µVision3 IDE for Microcontrollers

KEIL SOFTWARE®

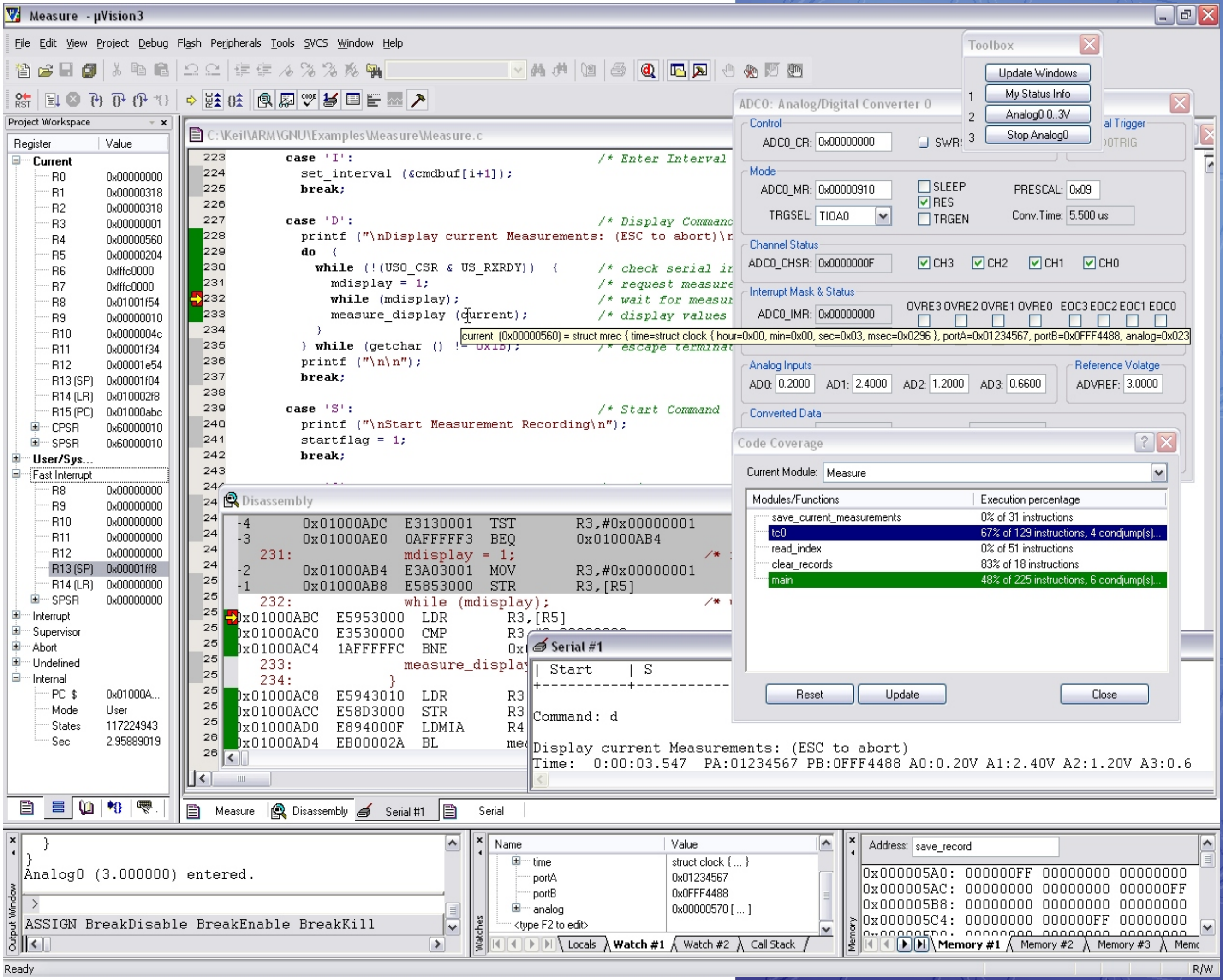
µVISION 3

Quick Start

µVision, the popular IDE from Keil Software, combines Project Management, Source Code Editing, Program Debugging, and Flash Programming in a single, powerful environment. This Quick Start guide gives you the information necessary to use µVision3 for your own projects. It provides an overview of the most commonly used µVision3 features including:

- Project Management, Device Setup, and Tool Configuration.
- Editor facilities for Creating, Modifying, and Correcting Programs.
- Target Debugging or CPU & Peripheral Simulation.

For experienced users, µVision3 adds new features such as Source Outlining, Function Navigation, Editor Templates, Incremental Search, Configuration Wizard, Logic Analyzer, CAN and I²C Simulation, Flash Programming, and JTAG Debugging.



Register	Value
Current	
R0	0x00000000
R1	0x00000318
R2	0x00000318
R3	0x00000001
R4	0x00000560
R5	0x00000204
R6	0xfffc0000
R7	0xfffc0000
R8	0x01001f54
R9	0x00000010
R10	0x0000004c
R11	0x00001f34
R12	0x00001e54
R13 (SP)	0x00001f04
R14 (LR)	0x010002f8
R15 (PC)	0x01000abc
CPSR	0x60000010
SPSR	0x60000010
User/Sys...	
Fast Interrupt	
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00001ff8
R14 (LR)	0x00000000
SPSR	0x00000000
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC $	0x01000A...
Mode	User
States	117224943
Sec	2.95889019

C:\Keil\ARM\GNU\Examples\Measure\Measure.c

```
      case 'I':                                /* Enter Interval
        set_interval (&cmdbuf[i+1]);
        break;

      case 'D':                                /* Display Command
        printf ("\nDisplay current Measurements: (ESC to abort)\r
        do  {
          while (!(US0_CSR & US_RXRDY))  {     /* check serial in
            mdisplay = 1;                      /* request measure
            while (mdisplay);                  /* wait for measu
            measure_display (current);         /* display values
          }
        } while (getchar () != 0x1B);          /* escape terminat
        printf ("\n\n");
        break;

      case 'S':                                /* Start Command
        printf ("\nStart Measurement Recording\n");
        startflag = 1;
        break;
```

```
current (0x00000560) = struct mrec { time=struct clock { hour=0x00, min=0x00, sec=0x03, msec=0x0296 }, portA=0x01234567, portB=0x0FFF4488, analog=0x023
```

```
-4    0x01000ADC  E3130001  TST      R3,#0x00000001
-3    0x01000AE0  0AFFFFF3  BEQ      0x01000AB4
   231:              mdisplay = 1;                     /*
-2    0x01000AB4  E3A03001  MOV      R3,#0x00000001
-1    0x01000AB8  E5853000  STR      R3,[R5]
   232:              while (mdisplay);                 /*
0x01000ABC  E5953000  LDR      R3,[R5]
0x01000AC0  E3530000  CMP      R3
0x01000AC4  1AFFFFFC  BNE      0x
   233:              measure_display
   234:           }
0x01000AC8  E5943010  LDR      R3
0x01000ACC  E58D3000  STR      R3
0x01000AD0  E894000F  LDMIA    R4
0x01000AD4  EB00002A  BL       me
```

ADC0: Analog/Digital Converter 0

Field	Value
ADC0_CR	0x00000000
ADC0_MR	0x00000910
TRGSEL	TIOA0
PRESCAL	0x09
Conv. Time	5.500 us
ADC0_CHSR	0x0000000F
ADC0_IMR	0x00000000
AD0	0.2000
AD1	2.4000
AD2	1.2000
AD3	0.6600
ADVREF	3.0000

SWR; SLEEP; [✓ RES]; TRGEN; [✓ CH3] [✓ CH2] [✓ CH1] [✓ CH0]; OVRE3 OVRE2 OVRE1 OVRE0 EOC3 EOC2 EOC1 EOC0

Code Coverage — Current Module: Measure

Modules/Functions	Execution percentage
save_current_measurements	0% of 31 instructions
tc0	67% of 129 instructions, 4 condjump(s)...
read_index	0% of 51 instructions
clear_records	83% of 18 instructions
main	48% of 225 instructions, 6 condjump(s)...

```
| Start      | S
+------------+---------

Command: d

Display current Measurements: (ESC to abort)
Time:  0:00:03.547  PA:01234567 PB:0FFF4488 A0:0.20V A1:2.40V A2:1.20V A3:0.6
```

```
  }
}
Analog0 (3.000000) entered.

>
ASSIGN BreakDisable BreakEnable BreakKill
```

Name	Value
time	struct clock { ... }
portA	0x01234567
portB	0x0FFF4488
analog	0x00000570 [...]
<type F2 to edit>	

Address: save_record

```
0x000005A0:  000000FF  00000000  00000000
0x000005AC:  00000000  00000000  000000FF
0x000005B8:  00000000  00000000  00000000
0x000005C4:  00000000  000000FF  00000000
```

The Keil µVision3 IDE supports over 700 devices including ARM-powered, XC16x, C16x, ST10, 251, and 8051-based microcontrollers.

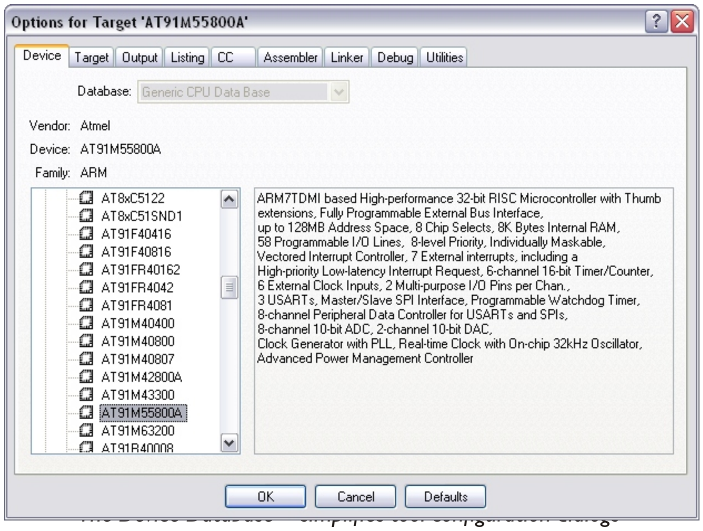
www.keil.com

Project Management

µVision3 ensures easy and consistent Project Management. A single project file stores source file names and saves configuration information for Compiler, Assembler, Linker, Debugger, Flash Loader, and other utilities. The **Project** menu provides access to project files and dialogs for project management.

Open Project... loads existing project files. The **Examples** folder contains many useful sample applications like the **Measure** example that is used throughout this Quick Start guide.

New Project... allows you to create your own projects.


