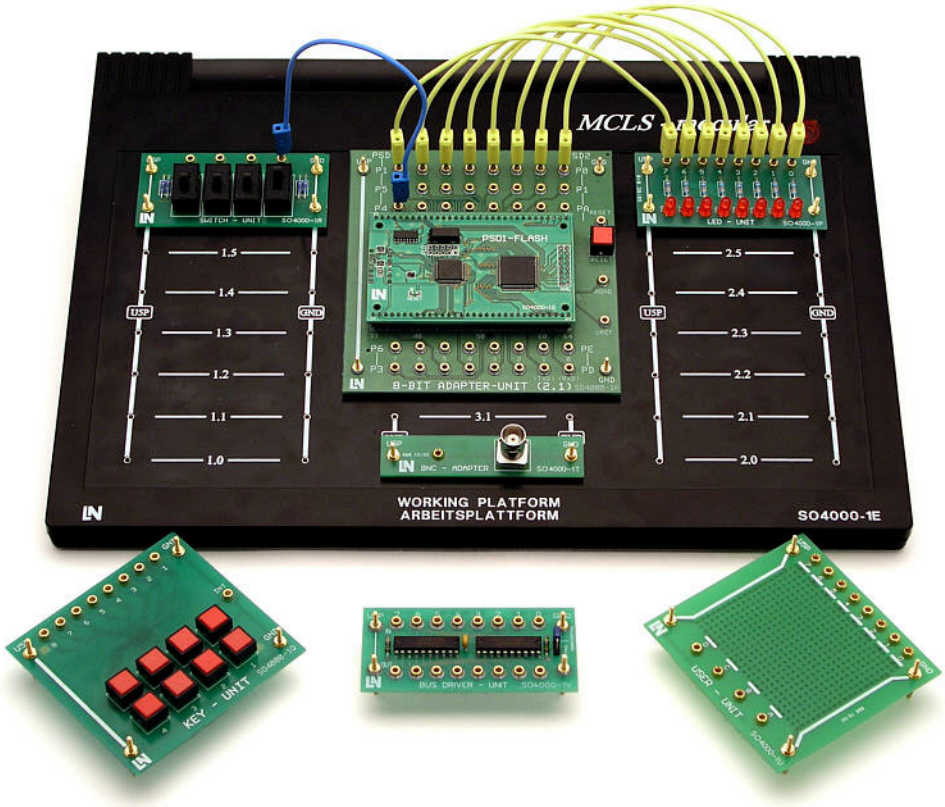
Introducción a la programación de microcontroladores 8051 (C515C)

CMC 1



Guía de ejercicios para el estudiante

SH5004-1A

3ra. edición

Autor: equipo de autores ACMC Mittweida

Lucas-Nülle Lehr- und Messgeräte GmbH
Siemensstraße 2 • D-50170 Kerpen

Índice

Objetivo de los ensayos

1. Transmisión de informaciones fundamentales para el entorno de programación compuesto por hardware y software, definiciones de conceptos, así como entrenamiento del manejo de los componentes. En todos los siguientes ensayos se utilizarán los conceptos del entorno de programación sin más comentarios.
2. Los aprendices / estudiantes se familiarizarán con la estructura del MC y su periferia. Conocerán el conjunto de comandos del MC.
3. Los aprendices / estudiantes se familiarizarán con la programación estructurada desde el problema hasta la solución mediante un ejemplo sencillo.

Resumen de ejercicios

CMC 1-1 Primeros pasos en la programación de un microcontrolador

- Introducción al entorno de desarrollo integrado
  Los componentes del software y del hardware
  Aclaración de conceptos
- Preparación del IDE y su utilización mediante un ejemplo de aplicación
- Tratamiento de errores típicos
- Trabajar con el Depurador de errores, entre otras cosas Configurar, Servicio por pasos de proceso, Servicio por pasos individuales, Puntos de interrupción

CMC 1-2 El microcontrolador C515C

- Estructura y funcionamiento de un microcontrolador (estructura interna)
- Periferia de un microcontrolador (puertos, reloj, reposición)
- Funcionamiento de un microcontrolador (temporización, unidad aritmética lógica, puertos I/O)
- Estructura de memoria del microcontrolador
- Lista de comandos del microcontrolador
- Mediante pequeños ejemplos de aplicación se presenta el funcionamiento del microcontrolador y se ilustra el control por medio del conjunto de comandos.

CMC 1-3 Un primer programa con el microcontrolador C515C

Programación de un primer programa por medio de los escalones:

- análisis de problema,
- solución,
- diseño estructural,
- programación,
- ensayo

mediante el ejemplo de una luz de desplazamiento sucesivo
Introducción a la programación estructurada mediante el ejemplo de una luz de desplazamiento sucesivo

Instalación del ensayo

Para la realización del ensayo efectiva, segura y sin fallos se indican a continuación por separado:

- Tabla con todos los módulos, aparatos, líneas de medición y accesorios necesarios.
- La instalación de aparatos en la mesa de trabajo.

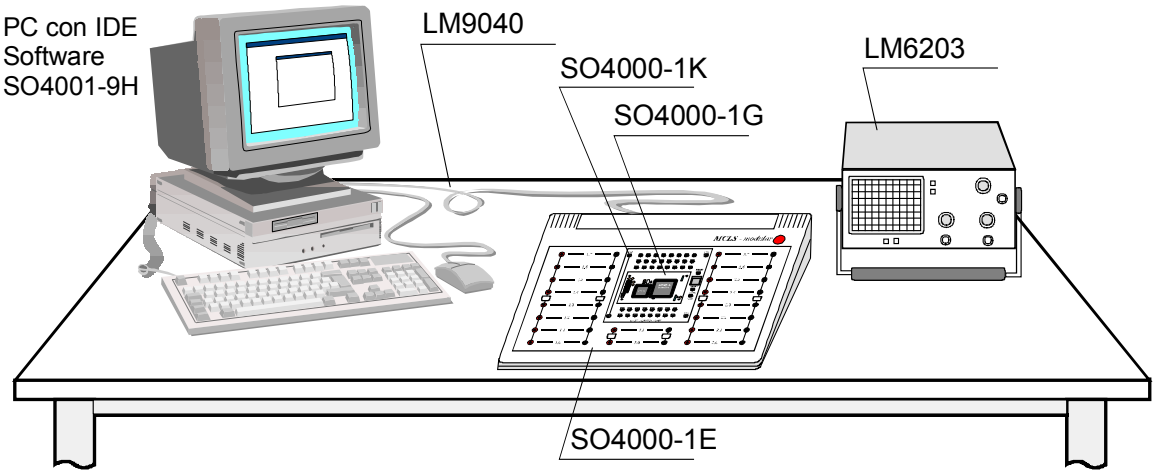


Fig. 01: Instalación en el puesto de trabajo

Recomendamos se sirvan de los manuales de instrucción de los módulos empleados en la realización del ensayo puesto que representan una ampliación valiosa para los respectivos ejericicios.
Los puntos de medición, los interruptores y los elementos de ajuste en los módulos se explican por separado. Los manuales de instrucciones también contienen indicaciones de aplicación fundamentales. Utilice para una enseñanza efectiva las ofertas del Internet bajo:

http://www.mcls-modular.de

donde, además de las listas, se ofrecen ayudas e instrucciones para el entorno de desarrollo integrado.

Componentes y aparatos necesarios

Unid.	Designación	Nº Id
1	Plataforma con módulo de alim. +5V	SO4000-1E
1	Fuente de alim. de enchufe AC 90...230V 45..65Hz, DC 9V 630mA	SO4000-1F
1	Módulo PSD1-FLASH Controlador C515C	SO4000-1G
1	Unidad de adaptador de 8 bit	SO4000-1K
1	Unidad de LED	SO4000-1P
1	Unidad de teclas	SO4000-1Q
1	Unidad de interruptores	SO4000-1R
1	Unidad de display 1	SO4000-1S
1	Adaptador de BNC	SO4000-1T
1	Unidad de usuario	SO4000-1U
1	Software IDE para MCLS (D,GB,F,E)	SO4001-9H
1	Osciloscopio incl. puntas de prueba	LM6203
1	Sonda de prueba lógica	LM8101
1	Cable de medición BNC / BNC	LM9034
1	Cable de interfaz en serie 9/9 polos	LM9040
8	Línea de medición 2mm 15cm azul	SO5126-5K
8	Línea de medición 2mm 15cm amarillo	SO5126-5M
1	CMC 1 Introducción a la programación de microcontroladores 8051	SH5019-1A
1	Maleta de almacenam. para MCLS	SO4000-1W

Introducción

Los microcontroladores han llegado a tener una importancia decisiva que no siempre es perceptible a primera vista. Estos sistemas casi siempre trabajan „en segundo plano“, sin ser reconocidos, pero sí de manera muy eficaz. La alta o más alta integración de procesador, RAM, ROM, EEPROM, Flash y de otras funciones permite unos campos de aplicación los cuales hace unos años aún eran inimaginables. Los sistemas de microcontrolador se emplean hoy p.e. en :

- Controles
- Sistemas de regulación
- Sistemas de dirección
- Sistemas de transmisión

El sistema de aprendizaje de microcontrolador modular MCLS-modular® se desarrolló especialmente para la formación en el campo de la „Técnica de microcontrolador“. Permite estudiar la estructura de hardware, así como la programación apoyada por software de un microcontrolador de la serie de construcción 8051.
El entorno de desarrollo integrado de Windows IDE permite, tras una breve iniciación, elaborar, ensayar y optimizar programas de microcontrolador. Para el empleo en el MCLS-modular, nos orientamos en el empleo de Depuradores de monitores o simuladores con función de Cargador de arranque inicial o de programación para la detección de fallos en programas. Depende del sistema final para el cual deben estar disponibles las correspondientes herramientas, qué herramienta se va a emplear. Con ello, el desarrollo de la programación corresponde a la vida cotidiana industrial.

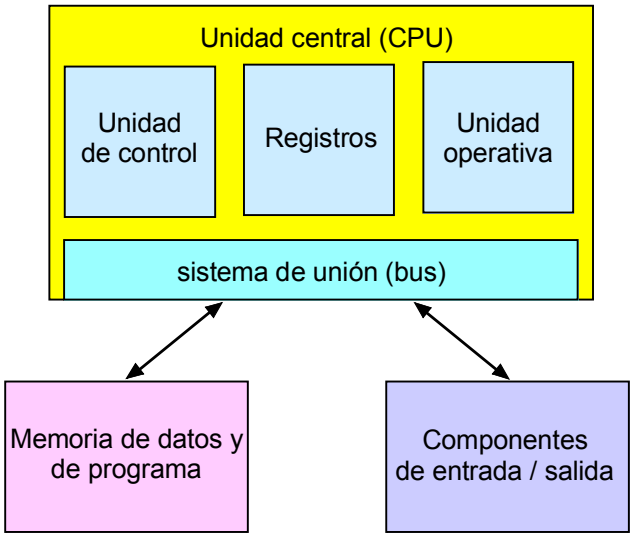


Fig. 02: Estructura básica de un ordenador

Los microprocesadores son el elemento central de cualquier microordenador. Como ordenador designamos generalmente una instalación de procesamiento de datos controlada por programas. Qué datos se procesan, o sea el objeto de aplicación, es determinado por la estructura y con ello la capacidad necesaria del microprocesador empleado. Esto se hace más claro cuando se compara un ordenador grande con una tarjeta de chip, ya que en ambos casos se trata de un ordenador.

Originalmente, el microprocesador - el CPU de un sistema de ordenador realizado en un chip - estaba concebido para solucionar cálculos numéricos. Hoy, los microprocesadores se emplean en un amplio entorno técnico, o sea el volumen de funciones se ha ampliado y especializado. Pensando en la variedad de campos de empleo, como los ordenadores de bordo de vehículos modernos, los procesadores de señales en la técnica de comunicación o también los ordenadores personales potentes, podemos realizar la siguiente clasificación:

- Microprocesadores estándar
- Microprocesadores de alto rendimiento
- Microcontroladores
- Procesadores de señales digitales

En este manual de ensayos, nos vamos a dedicar a la programación de microcontroladores generales.
Los microcontroladores generales constituyen probablemente una de las formas más compactas de un microordenador.

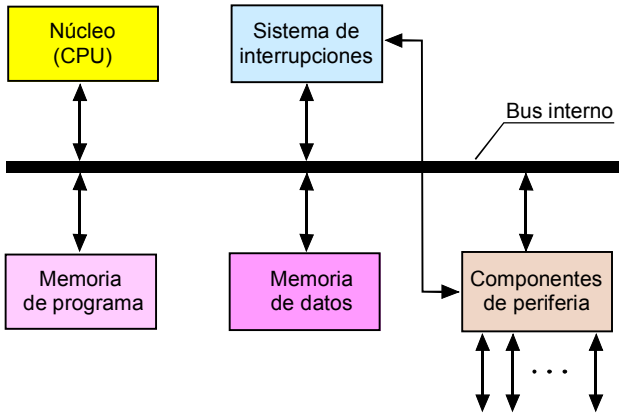


Fig. 03: Estructura básica de un microcontrolador

Un microcontrolador es un sistema de microordenador completo en un chip. El núcleo del microcontrolador (Core), la memoria, los componentes de periferia y el sistema de interrupción están integrados juntos en un chip y conectados entre ellos por medio de uno ó varios buses. [1]

Por „core“ o núcleo de microcontrolador entendemos la CPU integrada en el chip, que dispone en general de un mecanismo de control, un mecanismo de cómputo, un registro y de un control de bus.
En los microcontroladores, lo más importante es la periferia on-chip y no el rendimiento de cálculo. Ejemplos para la periferia on-chip son:

- Oscilador generador de la señal de reloj
- Puertos de entrada y de salida
- Temporizador
- Interfaces de comunicación (UART, I^2C, CAN, SPI ...)
- Convertidor analógico-digital
- Convertidor digital-analógico
- Interrupciones externas
- Real Time Clock (RTC) (reloj de tiempo real)
- Soporte para depuración

A menudo sucede que los componentes de periferia integrados no son suficientes para una aplicación, por lo que se deben completar por componentes externamente conectados. En la mayoría de los casos, esto se refiere a la memoria.

El desarrollo de programa en microcontroladores se realiza de manera próxima al procesador mediante el lenguaje de ensamblaje y el ensamblador correspondiente al microcontrolador o en un lenguaje de alto nivel (C o C++) y un compilador. El ensamblador o el compilador traducen el código fuente escrito en formato de texto a un ccódigo binario comprensible para el microcontrolador. Puesto que un microcontrolador representa un microordenador específico, tanto el ensamblador empleado como el compilador deben disponer de los conjuntos de comandos correspondientes al microcontrolador. En otro caso, el código generado no se puede ejecutar o se ejecuta incorrectamente.

La pregunta cuál de las dos herramientas de desarrollo se emplea depende de:

- la disponibilidad de estas herramientas para el MC,
- el requisito de capacidad de tiempo real,
- la compacidad del código generado,
- el volumen de las actividades de programación.

Dentro de los trabajos de desarrollo se emplean otras herramientas adicionales. Los dispositivos de enlace (Linker) insertan las funciones existentes de bibliotecas en el código de programa. Los convertidores generan del archivo de Ensamblador un archivo Intel Hex apto para descargar. Los sistemas de depuración sirven para la puesta en servicio y la optimación de código.

Qué herramientas de desarrollo finalmente se emplean depende de la aplicación por la selección del microcontrolador y los posibles costes de desarrollo. En los microcontroladores de 8 bits es muy habitual trabajar con Ensambladores y Depuradores de monitor.
En los microcontroladores de mayor integración, sobre todo en los microcontroladores de 16 y 32 bits se emplean plataformas de lenguaje de alto nivel para el desarrollo de aplicación. Éstas son apoyadas en los microcontroladores más nuevos por Depuradores „on chip“. Sin embargo, estos entornos de programación comfortables presuponen la existencia del interfaz necesario al microcontrolador.

[1] Taschenbuch Mikroprozessortechnik (Libro de bolsillo Técnica de microprocesador), Thomas Beierlein, Olaf Hagenbruch, 3ra. edición, 2004, Fachbuchverlag Leipzig

CMC 1-1 Primeros pasos en la programación de un microcontrolador

Manejo del entorno de desarrollo integrado, generación de un código de programa cargable, cargar el programa, manejo del Depurador

Fundamentos preliminares

Para que un microcontrolador pueda ejecutar una secuencia de comandos, es necesario archivar en su memoria de programa (memoria de códigos) un programa legible para la máquina.
Un procedimiento habitual es cargar la memoria de códigos con un archivo INTEL-HEX (secuencia definida de signos hexadecimales en formato ASCII) el cual respresenta el código de programa legible para máquinas.
Un código de programa se genera elaborando en un editor de texto un texto fuente según determinadas reglas para transformarlo a continuación por medio de una herramienta de desarrollo especial (Softwaretool). Este código puede ser un código intermediario o ya es el mismo archivo INTEL-HEX. Los ensambladores y compiladores (C) son herramientas de desarrollo que realizan esta transformación. Otras herramientas (p.e. Convertidores, Linker) procesan el código intermedio hasta obtener un código de programa cargable.
Los programas acabados se archivan por el fabricante por medio de programación por máscaras o por el usuario por medio de unidades de programación en la memoria de programa. La estructura física de la memoria de programa decide sobre el tipo de programación posible.
Para el ensayo de programas durante su elaboración se emplean emuladores o sistemas de desarrollo especiales (hardware y software, Depuradores).

Para utilizar estas diferentes herramientas de desarrollo y emplearlas de manera específica para cada proyecto, se dispone del entorno de desarrollo integrado IDE que inserta todas las herramientas bajo una superficie de manejo uniforme.

Para la realización de todos los ensayos se utilizan las siguientes herramientas de software :

- Entorno de desarrollo integrado (IDE)
- Editor de texto integrado del IDE
- Ensamblador (**as.exe**, genera el código intermedio)
- Convertidor (**p2hex.exe**, genera del código intermedio el archivo INTEL-HEX)
- Depurador (**debugger.exe**)
- Monitor de objetivo (programa especial para la carga y el ensayo de programas de usuario, instalado en el hardware del módulo PSD1-FLASH)– ver manual de instrucciones del módulo PSD1-FLASH (SO4000-1G).

La carga del código de programa generado (archivo INTEL-HEX) en la memoria de código del microcontrolador se realiza del interfaz en serie (RS232) de su PC al interfaz en serie del microcontrolador. El monitor de objetivo archiva el programa en la memoria de programa.
La ejecución del programa en el microcontrolador (ejecución en tiempo real, servicio por pasos individuales, servicio por pasos de proceso) es iniciada por el Depurador (herramienta de software en su PC) por medio de comandos.

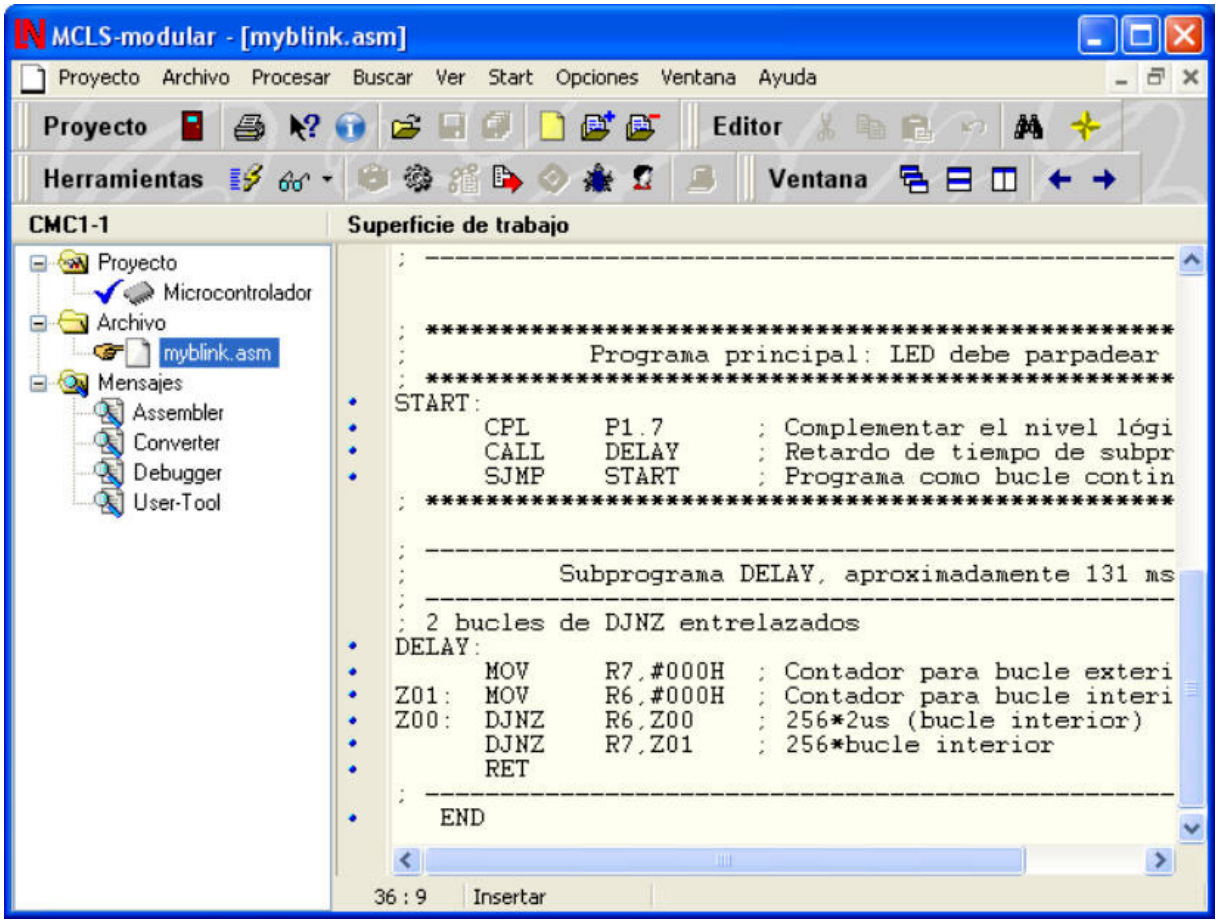


Fig. 101: IDE del *MCLS-modular®*

El IDE del MCLS-modular reúne bajo una misma interfaz de usuario las herramientas de desarrollo necesarias para la programación. El IDE está dividido en tres áreas. En la parte izquierda se encuentra el navegador, y en la derecha la superficie de trabajo con el editor de texto fuente. A través de los botones de la barra de herramientas pueden llamarse las diversas funciones del entorno de desarrollo.
Para que el estudiante se familiarice más rápidamente con la programación, el entorno cuenta con "perfiles integrados". Dichos perfiles son módulos de software con todas las herramientas de programación y ajustes de configuración necesarios para el controlador objeto de estudio. Entre los perfiles integrados se encuentra, por ejemplo, el PSD1_C515C_AS con los siguientes componentes:

- Ensamblador
- Convertidor
- Generador de archivo de símbolos
- Depurador

Los componentes inherentes del entorno de desarrollo integrado son:

- Editor de texto fuente
- Administrador de archivos

La descarga de un programa de aplicación en el sistema de destino se realiza con el depurador.

