1 Mitsubishi M16C62 Mikrocontroller

1.1 Überblick

Der M16C62 ist ein leistungsfähiger Mikrocontroller der M16C-Familie von Mitsubishi. Er verfügt über eine grosse Anzahl On Chip Peripherie und je nach Typ über grossen internen Speicherbereich. Der Chip ist direkt im System programmierbar und stellt auch eine Debug-Schnittstelle zur Verfügung. Die frei verfügbare Entwicklungssoftware erlaubt einen kostengünstigen Einstieg in die Programmierung mit Hochsprache C. Durch die RISC-ähnliche Struktur werden 8 MIPS bei 16MHz Takt erreicht. Für batteriebetriebene Geräte ist ein Betrieb bereits ab 3.0V (bei 7MHz) möglich bei 18mW Verbrauch.

Typisch benötigt ein M16C62 (M30624FGAFP) bei 5V, 16MHz, 0 Wait nur 32.5mA Strom. Bei 3V, 10MHz, 0 Wait sogar nur 12mA bei einem M30624FGMFP. Im Stop-Mode nennt Mitsubishi bei 25 °C 1uA Stromverbrauch.

1.2 Technische Daten

Eine Zusammenstellung der wichtigsten Kenndaten der M16C-Mikrocontroller Familie:

Interner Speicher	ROM (0K bis 256 Kbytes) RAM 4 K bis zu 20 Kbytes
Ausführungszeit für eine Instruktion	62.5ns(f(XIN)=16MHz)
Speisespannung	4.0 bis 5.5V (Takt 16MHz) 2.7 bis 5.5V (Takt 7MHz mit software one-wait)
Low Power Verbrauch	18mW (Takt 7MHz mit software one-wait, V_{CC}=3V)
Interrupt	20 Interne und 5 externe Interrupt Quellen, 4 Software Interrupts; 7 Levels.
16-Bit Multifunktions-Timer	8 programmierbare Autoreload Timer
Serielle Schnittstellen	3, UART oder clock-synchron. (Eine wird für SIM-Debugging verwendet.)
DMAC	2 Kanäle
ADC	8 Kanäle a 10 Bit (erweiterbar bis 10 Kanäle)
DAC	2 Kanäle a 8 Bits
CRC	1 CRC Calulation Unit
Watchdog	15-bit Watchdog Timer
Digitale I/O	87
Speichererweiterung	Möglich bis 1MB
Chip Select	4 dekodierte Leitungen
Taktgenerator	2 interne Oszillatoren (32KHz sub clock und 16MHz)

Bild 1-1: Tabelle der wichtigsten Kenndaten der M16C-Familie.

Je nach Ausführung stehen bis zu 256KB ROM/Flash-EPROM und 4 bis 20kB RAM zur Verfügung. Diese Information kann aus der Typennummer gelesen werden. Die gängigen M16C62 mit Flash-EPROM sind:

	Flash EPROM	Speisung	RAM	Sonstiges
100 Pin QFP Gehäuse (0.65mm)				
M30624FGAFP	256KB	5V	20KB	
M30624FGMFP	256KB	3V	20KB	
M306N0FGTFP	256KB	5V	10KB	2x CAN
M30620FCAFP	128KB	5V	10KB	
M30620FCMFP	128KB	3V	10KB	
M3062GF8NFP	64KB	3V	8KB	
100Pin QFP (0.5mm)				
M30624FGAGP	256KB	5V	20KB	
M30624FGMGP	256KB	3V	20KB	
M30620FCAGP	128KB	5V	10KB	
M30620FCMGP	128KB	3V	10KB	
M3062GF8NGP	64KB	3V	8KB	

Bild 1-2: Zusammenstellung der gängigsten M16C62 Controller mit Flash-Memory.