Options for Target 'AT91M55800A'

Database: Generic CPU Data Base

Vendor: Atmel
Device: AT91M55800A
Family: ARM

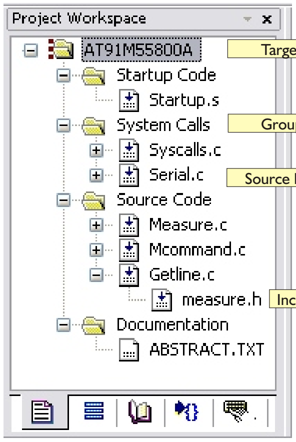
ARM7TDMI based High-performance 32-bit RISC Microcontroller with Thumb extensions, Fully Programmable External Bus Interface, up to 128MB Address Space, 8 Chip Selects, 8K Bytes Internal RAM, 58 Programmable I/O Lines, 8-level Priority, Individually Maskable, Vectored Interrupt Controller, 7 External interrupts, including a High-priority Low-latency Interrupt Request, 6-channel 16-bit Timer/Counter, 6 External Clock Inputs, 2 Multi-purpose I/O Pins per Chan., 3 USARTs, Master/Slave SPI Interface, Programmable Watchdog Timer, 8-channel Peripheral Data Controller for USARTs and SPIs, 8-channel 10-bit ADC, 2-channel 10-bit DAC, Clock Generator with PLL, Real-time Clock with On-chip 32kHz Oscillator, Advanced Power Management Controller

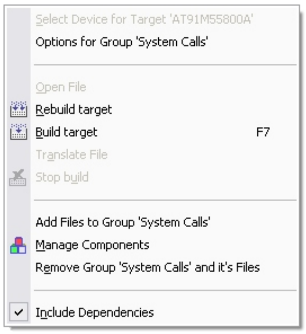
and provides the basic setup for the development environment.

Device Database™

When you start a new µVision3 project, you select the target device from the Device Database™ and µVision3 automatically sets required tool options and customizes dialogs.

µVision3 displays only those options that are relevant to the selected device and prevents selection of incompatible directives. Just a few dialogs are required to completely configure all the tools (Assembler, Compiler, Linker, Debugger, and Flash Download Utilities).



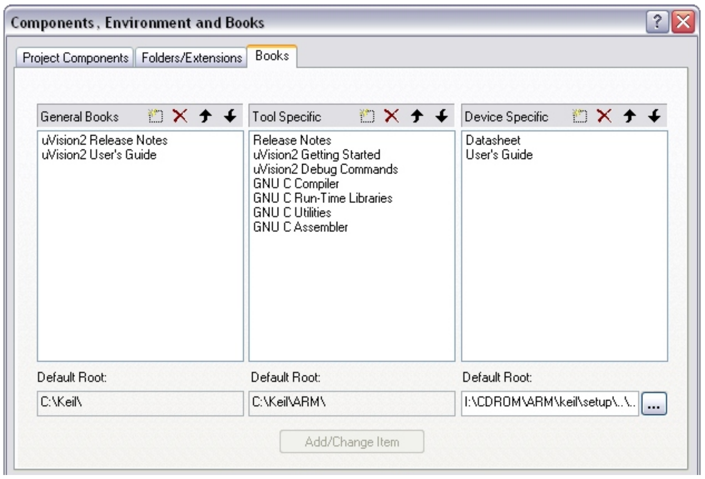

Select Device for Target 'AT91M55800A'
Options for Group 'System Calls'
Open File
Rebuild target
Build target F7
Translate File
Stop build
Add Files to Group 'System Calls'
Manage Components
Remove Group 'System Calls' and it's Files
Include Dependencies

*The **Files Tab** in the Project Workspace lists source & header files. Right-click the mouse to open context-sensitive menus.*

Project Components

File Groups allow you to group associated files together. They may be used to separate files into functional blocks or to identify engineers in your software team.

Project Targets allow you to create several programs from a single project. You may require one target for testing and another target for a release version of your application. Each target allows individual tool settings within the same project file.


Components, Environment and Books

General Books	Tool Specific	Device Specific
uVision2 Release Notes	Release Notes	Datasheet
uVision2 User's Guide	uVision2 Getting Started	User's Guide
	uVision2 Debug Commands	
	GNU C Compiler	
	GNU C Run-Time Libraries	
	GNU C Utilities	
	GNU C Assembler	
Default Root:	Default Root:	Default Root:
C:\Keil\	C:\Keil\ARM\	I:\CDROM\ARM\keil\setup\..\..

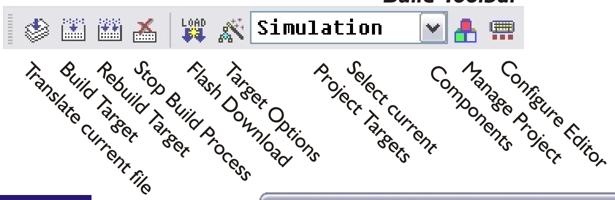
The Manage Components dialog allows you to create File Groups and Project Targets and add documents (Books) to a project.

User's Guides & Additional Books

The **Books Tab** in the Project Workspace lists extensive on-line manuals for the tools and the selected microcontroller. You may add your own additional books using the **Manage Components** dialog. This provides quick access to CPU and project documentation.

Tool Configuration & Make

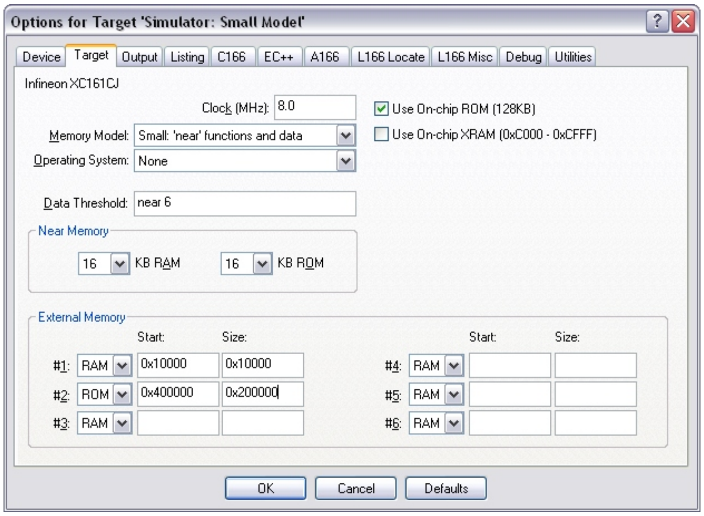
Build Toolbar



Development Tool Options

µVision3 allows you to set tool options by target, group, and source file. The options dialog opens via the context menu of the Project Workspace and provides tabs for:

- **Device**: Allows you to view and change the selected target CPU.
- **Target**: Allows you to specify CPU and memory options. These are used to configure basic tool settings including those of the linker, debugger, and simulator. Typically, these are the only settings you must make to begin a new project.
- **Output/Listing**: Allow you to configure the generated object and listing files.
- **Compiler/Assembler/Linker**: Allow you to view and change tool-specific options.
- **Debug**: Allows you to configure the µVision3 Simulator and Debugger.
- **Utilities**: Allows you to configure project-specific tools (like Flash programming).



Options for Target 'Simulator: Small Model'

Device | Target | Output | Listing | C166 | EC++ | A166 | L166 Locate | L166 Misc | Debug | Utilities

Infineon XC161CJ

Clock (MHz): 8.0
[✓] Use On-chip ROM (128KB)
Memory Model: Small: 'near' functions and data
[ ] Use On-chip XRAM (0xC000 - 0xCFFF)
Operating System: None
Data Threshold: near 6

Near Memory: 16 KB RAM, 16 KB ROM

External Memory

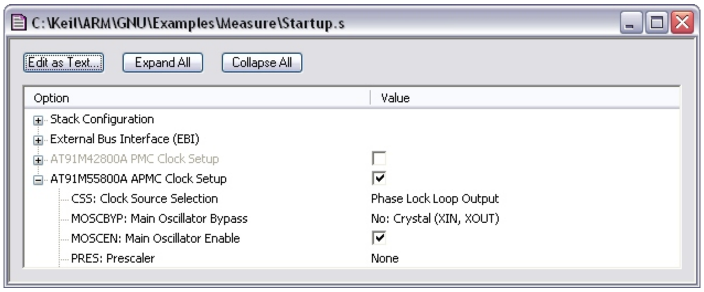
		Start:	Size:			Start:	Size:
#1:	RAM	0x10000	0x10000	#4:	RAM		
#2:	ROM	0x400000	0x200000	#5:	RAM		
#3:	RAM			#6:	RAM		

OK | Cancel | Defaults

The Device Database™ pre-configures tool settings for the selected CPU. Only few dialogs pages are required for complete tool setup.

Build Process

The **Build Target** command runs the Compiler and Assembler. The tools automatically generate file dependencies so only those files that have changed are re-translated. You may enabled additional Global Code Optimizations which are performed by incremental re-compilations of C modules.



C:\Keil\ARM\GNU\Examples\Measure\Startup.s

Edit as Text... | Expand All | Collapse All

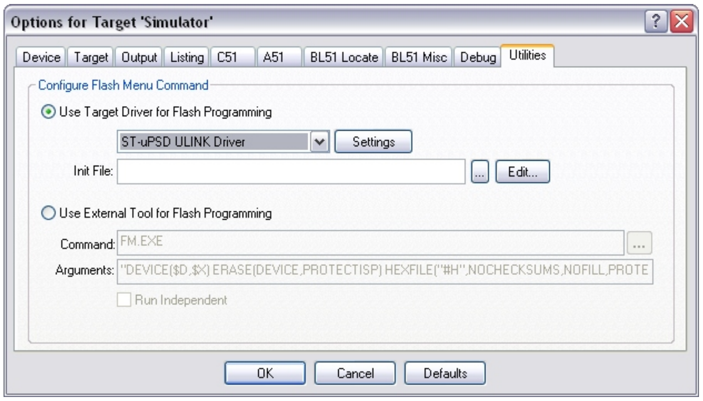
Option	Value
Stack Configuration	
External Bus Interface (EBI)	
AT91M42800A PMC Clock Setup	[ ]
AT91M55800A APMC Clock Setup	[✓]
CSS: Clock Source Selection	Phase Lock Loop Output
MOSCBYP: Main Oscillator Bypass	No: Crystal (XIN, XOUT)
MOSCEN: Main Oscillator Enable	[✓]
PRES: Prescaler	None

The Configuration Wizard provides a graphical editor for startup files, configuration code, debugger INI files, and linker scripts.

Flash Download

Flash Download features are included in some Target Drivers (like the **EPM900** LPC Emulator/Programmer and the Keil **ULINK** USB JTAG-Adapter).

Additionally, you may integrate external Flash utilities (and their command-line options) into the µVision3 project setup.



Options for Target 'Simulator'

Device | Target | Output | Listing | C51 | A51 | BL51 Locate | BL51 Misc | Debug | Utilities

Configure Flash Menu Command

(●) Use Target Driver for Flash Programming
ST-uPSD ULINK Driver | Settings
Init File: | ... | Edit...