Para trabajar en un proyecto nuevo, hay que seguir los siguientes pasos:

(1) Crear un proyecto nuevo en el IDE del MCLS
(2) Generar e integrar el archivo fuente principal en el proyecto
(3) Editar, ensamblar y convertir un texto fuente propio en ensamblador

La ejecución del programa en el microcontrolador (en tiempo real, por pasos o por rutinas) es iniciada por instrucciones del depurador (herramienta de software en el PC).

Una vez instalado el software, hay que iniciar el IDE a través del icono del escritorio, o a través del botón Inicio → Programas → MCLS-modular. Las instrucciones de instalación se encuentran en la ayuda en línea del sitio web del mcls-modular (www.mcls-modular.de), o como copia en el CD de instalación.
Un componente importante es el administrador de módulos, el cual permite integrar actualizaciones de programa o módulos adicionales. Los módulos complementarios son archivos especiales comprimidos en formato CAB que, una vez instalados, son guardados en la carpeta "Módulos". Los módulos complementarios más recientes pueden descargarse del sitio web. Éstos deben copiarse directamente al directorio "Módulos" y nunca abrirse o descomprimirse haciendo doble clic.
Antes de reiniciar el IDE, el administrador de módulos detecta automáticamente el nuevo módulo y le preguntará si desea integrarlo.

Nota:
Con el módulo PSD1-Flash se han puesto a disposición dos perfiles nuevos en forma de módulos complementarios. Si éstos no se encuentran en la selección ofrecida por el administrador de módulos (IDE versión 3.04 ó anterior), deberán copiarse desde la fuente antes citada al directorio "Módulos". Esto se aplica a los módulos "PSD1_C515C_AS" y "PSD1_C515C_SDCC" con todas sus herramientas y perfiles del mismo nombre para programar el C515C en ensamblador y en C, respectivamente.

Después de haber hecho la nueva instalación, se recomienda activar la opción de almacenamiento automático de todos los archivos de proyecto y archivos fuente. Esta función hace que, antes de realizar la traducción de un programa, cualquier modificación realizada en un archivo del proyecto sea primero automáticamente guardada.
Para activar la función haga lo siguiente:
(4) En el IDE, vaya al menú "Opciones/IDE/Propiedades..."
(5) En el cuadro de diálogo que aparece, haga clic en "Editor".
(6) Ahora, en el cuadro "Guardar todos los archivos antes de la ejecución" active la opción "Guardar todos los archivos sin confirmación".
(7) Para aceptar haga clic en "OK".

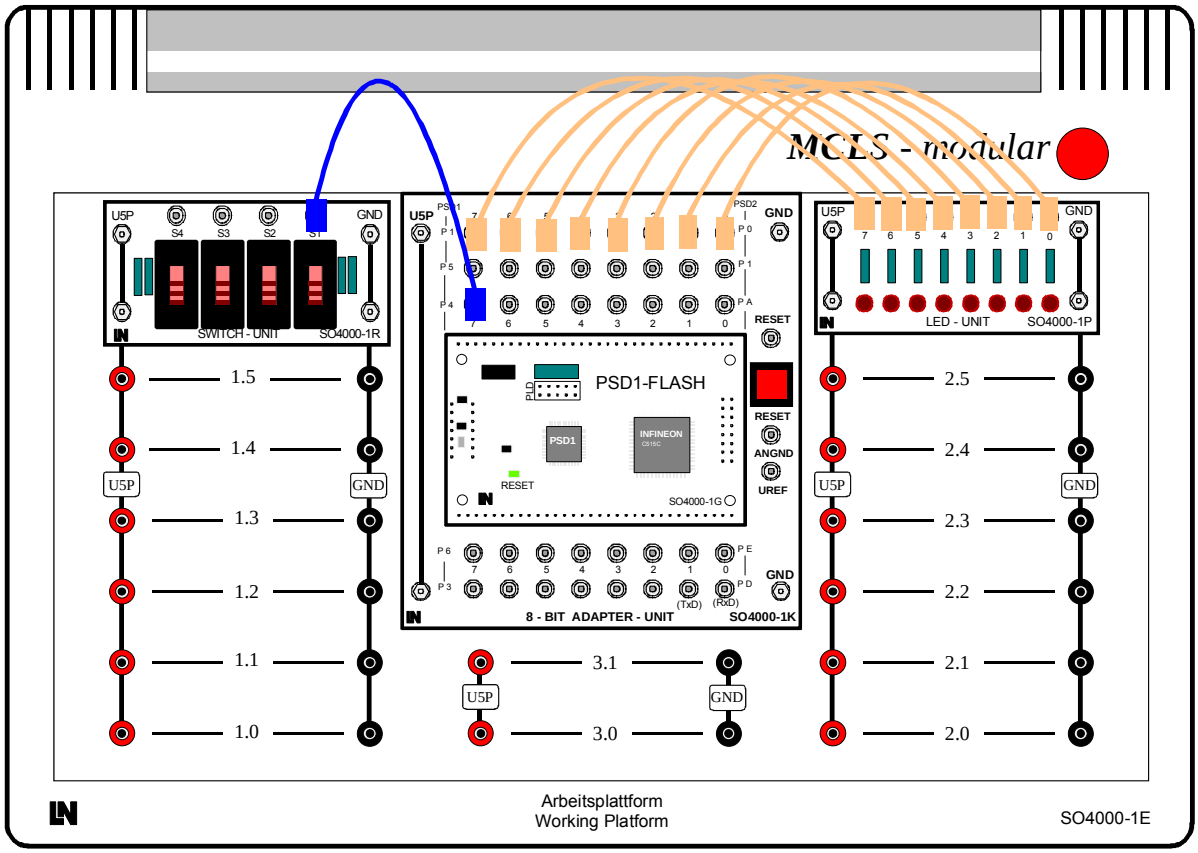


Fig. 101: Instalación de aparatos del ensayo CMC 1-1

Instalación experimental CMC 1-1

Aparatos y componentes necesarios

Unid.	Designación	Nº Id
1	Plataforma con mód. de alim. +5V	SO4000-1E
1	Fuente de alim. de enchufe AC 90..230V 45..65Hz, DC 9V 630mA	SO4000-1F
1	Mód. PSD1-FLASH Controlador C515C	SO4000-1G
1	Unidad de adaptador de 8 bits	SO4000-1K
1	Unidad de LED	SO4000-1P
1	Unidad de teclado	SO4000-1R
1	Software IDE para MCLS (D,GB,F,E)	SO4001-9H
1	Cable de interfaz en serie 9/9 pol.	LM9040
1	Línea de medición 2mm 15cm azul	SO5126-5K
8	Línea de med. 2mm 15cm amarillo	SO5126-5M
1	CMC 1 Introducción a la programación de microcontroladores 8051	SH5019-1A

Instalación de aparatos

Separe la alimentación de corriente del MCLS-*modular* !
Conecte la plataforma de trabajo por medio de su PC.
Encaje la unidad de adaptador de 8 bits con el módulo PSD1-FLASH adaptado en ésta en el campo de casquillos (CC) 4.0 - 4.5 en el centro de la plataforma. Tenga cuidado con las clavijas de contacto en el centro de la unidad de adaptador (TxD, RxD)! Encaje la unidad de LED a la derecha de la unidad de adaptador de 8 bit (p.e. CC 2.6 – 2.7). Úne sucesivamente LED 0 con el puerto P1 casquillo 0 (P1.0), LED1 con P1.1, LED2 con P1.2 etc. hasta LED7 con P1.7. Coloque la unidad de teclado en CC 1.6 – 1.7 en la plataforma de trabajo. Conecte S1 al puerto 4 casquillo 7 (P4.7)
Conecte la alimentación de corriente al MCLS-*modular*.

Nota relativa a la seguridad:
¡Asegúrese de conectar correctamente los potenciales de tensión (izda U5P, dcha GND)! No se deben aplicar tensiones extrañas a los puntos de medición llevados hacia fuera ¡ya que esto podría conducir a la destrucción de los componentes de las placas!

Indicación relativa a la literatura

Recurra para la realización del ensayo al manual del entorno de desarrollo integrado (IDE) y al manual de instrucciones de la unidad PSD1-FLASH.

Ejercicio 1:
Elaboración de un proyecto propio

Objetivo :

- Especificación/Selección de un perfil de microcontrolador
- Inserción de las herramientas de desarrollo
- Asignación de herramientas de desarrollo para la función MAKE
- Inserción de un archivo de texto fuente
- Guardar el archivo de texto fuente con otro nombre
- Declarar un archivo principal
- Archivar el proyecto
- Imprimir contenidos de proyecto
- Ensayo del programa en el MCLS-*modular*
- Cerrar un proyecto
- Finalizar el IDE

Ejercicio :

Elabore el proyecto „CMC1-1“.
Inserte el archivo de texto fuente „cmc111.asm“ y guárdelo con el nombre „myblink1.asm“.
Declare „myblink1.asm“ como archivo principal.
Archive el proyecto actual. Imprima la información de proyecto, el perfil de herramienta, así como el archivo de texto fuente.
Ensaye el programa en el MCLS-modular.

Forma de proceder :
Inicie desde el menú principal de el entorno de desarrollo integrado para el MCLS-*modular* (botón Inicio ⇨Programas ⇨MCLS-*modular*).

- Elabore un perfil de microcontrolador propio
  - Clic de en „Microcontrolador “ en el árbol de proyecto (o Menú Opciones ⇨Proyecto ⇨Microcontrolador)
  - Active la opción „Perfiles de usuario “ en la selección de perfiles ⇨ Clic de
  - En la selección de perfiles integrados para el microcontrolador, seleccione el perfil "PSD1_C515C_AS".

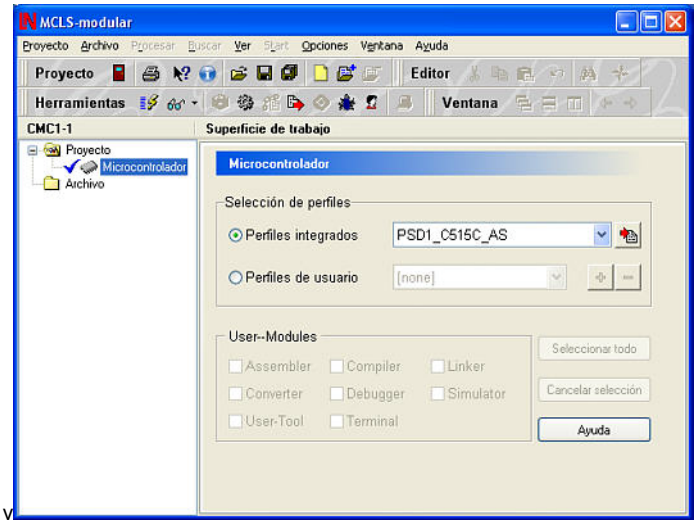


Fig. 103: Selección de perfiles para proyecto nuevo

Seguidamente, debe guardarse el proyecto en una carpeta.

- Guarde el proyecto:
  - Menú Proyecto ⇨Guardar proyecto ⇨Seleccionar directorio ⇨Introducir nombre de archivo deseado (tipo de archivo: archivo de proyecto MCLS) ⇨Guardar

Para generar un programa ejecutable, debe crearse primero un archivo fuente principal e integrarlo en el proyecto a través del menú Archivo ⇨Archivo nuevo.

- Creación e integración de un archivo fuente principal:
  - Clic en "Archivo" (árbol de proyecto) ⇨Clic con botón dcho del ⇨Insertar archivo en el proyecto ⇨Seleccionar archivo ⇨Abrir
  - (o menú Archivo ⇨Insertar archivo en el proyecto ⇨Seleccionar archivo ⇨Abrir)

NOTA: Es posible insertar varios archivos en un proyecto.

- Guarde el archivo de texto fuente actual (marcado) con otro nombre:
  - Menú Archivo ⇨Guardar archivo actual como. ⇨Seleccionar directorio ⇨Introducir nombre de archivo deseado (tipo de archivo: ensamblador) ⇨Guardar

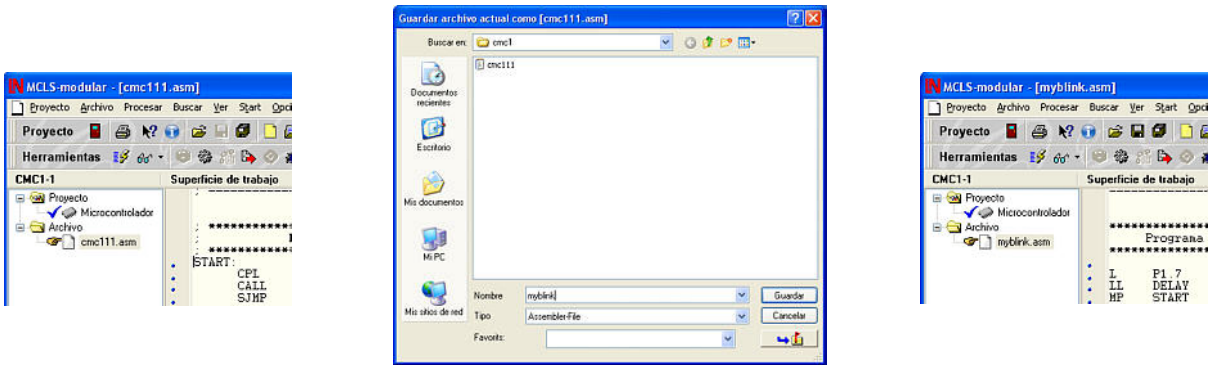


Fig. 104: Guardar el archivo fuente principal

El archivo creado deberá guardarse con la terminación *.asm* en el directorio de proyectos y declararse como archivo principal *(set mainfile)* en el árbol de proyecto haciendo clic sobre el mismo con el botón derecho del ratón.

- Declare el archivo actual (marcado) como archivo principal:
  - Clic en el archivo deseado en la rama del árbol de proyecto "Archivo" (para marcarlo) ⇨Clic con botón dcho del ⇨Set mainfile

ATENCIÓN: ¡Es imprescindible declarar un archivo principal, porque las herramientas solamente procesan el archivo de texto fuente declarado como tal! Sólo puede declararse un archivo como principal.

Nota:
Al guardar el archivo fuente, es imprescindible definir en tipo de archivo "Assembler-File" (ensamblador). De lo contrario pueden producirse errores al guardar el archivo con otra terminación, y éste no será compilado.

- Guarde el proyecto:
  - Menú Proyecto ⇨Guardar proyecto ⇨Seleccionar directorio ⇨Introducir nombre de archivo deseado (tipo de archivo: archivo de proyecto MCLS) ⇨Guardar

- Imprima los contenidos del proyecto:
  - Clic en en la barra de herramientas "Proyecto" del IDE
    ⇨ Seleccionar fuente ⇨OK

- Editar y compilar un texto fuente propio en ensamblador:

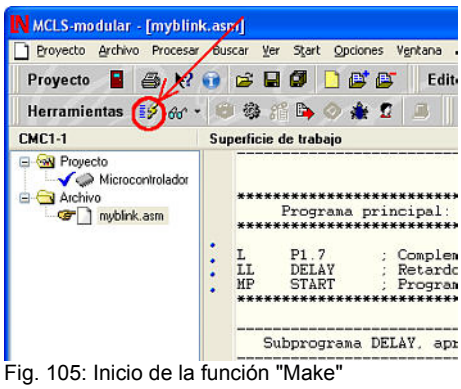


Fig. 105: Inicio de la función "Make"

Al abrir el archivo fuente principal, p.ej. el archivo myblink.asm, puede editarse la secuencia de programa en la superficie de trabajo. Al finalizar con la edición del texto fuente, éste tiene que ser traducido. Para ello haga clic en el botón *Make*.
Una vez terminado el proceso de traducción, podrá mostrarse en el navegador la carpeta Mensajes con los archivos Assembler, Converter, Symbol donde aparecen las informaciones pertinentes al proceso.
Después de eliminar los errores y reiniciar la función "Make", es creado un archivo ejecutable *.hex.

*NOTA: Cerciórese de que en la carpeta de trabajo donde se encuentra guardado el archivo MYBLINK.asm también esté el archivo **c515c.inc**. Si el archivo no se encuentra allí, cópielo de la carpeta donde se encuentra el ensamblador.*

- Pruebe el programa en el MCLS-*modular:*
  - ↳ Clic en el símbolo de la barra "Herramientas" del IDE (o menú Start ⇨Make)
  - ↳ Clic en el símbolo de la barra "Herramientas" del IDE (o menú Start ⇨Herramientas ⇨Depurador)

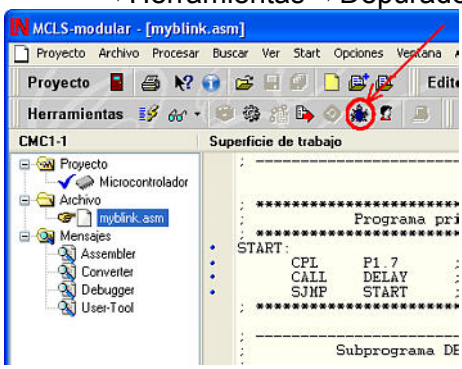


Fig. 106: Inicio del depurador

Durante el proceso de traducción, la herramienta Make crea con el ensamblador y el convertidor un archivo hex que contiene el código de programa ejecutable y que puede ser cargado en la RAM externa del módulo PSD1-Flash. Para ello, hay que iniciar el programa depurador haciendo clic en el botón .
Al iniciarse el programa, el archivo ejecutable es cargado automáticamente en la RAM del módulo PSD1-Flash.

  - ↳ Clic en “OK“ (o tecla ↵/Intro) (Confirmar mensaje en la ventana de error)

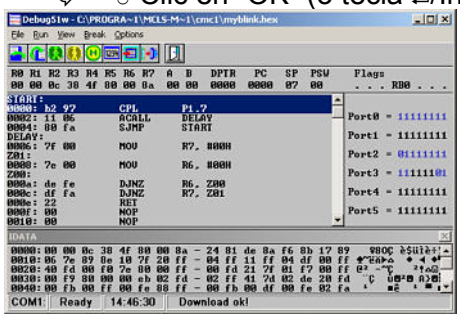


Fig. 107: Depurador del IDE

Ahora puede realizarse la depuración del texto fuente. Con ayuda de diferentes comandos de operación puede realizarse la primera puesta en servicio/prueba de funcionamiento del programa. Los comandos a emplear son:

- Modo de ejecución por pasos (single step) ***F7***
- Modo de ejecución por rutinas (single proc) ***F8***
- Ejecución en tiempo real ***F9***
- Reset ***CTRL + F3***
- Poner/quitar puntos de ruptura ***CTRL-B/ CTRL-E***

  - ↳ Iniciar la ejecución del programa en tiempo real
    ⇨Pulsar la tecla F9
  - ↳ Parar la ejecución en tiempo real
    ⇨Pulsar el botón RESET en la unidad adaptadora de 8 bits

- Cerrar el depurador
  - ↳ Hacer clic en el cuadro de cierre de la ventana

- Cierre el proyecto:
  - ↳ Menú Proyecto ⇨Cerrar proyecto

- Salga del IDE:
  Clic en en la barra de herramientas "Proyecto" del IDE

NOTA: De encontrarse algún otro proyecto abierto, éste será guardado primero automáticamente. Los mensajes de error al iniciar el depurador se describen en el ejercicio 3.

