

C51 Primer

An Introduction To The Use Of The Keil C51 Compiler On The 8051 Family

Edition 3.6 05 October 2003
by

Mike Beach

Editor for Edition 3.6
Chris Hills

Chris Hills
Digitally signed by Chris
Hills
DN: CN = Chris Hills, C =
GB, O = Phaedrus
systems, OU = Phaedrus
Systems
Reason: I am the author
of this document
Location: Tamworth UK
Date: 2003.10.05
19:23:35 +01'00'



Hitex (UK) Ltd.
Tel +44 24 7669 2066
Fax: +44 24 7669 2131
www.hitex.co.uk

© Copyright Hitex (UK) Ltd. 1996, 2002
& Phaedrus Systems 2002,2003
All Rights Reserved.

No Part of this publication may be transmitted, transcribed, stored in a retrieval system, translated into any language,
in any form, by any means without the written permission of Hitex (UK) Ltd.

Contents

0	About The C51 Primer	7
0.1	History.....	8
1	Introduction.....	11
2	Compiler Chain.....	13
3	C51 Basics - The 8051 Architecture	15
3.1	8051 Memory Configurations.....	15
3.1.1	Physical Location Of The Memory Spaces.....	15
3.2	Hardware Memory Models	19
3.2.1	External DATA.....	19
3.2.2	External Code	20
3.2.3	Write to CODE Space	20
3.3	Possible Memory Models	21
3.3.1	ROM Memory Models.....	21
3.3.2	RAM Memory Models	22
3.3.3	Choosing The Best Memory Configuration/Model.....	23
3.3.4	What data goes where?	25
3.4	Setting The Memory Model.....	26
3.5	Local Memory Model Specification.....	27
3.5.1	Overview	27
4	Declaring Variables And Constants	29
4.1	Constants.....	29
4.2	Variables	30
4.2.1	Uninitialised Variables	30
4.2.2	Initialised Variables	31
4.3	Watchdogs With Large Amounts Of Initialised Data.....	32
4.4	C51 Variables.....	33
4.4.1	Variable Types.....	33
4.4.2	Special Function Bits.....	35
4.4.3	Converting Between Types	36
4.4.4	A Non-ISO Approach To Checking Data Type Overflow	37
5	Program Structure And Layout.....	39
5.1	Modular Programming In C51	39
5.2	Accessibility Of Variables In Modular Programs	42
5.3	Building a C51 Modular Program.....	45
5.3.1	The Problem.....	45
5.3.2	Maintainable Inter-Module Links.....	45
5.4	Standard Templates (and Version Control).....	51
5.4.1	Version Control.....	51
5.5	Task Scheduling	52
5.5.1	Applications Overview	52
5.5.2	Simple 8051 multi-task Systems	53
5.5.3	Simple Scheduling - A Partial Solution	55
6	C Language Extensions For 8051 Programming.....	57
6.1	Accessing 8051 On-Chip Peripherals	57
6.2	Interrupts.....	58
6.2.1	The Interrupt Function Type	58
6.2.2	Using C51 With Target Monitor Debuggers.....	58
6.2.3	Coping Interrupt Spacings Other Than 8	59
7	Pointers In C51	61
7.1	Using Pointers And Arrays In C51	61