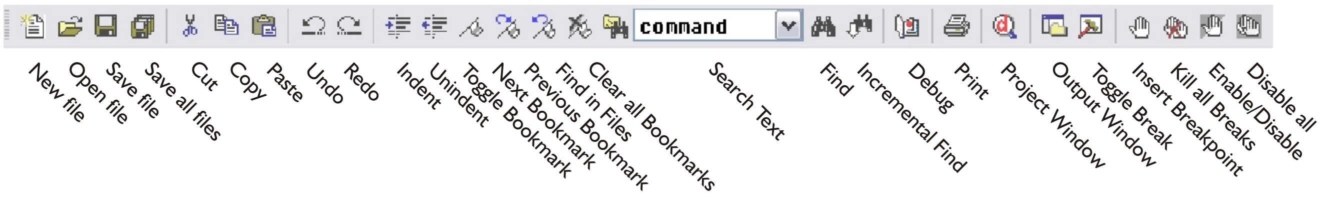
( ) Use External Tool for Flash Programming
Command: FM.EXE
Arguments: "DEVICE($D,$X) ERASE(DEVICE,PROTECTISP) HEXFILE("#H",NOCHECKSUMS,NOFILL,PROTE
[ ] Run Independent

OK | Cancel | Defaults

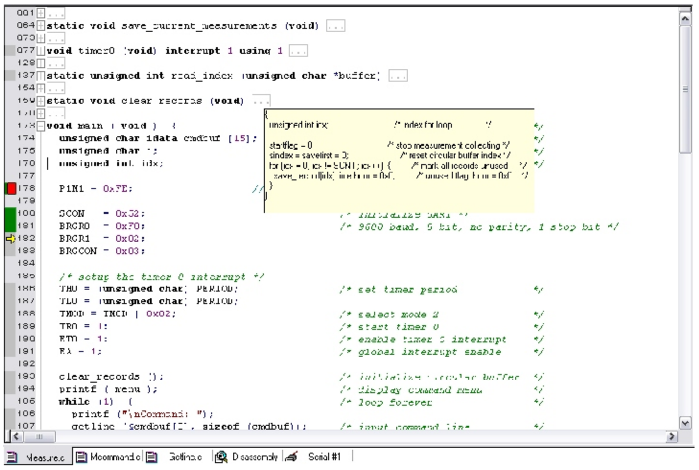
The Utilities tab adds Flash Download Utilities configuration to the project.

Source Code Editor

Editor Toolbar



The µVision3 editor offers many standard and advanced software editing features. Color syntax highlighting and text indentation are optimized for editing C source code. Most editor functions may be quickly accessed from the toolbar or the context menu. Program debugging is directly controlled in the editor. This natural environment allows you to quickly integrate and test source code changes.



```
...
static void save_current_measurements (void) ...
...
void timer0 (void) interrupt 1 using 1 ...
...
static unsigned int read_index (unsigned char *buffer) ...
...
static void clear_records (void) ...
...
void main ( void )  {
  unsigned char idata cmdbuf [15];
  unsigned char i;
  unsigned int idx;

  P1M1 = 0xFE;                                //

  SCON   = 0x52;                               /* initialize UART */
  BRGR0  = 0x70;                               /* 9600 baud, 8 bit, no parity, 1 stop bit */
  BRGR1  = 0x02;
  BRGCON = 0x03;

  /* setup the timer 0 interrupt */
  TH0 = (unsigned char) PERIOD;                /* set timer period            */
  TL0 = (unsigned char) PERIOD;
  TMOD = TMOD | 0x02;                          /* select mode 2               */
  TR0 = 1;                                     /* start timer 0               */
  ET0 = 1;                                     /* enable timer 0 interrupt    */
  EA = 1;                                      /* global interrupt enable     */

  clear_records ();                            /* initialize circular buffer  */
  printf ( menu );                             /* display command menu        */
  while (1)  {                                 /* loop forever                */
    printf ("\nCommand: ");
    getline (&cmdbuf[0], sizeof (cmdbuf));     /* input command line          */
```

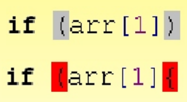
```
{
unsigned int idx;                      /* index for loop          */

startflag = 0                          /* stop measurement collecting */
sindex = savefirst = 0;                /* reset circular buffer index */
for (idx = 0; idx != SCNT; idx++) {    /* mark all records unused */
  save_record[idx].time.hour = 0xff;   /* unused flag: hour = 0xff */
}
}
```

µVision3 includes all standard editor features. Advanced Features provide an overview of source files and help you complete your task faster.

Advanced Features

µVision3 adds many new editor features like:

- **Incremental Find** positions the cursor while you type the search phrase.
- **Active Brace Checking** shows nesting and highlights mismatches while entering parentheses, braces, or brackets.
- **Text Block Functions** that Comment, Indent, Uppercase, Tabify, and Remove Whitespace from text blocks.
- **Document-Selective Settings** for tab spacing and syntax coloring in Assembler, C, and other file types.
- Detailed **Syntax Highlighting** that allows you to define a user keyword list. Colors are used in printed output.
- Document **Outlining** that provides a quick overview of complex source files.



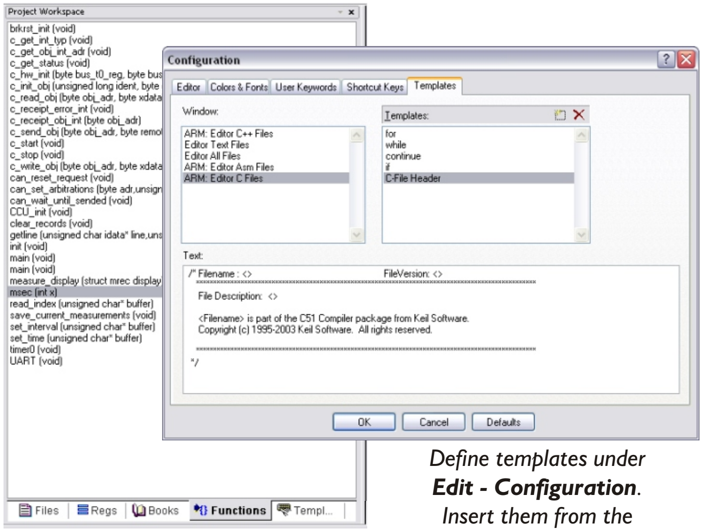
```
if (arr[1])
if (arr[1]{
```

***Active Brace Checking** provides an overview of complex statements. When editing source files, matching parentheses, braces, or brackets are marked in grey—mismatches are shown in red.*

Functions and Templates

The **Functions Tab** in the Project Workspace displays all functions in either your project or open editor files. You may click on a function to jump to its definition.

The **Templates Tab** shows user-defined text blocks (templates). To insert a template into your source file, type the first few letters of the name followed by **Ctrl+Space** or click the item in the Templates Tab.



Project Workspace

brkrst_init (void)
c_get_int_typ (void)
c_get_obj_int_adr (void)
c_get_status (void)
c_hw_init (byte bus_t0_reg, byte bus
c_init_obj (unsigned long ident, byte
c_read_obj (byte obj_adr, byte xdata
c_receipt_error_int (void)
c_receipt_obj_int (byte obj_adr)
c_send_obj (byte obj_adr, byte remo
c_start (void)
c_stop (void)
c_write_obj (byte obj_adr, byte xdata
can_reset_request (void)
can_set_arbitrations (byte adr,unsign
can_wait_until_sended (void)
CCU_init (void)
clear_records (void)
getline (unsigned char idata* line,uns
init (void)
main (void)
main (void)
measure_display (struct mrec display
msec (int x)
read_index (unsigned char* buffer)
save_current_measurements (void)
set_interval (unsigned char* buffer)
set_time (unsigned char* buffer)
timer0 (void)
UART (void)

Window:

ARM: Editor C++ Files
Editor Text Files
Editor All Files
ARM: Editor Asm Files
ARM: Editor C Files

Templates:

for
while
continue
if
C-File Header

Text:

```
/* Filename : <>                          FileVersion: <>
 ******************************************************************************
   File Description:  <>

   <Filename> is part of the C51 Compiler package from Keil Software.
   Copyright (c) 1995-2003 Keil Software.  All rights reserved.

 ******************************************************************************
*/
```

*The **Function Tab** alphabetically lists functions in your project. Click to jump to the function definition.*

*Define templates under **Edit - Configuration**. Insert them from the **Templates Tab** or by entering the first few letters of the name followed by **Ctrl+Space**.*

Editor Utilities

µVision3 contains powerful features and interfaces to external tools which help you complete your projects on time.

Source Browser

The integrated Source Browser (available for Keil Cx51, C251, and C166) maintains a database of your program symbols for quick navigation through your source files. You may use the detailed information to optimize the memory types of variables.

PC-Lint Syntax Checking

Gimpel Software's PC-Lint integrates into the µVision3 IDE and is independent of the Compiler. PC-Lint checks your C/C++ source code for bugs, glitches, inconsistencies, and non-portable constructs. PC-Lint Version 8 also includes checking for MISRA rules.

Version Control Systems

A configurable interface allows you to plug-in the Software Version Control System (SVCS) of your choice. Pre-configured templates are provided for: Intersolv PVCS, Microsoft SourceSafe, MKS Source Integrity, Rational Clear Case, Perforce, and CVS (available under the GNU open source license).

Remote Control Interface

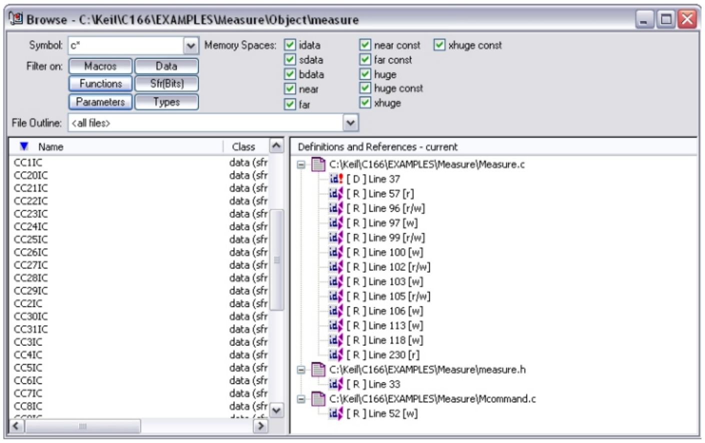
µVision3 may be controlled via the Development Tool Connection (DTC) Interface. DTC is used by third party tools like the iLogix/WST UML Bridge, BKR EasyCode, or Hitex Tessy to remotely control IDE and Debugger functions.

Find in Files

Find in Files performs a text search in all specified files. The search results are displayed in the Find in Files page of the Output window.

External Tools

The Tools menu integrates your favorite utility programs into µVision3. Project-related arguments (like file names or project folders) may be sent as arguments to these utilities.



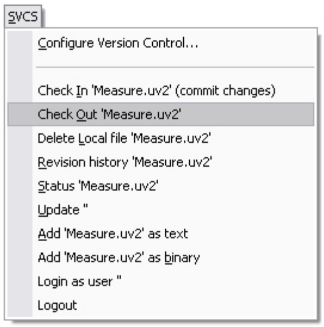
Browse - C:\Keil\C166\EXAMPLES\Measure\Object\measure

Symbol: c*
Filter on: Macros, Data, Functions, Sfr(Bits), Parameters, Types
Memory Spaces: ✓ idata, ✓ sdata, ✓ bdata, ✓ near, ✓ far, ✓ near const, ✓ far const, ✓ huge, ✓ huge const, ✓ xhuge, ✓ xhuge const
File Outline: <all files>

Name	Class
CC1IC	data (sfr
CC20IC	data (sfr
CC21IC	data (sfr
CC22IC	data (sfr
CC23IC	data (sfr
CC24IC	data (sfr
CC25IC	data (sfr
CC26IC	data (sfr
CC27IC	data (sfr
CC28IC	data (sfr
CC29IC	data (sfr
CC2IC	data (sfr
CC30IC	data (sfr
CC31IC	data (sfr
CC3IC	data (sfr
CC4IC	data (sfr
CC5IC	data (sfr
CC6IC	data (sfr
CC7IC	data (sfr
CC8IC	data (sfr

```
Definitions and References - current
C:\Keil\C166\EXAMPLES\Measure\Measure.c
    [ D ] Line 37
    [ R ] Line 57 [r]
    [ R ] Line 96 [r/w]
    [ R ] Line 97 [w]
    [ R ] Line 99 [r/w]
    [ R ] Line 100 [w]
    [ R ] Line 102 [r/w]
    [ R ] Line 103 [w]
    [ R ] Line 105 [r/w]
    [ R ] Line 106 [w]
    [ R ] Line 113 [w]
    [ R ] Line 118 [w]
    [ R ] Line 230 [r]
C:\Keil\C166\EXAMPLES\Measure\measure.h
    [ R ] Line 33
C:\Keil\C166\EXAMPLES\Measure\Mcommand.c
    [ R ] Line 52 [w]
```

Use the Source Browser for fast navigation to definitions and references.