Convertir un perfil integrado en un perfil de usuario

Todos los perfiles predefinidos que se encuentran integrados pueden ser adaptados a las necesidades individuales del usuario. Para ello, primero tienen que ser convertidos en un perfil de usuario.

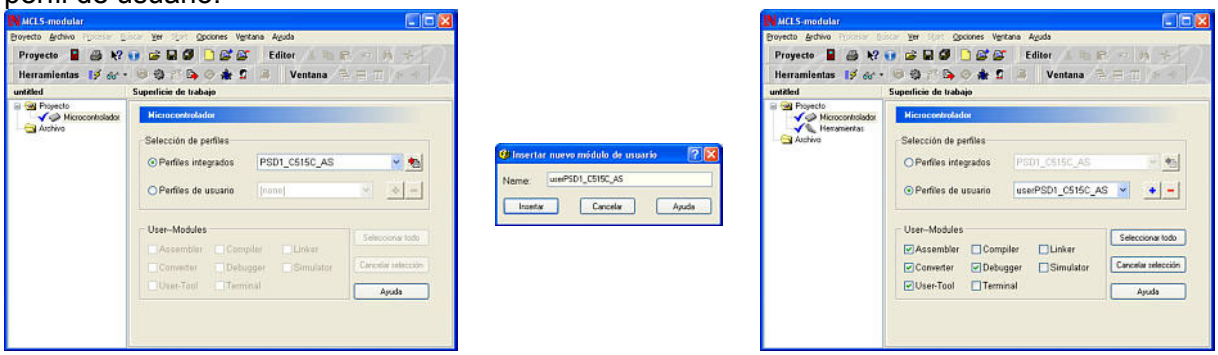


Fig. 108: Creacion de un perfil de usuario

Esto se hace de la siguiente manera:

(1) Ir a Opciones ⇨Proyecto ⇨Microcontrolador y seleccionar el perfil integrado "PSD1_C515C_AS".

(2) Hacer clic en el botón (copiar) que se encuentra a la derecha del cuadro de lista para convertir el perfil en un perfil de usuario.

(3) Darle un nuevo nombre al perfil y aceptar.

(4) Activar el botón "Perfiles de usuario".

(5) Seleccionar el perfil que se acaba de crear.

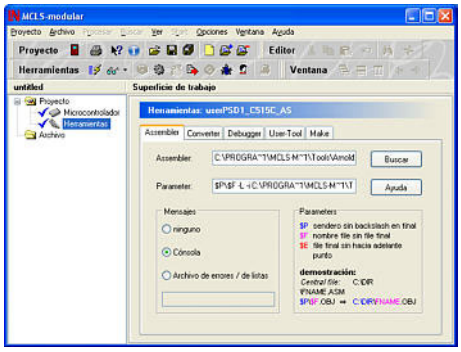


Fig. 109: Diálogo "Herramientas" para la configuración del perfil de usuario

Ahora puede definirse para el perfil de usuario seleccionado las herramientas que desean utilizarse. La configuración de las herramientas individuales se hace, seguidamente, en el cuadro de diálogo que aparece al hacer clic en el árbol de proyecto sobre "Herramientas".

Preguntas :

Indique un procedimiento habitual para cargar un código de programa legible para máquina en un microcontrolador. ¿Qué estructura especial tiene el contenido de este archivo?

¿Qué herramientas de desarrollo son necesarias para generar un código de programa cargable y para qué sirve cada una de ellas ?

¿Qué herramientas (designación de la herramienta, nombre del archivo ejecutable) se utilizan en el perfil integrado „PSD1_C515C_AS“? ¿Cómo se insertan éstas en el IDE y qué se debe observar especialmente?

¿Qué herramientas se indican en la lista impresa „Perfil de herramientas “ (Information of tool profile)?

¿Qué indicaciones exactas de ruta y de archivo encuentra usted detrás de „Project“ y bajo „List of sources“ en la lista impresa „Informaciones de proyecto“ (Information of project) ?

¿Qué LED de la unidad LED parpadea en la ejecución del programa en el microcontrolador?

Ejercicio 2:
Trabajar con el editor de texto, generación de un código de programa cargable

Objetivo :

- Abrir un proyecto existente
- Crear un archivo de texto fuente nuevo
- Editar en un archivo de texto fuente
- Utilización de señales en el editor
- Marcas necesarias, específicas de C515C, para el Ensamblador
- Reglas importantes del Ensamblador en la elaboración del texto fuente
- Ensamblar un archivo de texto fuente
- Ventana de mensajes para el Ensamblador
- Tratamientos de errores después del ensamblado
- Convertir el archivo de código intermedio
- Ventana de mensajes para el Convertidor
- Tratamiento de errores después de la conversión
- Ensayo en el MCLS-*modular*

Ejercicio :

Cargue el proyecto existente „CMC1-1“.
Inserte otro archivo de texto fuente con el nombre de „myblink1.asm“. Copie el contenido entero de „myblink1asm“ a „myblink2.asm“.
Deje „e1blink2“ en el editor de texto. Coloque, en este archivo en el subprograma TIEMPO, una señal y utilícela.

Sustituya en „myblink2.asm“ en la línea de comando „ TIEMPO: MOV R7,#00H“ (subprograma TIEMPO) el 2º parámetro #00H por #4FH. Guarde el archivo. Controle sus registros con respecto a errores ejecutando el Ensamblador.

Convierta el perfil integrado PSD1_C515C_AS en un perfil de usuario.
Realice en „myblink2.asm“ los errores descritos a continuación, ensamble y lea los mensajes del Ensamblador. ¡ Corrija los errores después del ensayo ! (Controlar todos los errores por separado !)

(1) Elimine el parámetro „$P\$F.asm“ de los registros para la inserción de la herramienta de desarrollo „Ensamblador“
(2) Elimine el parámetro „$P\$F.p“ de los registros para la inserción de la herramienta de desarrollo „Convertidor “
(3) Borre el registro „CPU 80515“
(4) Borre el registro „include c515c.inc“
(5) Borre todos los espacios en blanco delante de „CPL P1.7“
(6) Borre „TIEMPO:“ en el „subprograma TIEMPO “ (¡deje como mínimo 1 espacio delante de „MOV R7,#4FH“!)
(7) Sustituya en la línea de comando „ TIEMPO: MOV R7,#4FH“ (subprograma TIEMPO) el 2º parámetro #4FH por #FH.
(8) Sustituya en la línea de comando „ CPL P1.7“ el punto y coma delante del comentario por una coma

(9) Vuelva a cometer el error 5 e inicie el convertidor inmediatamente después del ensamblado. Lea el mensaje del convertidor. Corrija el error.

Compruebe el programa con respecto a la ausencia de errores. Genere de los archivos „myblink2.asm“ y „myblink1.asm“ el archivo INTEL-HEX. Ensaye ambos programas en el MCLS-modular.

Forma de proceder :

- Inicie desde el menú prinicpal de el entorno de desarrollo integrado para el MCLS-modular.

- Abra un proyecto existente.

  ↳ Click de sobre el símbolo en la barra de herramientas „Proyecto “ del IDE ⇨Seleccionar archivo ⇨Abrir (o Menú Proyecto ⇨ Insertar proyecto ⇨ Seleccionar archivo ⇨Abrir)

NOTA : Si ya está abierto un proyecto, éste es archivado automáticamente y cerrado a continuación.

- Crée un nuevo archivo de texto fuente.

  ↳ Menú Archivo ⇨ Archivo nuevo

ATENCIÓN : el archivo nuevo todavía no tiene nombre; guárdelo con un nombre apropiado.

- Edite un nuevo archivo de texto fuente.

  Seleccionar
  ↳ Colocar el cursor de a la izquierda del primer signo del texto a seleccionar ⇨Pulsar la tecla izda del y arrastrar el cursor del sobre el texto a seleccionar ⇨Soltar la tecla izquierda del
  Seleccionar todo
  ↳ Tecla derecha del ⇨Seleccionar todo

  Copiar
  ↳ Seleccionar texto a copiar ⇨Tecla derecha del ⇨Copiar
  Cortar
  ↳ Seleccionar texto a cortar ⇨Tecla derecha del ⇨Cortar
  Pegar
  ↳ Copiar o cortar texto a pegar ⇨ Colocar cursor del a la posición de pegado ⇨Tecla derecha del ⇨Pegar
  Deshacer Eliminar
  ↳ Tecla derecha del ⇨Deshacer Eliminar

O por medio de los siguientes

Procesar menú	Símbolos de la barra	Por medio de teclas (Short key)
Copiar		CTRL+Z
Cortar		CTRL+X
Pegar		CTRL+C
Deshacer Eliminar		CTRL+V
Seleccionar todo		CTRL+A

- Colocar una señal

  ↳ Colocar el cursor del 🖱 en el borde izquierdo de la línea en la que se debe colocar la señal (el cursor del 🖱 llega a ser verde cuando está en el borde) ⇨Tecla derecha del 🖱 ⇨Poner señal ... ⇨Introducir nombre de señal deseado ⇨OK

- Eliminar una señal

  ↳ Posicionar el cursor del 🖱 sobre el borde izquierdo de la línea en la cual está colocada la señal (el cursor del 🖱 llega a ser verde cuando está en el borde) ⇨Tecla derecha del 🖱 ⇨Borrar señal

- Hacer saltar el cursor de texto a una señal

  ↳ Click de 🖱 sobre el símbolo  en la barra de herramientas „Editor“ del IDE (o Menú Ver ⇨Navegador) ⇨Seleccionar el archivo deseado por medio de la oreja inferior del fichero ⇨ Click de 🖱 en la señal deseada

- Tenga cuidado de que estén introducidos los siguientes textos específicos del procesador

  ↳ CPU 80515
  ↳ Include c515c.inc

Observe en la introducción de estos textos las reglas relativas al Ensamblador de A.Arnold (ver anexo).
Si falta el registro „CPU 80515“, el Ensamblador no reconoce ningún comando y no sabe qué lista de registro de funciones especiales específica del procesador debe utilizar. ¡ El Ensamblador cancela el proceso !
Si falta el registro „Include c515c.inc“, ¡ el Ensamblador no reconoce los símbolos definidos de los registros de funciones especiales !
NOTA : Si falta el registro „Include bitfuncs“, el Ensamblador no puede dividir un valor de 16 bits en parte Alta (High) y Baja (Low). Si esta función no se necesita, se puede suprimir el registro.

- Ensamblar un archivo de texto fuente

  ↳ Click de 🖱 sobre el símbolo  en la barra de herramientas „Herramientas “ del IDE (o Menú Inicio⇨Herramientas⇨Ensamblador)

ATENCIÓN : Si el Ensamblador ha detectado errores ¡ no se genera un código intermedio (.p-File) !*

- Compruebe los mensajes del Ensamblador
  ↳ Tras cada ejecución de Ensamblador ya se encuentran en la ventana de mensajes
  (↳ O click de 🖱 en el Ensamblador en la rama de proyecto Mensajes)

  ⇨ Ventana de mensajes superior
  Informaciones generales como archivos empleados (*.inc, *.asm), número de líneas del texto fuente completo, número de errores, número de mensajes, así como algunas informaciones de memoria

  ⇨ Ventana de mensajes inferior
  Especificacines de errores con la siguiente estructura :

  „Nombre de archivo“.asm (número): error/aviso: indicación de error >>> Especificación de texto

„Nombre archivo “.asm	Archivo en el que se encuentra el error
(número)	Nº de la línea en la que está el error
Error/Aviso	¿Se trata de un error o de un aviso?
Indicación de error	¿De qué error se trata?
Especificación de texto	¿Qué texto causa el error ?

ATENCIÓN : Si en la inserción del Ensamblador para las herramientas de proyecto no se ha introducido el parámetro „ -x “, no se edita ninguna especificación de texto.

NOTA : Si el inicio de Ensamblador ha fracasado p.e. debido a una falta de parámetros, aparece una información general relativa al Ensamblador en la ventana de mensajes superior.

- Evalúe los errores comunicados por el Ensamblador y elimínelos
  ↳ Tras cada ejecución del Ensamblador, ya se encuentran en la ventana de mensajes
  (↳ O click de 🖱 sobre el Ensamblador en la rama de proyecto Mensajes)
  ↳ Leer los mensajes en la ventana inferior, especialmente indicaciones a errores y especificaciones de texto
  ↳ Salto a la línea con error en el texto fuente
  ⇨Doble click de 🖱 sobre una especificación de error
  ↳ Corregir texto fuente

- Convierta el archivo de código intermedio
  (*.p-File) en un código de programa cargable (archivo INTEL-HEX)

  ↳ Click de 🖱 sobre el símbolo en la barra de herramientas „Herramientas “ del IDE
  (o Menú Inicio ⇨Herramientas ⇨Convertidor)

ATENCIÓN : Si el Ensamblador no ha generado un código intermedio (.p-File) p.e. debido a errores, ¡se produce un mensaje del Convertidor y no se genera un archivo INTEL-HEX-File !*

- Compruebe los mensajes del Convertidor

  ↳ Tras cada ejecución del Convertidor ya se encuentran en la ventana de mensajes
  (↳ O click de 🖱 sobre el convertidor en la rama de proyecto Mensajes)

  Mensajes : Ruta y nombre del archivo intermedio y del archivo Intel-Hex generado, tamaño del archivo generado, eventual-mente indicación de errores.

- Evalúe los mensajes comunicados por el Convertidor y elimine las causas

  ↳ Tras cada ejecución del Convertidor ya se encuentran en la ventana de mensajes
  (↳ O click de 🖱 sobre el Convertidor en la rama de proyecto Mensajes)
  ↳ Los mensajes se refieren casi siempre a la falta de un archivo de código intermedio
  ↳ Elimine la causa (p.e. eliminiación de error en el texto fuente, a continuación en-samblar y convertir de nuevo)

  NOTA : Si el inicio del convertidor ha fracasado p.e. debido a la falta de parámetros, se visualiza una información general relativa al Convertidor en la ventana de mensajes.

- Genere en una operación de un texto fuente un código de programa cargable (archivo INTEL-HEX)

  ↳ Click de 🖱 sobre el símbolo ⚡ en la barra de herramientas „Herramientas “ del IDE (o Menú Inicio ⇨ Make)

NOTA : En la rama de proyecto „Herramientas“ deben estar correctamente introducidas en la ficha „Make“ las herramientas necesarias Ensamblador y Convertidor. (ver ejercicio 1)

- Ensaye el programa en el MCLS-*modular*

  ↳ Click de 🖱 en el símbolo 🐞 en la barra de herramientas „Herramientas “ del IDE (o Menú Inicio ⇨Herramientas ⇨Depurador)
  ↳ Click de 🖱 “Ok“ (o tecla ↵/Intro)
  (confirmar mensaje en ventana de errores)
  ↳ Iniciar el programa en servicio de tiempo real
  ⇨Pulsar la tecla F9
  ↳ Parar el servicio de tiempo real
  ⇨Pulsar la tecla RESET en la unidad de adaptador de 8 bits
  ↳ Finalizar Depurador
  ⇨Hacer clic en el cuadro de cierre de la ventana

NOTA : El programa ejecutable del Depurador y sus parámetros deben estar introducidos según las instrucciones (ver el ejercicio 1). (¡Los posibles errores se tratarán solamente en el ejercicio 3!)

Preguntas :

¿Por medio de qué símbolo se representa una señal existente ?

¿Qué sucede si en el Navegador del Editor se realiza un click de ratón sobre un registro de la tabla ?

¿Qué número se indica entre paréntesis detrás del nombre de archivo en la ventana de mensajes inferior del Ensamblador ?

¿Cómo llego lo más rápidamente posible de la ventana de mensajes del Ensamblador a una línea con error del texto fuente?

¿Por medio de qué herramienta se genera del código intermedio el archivo INTEL-HEX ?

¿Qué diferencia hay entre „myblink1“ y „myblink2“ en la ejecución de los programas en el microcontrolador ?

¿Qué especificación de error se indicó respectivamente para los errores ③ - ⑧ ? Explique las causas de todos los errores (① - ⑨) . Introduzca los errores en la siguiente tabla.

Se indicaron las siguientes especificaciones de errores / Causas de todos los errores

Nº de error	Nombre de archivo ?	Error / Aviso (E/A)	Indicación relativa al error	Especific. de texto	Causa
1	se suprime				
2	se suprime				
3	C515C.inc	E		-	
4	myblink2.asm	E		P1.7	
5	myblink2.asm	E		P1.7	
6	myblink2.asm	E		TIEMPO	
7	myblink2.asm	E		FH	
8	myblink2.asm	E		-	
9	se suprime	A		se supr.	

Ejercicio 3:
Trabajar con el Depurador

Objetivo :

- Iniciar el Depurador
- Conocer los posibles mensajes en el inicio
- Configurar el puerto COM
- Cargar un archivo INTEL-HEX
- Servicio por pasos de proceso
- Servicio por pasos individuales
- Ejecución en tiempo real hasta una dirección de programa definida
- Ejecución en tiempo real contínua en el microcontrolador
- Reposicionar el contador de programa (RESET)
- Poner/borrar puntos de interrupción
- Ver espacios y contenidos de memoria
- Modificar los contenidos de memoria y de registro, así como los contenidos del Port-Latch
- Cerrar el Depurador

Ejercicio :

Cargue el proyecto existente „CMC1-1“. Añada otro archivo de texto fuente existente con el nombre „cmc112.asm“. Guarde este archivo con el nombre „myled.asm“.
Genere de ello el código de programa cargable. Inicie el Depurador. Introduzca el puerto COM empleado para el MCLS-modular, archive las opciones.
Ponga el interruptor S1 hacia abajo.

Ejecute el servicio por pasos de proceso hasta haber ejecutado varias veces el bucle de programa. Compruebe los efectos en el MCLS-modular.
Ponga el contador de programa a 0000H (RESET). Ejecute el servicio por pasos individuales hasta que el contador de programa se encuentre en la dirección de comando 0013H. Ejecute varias veces el comando en esta dirección. Observe el contenido del registro R6 en la barra de registros. Salga del bucle de tiempo mediante une ejecución de programa hasta la dirección 0017H.
Ponga un punto de interrupción en la dirección 0004H. Inicie la ejecución en tiempo real del microcontrolador. Elimine el punto de interrupción y vuelva a iniciar la ejecución en tiempo real. Observe los efectos en el MCLS-modular. Finalice el servicio de tiempo real (RESET).
Simule una señal de entrada en P4.7 sustituyendo el contenido del Port-Latch de P4 (11111111) por 01111111. Inicie el servicio de tiempo real. Sustituya el contenido de la memoria de código en la dirección 0001H (FD) por F8. Compruebe los efectos en la ejecución de programa.
Compruebe la influencia del interruptor S1 en la indicación. Cierre el Depurador.

Forma de proceder :

- Inicie desde el menú prinipal de el entorno de desarrollo integrado para el MCLS-modular.
- Abra el proyecto „CMC1-1“.
-
- Inserte el archivo „cmc112.asm“ en el proyecto.

- Guarde este archivo como "myled.asm" y declárelo como archivo principal (mainfile).

- Genere de ello el archivo INTEL-HEX cargable. (Ensamblar + convertir)

- Inicie el Depurador
  ↳ Click de sobre el símbolo en la barra de herramientas „Herramientas “ del IDE (o Menú Inicio ⇨ Herramientas ⇨Depurador)

- Reaccione con respecto a los mensajes
  ↳ del sistema operativo
  „El sistema no puede abrir la conexión COM1(2) “ (o similar)

  Causa : El puerto COM está siendo usado para el
  ⇨ Click de sobre „Ignorar“

  ↳ del Depurador
  „Serial device error : please RESET the microcontroller system !“

  Causa 1: El puerto COM que se ha configurado en el depurador no coincide con el puerto COM del MCLS-*modular*.