1.3 Blockschaltbild

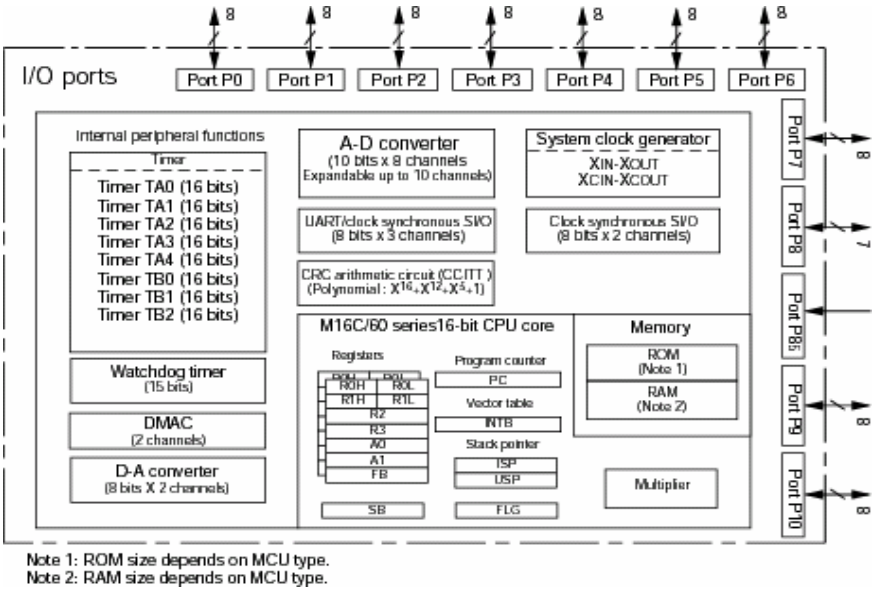


Bild 1-3: Blochschaltbild des M16C62.

1.4 Speicherbereich

Der M16C62 wird normalerweise im Mikrocontrollermodus (RAM/ROM intern) betrieben. Alternativ kann er auch im Prozessormodus gearbeitet werden, d.h.auf RAM/ROM wird extern zugegriffen. In diesem Modus gehen natürlich einige I/O für Adress- und Datenbus verloren. Dies, auch wenn der Speicher extern erweitert wird. Im Maximum sind 1MB adressierbar.

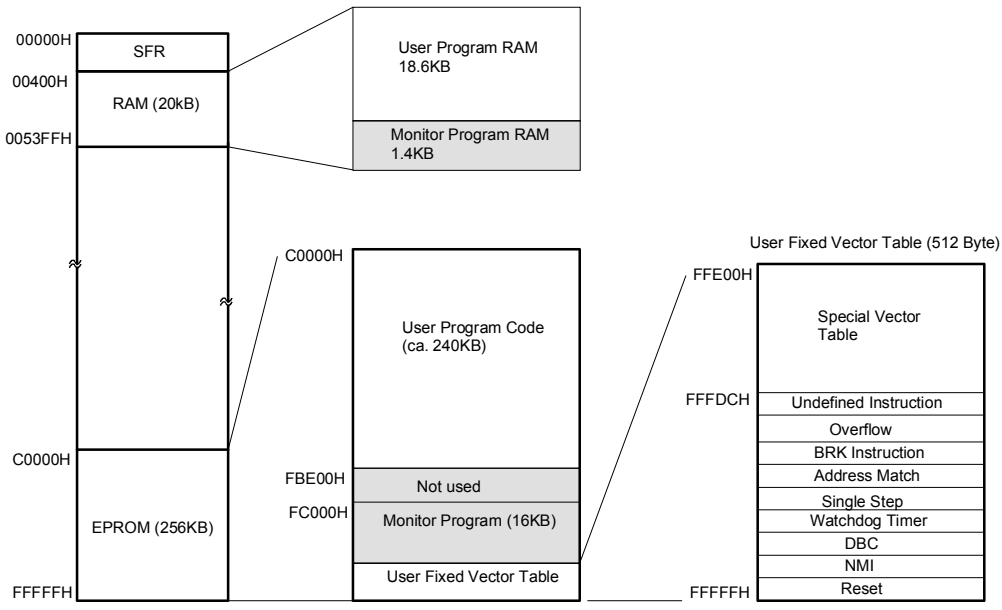


Bild 1-4: Speicherbereich des M16C62.

1.5 Ports

Mitsubishi benutzt beim M16C62 das Memory-Mapped I/O Konzept. Man unterscheidet also nicht Portadressen und Speicheradressen wie bei Intel, sondern Portadressen sind nur spezielle Speicheradressen (SFR, Special Function Register). Zur Ein-/Ausgabe auf Ports werden daher auch keine speziellen Prozessorbefehle benötigt.

2 C-Compiler für Mikrocontroller

Während früher wegen des knappen Speicherplatzes Assembler die einzig sinnvolle Programmiersprache für Mikrocontroller war, erfolgt heute meist die Programmierung in Hochsprache. Die früher nachgesagte schlechte Codeeffizienz einer Hochsprache entbehrt der Grundlage. Ein guter C-Compiler erzeugt den besseren Code als ein durchschnittlicher Assemblerprogrammierer, besonders bei komplexeren Programmierprojekten. Weiter sind Programme in Hochsprache besser zu dokumentieren, besser wartbar und haben kürzere Entwicklungs- und Testzeiten. Trotzdem bleibt bei Hochsprache ein Codeoverhead in der Grössenordnung von 20%-100% gegenüber einer guten Assemblerlösung.