SVCS
Configure Version Control...
Check In 'Measure.uv2' (commit changes)
Check Out 'Measure.uv2'
Delete Local file 'Measure.uv2'
Revision history 'Measure.uv2'
Status 'Measure.uv2'
Update ''
Add 'Measure.uv2' as text
Add 'Measure.uv2' as binary
Login as user ''
Logout

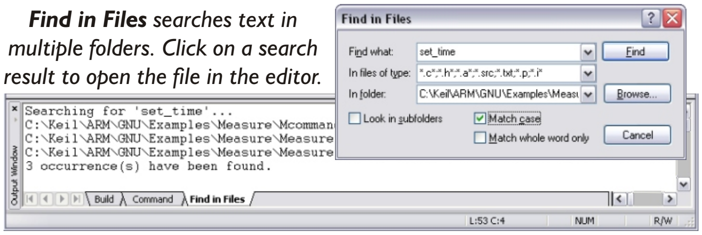
Project Workspace Icons

	Standard Source File
	File or File Group with specific Options
	File checked in SVCS (R/O)
	File excluded from linkage

Key Sequences are used for argument passing to external tools, like SVCS. Prefixes (%, $, @, ...) specify which part of the path to use.

Key	Passes the following item to external tools...
E	Current Editor File.
F	Selected file in the Project Workspace (MEASURE.C).
P	Name of the current project file (MY_APP.UV2).
L	Linker output file, typically the executable file (MY_APP).
H	Application HEX file (MY_APP.HEX).
X	µVision3 executable program file (C:\KEIL\UV2\UV2.EXE).
Q	File name that holds comments for the SVCS system.
R	String that holds a revision number for the SVCS system.
C	String that holds a check point string for the SVCS system.
U	User name specified for the SVCS system.
V	Database file name specified for the SVCS system.

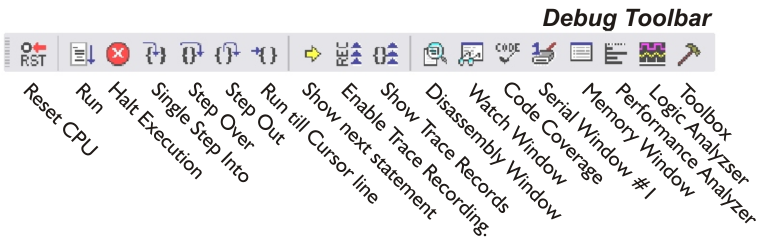
***Find in Files** searches text in multiple folders. Click on a search result to open the file in the editor.*



Find in Files
Find what: set_time
In files of type: *.c*;*.h*;*.a*;*.src;*.txt;*.p;*.i*
In folder: C:\Keil\ARM\GNU\Examples\Measu
☐ Look in subfolders ✓ Match case ☐ Match whole word only

```
Searching for 'set_time'...
C:\Keil\ARM\GNU\Examples\Measure\Mcomman
C:\Keil\ARM\GNU\Examples\Measure\Measure
C:\Keil\ARM\GNU\Examples\Measure\Measure
3 occurrence(s) have been found.
```

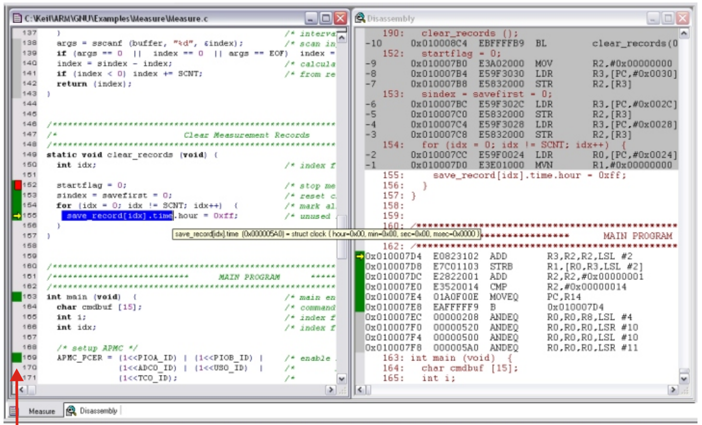
Debugger



The µVision3 IDE/Simulator/Debugger accelerates your learning efforts by providing a single environment for editing, simulating, and testing target hardware. Most debugger and editor functions may be quickly accessed from the toolbar.

You may use the context menu or the Editor Toolbar to insert breakpoints. Breakpoints you set while editing are activated when you start your debugging session.

µVision3 marks the status of each source line in the **Attributes** column of the editor window. This provides a quick overview of the current breakpoint and execution status.



C:\Keil\ARM\GNU\Examples\Measure\Measure.c

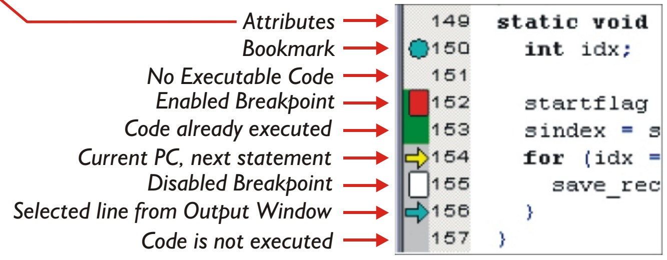
```
  }                                                         /* interva
  args = sscanf (buffer, "%d", &index);                     /* scan in
  if (args == 0  ||  index == 0  || args == EOF)  index =
  index = sindex - index;                                   /* calcula
  if (index < 0) index += SCNT;                             /* from re
  return (index);
}


/*************************************************************
/*                          Clear Measurement Records
/*************************************************************
static void clear_records (void) {
  int idx;                                                  /* index f

  startflag = 0;                                            /* stop me
  sindex = savefirst = 0;                                   /* reset c
  for (idx = 0; idx != SCNT; idx++)  {                      /* mark al
    save_record[idx].time.hour = 0xff;                      /* unused
  }
}


/*************************************************************
/***************************      MAIN PROGRAM      *****
/*************************************************************
int main (void)  {                                          /* main en
  char cmdbuf [15];                                         /* command
  int i;                                                    /* index f
  int idx;                                                  /* index f

  /* setup APMC */
  APMC_PCER = (1<<PIOA_ID) | (1<<PIOB_ID) |                 /* enable
              (1<<ADC0_ID) | (1<<US0_ID)  |                 /*
              (1<<TC0_ID);                                  /*
```

save_record[idx].time (0x000005A0) = struct clock { hour=0x00, min=0x00, sec=0x00, msec=0x0000 }

Disassembly

```
   190:   clear_records ();
-10        0x010008C4  EBFFFFB9  BL        clear_records(0
   152:   startflag = 0;
-9         0x010007B0  E3A02000  MOV       R2,#0x00000000
-8         0x010007B4  E59F3030  LDR       R3,[PC,#0x0030]
-7         0x010007B8  E5832000  STR       R2,[R3]
   153:   sindex = savefirst = 0;
-6         0x010007BC  E59F302C  LDR       R3,[PC,#0x002C]
-5         0x010007C0  E5832000  STR       R2,[R3]
-4         0x010007C4  E59F3028  LDR       R3,[PC,#0x0028]
-3         0x010007C8  E5832000  STR       R2,[R3]
   154:   for (idx = 0; idx != SCNT; idx++)  {
-2         0x010007CC  E59F0024  LDR       R0,[PC,#0x0024]
-1         0x010007D0  E3E01000  MVN       R1,#0x00000000
   155:       save_record[idx].time.hour = 0xff;
   156:   }
   157: }
   158:
   159:
   160: /*************************************************
   161: /*****************************        MAIN PROGRAM
   162: /*************************************************
0x010007D4  E0823102  ADD       R3,R2,R2,LSL #2
0x010007D8  E7C01103  STRB      R1,[R0,R3,LSL #2]
0x010007DC  E2822001  ADD       R2,R2,#0x00000001
0x010007E0  E3520014  CMP       R2,#0x00000014
0x010007E4  01A0F00E  MOVEQ     PC,R14
0x010007E8  EAFFFFF9  B         0x010007D4
0x010007EC  00000208  ANDEQ     R0,R0,R8,LSL #4
0x010007F0  00000520  ANDEQ     R0,R0,R0,LSR #10
0x010007F4  00000500  ANDEQ     R0,R0,R0,LSL #10
0x010007F8  000005A0  ANDEQ     R0,R0,R0,LSR #11
   163: int main (void)  {
   164:   char cmdbuf [15];
   165:   int i;
```

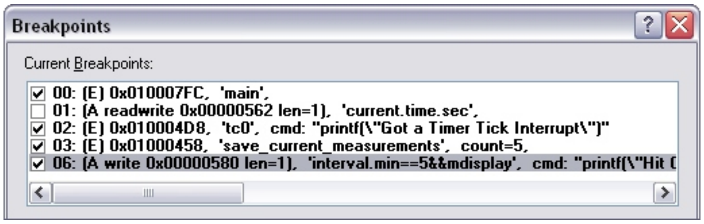
The Disassembly Window includes a trace history that shows the last CPU instructions executed prior to a breakpoint.



Complex Breakpoints

In addition to simple breakpoints, the µVision3 Debugger supports complex breakpoints (with conditional or logical expressions) and memory access breakpoints (with read and/or write access from an address or range).

Complex breakpoints may be defined in the **Breakpoint** dialog (available from the Debug menu). For each breakpoint, you may include a command to execute and a count before the breakpoint is triggered. Commands which are executed when the breakpoint is triggered may even invoke debug functions.



Breakpoints

Current Breakpoints:

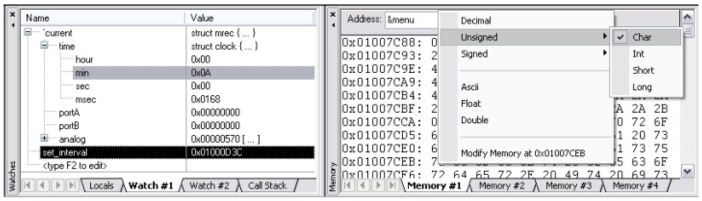
- [✓] 00: (E) 0x010007FC, 'main',
- [ ] 01: (A readwrite 0x00000562 len=1), 'current.time.sec',
- [✓] 02: (E) 0x010004D8, 'tc0', cmd: "printf(\"Got a Timer Tick Interrupt\")"
- [✓] 03: (E) 0x01000458, 'save_current_measurements', count=5,
- [✓] 06: (A write 0x00000580 len=1), 'interval.min==5&&mdisplay', cmd: "printf(\"Hit (

The Breakpoints Dialog allows you to define complex breakpoints on conditional expressions or different memory access types.