  ⇨ Clic en "OK" (o tecla ↵/Intro) ⇨ Configurar depurador con puerto COM del MCLS-*modular.*

  Causa 2: El microcontrolador aún se encuentra en servicio de tiempo real.
  ⇨ Accionar la tecla RESET del módulo de microcontrolador

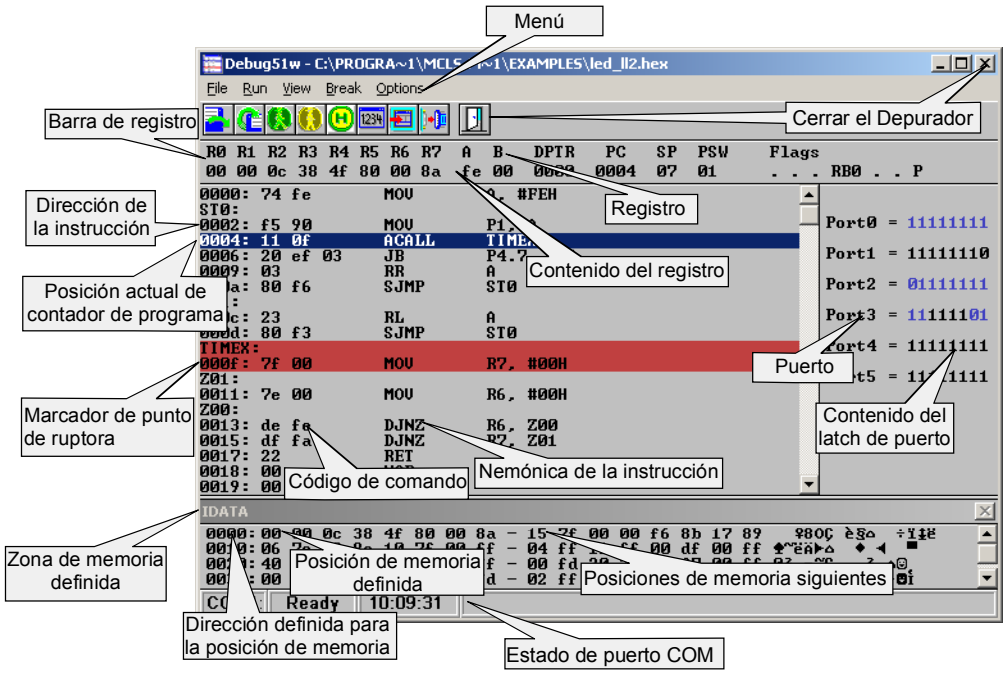


Fig.: 102: Ventana del depurador

- Configure el puerto COM
  - ↳ Menú Opciones ⇨Depurador ⇨Seleccionar puerto COM ⇨ OK
  - ↳ Menú Opciones ⇨Guardar opciones

- Cargue un archivo INTEL-HEX
  - ↳ Después del inicio del Depurador se carga automáticamente el archivo INTEL-HEX del archivo principal
    (↳ O Menú Archivo ⇨ Cargar archivo Intel Hex ⇨Seleccionar archivo ⇨Abrir

- Ejecute el servicio por pasos de proceso
  - ↳ Tecla F8 (O Menú Run ⇨Single proc)

NOTA : El subprograma que se llama con el comando indicado por el contador de programa, se ejecuta en tiempo real. Si el comando marcado no es una llamada de subprograma, se ejecuta un paso individual.

- Reposicione el contador de programa en la dirección 0000H (RESET)
  - ↳ Tecla CTRL+F3 (O Menú Run ⇨Program reset)
  - ↳ Para servicio de tiempo real Accionar la tecla de RESET en la unidad de adaptador de 8 bits

- Ejecute el servicio por pasos individuales
  - ↳ Tecla F7 (O Menú Run ⇨Single step)

NOTA : Se ejecuta el programa del comando indicado por el contador de programa. Para salir de bucles de tiempo, utilice el comando „Goto address“.

- Ejecute el programa hasta una dirección definida (Goto address)
  - Tecla F4 (O Menú Run ⇨Goto address) ⇨Introducir dirección ⇨OK

NOTA : El programa se ejecuta en tiempo real a partir de la posición actual del contador de programa hasta la dirección indicada.

- Ponga un punto de interrupción
  - Tecla CTRL+B (O Menú Break ⇨Set breakpoint) ⇨Edit ⇨Introducir dirección como número Hex ⇨ Accionar la tecla ↵/Intro ⇨OK

- Borre un punto de interrupción
  - Tecla CTRL+B (O Menú Break ⇨Set breakpoint) ⇨Delete ⇨Seleccionar dirección
  - ⇨OK

- Ejecute el servicio de tiempo real continuo (RUN)
  - Tecla F9 (O Menú Run ⇨Run program)

NOTA : Se ejecuta el programa total a partir de la posición de contador de programa actual. Si están insertados puntos de interrupción, se para en estos puntos.
Al alcanzar un punto de interrupción ¡no es posible la continuación inmediata del programa en tiempo real! El ensayo de programa se debe continuar en servicio por pasos individuales.
Al alcanzar una dirección de PC que se encuentre como mínimo 3 direcciones por encima de la dirección del punto de interrupción, se puede continuar con la ejecución de programa en tiempo real.

- Visualizar un espacio / contenido de memoria
  - Seleccionar espacio de memoria
    ⇨Menú View ⇨Seleccionar espacio de memoria (IDATA, DDATA, XDATA, CODE)
    La indicación se realiza a partir de la dirección 0000H.
  - Seleccionar la dirección de la localización de memoria ⇨Menú View ⇨View address ⇨Introducir dirección ⇨OK

NOTA : La indicación se realiza a partir de la dirección seleccionada.

- Modifique los contenidos de memoria y de registro, así como los contenidos del Port-Latch
  - Hacer indicar localización deseada
    ⇨ Click de sobre la localización, registro o puerto ⇨Introducir modificación

NOTA : La introducción es válida de inmediato.

- Cierre el Depurador
  - Hacer clic en el cuadro de cierre de la ventana (o menú File ⇨EXIT)

NOTA :En la ejecución en tiempo real no son posibles introducciones por medio del teclado o del ratón.

Preguntas :

¿Cómo se puede configurar el puerto COM ?

¿Qué diferencia decisiva hay entre el servicio por pasos individuales y el servicio por pasos de proceso y qué tienen en común?

¿Qué efecto tiene un punto de interrupción en la realización de una ejecución en tiempo real? ¿Qué se debe tener en cuenta al iniciar la ejecución en tiempo real desde un punto de interrupción?

¿Cómo se accede al contenido de la localización 87H en la memoria de datos interna indirectamente direccionable (IDATA) ? ¡Explique el correspondiente procedimiento !

¿Cómo se puede modificar el contenido de una posición de memoria ?

¿Con qué combinación de teclas se puede salir del Depurador?

CMC 1-2 El microcontrolador C515C

Estructuras de memoria, espacios para direcciones, tipos de direccionamiento, componentes on-chip, lista de comandos

Fundamentos preliminares

Los microcontroladores son sistemas de microordenador completos en un chip. CPU, memoria, componentes de periferia y sistema de interrupción están integrados conjuntamente en un chip y conectados entre ellos por medio de uno ó varios sistemas de bus.

Infineon Technologies AG ha creado con el C515C un poderoso derivado de la familia 8051 con amplia gama de periféricos on-chip, cuya CPU es prácticamente equivalente a la del 8051 de INTEL. La política liberal de otorgamiento de licencias de INTEL ha hecho que a partir de 1981 muchos fabricantes de chips hayan desarrollado derivados propios basados en la CPU del 8051, de manera que ahora existe una amplia gama de variantes haciéndose muy conocida la familia 8051.
Las características principales del C515C son:

- CPU de 8 bits con amplio juego de instrucciones
- Procesador booleano
- 256 bytes de RAM interna
- 2 KB de XRAM interna
- Hasta 64 KB de memoria de datos externa
- 8 punteros de datos
- Bus externo para el acoplamiento externo de la memoria de datos y la memoria de programa
- Tiempo de ciclo de 1 µs para las instrucciones con una frecuencia de 6 MHz
- Compatible con procesadores más antiguos, hasta el C515

En comparación con derivados más antiguos, el C515C-L(M) no sólo cuenta con un número mayor de componentes periféricos on-chip, tales como convertidor A/D, temporizador con funciones de comparación y de captura, diversos modos de ahorro de energía e interfaz de comunicación, sino que se ha prescindido expresamente de la memoria de programa onchip (ROM), de manera que ésta debe ser conectada externamente.
Las partes esenciales del CPU son el mecanismo de control con el descodificador de comandos, la unidad aritmética lógica, el procesador de Bool y algunos registros o registros con función de indicador.
Además de la memoria de datos y de programa externa hay una memoria de datos interna en el chip. Físicamente separados entre ellos hay 256 Byte RAM (direcciones 00H-FFH) y 128 Byte para registros de funciones especiales (dirección 80H-FFH).
La memoria de programa y de datos utilizan un bus de dirección y de datos común. Para el direccionamiento de las memorias se dispone de 5 diferentes tipos de direccionamiento :

- Direccionamiento inmediato
- Direccionamiento directo
- Direccionamiento indirecto (empleo de registros indicadores)
- Direccionamiento indirecto por medio de registros de base y de índice
- Direccionamiento de registros

Los espacios de la memoria de datos interna igualmente direccionados pero físicamente diferentes se alcanzan mediante el direccionamiento directo o indirecto. El tipo de direccionamiento decide qué memoria física se activa.

Todos los derivados de la familia 8051 tienen el mismo conjunto de comandos. Un comando completo se compone de un primer byte de código de comando y, según comando, de otros 1 a 2 bytes que especifican los operandos. El conjunto de comandos contiene comandos de transporte, enlaces aritméticos y lógicos, comandos de manipulación de bits, comandos de salto, así como comandos de disposición, de borrado y de desplazamientos.
Todos los comandos de los programas cargados en el microcontrolador por el archivo INTEL-HEX se archivan byte por byte en direcciones de la memoria de programa. A un comando pueden corresponder varios bytes de comando. Después de un cambio de nivel definido de Low a High en el pin de Reset del controlador, los comandos son procesados sucesivamente por el CPU a partir de la dirección 0000H. El CPU es controlado por la cadencia de oscilador.
Para un ciclo de comando (leer el comando, descodificar, ejecutar y procesar datos) se necesitan uno ó varios ciclos de máquina (CM). Un ciclo de máquina comprende para el C515C 6 cadencias. La cadencia resulta de 1/Frecuencia de cuarzo del cuarzo de oscilador. En el módulo PSD1-FLASH se encuentra un cuarzo de 6 MHz para el oscilador. O sea, el tiempo para un ciclo de máquina es de 1µs.
A diferencia de ello, la periferia „on-chip“ trabaja totalmente independiente del CPU. Sus componentes se programan por medio de registros de funciones especiales integrados en la memoria de datos. Por medio de éstos también se realizan las entregas de datos de los componentes. Algunos utilizan la cadencia de oscilador.

Véanse más detalles relativos al C515C en el capítulo 6 del manual de instrucciones del módulo PSD1-FLASH y de la información compacta.

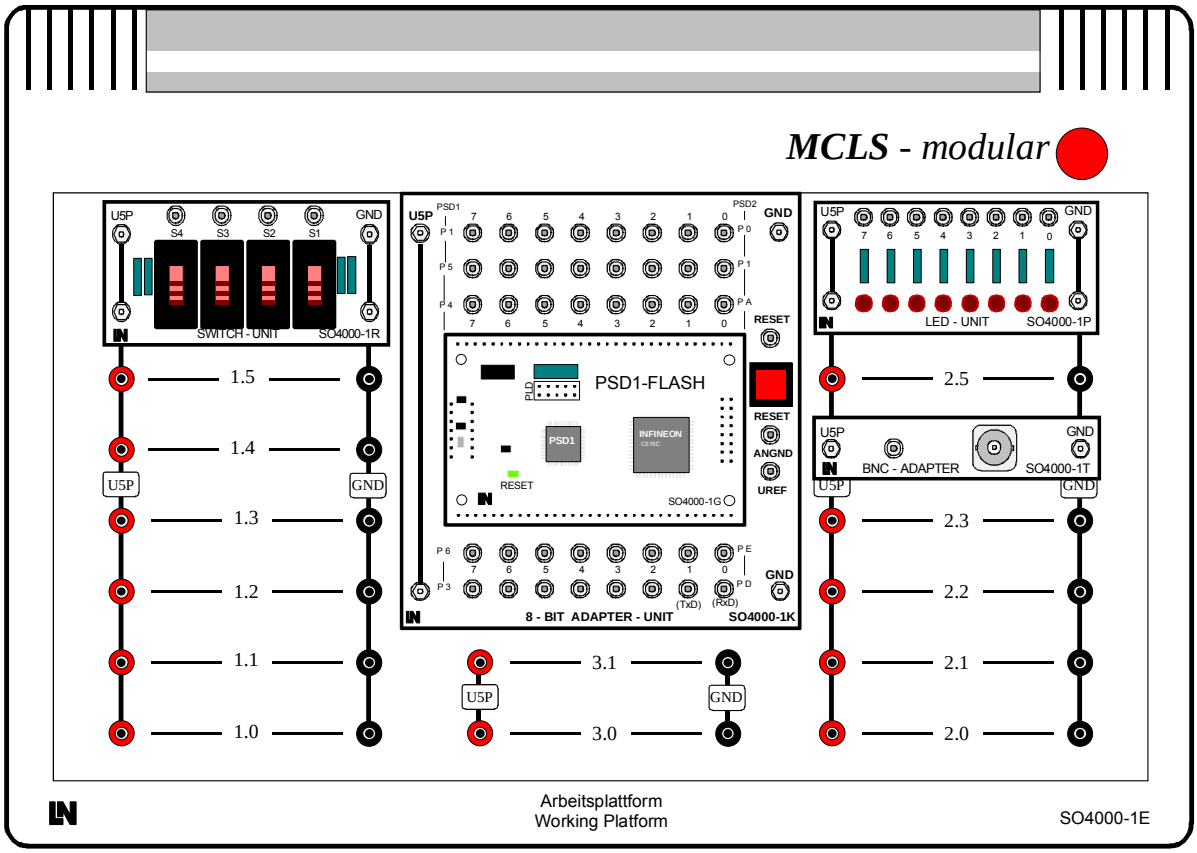


Fig.: 201: Instalación de aparatos del ensayo CMC 1-2

Instalación experimental CMC 1-2

Aparatos y accesorios necesarios

Unid.	Designación	Nº Id
1	Plataforma con mód. de alim. +5V	SO4000-1E
1	Unidad de alim. de enchufe AC 90...230V 45..65Hz, DC 9V 630mA	SO4000-1F
1	Mód. PSD1-FLASH Controlador C515C	SO4000-1G
1	Unidad de adaptador de 8 bit	SO4000-1K
1	Unidad de LED	SO4000-1P
1	Unidad de interruptores	SO4000-1R
1	Adaptador BNC	SO4000-1T
1	Osciloscopio	LM6203
1	Software IDE para MCLS (D,GB,F,E)	SO4001-9H
1	Cable de interfaz en serie 9/9 pol.	LM9040
1	Cable de medición BNC / BNC	LM9034
8	Línea de medición 2mm 15cm azul	SO5126-5K
8	Línea de medición 2mm 15cm amarillo	SO5126-5M
1	CMC 1 Introducción a la programación de microcontroladores 8051	SH5019-1A
1	Manual de instrucciones Mód. PSD1-FLASH	BD4000-1G

Instalación de aparatos

¡Separe la alimentación de corriente del MCLS-*modular* !
Conecte la plataforma de trabajo por medio del cable de interfaz en serie a su PC.
Encaje la unidad de adaptador de 8 bits con el módulo PSD1-FLASH adaptado sobre ella en el campo de casquillos 4.0 - 4.5 (CC) en el centro de la plataforma. ¡Tenga cuidado con las clavijas de contacto en el centro de la unidad de adaptador (TxD, RxD)! Encaje la unidad de LED a la derecha de la unidad de adaptador de 8 bits (p.e. CC 2.6 – 2.7). Posicione la unidad de interruptores en CC 1.6 – 1.7 y el adaptador BNC en CC 2.4 en la plataforma de trabajo. Conecte el osciloscopio a P1.3 (¡masa a GND !). Conecte la sonda de prueba lógica a la tensión de servicio (sondas U5P y GND).
Conecte la alimentación de corriente al MCLS-modular.

Indicación relativa a la seguridad:
¡ Asegúrese de conectar correctamente los potenciales de tensión (izda. U5P, dcha. GND) ! Los puntos de medición llevados hacia fuera no deben ser conectados a tensiones extrañas, ya que esto podría dar lugar a la destrucción de los componentes en las placas !

Indicación relativa a la literatura

Recurra para la realización del ensayo al manual del entorno de desarrollo integrado (IDE) y al manual de instrucciones de la unidad PSD1-FLASH. Lea la información compacta en el anexo del manual de instrucciones del módulo PSD1-FLASH.

Ejercicio 1:
Periferia „on-chip“

Objetivo :

- Conocer componentes de periferia importantes
- Reset de hardware
- Relación entre cadencia de oscilador y tiempos de ejecución de programa
- Relación entre componentes de periferia „on-chip“ y CPU

Ejercicio :

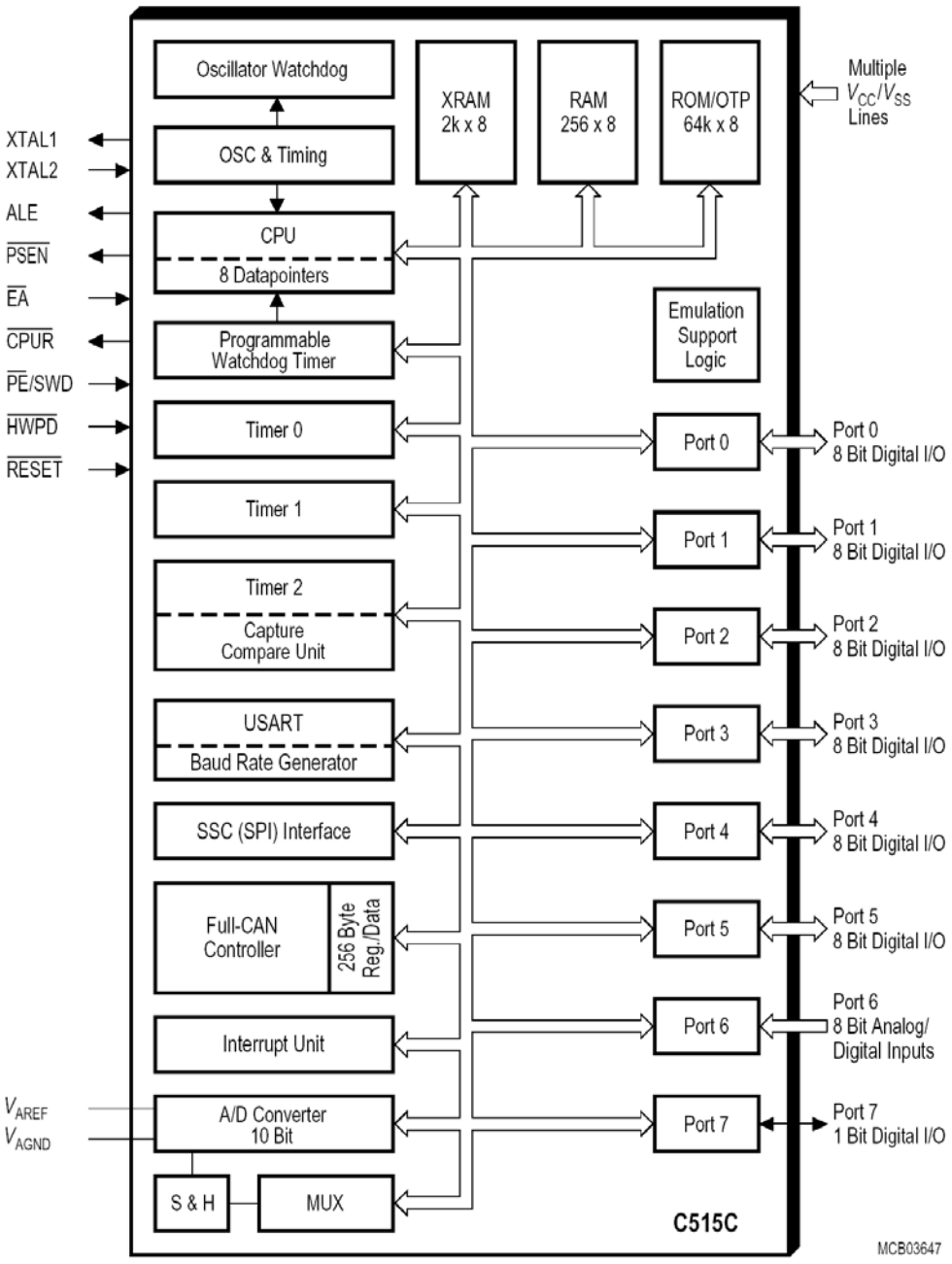
Elabore el proyecto „CMC1-2“ con el archivo „cmc121.asm“.

Calcule el tiempo para 2 ejecuciones de programa (programa principal). Controle el resultado midiendo con el osciloscopio en el puerto 1.3 la duración de periodo de una oscilación durante la ejecución en tiempo real.

Inserte el archivo “cmc122.asm” en el proyecto (este programa hace parpadear un LED e inicia al temporizador 2 como generador de frecuencia de anchura de pulso modulada para el puerto 1.3). Ensaye los efectos que tienen los programas principales largos en la duración de periodo de la oscilación generada en P1.3.
Ensaye durante una ejecución en tiempo real qué nivel en la entrada/salida “Reset” de la unidad de adaptación de 8 bits da lugar a un RESET .

Conocimiento básico :

- Arquitectura de controlador - Diagrama de bloque



Además de la periferia representada en el diagrama de bloques, el chip está dotado de un control de interrupción complejo.