7.1.1 Pointers In Assembler	61
7.1.2 Pointers In C51	61
7.2 Pointers To Absolute Addresses	63
7.3 Arrays And Pointers - Two Sides Of The Same Coin?	64
7.3.1 Uninitialised Arrays.....	64
7.3.2 Initialised Arrays.....	64
7.3.3 Using Arrays	65
7.3.4 Summary Of Arrays And Pointers	66
7.4 Structures	67
7.4.1 Why Use Structures?.....	67
7.4.2 Arrays Of Structures	68
7.4.3 Initialised Structures.....	69
7.4.4 Placing Structures At Absolute Addresses	69
7.4.5 Pointers To Structures.....	70
7.4.6 Passing Structure Pointers To Functions	70
7.4.7 Structure Pointers To Absolute Addresses	71
7.5 Unions	71
7.6 Generic Pointers.....	72
7.7 Spaced Pointers In C51	74
8 Accessing External Memory Mapped Peripherals	77
8.1 The XBYTE And XWORD Macros	77
8.2 Initialised XDATA Pointers.....	78
8.3 Run Time xdata Pointers	80
8.4 The “volatile” Storage Class	81
8.5 Placing Variables At Specific Locations - The Linker Method.....	81
8.6 Excluding External Data Ranges From Specific Areas.....	83
8.7 -missing ORDER and AT now in C51	83
8.8 Using The _at_and_ORDER_ Controls	84
9 Linking Issues And Stack Placement	85
9.1 Basic Use Of L51 Linker	85
9.2 Stack Placement	86
9.3 Using The Top 128 Bytes of the 8052 RAM.....	86
9.4 L51 Linker Data RAM Overlaying	87
9.4.1 Overlaying Principles	87
9.4.2 Impact Of Overlaying On Program Construction.....	88
9.4.3 Indirect Function Calls With Function Pointers (hazardous).....	88
9.4.4 Indirectly called functions solution.....	91
9.4.5 Function Jump Table Warning (Non-hazardous).....	92
9.4.6 Function Jump Table Warning Solution.....	93
9.4.7 Multiple Call To Segment Warning (Hazardous).....	94
9.4.8 Multiple Call To Segment Solution.....	95
9.4.9 Overlaying Public Variables	96
10 Other C51 Extensions.....	99
10.1 Special Function Bits	99
10.2 Support For 80C517/537 32-bit Maths Unit.....	100
10.2.1 The MDU - How To Use It	100
10.2.2 The 8 Datapointers	100
10.2.3 80C517 - Things To Be Aware Of.....	100
10.3 87C751 Support.....	101
10.3.1 87C751 - Steps To Take	101
10.3.2 Integer Promotion	101
11 Miscellaneous Points	103
11.1 Tying The C Program To The Restart Vector	103
11.2 Intrinsic Functions.....	103
11.3 EA Bit Control #pragma.....	104

11.4 16-Bit sfr Support	104
11.5 Function Level Optimisation	105
11.6 In-Line Functions In C51	105
12 Some C51 Programming Tricks	107
12.1 Accessing R0 etc. directly from C51	107
12.2 Making Use Of Unused Interrupt Sources	107
12.3 Code Memory Device Switching.....	108
12.4 Simulating A Software Reset.....	109
12.5 The Compiler Preprocessor - #define	110
13 C51 Library Functions	111
13.1 Library Function Calling	111
13.2 Memory-Model Specific Libraries.....	111
14 Outputs From C51	113
14.1 Object Files	113
14.2 HEX Files For EPROM Blowing.....	113
14.3 Assembler Output.....	113
15 Assembler Interfacing To C Programs.....	115
15.1 Assembler Function Example	115
15.2 Parameter Passing To Assembler Functions.....	117
15.3 Parameter Passing In Registers.....	117
16 General Things To Be Aware Of	119
16.1	119
16.2	119
16.3	119
16.4	119
16.5	119
16.6	120
16.7 Floating Point Numbers	120
17 Conclusion	121
18 Appendix A	125
19 Appendix B	127
20 Appendix C	139
20.1 Dhrystone	139
20.2 Whetstone	139
20.3 The Sieve of Eratosthenes	140
21 Appendix D	152
22 Appendix E Tile Hill Embedded C Style Guide	157
23 Appendix F A Standard History of C	160
23.1 From K&R to ISO-C99 :- A Standard History of C	161
23.1.1 K&R (1 st Edition) 1978	161
23.1.2 K&R (2 nd edition 1988)	162
23.1.3 ANSI C (1989)	162
23.1.4 ISO-C90 (1990)	162
23.1.5 ISO-C99 ISO/IEC 9899:1999	163
23.1.6 ISO/IEC 9899:1999 TC1 2001	164
23.2 The Future: Back to C. (Why C is not C++)	164
23.3 What to read for Embedded C?	165
24 Appendix G Timers & Delays	169
25 Appendix H Serial Ports and Baud rates	171
26 Appendix J ICE Connect your design	173
27 Appendix K 8051 Instruction set (in Hex order)	175
28 Appendix L References	181
29 Standards	187

0 About The C51 Primer

If you've looked at a few 8051 datasheets, other 8051 books or flicked through the chapters in this guide, you may be left thinking that it is necessary to be an 8051 expert to produce workable programs with C51. Nothing could be further from the truth. It is perfectly possible to write real commercial programs with nothing more than a reasonable knowledge of the ISO C language and some appreciation of hardware.

However, to get the maximum performance from the 8051 family, knowing a few tricks is very useful. This is particularly true if you are working on a very cost-sensitive project where needing a bigger RAM or EPROM can result in an unacceptable cost. After all, if cost was not a consideration, we would all be using 80C166s and 68040s!