Memory and Watch Window

The Memory Window displays various memory areas. You may select the output format or change values via the context menu.

The Watch Window display and allows you to change variables values. The variables of the current functions are shown under the **Locals** tab whereas the **Call Stack** tab lists the program call tree.



Name	Value
`current	struct mrec { ... }
time	struct clock { ... }
hour	0x00
min	0x0A
sec	0x00
msec	0x0168
portA	0x00000000
portB	0x00000000
analog	0x00000570 [...]
set_interval	0x01000D3C
<type F2 to edit>	

Address: &menu

```
0x01007C88: 0
0x01007C93: 2
0x01007C9E: 4
0x01007CA9: 4
0x01007CB4: 4
0x01007CBF: 2                              A 2A 2B
0x01007CCA: 0                              0 72 6F
0x01007CD5: 6                              1 20 73
0x01007CE0: 6                              1 73 75
0x01007CEB: 7                              5 63 6F
0x01007CF6: 72 64 65 72 2E 20 49 74 20 69 73
```

The Watch and Memory Windows allow you to view variables and memory. Select display formats with context menu commands.

CPU and Peripheral Simulation

µVision3 has powerful simulation capabilities that give you serious benefits for rapid, reliable embedded software development.

- Simulation allows software testing on your desktop with no hardware environment.
- Early software debugging on a functional basis improves overall software reliability.
- Simulation allows breakpoints that are impossible with hardware debuggers.
- Simulation offers optimal input signals (hardware debuggers add extra noise).
- Single-stepping through signal processing algorithms is possible. External signals are stopped when the CPU halts.
- Failure scenarios that would destroy real hardware peripherals are easily checked.

High-Speed Device Simulation

The µVision3 Debugger accurately simulates a complete microcontroller including the instruction set and on-chip peripherals. Peripheral simulation is composed of:

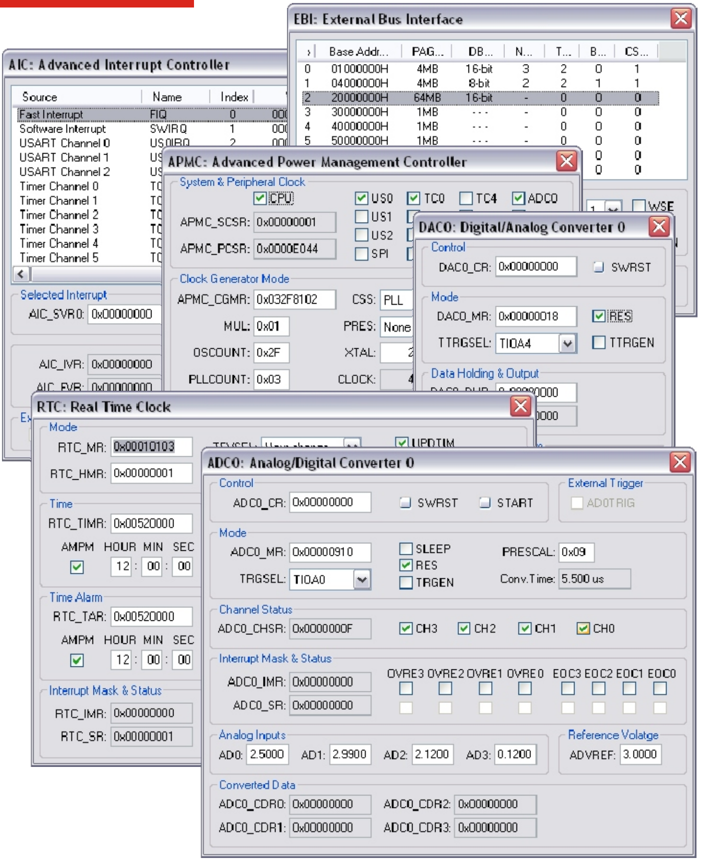
- Dialog boxes that allow you to view and rapidly configure peripherals.
- Virtual Registers (VTREGs) that allow you to control peripherals using breakpoints or the built-in debugging C script language.

You may use dialogs, virtual registers, and debug scripts to create high-fidelity simulations of your target hardware and environment.

The DLL-based Advanced Simulation Interface (AGSI) allows you to implement user-defined peripherals. You may define the behavior and dialogs necessary for a new peripheral. For instance, Dolphin Integration's SUCCESS™ uses AGSI for co-simulation of hardware and software mixed-signal peripherals in VHDL.

Communication Interfaces

µVision3 supports simulation of I/O interfaces like CAN, I^{2}C, and UARTs. Serial Windows provide terminal simulators for the on-chip UARTs. CAN and I^{2}C Windows show message traffic and bus configuration details.


AIC: Advanced Interrupt Controller
EBI: External Bus Interface
APMC: Advanced Power Management Controller
DAC0: Digital/Analog Converter 0
RTC: Real Time Clock
ADC0: Analog/Digital Converter 0

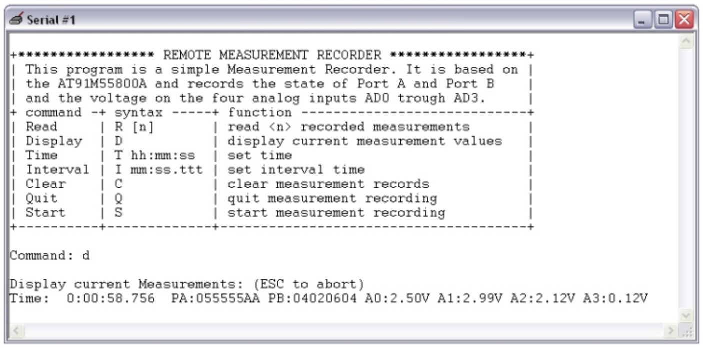
ADC0: Analog/Digital Converter 0
Control: ADC0_CR: 0x00000000 SWRST START
External Trigger: ADOTRIG
Mode: ADC0_MR: 0x00000910 SLEEP ✓ RES TRGEN PRESCAL: 0x09
TRGSEL: TIOA0 Conv.Time: 5.500 us
Channel Status: ADC0_CHSR: 0x0000000F ✓ CH3 ✓ CH2 ✓ CH1 ✓ CH0
Interrupt Mask & Status: ADC0_IMR: 0x00000000 ADC0_SR: 0x00000000 OVRE3 OVRE2 OVRE1 OVRE0 EOC3 EOC2 EOC1 EOC0
Analog Inputs: AD0: 2.5000 AD1: 2.9900 AD2: 2.1200 AD3: 0.1200
Reference Volatge: ADVREF: 3.0000
Converted Data: ADC0_CDR0: 0x00000000 ADC0_CDR2: 0x00000000 ADC0_CDR1: 0x00000000 ADC0_CDR3: 0x00000000

RTC: Real Time Clock
Mode: RTC_MR: 0x00010103 RTC_HMR: 0x00000001
Time: RTC_TIMR: 0x00520000 AMPM ✓ HOUR 12 MIN 00 SEC 00
Time Alarm: RTC_TAR: 0x00520000 AMPM ✓ HOUR 12 MIN 00 SEC 00
Interrupt Mask & Status: RTC_IMR: 0x00000000 RTC_SR: 0x00000001

Peripheral dialogs display detailed status information and aid in driver development.

VTREG	Description
PORTx	Represents the microcontroller I/O port pins.
ADx	Allows to supply a A/D input voltage as float value.
DAx	Shows the D/A output voltage as float value.
CLOCK	Represents the CPU clock frequency
SIN, SOUT	Input/Output registers of the Serial Interface (UART).

The peripheral VTREG's provide access to I/O peripherals from debug scripts, signal functions and the debugger command line.



```
+***************** REMOTE MEASUREMENT RECORDER *****************+
| This program is a simple Measurement Recorder. It is based on |
| the AT91M55800A and records the state of Port A and Port B    |
| and the voltage on the four analog inputs AD0 trough AD3.     |
+ command -+ syntax -----+ function ---------------------------+
| Read     | R [n]       | read <n> recorded measurements      |
| Display  | D           | display current measurement values  |
| Time     | T hh:mm:ss  | set time                            |
| Interval | I mm:ss.ttt | set interval time                   |
| Clear    | C           | clear measurement records           |
| Quit     | Q           | quit measurement recording          |
| Start    | S           | start measurement recording         |
+----------+-------------+-------------------------------------+

Command: d

Display current Measurements: (ESC to abort)
Time:  0:00:58.756  PA:055555AA PB:04020604 A0:2.50V A1:2.99V A2:2.12V A3:0.12V
```

Serial windows display UART output from your applications. You may type characters to receive into the selected serial window.

Debug Scripts and Signal Functions

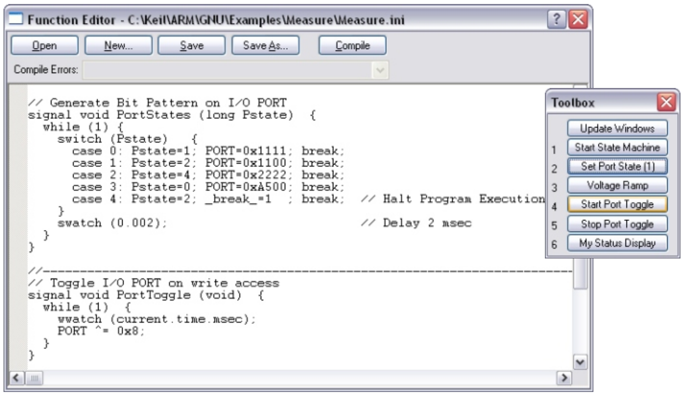
The µVision3 Debugger incorporates a C script language you can use to create:

- Signal functions that simulate analog and digital input to the microcontroller. Signal functions run in the background while µVision3 simulates your target program.
- User functions that extend the command scope or combine repetitive operations.

The µVision3 Debugger includes many built-in functions (like printf, memset, rand, sin, log, sqrt, and so on).

Debug functions may be invoked from the Command Window (by typing the function name), from triggered Breakpoints, or from Toolbox Buttons.

Various debugger output may be directed to log files for long-term storage, analysis, and regression testing.



```
// Generate Bit Pattern on I/O PORT
signal void PortStates (long Pstate)  {
  while (1) {
    switch (Pstate)   {
      case 0: Pstate=1; PORT=0x1111; break;
      case 1: Pstate=2; PORT=0x1100; break;
      case 2: Pstate=4; PORT=0x2222; break;
      case 3: Pstate=0; PORT=0xA500; break;
      case 4: Pstate=2; _break_=1  ; break;  // Halt Program Execution
    }
    swatch (0.002);                          // Delay 2 msec
  }
}

//-------------------------------------------------------------------------
// Toggle I/O PORT on write access
signal void PortToggle (void)  {
  while (1)  {
    wwatch (current.time.msec);
    PORT ^= 0x8;
  }
}
```

The Function Editor (Debug menu) allows you to create Signal and User Functions. You may define Toolbox buttons for fast command invocation.