- Mediante un RESET de hardware, el contador de programa (Programmcounter – PC) se pone a la dirección 0000H.
- 1 ciclo de máquina (CM) dura 6 cadencias, un comando puede requerir varios CM

Procedimiento :

- Inicie el IDE
- Elabore un proyecto nuevo „CMC 1-2“
  - Cargue el proyecto „CMC 1-1“ y asígnele un nuevo nombre, crée un archivo nuevo „cmc121.asm“ , copie todos los datos necesarios para el Ensamblador en este archivo, elimine todos los demás archivos
  - Introduzca el siguiente programa principal :

```
Start:  cpl     p1.3    ;cambia de nivel en la salida Puerto 1.3
        sjmp    Start   ;empieza el programa desde el ;principio

        end             ;Fin del programa
```

  - Calcule el tiempo de ejecución del programa
    - Determinar la cadencia de oscilador
      T_O = 1/Frecuencia de cuarzo
    - Determinar el tiempo de ciclo de la máquina
      $T_{CM} = T_O * 6$ (número cadencias por CM)
    - Determinar el número de CM por comando
      Lista de comandos
    - Determinar el tiempo de ejecución de comando
      $T_{Ej} = T_{CM} *$ Número de CM
    - Sumar todos los tiempos de ejecución de comando
      $T_{TE} = \Sigma T_{Bef}$
  - Elabore el archivo Intel-Hex y cárguelo en el controlador. Inicie la ejecución en tiempo real, mida la duración de periodo de la oscilación en el puerto 1.3

- Inserte el archivo „cmc122.asm“ en el proyecto
  - Crée para ello un archivo Intel-Hex y cárguelo en el controlador, mida la duración de periodo en el puerto 1.3
  - Realice los pasos de proceso, hasta haber ejecutado varias veces el programa principal ⇨Observe los efectos en los resultado de medición

    NOTA : Una vez iniciados los componentes de periferia, éstos trabajan independiente-mente del CPU y con ello independiente-mente de la ejecución de programa ulterior.

- Ensaye el RESET de hardware
  - El controlador se encuentra en servicio de tiempo real
  - Coloque la punta de la sonda de prueba lógica en el casquillo “RESET” de la unidad de adaptador de 8 bits
  - Accione la tecla de RESET en la unidad de adaptador de 8 bits

Preguntas :

¿Qué periferia „on chip“ posee el C515C? ¡Indique como mínimo 4 componentes diferentes!

¿Qué tiempo ha calculado para 2 ejecuciones del programa principal? ¡Comente su resultado!

¿Qué tiempo de periodo ha medido respectivamente para el programa „cmc121.asm“ y „cmc122.asm“ en el puerto 1.3?

¿Qué efecto tienen en el programa „cmc121.asm“ otros comandos dentro del programa principal en la duración de periodo de la oscilación en el puerto 1.3 ?

¿Qué efecto tiene el programa de parpadeo más lento en la salida controlada por temporizador de puerto 1.3 (programa „cmc122.asm“)? ¡Comente su respuesta!

¿Qué solución de programa se debe emplear especialmente en aplicaciones críticas de tiempo (p.e. como generador de frecuencia)? ¿Qué periferia „on-chip“ utiliza?

¿Cuánto tiempo dura 1 ciclo de máquina al emplear un cuarzo de 8 MHz ? ¡Comente su resultado!

¿Qué nivel da lugar a un RESET ? ¿En qué dirección se pone el contador de programa ?

Ejercicio 2:
Estructura de memoria de programa y de datos, direccionamiento de memoria

Objetivo :

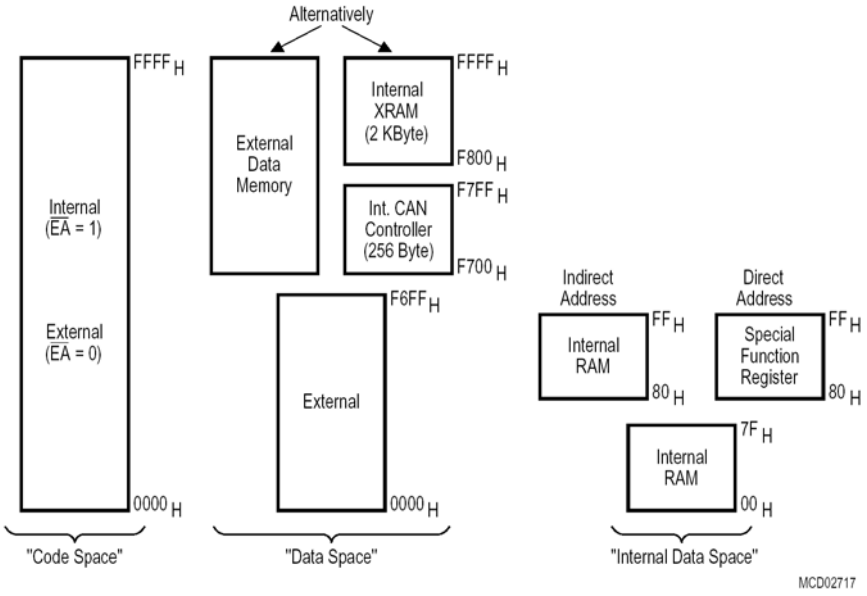
- Conocer la estructura de memoria
- Direccionamiento de las diferentes memorias

Ejercicio :

Lea el capítulo 6, así como los puntos 3 a 5 de la página Arquitectura de controlador en el anexo del manual de instrucciones del módulo PSD1-FLASH.
Ilustre los procesos en el direccionamiento directo e indirecto de localizaciones de memoria por la ejecución por pasos del programa „cmc123.asm“. Contemple, antes y después de cada ejecución de comando, el contenido de la localización de destino.
Compare, después de la ejecución del comando en la dirección 001BH, los contenidos de las direcciones 90H y F8H, respectivamente en la memoria de datos direccionada directa e indirectamente.

Conocimiento básico :

- Espacio para dirección Memoria de programa:
  El módulo PSD1-FLASH posee 32kByte de memoria de programa externa.
- Localizaciones predefinidas en la memoria de programa (direcciones de vector de interrupción)
- Espacio de dirección Memoria de datos:
  El módulo PSD1-FLASH posee 32kByte de memoria de datos externa.



ATENCIÓN : En el módulo PSD1-FLASH, la localización de memoria es utilizada en común por la memoria de programa y la memoria de datos.
¡Tenga en cuenta ésto en la definición de direcciones de memoria !

Tipos de direccionamiento

Direccionamiento = tipo de acceso de escritura o lectura a los datos de los registros, de la memoria de datos y de la memoria de constantes

Tipos importantes de direccionamiento en la familia 8051:

- Direccionamiento de registros
- Direccionamiento directo
- Direccionamiento indirecto
- Direccionamiento inmediato (asignación de valores)
- Direccionamiento indexado indirecto (registros base + registros índice)

Direccionamiento de registros

- Emplea los registros R0 a R7 de la base de registros actual, así como A, B, CY y el puntero de datos DPTR.
- Ejemplos:

```
mov   A,R2
mov   R0,A
```

Direccionamiento directo

- Emplea las posiciones de memoria inferiores de la RAM interna (128 bytes) y el área de registros de funciones especiales SFR.
- Ejemplos:

```
mov   A,60H
mov   P1,5FH
```

Direccionamiento inmediato

- Carga una constante en un registro o en una posición de memoria direccionada.
- **Para identificar las constantes, éstas van antecedidas por el símbolo "#".**
- Ejemplos:

```
mov   A,#12
mov   DPTR,#1000
```

Direccionamiento indirecto

Direccionamiento indirecto de la RAM interna.

- El indicador @Ri representa una posición determinada en la RAM on-chip, cuyo contenido ha de ser leído o modificado.
- Como indicadores de dirección se emplean los registros R0 y R1 (i=0 ó 1) antecedidos por el símbolo "@".
- Ejemplos:

```
mov   A,@R0
mov   @R1,#12
```

b) Direccionamiento indirecto de RAM externa

Instrucciones MOVX de 16 bits:

- El indicador @DPTR muestra una posición determinada de la memoria en el área de direcciones de 64 KB de la RAM externa (también la XRAM del C515C), cuyo contenido ha de ser leído o modificado.
- Como indicador de dirección se emplea el puntero de datos DPTR de 16 bits antecedido por el símbolo "@".
- Sólo existen 2 instrucciones:

```
movx    A,@DPTR    ; acceso de lectura
movx    @DPTR,A    ; acceso de escritura
```

- La parte alta de la dirección es transferida a P2.
- La parte baja de la dirección es multiplexada por división de tiempo y cargada antes de los datos en el puerto P0, el cual está conectado con la memoria externa.

Instrucciones MOVX de 8 bits:

- Para conectar una memoria RAM externa de pequeña capacidad (máx. 256 bytes) con P0, pueden emplearse instrucciones MOVX con el indicador @Ri.
- Sólo existen 4 instrucciones:

```
movx  @R0,A
movx  @R1,A
movx  A,@R0
movx  A,@R0
```

- ¡Tiene poca importancia a efectos prácticos!

NOTA : R0 y R1 solamente pueden direccionar direcciones de 8 bits, y el puntero de datos DPTR direcciones de 16 bits.
Si no se desea un índice, es necesario borrar previamente el acumulador.
Existen diferentes comandos de transporte para los accesos a las memorias de datos internas y externas, así como a la memoria de programa.

Procedimiento :

- Inicie el proyecto „CMC1-2“ e inserte el archivo „cmc123.asm“
- Imprima este archivo fuente
- Cree un archivo Intel-Hex y cárguelo en el controlador
- Ilustre el desarrollo de los comandos por medio del Depurador
  - Ajustar en el Depurador la localización de memoria de destino
  - Visualizar dirección de destino
  - Ejecutar los comandos
  - Comparar el contenido de la dirección de destino con el número a cargar (ver comentario por comando)

NOTA : Compruebe, en caso de desviaciones, la localización de memoria indicada en el Depurador y la dirección de memoria.

Preguntas :

¿Qué memorias de datos internas existen y en qué espacios para direcciones se encuentran éstas?

¿Qué se debe tener en cuenta en la descripción de la dirección 90H en el RAM interno ? ¿Qué sucede si se selecciona un tipo de direcccionamiento erróneo ? ¿Qué componente de periferia se vería afectado? ¿Es detectado este error por el Ensamblador?

¿Qué se debe tener en cuenta en la selección del registro de indicador para la memoria de datos externa?

¿Con qué se debe cargar el registro indicador antes del direccionamiento indirecto?

¿Con qué tipo de direccionamiento se pueden cargar constantes de la memoria de programa?

Ejercicio 3:
Trabajar con la lista de comandos

Objetivo :

- Estructura de la línea de comando
- Abreviaturas de la lista de comandos
- Tipos de comandos

Conocimiento básico :

- Estructura de la línea de comando

  ↳ *Operador Operando final, operando fuente*

  Operador : Código de comando
  Operando final : Registro sin dir. de mem.
  Operando fuente : Registro, dir. de memoria o constante

- Abreviaturas de la lista de comandos

A	-	Acumulador
B	-	Registro B (acumulador aux.)
Rr	-	Registros R0 a R7 del banco de registros actual
Ri	-	Registro R0 o R1 del banco de registros actual (registro indicador
@	-	Símbolo para direccionamiento indirecto (mediante registro indicador)
badr	-	Dirección de bit (memoria de datos interna direccionable, registros de funciones especiales direccionables por bits)
dadr	-	Dirección en la memoria de datos interna directamente direccionable (RAM inferior, R0-R7, registro de funciones especiales)
adr11	-	Dirección de 11 bit dentro de una página de 2 kByte
adr16	-	Dirección de 16 bits
rel	-	Dirección de offset de 8 bit relativa (-128 a +127)
#konst8	-	Constante de 8 bits
#konst16	-	Constante de 16 bits
AC	-	Auxiliary - Carry - Flag
CY	-	Carry - Flag
OV	-	Overflow - Flag
P	-	Parity - Flag

- Tipos de comando

  - **Comandos aritméticos**
    - Adición
    - Sustracción
    - Incremento (+1)
    - Decremento (-1)
    - Multiplicación
    - División
    - Corrección decimal (DA)

  - **Comandos lógicos**
    - Enlace Y
    - Enlace O
    - Enlace O EXCLUSIVO
    - INVERTIR
    - BORRAR (todos los bits =0)

  - **Comandos de transporte**
    - Comandos de carga
    - Comandos de intercambio
    - Salvar memoria de datos directamente direccionable (Stack)
    - Comando de vaciar

  - **Comandos de procesamiento de bits**
    - Carga acumulador de bits (Carry-Flag)
    - Pon bit
    - Borra bit
    - Invierte bit
    - Enlace Y
    - Enlace O

  - **Comandos de desplazamiento**
    - Rotar a la izquierda
    - Rotar a la derecha

  - **Comandos de salto**
    - Saltos incondicionales
    - Saltos condicionales

NOTA : Las condiciones de salto pueden ser estados de flag, contenidos de registro y constantes. Existen saltos condicionales cuya condición de salto resulta del resultado de un decremento o de una comparación (comando combinado).

  - **Comandos de subprograma**
    - Llamada del subprograma (SP)
    - Salto de retorno del SP al programa principal
    - Salto de retorno de la ruta de servicio de interrupción al programa principal

Ejercicio :

Escriba en el archivo “cmc124.asm” y para las siguientes instrucciones las líneas de comando correctas. Ensamble y crée el archivo Intel-Hex. Compruebe por medio del Depurador (servicio por pasos individuales) si los comandos han sido ejecutados correctamente.

- Carga la dirección 30H de la memoria de datos directamente direccionada con la constante 10H, Escriba al principio de esta línea de comando el símbolo “start” (marca de dirección)
- Carga R7 con el contenido de la localización de memoria directamente direccionada 30H
- Carga R6 con la constante 30H
- Carga R0 con la constante 80H, carga la localización de memoria interna de direccionamiento indirecto R0 con la constante 50H
- Carga R6 con la constante 01010101B
- Salta a la marca de dirección “show”
- Realiza un enlace Y del contenido de R6 con la constante 00001111B, Escriba al principio de esta línea de comando el símbolo “mask” (marca de dirección)
- Carga el acumulador con el contenido de R6, Escriba al principio de esta línea de comando el símbolo “show” (marca de dirección)
- Invierte el contenido de acumulador
- Carga el puerto 5 (registro de funciones especiales P5) con el contenido de acumulador
- Llama el subprograma con la marca de dirección “SP1”
- Llama el subprograma con la marca de dirección “SP1”
- Si el bit P5.6 no está puesto, salta a la marca de dirección “mask”
- Borra bit 7 del puerto 5 (direccionamiento de bit P5.7 del registro de funciones especiales)
- Invierte bit P5.7
- Pon bit P5.2
- Carga R6 con la constante 00H
- Carga el acumulador con la constante 01111101B
- Carga el puerto 5 (registro de funciones especiales P5) con el contenido de acumulador, Escriba al principio de esta línea de comando el símbolo “rot” (Marca de dirección)

- Llama el subprograma con la marca de dirección “SP1”
- Llama el subprogrrama con la marca de dirección “SP1”
- Desplaza el contenido del acumulador a la izquierda (sin utilización del Carry)
- Incrementa R6
- Compara el contenido del acumulador con la constante 04H y, si no son iguales, salta a continuación a la marca de dirección “rot” (¡emplear comando de combinación !)
- Borra el contenido del acumulador
- Salta a la marca de dirección “principio”

;----------Subprograma----------

- Salva el contenido de R6 en el Stack, Escriba al principio de esta línea de comando la marca de dirección “SP1”
- Salva el contenido de R7 en el Stack
- Carga R7 con la constante 0FFH
- Carga R6 con la constante 0FFH, Escriba al principio de esta línea de comando la marca de dirección “lo1”
- Decrementa R6 y a continuación, si R6 no es igual a cero, salta a la marca de dirección “lo2” (¡emplear comando de combinación!) Escriba al principio de esta línea de comando el símbolo “lo2” (marca de dirección)
- Decrementa R7 y a continuación, si R7 no es igual a cero, salta a la marca de dirección “lo1” (¡emplear comando de combinación!)
- Recupera el contenido salvado de R7 del Stack
- Recupera el contenido salvado de R6 del Stack
- Salta del subprograma al programa principal

Procedimiento :

- Abra el proyecto “CMC1-2” e inserte el archivo nuevo “cmc124.asm”. Inserte las instrucciones necesarias para el Ensamblador.
- Añada a éstas la siguiente línea nueva :
  - ➪ USING bank0
- Introduzca los comandos. Observe los símbolos indicados (marcas de dirección)
  - ➥ Determine los respectivos operandos final y fuente
    - ➪ ¿Qué registro ?

- ¿Qué dirección de memoria ?
- ¿Qué tipo de direccionamiento ?
- ¿Qué constante ?

- Busque el correspondiente código de comando de la lista de comandos
- Introduzca la línea de comando compl.

NOTA : R0-R7 pueden ser procesados como direcciones de datos (dadr) si se designan con AR0-AR7. ¡Para ello debe estar introducida la directiva „USING bank0“!
¡Emplée esta posibilidad p.e. para los comandos PUSH, POP y ANL !

- Crée un archivo Intel-Hex y cárguelo en el controlador.

- Ejecute el programa en servicio por pasos individuales.

NOTA : Abandone los bucles DJNZ introduciendo la constante 01 en el correspondiente registro de decremento.

Preguntas :

¿Qué diferencia hay entre las abreviaturas Ri y Rr?

¿Con qué registros se puede direccionar indirectamente? ¡Indique un ejemplo de comando!

¿Con qué signo se distinguen los constantes de direcciones? ¿Qué sucede si se olvida este signo?

¿Por medio de qué comando se puede invertir el nivel de salida de un pin de puerto? ¡Dé un ejemplo!

¿Con qué comandos se puede ensayar el nivel de entrada de un pin de puerto y, en función del resultado, saltar a otra dirección de programa? ¡Dé un ejemplo!

¡Explique el comando „RR A“! ¿Qué registros se pueden utilizar como operando para un comando de desplazamiento?

¡Explique el comando „DJNZ RR,rel“!

¿Por medio de qué comando se pueden archivar los contenidos de direcciones de memoria de datos en el Stack (memoria de apilamiento) (salvar)? ¿Cómo se pueden recuperar de allí?

¿Por medio de qué comando se vuelve desde un subprograma?

¿Qué lista de programa ha escrito usted?

```
; ********************************************************
; * Proyecto:  Lista de comandos C515C del 21.11.1999    *
; * Archivo de proyecto: CMC1-2                          *
; * Archivos fuente:     cmc124.asm                      *
; * MC-Tools:            as, p2hex, debugger             *
; * Perfil:              Perfil integrado PSD1_C515C_AS  *
; * Recurso MC:          sin                             *
; * Realizado por:                                       *
; ********************************************************


; -------------------Descripción de funcionamiento--------
; Se presentan ejemplos de programa / comandos para la
; solución del ejercicio 3 del ensayo CMC1-2
; --------------------------------------------------------


; ********************************************************
; -----Instrucciones necesarias para el Ensamblador -------

   CPU 80515          ; Selección de CPU para Ensamblador
   INCLUDE c515c.inc  ; Insertar definiciones SFR
   USING   bank0      ; Instrucción relativa al banco de registro
                      ; a utilizar

; ********************************************************

;                Programa principal
; --------------------------------------------------------

start:
```

```
END
```

CMC 1-3 Un primer programa con el microcontrolador C515C

Técnica de programación, técnica de planificación (análisis de problema, diagramas de flujo), lista de programa

Fundamentos preliminares

Al principio de cada programación debe haber un análisis detallado del problema a solucionar.

La tarea casi siempre verbalmente descrita se debe convertir en una forma descriptiva apropiada (p.e. diagrama de flujo). Por medio de etapas intermedias se elabora una serie de comandos de microcontrolador. Mediante el procesamiento secuencial de estos comandos se soluciona el problema. De la secuencia de comandos elaborada de esta manera (programa) se genera un archivo cargable (p.e. archivo Intel-Hex) para cargarlo en el controlador de destino.
A continuación el programa es depurado, o sea se buscan, se encuentran y se eliminan los errores de programa. Para ello se utiliza preferiblemente el servicio por pasos individuales.

En primer lugar se realiza un plan aproximativo del hardware necesario para el problema, con el objetivo de utilizar el mayor número posible de componentes de periferia „on-chip“. Los diagramas de flujo se elaboran desde el nivel de borrador hasta el nivel de comandos (diagrama de flujo de programa - DFP). Partiendo del DFP se puede escribir directamente la lista de programa. Todos los comandos se deberán dotar de un comentario breve de valor informativo. De un modo general se debe aspirar a una programación modular, o sea el programa es dividido en elementos pequeños (subprogramas) (p.e. operaciones de entrada/salida, operaciones aritméticas, bucles de espera). Los subprogramas de este tipo son componentes de programa de uso múltiple.

Normalmente un microcontrolador es iniciado por medio de una señal de RESET externa. Tras aplicar la señal, el contador de programa se encuentra en la dirección 0000H y todos los registros de funciones especiales han sido sometidos a una inicialización estandar.
En la dirección 0000H debe encontrarse el primer comando del programa. Los primeros comandos de un programa deben hacer que los componentes „on-chip“ del microcontrolador adopten el estado de inicio deseado (inicialización). Durante la inicialización se cargan p.e. los registros de control de la periferia „on-chip“ (temporizador, interfaz, convertidor análogo-digital, etc.) y se ajusta el sistema de interrupción. A continuación sigue el programa principal.

