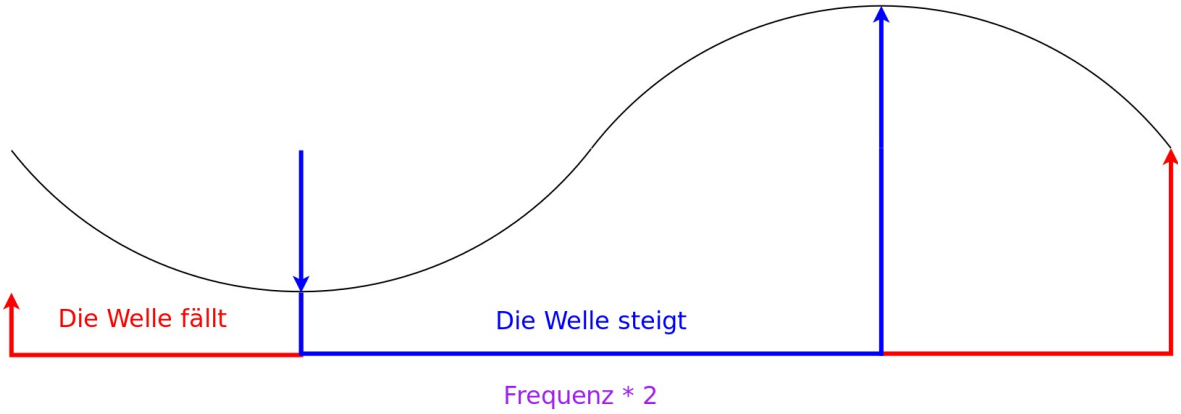
  CONTROL.C

  - **Zeile 896:** Maximalspannung minus Maximalfilter.
    Man multipliziert mit 23, weil 1024 ist der Maximal von ADC Converter, und 24 ist die Maximal Spannung , denn 24/1024=0,023=23mV

4 Programmierung: Ermittlung der Frequenz und Phasenverschiebung

ю Ermittlung der Frequenz aus dem Spannungssignal:
CONTROL.C

- **Zeile 507 bis 546:** Hier zählt man die Wiederholung, während des Steigens die Welle



ю Bestimmung der Güte der Messergebnisse:

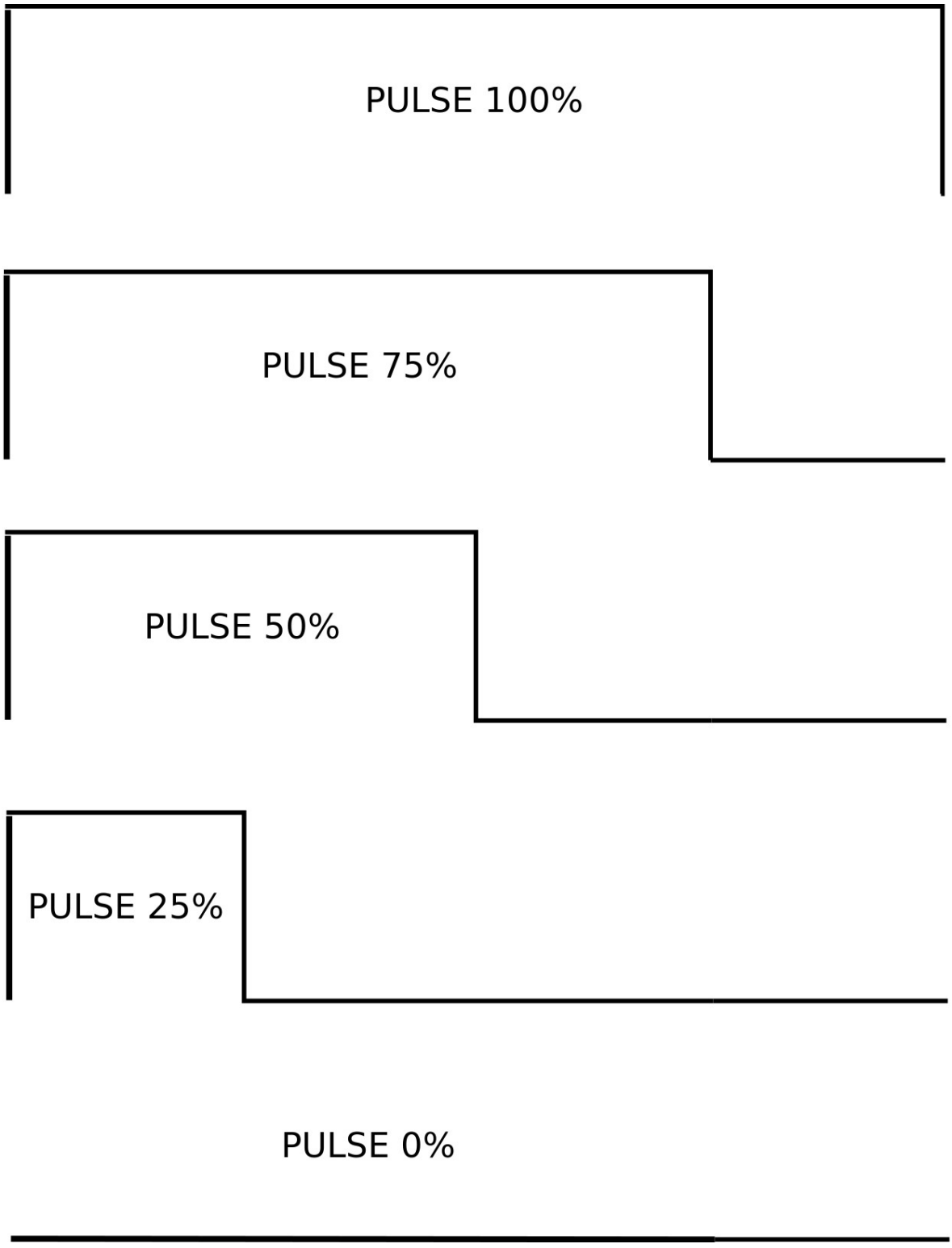
- **Zeile 591:** wenn fehler ist Positiv, die Spannung ist wiederstandfähig, wenn Negative, Kapazitiv

5 Inbetriebnahme des analogen Ausgangs

ю Programmierung der PWM-Einheit:
SYSDEF.C

- **Zeile 236 bis 260:**
  Man konfiguriert die Pins als Ausgang (P7) und die Register (PWMCON0), danach, man konfiguriert die Pulsweit als 0.

ю Bestimmung der Dynamik des analogen Ausgangs:
Man konfiguriert PW2 um die Anfangswert zu bekommen, wenn man die 100% haben will, den PW2=255, wenn 50%:127 und wenn 0%:0.


PULSE 100%
PULSE 75%
PULSE 50%
PULSE 25%
PULSE 0%

6 Programmierung eines digitalen PI – Reglers

- Umsetzung eines Spannungsreglers:

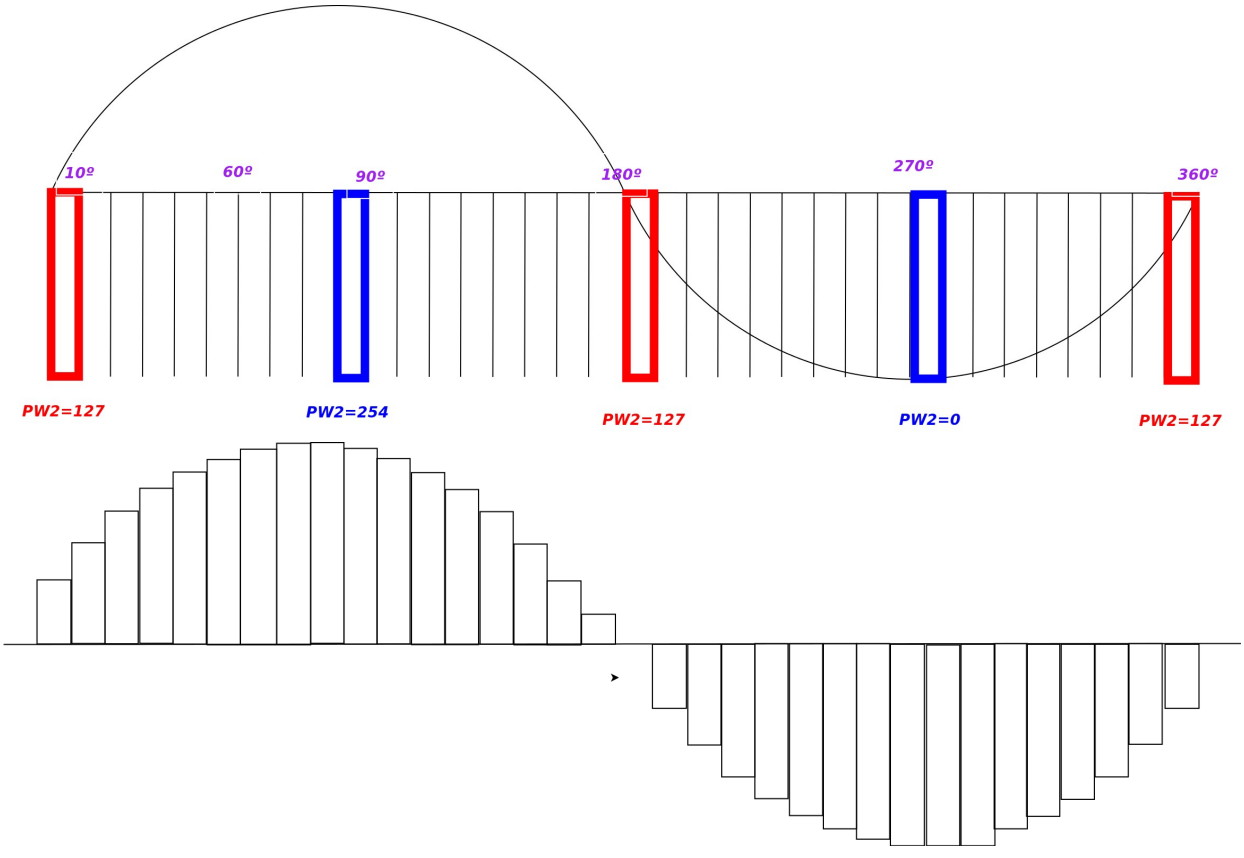
  Die folgende Funktion ist die Addition von dem P-Kontroler und I-Kontroler.

$$PI(t)=P(t)+I(t)=K_c\Theta(t)(1+\frac{T}{2T_i})-K_c\Theta(t-1)(1-\frac{T}{2T_i})+PI(t-1)$$

  Diese Funktion ist in **Zeile 587 bis 595** angewandt , wenn die Ausgang Negativ ist, die PWM ist langsamer als wenn Positive die Ausgang ist, das schwankt bis der Kontrolle stabilisiert ist.

7 Trace-Buffer für dynamische Vorgänge

- Entwerfung eines Tracebuffer für den Regler zur Kontrolle der dynamische Vorgänge: Das folgende Bild zeigt, wie die Regler kontrolliert ist, in **Zeile 608 bis 877** ist eine Tabelle gemacht, diese Tabelle zeigt 36 Werte für der verschieden Grad , hier zeigt man die generell Idee und unter dem Bild, zeigt man, wie die wirklich Welle gemacht ist.



8 Bibliografie:

- http://www.referate10.com/referate/Technik/8/Interrupts-reon.php
- www.wikipedia.de
- www.infineon.com
- Bericht von Herr Schmied