CAN Communication Example

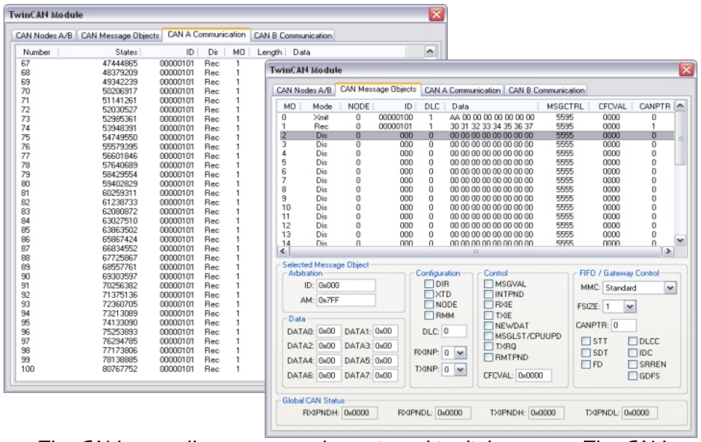
You may use the following VTREGs to access the CAN controller in user and signal functions.

VTREG	Description
CANxID	CAN Message Identifier
CANxL	Length of the CAN Message
CANxBx	Data bytes of the CAN Message
CANxIN	Status of incoming Message
CANxOUT	Status of outgoing Message
CANTIMING	Adjust CAN bus timing

Using these VTREGs you may create a user function that sends a CAN message with 3 data bytes to the application. For example:

```
FUNC void SendMessage (void)  {
  CANAID = 0x4422;   // CAN Identifier
  CANAL  = 3;        // Message Length
  CANAB0 = 0x11;     // 1st data byte
  CANAB1 = 0x22;     // 2nd data byte
  CANAB2 = 0x33;     // 3nd data byte
  CANAIN = 2;        // Send Message
}
```

This function may be called from a signal function to repeatedly send this CAN message. You may create user functions that generate automated replies to CAN messages. This is shown in application notes provided at www.keil.com/appnotes.



Number	States	ID	Dir	MO	Length	Data
67	47444865	00000101	Rec	1		
68	48379209	00000101	Rec	1		
69	49342239	00000101	Rec	1		
70	50206917	00000101	Rec	1		
71	51141261	00000101	Rec	1		
72	52030527	00000101	Rec	1		
73	52985361	00000101	Rec	1		
74	53948391	00000101	Rec	1		
75	54749550	00000101	Rec	1		
76	55579395	00000101	Rec	1		
77	56601846	00000101	Rec	1		
78	57640689	00000101	Rec	1		
79	58429554	00000101	Rec	1		
80	59402829	00000101	Rec	1		
81	60259311	00000101	Rec	1		
82	61238733	00000101	Rec	1		
83	62080872	00000101	Rec	1		
84	63027510	00000101	Rec	1		
85	63863502	00000101	Rec	1		
86	65867424	00000101	Rec	1		
87	66834552	00000101	Rec	1		
88	67725867	00000101	Rec	1		
89	68557761	00000101	Rec	1		
90	69303597	00000101	Rec	1		
91	70256382	00000101	Rec	1		
92	71375136	00000101	Rec	1		
93	72360705	00000101	Rec	1		
94	73213089	00000101	Rec	1		
95	74133090	00000101	Rec	1		
96	75253893	00000101	Rec	1		
97	76294785	00000101	Rec	1		
98	77173806	00000101	Rec	1		
99	78138885	00000101	Rec	1		
100	80767752	00000101	Rec	1		

MO	Mode	NODE	ID	DLC	Data	MSGCTRL	CFCVAL	CANPTR
0	Xmit	0	00000100	1	AA 00 00 00 00 00 00 00	5595	0000	0
1	Rec	0	00000101	1	30 31 32 33 34 35 36 37	5595	0000	1
2	Dis	0	000	0	00 00 00 00 00 00 00 00	5555	0000	0
3	Dis	0	000	0	00 00 00 00 00 00 00 00	5555	0000	0
4	Dis	0	000	0	00 00 00 00 00 00 00 00	5555	0000	0
5	Dis	0	000	0	00 00 00 00 00 00 00 00	5555	0000	0
6	Dis	0	000	0	00 00 00 00 00 00 00 00	5555	0000	0
7	Dis	0	000	0	00 00 00 00 00 00 00 00	5555	0000	0
8	Dis	0	000	0	00 00 00 00 00 00 00 00	5555	0000	0
9	Dis	0	000	0	00 00 00 00 00 00 00 00	5555	0000	0
10	Dis	0	000	0	00 00 00 00 00 00 00 00	5555	0000	0
11	Dis	0	000	0	00 00 00 00 00 00 00 00	5555	0000	0
12	Dis	0	000	0	00 00 00 00 00 00 00 00	5555	0000	0
13	Dis	0	000	0	00 00 00 00 00 00 00 00	5555	0000	0
14	Dis	0	000	0	00 00 00 00 00 00 00 00	5555	0000	0

Selected Message Object

Arbitration — ID: 0x000 AM: 0x7FF

Data — DATA0: 0x00 DATA1: 0x00 DATA2: 0x00 DATA3: 0x00 DATA4: 0x00 DATA5: 0x00 DATA6: 0x00 DATA7: 0x00

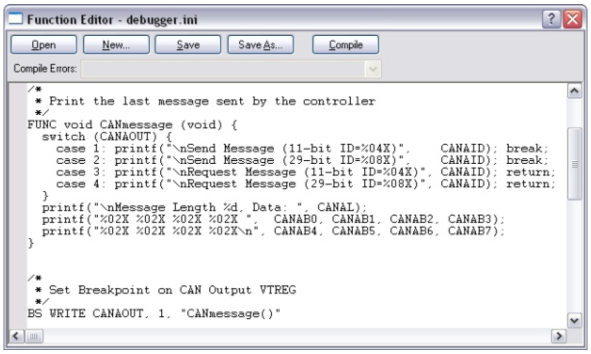
Configuration — ☐ DIR ☐ XTD ☐ NODE ☐ RMM DLC: 0 RXINP: 0 TXINP: 0

Control — ☐ MSGVAL ☐ INTPND ☐ RXIE ☐ TXIE ☐ NEWDAT ☐ MSGLST/CPUUPD ☐ TXRQ ☐ RMTPND CFCVAL: 0x0000

FIFO / Gateway Control — MMC: Standard FSIZE: 1 CANPTR: 0 ☐ STT ☐ SDT ☐ FD ☐ DLCC ☐ IDC ☐ SRREN ☐ GDFS

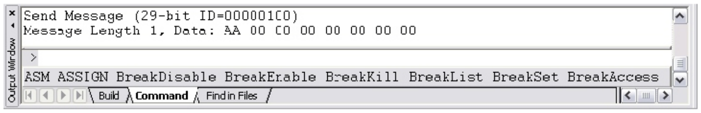
Global CAN Status — RXIPNDH: 0x0000 RXIPNDL: 0x0000 TXIPNDH: 0x0000 TXIPNDL: 0x0000

The CAN controller status can be reviewed in dialog pages. The CAN Communication Page is an analyser which displays all CAN message.



```
/*
 * Print the last message sent by the controller
 */
FUNC void CANmessage (void) {
  switch (CANAOUT) {
    case 1: printf("\nSend Message (11-bit ID=%04X)",    CANAID); break;
    case 2: printf("\nSend Message (29-bit ID=%08X)",    CANAID); break;
    case 3: printf("\nRequest Message (11-bit ID=%04X)", CANAID); return;
    case 4: printf("\nRequest Message (29-bit ID=%08X)", CANAID); return;
  }
  printf("\nMessage Length %d, Data: ", CANAL);
  printf("%02X %02X %02X %02X ",  CANAB0, CANAB1, CANAB2, CANAB3);
  printf("%02X %02X %02X %02X\n", CANAB4, CANAB5, CANAB6, CANAB7);
}


/*
 * Set Breakpoint on CAN Output VTREG
 */
BS WRITE CANAOUT, 1, "CANmessage()"
```

You may use debug scripts to analyse CAN messages. A breakpoint on write access to the VTREG CANxOUT may be used to trigger a user function.



```
Send Message (29-bit ID=000001C0)
Message Length 1, Data: AA 00 C0 00 00 00 00 00

>
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess
```

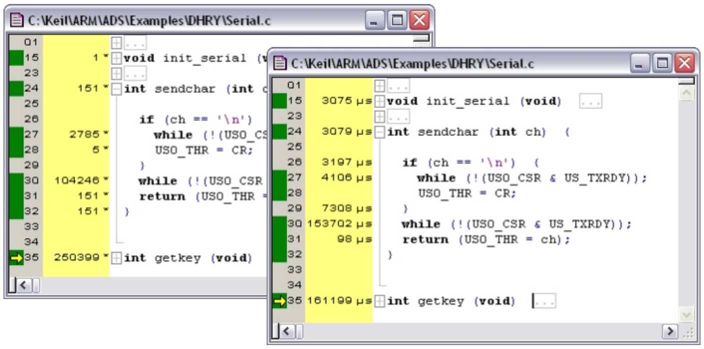
The Output Window displays printf status information of debug scripts.

Advanced Simulation Features

Execution Profiling

µVision3 for ARM provides an execution profiler that records execution statistics for each CPU instruction. You may review the execution count or execution time in editor and disassembly windows. Editor outlining provides an overview of the source code.

You may use the information gathered by the execution profiler to determine where your program spends most of its time. Then, you can concentrate your efforts on making that section run faster.


C:\Keil\ARM\ADS\Examples\DHRY\Serial.c

```
                ...
      1 *   void init_serial (v
                ...
    151 *   int sendchar (int c

              if (ch == '\n')
   2785 *       while (!(USO_CS
      5 *       USO_THR = CR;
              }
 104246 *     while (!(USO_CSR
    151 *     return (USO_THR =
    151 *   }


 250399 *   int getkey (void)
```

C:\Keil\ARM\ADS\Examples\DHRY\Serial.c

```
                ...
   3075 µs  void init_serial (void)  ...
                ...
   3079 µs  int sendchar (int ch)  {

   3197 µs    if (ch == '\n')  {
   4106 µs      while (!(USO_CSR & US_TXRDY));
                USO_THR = CR;
   7308 µs    }
 153702 µs    while (!(USO_CSR & US_TXRDY));
     98 µs    return (USO_THR = ch);
            }


 161199 µs  int getkey (void)  ...
```

Execution Profiling shows how many times each line of code has executed and the total execution time.

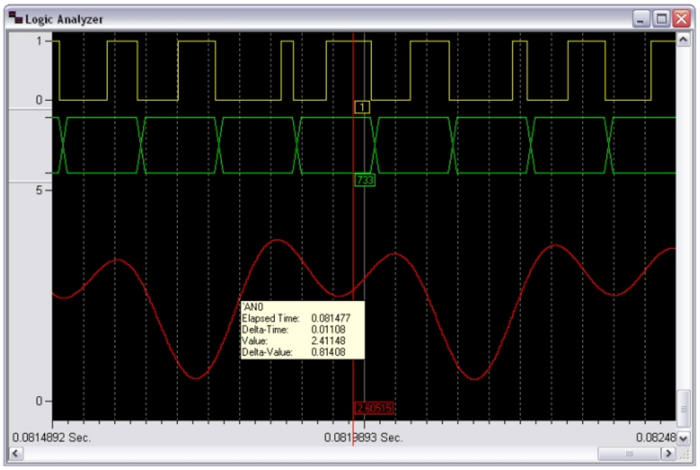
Logic Analyzer

The Logic Analyzer records the values of variables and peripheral I/O signals over time and displays them in the Logic Analyzer Window. The cursor line allows you to set a reference time and review delta values.