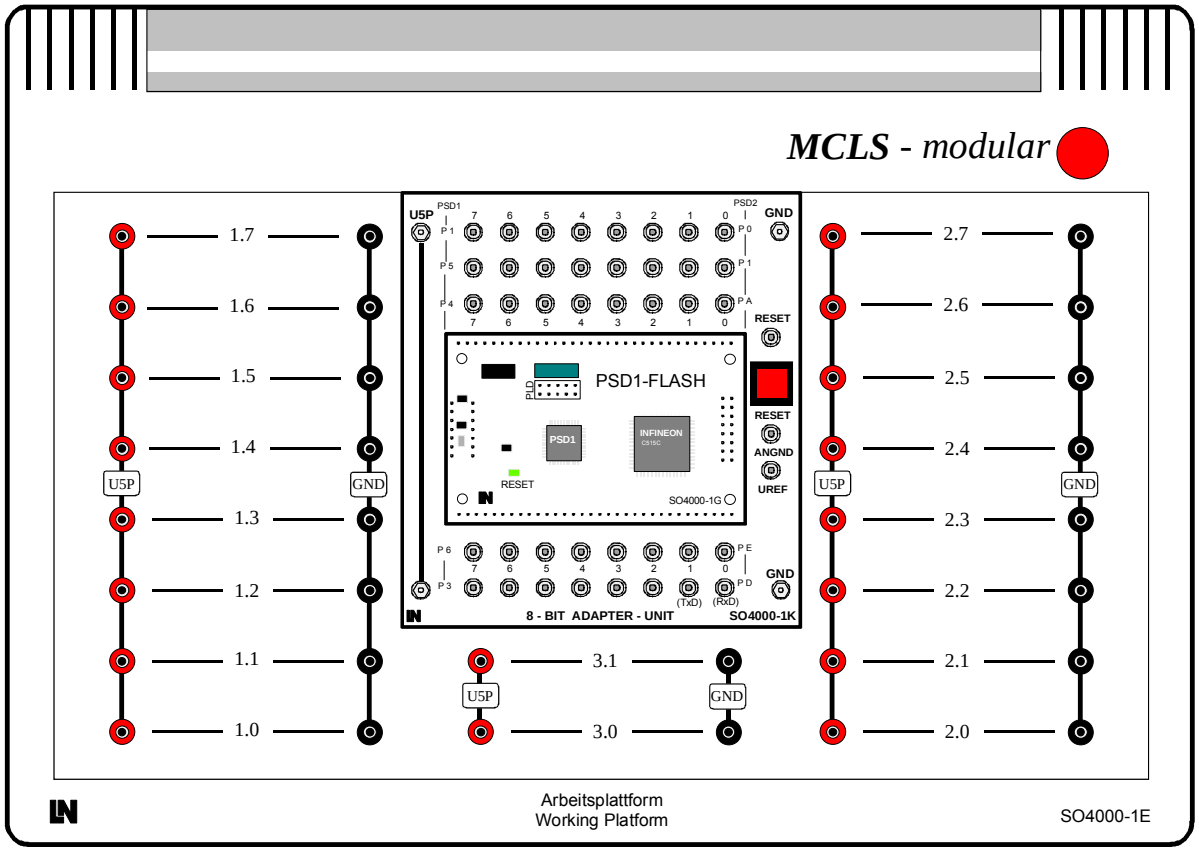


Fig.: 301: Instalación de aparatos del ensayo CMC 1-3

Instalación experimental CMC 1-3

Aparatos y accesorios necesarios

Unid.	Designación	Nº correl.
1	Plataforma con mód. de alim. +5V	SO4000-1E
1	Unid. de alim. de enchufe AC 90..230V 45..65Hz, DC 9V 630mA	SO4000-1F
1	Mód. PSD1-FLASH Controlador C515C	SO4000-1G
1	Unidad de adaptador de 8 bit	SO4000-1K
1	Unidad de LED	SO4000-1P
1	Unidad de teclas	SO4000-1Q
1	Unidad de interruptores	SO4000-1R
1	Adaptador BNC	SO4000-1T
1	Software IDE para MCLS (D,GB,F,E)	SO4001-9H
1	Oscilosc. incl. puntas de prueba	LM6203
1	Cable de medición BNC / BNC	LM9034
1	Cable de interfaz en serie 9/9 pol.	LM9040
8	Línea de medición 2mm 15cm azul	SO5126-5K
8	Línea de med. 2mm 15cm amarillo	SO5126-5M
1	CMC 1 Introducción a la programación de microcontroladores 8051	SH5019-1A
1	Manual de instrucciones Mód. PSD1-FLASH	BD4000-1G

Instalación de aparatos

¡Separe la alimentación de corriente del MCLS-*modular* !
Conecte la plataforma de trabajo por medio del cable de interfaz en serie a su PC.
Encaje la unidad de adaptador de 8 bits con el módulo de PSD1-FLASH adaptado en ella en el campo de casquillos 4.0 - 4.5 en el centro de la plataforma. Seleccione los módulos de experimentación necesarios según la tarea y conéctelos a los puertos seleccionados por usted.
Conecte la alimentación de corriente al MCLS-*modular*.

Nota relativa a la seguridad:
¡Asegúrese de conectar correctamente los potenciales de tensión (izda U5P, dcha GND)! No se deben aplicar tensiones extrañas a los puntos de medición llevados hacia fuera, ¡ya que esto podría dar lugar a la destrucción de los componentes de las placas!

Indicación relativa a la literatura

Recurra para la realización del ensayo al manual del entorno de desarrollo integrado (IDE) y al manual de instrucciones de la unidad PSD1-FLASH. Lea la información compacta en el anexo del manual de instrucciones del módulo PSD1-FLASH.

Ejercicio 1:
Técnicas de programación, análisis de problema, técnicas de planificación (diagramas de flujo), lista de programa

Objetivo :

- Fundamentos generales de la programación
- Programación en línea recta
- Distribuidores
- Técnica de bucles
- Técnica de subprogramas
- Análisis de problemas de un ejercicio
- Plan borrador como diagrama de flujo
- Plan de algoritmos como diagrama de flujo
- Plan de programa como diagrama de flujo a nivel de comandos
- Símbolos habituales de diagramas de flujo
- Elaboración de una lista de progr ama

Ejercicio

Elabore el plan borrador, de algoritmos y de programa (como diagrama de flujo respectivamente) para una luz de desplazamiento sucesivo. Realice previamente un análisis de problemas.
La luz de desplazamiento sucesivo se debe indicar en una línea de LED con 8 LED. La dirección de desplazamiento de la luz de desplazamiento sucesivo se cambiará por medio de un conmutador deslizante. El tiempo para los cambios entre los distintos LED deberá ser de aproximadamente 131 ms.
Escriba la lista de programa (texto fuente) partiendo del diseño de programa.

Conocimiento básico :

- Fundamentos generales de la programación

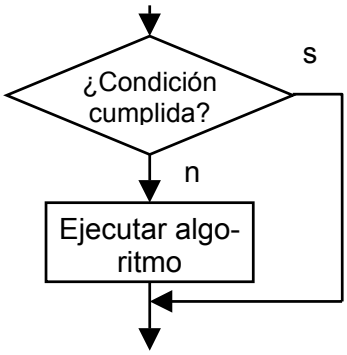
  - Configurar los programas de manera sencilla, clara, fiable y fácilmente aplicable
  - La división en programas parciales es conveniente (programación modular)
  - Documentar de manera clara las prestaciones y los interfaces de los programas parciales
  - Estandarizar los interfaces
  - Mantener el volumen de programación dentro de un margen económico
  - Tener en cuenta la economía de hardware (espacio en memoria, recursos „on-chip“)

  NOTA : Intente conseguir siempre una programación estructurizada ya que aumenta la claridad y facilita la detección de errores.

  ATENCIÓN : Emplée la programación „Top-Down“, o sea una planificación de programa que empiece con las propiedades principales del sistema (nivel más alto) y continua con los detalles (nivel bajo).

- Programación continua
  - Comandos secuenciales

- Distribuidor
  - Ramificación de programa
    - es realizada por una operación de decisión



- Técnica de bucles
  - Utilizar bucles de programa

    - Bucle de programa :
      Ejecutar varias veces una secuencia de comandos con operandos eventualmente cambiados

    - Componentes de un bucle de programa
      - Inicialización del criterio de bucle (p.e. registro)
      - Procesamiento de la función útil
      - Modificación del criterio de bucle
      - Consulta final

NOTA : Los bucles de programa reducen el volumen de programa y de memoria.

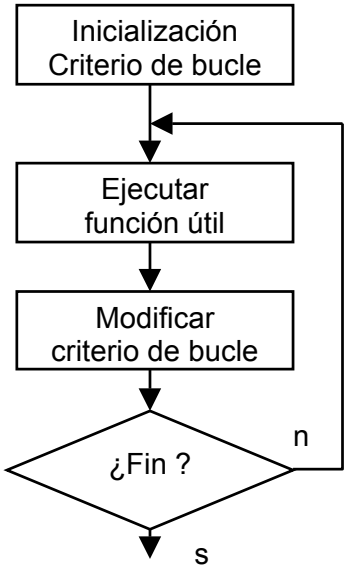
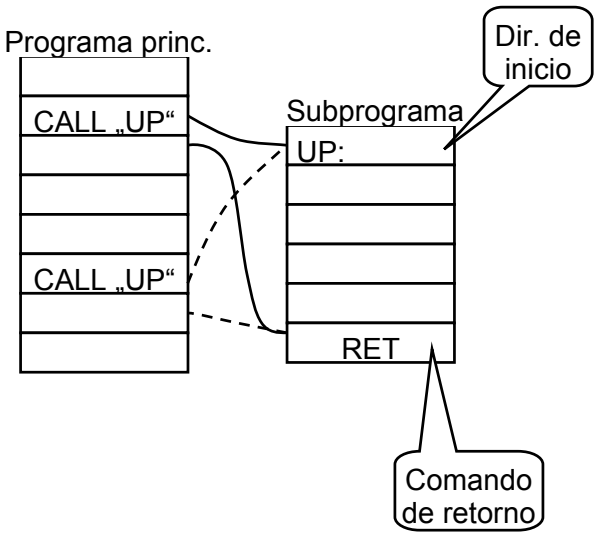


Fig.: Bucle de programa típico

- Técnica de subprograma
  - Utilización de subprogramas

    - Subprograma (SP) :
      - Secuencia de comandos que se necesita muchas veces en el programa. (Subrutina, procedi-miento)
      - Es llamado por el programa principal
      - La dirección donde se debe continuar con el programa principal después del SP, se archiva en el Stack (memoria de apilamiento)
      - Es finalizado con un comando de retorno especial (p.e. RET)

NOTA : Los subprogramas permiten una configuración de programa clara, así como la re-ducción del volumen de programación y de memoria.



ATENCIÓN : Los contenidos de los registros empleados en el SP se deben archivar en el Stack (memoria de apilamiento) tras entrar en el SP y deberían recuperarse de allí antes del retorno del SP

- Análisis de problema de una tarea

  - Objetivo : De la tarea verbal se elabora un plan aproximativo del hardware necesario con el objetivo de utilizar el mayor número de componentes de periferia „on chip“ posible.

  - Se deben contestar las siguientes preguntas:

    - ¿Es posible dividir el problema en problemas parciales?
    - ¿Qué se debe controlar, planificar, calcular?
    - ¿Qué periferia „on chip“ se necesita? ¿Presenta una potencia suficiente?
    - ¿Se necesitan interrupciones?
    - ¿Cuántas entradas/salidas se necesitan?
    - ¿Son de esperar estados críticos en cuanto al tiempo?
    - ¿Existen soluciones ya conocidas que podrían ser utilizadas?
    - ¿Qué estado inicial debe presentarse?

- Plan borrador como diagrama de flujo

  - Descripción aproximativa del desarrollo de procesamiento en un orden cronológico
  - Para cada bloque se introduce el procedimiento a realizar (¿Qué se debe hacer?)

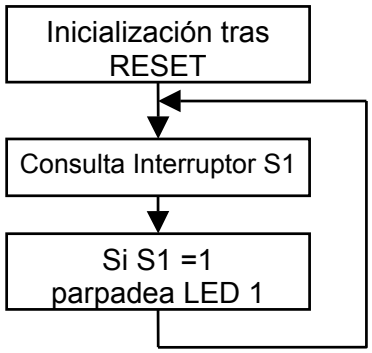


Fig. 303: Diagrama de flujo a nivel de borrador

ATENCIÓN : ¡ El plan borrador se debe elaborar independientemente del procesador !

- Plan de algoritmos como diagrama de flujo (independiente del procesador)

  - Más detalles relativos al nivel de borrador hasta obtener instrucciones de procesamiento fáciles de realizar.
  - Debe estar determinado para cada bloque qué acciones se deben realizar en qué orden (sin entrar en detalles de procesador)
  - Debe ser reconocible cómo se debe realizar la operación necesaria.
  - Utilizable como diagrama de flujo de programa (DFP)

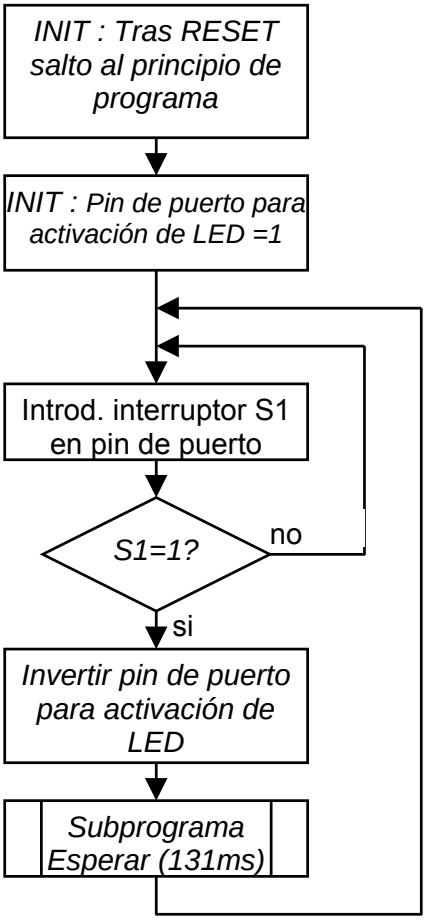


Fig. 304: Diagrama de flujo a nivel de algoritmos

ATENCIÓN : ¡El diseño de algoritmos se debe elaborar independientemente del procesador!

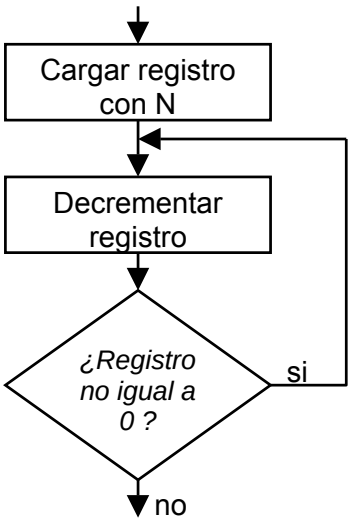


Fig. 305: Subprograma Esperar (Diagrama de flujo a nivel de algoritmos)

- Diseño de programa como diagrama de flujo a nivel de comandos (dependiente de procesador)

  - Más detalles relativos al nivel de algoritmos empleando el conjunto de comandos
  - Disolución de todos los algoritmos
  - Como máximo 3 comandos por bloque
  - El programa se puede deducir directamente
  - Diagrama de flujo de programa (DFP)

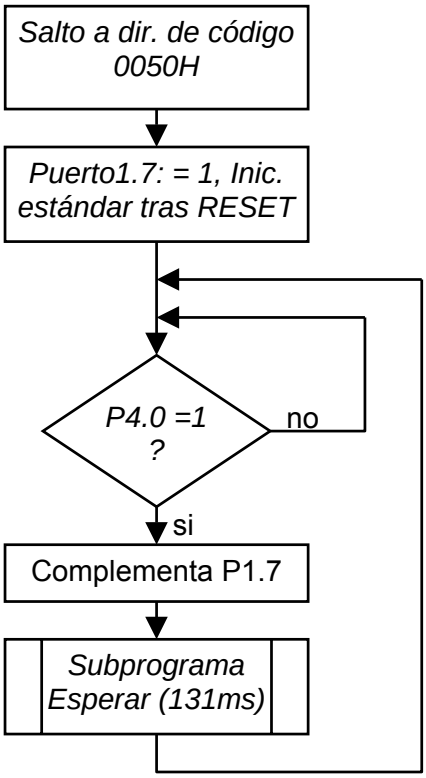


Fig. 306: Diagrama de flujo a nivel de comandos

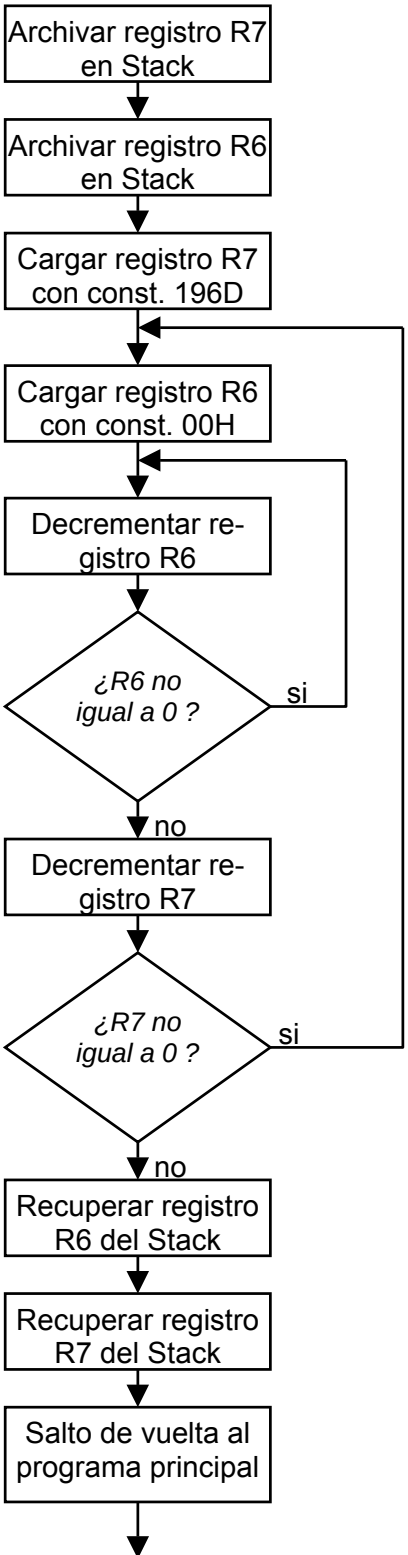
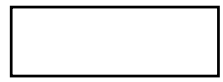
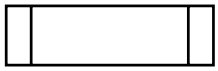


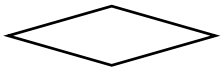
Fig. 307: Subprograma Esperar (diagrama de flujo a nivel de comandos)

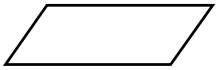
ATENCIÓN : ¡ El diseño de programa se debe elaborar dependientemente del procesador !
ATENCIÓN : ¡ Cada bloque debe contener como máximo 3 comandos !

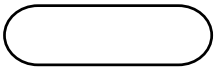
- Símbolo habituales de diagramas de flujo


Instrucción de trabajo


Subprograma

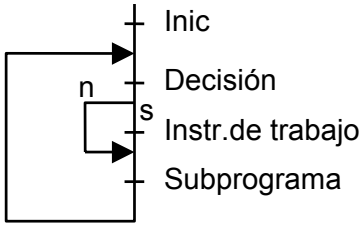

Operación de decisión con ramificación sucesiva


Operación de salida o entrada


Principio o fin de un ciclo de programa o punto de entrada de una rut-ina de interrupción


Línea de flujo

NOTA : Otro método igualmente habitual aparte del método de líneas de flujo presentado es el método de líneas de dirección.



Método de líneas de dirección

NOTA : La secuencia de plan borrador, plan de agoritmos y plan de comandos corresponde al principio del diseño „Top-Down“.

- Elaboración de una lista de programa

  - Estructura
    - Cabeza de programa :
      - Título de programa
      - Nombre del programador
      - Número de versión
      - Fecha de elaboración
    - Descripción breve :

- Función
- Condiciones de entrada y salida

➪ Instrucciones de Ensamblador :
- Archivos Include especiales

➪ Asignaciones de valores :
- Asignaciones de símbolo
- Asignaciones de direcciones

➪ Programa principal
- Comandos
- Comentarios
- El último comando es el salto al principio del programa principal o a sí mismo

➪ Subprogramas
- Eventual cabeza de subprograma (como cabeza de programa)
- Archivar el contenido de los registros empleados en el Stack
- Comandos
- Comentarios
- Recuperar el contenido archivado de los registros del Stack
- Retorno al programa principal

➪ Tablas de constantes
- Valores binarios constantes como fuente de datos

NOTA : La utilización constante de asignaciones de valores al principio de la lista permite la modificación rápida de constantes y direcciones en todo el programa.
ATENCIÓN : ¡Los comentarios deben explicar la operación y no el comando !

➪ Reglas de escritura
➪ Observar Ensamblador

Procedimiento :

- Conteste, partiendo del problema, las preguntas del análisis de problema.