Die Sprache C ist sicher erste Wahl, da sie die Freiheit von Assembler auf dem Niveau der Hochsprache bietet. Jedoch sind auch PASCAL und BASIC für einige Controller erhältlich. Für einige Bausteine sind sogar C++ Compiler erhältlich.

Führende Hersteller sind IAR, Mikrotek und Keil. Gute Produkte sind allerdings recht teuer: Eine vollständige Entwicklungsumgebung für eine MCU mit Editor, Compiler, Assembler/Linker und Debugger/Simulator sind bald im fünfstelligen Bereich.

Neuere Mikrocontroller unterstützen mit Ihrer Architektur und Befehlssatz die Programmierung in Hochsprache. Bei älteren MCU mit knappen Systemressourcen muss ressourcenbewusst programmiert werden. Meist verfügen die Compiler über Spracherweiterungen, so dass die knappen Ressourcen optimal ausgenutzt werden können. Trotzdem ist die Programmierung bei knappem RAM und ROM immer ein Problem.

Die Spracherweiterungen nutzen die speziellen Vorzüge, die die MCU in Befehlssatz und Architektur anbieten:

- Besondere Adressierung (short, long)
- Spezielle Datentypen (Register, Bitvariable)
- Prozessorspezifische Speichermodelle
- Steuerung der Platzierung von Code und Daten (Location)
- Interruptverarbeitung

Weiter beherrschen die meisten Compiler automatische Codeoptimierung. Dabei wird der Code bezüglich Speicherbedarf und Laufzeit optimiert.

3 Mitsubishi M16C62 C-Compiler

3.1 Überblick

Die Firma Mitsubishi stellt für ihre Mikrocontroller der M16-Reihe leistungsfähige Entwicklungswerkzeuge zur Verfügung. Sie können für eine 30-tägige Evaluationszeit kostenlos genutzt werden. Anschliessend bedarf es einer definitiven Lizenzierung oder Neuinstallation.

Die nachfolgenden Informationen sind als Ergänzung zur Dokumentation M16C Mikrontroller-Lernsystem der HTA-BE anzusehen.

Die angebotenen Entwicklungswerkzeuge umfassen:

KD30 Debugger
Dient zum Test der Programme auf dem Evaluationsboard MSA0654. Der Download des Maschinencodes und die Steuerung erfolgt über eine serielle Schnittstelle. Der Debugger ermöglicht recht komfortables Testen auf Niveau der Hochsprache. Bei Echtzeitanwendungen und bei Verwendung von Echtzeitbetriebssystemen werden aber die Grenzen überschritten. Nachteilig auch, dass nur zwei HW-Breakpoints möglich sind.

NC30A C-Compiler
ANSI-C Compiler mit zahlreichen Erweiterungen für die M16C-Familie. Die Laufzeitbibliothek liegt im Quellcode bei, so dass Anpassungen, vor allem für Ein- und Ausgaben problemlos möglich sind.

AS30
Assembler, Linker und Locator für die Erstellung von Assemblerprogrammen. Der Linker und Locator wird auch bei der Programmierung in Hochsprache benötigt.

TM M16C62
Projektmanager zur Verwaltung der Programmierprojekte. Über eine Werkzeugleiste können Editor, Compiler, Debugger, etc. direkt aktiviert werden.

Das Produkt enthält selbst keinen Editor. Man verwendet zweckmässigerweise Textpad oder ein ähnliches Produkt, das konfigurierbar ist und eine Syntaxhervorhebung ermöglicht.

Dokumentation
Alle Handbücher zu den Entwicklungswerkzeugen und Hardware sind auf der CD im PDF-Format vorhanden. Weiter können bei Mitsubishi weitere Dokumente und Updates geladen werden:

http://www.tool-spt.maec.co.jp/index_e.htm
http://www.mitsubishichips.com/products/mcu/products/m16c/index.html

3.2 Installation

Man beginnt mit der Installation des NC30A Compilers. Dabei wird der Compiler selbst, Assembler und die Utilities (Linker, Locator, etc.) installiert. Bei der Frage nach der Lizenznummer wird Evaluation gewählt. Es wird im Start-Folder immer ein Eintrag MISTUBISHI-TOOLS erzeugt. Zur Zeit (9.2001) sind die Versionen aktuell:

NC30A: 4.00
TVM: 3.11
KD30: 2.00.01

Die Programme sind unter Windows98 und Windows2000 lauffähig. NT4 und Windows95 wurden nicht getestet. Probleme wurden bei Laptop-Installation berichtet, die aber frühere Versionen betrafen.

Nach der Installation werden die Files für Standard Startup-Code (`crt0.a30, sect30.inc`) für eine an das Evaluationsboard MSA0654 geänderte Version ausgetauscht. Der neue Startup-Code unterstützt den Debugger und die andere Speicherorganisation. Ein Projekt mit den standardmässig installierten Startup-Files wäre nicht lauffähig.

