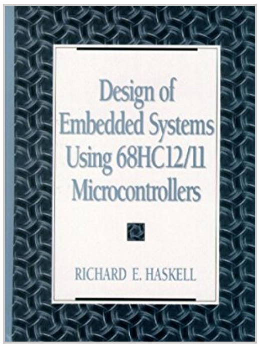

Design of Embedded Systems Using 68HC12/11 Microcontrollers

By Richard E. Haskell

Design of Embedded Systems Using 68HC12/11 Microcontrollers By Richard E. Haskell

This is the first book to describe, in detail, the new Motorola 68HC12 microcontroller, how to program it, and how to design embedded systems using the 68HC12. It shows how WHYP (a version of Forth written specifically for this book) can be used to program the new 68HC12 microcontroller in an efficient and interactive way. It includes an abundance of worked examples and complete C++ code for the WHYP host that runs on the PC. Subroutines and Stacks. 68HC12 Arithmetic. WHYP-An Extensible Language. Branching and Looping. Parallel Interfacing. The Serial Peripheral Interface (SPI). Analog-to-Digital Converter. Timers. The Serial Communications Interface (SCI). Designing with Interrupts. Strings and Number Conversions. Program Control and Data Structures. Fuzzy Control. Special Topics. WHYP12 C++ Classes. WHYP12 C++ Main Program. For electrical and computer engineers who want to learn about the new Motorola 68HC12 microcontroller, how to program it, and how to design embedded systems using it.

Design of Embedded Systems Using 68HC12/11 Microcontrollers

By Richard E. Haskell

Design of Embedded Systems Using 68HC12/11 Microcontrollers By Richard E. Haskell

This is the first book to describe, in detail, the new Motorola 68HC12 microcontroller, how to program it, and how to design embedded systems using the 68HC12. It shows how WHYP (a version of Forth written specifically for this book) can be used to program the new 68HC12 microcontroller in an efficient and interactive way. It includes an abundance of worked examples and complete C++ code for the WHYP host that runs on the PC.   Subroutines and Stacks. 68HC12 Arithmetic. WHYP-An Extensible Language. Branching and Looping. Parallel Interfacing. The Serial Peripheral Interface (SPI). Analog-to-Digital Converter. Timers. The Serial Communications Interface (SCI). Designing with Interrupts. Strings and Number Conversions. Program Control and Data Structures. Fuzzy Control. Special Topics. WHYP12 C++ Classes. WHYP12 C++ Main Program.   For electrical and computer engineers who want to learn about the new Motorola 68HC12 microcontroller, how to program it, and how to design embedded systems using it.

Design of Embedded Systems Using 68HC12/11 Microcontrollers By Richard E. Haskell Bibliography

- Sales Rank: #1458111 in Books
- Published on: 1999-10-25
- Original language: English
- Number of items: 1
- Dimensions: 9.75" h x 1.00" w x 7.50" l, 2.22 pounds
- Binding: Paperback
- 569 pages

Editorial Review

From the Inside Flap
Preface

Many people think of a computer as a PC on a desk with a keyboard and video monitor. However, most of the computers in the world have neither a keyboard nor a video monitor. Rather they are small microcontrollers—a microprocessor, memory, and 1/0 all on a single chip—that are embedded in a myriad of other products such as automobiles, televisions, VCRs, cameras, copy machines, cellular telephones, vending machines, microwave ovens, medical instruments, and hundreds of additional products of all kinds. This book is about how to program microcontrollers and use them in the design of embedded systems.

A popular microcontroller that has been used in a wide variety of different products is the Motorola 68HC11. Motorola has recently introduced an upgrade of this microcontroller, the 68HC12, that has new, more powerful instructions and addressing modes. This book emphasizes the use of the 68HC12 while at the same time providing information about the 68HC11. It can therefore be used in courses that use both 68HC12 and 68HCll microcontrollers.

This book is the result of teaching various microcomputer interfacing courses over the past 20 years. While the technology may change, the basic principles of microcomputer interfacing remain largely the same and these basic principles are stressed throughout this book. However, microcomputer interfacing is a subject that is learned only by doing. The courses that I have taught using this material have all been project-oriented courses in which the students design and build real microcomputer interfacing projects.