Whilst the C51 Primer is really aimed at users of the Keil C51 Compiler, it is applicable in part to compilers such as IAR and Tasking. However, as these compilers do not allow such low-level access and have fewer 8051-specific extensions, they are less likely to be used on projects where getting maximum performance is essential.

This edition of the C51 Primer will use the Keil C51 PK51 package version 6.0.2, released in June 2000.

The C51 Primer Will Help You

- Find your way around the basic 8051 architecture.
- Make a sensible choice of memory model and special things to watch out for.
- Locate things at specific addresses.
- Make best use of structures.
- Use bit-addressable memory.
- Think in terms of chars rather than ints.
- Get the best out of the various pointer types.
- Get a modular structure into programs.
- Access on and off-chip ports and peripherals.
- Deal with interrupts.
- Use registerbanks.
- Deal with the stack.
- Understand RAM overlaying.
- Interface C to assembler code.
- Use some of the special versions.
- Use efficient C.
- Help the optimiser to produce the smallest, fastest code.

The C51 Primer Will Not Help You:

Program in ISO C - get a good reference. Look on the Association of C and C++ Users web site (www.accu.org) where they have independent reviews of several thousand C, C++ and SW Engineering book reviews includinig an embedded section.

NOTE:- Whilst many swear by the Kernighan & Ritchie book it is not really the best book to learn C for embedded use. The K&R book is more of a language definition, it was written over 25 years ago for UNIX programmers. It has now been superseded by the International ISO C standards in 1989 and 1999. The syntax used in the K&R First Edition is now obsolete and should not be used. The K&R 2nd Edition followed the ISO C 1989 standard.

Write portable code - simply use the compiler without using any extensions. **NOTE:-** 100% portable code is difficult to write for the 8051 and will be inefficient. Although C is widely

touted as "portable" the vast majority of embedded applications will never be ported (other than to another, usually more powerful, part in the same family)

Set-up each and every on-chip peripheral on all of the 400 plus different 8051 variants!
Some are, however, covered in the appendices.

This guide should be read in association with a good C reference and is not meant to be a definitive work on the C language. It covers all the Keil 8051-specific language extensions and those areas where the CPU architecture has an impact on coding approach.

0.1 History

The C51 Primer was first concived and written by Mike Beach in 1989 as a guide to both the 8051 and the Keil compiler. Since it's initial publication it has been given away with all Keil C51 compiler sold by Hitex, put on the Hitex BBS and later on the web site www.Hitex.co.uk, www.hitex.de, www.keil.co.uk and numerous others it has become one of the standard texts on the 8051.

Issue I	1991	M Beach	
Issue II	<i>not issued</i>	M Beach	
Issue III	1994	M Beach	Based on Keil C51 V3.02
Issue 3.5	January 2002 (Issued I Draft form only)	Chris Hills	Revised for Keil C51 V6 Major re-write
Issue 3.6	October 2003	Chris Hills	Revised for Keil C51 V7 (and academic year)

One of the main changes since Issue III is the change in C syntax between C51 V4 and C51 V5. The declaration for variables before Version 5 was:

```
code unsigned char name;  
xdata int name;
```

this was changed for version 5 to

```
unsigned char code name;  
int xdata name;
```

bl (banked linker) is now standard

floating point maths improved

The other major visable change is the uVision IDE. The uVuision 1 series IDE was a 16 bit system that ran under Win 3.1 (and 9*, NT) This was available with Version 5 compilers. The current IDE , uVision2, is a wholly 32-bit system that will work with C51 V6 onwards Despite the IDE all the tools remain command line driven. This permits their use, as in the past, with most forms of make, other IDE's and script files.

Disclaimer and contact details

This book has been written by several humans and therefore may have errors and omissions. Should you find any errors and omissions please email the current editor, Chris Hills at chills@hitex.co.uk or chris@phaedsys.org. The first person to report a particular error will receive the highly prized Hitex ASCII Table mug (subject to availability) this excludes Hitex and Keil employees and distributors.

Eur Ing **Chris Hills** BSc (hons) C. Eng, MIEE, FRGS
Technical Specialist
Team 8051
Hitex (UK)

chills@hitex.co.uk
+44 (0)24 7669 2066

or

The author's personal email
chris@phaedsys.org
<http://www.phaedsys.org>
<http://Quest.phaedsys.org>

October 2003



The Quest series at <http://QuEST.phaedsys.org> contains this paper and papers on Embedded C in general, Embedded Debuggers, testing strategy etc.