The Logic Analyzer provides three different display formats:

- **Bit:** Shows output as a logic level (0 or 1).
- **State:** Shows state changes of a value.
- **Analog:** Shows a graphical display.

You can easily generate complex digital or analog input signal patterns using signal functions you define using the debugger's C script language. This allows detailed testing of signal processing algorithms at simulation level without using extensive hardware equipment.



```
AN0
Elapsed Time:   0.081477
Delta-Time:     0.01108
Value:          2.41148
Delta-Value:    0.81408
```

The Logic Analyzer records signals you select. The above recording shows a PWM output, A/D input, and analog signal generated by a signal function.

```
FUNC float sine_wave (float f)  {
  return __sin (f * (STATES / CLOCK) * 2 * PI);
}

AN0 = sine_wave (2200) + sine_wave (4500);
```

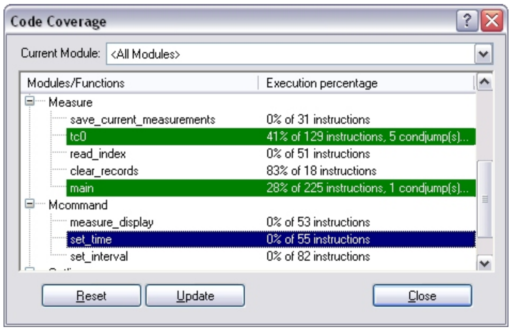
Signal Functions may generate analog and digital signal patterns using peripheral input pins (for example, AN0 is Analog Input 0).

Code Coverage

The µVision3 Debugger provides code coverage statistics for verification of safety-critical applications where certification testing and validation is required. Output includes:

- Non-executed Instructions,
- Skipped Instructions,
- Partially Executed Instructions,
- and Fully Executed Instructions.

Code Coverage information may be output to a file for detailed analysis. You may even include a complete CPU instruction listing in this report file.



Current Module: <All Modules>

Modules/Functions	Execution percentage
Measure	
save_current_measurements	0% of 31 instructions
tc0	41% of 129 instructions, 5 condjump(s)...
read_index	0% of 51 instructions
clear_records	83% of 18 instructions
main	28% of 225 instructions, 1 condjump(s)...
Mcommand	
measure_display	0% of 53 instructions
set_time	0% of 55 instructions
set_interval	0% of 82 instructions

Code Coverage provides detailed information about tested functions.

Target Debugging

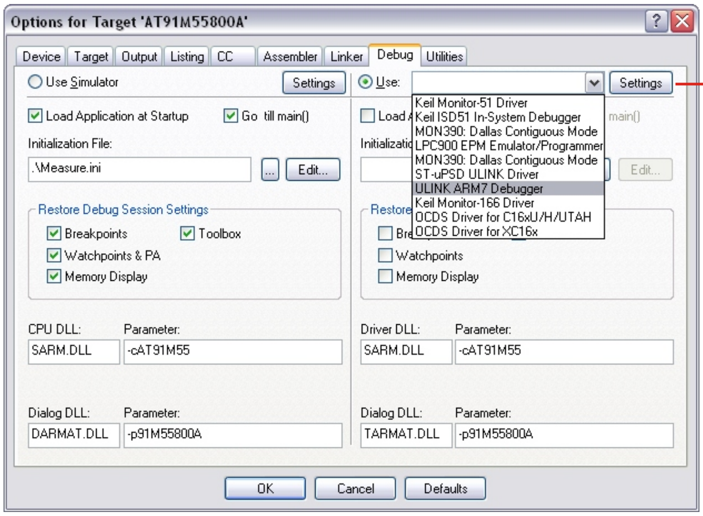
The µVision3 Debugger interfaces to your target hardware via the following Advanced Generic Debugger Interface (AGDI) drivers:

- 8051, 251, and C16x **Monitor**: May be integrated with user hardware. Included with many third-party evaluation boards.
- **ISD51** and **FlashMon**: Designed for 8051 single-chip devices.
- **EPM900**: Philips LPC9xx Emulator.
- **SmartMX DBox**: Philips SmartCards.

Third-party vendors offer the following AGDI target debugger drivers:

- ChipCon **CC1010** RF Transceiver.
- JTAG interface for **Cygnal 51Fxxx** series.
- Cypress **USB development kits**.
- Infineon **SLE66** SmartCard ROM Monitor.
- Quickcore **FPGA based Pro8051** device.
- SST **SoftICE** for FlashFlex51devices.
- JTAG interface for **Triscend E5** series.

In addition, many emulator vendors offer µVision3 AGDI drivers for their emulators.


Options for Target 'AT91M55800A'

Device | Target | Output | Listing | CC | Assembler | Linker | Debug | Utilities

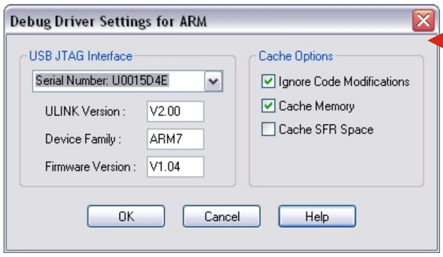
Left side: ( ) Use Simulator [Settings]
✓ Load Application at Startup ✓ Go till main()
Initialization File: .\Measure.ini [...] [Edit...]
Restore Debug Session Settings: ✓ Breakpoints ✓ Toolbox ✓ Watchpoints & PA ✓ Memory Display
CPU DLL: SARM.DLL Parameter: -cAT91M55
Dialog DLL: DARMAT.DLL Parameter: -p91M55800A

Right side: (●) Use: [Settings]
Keil Monitor-51 Driver
Keil ISD51 In-System Debugger
MON390: Dallas Contiguous Mode
LPC900 EPM Emulator/Programmer
MON390: Dallas Contiguous Mode
ST-uPSD ULINK Driver
ULINK ARM7 Debugger
Keil Monitor-166 Driver
OCDS Driver for C16xU/H/UTAH
OCDS Driver for XC16x

Load A... main()
Initializati... [Edit...]
Restore...: Bre... ☐ Watchpoints ☐ Memory Display
Driver DLL: SARM.DLL Parameter: -cAT91M55
Dialog DLL: TARMAT.DLL Parameter: -p91M55800A

[OK] [Cancel] [Defaults]

µVision3 gives you the choice between Simulator and Target Debugger. Simply select the AGDI driver (in the Options - Debug dialog page) that matches your target hardware configuration.


Debug Driver Settings for ARM

USB JTAG Interface
Serial Number: U0015D4E
ULINK Version: V2.00
Device Family: ARM7
Firmware Version: V1.04

Cache Options
✓ Ignore Code Modifications
✓ Cache Memory
☐ Cache SFR Space

[OK] [Cancel] [Help]

Click on Settings to open the Driver configuration dialog to specify hardware-related parameters.

ULINK USB-JTAG Adapter

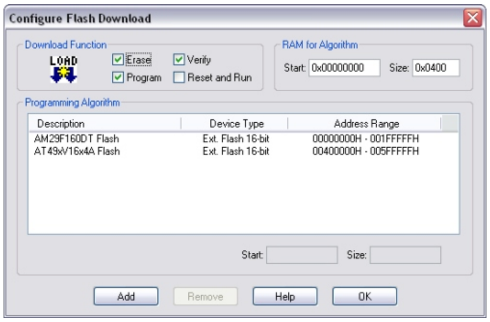
The ULINK USB-JTAG Adapter connects the µVision3 Debugger to the JTAG port pins of the embedded system. Currently ULINK supports ST µPSD 8051-based, Infineon xC16x, and several ARM-powered microcontrollers. ULINK allows you to:

- Download target programs.
- Examine memory and registers.
- Single-step through programs.
- Insert multiple breakpoints.
- Run programs in real-time.
- Program on-chip FLASH.

ULINK connects the PC USB Port to the JTAG or OCDS interface of the microcontroller system and allows Flash programming and debugging.

Flash Device Programming

ULINK supports Flash Device Programming via the µVision3 **Flash - Download** menu. In the **Options - Utilities - ULINK - Settings** dialog you may add programming algorithms for the various Flash devices of your system. ULINK even allows you to customize your own programming algorithms. Complete setup details are saved in the µVision3 project.


Configure Flash Download

Download Function: ✓ Erase ✓ Verify ✓ Program ☐ Reset and Run

RAM for Algorithm: Start: 0x00000000 Size: 0x0400

Programming Algorithm

Description	Device Type	Address Range
AM29F160DT Flash	Ext. Flash 16-bit	00000000H - 001FFFFFH
AT49xV16x4A Flash	Ext. Flash 16-bit	00400000H - 005FFFFFH

Start: Size:

[Add] [Remove] [Help] [OK]

Flash programming may be configured for user-provided devices. Several algorithms are available for standard Flash devices and on-chip Flash ROM.

Evaluation Boards

Evaluation Boards and Starter Kits are a great way to begin your microcontroller projects. Keil Software offers evaluation versions of the development tools and example programs which are ready-to-run. You can easily test device performance and the flexibility of the development tools at the same time.

Evaluation boards that interface to µVision3 are available from many chip vendors and thrid-party manufacturers (see the Device Database™ at www.keil.com/dd). In addition, Keil Software offers Development Hardware for the following platforms.

8051 & 251 Boards

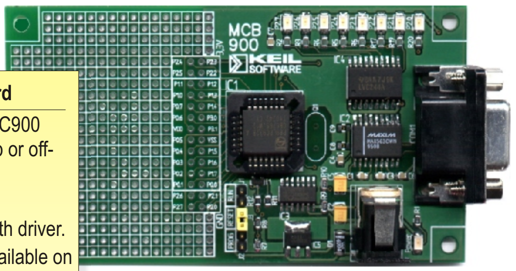
- **MCBx51:** Supports numerous 44-pin PLCC 8051 and 251 devices including many high-speed versions.
- **MCB517AC:** Based on the full-featured Infineon C517A. Includes 81C90 CAN controller and code banking support.
- **MCB900** and **EPM900:** Support the flexible Philips LPC900 low-power devices with a high-speed, two-cycle 8051 core.

XC16x/C16x/ST10 Boards

- **MCB167NET:** Supports all variants of the Infineon C167 and ST10-F168 devices (up to 50MHz CPU speed). It includes two CAN interfaces and an Ethernet controller for TCP/IP applications.
- **MCB-XC167:** Supports Infineon XC16x devices (up to 40MHz) including Flash ROM and MAC unit (for high-speed DSP processing). Supports devices with OCDS (On-Chip Debugging System).