A definite trend in microcomputer interfacing and in digital design in general is a shift from hardware design to software design. Microcomputer interfacing has always involved both hardware and software considerations. However, the increasingly large-scale integration of the hardware together with sophisticated software tools for designing hardware means that even traditional hardware design is becoming more and more a software activity.

In the past most software for microcomputer interfacing has been written in assembly language. This means that each time a new and better microprocessor comes out the designer must first learn the new assembly language. The advantage of assembly language is that it is "closest to the hardware" and will allow the user to do exactly what he or she wants in the most efficient manner. While some feel that assembly language programs are more difficult to write and maintain than programs written in a high-level language, the major disadvantage of assembly language programs is related to the obsolescence of the microprocessor—when upgrading to a new or different microprocessor, all of the software has to be rewritten! Even when upgrading from a 68HCI I to a 68HC12, which is upward compatible at the sourcecode level, to get the best performance from the 68HC12 you will need to rewrite the code to use the newer, more powerful instructions and addressing modes.

This has led to a trend of using high-level languages such as C or C+ + for microcomputer interfacing. While this helps to solve the obsolescence problem—much of the same high-level code might be reusable with a new microprocessor—high-level languages come with their own problems. The development environment is not always the most convenient. One has to edit the program, compile it, load it, and then run it to test it on the real hardware. This edit-compile-test cycle can be very time consuming for large programs. Without sophisticated run-time debugging tools the debugging of the program on real hardware can be very frustrating. When designing microcomputer interfaces you would like to be as close to the hardware as possible.

What you would like is a computer language with the advantages of both a high-level language and assembly language, with none of the disadvantages. It would be nice if the language were also interactive so that you could sit at your computer terminal and literally "talk" to the various hardware interfaces. The language should also produce compact code so that you can easily embed the code in PROMS or flash memory for a stand-alone system. While you're at it why not embed the entire language in your target system so that you can develop your program "on-line" and even upgrade the program in the field once the product is delivered. Impossible, you say? In fact, just such a language exists for almost any microprocessor you may want to use. The language is Forth and we will use a derivative of it in this book to illustrate how easy microcomputer interfacing can be.

We will use a unique version of Forth called WHYP (pronounced whip) that is designed for use in embedded systems. WHYP stands for Words to Help You Program. It is a subroutine threaded language which means that WHYP words are just the names of 68HC12(11) subroutines. New WHYP words can be defined simply by stringing previously defined WHYP words together.

A unique feature of Forth—and WHYP—is its simplicity. It is a simple language to learn, to use, and to understand. In fact, in this book we will develop the entire WHYP language from scratch. We will see that WHYP consists of two parts—some 68HC12 subroutines that reside on the target system (typically an evaluation board) and a C+ + program that runs on a PC and communicates with the 68HC12 target system through a serial line. In the process of developing the WHYP subroutines on the target system you will learn 68HC12 assembly language programming. When you finish the book you will also know Forth. Previous knowledge of C + + will be helpful in understanding the C+ + portion of WHYP that resides on the PC. The complete C+ + source code is included on the disk that accompanies this book and in discussed in Chapters 16 and 17. However, these chapters are optional and are not required in order to use WHYP to program the 68HC12.

You will discover that you can develop large software projects using WHYP in a much shorter time than you could develop the same program in either assembly language or C. You might be surprised at the number of industrial embedded systems projects that have been developed in Forth. Many small companies and consultants that use Forth don't talk much about it because many consider it a competitive advantage to be able to develop software in a shorter time than others who program in assembly language or C.

In Chapter 1 you will learn about the architecture of the 68HC12 and how to write a simple assembly language program, assemble it, download it to the target board, and execute it. You will see how to write 68HC12 subroutines in Chapter 2 where you will learn how the system stack works. We will then develop a separate data stack, using the 68HC12 index register, X, as a stack pointer. This data stack will be used throughout the book to pass parameters to and from our 68HC12 subroutines (WHYP words). We will see in Chapter 2 that this makes it possible to access our 68HC12 subroutines interactively, by simply typing the name of the subroutine on the PC keyboard.