- Elabore el diagrama de flujo para el diseño de concepto.
  ➪ Clasificar la tarea en problemas parciales (resultado del análisis de problema)

  ➪ Dibujar un diseño según el método de líneas de dirección

- Elabore el diagrama de flujo para el plan de algoritmos

  ➪ Convertir los problemas parciales del diseño de concepción en algoritmos generales

  ➪ Definición general de los componentes de periferia
  ➪ Determinar el componente de periferia para la edición en la línea LED
  ➪ Determinar el componente de periferia para la introducción del estado de interruptor

  ➪ Determinar generalmente las memorias intermedias (p.e. registros, memoria de datos)

- Determinar las ramificaciones, los bucles y los subprogramas
- Dibujar el diseño de programa principal y subprograma según el método de líneas de flujo

- Elabore el diagrama de flujo para el diseño de programa (para C515C)
  - Convertir todos los algoritmos del diseño de algoritmos en comandos concretos
  - Definición de los componentes de periferia
    - Designar componentes de periferia para la edición en línea LED
    - Designar componentes de periferia para la introducción del estado de interruptor
  - Determinar memorias intermedias (designación de registros, dirección de memoria de datos)
  - Descodificar ramificaciones, bucles, subprogramas en todos los algoritmos
  - Dibujar un diseño de programa principal y subprograma según el método de líneas de flujo
- Elabore del diagrama de flujo del diseño de programa la lista de programa
  - Elabore el proyecto „CMC1-3“, utilice el mismo perfil de usuario como en los ensayos CMC1-1 y CMC1-2, inserte un nuevo archivo de texto fuente „cmc13.asm“.
  - Introduzca la lista de programa según la estructura general en el archivo de texto fuente. Observe las reglas de Ensamblador.
  - Archive los contenidos de los registros empleados en el subprograma en el Stack
- Crée de este archivo el archivo Intel-Hex y cárguelo en el controlador.
- Ensaye su programa mediante Depurador
- Corrija los errores de programa hasta que el programa se ejecute sin errores

Preguntas :

¿Qué se entiene por programación „Top-Down“?

¿Qué secuencia de diseños corresponde a este principio?

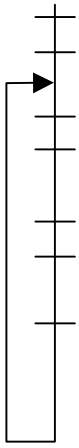
¿Qué dependencia de procesador existe respectivamente en el plan borrador, el plan de algoritmos y el plan de programa?

¿Cuántos comandos debe haber como máximo en un bloque ?

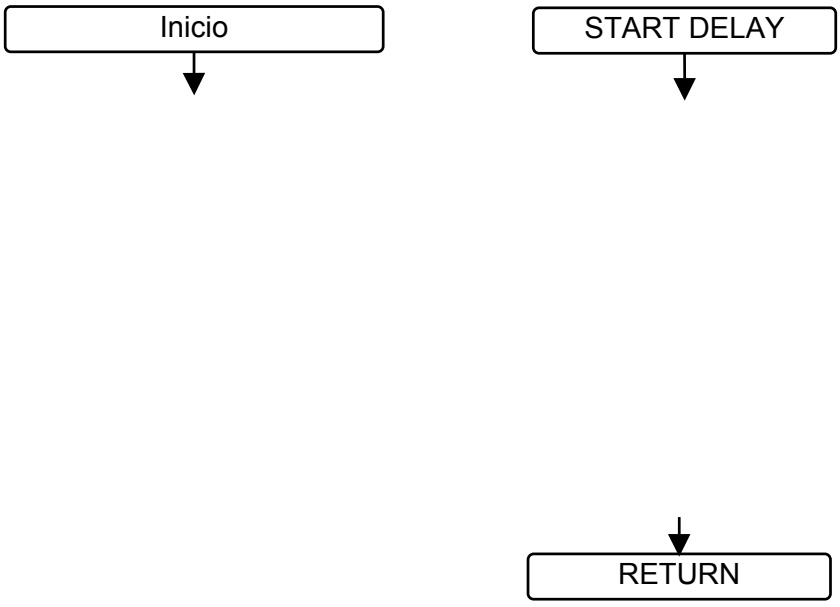
¿Qué métodos para la elaboración de diagramas de flujo conoce usted?

Soluciones de los planes y de la lista de programa

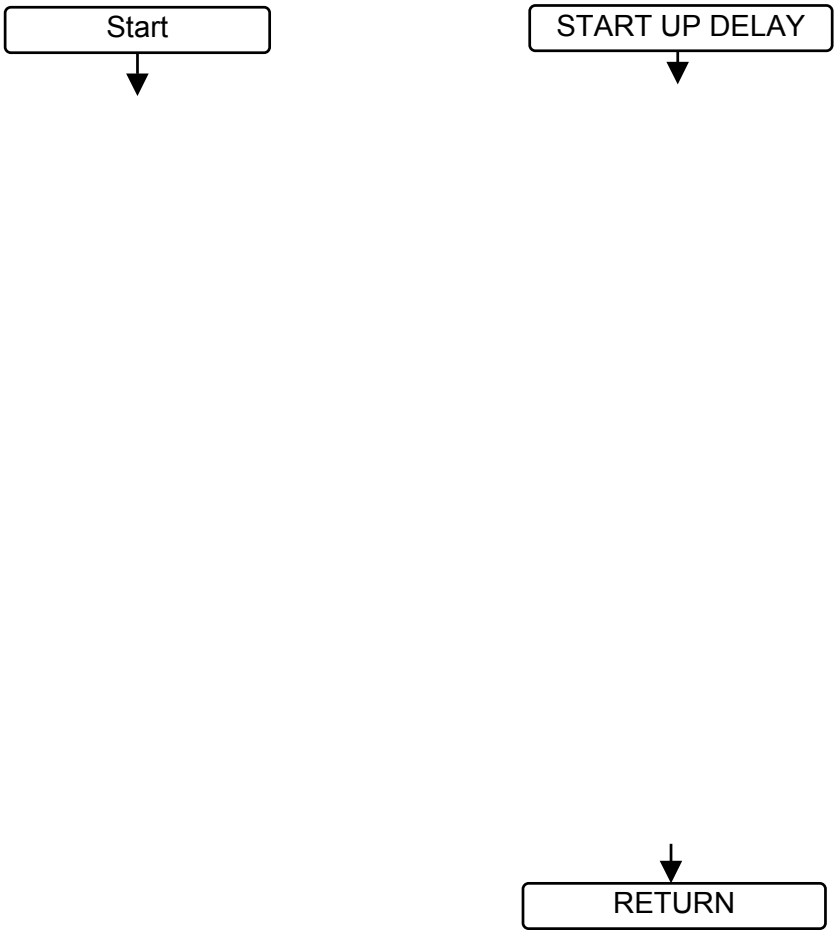
- Plan borrador



- Plan de algoritmos



➪ Plan de programa



Lista de programa

```
; ******************************************************
; * Proyecto: Luz de desplaz.sucesivo LED 23.09.2005  *
; * Archivos  de proyecto: CMC1-3                     *
; * Archivos fuente: cmc13.asm                        *
; * MC-Tools:   as, p2hex, debugger                   *
; * Perfil:     Perfil integrado PSD1_C515C_AS        *
; * Recursos MC:Unidad de LED completamente en P1     *
; * Elaborado por:                                    *
; ******************************************************
;              Descripción de funcionamiento
; Todos los LED de la unidad de LED están conectados al puerto
; completo 1 (P1.0 - P1.7) del MC. Un interruptor de la unidad
; de interruptores se conecta al pin de puerto P4.7.
; Por medio del nivel lógico ajustado por conmutador deslizante
; se determina la dirección de desplazamiento de la luz de
; desplazamiento sucesivo de LED. La consulta del interruptor
; se realiza por medio del comando de ensayo de bit 'JB'.
; El patrón de bits para la activación del puerto se desplaza
; con los comandos de rotación 'RL A' o 'RR A'.
; ------------------------------------------------------------
;           Instrucciones necesarias para el Ensamblador
   CPU      80515       ; Selección MC
   INCLUDE  c515c.inc   ; Definición SFR
   SEGMENT  code        ; Segmento de código
   ORG      0000H       ; Primer comando a partir de 0000H

; ************************************************************
; *   PROGRAMA PRINCIPAL Luz de desplazamiento sucesivo LED   *
; ************************************************************
START:









; ************************************************************
;           Subprograma DELAY, aproximadamente 131 ms
; ------------------------------------------------------------




; ------------------------------------------------------------
   END
```

Anexo

Reglas importantes para el Ensamblador de A.Arnold para derivados de 8051

Formato de introducción

<Símbolo (Etiqueta)> **<Comando / Directiva> <Parámetro (,Parámetro)>** ; Comentario

Símbolo (Etiqueta)

- Identificador para dirección de comando (marca) en la memoria de programa (la dirección exacta es asignada por el Ensamblador)
- Identificador para valor numérico asignado (ver asignación de valor, asignación de dirección)

hay símbolos fijamente definidos, específicos de procesador, p.e. nombres para registros y registros de funciones especiales) y símbolos libremente definibles

La 1ª columna de la línea (a partir de la 2ª columna son necesarios dos puntos) debe empezar con una letra

Ejemplo

```
Fijamente definido :
R0, R1 ... R7, A, B                          ; Símbolos para regist. de trabajo
P0, P0.1, SP, DPL, DPH, TL0 ... usw.         ; Símbolos para reg. de funciones
                                             ; especiales
Libremente definido :
TIEMPO          MOV       R1,R2              ; Símbolo en 1ª col., identificador
                                             ; para dirección de comando
 NEXT:          MOV       R0,A               ; Símbolo no en 1ª col.
Const. de tiempo EQU       0A8h               ; Símbolo en 1ª col., identificador
                                             ; para valor numérico
M02x3                                        ; Símbolo de letras y cifras
```

Comando Comando de lista de comandos del procesador

a partir de 2ª col. de la línea o 1 espacio en blanco o tabulador detrás de símbolo

Ejemplo

```
 MOV        R1,R2                            ; Comando en 2ª col.
            MOV       R0,R2                  ; modo de escribir habitual, tras
                                             ; tabulador
START  MOV  R1,R2                            ; Comando 1 espacio tras símbolo
```

Directiva	Instrucción para el control del Ensamblador	**a partir de la 2ª col. de la línea o 1 espacio en blanco o tabulador detrás de símbolo (todo como para comando)**
Parámetro	Indicación necesaria para comando o directiva (p.e. operandos), puede ser símbolo, valor numérico, signo ASCII o también una secuencia de signos	**1 espacio en blanco o tabulador detrás de comando o directiva, varios se separan por una coma**

Ejemplo

```
                MOV     R1,R2           ; 2 parámetros separados por
                                        ; una coma,1 espacio en blanco
                                        ; detrás de comando, símbolos
                CALL    TIEMPO          ; 1 parámetros, símbolo

Tabla           DB      10h,20h,30h     ; 3 parámetros, valores numéricos
                MOV     R0,#'a'         ; 2 parámetros, 1x símbolo, 1x
                                        ; signo ASCII (para a)
                DB      „¡ Hola !"      ; 1 parámetro, secuenc. de signos
```

Comentario	Indicaciones relativos al desarrollo del programa	**Empieza con punto y coma, termina al final de línea**

Ejemplo

```
; MOV  R0,#10   ¡Cada texto detrás de un punto y coma es valorado como comentario !
;este comentario empieza al principio de la línea
                MOV     R1,R2           ; este comentario empieza
                                        ; después de la introducción
                                        ; completa del comando y fina-
                                        ; liza siempre al final de la línea
```

INDICACIÓN, de un modo general se admiten tanto mayúsculas como minúsculas (también combinadas).

Formatos de cifras

Decimal	sin adición	12, -1000
Hexadecimal	H (h) postpuesta	11h, 0AAh
Binario	B (b) postpuesta	11100111b

ATENCIÓN ¡las expresiones hexadecimales que empiezan con letras deben estar precedidas por un cero!

Directivas (instrucciones de Ensamblador)

DETERMINAR EL PROCESADOR

- **CPU**
  Instrucción que indica qué lista de comandos específica de procesador se ha de utilizar.

  <en blanco> **CPU** **<Procesador>**

  Procesador :

  - 8051
  - 8052
  - 80515
  - 80517
  - 80C320
  - 87C750

INSERTAR ARCHIVOS

- **INCLUDE**
  Inserta, durante el ensamblado, en este punto un archivo en el texto

  <en blanco> **INCLUDE** **<Nombre de archivo>**

INDICACIÓN ¡si no se indica una terminación del archivo, se pone automáticamente la terminación „.inc“!

ATENCIÓN ¡para qué el Ensamblador reconozca los símbolos de los registros de funciones especiales, es necesario insertar el archivo Include „c515c.inc“!

FIN DE PROGRAMA

- **END**
  Caracteriza el fin del programa de Ensamblador.

  <en blanco> **END**

ATENCIÓN ¡los textos fuente que se encuentran después de END no se traducen !

Ejemplo

```
        cpu     80515           ;Procesador 80515
        include c515c.inc       ;Insertar archivo „c515c.inc“
          .
          .
          .
        end                     ;Fin de programa
        mov     r2,#zk2         ;¡ No se traduce !
```

ASIGNACIONES DE VALOR
En el ensamblado se sustituyen todos los símbolos existentes en el programa por sus valores asignados.

- **EQU**
  Instrucción para asignar al <símbolo> indicado el <valor> asignado.
  El valor asignado no se puede modificar posteriormente.

  <Símbolo> EQU <Valor>

- **SET**
  Instrucción para asignar al <símbolo> indicado el <valor> indicado.
  El valor asignado se puede modificar posteriormente.

  <Símbolo> SET <Valor>

Ejemplo

```
zk1     equ     0Aah    ;Símbolo „zk1“ = AAH
zk2     set     20h     ;Símbolo „zk2“ = 20H
        mov     r0,#zk1 ;Contenido registro R0 : = Aah
        mov     r1,#zk2 ;Contenido registro R1 : = 20h
zk2     set     40h     ;Símbolo „zk2“ = 40H
        mov     r2,#zk2 ;Contenido R2 : = 40h
```

PROCESAMIENTO DE VALORES DE 16 BITS
Instrucción para emplear la parte Low (High) del valor de 16 bits indicado entre paréntesis.

- **LO** (parte Low)
- **HI** (parte High)

<Símbolo> Comando Parámetro, **# LO/HI (valor)**

Atención: Para que el Ensamblador reconozca las instrucciones LO y HI es necesario insertar el archivo bitfuncs.inc por medio de include.

Ejemplo

```
Constdetiempo   equ   -5000  ;asigna al símbolo „Constdetiempo“ un valor de 16 bits

                mov   th0,# hi(constdetiempo)   ;carga parte High del registro
                                                ;contador de temp. 0
                                                ;con parte High del valor de
                                                ;16 bits „-5000“
                mov   tl0,# lo(constdetiempo)   ;carga parte Low del registro
                                                ;contador de temp. 0
                                                ;con parte Low del valor de
                                                ;16 bits „-5000“
```

ASIGNACIONES DE DIRECCIÓN
En el ensamblado se sustituyen todos los símbolos existentes en el programa por sus direcciones asignadas.

- **SFR**
  Instrucción para asignar al <símbolo> indicado la <dirección> indicada de la memoria de datos interna directamente direccionable.

  <Símbolo> SFR <Dirección>

  INDICACIÓN ¡el valor asignado no se puede modificar posteriormente !

- **SFRB**
  Instrucción para asignar al <símbolo> indicado la <dirección> indicada de la memoria de datos interna directamente direccionable (como SFR) y para crear para los distintos bits un símbolo direccionable por bits („símbolo.0“ – „símbolo.7“)

  <Símbolo> SFRB <Dirección>

  INDICACIÓN ¡el valor asignado no se puede modificar posteriormente !

ATENCIÓN ¡ El Ensamblador no comprueba si la dirección realmente es direccionable por bits !

- **BIT**
  Instrucción para asignar al <símbolo> indicado la <dirección de bit> indicada de la memoria de datos interna direccionable por bits.

  <Símbolo> BIT <Dirección de bit>

INDICACIÓN ¡ el valor asignado no se puede modificar posteriormente !

Ejemplo

```
valor1      sfr   30h          ;Símbolo „valor1“ = dirección 30H de la memoria de
                               ;datos
valor2      sfrb  22h          ;Símbolo „valor2“ = dirección 22H de la memoria de
                               ;datos
                               ;Símbolos „valor2.0 – valor2.7“ = bits 0 - 7
sts1        bit   10h          ;Símbolo „sts1“ = dirección de bit 10H de la memoria de
                               ;datos direccionable por bits
        mov   valor1,#1Ah      ;Contenido dirección 30H := 1AH (hex)
        mov   valor2,#11111111b ;Contenido dirección 22H := 11111111b (bin)
        clr   valor2.3         ;Contenido dirección 22H := 11110111b
        setb  sts1             ;Contenido dirección de bit 10H := 1
```

ASIGNACIÓN DE ZONA DE MEMORIA

- **SEGMENT**
  Instrucción para poner el contador de dirección a la zona de memoria indicada a continuación.

  <en blanco> **SEGMENT** **<Zona de memoria>**

  Zona de memoria :

  - **CODE**
    Memoria con código de programa (Dirección 0000H - FFFFH)

ATENCIÓN ¡contador de dirección preajustado a 0000H !

  - **DATA**
    Memoria de datos interna directamente direccionable

    - Bancos de registro (RB0 –RB3) (Dirección RAM 00H – 1FH)
    - Zona de memoria direccionable por bits (¡se direcciona por bytes !)
      (Dirección RAM 20H – 2FH)
    - Zona libremente disponible (sin funciones especiales)
      (Dirección RAM 30H - 7FH)
    - Registro de funciones especiales (SFR) (Dirección 80H - FFH)

ATENCIÓN ¡ contador de direcciones preajustado a 30H (zona libremente disponible)!

  - **IDATA**
    Memoria de datos interna indirectamente direccionable

    - Zona RAM interna inferior (Dirección RAM 00H – 7FH)
    - Zona libremente disponible (sin funciones especiales)
      (Dirección RAM 80H - FFH)

ATENCIÓN ¡contador de direcciones preajustado a 80H (zona libremente disponible) !

  - **BITDATA**
    Memoria de datos interna directamente direccionable por bits

    - Zona libremente disponible (sin funciones especiales)
      (Dirección de bit 00H - 7FH)
    - SFR direccionables por bits (Dirección 80H - FFH)

ATENCIÓN ¡contador de direcciones preajustado a la dirección de bit 00H (zona libremente disponible) ! ¡ Directiva ORG ajusta direcciones de bit !

  - **XDATA**
    Memoria de datos externa indirectamente direccionable
    (Dirección 0000H - FFFFH)

ATENCIÓN ¡contador de direcciones preajustado a 0000H!

ATENCIÓN ¡existen diferentes zonas de memoria físicamente separadas con los mismos nombres de dirección !

ATENCIÓN ¡el contador de direcciones está preajustado por zona de memoria, asignar otras direcciones deseadas mediante directiva ORG!

ASIGNACIÓN DE CONTADOR DE DIRECCIÓN

- **ORG**
  Instrucción para poner el contador de direcciones en el SEGMENTO actual (zona de memoria) a la dirección indicada a continuación.

  <en blanco> **ORG** **<Dirección>**

  ATENCIÓN ¡si no se ha asignado expresamente un SEGMENTO, el SEGMENTO DE CÓDIGO es considerado el segmento actual !

  Se admiten los siguientes valores de dirección :

  - CODE : 0000H - FFFFH (preajuste 0000H)
  - DATA : 00H - FFH (preajuste 30H)
  - IDATA : 00H - FFH (preajuste 80H)
  - BITDATA : 00H - FFH (preajuste 00H, *ATENCIÓN* ¡ direcciones de bit !)
  - XDATA : 0000H - FFFFH (preajuste 0000H)

Ejemplo

```
        segment   code    ;Zona de memoria de código, cont. de direcc. en 0000h
        org       100h    ;Contador de direcciones en 0100h
        mov       P0,a    ;Contenido dirección 0100h = comando „mov  P0,a“

        segment   data    ;Zona de memoria de datos, cont. de direcc. en 30h
var     db        ?       ;Símbolo „var“ = dirección 30H de la memoria de datos
                          ;Contenido dirección 30H = reservado

        segment   bitdata ;Zona de mem.de datos direcc. por bits, contador de dir.
                          ;de bits en 00h
sts     db        ?       ;Símbolo „sts“ = direcc. de bit 00H de la memoria direcc.
                          ;por bits de la memoria de datos
                          ;Contenido dirección 00H = reservado

        segment   xdata   ;Zona de memoria de datos ext. conect., contador de
                          ;dir. en 0000h
        org       1000h   ;Contador de direcciones en 1000h
```

DEFINICIONES DE DATOS

En el ensamblado se sustituyen todos los símbolos existentes en el programa por sus direcciones de memoria asignadas.

Reservación de memoria para variables

- **DB**
  reserva memoria por bytes (1 Byte)

- **DW**
  reserva memoria por palabras (2 Byte)

<Símbolo> **DB (DW)** **?**

Instrucción para reservar 1 BYTE (1 PALABRA) en la localización de memoria actualmente direccionada.
Al mismo tiempo asigna al <símbolo> indicado la dirección de memoria actual.

<Símbolo> **DB (DW)** **?,?,?,...n**

Instrucción para reservar n BYTE (n PALABRAS) a partir de la localización de memoria actualmente direccionada.
Al mismo tiempo asigna al <símbolo> indicado la dirección de memoria actual.

<Símbolo> **DB (DW)** **<Número de localizaciones >** **DUP** **(?)**

Instrucción para reservar por bytes (por palabras) el <número de localizaciones > indicado a partir de la localización actualmente direccionada.
Al mismo tiempo asigna al <símbolo> indicado la dirección de memoria actual.

ATENCIÓN ¡ al reservar espacio de memoria para variables, evitar las zonas (direcciones) con funciones especiales (bancos de registro, registros de funciones especiales)!