3.3 Literatur

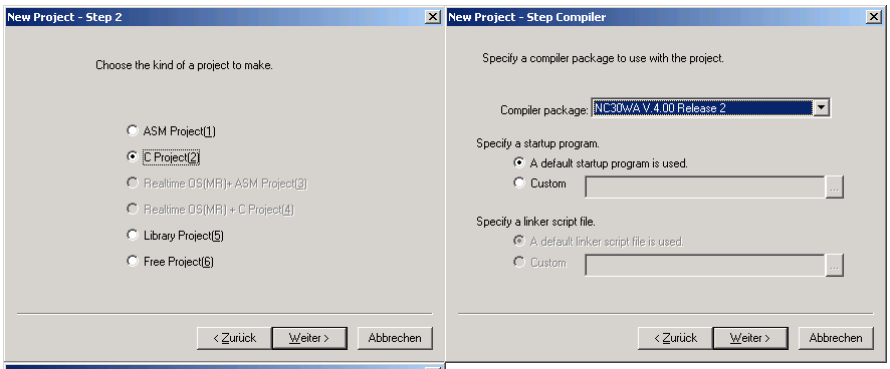
[MAN00] Bernhard Mann, C für Mikrocontroller, Franzis Verlag 2000, 250 Seiten, CD, ISBN 3-7723-4154-3.
Allgemeine Einführung in Mikrocontrollerprogrammierung in der Hochsprache C. schwerpunktmässig auf ATMEL-Controller und Compiler der Firma IAR ausgerichtet, aber inhaltlich durchaus empfehlenswert.

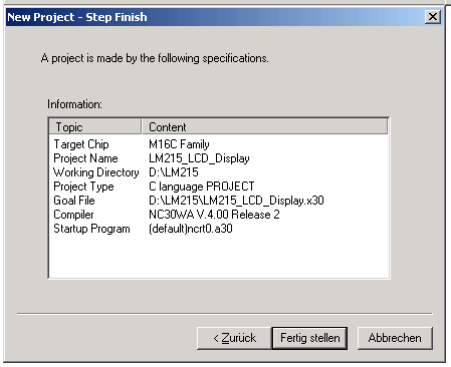
[HTA02] M16C, Das Microcontroller-Lernsystem der HTA-BE. Beschreibung zum M16C62 Evaluationsboard der HTA Bern.

Neues Projekt eröffnen

Den Projektmanager TVM starten. Es erscheint eine Werkzeugleiste, die frei auf der Arbeitsfläche positioniert werden kann.

Im Projektnamen sind keine Leerschläge erlaubt.





Der Quellcode wird mit dem Editor eingegeben. Dazu wird über die Werkzeugleiste der Editor gestartet. Eventuell müssen erstmalig die Editoreinstellungen konfiguriert werden, wenn nicht Notepad benutzt wird.

Nach Kommentarkopf ist direkt das Includefile `sfr62.h` einzubinden. Es enthält Definitionen der On-Chip-Periferie des M16C62.

Zur Arbeit mit dem Evaluationsboard ist eine geänderte Version des Statup-Moduls `NCRTC0.A30` einzubinden. Wird dies nicht gemacht, bleibt der Mikrocontroller beim Initialisieren (Laden der Interuptvektortabelle) hängen.
Der Quellcode wird ins gewünschte Verzeichnis gespeichert. Das wird die Datei im Projektmanagerfenster mit ...

Hinweise

- Benutzte Interruptvektoren müssen im Assemblerfile definiert werden.

-Wird das MSA0654 Board benutzt, muss das angepasste Startup-Modul benutzt werden. (Gilt auch für HTA-BE System)

- In `SECT30.INC` müssen die Debug-Vektoren für das Monitorprogramm bei den Interrupt vektoren für UART1 eingetragen sein.

- Ein Indiz, dass der falsche Startup-Code benutzt wurde ist, dass das Programm in `crt0.a30` beim Laden der Interruptvektortabelle hängen bleibt.

- Sollen Ausgaben über die seriellen Schnittstellen erfolgen, sind die atomaren Ausgabe- funktionen für Zeichenausgabe und –eingabe zu definieren. Alternativ kann man eigene Ausgabefunktionen codieren.

- Sämtlich benutzte Periferie ist zu initialisieren: Parallele Schnittstellen in Richtung und Pegel. serielle Schnittstellen: Baudrate, Modus, Interrupt,..

- Initialisierte Arrays sind nur const möglich. Eventuell müsste der Lader `ncrt0.asm` angepasst werden.

- Der Projektmanager unterstützt keine File- und Pfadnamen mit Leerschlägen.