In Chapter 3 we will study 68HC12 arithmetic with emphasis on the new 16-bit signed and unsigned multiplication and division instructions available on the 68HC12. We will use these instructions to create WHYP words for all of the arithmetic operations.

The power of WHYP comes from the fact that you can define new WHYP words in terms of previously defined words. This makes WHYP an extensible language in which every time you write a WHYP program you are really extending the language by adding new words to its dictionary. You will learn how to do this in Chapter 4.

In Chapter 5 we will took at the 68HC12 branching and looping instructions and see how we can use them to build some high-level WHYP branching and looping words such as an IF ... ELSE ... THEN construct and a FOR ... NEXT loop. We will also see in this chapter how we can do recursion in WHYP, that is, how we can have a WHYP word call itself.

After the first five chapters you should have a good understanding of the 68HC12 instructions and how they are used to create the WHYP language. The next six chapters will use WHYP as a tool to explore and understand the 1/0 capabilities of the 68HC12 (and 68HC11). The important topic of interrupts is introduced in Chapter 6 and specific examples of using interrupts in conjunction with various 1/0 functions are given in Chapters 7-11.

Parallel interfacing will be discussed in Chapter 7 where examples will be given of interfacing a 6SHC12 to seven-segment displays, hex keypads, and liquid crystal displays. Real-time interrupts are used to program interrupt-driven traffic lights.

Chapter 8 will cover the 68HC12 Serial Peripheral Interface (SPI) where it will be shown how to interface keypads and sevensegment displays using the SPI. The 68HCll and 68HC12 Analog-to-Digital (A/D) converter is described in Chapter 9 where an example is given of the design of a digital compass.

The 68HC12 programmable timer is discussed in Chapter 10 where examples are given of using output compares, input captures, an

From the Back Cover

This is the first book to describe, in detail, the new Motorola 68HC12 microcontroller, how to program it, and how to design embedded systems using the 68HC12. It shows how WHYP (a version of Forth written specifically for this book) can be used to program the new 68HC12 microcontroller in an efficient and interactive way.

FEATURES/BENEFITS

- **A bridge between the 68HC12 and the 68HC11**—Focuses on the 68HC12, but includes material for (and provides software for) the older 68HC11.
- **A new version of Forth—WHYP (Words to Help You Program)**—designed for use in embedded systems