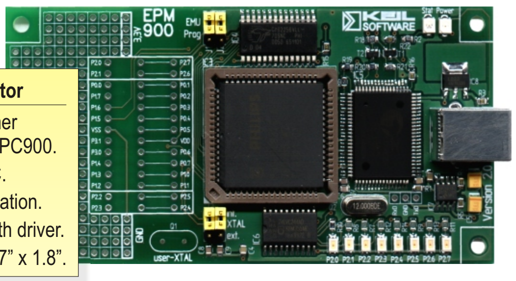
ARM Boards

- **MCB2100:** Supports the ARM7TDMI-based Philips LPC2100 devices and the JTAG debug interface. Members of this device family offer high-speed, on-chip Flash ROM with ISP/IAP programming and include a wide range of on-chip peripherals like A/D converters and communication interfaces like CAN and I^{2}C.



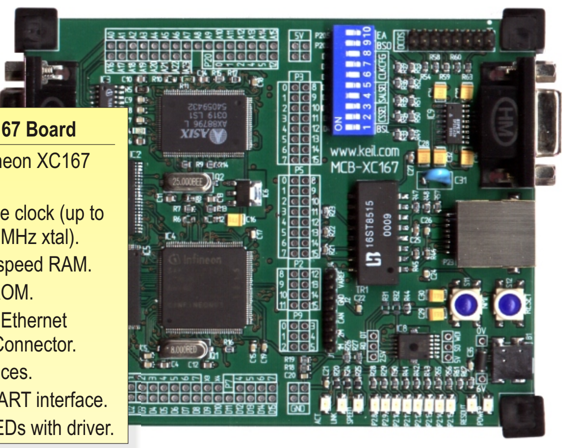
MCB900 Board

- Supports Philips LPC900 devices with on-chip or off-chip clock.
- UART interface.
- Port status LEDs with driver.
- All CPU pins are available on 2.1" x 1.8" prototyping area.



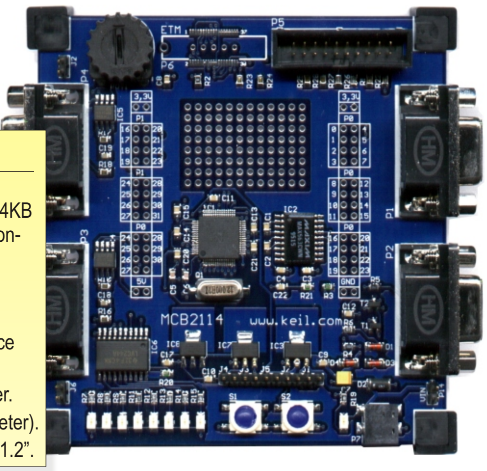
EPM900 Emulator

- Emulator/Programmer Module for Philips LPC900.
- USB interface to PC.
- Flexible clock generation.
- Port status LEDs with driver.
- Prototyping area: 0.7" x 1.8".



MCB-XC167 Board

- Supports Infineon XC167 (XC161).
- PLL for flexible clock (up to 40MHz with 8MHz xtal).
- 512 KB high-speed RAM.
- 2 MB Flash ROM.
- 10/100 Mbps Ethernet Controller & Connector.
- 2 CAN Interfaces.
- OCDS and UART interface.
- Port status LEDs with driver.



MCB2100 Board

- Supports Philips ARM LPC21xx series (up to 64KB RAM and 256KB Flash on-chip).
- 2 CAN interfaces.
- 2 UART interfaces.
- JTAG debugging interface (ETM optional).
- Port status LEDs & driver.
- Analog Input (potentiometer).
- Prototyping area: 0.7" x 1.2".

µVision3: The Intelligent Embedded IDE

Product Highlights

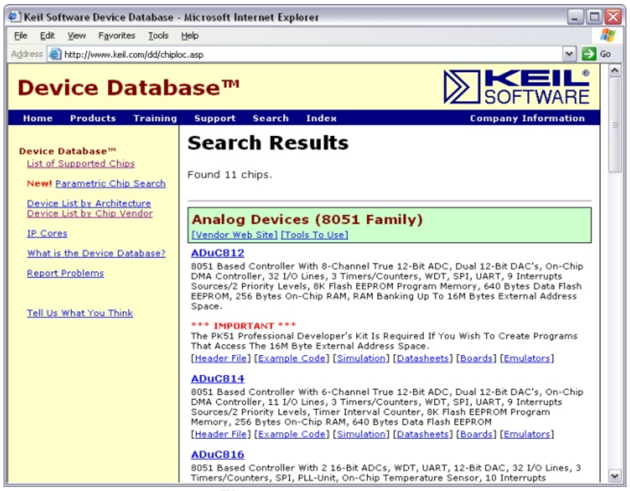
µVision3 reduces your learning curve, shortens the development/testing cycle, and helps you build reliable, stable applications. To accomplish this, µVision3 offers:

- The Device Database™ (available on Internet and in µVision3) to pre-configure the tools. This minimizes mistakes in project setup and gets you started quickly.
- A full-featured Editor (integrated into the Debugger) that allows source code changes at any time.
- The Debugger which accurately simulates target systems including instructions and on-chip peripherals. Test application code before hardware is available. Optimize your hardware design prior to production.
- Chip-specific dialogs which allow you to review status information and rapidly configure peripherals.
- Identical Simulator and Target Debugger interfaces.
- Integration with third-party tools like SVCS, CASE, and Flash/Device Programming utilities. All configuration details are saved in a single, consistent project.
- The µVision3 Debugger which interfaces to the Keil ULINK USB-JTAG Adapter and other third-party diagnostic tools.

New Device Support

Frequent updates to the Device Database™ ensure that µVision3 always supports the latest 8051, 251, C16x. XC16x, ST10, and ARM powered microcontrollers.

Find out more about the Device Database™ and the chips supported! Refer to **www.keil.com/dd**.


Keil Software Device Database - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://www.keil.com/dd/chiploc.asp
Go
Device Database™
KEIL SOFTWARE
Home Products Training Support Search Index Company Information
Device Database™
List of Supported Chips
New! Parametric Chip Search
Device List by Architecture
Device List by Chip Vendor
IP Cores
What is the Device Database?
Report Problems
Tell Us What You Think

Search Results

Found 11 chips.

Analog Devices (8051 Family)
[Vendor Web Site] [Tools To Use]

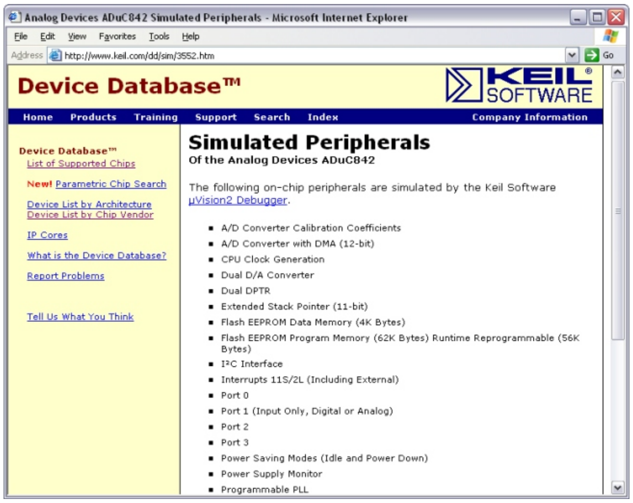
ADuC812
8051 Based Controller With 8-Channel True 12-Bit ADC, Dual 12-Bit DAC's, On-Chip DMA Controller, 32 I/O Lines, 3 Timers/Counters, WDT, SPI, UART, 9 Interrupts Sources/2 Priority Levels, 8K Flash EEPROM Program Memory, 640 Bytes Data Flash EEPROM, 256 Bytes On-Chip RAM, RAM Banking Up To 16M Bytes External Address Space.

*** IMPORTANT ***
The PK51 Professional Developer's Kit Is Required If You Wish To Create Programs That Access The 16M Byte External Address Space.
[Header File] [Example Code] [Simulation] [Datasheets] [Boards] [Emulators]

ADuC814
8051 Based Controller With 6-Channel True 12-Bit ADC, Dual 12-Bit DAC's, On-Chip DMA Controller, 11 I/O Lines, 3 Timers/Counters, WDT, SPI, UART, 9 Interrupts Sources/2 Priority Levels, Timer Interval Counter, 8K Flash EEPROM Program Memory, 256 Bytes On-Chip RAM, 640 Bytes Data Flash EEPROM
[Header File] [Example Code] [Simulation] [Datasheets] [Boards] [Emulators]

ADuC816
8051 Based Controller With 2 16-Bit ADCs, WDT, UART, 12-Bit DAC, 32 I/O Lines, 3 Timers/Counters, SPI, PLL-Unit, On-Chip Temperature Sensor, 10 Interrupts

*The Device Database™ on **www.keil.com/dd** provides detailed information about the chips supported by Keil development tools.*


Analog Devices ADuC842 Simulated Peripherals - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://www.keil.com/dd/sim/3552.htm
Go
Device Database™
KEIL SOFTWARE
Home Products Training Support Search Index Company Information
Device Database™
List of Supported Chips
New! Parametric Chip Search
Device List by Architecture
Device List by Chip Vendor
IP Cores
What is the Device Database?
Report Problems
Tell Us What You Think

Simulated Peripherals
Of the Analog Devices ADuC842

The following on-chip peripherals are simulated by the Keil Software µVision2 Debugger.

- A/D Converter Calibration Coefficients
- A/D Converter with DMA (12-bit)
- CPU Clock Generation
- Dual D/A Converter
- Dual DPTR
- Extended Stack Pointer (11-bit)
- Flash EEPROM Data Memory (4K Bytes)
- Flash EEPROM Program Memory (62K Bytes) Runtime Reprogrammable (56K Bytes)
- I²C Interface
- Interrupts 11S/2L (Including External)
- Port 0
- Port 1 (Input Only, Digital or Analog)
- Port 2
- Port 3
- Power Saving Modes (Idle and Power Down)
- Power Supply Monitor
- Programmable PLL

Complete peripheral simulation is available for most devices.

Keil Elektronik GmbH and **Keil Software, Inc.**, have representatives in the following countries:

Australia, Austria, Belgium, Brazil, Canada, China, Czech, Denmark, Finland, France, Germany, India, Ireland, Israel, Italy, Japan, Korea, Malaysia, Mexico, Netherlands, New Zealand, Norway, Poland, Portugal, Russia, Romania, Singapore, Slovenia, Slovakia, Spain, South Africa, Sweden, Switzerland, Taiwan, Turkey, Thailand, United Kingdom, and United States.

Europe:

Keil Elektronik GmbH
Bretonischer Ring 15
85630 Grasbrunn
Germany

Phone	**++49 89 / 45 60 40 - 0**
Support	**++49 89 / 45 60 40 - 24**
FAX	**++49 89 / 46 81 62**
Email	**sales.intl@keil.com**
	support.intl@keil.com

United States:

Keil Software, Inc.
1501 10th Street, Suite 110
Plano, Texas 75074
USA

Phone	**800-348-8051**
	972-312-1107
FAX	**972-312-1159**
Email	**sales.us@keil.com**
	support.us@keil.com

KEIL® SOFTWARE

www.keil.com

Copyright © 1998 - 2004 Keil Elektronik GmbH & Keil Software, Inc. All rights reserved.
Information in this data sheet is subject to change without notice and does not represent a commitment on the part of Keil Software, Inc.