Ejemplo

```
        segment   data      ;Zona de memorias de datos, contador de dir. en 30h
var     db        ?         ;Símbolo „var“ = Dirección 30H de memoria de datos
                            ;Contenido dirección 30H = reservado

        segment   bitdata   ;Zona de memoria de datos direcc. por bits, contador de
                            ;direcciones de bit en 00h
sts     db        ?,?       ;Símbolo „sts“ = dirección 00H de la memoria de datos
                            ;direccionable por bits
                            ;Contenido dirección 00H = reservado
                            ;Contenido dirección 01H = reservado

        segment   xdata     ;Zona de memoria de datos externa, cont. de dir. 0000H
        org       1000h     ;Contador de direcciones en 1000h
mess    db   20   dup  (?)  ;Símbolo „mess“ = Dirección 1000H de la memoria
                            ;de datos externa
                            ;Contenidos direcciones 1000H – 1013H = reservado
```

Descripción de memoria con constantes

ATENCIÓN ¡solamente válido para SEGMENT CODE !

- **DB**
  describe memoria por bytes (1 Byte)

- **DW**
  describe memoria por palabras (2 Byte)

<Símbolo> **DB (DW)** **<Valor>**

Instrucción para reservar 1 BYTE (1 PALABRA) en la localización de memoria de programa actualmente direccionada y describirlo al mismo tiempo con el <valor> indicado.
Además asigna al <símbolo> indicado la dirección de memoria actual.

<Símbolo> **DB (DW)** **<Valor 1, Valor 2, ...Valor n>**

Instrucción para reservar n BYTES (n PALABRAS) a partir de la localización actual y describirlos sucesivamente con el <valor 1...n> indicado.
Además asigna al <símbolo> indicado la dirección de memoria actual.

<Símbolo> **DB** **'Signo '**

Instrucción para reservar 1 BYTE en la localización de memoria de programa actualmente direccionado y describirlo al mismo tiempo con el valor ASCII del <signo> indicado.
Además asigna al <símbolo> indicado la dirección de memoria actual.

<Símbolo> **DB** **'Signo 1' , 'Signo 2', ... 'Signo n'**

Instrucción para reservar n BYTES a partir de la localización actualmente direccionada y describirlos sucesivamente con los valores ASCII de los <signos 1...n> indicados.
Además asigna al <símbolo> indicado la dirección de memoria actual.

<Símbolo> **DB** **„Secuencia de signos"**

Instrucción para reservar 1 BYTE por signo a partir de la localización actualmente direccionada y describirlos sucesivamente con los valores ASCII de los signos individuales de la <secuencia de signos> indicada.
Además asigna al <símbolo> indicado la dirección de memoria actual.

Ejemplo

```
        segment     code        ;Zona de mem. de códigos, contador de dir. 0000H
        org         100h        ;Contador de direcciones en 100H
const   db          0c9h        ;Símbolo „const“ = dirección 100H de mem. de códigos
                                ;Contenido dirección 100H = C9H
korr    db          10,11       ;Símbolo „korr“ = dirección 101H de mem. de códigos
                                ;Contenido dirección 101H = 10 (decimal)
                                ;Contenido dirección 102H = 11 (decimal)

        segment     code        ;Zona mem. de cód., contador de direcciones 0000H
        org         200h        ;Contador de direcciones en 200H
str     db          ‚o‘,‘l‘,‘a‘ ;Símbolo „str“ = Dirección 200H de la memoria de cód.
                                ;Contenido dir. 0200H = 6Fh = Código ASCII   o
                                ;Contenido dir. 0201H = 6Ch = Código ASCII   l
                                ;Contenido dir. 0202H = 61h = Código ASCII   a

        segment     code        ;Zona de mem. de cód., cont. de dir. 0000H
        org         300h        ;Cont. de dir. en 300H
text    db          „Hola !“    ;Símbolo „text“ = Dire. 300H de la memoria de código
                                ;Contenido dir. 0300H = 48h = Código ASCII   H
                                ;Contenido dir. 0301H = 6Fh = Código ASCII   o
                                ;Contenido dir. 0302H = 6Ch = Código ASCII   l
                                ;Contenido dir. 0303H = 61h = Código ASCII   a
                                ;Contenido dir. 0304H = 20h = Código ASCII   espacio
                                ;Contenido dir. 0305H = 21h = Código ASCII   !
```

ASIGNACIÓN DE BANCOS DE REGISTRO

- USING
  Instrucción para ajustar el banco de registro indicado a continuación

  <en blanco> **USING** **<Banco de registro>**

  Bancos de registro posibles :

  - Bank0 Memoria de datos interna dirección 00H-07H
  - Bank1 Memoria de datos interna dirección 08H-0FH
  - Bank2 Memoria de datos interna dirección 10H-17H
  - Bank3 Memoria de datos interna dirección 18H-1FH

ATENCIÓN : ¡Los registros R0-R7 se asignan respectivamente al banco de registro actualmente ajustado!
¡Si no se ha asignado expresamente un banco de registro, banco 0 se considera el banco de registro actual !

ASIGNACIÓN DE DIRECCIONES A LOS REGISTROS DE TRABAJO

- AR

  Instrucción para asignar a los registros de trabajo R0-R7 la dirección de memoria de datos correcta dentro del banco de registro actual.

  AR<Número de registro>

  Número de registro : 0 – 7 R0 – R7

  ATENCIÓN : ¡ Los registros R0-R7 se asignan respectivamente al banco de registro actualmente ajustado!
  ATENCIÓN : ¡ Para qué los registros R0-R7 del banco de registro 0 se puedan procesar como dirección de memoria de datos, debe estar asignado expresamente el banco de registro 0 !

  NOTA : Mediante el empleo de AR0-AR7, R0-R7 pueden ser operandos de la memoria de datos interna directamente direccionada.

Ejemplo

```
                segment     code        ;Zona de mem. de cód., contador de dir. 0000H
                using       bank0       ;Banco de registro 0 es el actual
                org         100         ;Contador de direcciones en 100H
                    .
                    .
                    .
; ---Subprograma----

UP1             push        AR0         ;guarda R0 en el Stack
                push        AR1         ;guarda R1 en el Stack
                    .
                    .
                    .
                pop         AR1         ;recupera contenido guardado de R1 del Stack
                pop         AR0         ;recupera contenido guardado de R0 del Stack
                ret                     ;retorno al programa principal
```

Listas para el manual

cmc111.asm

```
; ********************************************************
; * Proyecto:   Programa de parpadeo del día 23.09.2005  *
; * Archivo de proyecto: CMC1-1                          *
; * Archivos fuente:     cmc111.asm                      *
; * MC-Tools:            as, p2hex, debugger             *
; * Perfil:              Perfil integrado PSD1_C515C_AS  *
; * MC-Ressource:        LED a clavija de puerto P1.7    *
; * Encargado:           HAG                             *
; ********************************************************
; ----------------Descripción de funcionamiento-----------
; Un LED libremente elegible de la unidad de LED se conecta a
; la clavija de puerto P1.7 de la unidad PSD1. Por medio del
; comando cíclico se invierte el nivel lógico en la clavija de
; puerto.
; Con el nivel 1 se desconecta el LED (APAGADO).
; Con el nivel 0 se conecta el LED (ENCENDIDO).
; El retardo de tiempo necesario entre los estados de LED es
; preparado por el subprograma TIEMPO. El tiempo de ejecución
; del subprograma Tiempo es de aprox. 131 ms.
; --------------------------------------------------------
; --------Acuerdos necesarios para el Ensamblador ---------

      CPU     80515     ; Selección de CPU para el Ensamblador
      INCLUDE c515c.inc ; Insertar definiciones SFR
      SEGMENT code      ; A partir de aquí segmento
                        ; de código de programa
      ORG     0000H     ; Código de programa a partir de 0000H

; ********************************************************
;            Programa principal: LED debe parpadear
; ********************************************************
START:
      CPL   P1.7        ; Complementar el nivel lógico en P1.7
      CALL  TIEMPO      ; Retardo de tiempo de subprograma
      SJMP  START       ; Programa como bucle continuo
; ********************************************************

; --------------------------------------------------------
;          Subprograma TIEMPO, aproximadamente 131 ms
; --------------------------------------------------------
; 2 bucles de DJNZ entrelazados
TIEMPO:
      MOV     R7,#000H  ; Contador para bucle exterior
Z01:  MOV     R6,#000H  ; Contador para bucle interior
Z00:  DJNZ    R6,Z00    ; 256*2µs (bucle interior)
      DJNZ    R7,Z01    ; 256*bucle interior
      RET
; --------------------------------------------------------
      END
```

cmc112.asm

```
; ********************************************************
; * Proyecto:    Luz de desplaz.sucesivo LED 23.09.2005  *
; * Archivos de proyecto: CMC1-1                         *
; * Archivos fuente:      cmc112.asm                     *
; * MC-Tools:             as, p2hex, debugger            *
; * Perfil:               Perfil integrado PSD1_C515C_AS *
; * Recursos MC: Unidad de LED completamente en P1       *
; * Elaborado por:                                       *
; ********************************************************
;              Descripción de funcionamiento
; Todos los LED de la unidad de LED están conectados al puerto
; completo 1 (P1.0 - P1.7) del MC. Un interruptor de la unidad
; de interruptores se conecta al pin de puerto P4.7.
; Por medio del nivel lógico ajustado por conmutador deslizante
; se determina la dirección de desplazamiento de la luz de
; desplazamiento sucesivo de LED. La consulta del interruptor
; se realiza por medio del comando de ensayo de bit 'JB'.
; El patrón de bit para la activación del puerto se desplaza
; con los comandos de rotación 'RL A' o 'RR A'.
; ------------------------------------------------------------
;           Instrucciones necesarias para el Ensamblador
   CPU      80515       ; Selección MC
   INCLUDE  c515c.inc   ; Definición SFR
   SEGMENT  code        ; Segmento de código
   ORG      0000H       ; Primer comando a partir de 0000H

; ************************************************************
; *   PROGRAMA PRINCIPAL Luz de desplazamiento sucesivo LED   *
; ************************************************************
START: MOV   A,#11111110B  ; Definir patrón de bit
ST0:   MOV   P1,A          ; Patrón de bit a puerto 1
       CALL  TIEMPO        ; Esperar aproximadamente 131 ms
       JB    P4.7, ST1     ; Consulta de interruptor
                           ; ¿Pin de puerto en nivel 1?
                           ; Si es así, a ST1
; ************************************************************
       RR    A             ; 1 posición de bit a la derecha
       SJMP  ST0           ;
; ************************************************************
ST1:   RL    A             ; 1 posición de bit a la izquierda
       SJMP  ST0
; ************************************************************
;           Subprograma TIEMPO, aproximadamente 131 ms
; ------------------------------------------------------------
; 2 bucles DJNZ entrelazados
TIEMPO:  MOV   R7,#000H      ; Contador para bucle exterior
Z01:     MOV   R6,#000H      ; Contador para bucle interior
Z00:     DJNZ  R6,Z00        ; 256*2us (bucle interior)
         DJNZ  R7,Z01        ; 256*bucle interior
         RET
; ------------------------------------------------------------
   END
```

cmc121.asm

```
; ******************************************************
; * Proyecto:      Programa de reloj del 23.09.2005    *
; * Archivo de proyecto: CMC1-2                        *
; * Archivos fuente:     cmc121.asm                    *
; * MC-Tools:            as, p2hex, debugger           *
; * Perfil:              Perfil integrado PSD1_C515C_AS *
; * Recurso MC:          Portpin P1.3                  *
; * Elaborado por:                                     *
; ******************************************************


; ------------Descripción de funcionamiento-----------------
; En el pin de puerto P1.3 de la unidad PSD1-FLASH sale
; una cadencia con T=6µs.
; Por medio del comando de complemento se invierte
; el nivel lógico en el pin de puerto.
; Nivel 1 conecta el LED a DESCON.
; Nivel 0 conecta el LED a CON.
; ----------------------------------------------------------


; -----Instrucciones necesarias para el Ensamblador --------

      CPU     80515       ; Selección de CPU para Ensamblador
      INCLUDE c515c.inc   ; Insertar definiciones de SFR
      SEGMENT code        ; a partir de aquí segmento de
                          ; código de programa
      ORG     0000H       ; Código de programa a partir de 0000H
; ----------------------------------------------------------


; ******************************************************
;       Programa principal: En P1.3 debe salir cadencia
; ******************************************************
START:
      CPL   P1.3          ; Complementar nivel lógico en P1.3

      SJMP   START        ; Programa como bucle sin fin
; ******************************************************

      END
```

cmc122.asm

```
; ***********************************************************
; * Proyecto:         Generador de reloj del 27.09.2005     *
; * Archivo de proyecto: CMC1-2                             *
; * Archivos fuente:     cmc122.asm                         *
; * MC-Tools:            as, p2hex, debugger                *
; * Perfil:              Perfil integrado PSD1_C515C_AS     *
; * Recurso MC:  LED en Portpin P5.7,salida de reloj P1.3   *
; * Elaborado por:    Bernd Bader                           *
; ***********************************************************



; ------------Descripción de funcionamiento-----------------
; Al pin de puerto P5.7 de la unidad PSD1-FLASH se conecta
; un LED libremente elegible de la unidad de LED. En el pin
; de puerto P1.3 sale una cadencia con T = 6 µs. Como programa
; principal se ejecuta un programa de parpadeo para el LED.
; Por medio del comando de complemento se invierte el nivel
; lógico en el pin de puerto.
; Nivel 1 conecta el LED a DESCON.
; Nivel 0 conecta el LED a CON.
; El retardo necesario entre los estados de LED
; se proporciona por medio del subprograma TIEMPO.
; El tiempo de ejecución del SP es de aproximadamente 131 ms.
; El generador de reloj emplea al temporizador 2 en Autoreload
; y Compare
; ---------------------------------------------------------

; ----Instrucciones necesarias para el Ensamblador ---------

      CPU 80515           ; Selección de CPU para Ensamblador
      INCLUDE c515c.inc   ; Insertar definiciones de SFR
      include bitfuncs

;---------------------------------------------------------
; Asignación de valores
;---------------------------------------------------------

reload_valor   equ   -6   ;Ajusta la duración de periodo a 6µs
comp_valor     equ   -3   ;Tiempo para semionda = 3µs

;---------------------------------------------------------

      SEGMENT  code       ; a partir de aquí segmento de código
                          ; de programa
      ORG      0000H      ; Código de programa a partir de 0000H

; ***********************************************************
;          Programa principal: LED debe parpadear
; ***********************************************************
      CALL  pwm_init

START:
      CPL   P5.7        ; Complementar nivel lógico en P5.7
      CALL  TIEMPO      ; SP retardo
```

```
      SJMP  START     ; Programa como bucle sin fin
; ************************************************************


; ************************************************************
;             Subprograma para inicializar
; ************************************************************
pwm_init:
      mov   crch,#hi(reload_wert)  ;Cargar valor Reload (H)
      mov   crcl,#lo(reload_wert)  ;Cargar valor Reload (L)
      mov   cch3,#hi(comp_wert)    ;Cargar valor Compare (H)
      mov   ccl3,#lo(comp_wert)    ;Cargar valor Compar (L)
      mov   th2,#hi(reload_wert)    ;Carg.reg.cont.p 1°periodo
      mov   tl2,#lo(reload_wert)
      mov   ccen,#10000000b        ;Asignar canal Compare 3
      mov   t2con,#00010001b       ;Autoreload y Compare modo 0
      ret


; ************************************************************
;           Subprograma TIEMPO, aprox. 131 ms
; ************************************************************
; 2 bucles DJNZ entrelazados
TIEMPO:
      MOV   R7,#00H   ; Contador para bucle exterior
Z01:  MOV   R6,#00H   ; Contador para bucle interior
Z00:  DJNZ  R6,Z00    ; 256*2µs (bucle interior)
      DJNZ  R7,Z01    ; 256*bucle interior
      RET
; ------------------------------------------------------------

      END
```

cmc123.asm

```
; ********************************************************
; * Proyecto:      Direccionam. de mem. del 27.09.2005   *
; * Archivo de proyecto: CMC1-2                          *
; * Archivos fuente:     cmc123.asm                      *
; * MC-Tools:            as, p2hex, debugger             *
; * Perfil:              Perfil integrado PSD1_C515C_AS  *
; * Recurso MC:          sin                             *
; * Elaborado por:       Bernd Bader                     *
; ********************************************************


; -----------Descripción de funcionamiento-----------------
; Se realizan ejemplos de programa para el direccionamiento de
; memorias de datos y de constantes.
; Los siguientes tipos de direccionamiento son contenido del
; progr. :
; - Direccionamiento directo
; - Direccionamiento indirecto
; - Direccionamiento indirecto con registro de base e índice
; ---------------------------------------------------------


; ********************************************************
; ----Instrucciones necesarias para el Ensamblador ---------

      CPU 80515          ; Selección de CPU para Ensamblador
      INCLUDE c515c.inc  ; Insertar definiciones SFR

; ********************************************************

;                     Programa principal
START:

;                  Memoria de datos interna
; ********************************************************

;              directamente direccionable (DDATA)
;              ----------------------------------
; RAM inferior
;-----------

dir_iRAM:

   mov   30H,#0AAH ;carga dirección (Adr) 30H con el número AAH
   mov   31H,30H   ;carga Adr 31H con el contenido de
                   ;dirección 30H
   mov   R7,#10H   ;carga R7 (Adr 07H) con el número 10H
```

```
;         Registro de funciones especiales (SFR)
;         ---------------------------------------

dir_SFR:

   mov    90H,#55H    ;carga Adr 90H (Port1) en zona SFR
                      ;con el número 55H
   mov    P5,#0AAH    ;carga Port4 (Adr F8H zona SFR)
                      ;con el número AAH


;          indirectamente direccionable (IDATA)
;          ------------------------------------

; RAM inferior
;-------------

indir_iRAM:

   mov    R0,#30H     ;carga registro indicador de dirección R0
                      ;con la dirección de destino 30H
   mov    @R0,#0BBH   ;carga la localización 30H
                      ;direccionada por R0 con el número BBH
   inc    R0          ;aumenta el indicador de dirección  en 1
                      ;R0:= 31H
   mov    @R0,#0BBH   ;carga en la siguiente dirección el
                      ;número BBH


; RAM superior
;-------------

indir_sRAM:

   mov    R0,#90H     ;carga el registro de indicador de
                      ;dirección R0 con la dirección de destino 90H
   mov    @R0,#01H    ;carga la localización 90H direccionada
                      ;por R0 con el número 01H
   mov    R0,#0F8H    ;carga el registro indicador de dirección R0
                      ;con la dirección de destino F8H
   mov    @R0,#01H    ;carga la dirección F8H direccionada por R0
                      ;con el número 01H



;             Memoria de datos externa (XDATA)
; ***********************************************************

;            sólo indirectamente direccionable
;            ---------------------------------

indir_xdata:

   mov    A,#0CCH     ;carga acumulador con el número CCH

   mov    dptr,#0100H    ;carga el datapointer DPTR con la
                         ;dirección de destino de 16 bits 0100H
   movx   @dptr,A        ;carga la dirección de destino 0100H
                         ; con el contenido CCH del acumulador
```

```
;                  Memoria de programa (CODE)
; *********************************************************

; sólo indirectamente direccionable por medio de registro
; base e índice
; ---------------------------------------------------------


   mov   dptr,#0050H   ;carga la dirección de la constante
                       ;en la memoria de programa
   clr   A             ;borra el contenido del acumulador
   movc  A,@A+dptr     ;carga el acumulador con el contenido de
                       ;la localización de memoria de programa
                       ;que es indicada por la suma de
                       ; acumulador y datapointer
   inc   dptr          ;siguiente localización de una constante
   clr   A             ;borra el contenido del acumulador
   movc  A,@A+dptr     ;carga el acumulador con la siguiente
                       ;constante


; *********************************************************
   sjmp  START         ;empieza desde el principio


;   Definición de constantes en la mem. de programa a partir
;   de Adr 0050H
; *********************************************************

   org   0050H
   db   "Hola"

; *********************************************************

   end
```

Nos gusta trabajar con y para usted.

Usted es nuestro mejor colaborador. En base a las experiencias que haya tenido con nuestros equipos, puede contribuir a mejorarlos dándonos sugerencias o avisándonos de cualquier error encontrado. Toda observación que nos haga será tratada con importancia y tomada en cuenta a la hora de actualizar nuestras guías de ejercicios.

¡Muchas gracias por su interés y su colaboración!

.....

Asunto: Guía de ejercicios
Notas: .....

.....

.....

.....

.....

Fecha: .....

LUCAS-NÜLLE Lehr- und Messgeräte GmbH
Dirección: Siemensstr. 2 • D-50170 Kerpen (Sindorf) • Alemania
Apdo. postal: Postfach 11 40 • D-50140 Kerpen • Alemania
Telf.: + 49 / (0)2273 / 567-0 • Fax: +49 / (0)2273 / 567-30 • vertrieb@lucas-nuelle.com

Lucas-Nülle Lehr- und Meßgeräte GmbH

Siemensstraße 2 · D-50170 Kerpen-Sindorf
Telefon +49 2273 567-0 · Fax +49 2273 567-30

www.lucas-nuelle.de

LUCAS-NÜLLE