Complete C++ code for the WHYP host that runs on the PC—Provided with the book and described in Chs. 16 and 17. Includes both the C++ program that runs on the PC, and the 68HC12 assembly language program that runs on the target 68HC12 board. Excerpt. © Reprinted by permission. All rights reserved. PrefaceMany people think of a computer as a PC on a desk with a keyboard and video monitor. However, most of the computers in the world have neither a keyboard nor a video monitor. Rather they are small microcontrollers—a microprocessor, memory, and 1/0 all on a single chip—that are embedded in a myriad of other products such as automobiles, televisions, VCRs, cameras, copy machines, cellular telephones, vending machines, microwave ovens, medical instruments, and hundreds of additional products of all kinds. This book is about how to program microcontrollers and use them in the design of embedded systems. A popular microcontroller that has been used in a wide variety of different products is the Motorola 68HC11. Motorola has recently introduced an upgrade of this microcontroller, the 68HC12, that has new, more powerful instructions and addressing modes. This book emphasizes the use of the 68HC12 while at the same time providing information about the 68HC11. It can therefore be used in courses that use both 68HC12 and 68HCll microcontrollers. This book is the result of teaching various microcomputer interfacing courses over the past 20 years. While the technology may change, the basic principles of microcomputer interfacing remain largely the same and these basic principles are stressed throughout this book. However, microcomputer interfacing is a subject that is learned only by doing. The courses that I have taught using this material have all been project-oriented courses in which the students design and build real microcomputer interfacing projects. A definite trend in microcomputer interfacing and in digital design in general is a shift from hardware design to software design. Microcomputer interfacing has always involved both hardware and software considerations. However, the increasingly large-scale integration of the hardware together with sophisticated software tools for designing hardware means that even traditional hardware design is becoming more and more a software activity. In the past most software for microcomputer interfacing has been written in assembly language. This means that each time a new and better microprocessor comes out the designer must first learn the new assembly language. The advantage of assembly language is that it is "closest to the hardware" and will allow the user to do exactly what he or she wants in the most efficient manner. While some feel that assembly language programs are more difficult to write and maintain than programs written in a high-level language, the major disadvantage of assembly language programs is related to the obsolescence of the microprocessor—when upgrading to a new or different microprocessor, all of the software has to be rewritten! Even when upgrading from a 68HCI I to a 68HC12, which is upward compatible at the sourcecode level, to get the best performance from the 68HC12 you will need to rewrite the code to use the newer, more powerful instructions and addressing modes. This has led to a trend of using high-level languages such as C or C+ + for microcomputer interfacing. While this helps to solve the obsolescence problem—much of the same high-level code might be reusable with a new microprocessor—high-level languages come with their own problems. The development environment is not always the most convenient. One has to edit the program, compile it, load it, and then run it to test it on the real hardware. This edit-compile-test cycle can be very time consuming for large programs. Without sophisticated run-time debugging tools the debugging of the program on real hardware can be very frustrating. When designing microcomputer interfaces you would like to be as close to the hardware as possible. What you would like is a computer language with the advantages of both a high-level language and assembly language, with none of the disadvantages. It would be nice if the language were also interactive so that you could sit at your computer terminal and literally "talk" to the various hardware interfaces. The language should also produce compact code so that you can easily embed the code in PROMS or flash memory for a stand-alone system. While you're at it why not embed the entire language in your target system so that you can develop your program "on-line" and even upgrade the program in the field once the product is delivered. Impossible, you say? In fact, just such a language exists for almost any microprocessor you may want to use. The language is Forth and we will use a derivative of it in this book to illustrate how easy microcomputer interfacing can be. We will use a unique version of Forth called *WHYP* (pronounced *whip*) that is designed for use in embedded systems. WHYP stands for *Words* to *Help You Program*. It is a subroutine threaded language which means that WHYP words are just the names of 68HC12(11) subroutines. New WHYP words can be defined simply by stringing previously defined WHYP words together. A unique feature of Forth—and WHYP—is its simplicity. It is a simple language to learn, to use, and to understand. In fact, in this book we will develop the entire WHYP language from scratch. We will see that WHYP consists of two parts—some 68HC12 subroutines that reside on the target system (typically an evaluation board) and a C+ + program that runs on a PC and communicates with the 68HC12 target system through a serial line. In the process of developing the WHYP subroutines on the target system you will learn 68HC12 assembly language programming. When you finish the book you will also know Forth. Previous knowledge of C + + will be helpful in understanding the C+ + portion of WHYP that resides on the PC. The complete C+ + source code is included on the disk that accompanies this book and in discussed in Chapters 16 and 17. However, these chapters are optional and are not required in order to use WHYP to program the 68HC12. You will discover that you can develop large software projects

using WHYP in a much shorter time than you could develop the same program in either assembly language or C. You might be surprised at the number of industrial embedded systems projects that have been developed in Forth. Many small companies and consultants that use Forth don't talk much about it because many consider it a competitive advantage to be able to develop software in a shorter time than others who program in assembly language or C. In Chapter 1 you will learn about the architecture of the 68HC12 and how to write a simple assembly language program, assemble it, download it to the target board, and execute it. You will see how to write 68HC12 subroutines in Chapter 2 where you will learn how the system stack works. We will then develop a separate data stack, using the 68HC12 index register, X, as a stack pointer. This data stack will be used throughout the book to pass parameters to and from our 68HC12 subroutines (WHYP words). We will see in Chapter 2 that this makes it possible to access our 68HC12 subroutines interactively, by simply typing the name of the subroutine on the PC keyboard. In Chapter 3 we will study 68HC12 arithmetic with emphasis on the new 16-bit signed and unsigned multiplication and division instructions available on the 68HC12. We will use these instructions to create WHYP words for all of the arithmetic operations. The power of WHYP comes from the fact that you can define new WHYP words in terms of previously defined words. This makes WHYP an extensible language in which every time you write a WHYP program you are really extending the language by adding new words to its dictionary. You will learn how to do this in Chapter 4. In Chapter 5 we will took at the 68HC12 branching and looping instructions and see how we can use them to build some high-level WHYP branching and looping words such as an *IF ... ELSE ... THEN* construct and a *FOR ... NEXT* loop. We will also see in this chapter how we can do recursion in WHYP, that is, how we can have a WHYP word call itself. After the first five chapters you should have a good understanding of the 68HC12 instructions and how they are used to create the WHYP language. The next six chapters will use WHYP as a tool to explore and understand the 1/0 capabilities of the 68HC12 (and 68HC11). The important topic of interrupts is introduced in Chapter 6 and specific examples of using interrupts in conjunction with various 1/0 functions are given in Chapters 7-11. Parallel interfacing will be discussed in Chapter 7 where examples will be given of interfacing a 6SHC12 to seven-segment displays, hex keypads, and liquid crystal displays. Real-time interrupts are used to program interrupt-driven traffic lights. Chapter 8 will cover the 68HC12 Serial Peripheral Interface (SPI) where it will be shown how to interface keypads and sevensegment displays using the SPI. The 68HCll and 68HC12 Analog-to-Digital (A/D) converter is described in Chapter 9 where an example is given of the design of a digital compass. The 68HC12 programmable timer is discussed in Chapter 10 where examples are given of using output compares, input captures, and the pulse accumulator. Examples of using interrupts include the generation of a pulse train and the measurement of the period of a pulse train. An example of storing hex keypad pressings in a circular queue using interrupts is also included in Chapter 10. As a final example of using interrupts a design is given of a sonar tape measure using the Polaroid ultrasonic transducer. Chapter 11 deals with the Serial Communication Interface (SCI) which is the module used by the 68HC12 to communicate with the PC. Chapters 1-11 provide all the basic material needed to program a 68HC12 microcontroller for most applications. These chapters can form the basis of a one-term projects-oriented capstone design course at the senior/graduate level. The material in Chapters 12 and 13 will be of interest to those who want access to more advanced topics related to programming in WHYP Chapter 12 describes how to convert ASCII number strings to binary numbers and vi... Users Review**From reader reviews:**
Charles Alexander:The knowledge that you get from Design of Embedded Systems Using 68HC12/11 Microcontrollers may be the more deep you rooting the information that hide into the words the more you get interested in reading it. It doesn't mean that this book is hard to be aware of but Design of Embedded Systems Using 68HC12/11 Microcontrollers giving you buzz feeling of reading. The article writer conveys their point in a number of way that can be understood by means of anyone who read it because the author of this publication is well-known enough. This specific book also makes your own vocabulary increase well. Making it easy to understand then can go with you, both in printed or e-book style are available. We advise you for having this specific Design of Embedded Systems Using 68HC12/11 Microcontrollers instantly.
Jesse Reid:Often the book Design of Embedded Systems Using 68HC12/11 Microcontrollers has a lot details

on it. So when you check out this book you can get a lot of advantage. The book was written by the very famous author. Mcdougal makes some research prior to write this book. This particular book very easy to read you can get the point easily after looking over this book.
Lamont Williams:Reading can called imagination hangout, why? Because when you are reading a book particularly book entitled Design of Embedded Systems Using 68HC12/11 Microcontrollers the mind will drift away trough every dimension, wandering in every single aspect that maybe unfamiliar for but surely will end up your mind friends. Imaging each and every word written in a publication then become one form conclusion and explanation this maybe you never get just before. The Design of Embedded Systems Using 68HC12/11 Microcontrollers giving you one more experience more than blown away your brain but also giving you useful info for your better life in this era. So now let us explain to you the relaxing pattern at this point is your body and mind will be pleased when you are finished examining it, like winning a casino game. Do you want to try this extraordinary spending spare time activity?
Glenn Herrera:Design of Embedded Systems Using 68HC12/11 Microcontrollers can be one of your basic books that are good idea. We all recommend that straight away because this publication has good vocabulary that can increase your knowledge in language, easy to understand, bit entertaining but delivering the information. The article writer giving his/her effort to place every word into delight arrangement in writing Design of Embedded Systems Using 68HC12/11 Microcontrollers but doesn't forget the main level, giving the reader the hottest along with based confirm resource facts that maybe you can be among it. This great information can certainly drawn you into brand-new stage of crucial considering.